Expected Length of Longest Common Subsequences

Vladimír Dančík

A thesis submitted for the
Degree of Doctor of Philosophy

Department of Computer Science
University of Warwick

September 1994

Contents

List of Figures

Acknowledgements

Without the help of my supervisor Mike Paterson this work would never have grown to the present shape. His comments were extremely helpful and his suggestions have improved this thesis in many directions. I have learnt a lot from his approach to tackling problems. He deserves special thanks for his patience when correcting definite and indefinite articles in all my texts.

During my studies at Warwick I had very useful discussions on various topics with many colleagues from and outside the Computer Science Department. Many people made my stay at Warwick very pleasant and I have found a lot of friends here. My thanks goes to all of them.

These three years were very hard for my wife Jurika and I am very obliged to her. Her love was the best support for me.

I was supported by an East European Scholarship from the University of Warwick, an ORS Award from the CVCP, and the ESPRIT II BRA Programme of the EC under contract 7141 (ALCOM II).

Declaration

This thesis is submitted to the University of Warwick in fulfilment of the requirements of the degree of Doctor of Philosophy. No part of the thesis has been submitted in support of an application for another degree or qualification of this or any other institution of learning. Some parts of the thesis have appeared in the following papers in which my own work was that of full pro-rata contribution.

1. Vlado Dančík and Mike Paterson, Upper bounds for the expected length of a longest common subsequence of two binary sequences. In *Proceedings of* 11^{th} *Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 775, Springer-Verlag, 1994.

2. Mike Paterson and Vlado Dančík, Longest common subsequences. In *Proceedings of 19th International Symposium on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, Springer-Verlag, 1994.

V. Dančík September 27, 1994

Summary

A longest common subsequence of two sequences is a sequence that is a subsequence of both the given sequences and has largest possible length. It is known that the expected length of a longest common subsequence is proportional to the length of the given sequences. The proportion, denoted by γ_k, is dependent on the alphabet size k and the exact value of this proportion is not known even for a binary alphabet.

To obtain lower bounds for the constants γ_k, finite state machines computing a common subsequence of the inputs are built. Analysing the behaviour of the machines for random inputs we get lower bounds for the constants γ_k. The analysis of the machines is based on the theory of Markov chains. An algorithm for automated production of lower bounds is described.

To obtain upper bounds for the constants γ_k, collations – pairs of sequences with a marked common subsequence – are defined. Upper bounds for the number of collations of 'small size' can be easily transformed to upper bounds for the constants γ_k. Combinatorial analysis is used to bound the number of collations.

The methods used for producing bounds on the expected length of a common subsequence of two sequences are also used for other problems, namely a longest common subsequence of several sequences, a shortest common supersequence and a maximal adaptability.

To Jurika,
Tonka, and Radka

Chapter 1

Introduction

Many different problems, when described in the terms of sequences, become independent of disturbing influences and are easier to handle. Very often sequences can capture many, if not all, interesting features.

One such abstract problem is the sequence alignment problem. Given two input sequences of symbols we have to find an output sequence that occurs in the input sequences. We can demand that the symbols involved in the output must form a continuous block within the input sequences. The continuous alignment problems lead to common substring problems. The alignment problems that do not require the continuity condition correspond to common subsequence problems.

The alignment problem arises in various situations in different fields. As typical within computer science we can mention the string edit problem. Imagine we have a terminal connected through a very slow line. Then when the screen of the terminal has to be upgraded, it might be much more efficient to send editing instructions instead of the full screen of information. Thus, given two sequences, we want to find a shortest sequence of editing steps that transforms one sequence into the other. The length of such a sequence

is called the edit distance of the two sequences.

In the pattern matching problems we have to find one or all occurrences of a pattern in the image. There are cases when we would be satisfied with occurrences of similar patterns. For example, if we are searching in a database of names and we do not remember the name properly, or when recognizing a handwritten text or fingerprints.

Molecular biology is an important area where alignment problems appear quite often. Both proteins and nucleic acids can be treated as sequences of symbols. In molecular biology it is common to search for homologies – the parts of macromolecules that have only minor differences.

Chromatography is other field creating alignment problems. The presence of certain elements in burning gas can change the colour spectrum of the light produced. The results of such experiments can be described by chromatograms and comparing chromatograms can be seen as an alignment problem.

In this thesis we shall focus on the longest common subsequence problem. The structure of the thesis is as follows. Basic notation is given and main terms are defined in the first section of Chapter 2. Then we shall describe the longest common subsequence problem and will give a small survey of algorithms for solving this problem. In the last section of the chapter the expected length of a longest common subsequence is defined and its basic properties are shown.

We know that the expected length of a longest common subsequence is linear with respect to the length of the sequences, however we are not able to determine their exact ratio. Bounds on the expected length of a longest common subsequence are the topics of the next two chapters.

In Chapter 3 will be given lower bounds based on analysis of Markov chains. An algorithm for producing better and better lower bounds will be described. Special treatment is required for lower bounds over general alphabets of size k.

In Chapter 4 collations – pairs of sequences with marked matches – are described. Upper bounds for the number of collations based on combinatorial analysis are given. From these, upper bounds for the expected length of a longest common subsequence can be obtained.

Some problems related to the longest common subsequence problem will be described in Chapter 5. We shall apply the methods from the previous chapters when investigating the maximal adaptability, the shortest supersequence, and the longest subsequence problems for more sequences. The chapter is closed with a small survey of the longest common substring problem.

Chapter 2

Notation and preliminaries

2.1 Notation and basic definitions

Let $\Sigma = \{0, 1, \ldots, k-1\}$ be a fixed alphabet of size k. Let Σ^n be the set of all sequences of length n over the alphabet Σ. We can define Σ^n recursively by:

$$\begin{aligned} \Sigma^0 &= \{\lambda\}, \\ \Sigma^{n+1} &= \Sigma \times \Sigma^n, \quad \text{for } n \geq 0, \end{aligned}$$

here λ is the *empty sequence*, i.e. a sequence consisting of no symbols. Sequences are also commonly called strings. If sequence u is from Σ^n, we shall say that the length of sequence u is n and write $|u| = n$. The number of different symbols that occur in the sequence u will be denoted by $\|u\|$. The set of all sequences is $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$ and the set of all nonempty sequences is $\Sigma^+ = \bigcup_{n=1}^{\infty} \Sigma^n$. *Concatenation* is a basic operation over sequences. If $u = (u_1, \ldots, u_m) \in \Sigma^m$ and $v = (v_1, \ldots, v_n) \in \Sigma^n$, then $\mathrm{cat}(u, v) = (u_1, \ldots, u_m, v_1, \ldots, v_n) \in \Sigma^{m+n}$ is the concatenation of sequences u and v. To simplify the notation we shall

write $u = u_1 \cdots u_m$ instead of $u = (u_1, \ldots, u_m)$ and $u.v$ or uv instead of $\mathrm{cat}(u, v)$.

Sequence u is a subsequence of sequence v, if u can be obtained by deleting some symbols from v, the notion of subsequence is given formally in Definition 2.1

Definition 2.1 Sequence $u = u_1 \cdots u_m$ is called a *subsequence* of $v = v_1 \cdots v_n$ ($u \sqsubseteq v$), if there are m indices $i_1 < \cdots < i_m$ such that $u_1 = v_{i_1}, \ldots, u_m = v_{i_m}$.

Note that λ is a subsequence of every sequence and that every sequence is a subsequence of itself. If u is a subsequence of v, we shall also say that v is a *supersequence* of u. We shall write $u \not\sqsubseteq v$ if u is not a subsequence of v.

Similar, but not the same, is the notion of substring. Sequence u is a *substring* of v, if there are sequences x and y such that $xuy = v$. If $v = v_1 \cdots v_n$, we shall sometimes denote the substring of v beginning at the i-th position and ending at the j-th position by $v(i \,..\, j)$. We shall say that u is a *prefix* of v ($u \trianglelefteq v$), if there is an index j such that $u = v(1 \,..\, j)$. Sequence u is a *proper* prefix of v ($u \triangleleft v$) if $u \trianglelefteq v$ and $u \neq v$.

The set of all natural numbers $\{0, 1, \ldots\}$ will be denoted by $\mathbf{N}$. The set of all real numbers will be denoted by $\mathbb{R}$. While describing sequences we use $\in$ to denote the membership of an element in a set. We also use $\cup$ to denote the union of sets and $\times$ to denote the product of sets. Other symbols used to denote operations and relations on sets are $\cap$ for intersection, $\setminus$ for difference, $\subseteq$ for subset, and $\subset$ for proper subset. The set of all elements x having property P is denoted by $\{x : P(x)\}$. Instead of $\{y :$ there is x such that $y = f(x)$ and $P(x)\}$ we shall write $\{f(x) : P(x)\}$.

For a (linearly) ordered set M we denote the maximal element of M by $\max M$. Similarly $\min M$ is the minimal element of M and $\inf M$ is an infimum of M. Sometimes we use $\max_{P(x)} f(x)$ instead of $\max\{f(x) : P(x)\}$, especially when $P(x)$ is simple. Similarly for min and inf.

We use & to denote the conjunction of logical propositions. For disjunctions we shall use the symbol $\vee$, for implications $\Longrightarrow$, and for equivalences $\Longleftrightarrow$. The negation of a proposition is denoted by $\neg$.

We shall intensively exploit generating functions. Let $a_0, \ldots, a_n, \ldots$ be an infinite sequence of numbers, then $a(z) = \sum_{i=0}^{\infty} a_i z^i$ is the corresponding (ordinary) generating function. For example, the generating function corresponding to the sequence $1, 1, \ldots, 1, \ldots$ is $\frac{1}{1-z}$. The addition of generating functions corresponds to the addition of sequences and the multiplication of generating functions corresponds to the convolution of sequences. Knowing the generating function for a nonnegative sequence, we can bound the elements of a sequence by

$$a_i \leq \inf_{z \in Z} \frac{a(z)}{z^i}, \tag{2.1}$$

where Z is the set of all positive $z \in \mathbb{R}$ such that $a(z) = \sum_{i=0}^{\infty} a_i z^i$ converges.

We shall also use exponential generating functions $A(z) = \sum_{i=0}^{\infty} a_i \frac{z^i}{i!}$. For example, the generating function corresponding to the sequence $1, 1, \ldots, 1, \ldots$ is e^z. We can transform exponential generating functions into ordinary generating functions using

$$a(z) = \int_0^{\infty} e^{-x} A(zx) dx. \tag{2.2}$$

The addition of exponential generating functions corresponds to the addition of corresponding sequences. The product of the exponential generating

functions $A(z) = \sum_{i=0}^{\infty} a_i \frac{z^i}{i!}$ and $B(z) = \sum_{i=0}^{\infty} b_i \frac{z^i}{i!}$ is the exponential generating function $C(z) = \sum_{i=0}^{\infty} c_i \frac{z^i}{i!}$, where

$$c_k = \sum_{j=0}^{k} \binom{k}{j} a_{k-j} c_j \,.$$

More about generating functions can be found in [Rio78] or [Liu85].

We shall use standard O-notation. For functions $f(n)$ and $g(n)$ we say that f is $O(g)$ (f is $\Omega(g)$) if there is a constant C such that $f(n) \leq Cg(n)$ ($f(n) \geq Cg(n)$ respectively) for sufficiently large n. We say that f is $o(g)$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

2.2 Longest common subsequences

We shall work with pairs of sequences. The set of all pairs of sequences will be denoted by $\Pi = \Sigma^* \times \Sigma^*$. The total length of a pair is the sum of the lengths of the sequences from the pair, $l\binom{u}{v} = |\binom{u}{v}| = |u| + |v|$. Let t and b be the *projections* of a pair $\binom{u}{v}$ to its members: $t\binom{u}{v} = u$ and $b\binom{u}{v} = v$. Let $\mathrm{cat}(p, q) = \binom{t(p)t(q)}{b(p)b(q)}$ be the *concatenation* of pairs p, q. The definitions of concatenation, projections, and length are naturally extended to sequences of pairs:

$$\begin{aligned}
\mathrm{cat}(p_1, p_2, \ldots, p_n, p_{n+1}) &= \mathrm{cat}(\mathrm{cat}(p_1, p_2, \ldots, p_n), p_{n+1}) \,, \\
t(p_1, p_2, \ldots, p_n) &= t(\mathrm{cat}(p_1, p_2, \ldots, p_n)) = t(p_1) \cdots t(p_n) \,, \\
b(p_1, p_2, \ldots, p_n) &= b(\mathrm{cat}(p_1, p_2, \ldots, p_n)) = b(p_1) \cdots b(p_n) \,, \\
l(p_1, p_2, \ldots, p_n) &= l(\mathrm{cat}(p_1, p_2, \ldots, p_n)) = |p_1| + \cdots + |p_n| \,.
\end{aligned}$$

A sequence w is a *common subsequence* of u and v if it is a subsequence of both u and v. A longest common subsequence is a common subsequence of maximal possible length.

Definition 2.2 Let $u, v \in \Sigma^*$. Sequence $w \in \Sigma^*$ is a *longest common subsequence* if

1. $w \sqsubseteq u$ and $w \sqsubseteq v$,
2. $\forall w' \in \Sigma^* \ (w' \sqsubseteq u \ \& \ w' \sqsubseteq v \Longrightarrow |w'| \leq |w|)$.

We shall denote the length of a longest common subsequence of sequences u and v by $\mathbf{L}(u, v)$. While two sequences can have several different longest common subsequences, $\mathbf{L}(u, v)$ is unique. Sometimes we use CSS as an abbreviation for a common subsequence and LCSS for a longest common subsequence.

Example 2.1 Let $u = 10023211022233101$ and $v = 01113330212110121$. Common subsequences of u and v are for example 1321211 or 0113301. Both 011022101 and 102211021 are longest common subsequences and $\mathbf{L}(u, v) = 9$. ◇

Now we shall describe the basic properties of the length of longest common subsequences. The following lemma shows the invariant properties of $\mathbf{L}(u, v)$.

Lemma 2.1 *For permutation $\pi : \Sigma \to \Sigma$ let $S_\pi(u)$ denote the sequence obtained from sequence u after substitutions defined by π. For $u = u_1 \cdots u_n$ let $u^R = u_n \cdots u_1$ be the reversed sequence. Then for every $u, v \in \Sigma^*$*

1. $\mathbf{L}(u, v) = \mathbf{L}(v, u)$,
2. $\mathbf{L}(u, v) = \mathbf{L}(u^R, v^R)$,
3. $\mathbf{L}(u, v) = \mathbf{L}(S_\pi(u), S_\pi(v))$.

Proof. This lemma is a simple consequence of Definition 2.2. □

The following lemma about the monotonicity of $\mathbf{L}(u, v)$ is also a simple derivation from the definition of longest common subsequences.

Lemma 2.2 *For every* $u, v, u', v' \in \Sigma^*$,

1. *if* $u' \sqsubseteq u$ *and* $v' \sqsubseteq v$, *then* $\mathbf{L}(u', v') \leq \mathbf{L}(u, v)$,
2. $\mathbf{L}(u, v) + \mathbf{L}(u', v') \leq \mathbf{L}(uu', vv')$.

The basic recurrences for computing $\mathbf{L}(u, v)$ are given by Lemma 2.3.

Lemma 2.3 *For every* $u, v \in \Sigma^*$ *and* $a, b \in \Sigma$,

1. *if* $a = b$ *then* $\mathbf{L}(ua, vb) = \mathbf{L}(u, v) + 1$,
2. *if* $a \neq b$ *then* $\mathbf{L}(ua, vb) = \max\{\mathbf{L}(ua, v), \mathbf{L}(u, vb)\}$.

Proof. 1. Let $w = w_1 \cdots w_m$ be a longest common subsequence of ua and va. Then for $w' = w_1 \cdots w_{m-1}$ we have $w' \sqsubseteq u$ and $w' \sqsubseteq v$. Therefore, using the previous lemma, part 2, we get

$$\mathbf{L}(u, v) + 1 = \mathbf{L}(u, v) + \mathbf{L}(a, b) \leq \mathbf{L}(ua, vb) = |w| = |w'| + 1 \leq \mathbf{L}(u, v) + 1 .$$

2. Let $w = w_1 \cdots w_m$ be a longest common subsequence of ua and vb. If the last symbol of w is a, then $w \sqsubseteq v$ and $|w| \leq \mathbf{L}(ua, v)$. If the last symbol of w is not a, then $w \sqsubseteq u$ and $|w| \leq \mathbf{L}(u, vb)$. Hence $|w| \leq \max\{\mathbf{L}(ua, v), \mathbf{L}(u, vb)\}$. The opposite inequality is a consequence of Lemma 2.2, part 1. □

Part 2 of the previous lemma can be generalized in the following manner.

Lemma 2.4 *If* $u, v \in \Sigma^*$ *are such that* $\mathbf{L}(u, v) = 0$ *then for every* $u', v' \in \Sigma^*$ *we have* $\mathbf{L}(uu', vv') = \max\{\mathbf{L}(uu', v), \mathbf{L}(u, vv')\}$.

2.3 Algorithms for computing longest common subsequences

Finding a longest common subsequence is a widespread problem. There is strong motivation for finding efficient algorithms to compute longest common subsequences.

The algorithm presented by Wagner and Fischer [WF74] belongs now among classical dynamic programming methods. It is based on the recursive computation of $\mathbf{L}(x, y)$ for every $x \trianglelefteq u$ and $y \trianglelefteq v$. This can be described by the matrix $\boldsymbol{D} = \{d_{i,j}\}$, where $d_{i,j} = \mathbf{L}(u(1\,..\,i), v(1\,..\,j))$. Clearly $d_{0,j} = 0$ and $d_{i,0} = 0$ for all $i = 0, \ldots, |u|$ and $j = 0, \ldots, |v|$. For $i, j > 0$ we have the following recurrence based on Lemma 2.3.

$$d_{i,j} = \begin{cases} d_{i-1,j-1} + 1 & \text{if } \; u(i) = v(j)\,, \\ \max\{d_{i-1,j}, d_{i,j-1}\} & \text{if } \; u(i) \neq v(j)\,. \end{cases}$$

The time and space complexity of this algorithm is $O(mn)$, where $n = |u|$ and $m = |v|$.

Example 2.2 Matrix D computed by the dynamic programming algorithm for input sequences $u = 10023211022233101$ and $v = 01113330212110121$ is given in Figure 2.1. ◇

Hirschberg [Hir75] gave a variation of the dynamic programming algorithm, that computes $\boldsymbol{D}$ using only linear space.

Hunt and Szymanski [HS77] have improved the dynamic programming algorithm by computing only those entries of matrix $\boldsymbol{D}$ that correspond to

		1	0	0	2	3	2	1	1	0	2	2	2	3	3	1	0	1
0	0	①	**1**	1	1	1	1	1	**1**	1	1	1	1	1	1	**1**	1
1	①	1	1	1	1	1	②	**2**	2	2	2	2	2	2	**2**	2	**2**
1	**1**	1	1	1	1	1	**2**	③	3	3	3	3	3	3	**3**	3	**3**
1	**1**	1	1	1	1	1	**2**	**3**	3	3	3	3	3	3	④	4	**4**
3	1	1	1	1	②	2	2	3	3	3	3	3	④	**4**	4	4	4
3	1	1	1	1	**2**	2	2	3	3	3	3	3	**4**	⑤	5	5	5
3	1	1	1	1	**2**	2	2	3	3	3	3	3	**4**	**5**	5	5	5
0	1	②	**2**	2	2	2	2	3	④	4	4	4	4	5	5	⑥	6
2	1	2	2	③	3	**3**	3	3	4	⑤	**5**	**5**	5	5	5	6	6
1	**1**	2	2	3	3	3	④	**4**	4	5	5	5	5	5	⑥	6	⑦
2	1	2	2	**3**	3	④	4	4	4	**5**	⑥	**6**	6	6	6	6	7
1	**1**	2	2	3	3	4	⑤	**5**	5	5	6	6	6	6	⑦	7	**7**
1	**1**	2	2	3	3	4	**5**	⑥	6	6	6	6	6	6	**7**	7	⑧
0	1	**2**	③	3	3	4	5	6	⑦	7	7	7	7	7	7	⑧	8
1	**1**	2	3	3	3	4	**5**	**6**	7	7	7	7	7	7	⑧	8	⑨
2	1	2	3	④	4	**4**	5	6	7	⑧	**8**	**8**	8	8	8	8	9
1	**1**	2	3	4	4	4	**5**	**6**	7	8	8	8	8	8	⑨	9	**9**

Figure 2.1: Computing longest common subsequence using dynamic programming.

matches between symbols from u and v. These are typeset in bold in Figure 2.1.

Hirschberg [Hir77] concentrates on dominant matches. A match $a \leftrightarrow b$ is dominant, if $\mathbf{L}(ua, vb) > \max\{\mathbf{L}(u, v), \mathbf{L}(ua, v), \mathbf{L}(u, vb)\}$. Dominant matches are circled in Figure 2.1.

An algorithm with the best worst-case complexity is given by Masek and Paterson [MP80]. Their approach is based on the Four Russians' algorithm for computing transitive closure. We can split matrix $\boldsymbol{D}$ into small submatrices of size $\Theta(\log n)$. These submatrices can be precomputed and the relevant part of matrix $\boldsymbol{D}$ recovered in time $O(n^2/\log n)$.

Ukkonen [Ukk85] formulates the problem of finding longest common sub-

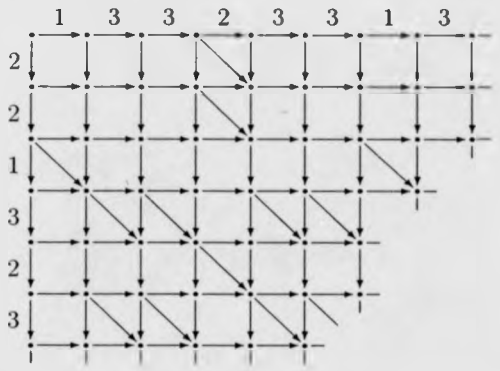

Figure 2.2: Graph corresponding to longest common subsequence computation.

sequences in terms of directed graphs. The graph corresponding to a pair of sequences is an oriented mesh with diagonals in place of matches (Figure 2.2). Computing **L** thus can be seen as finding the shortest path in the directed graph corresponding to pair $\binom{u}{v}$.

There are many improvements and variations of these algorithms. A list of some of them is in Figure 2.3. More detailed description of basic algorithms together with algorithms for other string problems can be found in [Ste92]. Problems of practical implementations of the longest common subsequence algorithms is in [Sim89]

A lot of effort have been made over the years to search for quick algorithms computing longest common subsequence. Despite this there is still quite a large gap between lower and upper bounds. The trivial lower bound is $\Omega(n)$. In the case of an unknown (infinite) alphabet Masek-Paterson's algorithm has complexity $O(n^2(\log\log n)^2/\log n)$. Hirschberg [Hir78] has shown lower bound $\Omega(n\log n)$ for the number of less than – equal – greater than queries

	Time	Space
Wagner-Fischer [WF74]	$O(mn)$	$O(mn)$
Hirschberg [Hir75]	$O(mn)$	$O(n)$
Hunt-Szymanski [HS77]	$O((n+R)\log n)$	$O(R+n)$
Hirschberg [Hir77]	$O(Ln + n\log n)$	$O(Ln)$
Hirschberg [Hir77]	$O(L(m-L)\log n)$	$O((m-L)^2+n)$
Masek-Paterson [MP80]	$O(n\max\{1, m/\log n\})$	$O(n^2/\log n)$
Nakatsu at al. [NKY82]	$O(n(m-L))$	$O(m^2)$
Hsu-Du [HD84, Apo87]	$O(Lm\log(n/L)+Lm)$	$O(Lm)$
Ukkonen [Ukk85]	$O(Em)$	$O(E\min\{m, E\})$
Apostolico [Apo86]	$O(n + m\log n + D\log(mn/D))$	$O(R+n)$
Kumar-Rangan [KR87]	$O(n(m-L))$	$O(n)$
Apostolico-Guerra [AG87]	$O(Lm+n)$	$O(D+n)$
Wu at al. [WMMM90]	$O(n(m-L))$	$O(n)$
Chin-Poon [CP90]	$O(n+\min\{D, Lm\})$	$O(D+n)$
Apostolico at al. [ABG92]	$O(n(m-L))$	$O(n)$
Apostolico at al. [ABG92]	$O(Lm)$	$O(n)$
Eppstein at al. [EGGI92]	$O(n + D\log\log\min\{D, mn/D\})$	$O(D+m)$

Figure 2.3: Time and space complexity of algorithms computing LCSS. Here $m = |u|$, $n = |v|$, $m \leq n$, R is the number of matches, L is the length of a longest common subsequence, $E = m + n - 2L$ is the edit distance, and D is the number of dominant matches.

over infinite alphabet. Wong and Chandra [WC76] consider a restricted model for deriving lower bounds. They have shown, that every algorithm over an infinite alphabet using only equal-unequal queries requires $O(n^2)$ operations.

Lin, Lu, and Fang [LLF91] gave a parallel algorithm (CREW PRAM) with complexity $O(\log^2 m \log\log m)$ using $mn/\log^2 m \log\log m$ processors when $\log^2 m \log\log m > \log n$ and with complexity $O(\log n)$ using mn processors otherwise. This improves the algorithm of Lu [Lu90], that has complexity $O(\log L \log^2 n)$ and uses R processors. For implementations of longest common subsequence algorithms on other parallel models see [And86, ITH92].

We can extend the dynamic programming algorithm to compute the length

of a longest common subsequence of l sequences $u_1, \ldots, u_l$. The task is then to compute an l-dimensional array and the algorithm complexity is $O(n^l)$. This is an exponential algorithm in terms of l and there is little expectation of an polynomial algorithm, since Maier [Mai78] has shown this problem to be NP-complete. However, there are algorithms for finding $\mathbf{L}(u_1, \ldots, u_l)$ with better complexity than $O(n^l)$ (but still exponential in l) [HD84, IF92, HI92, BY91].

A slightly different aproach is needed for the approximate string matching problem. In such case we have to compute longest common subsequences of many short sequences [HD80, CL92].

2.4 Expected length of longest common subsequences

We can use the length of the longest common subsequence as a measure of similarity of two sequences. Essential for a decision, as to whether two sequences are similar or not, is knowledge of the expected length of a longest common subsequence.

Definition 2.3 The expected length $\mathbf{EL}_n$ of a longest common subsequence is the average value of the longest common subsequences over all pairs of strings of the same length n, i.e.,

$$\mathbf{EL}_n = \frac{1}{k^{2n}} \sum_{u,v \in \Sigma^n} \mathbf{L}(u, v). \tag{2.3}$$

Chvátal and Sankoff [CS75] were the first who investigated the properties of the expected length of a longest common subsequence. They have computed values of $\mathbf{EL}_n$ for $n = 1, \ldots, 5$ and $k = 1, \ldots, 15$. They also showed that $\mathbf{EL}_n$ is linear in n and gave bounds on $\frac{\mathbf{EL}_n}{n}$.

Theorem 2.1 *For every $k \geq 2$ there is γ_k such that*

$$\gamma_k = \lim_{n\to\infty} \frac{\mathbf{EL}_n}{n} = \sup\left\{\frac{\mathbf{EL}_n}{n} : n \in \mathbb{N}, n > 0\right\}. \tag{2.4}$$

Before we prove the theorem, we shall show some basic properties of $\mathbf{EL}_n$. The expected length of a longest common subsequence is *superadditive* in the following sense.

Lemma 2.5 *For every $m, n \in \mathbb{N}$, $m, n \geq 1$,*

$$\mathbf{EL}_m + \mathbf{EL}_n \leq \mathbf{EL}_{m+n}$$

Proof. From Lemma 2.2 we have

$$\mathbf{L}(u, v) + \mathbf{L}(u', v') \leq \mathbf{L}(uu', vv').$$

For the expected length we then have

$$\mathbf{EL}_m + \mathbf{EL}_n = \mathbf{E}(\mathbf{L}(u, v) + \mathbf{L}(u', v')) \leq \mathbf{EL}_{m+n}.$$

□

Corollary 2.1 *For every $m, n \in \mathbb{N}$, $m, n \geq 1$,*

$$m\mathbf{EL}_n \leq \mathbf{EL}_{mn}$$

Proof. By induction on m. For $m = 1$ we have $\mathbf{EL}_n \leq \mathbf{EL}_n$. For $m > 1$ we have

$$m\mathbf{EL}_n = (m-1)\mathbf{EL}_n + \mathbf{EL}_n \leq \mathbf{EL}_{(m-1)n} + \mathbf{EL}_n \leq \mathbf{EL}_{mn}$$

□

Now, using superadditivity of $\mathbf{EL}_n$ we can prove Theorem 2.1

Proof of Theorem 2.1. Let γ_k be $\sup\left\{\frac{\mathbf{EL}_n}{n} : n \in \mathbb{N}, n > 0\right\}$. Let $\varepsilon > 0$ be any real number. Since $\gamma_k = \sup \frac{\mathbf{EL}_n}{n}$, there is $m \in \mathbb{N}$ such that $\mathbf{EL}_m > (\gamma_k - \varepsilon)m$. Now let n be any natural number such that $\varepsilon n > \mathbf{EL}_m$. We can express n as $am + b$ where $a, b \in \mathbb{N}$ and $0 \leq b < m$. Then, using Lemma 2.5 and Corollary 2.1, we have

$$\begin{aligned}\gamma_k \geq \frac{\mathbf{EL}_n}{n} &\geq \frac{a\mathbf{EL}_m}{n} + \frac{\mathbf{EL}_b}{n} \\ &\geq \frac{(n-b)\mathbf{EL}_m}{mn} \\ &\geq \frac{\mathbf{EL}_m}{m} - \frac{\mathbf{EL}_m}{n} > \gamma_k - 2\varepsilon\end{aligned}$$

This means that $\lim_{n\to\infty} \frac{\mathbf{EL}_n}{n}$ exists and its value is γ_k. □

Exact values of the constants γ_k are not known even in the case $k = 2$. Upper and lower bounds on γ_k will be described in the forthcoming chapters. However, it is possible to generate pairs of pseudo-random sequences and to compute the lengths of their longest common subsequences. In Figure 2.4 are shown values of $\frac{\mathbf{L}(u,v)}{|u|}$ for successive prefixes u, v of four pairs of binary sequences of length one million. To interpret this data we use the following estimate about the convergence behaviour of $\mathbf{EL}_n$ due to Alexander [Ale92]:

$$\gamma_k n - O(\sqrt{n \log n}) \leq \mathbf{EL}_n \leq \gamma_k n\,.$$

The best fit method used with the data from Figure 2.4 gives the function $0.812375n - 0.0718853\sqrt{n \log n}$. That suggests that γ_2 is likely to lie in the interval $(0.8120, 0.8125)$.

For large k we can see that $\gamma_k \to 0$ as $k \to \infty$. Deken [Dek79] has shown, that the speed of this convergence is not smaller than $1/\sqrt{k}$, more precisely

$$\lim_{k\to\infty} \gamma_k \sqrt{k} \geq 1\,.$$

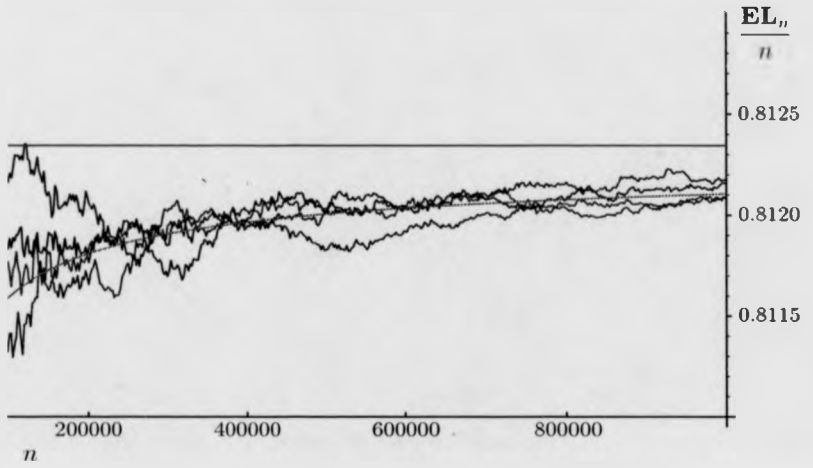

Figure 2.4: Random strings of length one million.

An upper bound of Chvátal and Sankoff [CS75] leads to following estimate:

$$\lim_{k\to\infty} \gamma_k \sqrt{k} \leq e\,,$$

while Sankoff and Mainville [SK83] conjecture that

$$\lim_{k\to\infty} \gamma_k \sqrt{k} = 2\,.$$

When using the dynamic programming algorithm to compute $\mathbf{L}(u, v)$, we actually compute $\mathbf{L}(x, y)$ for all prefixes $x \trianglelefteq u$ and $y \trianglelefteq v$. We can group these prefixes according to diagonals.

Let $D_m(u, v)$ be the set of all pairs $\binom{x}{y}$ of total length m such that $x \trianglelefteq u$ and $y \trianglelefteq v$. If $m = \min\{|u|, |v|\}$, we shall omit the index and simply write $D(u, v)$. Now we can define a 'diagonal' longest common subsequence.

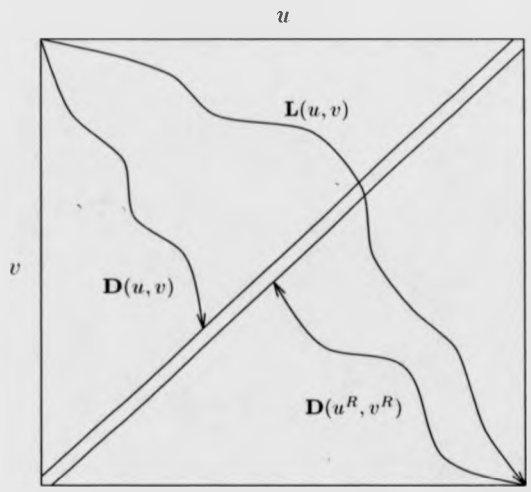

Figure 2.5: Diagonal longest common subsequence.

Definition 2.4 Let $u, v \in \Sigma^*$. We define a *diagonal longest common subsequence* by

$$\mathbf{D}(u, v) = \max\left\{\mathbf{L}(x, y) : \binom{x}{y} \in D(u, v)\right\}.$$

The expected length of a diagonal longest common subsequence, $\mathbf{ED}_n$, is defined analogously to (2.3). When computing $\mathbf{L}(u, v)$ we have to 'hit' the opposite corner, but to compute $\mathbf{D}(u, v)$ it is enough to achieve an entry somewhere on the diagonal of the matrix computed by the dynamic programming algorithm.

The $\mathbf{D}(u, v)$ and $\mathbf{D}(u^R, v^R)$ are the maxima on diagonals, but if a match from a longest common subsequence appear exactly on the main diagonal, it contributes neither to $\mathbf{D}(u, v)$ nor $\mathbf{D}(u^R, v^R)$ (illustrated in Figure 2.5). Therefore $\mathbf{L}(u, v) \leq \mathbf{D}(u, v) + \mathbf{D}(u^R, v^R) + 1$ and this gives $2\mathbf{ED}_n + 1$ as an

upper bound for $\mathbf{EL}_n$. A lower bound for $\mathbf{EL}_n$ in the terms of $\mathbf{ED}_n$ is given by Alexander [Ale92].

Lemma 2.6 (Alexander [Ale92]) *There is a constant α such, that for all n*

$$2\mathbf{ED}_n - \alpha\sqrt{n \log n} \leq \mathbf{EL}_n \leq 2\mathbf{ED}_n + 1$$

Combining Lemma 2.6 and Theorem 2.1 we immediately get the following theorem.

Theorem 2.2 *For every k there is δ_k such that*

$$\delta_k = \lim_{n\to\infty} \frac{\mathbf{ED}_n}{n}.$$

Moreover $2\delta_k = \gamma_k$.

This theorem will be helpful when analyzing the $\mathbf{EL}_n$. It means that we can omit the condition $|u| = |v| = n$, it is enough if u and v satisfy $|u| + |v| = 2n$.

The *variance* $\mathrm{Var}(\mathbf{L}_n)$ is another value describing the properties of the length of longest common subsequences,

$$\mathrm{Var}(\mathbf{L}_n) = \frac{1}{k^{2n}} \sum_{u,v\in\Sigma^n} (\mathbf{L}(u,v) - \mathbf{EL}_n)^2.$$

Chvátal and Sankoff [CS75] conjecture that $\mathrm{Var}(\mathbf{L}_n)$ is $o(n^{2/3})$. Steele [Ste82] has shown that $\mathrm{Var}(\mathbf{L}_n) \leq (\sqrt{n}+1)^2$. Later, in [Ste86], he improved his bound to

$$\mathrm{Var}(\mathbf{L}_n) \leq \left(1 - \tfrac{1}{k}\right)n\,.$$

Both bounds are based on the Effron-Stein inequality.

Chapter 3

Lower Bounds

In the last section of the previous chapter we have defined the expected length $\mathbf{EL}_n$ of a longest common subsequence and the constants γ_k. In Section 2.3 we have described algorithms computing $\mathbf{L}(u, v)$ but unfortunately we are not able to analyse the behaviour of these algorithms for a random input. Thus the constants γ_k remain unknown yet. In this chapter we shall derive new lower bounds for γ_2. Also we can achieve previously known lower bounds for γ_k using a different approach.

3.1 Css machines

The basic idea for obtaining lower bounds is quite simple. Let $\mathfrak{C}$ be any algorithm such that for input $u, v \in \Sigma^*$ it computes $w \in \Sigma^*$ — some common subsequence of u and v. If we denote $\mathbf{C}(u, v) = |w|$ then $\mathbf{C}(u, v) \leq \mathbf{L}(u, v)$ and $\mathbf{EC}_n \leq \mathbf{EL}_n$, where $\mathbf{EC}_n$ is the expected length of $\mathbf{C}(u, v)$ for random inputs $u, v \in \Sigma^n$.

The main problem is to design an algorithm $\mathfrak{C}$ in such way that we are able to analyse it for a random input. As a model for algorithms computing

common subsequences we choose finite state machines. Their analysis will rely on methods developed to analyse finite Markov chains. Our finite state machines will have two input tapes and a counter as output.

Definition 3.1 A *finite state machine* $\mathfrak{M}$ is a 5-tuple $\mathfrak{M} = \langle S, \Sigma, I, T, O \rangle$, where S is a finite set of states, Σ is a set of input symbols, $I : S \to \{\uparrow, \downarrow\}$ is a *tape switch function*, $T : S \times \Sigma \to S$ is a *transition function*, and $O : S \times \Sigma \to \mathbb{N}$ is an *output function*.

Every state $s \in S$ and symbol $a \in \Sigma$ determine a *transition* $s \xrightarrow{a} T(s,a)$. We sometimes also write $s \xrightarrow{a+O(s,a)} T(s,a)$ when the value of output function is important.

In one step of the computation we decide which tape to read according to the value of the tape switch function in the current state. We read one symbol from the tape and advance the input head of the tape. The transition function determines a new state of the machine and the output counter is advanced by a value given by the output function.

Suppose we start computation in the state s with u and v being the contents of the tapes. The state of the machine after m steps of computation determines the value of an *extended transition function* $T^m : S \times \Sigma^* \times \Sigma^* \to S$. For $m = 0$ we define $T^0(s,u,v) = s$ and for $m > 0$ and $a \in \Sigma$ we define

$$\begin{aligned}
T^m(s,\lambda,v) &\quad \text{not defined} && \text{if } I(s) = \uparrow, \\
T^m(s,au,v) &= T^{m-1}(T(s,a),u,v) && \text{if } I(s) = \uparrow, \\
T^m(s,u,\lambda) &\quad \text{not defined} && \text{if } I(s) = \downarrow, \\
T^m(s,u,av) &= T^{m-1}(T(s,a),u,v) && \text{if } I(s) = \downarrow .
\end{aligned}$$

We shall write $T^*(s,u,v)$ instead of $T^{|u|+|v|}(s,u,v)$.

A computation on machine $\mathfrak{M}$ yields a function $\mathbf{C}_{\mathfrak{M}}(s, u, v, m)$. Informally $\mathbf{C}_{\mathfrak{M}}(s, u, v, m)$ is the value of the output counter after an m-step computation on $\mathfrak{M}$ when starting in state s and sequences u and v are on the input tapes.

Definition 3.2 The *computation function* of machine $\mathfrak{M} = \langle S, \Sigma, I, T, O\rangle$ on input u, v starting in state s is the function $\mathbf{C}_{\mathfrak{M}} : S \times \Sigma^* \times \Sigma^* \to \mathbb{N}$ defined recursively:

$$\begin{aligned}
\mathbf{C}_{\mathfrak{M}}(s, u, v, 0) &= 0 \\
\mathbf{C}_{\mathfrak{M}}(s, \lambda, v, m) &= 0 && \text{if } I(s) = \uparrow, \\
\mathbf{C}_{\mathfrak{M}}(s, au, v, m) &= \mathbf{C}_{\mathfrak{M}}(T(s, a), u, v, m-1) + O(s, a) && \text{if } I(s) = \uparrow, \\
\mathbf{C}_{\mathfrak{M}}(s, u, \lambda, m) &= 0 && \text{if } I(s) = \downarrow, \\
\mathbf{C}_{\mathfrak{M}}(s, u, av, m) &= \mathbf{C}_{\mathfrak{M}}(T(s, a), u, v, m-1) + O(s, a) && \text{if } I(s) = \downarrow .
\end{aligned}$$

We shall omit the index $\mathfrak{M}$ when there is no ambiguity. To obtain lower bounds we need an analysable machine computing the length of some common subsequence of input sequences. To get such a machine we have to impose some more constraints on the definition of finite state machine.

The states of a machine will be represented by two sequences. The sequences consist of symbols already read from the input tapes but not used for producing a common subsequence yet.

Every transition appends the symbol just read to the corresponding sequence from the state. Some transitions can also match some symbols and thus reduce the 'size' of the state. This is done by cutting off prefixes and adding the length of a longest common subsequence of the prefixes to the output counter. In some favorable states we can leave the decision about the matching for the future. Such states have the form $\binom{xyu}{yxv}$, where $|x| = |y| > 0$ and $\mathbf{L}(x, y) = 0$. The validity of such a move is supported by Lemma 2.4.

A computation on machine $\mathfrak{M}$ yields a function $\mathbf{C}_{\mathfrak{M}}(s, u, v, m)$. Informally $\mathbf{C}_{\mathfrak{M}}(s, u, v, m)$ is the value of the output counter after an m-step computation on $\mathfrak{M}$ when starting in state s and sequences u and v are on the input tapes.

Definition 3.2 The *computation function* of machine $\mathfrak{M} = \langle S, \Sigma, I, T, O\rangle$ on input u, v starting in state s is the function $\mathbf{C}_{\mathfrak{M}} : S \times \Sigma^* \times \Sigma^* \to \mathbb{N}$ defined recursively:

$$
\begin{aligned}
\mathbf{C}_{\mathfrak{M}}(s, u, v, 0) &= 0 \\
\mathbf{C}_{\mathfrak{M}}(s, \lambda, v, m) &= 0 && \text{if } I(s) = \uparrow, \\
\mathbf{C}_{\mathfrak{M}}(s, au, v, m) &= \mathbf{C}_{\mathfrak{M}}(T(s, a), u, v, m-1) + O(s, a) && \text{if } I(s) = \uparrow, \\
\mathbf{C}_{\mathfrak{M}}(s, u, \lambda, m) &= 0 && \text{if } I(s) = \downarrow, \\
\mathbf{C}_{\mathfrak{M}}(s, u, av, m) &= \mathbf{C}_{\mathfrak{M}}(T(s, a), u, v, m-1) + O(s, a) && \text{if } I(s) = \downarrow .
\end{aligned}
$$

We shall omit the index $\mathfrak{M}$ when there is no ambiguity. To obtain lower bounds we need an analysable machine computing the length of some common subsequence of input sequences. To get such a machine we have to impose some more constraints on the definition of finite state machine.

The states of a machine will be represented by two sequences. The sequences consist of symbols already read from the input tapes but not used for producing a common subsequence yet.

Every transition appends the symbol just read to the corresponding sequence from the state. Some transitions can also match some symbols and thus reduce the 'size' of the state. This is done by cutting off prefixes and adding the length of a longest common subsequence of the prefixes to the output counter. In some favorable states we can leave the decision about the matching for the future. Such states have the form $\binom{xyu}{yxv}$, where $|x| = |y| > 0$ and $\mathbf{L}(x, y) = 0$. The validity of such a move is supported by Lemma 2.4.

Definition 3.3 A finite state machine $\mathfrak{M} = \langle S, \Sigma, I, T, O \rangle$ is called a *css machine* if $S \subseteq \Sigma^* \times \Sigma^*$, $\binom{\lambda}{\lambda} \in S$, and for every $s = \binom{u}{v} \in S$ and $a \in \Sigma$ at least one of the following three conditions is satisfied. We denote $u' = ua$, $v' = v$ if $I(s) = \uparrow$, and $u' = u$, $v' = va$ if $I(s) = \downarrow$.

1. $T(s,a) = \binom{u'}{v'}$, and $O(s,a) = 0$,
2. $T(s,a) = \binom{x'}{y'}$, and $O(s,a) = \mathbf{L}(x,y)$, where $xx' = u'$, $yy' = v'$,
3. $T(s,a) = \binom{xx'}{yy'}$, and $O(s,a) = |x| = |y| > 0$, where $yxx' = u'$, $xyy' = v'$, $\mathbf{L}(x,y) = 0$,

Transitions of the form 1 and 3 are called *saturated*. State $s \in S$ is saturated if transitions $s \xrightarrow{a} T(s,a)$ are saturated for every $a \in \Sigma$.

Example 3.1 A css machine with 11 states is described by the table in Figure 3.1. ◇

A css machines are designed to produce the length of a possible common subsequence of its inputs.

Lemma 3.1 *Let $\mathfrak{M}$ be a css machine and $\mathbf{C}$ its computation function. Then for every $s = \binom{w}{z} \in S$ and $m \in \mathbb{N}$*

$$\mathbf{C}(s,u,v,m) \leq \mathbf{L}(ww', zz'),$$

where $w' \trianglelefteq u$ ($z' \trianglelefteq v$) is the sequence of symbols that are read by $\mathfrak{M}$ from the first (second) tape while computing $\mathbf{C}(s,u,v,m)$.

s	$I(s)$	$T(s,0)$	$O(s,0)$	$T(s,1)$	$O(s,1)$
$s_0 = \binom{\lambda}{\lambda}$	$\uparrow$	$\binom{0}{\lambda}$	0	$\binom{1}{\lambda}$	0
$s_1 = \binom{0}{\lambda}$	$\downarrow$	$\binom{\lambda}{\lambda}$	1	$\binom{0}{1}$	0
$s_2 = \binom{1}{\lambda}$	$\downarrow$	$\binom{1}{0}$	0	$\binom{\lambda}{\lambda}$	1
$s_3 = \binom{0}{1}$	$\uparrow$	$\binom{00}{1}$	0	$\binom{01}{1}$	0
$s_4 = \binom{1}{0}$	$\uparrow$	$\binom{10}{0}$	0	$\binom{11}{0}$	0
$s_5 = \binom{00}{1}$	$\downarrow$	$\binom{0}{\lambda}$	1	$\binom{00}{1}$	0
$s_6 = \binom{01}{1}$	$\downarrow$	$\binom{1}{0}$	1	$\binom{\lambda}{1}$	1
$s_7 = \binom{10}{0}$	$\downarrow$	$\binom{\lambda}{0}$	1	$\binom{0}{1}$	1
$s_8 = \binom{11}{0}$	$\downarrow$	$\binom{11}{0}$	0	$\binom{1}{\lambda}$	1
$s_9 = \binom{\lambda}{0}$	$\uparrow$	$\binom{\lambda}{\lambda}$	1	$\binom{1}{0}$	0
$s_{10} = \binom{\lambda}{1}$	$\uparrow$	$\binom{0}{1}$	0	$\binom{\lambda}{\lambda}$	1

Figure 3.1: Css machine with 11 states.

Proof. By induction on m. If $m = 0$, then we have $\mathbf{C}(s,u,v,m) = 0 \leq \mathbf{L}(ww', zz')$. Now let $m > 0$ and let $\mathbf{C}(s,u,v,m-1) \leq \mathbf{L}(ww', zz')$ for all u, v, s. Without loss of generality we can suppose $I(s) = \uparrow$. We have $\mathbf{C}(s,\lambda,v,m) = 0 \leq \mathbf{L}(ww', zz')$. While considering $\mathbf{C}(s,au,v,m)$ we have the following cases:

1. $T(s,a) = \binom{wa}{z}$, $O(s,a) = 0$. $\mathbf{C}(s,au,v,m) = \mathbf{C}(T(s,a),u,v,m-1) \leq \mathbf{L}(waw', zz')$.

2. $T(s,a) = \binom{x'}{y'}$, $O(s,a) = \mathbf{L}(x,y)$, where $xx' = wa$, $yy' = z$. Then we have

$$\mathbf{C}(s,au,v,m) = \mathbf{L}(x,y) + \mathbf{C}(T(s,a),u,v,m-1)$$

$$\begin{aligned} &\leq \mathbf{L}(x,y) + \mathbf{L}(x'w', y'z') \\ &\leq \mathbf{L}(xx'w', yy'z') = \mathbf{L}(waw', zz')\,. \end{aligned}$$

3. $T(s,a) = \binom{xx'}{yy'}$, $O(s,a) = |x| = |y|$, where $yxx' = wa$ and $xyy' = z$. According to Lemma 2.4, then $\mathbf{L}(xx'w', yy'z')$ must be either $\mathbf{L}(xx'w', y'z')$ or $\mathbf{L}(x'w', yy'z')$. In the first case we have

$$\begin{aligned} \mathbf{C}(s, au, v, m) &= O(s,a) + \mathbf{C}(T(s,a), u, v, m-1) \\ &= \mathbf{L}(y, xy) + \mathbf{C}(T(s,a), u, v, m-1) \\ &\leq \mathbf{L}(y, xy) + \mathbf{L}(xx'w', yy'z') = \mathbf{L}(y, xy) + \mathbf{L}(xx'w', y'z') \\ &\leq \mathbf{L}(yxx'w', xyy'z') = \mathbf{L}(waw', zz')\,. \end{aligned}$$

In the second case we can prove that $\mathbf{C}(s, au, v, m) \leq \mathbf{L}(waw', zz')$ in the same way. □

In most cases we are interested in computations started in state $s_0 = \binom{\lambda}{\lambda}$ and we set $\mathbf{C}(u, v, m) = \mathbf{C}_{\mathfrak{M}}(s_0, u, v, m)$.

Corollary 3.1 *Let $\mathfrak{M}$ be a css machine and $\mathbf{C}$ its computation function. Then for every input u, v the computation function satisfies*

1. $\mathbf{C}(u, v, |u| + |v|) \leq \mathbf{L}(u, v)$,
2. $\mathbf{C}(u, v, \min\{|u|, |v|\}) \leq \mathbf{D}(u, v)$.

Proof. 1. Let w', z' be the sequences read from the input tapes while computing $\mathbf{C}(u, v, |u| + |v|)$. Then

$$\mathbf{C}(u, v, |u| + |v|) = \mathbf{C}(\tbinom{\lambda}{\lambda}, u, v, |u| + |v|) \leq \mathbf{L}(w', z') \leq \mathbf{L}(u, v)\,,$$

since w' is a subsequence of u and z' is a subsequence of v.

2. Let $m = \min\{|u|, |v|\}$. Let w', z' be the sequences read from the input tapes while computing $\mathbf{C}(u, v, m)$. During every step of the computation machine $\mathfrak{M}$ read exactly one input symbol. Moreover, m is sufficiently small that we cannot run out of input symbols. So we have $|w'| + |z'| = m$ and $\binom{w'}{z'} \in D(u, v)$, therefore

$$\begin{aligned} \mathbf{C}(u, v, m) &= \mathbf{C}(\tbinom{\lambda}{\lambda}, u, v, m) \leq \mathbf{L}(w', z') \\ &\leq \max\left\{\mathbf{L}(x, y) : \tbinom{x}{y} \in D(u, v)\right\} = \mathbf{D}(u, v)\,. \end{aligned}$$

□

Previous lower bounds for the expected length of a longest common subsequence were derived using the first inequality of Corollary 3.1. To simplify the analysis of a machine we shall use the second inequality in conjunction with Theorem 2.2. After taking the average over all sequences of length m we directly get the following theorem.

Theorem 3.1 *Let $\mathfrak{M}$ be a css machine and $\mathbf{C}_{\mathfrak{M}}$ its computation function. Let*

$$\mathbf{EC}_m(\mathfrak{M}) = \frac{1}{k^{2m}} \sum_{u,v \in \Sigma^m} \mathbf{C}_{\mathfrak{M}}(u, v, \min\{|u|, |v|\})\,.$$

If $\lim\limits_{m\to\infty} \dfrac{\mathbf{EC}_m(\mathfrak{M})}{m}$ exists, then

$$\delta'(\mathfrak{M}) = \lim_{m\to\infty} \frac{\mathbf{EC}_m(\mathfrak{M})}{m} \leq \lim_{m\to\infty} \frac{\mathbf{ED}_m}{m} = \delta_k = \frac{\gamma_k}{2}.$$

3.2 Analysis of css machines

To compute $\mathbf{EC}_m$ we have to analyse the behaviour of css machines when a random input is given. This analysis is similar to the analysis of finite Markov

chains [KS60]. Changes of current states are described by the 'transition' matrix of the machine.

Definition 3.4 Let $\mathfrak{M} = \langle S, \Sigma, I, T, O \rangle$ be a finite state machine. Let $\boldsymbol{T} = \{t_{i,j}\}$ be a matrix such that

$$t_{i,j} = \frac{1}{k} |\{a \in \Sigma : T(s_i, a) = s_j\}|, \tag{3.1}$$

where $k = |\Sigma|$. Then matrix $\boldsymbol{T}$ is called the *transition matrix* of machine $\mathfrak{M}$.

We shall introduce 'regular' machines that are easier to analyse.

Definition 3.5 A finite state machine $\mathfrak{M}$ with transition matrix $\boldsymbol{T} = \{t_{i,j}\}$ is called *regular* if there is $n \in \mathbb{N}$ such that $t_{i,j}^{(n)} > 0$ for all i, j, where $t_{i,j}^{(n)}$ is the (i,j)-th entry of matrix $\boldsymbol{T}^n$.

It is not hard to see that a css machine $\mathfrak{M}$ is regular if and only if there is some $m \in \mathbb{N}$ such that for every two states $s, t \in S$ there is an input $u, v \in \Sigma^*$ such that $|u| + |v| = m$ and $T^*(s, u, v) = t$.

Lemma 3.2 *Let $\boldsymbol{T}$ be the transition matrix of a regular css machine $\mathfrak{M}$. Let $t_{i,j}^{(n)}$ be the (i,j)-th entry of the n-th power of matrix T. Then*

$$t_{i,j}^{(n)} = \frac{1}{k^{2n}} |\{u, v \in \Sigma^n : T^n(s_i, u, v) = s_j\}|.$$

Definition 3.6 Let $\mathfrak{M} = \langle S, \Sigma, I, T, O \rangle$ be a finite state machine and let $s, t \in S$. We say, that state t is *reachable* from s if there are $u, v \in \Sigma^*$ such that $T^*(s, u, v) = t$.

Lemma 3.3 *Let $\mathfrak{M} = \langle S, \Sigma, I, T, O\rangle$ be a css machine satisfying the following conditions:*

1. *Every state $s \in S$ is reachable from $\binom{\lambda}{\lambda}$,*
2. *$\binom{\lambda}{\lambda}$ is reachable from every state $s \in S$,*
3. *there are a state $s \in S$ and a symbol $a \in \Sigma$ such that $T(s,a) = s$.*

Then machine $\mathfrak{M}$ is regular.

Proof. For $s, t \in S$, t is reachable from s, and we denote by $d(s,t)$ their 'distance', i.e.,

$$d(s,t) = \min\{|u| + |v| : T^*(s,u,v) = t\}.$$

Let $m_1 = \max_{t \in S} d(\binom{\lambda}{\lambda}, t)$ and $m_4 = \max_{t \in S} d(t, \binom{\lambda}{\lambda})$. Now let s_l be the state whose existence is guaranteed by condition 3. We put $m_2 = d(\binom{\lambda}{\lambda}, s_l)$, $m_3 = d(s_l, \binom{\lambda}{\lambda})$, and we set $m = m_1 + m_2 + m_3 + m_4$.

Let $s, t \in S$ be any two states of $\mathfrak{M}$. Let $u_1, v_1 \in \Sigma^*$, $|u_1| + |v_1| \leq m_1$ be an input such that $T^*(s, u_1, v_1) = \binom{\lambda}{\lambda}$. Let $u_2, v_2 \in \Sigma^*$, $|u_2| + |v_2| = m_2$ be an input such that $T^*(\binom{\lambda}{\lambda}, u_2, v_2) = s_l$. Let $u_3, v_3 \in \Sigma^*$, $|u_3| + |v_3| = m_3$ be an input such that $T^*(s_l, u_3, v_3) = \binom{\lambda}{\lambda}$. Let $u_4, v_4 \in \Sigma^*$, $|u_4| + |v_4| \leq m_4$ be an input such that $T^*(\binom{\lambda}{\lambda}, u_4, v_4) = t$. Let $i = m - \sum_{j=1}^{4}(|u_j| + |v_j|)$. If $I(s_l) = \uparrow$ we set $u = u_1u_2a^iu_3u_4$ and $v = v_1v_2v_3v_4$, otherwise $u = u_1u_2u_3u_4$ and $v = v_1v_2a^iv_3v_4$. Then $|u| + |v| = m$ and $T^*(s,u,v) = t$. This guarantees $\mathfrak{M}$ to be regular. □

From the theory of Markov chains [KS60, Fel68] we get the following lemma.

Lemma 3.4 *Let $\boldsymbol{T}$ be the transition matrix of a regular css machine $\mathfrak{M}$ with l states. Let $\boldsymbol{e} = (e_1, \ldots, e_l) \in \mathbb{R}^l$ be any vector such that $\sum e_i = 1$ and let $\boldsymbol{c}_n = (c_{n,1}, \ldots, c_{n,l}) = \boldsymbol{e}\boldsymbol{T}^n$. Then for every $j = 1, \ldots, l$*

$$\lim_{n\to\infty} c_{n,i} = d_i\,,$$

where $\boldsymbol{d} = (d_1, \ldots, d_l) \in \mathbb{R}^l$ is the unique vector such that $\boldsymbol{d}\boldsymbol{T} = \boldsymbol{d}$ and $\sum d_i = 1$.

The vector $\boldsymbol{d} = (d_1, \ldots, d_l)$ whose existence is guaranteed by Lemma 3.4 is called a *stationary probability distribution* of the regular css machine. For state $s_i \in S$ we define $d(s_i) = d_i$, the stationary probability distribution of machine $\mathfrak{M}$ in the state s_i. As an immediate corollary of Lemma 3.4 we get following theorem.

Theorem 3.2 *Let $\boldsymbol{T}$ be the transition matrix of a regular css machine $\mathfrak{M}$ with l states. Let $\boldsymbol{d} = (d_1, \ldots, d_l)$ be a stationary distribution of machine $\mathfrak{M}$. Then the expected length of a common subsequence produced by css machine $\mathfrak{M}$ is asymptotically $\delta'(\mathfrak{M})n$, where*

$$\delta'(\mathfrak{M}) = \sum_{i=1}^{l} d_i \sum_{a\in\Sigma} O(s_i, a)\,. \tag{3.2}$$

Example 3.2 Because of symmetries given by parts 1 and 3 of Lemma 2.1, we can reduce the size of css machines. The machine from Example 3.1 after reduction will have five states $t_0 = \left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]$, $t_1 = \left[\begin{smallmatrix}0\\ \lambda\end{smallmatrix}\right]$, $t_2 = \left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right]$, $t_3 = \left[\begin{smallmatrix}00\\ 1\end{smallmatrix}\right]$, and $t_4 = \left[\begin{smallmatrix}01\\ 1\end{smallmatrix}\right]$. The reduced machine $\mathfrak{R}$ is described by the transition graph in Figure 3.2. The value of the tape switch function is denoted by the symbol $\bullet$,

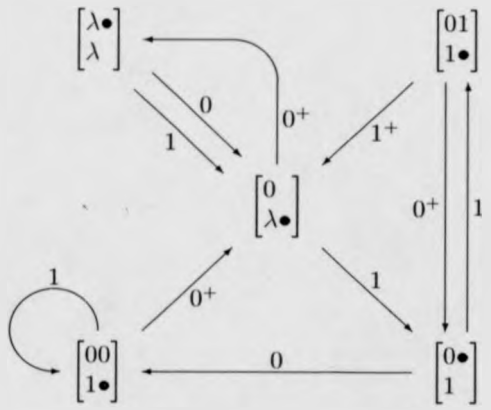

Figure 3.2: A css machine $\mathfrak{R}$ that yields the lower bound 0.7368.

and the transitions for which the output function is 1 are marked by the symbol +. The transition matrix defined by (3.1) is

$$\boldsymbol{T} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix}.$$

Since

$$\boldsymbol{T}^6 = \begin{pmatrix} \frac{1}{8} & \frac{5}{16} & \frac{1}{8} & \frac{1}{4} & \frac{3}{16} \\ \frac{5}{32} & \frac{11}{32} & \frac{1}{4} & \frac{3}{16} & \frac{1}{16} \\ \frac{7}{32} & \frac{1}{4} & \frac{9}{32} & \frac{3}{16} & \frac{1}{16} \\ \frac{5}{32} & \frac{5}{16} & \frac{3}{16} & \frac{7}{32} & \frac{1}{8} \\ \frac{3}{32} & \frac{3}{8} & \frac{1}{8} & \frac{1}{4} & \frac{5}{32} \end{pmatrix},$$

the machine $\mathfrak{R}$ is regular. The system of linear equations $\boldsymbol{dT} = \boldsymbol{d}$ is

$$d_1 = 2d_0\,, \quad 2d_0 + d_3 + d_4 = 2d_1\,, \quad d_1 + d_4 = 2d_2\,, \quad d_2 + d_3 = 2d_3\,, \quad d_2 = 2d_4\,.$$

Solving this system we get

$$d_0 = \frac{3}{19}, \quad d_1 = \frac{6}{19}, \quad d_2 = \frac{4}{19}, \quad d_3 = \frac{4}{19}, \quad d_4 = \frac{2}{19}.$$

From (3.2) we get

$$\delta'(\mathfrak{M}) = \frac{1}{2}d_1 + \frac{1}{2}d_3 + d_4 = \frac{7}{19}.$$

and this yields the lower bound $\gamma_2 \geq \frac{14}{19} = 0.736842\ldots$. ◇

As we have seen in Example 3.2 we can use symmetries to decrease the size of css machines. There is one symmetry between the top and bottom sequences of states of machine. Another symmetry can be obtained by the substitution of symbols according to some permutation $\pi : \Sigma \to \Sigma$ as specified by Lemma 2.1. Therefore we can split states of sequences into equivalence classes, where two states are equivalent if we can get one from the other by permutation of symbols and by exchanging the top and bottom sequence of the state. We shall denote all states equivalent to state $\binom{u}{v}$ by $\left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right]$.

3.3 Design of css machines

For automated production of lower bounds we need stronger constraints on css machines. Every new condition is natural and efficient css machines are likely to satisfy it. We shall not allow sequences in any state to begin with the same symbol as this match should be taken out before getting to such a state. This corresponds to the first condition from Definition 3.7. If one of the sequences from a state is empty, we should read a symbol from the corresponding input tape, otherwise we could lose some matches (corresponds to conditions 2 and 3). If we move from a state of the form $\left[\begin{smallmatrix} a'u \\ b'v \end{smallmatrix}\right]$ to a 'smaller'

state, we could also lose some information (corresponds to conditions 4 and 5). States that are not reachable from the initial state can be deleted from the machine without any influence on the output of the machine (corresponds to condition 6). Thus we get the definition of 'strong' css machines.

Definition 3.7 Let $\mathfrak{M} = \langle S, \Sigma, I, T, O \rangle$ be a css machine such that for every $s = \begin{bmatrix} u \\ v \end{bmatrix} \in S$

1. $u(1) \neq v(1)$,
2. if $I(s) = \uparrow$ and $u \neq \lambda$ then $v \neq \lambda$,
3. if $I(s) = \downarrow$ and $v \neq \lambda$ then $u \neq \lambda$,
4. if $u = a^i$, $v = b^j$, $a > b$, then $I(s) = \uparrow$ and ($T(s,a) = \begin{bmatrix} a^{i+1} \\ b^j \end{bmatrix}$ or $T(s,a) = s$),
5. if $u = a^i$, $v \neq b^j$, $I(s) = \uparrow$, and $T(s,b) = \begin{bmatrix} a^l \\ v \end{bmatrix}$ then $l = i + 1$.
6. s is reachable from $\begin{bmatrix} \lambda \\ \lambda \end{bmatrix}$.

Such a machine $\mathfrak{M}$ is called a *strong* css machine.

Strong css machines automatically satisfy Lemma 3.3.

Lemma 3.5 *Every strong css machine is regular.*

Proof. Let $\mathfrak{M} = \langle S, \Sigma, I, T, O \rangle$ be a strong css machine. Let us consider states $t_i = T^i(s, 0^l, 0^l)$ for $i = 0, \ldots, l$, where $l = |S|$ is the number of states of machine $\mathfrak{M}$. Let t be the state with a repeated occurrence among $t_0, \ldots, t_l$. Since we are discarding symbols from the left and adding ones from the right, the state t has to have the form $t = \begin{bmatrix} 0^j \\ \lambda \end{bmatrix}$. Since we have to read from the

tape corresponding to the empty sequence, there is some state $\begin{bmatrix}\lambda\\ \lambda\end{bmatrix}$ between two occurrences of state t.

Let s be a state of the form $\begin{bmatrix}a^i\\ b^j\end{bmatrix}$ with maximal $i+j$. Then according to conditions 4 and 5 from Definition 3.7, $T(s,c)$ must be s, where c is a or b depending on $I(s)$.

Hence from Lemma 3.3 we have that $\mathfrak{M}$ is regular. □

The structure of strong css machines can be used for their automated production. An algorithm $\mathfrak{E}$ for such production can be found in Figure 3.3.

The basis of this algorithm is the function add$(\mathfrak{M}, s)$. The input of the function 'add' is a machine $\mathfrak{M}$ and a pair of sequences s. The output of the function 'add' is an extension of the machine $\mathfrak{M}$ that contains the pair s as one of its states. If pair s is already a state of $\mathfrak{M}$ then the output is simply $\mathfrak{M}$. Otherwise for every symbol $a \in \Sigma$, a transition $T(s,a)$ is found and the value of the output function $O(s,a)$ is computed. If required, function 'add' is called recursively. Finally the pair s is added among the states of the $\mathfrak{M}$. The new transitions are built so that the new state s automatically satisfies conditions 2–5 from Definition 3.7 and every new state is reachable from s.

The performance of the algorithm $\mathfrak{E}$ is as follows. A nonsaturated transition that will be upgraded is selected. The new target of the transition is added to the machine $\mathfrak{M}$. This can lead to the situation where the old target of the transition become unreachable from the initial state. In such a case the old target is removed from the states of the $\mathfrak{M}$. The transition is then upgraded.

This algorithm preserves the properties of strong css machines.

Theorem 3.3 *Let procedure* cut(**in** u, v; **out** x, x', y, y') *be such that* $xx' = u$, $yy' = v$, $u(1) \neq v(1)$, *and* $xy \neq \lambda$. *Let* reduce$(\mathfrak{M})$ *be a machine identical with*

algorithm $\mathfrak{E}$

input regular css machine $\mathfrak{M} = \langle S, \Sigma, I, T, O\rangle$
output expanded css machine $\mathfrak{M}' = \mathfrak{E}(\mathfrak{M})$

function add(css machine $\mathfrak{M}$, pair $s = \binom{u}{v}$)
begin
if $[s] \in S$ **then**
return $\mathfrak{M}$;
if $|u| < |v|$ **then**
exchange(u, v);
for $a \in \Sigma$ **do**
if $\left[\begin{smallmatrix} u \\ va \end{smallmatrix}\right] \in S$ **then**
$T[s, a] := \left[\begin{smallmatrix} u \\ va \end{smallmatrix}\right]$;
$O[s, a] := 0$
else if $\mathbf{L}(u, va) = 0$ & $v = aa \ldots a$ **then**
$T[s, a] := s$;
$O[s, a] := 0$
else if $\exists x, x', y, y'$ $(u = yxx', va = xyy', |x| = |y| > 0, \mathbf{L}(x, y) = 0)$**then**
$\mathfrak{M} :=$ add$(\mathfrak{M}, \binom{xx'}{yy'})$;
$T[s, a] := \left[\begin{smallmatrix} xx' \\ yy' \end{smallmatrix}\right]$;
$O[s, a] := |x|$
else
cut(u, va, x, x', y, y');
$\mathfrak{M} :=$ add$(\mathfrak{M}, \binom{x'}{y'})$;
$T[s, a] := \left[\begin{smallmatrix} x' \\ y' \end{smallmatrix}\right]$;
$O[s, a] := \mathbf{L}(x, y)$
endif
endfor;
$S = S \cup [s]$;
$I[s] :=\downarrow$;
return $\mathfrak{M}$
end add;

begin
select $s = \binom{u}{v} \xrightarrow{a} T(s, a)$;
if $I[s] :=\uparrow$ **then**
$\mathfrak{M} :=$ add$(\mathfrak{M}, \binom{ua}{v})$
$T[s, a] := \left[\begin{smallmatrix} ua \\ v \end{smallmatrix}\right]$;
else
$\mathfrak{M} :=$ add$(\mathfrak{M}, \binom{u}{va})$
$T[s, a] := \left[\begin{smallmatrix} u \\ va \end{smallmatrix}\right]$;
endif;
reduce$(\mathfrak{M})$;
$O[s, a] := 0$
end.

Figure 3.3: Algorithm $\mathfrak{E}$ for expanding css machines.

$\mathfrak{M}$ except that all states not reachable from $\binom{\lambda}{\lambda}$ are deleted. If $\mathfrak{M}$ is a strong css machine, then $\mathfrak{E}(\mathfrak{M})$ is also a strong css machine.

Proof. The total length of the pair s does not increase in recursive calls of procedure 'add'. Therefore algorithm $\mathfrak{E}$ finishes after a finite number of steps.

Let $\mathfrak{M}$ be a strong css machine. Procedure 'add' preserves properties 1–6 from Definition 3.7. Newly added states satisfy conditions 2–5. The satisfiability of condition 1 is guaranteed by the properties of procedure 'cut'. Procedure 'reduce' causes the resulting machine to satisfy condition 6. Therefore $\mathfrak{E}(\mathfrak{M})$ is a strong css machine. □

Algorithm $\mathfrak{E}$ for extending strong css machines carries two heuristic features. The first one is selecting a transition that will be upgraded. The second one is the decision how to create transitions for newly added states. It is not difficult to create a sequence of strong css machines $\mathfrak{M}_1, \ldots$ over the alphabet of size k such that $\mathfrak{M}_{i+1} = \mathfrak{E}(\mathfrak{M}_i)$ and $2\delta'(\mathfrak{M}_i) \to \gamma_k$ as $i \to \infty$.

To achieve this we select a nonsaturated transition $s \xrightarrow{a} T(s,a)$ with minimal $|s|$ and set procedure $\mathrm{cut}(u, v; x, x', y, y')$ to return $x = u$, $x' = \lambda$, $y = v$, $y' = \lambda$. The machines $\mathfrak{M}$ can be seen as (almost) complete balanced k-ary trees. State $\binom{\lambda}{\lambda}$ corresponds to the root of the tree, saturated states are the internal vertices of the tree and nonsaturated vertices are the leaves of the tree. From every leaf there are nonsaturated transitions to the root. However, there are exceptions in this structure caused by part 3 Definition 3.3 and by Definition 3.7. We do not have transitions of the form $s \to \binom{xy}{yx}$, these transitions rather have the form $s \xrightarrow{+|x|} \binom{y}{x}$. Also transition $\binom{a}{\lambda\bullet} \xrightarrow{a} \binom{a}{a}$ will be rather $\binom{a}{\lambda\bullet} \xrightarrow{a+1} \binom{\lambda}{\lambda}$.

Although $\delta'(\mathfrak{M}_i)$ converges to γ_k, the convergence is slow. To get a faster convergence we shall change the heuristics. We shall describe an improved heuristic for the alphabet $\Sigma = \{0, 1\}$. When selecting the transition to be upgraded, we shall choose such a transition that is likely to make a large increment of the lower bound produced. There are two factors contributing to the decision of selecting which transition to upgrade. The first factor is the 'quality' of a transition and the second one is the 'quality' of a state.

For every transition we can estimate how many matches we lose when using this transition. For transition

$$s = \begin{bmatrix} u\bullet \\ v \end{bmatrix} \xrightarrow{a+O(s,a)} \begin{bmatrix} u' \\ v' \end{bmatrix} \tag{3.3}$$

this loss will occur when

$$\mathbf{L}(uax, vy) > \mathbf{L}(u'x, v'y) + O(s, a)$$

for some $x, y \in \Sigma^*$. A *failure ratio for depth* j for the transition (3.3) is then defined by

$$R_j(s, a) = \frac{1}{2^{2j}} |\{(x, y) \in \Sigma^j \times \Sigma^j : \mathbf{L}(uax, vy) > \mathbf{L}(u'x, v'y) + O(s, a)\}| .$$

From (3.2) we can see that the contribution of every state is proportional to the value of the stationary probability distribution in this state. Therefore when extending css machines we shall select the transition $s \xrightarrow{a} T(s, a)$ with the largest value of the product $d(s)R_j(s, a)$.

When adding a new symbol s to machine $\mathfrak{M}$ we have to built new transitions $s = \begin{bmatrix} u \\ v \end{bmatrix} \xrightarrow{a} T(s, a)$ for every $a \in \Sigma$. Some of these are determined by properties of strong css machines, but in the most cases we are not bound when choosing $T(s, a)$. We shall match as few symbols as possible. This will

```
procedure cut(u, v; x, x', y, y');
    input   u, v ∈ Σ*
    output x, x', y, y' ∈ Σ* such that xx' = u, yy' = v, u(1) ≠ v(1), and xy ≠ λ
begin
    t := find_first(u, v(1));
    if t < ∞ then
        y0 := common_prefix(u(t . . |u|), v);
        x0 := u(1 . . t − 1 + |y0|);
        y'0 := v(|y0| + 1 . . |v|);
        x'0 := u(t + |y0| . . |u|);
    endif
    b := find_first(v, u(1));
    if b < ∞ then
        x1 := common_prefix(u, v(b . . |v|));
        y1 := v(1 . . b − 1 + |x0|);
        x'1 := u(|x0| + 1 . . |u|);
        y'1 := v(b + |x0| . . |v|);
        if t < ∞ & δ'(add(M, (x'0, y'0))) > δ'(add(M, (x'1, y'1))) then
            return(x0, y0, x'0, y'0)
        endif
    endif
    return(x1, y1, x'1, y'1);
end;
```

Figure 3.4: Procedure 'cut' from algorithm $\mathfrak{E}$. Function find_first(u, a) returns the first position of symbol a in sequence u or ∞ if $a \not\sqsubseteq u$. Function common_prefix(u, v) returns the common prefix of sequences u and v.

allow us to make further matchings dependent on symbols not yet read. Since $k = 2$ and $u(1) \neq v(1)$, then either $u(1)$ or $v(1)$ must be involved in the next match. Out of two possible matches we shall choose the one which yields a better lower bound. The corresponding piece of code can be found in Figure 3.4.

We have used these heuristics to produce larger and larger css machines and gain better and better lower bounds for γ_2. The actual depth used for computing the failure ratio was $j = 6$. The best machine we have created has

931 states and gives us a lower bound of $0.773911 \leq \gamma_2$. It can be found in Appendix A. If $I(s,a)$ is nonzero, it appears in column $T(s,a)$ as a number after the name of the state $T(s,a)$. In the column $d(s)$ is the value of the stationary probability distribution of the state multiplied by 10^3.

3.4 Labeled css machines

Deken [Dek79] describes and analyses algorithms for obtaining common subsequences. The best of these algorithms is given in Figure 3.5. He also observed that describing this algorithm by finite css machine requires a large number of states. However, we shall introduce a variant of css machines, that allows us to describe Deken's algorithm by a machine with a small number of states.

The basic idea of Deken's algorithm is as follows. We have two markers, one for each tape. The algorithm starts with markers marking the beginning of the input tapes. In one step the algorithm reads symbols from one of the input tapes until a new symbol, with respect to symbols between the marker and input head, is read. If this symbol matches a symbol on the other tape (again in the area between the marker and the input head), we copy the symbol to the output, move the markers to the matched symbols and new scanning starts from the markers. We continue in this manner and alternate the tape from which the input is read. This is illustrated in Figure 3.6.

The analysis of the algorithm is based on the fact that the numbers of symbols scanned between matches are independent. Therefore if **EN** is the expected number of symbols scanned between two matches, then $\gamma_k \geq 2/\mathbf{EN}$.

Example 3.3 State diagram $\mathfrak{D}$ corresponding to Deken's algorithm for al-

```
algorithm 𝔇
    input   u, v ∈ Σ*
    output w ∈ Σ* – common subsequence of u, v and css = |w|
begin
    n_t := 1;
    n_b := 1;
    css := 0;
    TopPos[1, ..., k] := 0;
    BotPos[1, ..., k] := 0;
    loop
        while TopPos[u(n_t)] ≠ 0 do
            n_t := n_t + 1;
            if n_t > |u| then
                return css
            endif
        endwhile;
        TopPos[u(n_t)] := n_t;
        if BotPos[u(n_t)] ≠ 0 then
            css := css + 1;
            output u(n_t);
            n_b := BotPos[u(n_t)] + 1;
            TopPos[1, ..., k] := 0;
            BotPos[1, ..., k] := 0
        endif;
        n_t := n_t + 1;
        while BotPos[v(n_b)] ≠ 0 do
            n_b := n_b + 1;
            if n_b > |v| then
                return css
            endif
        endwhile
        BotPos[v(n_b)] := n_b;
        if TopPos[v(n_b)] ≠ 0 then
            css := css + 1;
            output v(n_b);
            n_t := TopPos[v(n_b)] + 1;
            TopPos[1, ..., k] := 0;
            BotPos[1, ..., k] := 0
        endif;
        n_b := n_b + 1
    endloop
end.
```

Figure 3.5: Deken's matching algorithm.

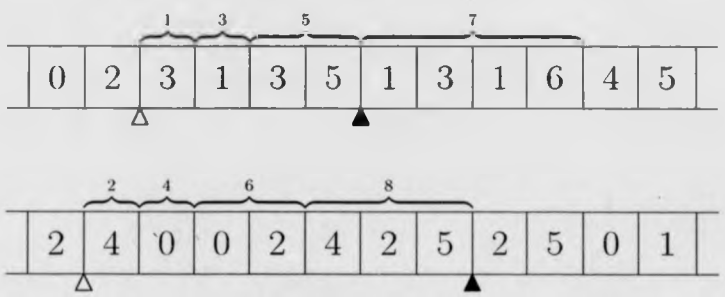

Figure 3.6: One cycle of Deken's algorithm.

phabet $\{0,1,2,3\}$ can be found in Figure 3.7. There are five states $\left[\begin{smallmatrix}\lambda\bullet\\ \lambda\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}0\\ \lambda\bullet\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}0\bullet\\ 1\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}02\\ 1\bullet\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}02\bullet\\ 13\end{smallmatrix}\right]$ with $\bullet$ signalling where the next symbol will be read. One step of Deken's algorithm corresponds to a movement from one state to a different state. Finding a match corresponds to a transition leading to $\left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]$ and one cycle of the algorithm corresponds to a path from $\left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]$ back to $\left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]$. We shall analyse machine $\mathfrak{D}$ in two steps. First we shall not consider loops and multiple transitions will be seen as one transition.

Let $P(s,a)$ be a probability of a transition $s \xrightarrow{a} T(s,a)$ where $T(s,a) \neq s$. We have

$$P(s,a) = \begin{cases} \text{undefined} & \text{if } T(s,a) \neq s\,, \\ \frac{1}{4} & \text{for } s = \left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right], \left[\begin{smallmatrix}0\\ \lambda\end{smallmatrix}\right], \\ \frac{1}{3} & \text{for } s = \left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right], \left[\begin{smallmatrix}02\\ 1\end{smallmatrix}\right], \\ \frac{1}{2} & \text{for } s = \left[\begin{smallmatrix}02\\ 13\end{smallmatrix}\right]. \end{cases}$$

For every state $s \neq \left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]$ there is a unique path from $\left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]$ to s. Let $U(s)$ be the probability of moving along this path and let $p(s,a)$ be the probability of moving from $\left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]$ to s and then to $T(s,a) = \left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]$. It is $p(s,a) = P(s,a)U(s)$

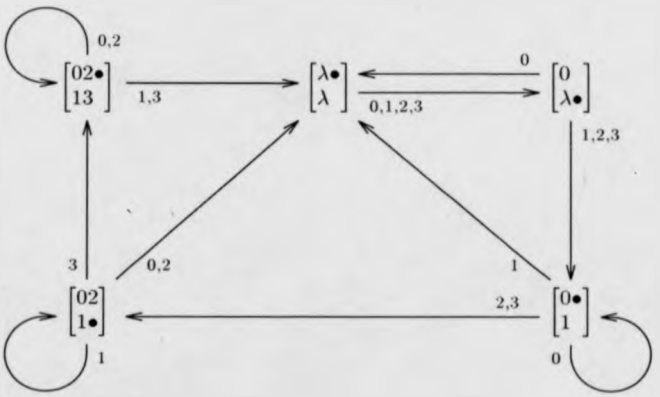

Figure 3.7: Labeled css machine for $k = 4$.

and

$$\begin{aligned}
U\left(\left[\begin{smallmatrix}0\\ \lambda\end{smallmatrix}\right]\right) &= 1 & p\left(\left[\begin{smallmatrix}0\\ \lambda\end{smallmatrix}\right], 0\right) &= \tfrac{1}{4}, \\
U\left(\left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right]\right) &= \tfrac{3}{4} & p\left(\left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right], 1\right) &= \tfrac{1}{3}\tfrac{3}{4} = \tfrac{1}{4}, \\
U\left(\left[\begin{smallmatrix}02\\ 1\end{smallmatrix}\right]\right) &= \tfrac{1}{2} & p\left(\left[\begin{smallmatrix}02\\ 1\end{smallmatrix}\right], 0\right) &= p\left(\left[\begin{smallmatrix}02\\ 1\end{smallmatrix}\right], 2\right) = \tfrac{1}{3}\tfrac{1}{2} = \tfrac{1}{6}, \\
U\left(\left[\begin{smallmatrix}02\\ 13\end{smallmatrix}\right]\right) &= \tfrac{1}{6} & p\left(\left[\begin{smallmatrix}02\\ 13\end{smallmatrix}\right], 1\right) &= p\left(\left[\begin{smallmatrix}02\\ 13\end{smallmatrix}\right], 3\right) = \tfrac{1}{2}\tfrac{1}{6} = \tfrac{1}{12}.
\end{aligned}$$

We put $p(s,a) = 0$ if $T(s,a) \neq s_0$, i.e., transition $s \xrightarrow{a} T(s,a)$ does not correspond to a match. Thus $p(s,a)$ is the probability distribution of matches and $\sum p(s,a) = 1$.

Let $q(s,a)$ be the mean number of symbols read while moving along the path $\left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right] \to \cdots \to s \xrightarrow{a} \left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]$. The $q(s,a)$ would be the length of the path, except that we have to take the loops into account. Therefore $q(s,a)$ will be the number of steps plus the expected number of moves along the loops in every state. The average progress on the loops $\left[\begin{smallmatrix}0\bullet\\ 1\end{smallmatrix}\right] \xrightarrow{0} \left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}02\\ 1\bullet\end{smallmatrix}\right] \xrightarrow{1} \left[\begin{smallmatrix}02\\ 1\end{smallmatrix}\right]$ is $\frac{1}{4} + \left(\frac{1}{4}\right)^2 + \cdots = \frac{1}{3}$, and for the loops $\left[\begin{smallmatrix}02\bullet\\ 13\end{smallmatrix}\right] \xrightarrow{0} \left[\begin{smallmatrix}02\\ 13\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}02\bullet\\ 13\end{smallmatrix}\right] \xrightarrow{2} \left[\begin{smallmatrix}02\\ 13\end{smallmatrix}\right]$ it is

$\frac{1}{2} + \left(\frac{1}{2}\right)^2 + \cdots = 1$. Therefore the mean progress to collect the first symbol on one tape is $W(1) = 1$, for the second symbol it is $W(2) = 2\frac{1}{3}$, and for the third symbol it is $W(3) = 4\frac{1}{3}$. For $q(s,a)$ we now have

$$\begin{array}{llll} q\left(\left[\begin{smallmatrix}0\\ \lambda\bullet\end{smallmatrix}\right],0\right) = W(1)+W(1) = 2 & & q\left(\left[\begin{smallmatrix}0\bullet\\ 1\end{smallmatrix}\right],1\right) = W(2)+W(1) = 3\frac{1}{3} \\ q\left(\left[\begin{smallmatrix}02\\ 1\bullet\end{smallmatrix}\right],0\right) = W(1)+W(2) = 3\frac{1}{3} & & q\left(\left[\begin{smallmatrix}02\\ 1\bullet\end{smallmatrix}\right],2\right) = W(2)+W(2) = 4\frac{2}{3} \\ q\left(\left[\begin{smallmatrix}02\bullet\\ 13\end{smallmatrix}\right],1\right) = W(3)+W(1) = 5\frac{1}{3} & & q\left(\left[\begin{smallmatrix}02\bullet\\ 13\end{smallmatrix}\right],3\right) = W(3)+W(2) = 6\frac{2}{3} \end{array}$$

Having found $p(s,a)$ and $q(s,a)$, we can now compute the expected number of symbols read between two matches $\mathbf{EN} = \sum_{s\in S}\sum_{a\in\Sigma} p(s,a)\,q(s,a) = \frac{11}{3}$. This yields the lower bound $\frac{6}{11} = 0.545454\ldots$. ◇

We can build a css machine $\mathfrak{M}$ that simulates the behaviour of Deken's algorithm for any alphabet $\Sigma = \{0,\ldots,k-1\}$. Because of the symmetries (Lemma 2.1, part 3) we can suppose that symbols read on the top tape are even and in the order 0,2,... and symbols on the bottom tape are odd and in the order 1,3,.... Machine $\mathfrak{M}$ will have $k+1$ states $s_0 = \left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]$, $s_1 = \left[\begin{smallmatrix}0\\ \lambda\end{smallmatrix}\right]$, $s_2 = \left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right]$, ..., $s_{2i} = \left[\begin{smallmatrix}0\,2\,\ldots\,2i-2\\ 1\,3\,\ldots\,2i-1\end{smallmatrix}\right]$, $s_{2i+1} = \left[\begin{smallmatrix}0\,2\,\ldots\,2i-2\,2i\\ 1\,3\,\ldots\,2i-1\end{smallmatrix}\right]$, ... For the tape switch function of the machine $\mathfrak{M}$ we have $I(s_{2i}) = \uparrow$ and $I(s_{2i+1}) = \downarrow$. The appropriate transition function is given by

$$T(s_{2i},a) = \begin{cases} s_0 & \text{for } a = 1,3,\ldots,2i-1 \\ s_{2i} & \text{for } a = 0,2,\ldots,2i-2 \\ s_{2i+1} & \text{for } a \geq 2i \end{cases}$$

$$T(s_{2i+1},a) = \begin{cases} s_0 & \text{for } a = 0,2,\ldots,2i \\ s_{2i+1} & \text{for } a = 1,3,\ldots,2i-1 \\ s_{2i+2} & \text{for } a \geq 2i+1 \end{cases}$$

For the output function, we have

$$O(s,a) = \begin{cases} 1 & \text{if } T(s,a) = s_0 \\ 0 & \text{otherwise} \end{cases}$$

To compute **EN**, we shall introduce a value function $V : S \times \Sigma \to \mathbb{R}$. If $T(s,a) = s_0$, we define $V(s,a) = p(s,a)q(s,a)$, where $p(s,a)$ is the probability of matching using the transition $s \xrightarrow{a} s_0$ and $q(s,a)$ is the expected number of symbols read to achieve this transition. If $T(s,a) \neq s_0$, we put $V(s,a) = 0$. Clearly

$$\mathbf{EN} = \sum_{s \in S} \sum_{a \in \Sigma} V(s,a)\,.$$

To evaluate $V(s,a)$ we shall introduce the functions $U(j)$ and $W(j)$. $U(j)$ is the probability of achieving the j-th state of the machine $\mathfrak{M}$ and $W(j)$ is the expected length of a sequence read until j different symbols appear in the sequence. For $U(1)$ we have $U(1) = 1$. To express $U(i+1)$ in the terms of $U(i)$ we can observe that for $k - \lfloor i/2 \rfloor$ symbols the transition leads out of the state s_i. In $k - i$ cases this is transition $s_i \xrightarrow{a} s_{i+1}$ and in $\lceil i/2 \rceil$ cases it is transition $s_i \xrightarrow{a} s_0$. Therefore $U(i+1) = U(i)\frac{k-i}{k-\lfloor i/2 \rfloor}$. This gives us

$$U(i) = \prod_{l=0}^{i-1} \frac{k-l}{k-\lfloor l/2 \rfloor}$$

and

$$p(s_i,a) = \frac{1}{k-\lfloor l/2 \rfloor} \prod_{l=0}^{i-1} \frac{k-l}{k-\lfloor l/2 \rfloor}\,,$$

where a is such that $T(s,a) = s_0$.

Since the waiting time for the $(j+1)$-th symbol, having already collected j symbols, is $1 + \frac{j}{k} + (\frac{j}{k})^2 + \ldots = \frac{k}{k-j}$, we have

$$W(j) = \sum_{l=0}^{j-1} \frac{k}{k-l}\,.$$

Alphabet size	Lower bound	Alphabet size	Lower bound	Alphabet size	Lower bound
2	0.77391	7	0.44502	12	0.35899
3	0.61538	8	0.42237	13	0.34737
4	0.54545	9	0.40321	14	0.33687
5	0.50615	10	0.38656	15	0.32732
6	0.47169	11	0.37196		

Figure 3.8: Lower bounds for γ_k.

Therefore

$$q(s_i, a) = W(\lfloor i/2 \rfloor) + W(\lfloor a/2 \rfloor),$$

where again a is such that $T(s, a) = s_0$.

All this together gives us the following lower bound:

Theorem 3.4 *For every $k \geq 2$ we have*

$$\gamma_k \geq \frac{2}{\sum_{i=1}^{k} \sum_{j=0}^{\lfloor (i-1)/2 \rfloor} \left(\sum_{l=0}^{\lfloor i/2 \rfloor} \frac{k}{k-l} + \sum_{l=0}^{j} \frac{k}{k-l} \right) \frac{1}{k - \lfloor i/2 \rfloor} \prod_{l=1}^{i-1} \frac{k-l}{k - \lfloor l/2 \rfloor}}$$

Actual lower bounds for $k = 2, \ldots, 15$ can be found in Figure 3.8

Functions $U(j)$ and $W(j)$ can be seen as labels on the states and functions $p(s, a)$ and $q(s, a)$ can be seen as labels on the transitions. So machines described in this section can be called *labeled* css machines.

So far, all attempts to push this approach further break down on the independence condition for numbers collected between matches.

Chapter 4

Upper bounds

In this chapter we shall describe a new method for obtaining upper bounds for the expected length of a longest common subsequence. First we shall outline the method and define collations — pairs of sequences with marked matches. Then we shall describe simpler versions of the new method and finally full versions of the method will be given.

4.1 Collations

Let $\mathcal{F}(i,n)$ be the set of all pairs of sequences, both of length n, with a longest common subsequence of length i, i.e.

$$\mathcal{F}(i,n) = \{u, v \in \Sigma^n : \mathbf{L}(u,v) = i\}.$$

The number of elements in set $\mathcal{F}(i,n)$ will be denoted by $F(i,n)$. Since $\bigcup_{i=0}^{n} \mathcal{F}(i,n) = \Sigma^n \times \Sigma^n$, we have

$$\sum_{i=0}^{n} F(i,n) = k^{2n}. \tag{4.1}$$

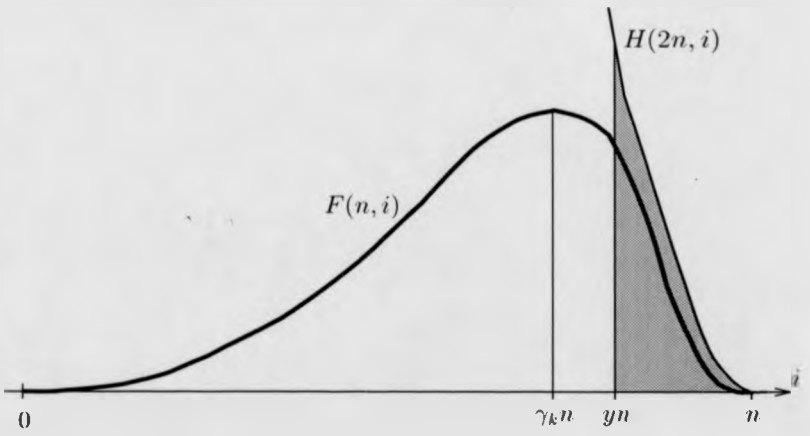

Figure 4.1: Upper bounds for the expected length of a longest common supersequence.

This allows us to rewrite the definition of the expected length of a longest common subsequence in the following way:

$$\mathbf{EL}_n = \frac{1}{k^{2n}} \sum_{i=0}^{n} iF(i,n)\,. \tag{4.2}$$

Let $H(i,n)$ be any upper bound for $F(i,n)$. If y is such that $\sum_{i=\lceil yn \rceil}^{n} H(i,n)$ is very small compared with k^{2n}, then pairs with a longest common subsequence larger than yn do not really contribute to the average and therefore the expected length must be smaller than yn.

This is the basic idea of the first upper bound given by Chvátal and Sankoff [CS75] and also the basic idea of all further upper bounds. It is illustrated in Figure 4.1 and formally proved in Lemma 4.1.

Lemma 4.1 (Chvátal and Sankoff [CS75]) *Let $F(i,n)$ be the number of pairs of strings of length n over the alphabet of size k with a longest common*

subsequence of length i. Let $H(i,n)$ be an upper bound for $F(i,n)$, i.e., $F(i,n) \le H(i,n)$ for all i,n. If y is such that

$$\sum_{i=\lceil yn \rceil}^{n} H(i,n) = o(k^{2n}) \tag{4.3}$$

then $\gamma_k \le y$.

Proof. We can split the summation in (4.2) into two parts, from 0 to yn and from yn to n. We get

$$\frac{\mathbf{EL}_n}{n} = \frac{1}{nk^{2n}} \left(\sum_{i=0}^{\lceil yn \rceil - 1} iF(i,n) + \sum_{i=\lceil yn \rceil}^{n} iF(i,n) \right).$$

For the first sum we have $i \le \lceil yn \rceil - 1 \le yn$ and for the second sum we have $i \le n$. Hence

$$\frac{\mathbf{EL}_n}{n} \le \frac{1}{nk^{2n}} \left(yn \sum_{i=0}^{n} F(i,n) + n \sum_{i=\lceil yn \rceil}^{n} H(i,n) \right).$$

We can use (4.1) to evaluate the first sum and (4.3) to estimate the second one. We get

$$\frac{\mathbf{EL}_n}{n} \le \frac{1}{nk^{2n}} \left(ynk^{2n} + no(k^{2n}) \right) = y + o(1).$$

For $n \to \infty$ this gives $\gamma_k = \lim \frac{\mathbf{EL}_n}{n} \le y$. □

To obtain upper bounds for $F(i,n)$ we shall concentrate on the relation between $F(i,n)$ and $F(i+1,n)$. We shall try to observe the changes of $\mathcal{F}(i,n)$ when increasing i. To realize it we introduce the notion of a match – a pair of sequences that have the same last symbol, and a collation – a pair of sequences with a marked common subsequence.

Definition 4.1 Let $u = u_1 \cdots u_m$ and $v = v_1 \cdots v_n$ be two sequences. We say that pair $\binom{u}{v}$ is a *match*, if $u_m = v_n$. The set of all matches will be denoted by Δ. For match $\binom{u}{v} \in \Delta$ we denote $\left\lfloor \binom{u}{v} \right\rceil = \binom{v}{u}$.

Given two sequences u and v and their common subsequence w, we can chop u and v just behind the corresponding symbols from w. Every piece we get (except the last one) is a match and such a chopped pair of sequences will be called a 'collation'. The sum of the positions of $w(i)$ in the input sequences u and v is the i-th entry of a 'collation key'.

Definition 4.2 A sequence $\mathbf{p} = p_1, p_2, \ldots, p_n, p_{n+1}$ of pairs is called a *collation of order* n if the pairs $p_1, p_2, \ldots, p_n$ are matches. We say that collation $p_1, p_2, \ldots, p_n, p_{n+1}$ *generates* the pair $\binom{u}{v}$ if $\mathrm{cat}(p_1, p_2, \ldots, p_n, p_{n+1}) = \binom{u}{v}$. The positions of pairs of matching symbols form the *collation key* $L(\mathbf{p})$, where $L(\mathbf{p}) = l(p_1), l(p_1, p_2), \ldots, l(p_1, p_2, \ldots, p_n)$.

We shall use $L^R(\mathbf{p})$ to denote the reverse of collation key $L(\mathbf{p})$. Let '$\prec$' be the lexicographical order on integer sequences. We shall introduce the notion of dominance. Informally, having two ways of cutting the sequences we shall prefer it when the cuts appear as soon as possible. We shall say that a collation is 'dominated' when symbols in the collation are not matched optimally.

Definition 4.3 We say that collation $\mathbf{p}$ *dominates* collation $\mathbf{q}$, if $\mathbf{p}$ and $\mathbf{q}$ have the same order $n \geq 1$, generate the same pair, and $L^R(\mathbf{p}) \prec L^R(\mathbf{q})$. Collation $\mathbf{q} = q_1, \ldots, q_{n+1}$ of order $n \geq 0$ is *dominated* if there exists a collation $\mathbf{p}$ such that $\mathbf{p}$ dominates $\mathbf{q}$ or if $\mathbf{L}(q_{n+1}) > 0$.

The following example illustrates the definitions.

Example 4.1 Let $\Sigma = \{0, 1\}$. Pairs $\binom{0}{0}$, $\binom{01}{1}$, $\binom{10}{00}$ are matches, pairs $\binom{00}{11}$ and $\binom{10}{01}$ are not. Collation $\binom{01}{1}\binom{1}{01}\binom{\lambda}{\lambda}$ generates the pair $\binom{011}{101}$ and its collation key is $3, 6$. Another collation generating pair $\binom{011}{101}$ is $\binom{0}{10}\binom{1}{1}\binom{1}{\lambda}$ and its collation key is $3, 5$. When we compare these collation keys, we observe that $\binom{0}{10}\binom{1}{1}\binom{1}{\lambda}$ dominates $\binom{01}{1}\binom{1}{01}\binom{\lambda}{\lambda}$. Thus $\binom{01}{1}\binom{1}{01}\binom{\lambda}{\lambda}$ is a dominated collation. ◇

If a collation is dominated, it remains dominated after any permutation of symbols from Σ. There is another symmetry between the top and bottom sequences of a pair. Therefore we can split pairs of sequences into equivalence classes, where two pairs p and q are equivalent if we can get q by permutation of the symbols in p or $\lfloor p \rfloor$. We shall denote all pairs equivalent to pair $\binom{u}{v}$ as $\left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right]$. These equivalences (and notations) are naturally extended to matches, collations and sets of collations. Reversing is a third symmetry, that works for longest common subsequences, however this symmetry does not make sense for matches or collations.

Let $\mathcal{C}(m)$ be the set of all collations generating a pair of total length m. The set of all collations will be denoted by $\mathcal{C} = \bigcup_{m=0}^{\infty} \mathcal{C}(m)$. Let $\mathcal{N}(i)$ be the set of all nondominated collations of order i. Let $\mathcal{N}(i, m) = \mathcal{N}(i) \cap \mathcal{C}(m)$ be the set of all nondominated collations of order i generating a pair of total length m. These sets of collations can serve us as upper bounds for $F(i, n)$.

Lemma 4.2 *For every $i, n \in \mathbb{N}$*

$$F(i, n) \leq |\mathcal{N}(i, 2n)|$$

Proof. This lemma is a simple consequence of a fact that for every pair $\binom{u}{v}$ with longest common subsequence of length i there is at least one nondominated collation of order i generating pair $\binom{u}{v}$. □

We are still unable to count the number of elements in $\mathcal{F}(i,n)$ or in $\mathcal{N}(i,n)$. But we can define sets of collations $\mathcal{H}(i,n)$ which contain all nondominated collations and are easier to handle.

Suppose we have expressed the number of elements in $\mathcal{H}(i,n)$ in terms of the generating functions $\sum_{n=0}^{\infty} |\mathcal{H}(i,n)|z^n$. Theorem 4.1 gives us conditions when we can transform upper bounds for these generating functions to upper bounds for γ_k.

Theorem 4.1 (Dančík and Paterson [DP94]) *Let $\mathcal{H}(i)$, $i \geq 0$, be sets of collations of order i such that every nondominated collation of order i is in $\mathcal{H}(i)$. Let $h^{(i)}(z) = \sum_{m=0}^{\infty} |\mathcal{H}(i) \cap \mathcal{C}(m)|z^m$ be the generating functions for $\mathcal{H}(i,m) = \mathcal{H}(i) \cap \mathcal{C}(m)$. Suppose the $h^{(i)}(z)$ satisfy*

$$h^{(i)}(z) \leq p(z)q(i)(\lambda(z))^i \tag{4.4}$$

where $p(z)$ and $\lambda(z)$ are functions independent of i and $q(i)$ is a nondecreasing polynomial. If $z_0 \in (0, \frac{1}{k-1})$ is such that $\lambda(z_0) < 1$ then

$$\gamma_k \leq \frac{2 \log k z_0}{\log \lambda(z_0)}.$$

Proof. We shall denote $|\mathcal{H}(i) \cap \mathcal{C}(m)|$ by $H(i,m)$. The set $\mathcal{N}(i,m)$ is a subset of the set $\mathcal{H}(i) \cap \mathcal{C}(m)$ and therefore $H(i,2n)$ is an upper bound for $F(i,n)$ according to Lemma 4.2. Let Z be the set of all $z \in (0, \frac{1}{k-1})$ such that $\lambda(z) < 1$. For every $y > \frac{2 \log k z_0}{\log \lambda(z_0)} > 0$ from (2.1) we get

$$\sum_{i=\lceil yn \rceil}^{n} H(i,2n) \leq \sum_{i=\lceil yn \rceil}^{n} \inf_{z \in Z} \frac{h^{(i)}(z)}{z^{2n}}.$$

From (4.4) and from the fact that $\sum \inf \leq \inf \sum$ we have

$$\sum_{i=\lceil yn \rceil}^{n} H(i,2n) \leq \inf_{z \in Z} \sum_{i=\lceil yn \rceil}^{n} \frac{p(z)q(i)(\lambda(z))^i}{z^{2n}}.$$

Since $\lambda(z) < 1$, we get $(\lambda(z))^i \leq (\lambda(z))^{yn}$ for $i \geq yn$. Hence

$$\sum_{i=\lceil yn \rceil}^{n} H(i,2n) \leq \inf_{z \in Z} \frac{p(z)nq(n)(\lambda(z))^{yn}}{z^{2n}}$$

and from the properties of infimum for z_0 we get

$$\sum_{i=\lceil yn \rceil}^{n} H(i,2n) \leq p(z_0)nq(n) \left(\frac{(\lambda(z_0))^y}{z_0^2} \right)^n .$$

For y we have $(\lambda(z_0))^y < k^2 z_0^2$ and therefore $\sum_{i=\lceil yn \rceil}^{n} H(i,2n) = o(k^{2n})$. Finally Lemma 4.1 gives us $\gamma_k \leq y$ for every $y > \frac{2 \log k z_0}{\log \lambda(z_0)}$. □

The numerical computations are performed with the help of Mathematica. The actual command used to compute the upper bound y is

```
FindMinimum[2*Log[k*z]/Log[lambda[k,z]],{z,0.5/k}]
```

4.2 Previous upper bounds

In this section we shall show how older upper bounds for the expected length of a longest common subsequence fit into the new framework given by Theorem 4.1. These upper bounds for alphabet size $k = 2, \ldots, 15$ together with upper bounds developed later in this chapter are given by the table in Figure 4.2.

The first upper bound due to Chvátal and Sankoff [CS75] corresponds to sets of all collations $\mathcal{H}(i)$ of the form

$$\binom{\Sigma^* a_1}{\Sigma^* a_1} \binom{\Sigma^* a_2}{\Sigma^* a_2} \cdots \binom{\Sigma^* a_i}{\Sigma^* a_i} \binom{\Gamma^*}{(\Sigma \setminus \Gamma)^*}$$

for some $\Gamma \subseteq \Sigma$. The generating function for the number of sequences of the form ua, for some $u \in \Sigma^*$, is $\frac{z}{1-kz}$, having k choices for a we get the generating

k	New results	Deken [Dek83]	Chvátal–Sankoff [CS83]	Chvátal–Sankoff [CS75]
2	0.83763	0.85750	0.86660	0.90512
3	0.76581	0.77682	0.78648	0.82999
4	0.70824	0.71810	0.72971	0.77291
5	0.66443	0.67323	0.68612	0.72767
6	0.62932	0.63721	0.65099	0.69056
7	0.60019	0.60731	0.62172	0.65932
8	0.57541	0.58189	0.59676	0.63250
9	0.55394	0.55987	0.57508	0.60909
10	0.53486	0.54052	0.55598	0.58841
11	0.51785	0.52331	0.53895	0.56995
12	0.50260	0.50786	0.52363	0.55331
13	0.48880	0.49387	0.50973	0.53820
14	0.47620	0.48112	0.49704	0.52440
15	0.46462	0.46942	0.48538	0.51172

Figure 4.2: Upper bounds for alphabet size $k = 2, \ldots, 15$.

function for the number of matches $\lambda(z) = \frac{kz^2}{(1-kz)^2}$. The generating function for the number of collations in $\mathcal{H}(i)$ then is $(\lambda(z))^t p(z)$, where $p(z)$ is the generating function for the number of matches of form $\binom{\Gamma^*}{(\Sigma \setminus \Gamma)^*}$. We then can use Theorem 4.1 to produce upper bounds. Analysing these upper bound for $k \to \infty$ we get the following theorem.

Theorem 4.2

$$\lim_{k \to \infty} \gamma_k \sqrt{k} \leq e$$

Proof. This is a consequence of the more general Theorem 5.2. □

Chvátal and Sankoff [CS83] have observed that if a is the matching symbol, then Σ^* can be replaced by $(\Sigma \setminus a)^*$. They have improved their upper bounds

using collations of the form

$$\binom{(\Sigma \setminus a_1)^* a_1}{(\Sigma \setminus a_1)^* a_1} \binom{(\Sigma \setminus a_2)^* a_2}{(\Sigma \setminus a_2)^* a_2} \cdots \binom{(\Sigma \setminus a_i)^* a_i}{(\Sigma \setminus a_i)^* a_i} \binom{\Gamma^*}{(\Sigma \setminus \Gamma)^*} .$$

The generating function for the number of matches of this new form then is $\lambda(z) = \frac{kz^2}{(1-(k-1)z)^2}$.

Deken [Dek83] goes further in the elimination of dominated collations. His approach involves 'minimal' matches – matches with only one possibility of matching symbols.

Definition 4.4 The match p is *minimal* if there is no match q such that $p = \mathrm{cat}(q, r)$ for some $r \in \Pi$ with $l(r) > 0$.

For example, in the binary alphabet the only minimal matches are in $\binom{0^*1}{1}$, $\binom{1}{0^*1}$, $\binom{1^*0}{0}$, and $\binom{0}{1^*0}$. For alphabet $\{0,1,2,3\}$, matches $\binom{332}{012}$ and $\binom{0123}{3}$ are minimal while $\binom{03}{1323}$ and $\binom{012}{132}$ are not. The usefulness of minimal matches is supported by the following lemma.

Lemma 4.3 *All matches in a nondominated collation are minimal.*

Proof. Let $\mathbf{p} = p_1, \ldots, p_{n+1}$ be a collation such that p_i is not a minimal match. We have $p_i = qr$ where q is a match and $l(r) > 0$. Collation $\mathbf{p}$ is then dominated by collation $p_1, \ldots, p_{i-1}, q, rp_{i+1}, \ldots, p_{n+1}$. □

Now we set $\mathcal{H}(i)$ to be the set of all collations consisting of minimal matches. To evaluate the generating function for the number of collations in $\mathcal{H}(i)$ we shall need the generating function $p(d, z)$ for the number of sequences using exactly d given symbols.

For every sequence using exactly d symbols we can mark the first occurrence of the last used symbol. We have d choices for the symbol and its contribution to generating function is dz. The sequence before the marked symbol (exclusive) uses exactly $d-1$ symbols and the generating function for the number of these is $p(d-1,z)$. After the marked symbol can be any sequence and the corresponding generating function is $\frac{1}{1-dz}$. For $p(d,z)$ we then get

$$p(d,z) = \frac{dz}{1-dz}\, p(d-1,z)$$

which leads to

$$p(d,z) = \prod_{j=1}^{d} \frac{jz}{1-jz}. \tag{4.5}$$

Every minimal match has the form $\binom{ua}{va}$ where $a \in \Sigma$, $u, v \in \Sigma^*$, and $\mathbf{L}(ua, v) = \mathbf{L}(u, va) = 0$. Let Γ be the set of all symbols that occur in u, then $u \in \Gamma^*$ and $v \in (\Sigma \setminus (\Gamma \cup a))^*$. The size d of Γ can range from 0 to $k-1$ and we have $\binom{k-1}{d}$ ways of selecting Γ. The number of sequences using all symbols from Γ is given by the generating function $p(d,z)$, for the number of sequences over $\Sigma \setminus (\Gamma \cup a)$ we have the generating function $\frac{1}{1-(k-d-1)z}$, and the contribution of the matching symbol is kz^2. The generating function for the number of minimal matches then is

$$\lambda(z) = \sum_{d=0}^{k-1} \binom{k-1}{d} \frac{kz^2}{1-(k-d-1)z} \prod_{j=1}^{d} \frac{jz}{1-jz}.$$

The generating function for the number of collations in the sets $\mathcal{H}(i)$ then is $(\lambda(z))^i p(z)$, where $p(z)$ is the generating function for the number of matches of the form $\binom{\Gamma^*}{(\Sigma\setminus\Gamma)^*}$. Using Theorem 4.1 we can get the upper bounds in the second column of the table in Figure 4.2.

4.3 Simple upper bound (binary alphabet)

In this section and Section 4.5 we shall work with the binary alphabet $\Sigma = \{0, 1\}$. Previous methods for an estimate of $F(n, i)$ counted the number of all possible pairs that have a specific sequence as a common subsequence. To improve the upper bound we must avoid counting any one pair too many times. To do so we shall use the dominance partial order. We shall not count a pair $\binom{u}{v}$ in association with subsequence w when we know that there is some subsequence w' of $\binom{u}{v}$ such that the collation based on w' dominates that of w.

We shall take advantage of the following idea. The collation $\binom{01}{1}\binom{1}{01}\binom{0}{0}$ generates a pair $\binom{0110}{1010}$ and has the collation key $3, 6, 8$. But having the match $\binom{01}{1}$ followed by the match $\binom{1}{01}$ is not optimal, because it is possible to arrange matches in a better way, namely $\binom{0}{10}\binom{1}{1}\binom{10}{0}$, with the collation key $3, 5, 8$. This idea is generalized and made precise in the following Lemma.

Lemma 4.4 *If collation* $\mathbf{p}$ *contains matches of the form* $\left[\binom{0^+1}{1}\binom{1}{0^+1}\right]$ *then* $\mathbf{p}$ *is dominated.*

Proof. Let $\mathbf{p} = p_1, \ldots, p_{n+1}$ be a collation such that $p_i = \binom{0^+1}{1}$ and $p_{i+1} = \binom{1}{0^+1}$, $i < n$. Then $\mathbf{p}$ is dominated by the collation

$$p_1, \ldots, p_{i-1}, \binom{0^+}{10^+}, \binom{1}{1}, \binom{1t(p_{i+2})}{b(p_{i+2})}, p_{i+3}, \ldots, p_{n+1}\,.$$

□

These inefficient collations from Lemmas 4.3 and 4.4 are said to be *rejected.* Collations that are not rejected, are *accepted.*

For $i > 0$ let $\mathcal{H}(i, m)$ be the set of all accepted collations $\mathbf{p}$ of order i generating pairs of total length m and let $H(i, m)$ be the number of pairs in

the $\mathcal{H}(i,m)$. For $i=0$ we put $\mathcal{H}(0,m)=\mathcal{N}(0,m)$. Let $\mathcal{H}(i)=\bigcup_{m=0}^{\infty}\mathcal{H}(i,m)$. Every dominated collation is accepted and therefore $\mathcal{H}(i)$ can be used for obtaining lower bounds through Theorem 4.1.

To count $H(i,m)$ we can split $\mathcal{H}(i,m)$, $i\geq 1$, into two sets $\mathcal{H}_1(i,m)$ and $\mathcal{H}_2(i,m)$ such that all collations beginning with a match of type $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$ will form $\mathcal{H}_1(i,m)$ and all collations beginning with a match of type $\left[\begin{smallmatrix}1^+0\\0\end{smallmatrix}\right]$ will form $\mathcal{H}_2(i,m)$. For $i=0$ we set $\mathcal{H}_1(0,m)=\mathcal{H}(0,m)$ and $\mathcal{H}_2(0,m)=\emptyset$.

We build the sets $\mathcal{H}_1(i,m)$ and $\mathcal{H}_2(i,m)$ by induction on i. All the collations of order 0 (forming the set $\mathcal{H}_1(0,m)$) are those in $\left[\begin{smallmatrix}\{0,1\}^*\\ \lambda\end{smallmatrix}\right]\cup\left[\begin{smallmatrix}0^+\\1^+\end{smallmatrix}\right]$. Now let us suppose we have the sets $\mathcal{H}_1(i-1,m)$ and $\mathcal{H}_2(i-1,m)$ for all m. To get all accepted collations from $\mathcal{H}(i,m')$ we shall extend the collations from $\mathcal{H}(i-1,m)$ to the left by one match. The extensions by the matches from $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$ form $\mathcal{H}_1(i,m')$. The situation in the case of extensions by the matches from $\left[\begin{smallmatrix}1^+0\\0\end{smallmatrix}\right]$ is a bit more complicated. When the collation from $\mathcal{H}_2(i-1,m)$ begins with a match from $\left(\begin{smallmatrix}1^+0\\0\end{smallmatrix}\right)$, the extension by a match from $\left(\begin{smallmatrix}0\\1^+0\end{smallmatrix}\right)$ does not create an accepted collation. Therefore $\mathcal{H}_2(i,m')$ is formed by the extensions of collations from $\mathcal{H}_1(i-1,m)$ by the matches from $\left[\begin{smallmatrix}1^+0\\0\end{smallmatrix}\right]$, and by the extensions of collations from $\mathcal{H}_2(i-1,m)$ beginning with a match from $\left(\begin{smallmatrix}1^+0\\0\end{smallmatrix}\right)$ by matches from $\left(\begin{smallmatrix}1^+0\\0\end{smallmatrix}\right)\cup\left(\begin{smallmatrix}0^+1\\1\end{smallmatrix}\right)\cup\left(\begin{smallmatrix}1\\0^+1\end{smallmatrix}\right)$, and similarly for the symmetric cases. The containments between the sets $\mathcal{H}_1(i,m)$ and $\mathcal{H}_2(i,m)$ can be described by the diagram in Figure 4.3.

Let $H_j(i,m)$ be the number of collations in $\mathcal{H}_j(i,m)$, $j=1,2$. From the definitions of $\mathcal{H}_1(0,m)$ and $\mathcal{H}_2(0,m)$ we have

$$\begin{aligned} H_1(0,0) &= 1\,,\\ H_1(0,m) &= 2^{m+1}+2m-2 \quad \text{for } m>0\,,\\ H_2(0,m) &= 0\,. \end{aligned}$$

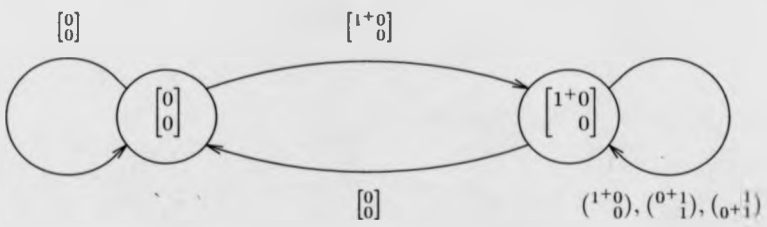

Figure 4.3: The containments between the sets $\mathcal{H}_1(i,m)$ and $\mathcal{H}_2(i,m)$.

For convenience we take $H_1(i,m) = H_2(i,m) = 0$ for $m < 0$ and all $i \geq 0$. Now we can transform the containments from the diagram into the following recurrences.

$$\begin{aligned} H_1(i,m) &= 2H_1(i-1,m-2) + 2H_2(i-1,m-2)\,, \\ H_2(i,m) &= 4\sum_{j=3}^{m} H_1(i-1,m-j) + 3\sum_{j=3}^{m} H_2(i-1,m-j)\,. \end{aligned}$$

We can express these recurrences more compactly in terms of the generating functions $h_1^{(i)}(z) = \sum H_1(i,m)z^m$, $h_2^{(i)}(z) = \sum H_2(i,m)z^m$, and $h^{(i)}(z) = h_1^{(i)}(z) + h_2^{(i)}(z) = \sum H(i,m)z^m$. The contribution of each pair from $\binom{0}{0}$ is z^2 and the total contribution of the pairs from $\binom{1^+0}{0}$ is $\frac{z^3}{1-z}$. For $\mathcal{H}_1(0) = \left[\begin{smallmatrix}\{0,1\}^* \\ \lambda\end{smallmatrix}\right] \cup \left[\begin{smallmatrix}0^+ \\ 1^+\end{smallmatrix}\right]$ we have generating function $\frac{1+2z}{1-2z} + \frac{2z^2}{(1-z)^2}$. The resulting recurrences are the following.

$$\begin{aligned} h_1^{(0)} &= \tfrac{1+2z}{1-2z} + \tfrac{2z^2}{(1-z)^2}\,, \\ h_2^{(0)} &= 0\,, \\ h_1^{(i)} &= 2z^2 h_1^{(i-1)} + 2z^2 h_2^{(i-1)}, \\ h_2^{(i)} &= \tfrac{4z^3}{1-z} h_1^{(i-1)} + \tfrac{3z^3}{1-z} h_2^{(i-1)}. \end{aligned} \tag{4.6}$$

To solve the system of the linear recurrences we can use the following well-known lemma.

Lemma 4.5 *Let* $\left\{\boldsymbol{x}^{(i)} = \boldsymbol{Q}\boldsymbol{x}^{(i-1)}; \boldsymbol{x}^{(0)} = (x_1^{(0)}, \ldots, x_r^{(0)})^T\right\}$ *be a system of* r *linear recurrences. Let* $\lambda_1, \ldots, \lambda_l$ *be the characteristic values of the matrix* $\boldsymbol{Q}$ *and let* $e_1, \ldots, e_l$ *be their multiplicities. Then there are constants* $s_{111}, \ldots, s_{rle_l}$ *such that, for sufficiently large* i,

$$x_t^{(i)} = \sum_{j=1}^{l} \left(s_{tj1} + s_{tj2}i + \cdots + s_{tje_j}i^{e_j-1}\right)\lambda_j^i\,, \quad 1 \le t \le r\,.$$

Proof. Let $\boldsymbol{Q} = \boldsymbol{P}\boldsymbol{R}\boldsymbol{P}^{-1}$ be a decomposition of $\boldsymbol{Q}$ into Jordan normal form. Therefore $\boldsymbol{x}^{(i)} = \boldsymbol{Q}^i\boldsymbol{x}^{(0)} = \boldsymbol{P}\boldsymbol{R}^i\boldsymbol{P}^{-1}\boldsymbol{x}^{(0)}$, where

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{B}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{B}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{B}_l \end{pmatrix}, \quad \boldsymbol{R}^i = \begin{pmatrix} \boldsymbol{B}_1^i & 0 & \cdots & 0 \\ 0 & \boldsymbol{B}_2^i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{B}_l^i \end{pmatrix},$$

and

$$\boldsymbol{B}_j = \begin{pmatrix} \lambda_j & 1 & 0 & \cdots & 0 \\ 0 & \lambda_j & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_j & 1 \\ 0 & 0 & \cdots & 0 & \lambda_j \end{pmatrix}.$$

Hence for $i \ge e_j$ we have

$$\boldsymbol{B}_j^i = \begin{pmatrix} \lambda_j^i & \binom{i}{1}\lambda_j^{i-1} & \binom{i}{2}\lambda_j^{i-2} & \cdots & \binom{i}{e_j-1}\lambda_j^{i-e_j+1} \\ 0 & \lambda_j^i & \binom{i}{1}\lambda_j^{i-1} & \cdots & \binom{i}{e_j-2}\lambda_j^{i-e_j+2} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_j^i & \binom{i}{1}\lambda_j^{i-1} \\ 0 & 0 & \cdots & 0 & \lambda_j^i \end{pmatrix}.$$

Putting all this together, gives us $\boldsymbol{x}_t^{(i)}$ in the required form. □

Lemma 4.5 gives us upper bounds in the form required by Theorem 4.1

Corollary 4.1 *Let* $\left\{\boldsymbol{x}^{(i)}(z) = \boldsymbol{Q}(z)\boldsymbol{x}^{(i-1)}(z); \boldsymbol{x}^{(0)}(z) = (x_1^{(0)}, \ldots, x_r^{(0)})^T\right\}$ *be a system of* r *linear recurrences. Let* $\lambda_1(z), \ldots, \lambda_l(z)$ *be the characteristic values of the matrix* $\boldsymbol{Q}(z)$ *and let* $\lambda(z) = \max_{j=1,\ldots,l} |\lambda_j(z)|$. *There is some* $s(z)$, *independent of* i, *such that*

$$x^{(i)}(z) \leq ri^r s(z)\,(\lambda(z))^i .$$

For the system of linear recurrences (4.6) we have

$$\boldsymbol{Q}(z) = \begin{pmatrix} 2z^2 & 2z^2 \\ \frac{4z^3}{1-z} & \frac{3z^3}{1-z} \end{pmatrix}$$

with characteristic values

$$\begin{aligned} \lambda(z) \;=\; \lambda_1(z) &= \tfrac{z^2}{2-2z}\left(2 + z + \sqrt{(2-z)(2+7z)}\right), \\ \lambda_2(z) &= \tfrac{z^2}{2-2z}\left(2 + z - \sqrt{(2-z)(2+7z)}\right). \end{aligned}$$

From Corollary 4.1 we get

$$h^{(i)}(z) \leq s(z)(\lambda(z))^i$$

for some $s(z)$. Theorem 4.1 then yields an upper bound $\gamma_2 \leq 0.853173$, with $z_0 = 0.185199$ and $\lambda(z_0) = 0.0974715$.

4.4 Simple upper bound (alphabet size 3)

In the case of a three-letter alphabet we shall proceed in a way similar to the case of the binary alphabet. Let P be the set of all minimal matches, $P = \begin{bmatrix}(2,1)^*0\\0\end{bmatrix} \cup \begin{bmatrix}1^+0\\2^+0\end{bmatrix}$. We can divide P into the sets of 'short' minimal matches, $S = \begin{bmatrix}0\\0\end{bmatrix}$, 'medium' minimal matches $M = \begin{bmatrix}1^+0\\0\end{bmatrix} \cup \begin{bmatrix}1^+0\\2^+0\end{bmatrix}$ and 'long'

minimal matches $L = \left[{}^{\{2,1\}^*2^+1^+0}_{0}\right]$. However, without loss of generality we can suppose that if p is from M then it has form $\binom{1^+0}{2^*0}$ and similarly for S and L.

We need something that corresponds to Lemma 4.4. We define O_1 (O_2) to be the set of all minimal matches $p = \binom{u}{v}$ such that any symbols 0,1 always appear in the order 01 in v (symbols 0,1,2 always appear in the order 012 in v respectively). So

$$O_2 = \binom{\{2,1\}^*0}{0} \cup \binom{\{2,0\}^*1}{1} \cup \binom{\{1,0\}^*2}{2} \cup \binom{2^*1}{0^+1} \cup \binom{0^*2}{1^+2} \cup \binom{1^*2}{0^+2} \cup \binom{2}{0^+1^+2}$$

and

$$O_1 = O_2 \cup \binom{1^*0}{2^+0} \cup \binom{0^+1}{2^+1} \cup \binom{1}{\{2,0\}^+1} .$$

From the definition of the set O_1 we know that if $p \in P \setminus O_1$ and $q \in M$ (or $p \in P \setminus O_2$ and $q \in L$) then collation $pq\binom{\lambda}{\lambda}$ is dominated. Using the following construction we are able to eliminate collations containing such patterns and thus get better upper bounds.

Let $Q_{1,1}, \ldots, Q_{3,3}$ be the following sets of matches:

$$\begin{aligned} Q_{1,1} &= S\,, & Q_{1,2} &= S \cap O_1\,, & Q_{1,3} &= S \cap O_2\,, \\ Q_{2,1} &= M\,, & Q_{2,2} &= M \cap O_1\,, & Q_{2,3} &= M \cap O_2\,, \\ Q_{3,1} &= L\,, & Q_{3,2} &= L \cap O_1\,, & Q_{3,3} &= L \cap O_2\,. \end{aligned}$$

We shall define pairwise disjoint sets $\mathcal{H}_1(i)$, $\mathcal{H}_2(i)$, and $\mathcal{H}_3(i)$ recursively according to i. First we define $\mathcal{H}_1(0)$ as the set of all nondominated collations of order 0. Clearly these collations are exactly the collations that consist of one pair with no match. Moreover we define $\mathcal{H}_2(0) = \mathcal{H}_3(0) = \emptyset$. For $i > 0$, the set $\mathcal{H}_r(i)$ is formed by all collations of the form $p\mathbf{q}$, where $p \in Q_{r,s}$ and

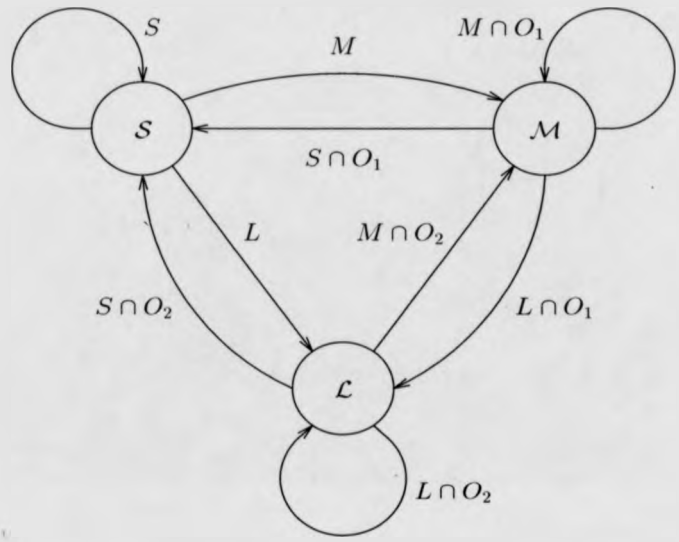

Figure 4.4: The containments between the sets $\mathcal{S}$, $\mathcal{M}$, and $\mathcal{L}$.

$\mathbf{q} \in \mathcal{H}_s(i-1)$, for $r, s \in \{1,2,3\}$. We shall give mnemonic names to the sets $\mathcal{H}_j(i)$, namely $\mathcal{H}_1(i)$ will be $\mathcal{S}(i)$, $\mathcal{H}_2(i)$ will be $\mathcal{M}(i)$ and $\mathcal{H}_3(i)$ will be $\mathcal{L}(i)$. Finally we define $\mathcal{H}(i) = \mathcal{S}(i) \cup \mathcal{M}(i) \cup \mathcal{L}(i)$. These recurrences can be described by the diagram in Figure 4.4

Now we shall introduce a more sophisticated technique to show that every nondominated collation of order i is in the set $\mathcal{H}(i)$. This will mean that the sets $\mathcal{H}(i)$ can be used to produce upper bounds for the expected length of a longest common subsequence as given by Theorem 4.1. We also redefine sets $Q_{1,1}, \ldots, Q_{3,3}$ to get better upper bounds.

From Lemma 4.4 we already know that if the first match of collation $\mathbf{p}$ has the form $\binom{10}{0}$ then the collation $\binom{0}{10}\mathbf{p}$ is dominated. We shall capture

such collations $\mathbf{p}$ in the set $\mathcal{U}(10)$. For general u, we shall define the set $\mathcal{U}(u)$.

Definition 4.5 For every $u \in \Sigma^*$, let

$$\mathcal{U}(u) = \{\mathbf{p} : \forall \binom{v}{w} \in \Delta\ (\mathbf{L}(u,w) > 1 \Longrightarrow \binom{v}{w}\mathbf{p} \text{ is dominated})\}.$$

These sets possess the following properties analogous to Lemma 4.4. With every property there is also shown a pattern illustrating it.

P.1 $\binom{10}{0}\mathbf{p} \in \mathcal{U}(10)$, $\binom{0}{10}\binom{10}{0}$,

P.2 $\binom{210}{0}\mathbf{p} \in \mathcal{U}(210)$, $\binom{1}{21}\binom{210}{0}$,

P.3 if $\mathbf{p} \in \mathcal{U}(10)$ then $\binom{0}{0}\mathbf{p} \in \mathcal{U}(01)$, $\binom{1}{01}\binom{0}{0}\binom{10}{0}$,

P.4 if $\mathbf{p} \in \mathcal{U}(210)$ then $\binom{0}{0}\mathbf{p} \in \mathcal{U}(021)$, $\binom{2}{02}\binom{0}{0}\binom{210}{0}$,

P.5 if $\mathbf{p} \in \mathcal{U}(210)$ then $\binom{1}{1}\mathbf{p} \in \mathcal{U}(12)$, $\binom{2}{12}\binom{1}{1}\binom{210}{0}$.

Using the sets $\mathcal{U}(a)$ we can extract the significant properties from the collations. The equivalence relation for pairs of strings is naturally extendable to collations. The set of all collations equivalent to some collation from $\mathcal{U}(a)$ is denoted by $\mathcal{U}[a]$.

Clearly, if a collation $\mathbf{p}$ starts with a match from M then $\mathbf{p}$ is in $\mathcal{U}[10]$ (property P.1), and if $\mathbf{p}$ starts with a match from L then $\mathbf{p} \in \mathcal{U}[210]$ (property P.2). Because of the significance of the sets $\mathcal{U}[10]$ and $\mathcal{U}[210]$ we shall denote them by $\mathcal{U}_M$ and $\mathcal{U}_L$ respectively. Let $\mathcal{U}_S$ be the set of all collations. These sets were designed to satisfy Lemma 4.6.

Lemma 4.6 *For every $i \in \mathbb{N}$.*

$$\mathcal{S}(i) \subseteq \mathcal{U}_S\,, \qquad \mathcal{M}(i) \subseteq \mathcal{U}_M\,, \qquad \mathcal{L}(i) \subseteq \mathcal{U}_L\,.$$

Before proving this lemma we shall use the remaining properties of the $\mathcal{U}$'s to improve the upper bounds further. Let P_1 be the set of all matches from P having 1 as a matching symbol. According to property P.5, if $\mathbf{p} \in \mathcal{L}(i)$ then collation $\binom{1}{1}\mathbf{p}$ can be put in $\mathcal{M}(i+1)$ instead of $\mathcal{S}(i+1)$. This allows us to redefine the Q's.

$$Q'_{1,3} = Q_{1,3} \setminus P_1\,, \quad Q'_{2,3} = Q_{2,3} \cup (Q_{1,3} \cap P_1)\,, \quad Q'_{r,s} = Q_{r,s} \text{ otherwise.}$$

Let P_0 be the set of all minimal matches with matching symbol 0. We shall use properties P.3 and P.4, which for $\mathbf{p} \in \mathcal{M}(i)$ put $\binom{0}{0}\mathbf{p}$ into $\mathcal{M}(i+1)$ and for $\mathbf{p} \in \mathcal{L}(i)$ put $\binom{0}{0}\mathbf{p}$ into $\mathcal{L}(i+1)$. Thus we get the final refinement of the Q's.

$$\begin{aligned}
Q''_{1,2} &= Q'_{1,2} \setminus P_0\,, & Q''_{2,2} &= Q'_{2,2} \cup (Q_{1,2} \cap P_0)\,,\\
Q''_{2,3} &= Q'_{2,3} \setminus P_0\,, & Q''_{3,3} &= Q'_{3,3} \cup (Q_{2,3} \cap P_0)\,,\\
Q''_{r,s} &= Q'_{r,s} \text{ otherwise.}
\end{aligned}$$

Now let the sets $\mathcal{S}(i)$, $\mathcal{M}(i)$, and $\mathcal{L}(i)$ be constructed using the Q'''s instead of the Q's. We can now return to the proof of Lemma 4.6.

Proof (of Lemma 4.6). Since $\mathcal{U}_S$ is the set of all collations, the first inclusion is trivial. We prove the second inclusion by induction on i.

For $i = 0$ we have $\mathcal{M}(0) = \emptyset \subseteq \mathcal{U}(10)$. For $i > 0$ let $\mathbf{p} = p\mathbf{q}$ be a collation from $\mathcal{M}(i)$. The following cases are based on the construction of the set $\mathcal{M}(i)$.

Case 1. $p \in Q''_{2,1} = M$, $\mathbf{q} \in \mathcal{S}(i-1)$. From P.1 we can conclude that $\mathbf{p} \in \mathcal{U}_M$.

Case 2. $p \in Q''_{2,2} = (M \cap O_1) \cup (S \cap O_1 \cap P_0)$, $\mathbf{q} \in \mathcal{M}(i-1)$. Without loss of generality we can suppose $\mathbf{q} \in \mathcal{U}(10)$. If $p \in M \cap O_1$ then we can conclude $\mathbf{p} \in \mathcal{U}_M$ using P.1. If $p \in S \cap O_1 \cap P_0$ then, from P.3, we get $\mathbf{p} \in \mathcal{U}_M$.

Case 3. $p \in Q''_{2,3} = (M \cap O_2 \setminus P_0) \cup (S \cap O_2 \cap P_1 \setminus P_0)$, $\mathbf{q} \in \mathcal{L}(i-1)$. If $p \in M \cap O_2$ then we again conclude $\mathbf{q} \in \mathcal{U}_M$ using P.1. If $p \in S \cap O_2 \cap P_1$, we get $\mathbf{q} \in \mathcal{U}_M$ from P.5.

The third inclusion can be proved in the same way as the second. □

To get upper bounds for γ_k we need to show that the sets $\mathcal{H}(i)$ cover all nondominated collations.

Theorem 4.3 *If* $\mathbf{p}$ *is a nondominated collation of order* i *then* $\mathbf{p} \in \mathcal{H}(i)$.

Proof. We use induction on the order of the collation $\mathbf{p}$. The base of the induction holds since $\mathcal{H}(0)$ is exactly the set of all nondominated collations of order 0. Now let $\mathbf{p}$ be a nondominated collation of order $i > 0$. We can express $\mathbf{p}$ as $p\mathbf{q}$ where p is a match and $\mathbf{q}$ is a collation of order $i-1$. Since $\mathbf{p}$ is nondominated, p must be a minimal match and $\mathbf{q}$ must be a nondominated collation. Hence $\mathbf{q} \in \mathcal{H}(i-1) = \mathcal{S}(i-1) \cup \mathcal{M}(i-1) \cup \mathcal{L}(i-1)$, and we can consider the following three cases.

Case 1. $\mathbf{q} \in \mathcal{S}(i-1)$. We have $p \in P = S \cup M \cup L$. If $p \in S = Q''_{1,1}$ then $p\mathbf{q} \in \mathcal{S}(i)$, if $p \in M = Q''_{2,1}$ then $p\mathbf{q} \in \mathcal{M}(i)$, if $p \in L = Q''_{3,1}$ then $p\mathbf{q} \in \mathcal{L}(i)$. Hence $\mathbf{p} = p\mathbf{q} \in \mathcal{H}(i)$.

Case 2. $\mathbf{q} \in \mathcal{M}(i-1) \subseteq \mathcal{U}[10]$. Again without loss of generality we can suppose $\mathbf{q} \in \mathcal{U}(10)$. Collation $p\mathbf{q}$ is nondominated and p must be from O_1. Since $Q''_{1,2} \cup Q''_{2,2} \cup Q''_{3,2} = O_1$ the construction of the sets $\mathcal{S}(i)$, $\mathcal{M}(i)$ and $\mathcal{L}(i)$ guarantees that $\mathbf{p} = p\mathbf{q} \in \mathcal{H}(i)$.

Case 3. $\mathbf{q} \in \mathcal{L}(i-1) \subseteq \mathcal{U}[210]$. This case is similar to Case 2, and so we have $p \in O_2 = Q''_{1,3} \cup Q''_{2,3} \cup Q''_{3,3}$ and $\mathbf{p} = p\mathbf{q} \in \mathcal{H}(i)$. □

This recursive construction allows us to derive upper bounds for the number of collations in $\mathcal{H}(i)$. Let $q''_{r,s}(z)$ be the generating functions for the number of elements in $Q''_{r,s}$, i.e., $q''_{r,s}(z) = \sum_{m=0}^{\infty} |Q''_{r,s} \cap P(m)| z^m$, where $P(m) = \{\binom{u}{v} : l(\binom{u}{v}) = m\}$ is the set of all minimal matches with total length m.

Let $s^{(i)}(z)$, $m^{(i)}(z)$, $l^{(i)}(z)$, $h^{(i)}(z)$ be the generating functions for the number of collations in $\mathcal{S}(i)$, $\mathcal{M}(i)$, $\mathcal{L}(i)$, and $\mathcal{H}(i)$, i.e.,

$$s^{(i)}(z) = \sum_{m=0}^{\infty} |\mathcal{S}(i) \cap \mathcal{C}(m)| z^m, m^{(i)}(z) = \sum_{m=0}^{\infty} |\mathcal{M}(i) \cap \mathcal{C}(m)| z^m,$$

$$l^{(i)}(z) = \sum_{m=0}^{\infty} |\mathcal{L}(i) \cap \mathcal{C}(m)| z^m,$$

$$h^{(i)}(z) = s^{(i)}(z) + m^{(i)}(z) + l^{(i)}(z) = \sum_{m=0}^{\infty} |\mathcal{H}(i) \cap \mathcal{C}(m)| z^m.$$

The recursion defining the sets $\mathcal{S}(i)$, $\mathcal{M}(i)$, and $\mathcal{L}(i)$ gives the following recurrence equations for $s^{(i)}(z)$, $m^{(i)}(z)$, and $l^{(i)}(z)$:

$$\begin{aligned} s^{(i)}(z) &= q''_{1,1}(z)s^{(i-1)}(z) + q''_{1,2}(z)m^{(i-1)}(z) + q''_{1,3}(z)l^{(i-1)}(z), \\ m^{(i)}(z) &= q''_{2,1}(z)s^{(i-1)}(z) + q''_{2,2}(z)m^{(i-1)}(z) + q''_{2,3}(z)l^{(i-1)}(z), \\ l^{(i)}(z) &= q''_{3,1}(z)s^{(i-1)}(z) + q''_{3,2}(z)m^{(i-1)}(z) + q''_{3,3}(z)l^{(i-1)}(z). \end{aligned} \quad (4.7)$$

The corresponding matrix is

$$\mathbf{Q}(z) = \begin{pmatrix} 3z^2 & 2z^2 & z^2 \\ \frac{12z^3}{1-z} + \frac{6z^4}{(1-z)^2} & z^2 + \frac{11z^3}{1-z} + \frac{5z^4}{(1-z)^2} & z^2 + \frac{7z^3}{1-z} + \frac{3z^4}{(1-z)^2} \\ \frac{12z^4}{(1-2z)(1-z)} & \frac{z^4}{(1-z)^2} + \frac{8z^4}{(1-2z)(1-z)} & z^2 + \frac{2z^3}{1-z} + \frac{z^4}{(1-z)^2} + \frac{6z^4}{(1-2z)(1-z)} \end{pmatrix}.$$

Let $\lambda_1(z)$, $\lambda_2(z)$, $\lambda_3(z)$ be the characteristic values (possibly repeated or complex) of matrix $\mathbf{Q} = \{q''_{r,s}\}$. Let $\lambda(z) = \max\{|\lambda_1(z)|, |\lambda_2(z)|, |\lambda_3(z)|\}$. Solving the system of recurrence equations (4.7) gives us

$$h^{(i)}(z) \leq s(z) i^3 (\lambda(z))^i .$$

This allows us to use Theorem 4.1 to get upper bounds for γ_3. We get the best results if we set $z_0 = 0.145256$. Then $\lambda_1(z_0) = 0.114253$, $\lambda_2(z_0) = 0.022959 + 0.002789i$, $\lambda_3(z_0) = 0.022959 - 0.002789i$ and $\gamma_3 < 0.765803$.

4.5 Upper bounds for binary alphabet

To improve the upper bound for the binary alphabet we can find other cases where specific collations are not efficient enough, for example, when $\mathbf{p}$ is a collation containing any of the following patterns:

$$\left[\binom{1}{01}\binom{0}{0}\left(\binom{1}{1}\binom{0}{0}\right)^*\binom{10}{0}\right], \qquad \left[\binom{0}{10}\left(\binom{1}{1}\binom{0}{0}\right)^*\binom{10}{0}\right],$$

$$\left[\binom{0}{110}\binom{1}{1}\left(\binom{1}{1}\binom{0}{0}\right)^*\binom{10}{0}\right], \qquad \left[\binom{0}{110}\binom{0}{0}\binom{110}{0}\right].$$

Again, in each case, the collation $\mathbf{p}$ is dominated. To capture such patterns we have to split set $\mathcal{H}(i,m)$ into more than just two sets. We also refine the classification of matches into 10 sets: $\binom{0}{0}$, $\binom{1}{1}$, $\binom{10}{0}$, $\binom{0}{10}$, $\binom{01}{1}$, $\binom{1}{01}$, $\binom{1^+10}{0}$, $\binom{0}{1^+10}$, $\binom{0^+01}{1}$, $\binom{1}{0^+01}$. We have to redefine the sets $\mathcal{U}(u)$ as well. First we strengthen the definition of the dominance relation. A collation generating $\binom{uv}{w}$ will be 'u-dominated' if it can be dominated by a collation that does not involve u to make matches.

Definition 4.6 Let $u \in \Sigma^*$. The collation $\mathbf{q}$ is *u-dominated*, if there is a collation $\mathbf{p} = p_1, \ldots, p_{n+1}$ that dominates $\mathbf{q}$ and $u \sqsubseteq v \lhd t(p_1)$ for some $v \in \Sigma^*$.

For $u, w \in \Sigma^*$ and $m \in \mathbf{N}$ we shall denote by $N_m(u,w)$ the smallest suffix v of u such that $\mathbf{L}(v,w) = m$. Let $K_m(u,w)$ be the remaining prefix, i.e. $u = K_m(u,w)N_m(u,w)$. We will omit the index m when $\mathbf{L}(u,w) = m$.

Definition 4.7 For every $u \in \Sigma^*$ and $i \in \mathbb{N}$ we define

$$\mathcal{U}_t(u,i) = \{\mathbf{p} : \forall \binom{v}{w} \in \Delta\ (\mathbf{L}(u,w) > i \Longrightarrow \binom{v}{w}\mathbf{p} \text{ is } K_{i+1}(u,w)\text{-dominated})\}.$$

and

$$\mathcal{U}_b(u,i) = \{\mathbf{p} : \lfloor \mathbf{p} \rceil \in \mathcal{U}_t(u,i)\}.$$

If collation $\binom{v}{w}\mathbf{p}$ is $K_{i+1}(u,w)$-dominated then collation $\binom{v}{N_{i+1}(w,u)}\mathbf{p}$ is also $K_{i+1}(u,w)$-dominated and $L(u, N_{i+1}(w,u)) = i+1$. Therefore

$$\mathcal{U}_t(u,i) = \{\mathbf{p} : \forall \binom{v}{w} \in \Delta\ (\mathbf{L}(u,w) = i+1 \Longrightarrow \binom{v}{w}\mathbf{p} \text{ is } K(u,w)\text{-dominated})\}.$$

These sets are monotone in the following sense.

Lemma 4.7 *For every $i, j \in \mathbb{N}$ and $u, v \in \Sigma^*$*

1. *If $i \le j$ then $\mathcal{U}(u,i) \subseteq \mathcal{U}(u,j)$,*
2. *if $u \sqsubseteq v$ then $\mathcal{U}(v,i) \subseteq \mathcal{U}(u,i)$,*
3. *if $i \le |u|$ then $\mathcal{U}(u,i) = \mathcal{C}$,*

where $\mathcal{U}$ is either $\mathcal{U}_t$ or $\mathcal{U}_b$ and $\mathcal{C}$ is the set of all collations.

Proof. 1. Let $\mathbf{p} \in \mathcal{U}(u,i)$ and let $\binom{v}{w} \in \Delta$ be such that $\mathbf{L}(u,w) > j \ge i$. Since $\mathbf{p} \in \mathcal{U}(u,i)$ and $\mathbf{L}(u,w) > i$, there is a collation $\mathbf{q}$ that $K_{i+1}(u,w)$-dominates $\mathbf{p}$. For $i \le j$ we have $K_{j+1}(u,w) \trianglelefteq K_{i+1}(u,w)$ and therefore collation $\mathbf{q}$ also $K_{j+1}(u,w)$-dominates $\mathbf{p}$. Hence $\mathbf{p} \in \mathcal{U}(u,j)$.

2. Let $\mathbf{p} \in \mathcal{U}(v,i)$ and let $\binom{x}{w} \in \Delta$ be such that $\mathbf{L}(u,w) > i$. Since $u \sqsubseteq v$, then also $\mathbf{L}(v,w) > i$ and $\binom{x}{w}\mathbf{p}$ is $K_{i+1}(v,w)$-dominated. For $u \sqsubseteq v$ we have $K_{i+1}(u,w) \sqsubseteq K_{i+1}(v,w)$ and therefore $\binom{x}{w}\mathbf{p}$ is $K_{i+1}(u,w)$-dominated.

3. If $i \leq |u|$ then it can never happen that $\mathbf{L}(u, w) > i$, and therefore every collation is in $\mathcal{U}(u, i)$. □

The main properties of these sets are captured by the following theorem.

Theorem 4.4 *Let x, y, w be sequences from Σ^* such that $\binom{x}{y}$ is a match.*

1. *If $\mathbf{p}$ is any collation, then $\binom{x}{y}\mathbf{p} \in \mathcal{U}_t(x, 1) \cap \mathcal{U}_b(y, 1)$,*
2. *if $\mathbf{p}$ is a collation from $\mathcal{U}_t(w, i)$, $i \geq 1$, and u, v are sequences from Σ^* such that $uv = xw$, then $\binom{x}{y}\mathbf{p} \in \mathcal{U}_t(u, i - \mathbf{L}(v, y) + 1)$.*

Proof. 1. Let $\binom{v}{w} \in \Delta$ be a match such that $\mathbf{L}(x, w) \geq 2$, hence there is a collation $\binom{x'}{w'}\binom{x''}{w''}\binom{x'''}{w'''}$ of order 2 that generates $\binom{x}{w}$. Collation $\binom{v}{w}\binom{x}{y}\mathbf{p} = \binom{v}{w}\binom{x}{y}p_1 \cdots p_{n+1}$ is then dominated by $\binom{vx'}{w'}\binom{x''}{w''}\binom{x'''t(p_1)}{w'''yb(p_1)}p_2 \cdots p_{n+1}$ and therefore from Definition 4.7 we have $\binom{x}{y}\mathbf{p} \in \mathcal{U}_t(x, 1)$. In the same manner we can prove that $\binom{x}{y}\mathbf{p} \in \mathcal{U}_b(y, 1)$.

2. It is sufficient to prove that $\binom{z'}{z}\binom{x}{y}\mathbf{p}$ is $K(u, z)$-dominated for all $\binom{z'}{z} \in \Delta$ such that $\mathbf{L}(u, z) = i - \mathbf{L}(v, y) + 2$. Let $u' = N(u, z)$ be the smallest suffix of u such that $\mathbf{L}(u, z) = \mathbf{L}(u', z)$, then $\mathbf{L}(u', z) + \mathbf{L}(v, y) = i + 2$. Since $\mathbf{L}(u'v, zy) \geq i + 2$ there is a collation $\mathbf{q} = q_1, \ldots, q_{i+3}$ of order $i + 2$ generating $\binom{u'v}{zy}$. Since $t(q_1, \ldots, q_{i+3}) = u'v$ is a substring of $uv = xw$, then either $t(q_1)t(q_2)$ is a substring of x or $a.t(q_3, \ldots, q_{i+3})$ is a substring of w where a is the matching symbol from q_2.

If $t(q_1)t(q_2)$ is a substring of x, say $t(q_1)t(q_2) = x(j' \,..\, j)$, then collation

$$\binom{z'K(u,z)t(q_1)}{b(q_1)}\binom{t(q_2)}{b(q_2)}\binom{x(j+1\,..\,|x|)t(p_1)}{b(q_3\,..\,q_{i+3}p_1)}p_2 \cdots p_{n+1}$$

$K(u, z)$-dominates collation $\binom{z'}{z}\binom{x}{y}\mathbf{p}$ because $i \geq 1$ and $b(q_3 \,..\, q_{i+3}) \neq \lambda$.

If $a.t(q_3,\ldots,q_{i+3})$ is a substring of w then $\mathbf{L}(w, b(q_2,\ldots,q_{i+3})) > i$. Let j be the order of collation $\mathbf{p}$. Since $\mathbf{p} \in \mathcal{U}_t(w,i)$, there is a collation $\mathbf{r} = r_1,\ldots,r_{j+2}$ of order $j+1$ that $K_{i+1}(w, b(q_2,\ldots,q_{i+3}))$-dominates collation $\binom{x}{b(q_2,\ldots,q_{i+3})}\mathbf{p}$. Let x' be such that

$$K(u,z)t(q_1)x' = xK_{i+1}(w, b(q_2,\ldots,q_{i+3}))r_1$$

. Sequence x' is nonempty becouse it contains symbol a. The collation

$$\binom{z'K(u,z)t(q_1)}{b(q_1)}\binom{x'}{b(r_1)} r_2\cdots r_{j+2}$$

then $K(u,z)$-dominates collation $\binom{z'}{z}\binom{x}{y}\mathbf{p}$. □

Again we can extend the equivalence relation from collations to the sets $\mathcal{U}_{t,b}(u,i)$. Then $\mathcal{U}[u,i] = \mathcal{U}_t[u,i] = \mathcal{U}_b[u,i]$ will be the set of all collations equivalent to some collation from $\mathcal{U}_t(u,i)$.

Now we split set $\mathcal{H}(i,m)$ into the eight sets $\mathcal{H}_1(i,m),\ldots,\mathcal{H}_8(i,m)$ (pairwise disjoint) in such way, that

$$\begin{aligned}
&\mathcal{H}_1(i,m) = \mathcal{V}^{\lambda}_{\lambda} \subseteq \mathcal{U}[\lambda,0], && \mathcal{H}_2(i,m) = \mathcal{V}^{10} \subseteq \mathcal{U}[10,1],\\
&\mathcal{H}_3(i,m) = \mathcal{V}^{110} \subseteq \mathcal{U}[110,1], && \mathcal{H}_4(i,m) = \mathcal{W}^{110} \subseteq \mathcal{U}[110,2],\\
&\mathcal{H}_5(i,m) = \mathcal{V}^{101} \subseteq \mathcal{U}[101,1], && \mathcal{H}_6(i,m) = \mathcal{V}^{11}_{01} \subseteq [\mathcal{U}_t(11,1)\cap\mathcal{U}_b(01,1)],\\
&\mathcal{H}_7(i,m) = \mathcal{V}^{011} \subseteq \mathcal{U}[011,1], && \mathcal{H}_8(i,m) = \mathcal{V}^{01}_{10} \subseteq [\mathcal{U}_t(01,1)\cap\mathcal{U}_b(10,1)],
\end{aligned}$$

where $\mathcal{V}^{\lambda}_{\lambda},\ldots$ are more meaningful names for $\mathcal{H}_1(i,m),\ldots,\mathcal{H}_8(i,m)$. Using Lemma 4.7 and Theorem 4.4 we can establish containments among the sets $\mathcal{H}_1(i,m),\ldots,\mathcal{H}_8(i,m)$ as shown in Figure 4.5. For example if $\mathbf{p} \in \mathcal{V}^{10} \subseteq \mathcal{U}_t(10,1)$, then we get $\binom{1}{01}\mathbf{p} \in \mathcal{U}_t(11,1)$ using part 2 of Theorem 4.4 with $u = 11$, $v = 0$, $w = 10$, $x = 1$, $y = 01$, and $i = 1$. But according to part 1 of the theorem also $\binom{1}{01}\mathbf{p} \in \mathcal{U}_b(01,1)$. Hence $\binom{1}{01}\mathbf{p}$ can be put into $\mathcal{V}^{11}_{01}$.

	$\binom{0}{0}$	$\binom{1}{1}$	$\binom{10}{0}$	$\binom{0}{10}$	$\binom{01}{1}$	$\binom{1}{01}$	$\binom{1^+10}{0}$	$\binom{0}{1^+10}$	$\binom{0^+01}{1}$	$\binom{1}{0^+01}$
$\mathcal{V}^{\lambda}_{\lambda}$	$\mathcal{V}^{\lambda}_{\lambda}$	$\mathcal{V}^{\lambda}_{\lambda}$	$\mathcal{V}^{10}$	$\mathcal{V}_{10}$	$\mathcal{V}^{01}$	$\mathcal{V}_{01}$	$\mathcal{V}^{110}$	$\mathcal{V}_{110}$	$\mathcal{V}^{001}$	$\mathcal{V}_{001}$
$\mathcal{V}^{10}$	$\mathcal{V}^{01}$	$\mathcal{W}^{110}$	$\mathcal{V}^{101}$	$\times$	$\mathcal{V}^{01}$	$\mathcal{V}^{11}_{01}$	$\mathcal{V}^{101}$	$\times$	$\mathcal{V}^{001}$	$\mathcal{V}_{001}$
$\mathcal{V}^{110}$	$\mathcal{V}^{011}$	$\mathcal{W}^{110}$	$\mathcal{V}^{101}$	$\times$	$\mathcal{V}^{011}$	$\mathcal{V}^{11}_{01}$	$\mathcal{V}^{101}$	$\times$	$\mathcal{V}^{011}$	$\mathcal{V}_{001}$
$\mathcal{W}^{110}$	$\mathcal{V}^{\lambda}_{\lambda}$	$\mathcal{V}^{\lambda}_{\lambda}$	$\mathcal{V}^{10}$	$\mathcal{V}^{01}_{10}$	$\mathcal{V}^{01}$	$\mathcal{V}_{01}$	$\mathcal{V}^{110}$	$\times$	$\mathcal{V}^{001}$	$\mathcal{V}_{001}$
$\mathcal{V}^{101}$	$\mathcal{V}^{01}$	$\mathcal{V}^{110}$	$\mathcal{V}^{101}$	$\times$	$\mathcal{V}^{010}$	$\times$	$\mathcal{V}^{101}$	$\times$	$\mathcal{V}^{010}$	$\times$
$\mathcal{V}^{11}_{01}$	$\mathcal{W}_{001}$	$\mathcal{V}_{10}$	$\mathcal{V}^{10}_{00}$	$\mathcal{V}^{01}_{10}$	$\times$	$\mathcal{V}_{010}$	$\mathcal{V}^{110}$	$\times$	$\times$	$\mathcal{V}_{010}$
$\mathcal{V}^{011}$	$\mathcal{W}^{001}$	$\mathcal{V}^{101}$	$\mathcal{V}^{10}$	$\mathcal{V}^{01}_{10}$	$\mathcal{V}^{010}$	$\times$	$\mathcal{V}^{110}$	$\times$	$\mathcal{V}^{010}$	$\times$
$\mathcal{V}^{01}_{10}$	$\mathcal{V}_{01}$	$\mathcal{V}^{10}$	$\times$	$\mathcal{V}_{101}$	$\mathcal{V}^{010}$	$\times$	$\times$	$\mathcal{V}_{101}$	$\mathcal{V}^{010}$	$\times$

Figure 4.5: The containments among the sets $\mathcal{H}_1(i,m),\ldots,\mathcal{H}_8(i,m)$.

Symbol $\times$ in row $\mathcal{V}$ and column $\binom{u}{v}$ denotes the case when collations $\binom{u}{v}\mathcal{V}$ are dominated by Definition 4.7 and we can exclude such collations from further extension.

Hence we can create a system of eight linear recurrences

$$\begin{aligned}\mathbf{x}^{(i)}(z) &= \mathbf{Q}(z)\mathbf{x}^{(i-1)}(z)\,,\\ \mathbf{x}^{(0)}(z) &= \left(\tfrac{1+2z}{1-2z}+\tfrac{2z^2}{(1-z)^2},0,0,0,0,0,0,0\right)^T,\end{aligned}$$

where

$$\mathbf{Q}(z)=\begin{pmatrix}
2z^2 & 0 & 0 & 2z^2 & 0 & 0 & 0 & 0\\
4z^3 & z^2+z^3 & 0 & 3z^3 & z^2 & z^2 & z^3 & 2z^2\\
\frac{4z^4}{1-z} & \frac{2z^4}{1-z} & \frac{z^4}{1-z} & \frac{3z^4}{1-z} & z^2 & \frac{z^4}{1-z} & \frac{z^4}{1-z} & 0\\
0 & z^2 & z^2 & 0 & 0 & z^2 & z^2 & 0\\
0 & \frac{z^3}{1-z} & \frac{z^3}{1-z} & 0 & \frac{2z^3}{1-z} & \frac{z^3}{1-z} & \frac{z^2}{1-z} & \frac{2z^3}{1-z}\\
0 & z^3 & z^3 & 0 & 0 & z^3 & 0 & 0\\
0 & 0 & \frac{z^2}{1-z} & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & z^3 & 0 & z^3 & z^3 & 0
\end{pmatrix}.$$

Let $\lambda(z)$ be the maximal characteristic value of matrix $\mathbf{Q}(z)$. Using Theorem 4.1 and Corollary 4.1 with $z_0 = 0.228424$, we get $\lambda(z_0) = 0.155602 < 1$ and $\gamma_2 \leq y = 0.842166$. However, for the best bound we have achived so far, we split $\mathcal{H}(i,m)$ into 52 sets, build a system of 52 linear recurrences and use Theorem 4.1 with $z_0 = 0.252652$. As a result we get $\lambda(z_0) = 0.195960 < 1$ and $\gamma_2 \leq y = 0.837623$. The transition diagram and description of sets $\mathcal{H}_1(i,m), \ldots, \mathcal{H}_{52}(i,m)$ are given in Appendix B. There is a special containment between sets $\mathcal{H}_{25}$ and $\mathcal{H}_{32}$. Since $\mathcal{H}_{25} \subseteq \mathcal{U}[1110,3]$, every collation $\binom{0}{1+110}\mathbf{p}$, $\mathbf{p} \in \mathcal{H}_{25}$, is dominated. Therefore only the match $\binom{0}{110}$ can create a nondominated collation and the contribution of this match is only z^4 instead of $\frac{z^4}{1-z}$.

4.6 Upper bounds for larger alphabets

S, M, and L were the sets of minimal matches used to obtain lower bound for the expected length of a longest common subsequence in case $k = 3$. We can interpret these sets as $S = \{p \in P : \|p\| = 1\}$, $M = \{p \in P : \|p\| = 2\}$, $L = \{p \in P : \|p\| = 3\}$, where for a pair $p = \binom{u}{v}$ we define $\|p\| = \max\{\|u\|, \|v\|\}$. For larger alphabets we can set two boundaries – integers m and m' such that $0 < m < m' < k$, and redefine S, M, and L by $S = \{p \in P : 0 < \|p\| \leq m\}$, $M = \{p \in P : m < \|p\| \leq m'\}$, $L = \{p \in P : m' < \|p\| \leq k\}$. Sets corresponding to O_1 and O_2 will be the sets O_m and $O_{m'}$, where O_m is defined as the set of all minimal matches $p = \binom{u}{v}$ such that symbols $0, 1, \ldots, m$ appear in increasing order in v.

Let $Q_{1,1}, \ldots, Q_{3,3}$ be the following sets of matches:

$$Q_{1,1} = S, \quad Q_{1,2} = S \cap O_m, \quad Q_{1,3} = S \cap O_{m'},$$
$$Q_{2,1} = M, \quad Q_{2,2} = M \cap O_m, \quad Q_{2,3} = M \cap O_{m'},$$
$$Q_{3,1} = L, \quad Q_{3,2} = L \cap O_m, \quad Q_{3,3} = L \cap O_{m'}.$$

We need to express the number of matches in $Q_{r,s}$ in terms of generating functions. First we introduce some auxiliary generating functions.

From 4.5, for the generating function for the number of sequences using exactly d given symbols we have

$$p(d, z) = \prod_{j=1}^{d} \frac{jz}{1 - jz}.$$

Let $v(d, c, z)$ be the generating function for the number of sequences using exactly $d + c$ symbols and where the symbols $\{0, \ldots, c-1\}$ occur in increasing order. This is first expressed in the terms of an exponential generating function. The exponential generating function for the number of sequences using exactly one symbol is $e^z - 1$. From the rule for the multiplication of exponential generating functions we get the exponential generating function for the number of sequences using exactly d given symbols $W(d, z) = (e^z - 1)^d$.

Let $V(c, z)$ be the exponential generating function for the number of increasing sequences using exactly c symbols. The exponential generating function for the number of ways to select c positions from j possibilities is $\frac{z^c}{c!} e^z = \sum_{j=c}^{\infty} \binom{j}{c} \frac{z^j}{j!}$. We can see the selected positions as positions for the end of a block of the same symbols in an increasing subsequence. If the last position is selected, the corresponding increasing sequence uses exactly c symbols, if last position is not selected, the corresponding increasing sequence uses exactly

$c+1$ symbols. Therefore

$$V(c,z)+V(c+1,z)=\frac{z^c}{c!}e^z\,.$$

For $c=1$ we have $V(1,z)=e^z-1$. This specifies the recurrence relation

$$\begin{aligned} V(1,z) &= e^z-1\\ V(c+1,z) &= \frac{z^c}{c!}e^z-V(c,z)\end{aligned}$$

with the solution

$$V(c,z)=e^{sz}\left(\sum_{j=0}^{c-1}(-1)^{c-1-j}\frac{z^j s^j}{j!}\right)+(-1)^c\,.$$

The exponential generating function for the number of sequences using exactly $a+c$ symbols and where the symbols $\{0,\ldots,c-1\}$ occur in increasing order is therefore $W(d,z)V(c,z)$. Using (2.2) we transform it to an ordinary generating function

$$v(d,c,z)=\int_0^\infty e^{-s}(e^{sz}-1)^d\left(e^{sz}\left(\sum_{j=0}^{c-1}(-1)^{c-1-j}\frac{z^j s^j}{j!}\right)+(-1)^c\right)ds$$

Let $r(d,b,z)$ be the generating function for the number of all minimal matches q with $\|t(q)\|=d+1$ and $\|b(q)\|=b+1$. We have k choices for the matching symbol, $\binom{k-1}{d}$ choices for the remaining symbols in $t(q)$ and $\binom{k-1-d}{b}$ choices for the remaining symbols in $b(q)$. This gives

$$r(d,b,z)=k\binom{k-1}{d}\binom{k-1-d}{b}z^2p(d,z)p(b,z)\,.$$

Let $s(m,d,b,z)$ be the generating function for the number of all minimal matches $q\in O_m$ with $\|t(q)\|=b+1$ and $\|b(q)\|=d+1$. For matches $q\in O_m$, symbols $0,1,\ldots,m$ have to appear in increasing order in $b(q)$. Suppose c of

these symbols are used in $b(q)$ (excluding the matching symbol). We have to distinguish cases when the matching symbol is among $0, 1, \ldots, m$ and when it is not. This for $s(m, d, b, z)$ gives

$$s(m,d,b,z) = \sum_{c=0}^{d} \binom{m+1}{c+1} \binom{k-m-1}{d-c} \binom{k-1-d}{b} z^2 d(d-c,c,z) p(b,z) + \sum_{c=0}^{d} (k-m-1) \binom{m+1}{c} \binom{k-m-2}{d-c} \binom{k-1-d}{b} z^2 d(d-c,c,z) p(b,z).$$

Now we are able to express the generating functions $q_{r,s}$ for the the number of matches in $Q_{r,s}$.

$$\begin{aligned}
q_{1,1} &= \sum_{d=0}^{m-1} \sum_{b=0}^{m-1} r(d,b,z) \\
q_{1,2} &= \sum_{d=0}^{m-1} \sum_{b=0}^{m-1} s(m,d,b,z) \\
q_{1,3} &= \sum_{d=0}^{m-1} \sum_{b=0}^{m-1} s(m',d,b,z) \\
q_{2,1} &= \sum_{d=m}^{m'-1} \sum_{b=0}^{m'-1} r(d,b,z) + \sum_{d=0}^{m-1} \sum_{b=m}^{m'-1} r(d,b,z) \\
q_{2,2} &= \sum_{d=m}^{m'-1} \sum_{b=0}^{m'-1} s(m,d,b,z) + \sum_{d=0}^{m-1} \sum_{b=m}^{m'-1} s(m,d,b,z) \\
q_{2,3} &= \sum_{d=m}^{m'-1} \sum_{b=0}^{m'-1} s(m',d,b,z) + \sum_{d=0}^{m-1} \sum_{b=m}^{m'-1} s(m',d,b,z) \\
q_{3,1} &= \sum_{d=m'}^{k-1} \sum_{b=0}^{k-1-d} r(d,b,z) + \sum_{d=0}^{m'-1} \sum_{b=m'}^{k-1-d} r(d,b,z) \\
q_{3,2} &= \sum_{d=m'}^{k-1} \sum_{b=0}^{k-1-d} s(m,d,b,z) + \sum_{d=0}^{m'-1} \sum_{b=m'}^{k-1-d} s(m,d,b,z) \\
q_{3,3} &= \sum_{d=m'}^{k-1} \sum_{b=0}^{k-1-d} s(m',d,b,z) + \sum_{d=0}^{m'-1} \sum_{b=m'}^{k-1-d} s(m',d,b,z)
\end{aligned}$$

As a consequence of Theorem 4.4 we have the following two properties of the sets $\mathcal{U}$. For every l, $0 \leq l < m$

$$\mathbf{p} \in \mathcal{U}(m\ (m-1) \cdots 0, 1) \implies \binom{l}{l}\mathbf{p} \in \mathcal{U}(l\ m\ (m-1) \cdots (l+1), 1)\,,$$

$$\mathbf{p} \in \mathcal{U}(m\,(m-1)\,\cdots\,0,1) \implies (_{l\,j}^{\,j})\mathbf{p} \in \mathcal{U}(m\,(m-1)\,\cdots\,(l+1),1)\,.$$

We shall use these properties to improve the upper bounds further. Let R_j, for $1 \le j \le k$, be the set of all minimal matches $p = \binom{u}{v}$ such that the symbols $0, 1, \ldots, j-1$ do not occur in v. Let P_j be the set of all matches from R_j having j as a matching symbol. We redefine the Q's.

$$\begin{aligned} Q'_{1,3} &= Q_{1,3} \cap R_{m'-m} \setminus P_{m'-m}\,, \\ Q'_{2,3} &= Q_{2,3} \cup (Q_{1,3} \setminus R_{m'-m}) \cup (Q_{1,3} \cap P_{m'-m})\,, \\ Q'_{r,s} &= Q_{r,s} \text{ otherwise.} \end{aligned}$$

Let $u(m, j, d, b, z)$ be the generating function for the number of all minimal matches $q \in O_m \cap R_j$ with $\|t(q)\| = d+1$ and $\|b(q)\| = b+1$. Function u is similar to function s, actually $s(m, d, b, z) = u(m, 0, d, b, z)$. The difference in selecting c symbols from $0, 1, \ldots, m$ then gives

$$\begin{aligned} u(m,j,d,b,z) = &\sum_{c=0}^{d} \binom{m-j+1}{c+1} \binom{k-m-1}{d-c} \binom{k-1-d}{b} z^2 v(d-c,c,z)p(b,z) + \\ &\sum_{c=0}^{d} (k-m-1) \binom{m-j+1}{c} \binom{k-m-2}{d-c} \binom{k-1-d}{b} z^2 v(d-c,c,z)p(b,z). \end{aligned}$$

Let $t(m, d, b, z)$ be the generating function for the number of all minimal matches $q \in O_m \cap P_j$ with $\|t(q)\| = d+1$ and $\|b(q)\| = b+1$. Since symbols $\{j, \ldots, m\}$ have to appear in increasing order in $b(q)$ and j is a matching symbol, these symbols cannot be in $b(q)$. The set P_j is a subset of R_j and therefore symbols $0, \ldots, j-1$ are not in $b(q)$. Thus $T(m, d, b, z)$ is not dependent on j and

$$t(m,d,b,z) = \binom{k-1-m}{d} \binom{k-1-d}{b} z^2 p(d,z)p(b,z).$$

For generating functions this means

$$q'_{1,3} = \sum_{d=0}^{m-1} \sum_{b=0}^{m-1} (u(m', m'-m, d, b, z) - t(m', d, b, z))$$

$$q'_{2,3} = q_{2,3} + q_{1,3} - q'_{1,3}$$
$$q'_{r,s} = q_{r,s} \text{ otherwise.}$$

Let P_0 be the set of all minimal matches with matching symbol 0. Thus we get a final refinement of Q's.

$$Q''_{1,2} = Q'_{1,2} \setminus P_0, \qquad Q''_{2,2} = Q'_{2,2} \cup (Q_{1,2} \cap P_0),$$
$$Q''_{2,3} = Q'_{2,3} \setminus P_0, \qquad Q''_{3,3} = Q'_{3,3} \cup (Q_{2,3} \cap P_0),$$
$$Q''_{r,s} = Q'_{r,s} \text{ otherwise.}$$

The generating function for the number of matches in $O_m \cap P_0$ is also $t(m, d, b, z)$.

Now we are able to express the generating functions for the Q'''s.

$$q''_{1,1} = \sum_{d=0}^{m-1} \sum_{b=0}^{m-1} r(d, b, z)$$
$$q''_{1,2} = \sum_{d=0}^{m-1} \sum_{b=0}^{m-1} (s(m, d, b, z) - t(m, d, b, z))$$
$$q''_{1,3} = \sum_{d=0}^{m-1} \sum_{b=0}^{m-1} (u(m', m' - m, d, b, z) - t(m', d, b, z))$$
$$q''_{2,1} = \sum_{d=m}^{m'-1} \sum_{b=0}^{m'-1} r(d, b, z) + \sum_{d=0}^{m-1} \sum_{b=m}^{m'-1} r(d, b, z)$$
$$q''_{2,2} = \sum_{d=m}^{m'-1} \sum_{b=0}^{m'-1} s(m, d, b, z) + \sum_{d=0}^{m-1} \sum_{b=m}^{m'-1} s(m, d, b, z) + \sum_{d=0}^{m-1} \sum_{b=0}^{m-1} t(m, d, b, z)$$
$$q''_{2,3} = \sum_{d=m}^{m'-1} \sum_{b=0}^{m'-1} s(m', d, b, z) + \sum_{d=0}^{m-1} \sum_{b=m}^{m'-1} s(m', d, b, z) - \sum_{d=0}^{m'-1} \sum_{b=0}^{m'-1} t(m', d, b, z) + \sum_{d=0}^{m-1} \sum_{b=0}^{m-1} (s(m', d, b, z) - u(m', m' - m, d, b, z) + t(m', d, b, z))$$
$$q''_{3,1} = \sum_{d=m'}^{k-1} \sum_{b=0}^{k-1-d} r(d, b, z) + \sum_{d=0}^{m'-1} \sum_{b=m'}^{k-1-d} r(d, b, z)$$
$$q''_{3,2} = \sum_{d=m'}^{k-1} \sum_{b=0}^{k-1-d} s(m, d, b, z) + \sum_{d=0}^{m'-1} \sum_{b=m'}^{k-1-d} s(m, d, b, z)$$
$$q''_{3,3} = \sum_{d=m'}^{k-1} \sum_{b=0}^{k-1-d} s(m', d, b, z) + \sum_{d=0}^{m'-1} \sum_{b=m'}^{k-1-d} s(m', d, b, z) + \sum_{d=0}^{m'-1} \sum_{b=0}^{m'-1} t(m', d, b, z)$$

Having specified the sets of matches $Q''_{r,s}$ we can recursively build sets of collations $\mathcal{S}(i)$, $\mathcal{M}(i)$, $\mathcal{L}(i)$, as in the case of alphabet size 3. Let $\mathcal{H}(i) = \mathcal{S}(i) \cup \mathcal{M}(i) \cup \mathcal{L}(i)$.

Theorem 4.5 *Every nondominated collation is in* $\mathcal{H}(i)$.

Proof. The proof of this theorem is analogous to the proof of Theorem 4.3. □

The number of collations in $\mathcal{H}$ can be bounded using Corollary 4.1 for

$$\boldsymbol{Q}(z) = \begin{pmatrix} q''_{1,1}(z) & q''_{1,2(z)} & q''_{1,3}(z) \\ q''_{2,1}(z) & q''_{2,2(z)} & q''_{2,3}(z) \\ q''_{3,1}(z) & q''_{3,2}(z) & q''_{3,3}(z) \end{pmatrix}.$$

Now we can use Theorem 4.1 to get upper bounds for γ_k. For $k = 2, \ldots, 15$ these can be found in Figure 4.6 together with values m and m' that give the best results.

k	Upper bound	m	m'	z_0	k	Upper bound	m	m'	z_0
2	0.837623				9	0.553937	1	2	0.062187
3	0.765803	1	2	0.145256	10	0.534855	2	3	0.057651
4	0.708236	1	2	0.116369	11	0.517842	2	3	0.053505
5	0.664428	1	2	0.098038	12	0.502591	2	3	0.049949
6	0.629316	1	2	0.085203	13	0.488800	2	3	0.046863
7	0.600184	1	2	0.075633	14	0.476198	2	4	0.044248
8	0.575407	1	2	0.068181	15	0.464619	2	4	0.041868

Figure 4.6: The upper bounds for γ_k.

Chapter 5

Related problems

There are many ways to generalize the longest common subsequence problem. As the most natural we shall consider the expected length of a longest common subsequence of more than two sequences and we shall extend the methods from the previous chapters to cover it. These methods can be also adjusted for the case of shortest common supersequences and for the investigation of the adaptability of sequences. The chapter is closed with a small survey of longest common substring problem.

5.1 Several sequences

In the previous chapters we have described upper and lower bounds for the expected length of a longest common subsequence of two sequences. The natural generalization of this problem is to consider more than just two sequences.

Let Σ be an alphabet of size k. We shall work over the set, $\Psi^n = (\Sigma^n)^l$, of l-tuples of sequences of length n. We denote the set of all l-tuples of sequences by $\Psi = (\Sigma^*)^l$. The total length of the l-tuple $\boldsymbol{u} = (u_1, \ldots, u_l) \in \Psi$ is the

sum of lengths of all sequences from the l-tuple, i.e. $|\boldsymbol{u}| = l(u_1, \ldots, u_l) = |u_1| + \cdots + |u_l|$. Let $\mathbf{L}(\boldsymbol{u}) = \mathbf{L}(u_1, \ldots, u_l)$ be the length of the longest common subsequence of l-tuple $\boldsymbol{u} = (u_1, \ldots, u_l) \in \Psi$. Let $\mathbf{EL}_n^{(l)}$ be the expected value of $\mathbf{L}(u_1, \ldots, u_l)$ for random sequences $u_1, \ldots, u_l \in \Sigma^n$, i.e.

$$\mathbf{EL}_n^{(l)} = \frac{1}{k^{ln}} \sum_{\boldsymbol{u} \in \Psi^n} \mathbf{L}(\boldsymbol{u}) .$$

Similarly to the case of $l = 2$, the expected length $\mathbf{EL}_n^{(l)}$ is linear with respect to n.

Theorem 5.1 *For every $k \geq 2$ and every $l \geq 2$ there is $\gamma_k^{(l)}$ such that*

$$\gamma_k^{(l)} = \lim_{n \to \infty} \frac{\mathbf{EL}_n^{(l)}}{n} = \sup_n \frac{\mathbf{EL}_n^{(l)}}{n} .$$

Proof. The existence of constants $\gamma_k^{(l)}$ follows from the fact that $\mathbf{EL}_n^{(l)}$ is superadditive. Superadditivity of $\mathbf{EL}_n^{(l)}$ can be proved in the same manner as in Lemma 2.2. □

The methods from Chapter 4 can be used to obtain upper bounds for $\mathbf{EL}_n^{(l)}$. Let $F(i, n)$ be the number of l-tuples from Ψ^n with a longest common subsequence of length i, i.e.

$$F(i, n) = |\{\boldsymbol{u} \in \Psi^n : \mathbf{L}(\boldsymbol{u}) = i\}| .$$

Let $H(i, n)$ be any upper bound for $F(i, n)$. The following lemma, a natural generalization of Lemma 4.1, will be used to transform upper bounds for $F(i, n)$ to upper bounds for $\gamma_k^{(l)}$.

Lemma 5.1 *If y is such that $\sum_{i=\lceil yn \rceil}^{n} H(i, n) = o(k^{ln})$ then $\gamma_k^{(l)} \leq y$.*

Proof.

$$\begin{aligned}\frac{\mathbf{EL}_n^{(l)}}{n} &= \frac{1}{nk^{ln}}\sum_{i=0}^{n} iF(i,n)\\ &= \frac{1}{nk^{ln}}\left(\sum_{i=0}^{\lceil yn\rceil-1} iF(i,n)+\sum_{i=\lceil yn\rceil}^{n} iF(i,n)\right)\\ &\le \frac{1}{nk^{ln}}\left(yn\sum_{i=0}^{n}F(i,n)+n\sum_{i=\lceil yn\rceil}^{n}H(i,n)\right)=y+o(1)\end{aligned}$$

□

An l-tuple of sequences $(u_1,\ldots,u_l)\in\Psi$ is a match if $u_1,\ldots,u_l$ have the same last symbol. A collation of order i is a sequence of i matches followed by an l-tuple. We already know that collations are good objects to produce upper bounds for $F(i,n)$. Let $H(i,m)$ be the number of all collations of order i generating an l-tuple of total length m. For every l-tuple $\boldsymbol{u}$ with a longest common subsequence of length i there is at least one collation of order i generating $\boldsymbol{u}$, therefore $F(i,n)\le H(i,ln)$.

Theorem 5.2 *Let $h(z)=\frac{kz^l}{(1-kz)^l}$. Let z_0 be such that $h(z_0)<1$. Then*

$$\gamma_k^{(l)}\le\frac{\log(k^l z_0^l)}{\log h(z_0)}. \tag{5.1}$$

Proof. To get an upper bound for $F(i,n)$ we have to count $H(i,m)$. Again we shall use generating functions to achieve this. There are k possibilities for a matching symbol, so its contribution then is kz^l. The contribution of every sequence, not counting the matching symbol, is $\frac{1}{1-kz}$, therefore the generating function for the number of matches is

$$h(z)=\frac{kz^l}{(1-kz)^l}.$$

The generating function for the number of collations of order i then is $s(z)(h(z))^i$; where $s(z)$ is independent of i. Now we can bound $H(i,n)$.

$$\sum_{i=\lceil yn\rceil}^{n} H(i,ln) \le \sum_{i=\lceil yn\rceil}^{n} \frac{s(z_0)(h(z_0))^i}{z_0^{ln}} \le ns(z_0)\left(\frac{(h(z_0))^y}{z_0^l}\right)^n .$$

For every $y > \frac{\log(k^l z_0^l)}{\log h(z_0)}$ we have $\frac{(h(z_0))^y}{z_0^l} < k^l$ and therefore $\sum_{i=\lceil yn\rceil}^{n} H_{i,ln} = o(k^{ln})$. Using Lemma 5.1 we can conclude that $\gamma_k^{(l)} \le y$. □

For $l = 2$ the upper bound from Theorem 5.2 corresponds to the upper bound of Chvátal and Sankoff [CS75]. Now we can analyse the speed of convergence of the upper bounds (5.1).

Corollary 5.1 *For every $l \ge 2$ we have*

$$\lim_{k\to\infty} k^{1-1/l}\gamma_k^{(l)} \le e .$$

Proof. Let $z_0 = \frac{k^{1-1/l}-e}{k^{2-1/l}}$, then $kz_0 = 1 - ek^{1/l-1}$. For $h(z_0)$ we have

$$h(z_0) = \left(\frac{k^{1/l-1}kz_0}{1-kz_0}\right)^l ,$$

and after substitution of kz_0 we get

$$h(z_0) = \left(\frac{k^{1/l-1}(1-ek^{1/l-1})}{ek^{1/l-1}}\right)^l = \frac{(1-ek^{1/l-1})^l}{e^l} .$$

From (5.1) we have

$$k^{1-1/l}\gamma_k^{(l)} \le k^{1-1/l}\frac{\log(1-ek^{1/l-1})}{\log(1-ek^{1/l-1})-1} .$$

Since $\lim_{x\to 0+}\frac{\log(1+ax)}{x} = a$, we get

$$\begin{aligned}\lim_{k\to\infty} k^{1-1/l}\gamma_k^{(l)} &\le \lim_{k\to\infty}\left(\frac{\log(1-ek^{1/l-1})}{k^{1/l-1}}\cdot\frac{-1}{1-\log(1-ek^{1/l-1})}\right)\\ &\le (-e)(-1) = e .\end{aligned}$$

□

We can use the following algorithm $\mathfrak{K}$ to obtain lower bounds matching to within a constant factor. We shall scan the first input tape until $k^{1-1/l}$ different symbols are found. Let K_1 be the set of these symbols. Then we scan the second tape until $k^{1-2/l}$ different symbols from K_1 are read. We denote the set of symbols from K_1 scanned on the second tape by K_2. We shall continue in this way. Suppose we have determined the set K_{i-1}. We scan the i-th tape until $k^{1-i/l}$ different symbols from K_{i-1} are found. These symbols form the set K_i. Having specified K_{l-1}, we search for some symbol $a \in K_{l-1}$ on the last tape. Since

$$a \in K_{l-1} \subseteq K_{l-2} \subseteq \cdots \subseteq K_1 ,$$

we can put a into a common subsequence produced by algorithm $\mathfrak{K}$. Computing the average progress on each tape yields following lower bounds.

Theorem 5.3 *For every $l \geq 2$ we have*

$$\lim_{k\to\infty} k^{1-1/l}\gamma_k^{(l)} \geq 1 .$$

Proof. Let us consider the i-th tape. We have constructed the set K_{i-1} of $k^{1-(i-1)/l}$ symbols. We are reading the i-th tape until $k^{1-i/l}$ symbols from K_{i-1} are found. Let $\mathbf{EK}_i$ be the expected number of symbols read to achieve this. Since $\lim_{n\to\infty} \frac{\mathbf{EL}_n}{n} \geq \frac{1}{\max\{\mathbf{EK}_i : 1\leq i\leq l\}}$, to prove the theorem we have to show that for every i

$$\mathbf{EK}_i \leq k^{1-1/l} + o(k^{1-1/l}) .$$

The waiting time to collect one of j symbols is $1+(1-\frac{j}{k})+(1-\frac{j}{k})^2+\cdots = \frac{k}{j}$, therefore

$$\mathbf{EK}_i = \sum_{|K_{i-1}|-|K_i|}^{j=|K_{i-1}|} \frac{k}{j} .$$

There are $k^{1-i/l}$ summands and the largest of them is $\frac{k}{k^{1-(i-1)/l}-k^{1-i/l}}$, hence

$$\begin{aligned}\mathbf{EK}_i &\leq k^{1-i/l}\frac{k}{k^{1-(i-1)/l}-k^{1-i/l}}\\ &\leq k^{1-1/l}\frac{1}{1-k^{-1/l}}\\ &\leq k^{1-1/l}(1+o(1)) = k^{1-1/l}+o(k^{1-1/l})\,.\end{aligned}$$

□

In [Ste86] Steele conjectures that $\gamma_k^{(l)} = \gamma_k^{l-1}$ for $l > 2$. As we can see from Corollary 5.1 and Theorem 5.3 this is not the case.

5.2 Super-, nonsub-, and nonsupersequences

Our major topic is longest common subsequences, however, there is a 'dual' notion, namely shortest common supersequences.

Definition 5.1 Let $u, v \in \Sigma^*$. We say, that $w \in \Sigma^*$ is a *common supersequence* of u and v, if $u \sqsubseteq w$ and $v \sqsubseteq w$. We say, that $w \in \Sigma^*$ is a *shortest common supersequence* (SCSS) of u and v, if w is a common supersequence of u and v and for every $w' \in \Sigma^*$ the following condition is true:

$$u \sqsubseteq w' \,\&\, v \sqsubseteq w' \Longrightarrow |w| \leq |w'|\,.$$

The length of a shortest common supersequence is denoted by $\mathbf{S}(u, v)$. Creating shortest common supersequences is natural when merging sequences. This is useful for some types of compression [Sto88] or for efficient planning [FLY92]. Longest common subsequences and shortest common supersequences are dual in the following sense.

Lemma 5.2 *For every* $u, v \in \Sigma^*$

$$\mathbf{S}(u, v) + \mathbf{L}(u, v) = |u| + |v| .$$

Proof. The lemma is a consequence of the simple property of finite sets:

$$|A| + |B| = |A \cap B| + |A \cup B| .$$

□

Corollary 5.2 *Let* $\mathbf{ES}_n$ *be the expected length of a shortest common supersequence of two random sequences from* Σ^n. *Then* $\lim_{n\to\infty} \frac{\mathbf{ES}_n}{n}$ *exists and its value is* $2 - \gamma_k$.

For more sequences $u_1, \ldots, u_l$ the length of their shortest common supersequence is denoted by $\mathbf{S}(u_1, \ldots, u_l)$. If we have more than two sequences and we know only their lengths and the length of their longest common subsequence then we are not able to compute the length of a shortest common supersequence.

Example 5.1 We have $\mathbf{L}(11, 22, 33) = \mathbf{L}(12, 23, 31) = 0$, but $\mathbf{S}(11, 22, 33) = 6$ and $\mathbf{S}(12, 23, 31) = 4$. ◇

We can define $\mathbf{ES}_n^{(l)}$ – the expected length of a shortest common supersequence of l sequences of length n analogously to the $\mathbf{EL}_n^{(l)}$. Properties of $\mathbf{ES}_n^{(l)}$ are similar and there are constants $\sigma_k^{(l)} = \lim_{n\to\infty} \frac{\mathbf{ES}_n^{(l)}}{n}$.

The expected length of a shortest common supersequence does not appear to have been investigated in the literature.

To obtain lower bounds for $\sigma_k^{(l)}$ we shall use methods similar to the methods from Chapter 4. The following lemma corresponds to Lemma 4.1.

Lemma 5.3 *Let* $F(i,n) = |\{\boldsymbol{u} \in \Psi^n : \mathbf{S}(\boldsymbol{u}) = i\}|$ *and let* $H(i,n)$ *be any upper bound for* $F(i,n)$. *If* y *is such that* $\sum_{i=0}^{\lfloor yn \rfloor} H(i,n) = o(k^{ln})$ *then* $y \le \sigma_k^{(l)}$.

For common supersequences the role of matches will be played by 'placements' and a 'distribution' will be the notion analogous to a collation.

Definition 5.2 An l-tuple $u_1, \ldots, u_l$ is a *placement* if $u_1, \ldots, u_l \in \{\lambda, a\}$ for some $a \in \Sigma$ and not all of $u_1, \ldots, u_l$ are empty sequences. A sequence of i placements will be called a *distribution* of order i. Distribution $\mathbf{d}$ *generates* l-tuple $\boldsymbol{u}$ if $\mathrm{cat}(\mathbf{d}) = \boldsymbol{u}$.

Let $\mathcal{D}(i)$ be the set of all distributions of order i. Let $\mathcal{G}(m)$ be the set of all distributions generating an l-tuple of total length m, i.e. $\mathcal{G}(m) = \{\mathbf{d} : |\mathrm{cat}(\mathbf{d})| = m\}$. Since for every l-tuple $\boldsymbol{u} \in \Psi^n$, there exists a distribution of order $\mathbf{S}(\boldsymbol{u})$ that generates $\boldsymbol{u}$, we have

$$F(i,n) \le |\mathcal{G}(ln) \cap \mathcal{D}(i)| = G(i,n)\,.$$

Every placement of total length m corresponds to a nonempty subset of $\{1, \ldots, l\}$ of size m, thus the generating function for the number of placements is $k((1+z)^l - 1)$. The generating function for the number of distributions in the $\mathcal{D}(i)$ then is $k^i((1+z)^l - 1)^i$. From (2.1) for $z_0 > 2^{1/l} - 1$ and $y < \frac{l \log(z_0 k)}{\log(k(1+z_0)^l - k)}$ we have

$$\begin{aligned}
\sum_{i=0}^{\lfloor yn \rfloor} G(i,n) &\le \sum_{i=0}^{\lfloor yn \rfloor} \inf_{z \in Z} \frac{k^i((1+z)^l - 1)^i}{z^{ln}} \\
&\le \inf_{z \in Z} \sum_{i=0}^{\lfloor yn \rfloor} \frac{k^i((1+z)^l - 1)^i}{z^{ln}} \\
&\le \sum_{i=0}^{\lfloor yn \rfloor} \frac{k^i((1+z_0)^l - 1)^i}{z_0^{ln}} \\
&\le yn \left(\frac{k^y((1+z_0)^l - 1)^y}{z_0^l} \right)^n = o(k^{ln})\,.
\end{aligned}$$

Thus we have just proved the following theorem.

Theorem 5.4 *For every $l \geq 2$ and $k \geq 2$ we have*

$$\sigma_k^{(l)} \geq \frac{l \log z_0 k}{\log k((1+z_0)^l - 1)},$$

where $z_0 \in \mathbb{R}$ is such that $z_0 > 2^{1/l} - 1$.

To obtain upper bounds for the expected length of a shortest common supersequence of l sequences we shall design an algorithm which produces a common supersequence of its input. If the algorithm is simple enough, we can analyse its behaviour for a random input. The expected length of a common supersequence produced by the algorithm will be an upper bound for $\sigma_k^{(l)}$.

We can use a tournament-style algorithm $\mathfrak{T}$ from Figure 5.1 to produce a common supersequence of l sequences $u_1, \ldots, u_l$. In the first round u_1 and u_2 produce a common supersequence $v_{1,1}$, u_3 and u_4 produce a common supersequence $v_{1,2}$, and so on. In the second round we repeat the process with $v_{1,1}, \ldots, v_{1,\lceil l/2 \rceil}$. After $\lceil \log l \rceil$ steps we get a 'winner' $v_{\lceil \log l \rceil,1}$ — a common supersequence of $u_1, \ldots, u_l$. This is a consequence of the simple inductive argument saying that u_l is subsequence of $v_{i,j}$ for $l = (j-1)2^i + 1, \ldots, j2^i$.

Timkovskij [Tim89] has asked a question about how good an approximation is the tournament algorithm. Bradford and Jenkyns [BJ91] have found an example of three sequences of length 12 over the alphabet of size nine such that no tournament-like algorithm can compute their shortest common supersequence. They also asked the question of when the tournament algorithm is not optimal. As a partial answer to this question we shall describe an algorithm that produces shorter common supersequences when the alphabet size k is relatively small compared to the number of sequences l.

algorithm $\mathfrak{T}$

input $u_1, \ldots, u_l \in \Sigma^*$
output $w \in \Sigma^*$ – common supersequence of $u_1, \ldots, u_l$

begin
$m := l$;
for $i := 1$ **to** $\lceil \log l \rceil$ **do**
for $j = 1$ **to** m **do**
$v_{i,j} := \mathbf{S}(v_{i-1,2j-1}, v_{i-1,2j})$
endfor;
$m := \lceil m/2 \rceil$
endfor;
output $v_{\lceil \log l \rceil, 1}$
end.

Figure 5.1: The tournament algorithm for producing common supersequences.

Suppose every input sequence has a marker marking an active symbol. By 'processed prefix' we shall understand a prefix of the input sequence up to, but not including, the active symbol. At the start the first symbol of every input is the active symbol.

In every step of algorithm the input sequence with shortest processed prefix is selected. In the case of unequal input sequences the one with the smallest proportion of processed prefix will be selected. Let a be the active symbol of the selected input sequence. Symbol a is added to the output sequence and a marker on every input sequence having an active symbol a is advanced to the next symbol. This will ensure that after every step of the algorithm the output sequence is a common supersequence of processed prefixes. The entire algorithm can be found in Figure 5.2.

We shall express the expected progress of markers corresponding to one output symbol. The expected progress on the selected tape is 1. The expected

```
algorithm 𝔖
    input   u_1, ..., u_l ∈ Σ*
    output w ∈ Σ* – common supersequence of u_1, ..., u_l
begin
    while i_1 < |u_1| ∨ ... ∨ i_l < |u_l| do
        m := 1;
        for j = 2 to l do
            if i_j < i_m then  (i_j/|u_j| < i_m/|u_m|)
                m := j;
            endif
        endfor;
        a := u_m[i_m];
        output a;
        for j = 1 to l do
            if u_j[i_j] = a then
                i_j := i_j + 1;
            endif
        endfor
    endwhile
end.
```

Figure 5.2: An algorithm producing a common supersequence.

progress on the remaining $l-1$ tapes is $\frac{1}{k}$. Therefore the overall expected progress on the input tapes is $1+\frac{l-1}{k}$ symbols. There are ln input symbols, hence the expected length of a common subsequence produced by algorithm $\mathfrak{S}$ is

$$\frac{ln}{1+\frac{l-1}{k}} = \frac{lkn}{l+k-1}.$$

We summarize upper bounds in the following theorem.

Theorem 5.5 *For every $l > 3$ and $k \geq 2$ we have*

$$\sigma_k^{(l)} \leq \frac{lk}{l+k-1}.$$

k	$l=2$	$l=3$	$l=4$	$l=5$	$l=6$	$l=8$	$l=16$
2	1.33333	1.50000	1.60000	1.66667	1.71429	1.77778	1.88236
	1.09321	1.14253	1.17267	1.19287	1.20731	1.22656	1.25822
3	1.38462	1.80000	2.00000	2.14286	2.25000	2.40000	2.66667
	1.16377	1.25982	1.32366	1.36923	1.40336	1.45107	1.53569
4	1.45455	2.00000	2.28572	2.50000	2.66667	2.90910	3.36843
	1.21514	1.34913	1.44278	1.51244	1.56642	1.64476	1.79311
5	1.49385	2.14286	2.50000	2.77778	3.00000	3.33333	4.00000
	1.25447	1.41951	1.53883	1.63017	1.70270	1.81106	2.02791
6	1.52831	2.25000	2.66667	3.00000	3.27273	3.69231	4.57143
	1.28583	1.47679	1.61830	1.72893	1.81841	1.95512	2.24164
7	1.55498	2.33334	2.80000	3.18182	3.50000	4.00000	5.09091
	1.31162	1.52463	1.68549	1.81331	1.91818	2.08131	2.43661
8	1.57763	2.40000	2.90910	3.33333	3.69231	4.26667	5.56522
	1.33333	1.56541	1.74333	1.88656	2.00542	2.19301	2.61512
9	1.59679	2.45455	3.00000	3.46154	3.85715	4.50000	6.00000
	1.35196	1.60076	1.79387	1.95098	2.08260	2.29285	2.77923
10	1.61344	2.50000	3.07693	3.57143	4.00000	4.70589	6.40000
	1.36819	1.63183	1.83857	2.00828	2.15158	2.38282	2.93073
11	1.62804	2.53847	3.14286	3.66667	4.12500	4.88889	6.76924
	1.38251	1.65943	1.87851	2.05972	2.21377	2.46452	3.07115
12	1.64101	2.57143	3.20000	3.75000	4.23530	5.05264	7.11111
	1.39527	1.68418	1.91451	2.10627	2.27025	2.53919	3.20180
13	1.65263	2.60000	3.25000	3.82353	4.33333	5.20000	7.42858
	1.40675	1.70657	1.94721	2.14871	2.32189	2.60782	3.32378
14	1.66313	2.62500	3.29412	3.88889	4.42106	5.33334	7.72414
	1.41715	1.72696	1.97710	2.18762	2.36938	2.67122	3.43804
15	1.67268	2.64706	3.33334	3.94737	4.50000	5.45455	8.00000
	1.42664	1.74564	2.00458	2.22349	2.41326	2.73007	3.54540

Figure 5.3: Upper and lower bounds for the expected length of a shortest common supersequence of l sequences over an alphabet of size k.

Upper and lower bounds for $k = 2, \ldots, 15$ and $l = 2, 3, 4, 5, 6, 8, 16$ obtained using Theorems 5.4 and 5.5 can be found in the table in Figure 5.3. Better bounds for $l = 2$ follow from bounds for longest common subsequences

using duality properties.

Shortest common supersequences can be computed using the dynamic programming algorithm. This algorithm is not polynomial when the number of sequences is unbounded. Maier [Mai78] has shown that the shortest common supersequence problem for $k \geq 5$ and the longest common subsequence problem for $k \geq 2$ are NP-complete. Räihä and Ukkonen [RU81] have strengthened Maier's result and have shown the shortest common supersequence problem to be NP-complete for $k \geq 2$. Middendorf [Mid94] has shown that the shortest common supersequence problem over the binary alphabet remains NP-complete even if all the given sequences have the same length and each of them contains exactly two ones.

Even approximating longest common subsequences and shortest common supersequences is not much simpler. Papadimitriou and Yannakakis [PY88] have defined a class of MAX SNP problems, an approximation analogue of NP problems. Jiang and Li [JL94] have shown that the approximation versions of longest common subsequence and shortest common supersequence problems are MAX SNP-hard, i.e. no polynomial-time algorithm can achieve approximation ratio $1 + \varepsilon$, unless P=NP.

We ought to mention also some interesting variations of common subsequences and supersequences. Timkovskij [Tim89] has considered negative variants, namely common nonsubsequences and nonsupersequences. Sequence v is a *common nonsubsequence* of $u_1, \ldots, u_l$ if, for every $i = 1, \ldots, l$, sequence v is not a subsequence of u_i. Similarly sequence v is a common nonsupersequence of $u_1, \ldots, u_l$ if, for every $i = 1, \ldots, l$, sequence u_i is not a subsequence of v. We are then interested in finding shortest common nonsubsequences and longest common nonsupersequences. An interesting feature of common nonsupersequences is that there are l-tuples of sequences such that their longest

common nonsupersequences does not exist. Timkovskij [Tim89] has shown that the problem of the existence of a longest common nonsupersequence with polynomially bounded length is in the class Π_2^p of the polynomial hierarchy (as defined by Stockmeyer [Sto77]). Middendorf [Mid93] has proved that the shortest common nonsubsequence problem is NP-complete for $k \geq 2$. He also proved the NP-completeness of the similar problem of shortest distinguishing sequence. Sequence w is said to *distinguish* two sequences u and v if w is a subsequence of one of them and is not a subsequence of the other. Sequence w distinguishes sequence u and a set of sequences $\{v_1, \ldots, v_l\}$, if it distinguishes u and v_i for every $i = 1, \ldots, l$. The dynamic programming algorithm for finding shortest distinguishing sequence of two sequences was described by Hebrard [Heb91].

We can change the view on longest common subsequences and consider maximal common subsequences. Sequence v is a *maximal common subsequence* of $u_1, \ldots, u_l$ if v is a common subsequence of $u_1, \ldots, u_l$ and no (proper) supersequence of v is a common subsequence. Every longest common subsequence is also a maximal common subsequence, but there may exist maximal common subsequences that are not longest common subsequences. The corresponding dual notion is a *minimal common supersequence*. Irving and Fraser [IF94] have described a dynamic programming algorithm that solves the maximal common subsequence and minimal common supersequence problems for a constant number of sequences in polynomial time. They also have shown the NP-completeness of the shortest maximal common subsequence problem. The question of the complexity of the longest minimal common supersequence problem remains open.

5.3 Adaptability

We have defined the expected length of a longest common subsequence as the average over all pairs of sequences. We can separate the averaging process so that we first take the average with respect to the first sequence and then with respect to the second sequence. Thus we get following formula:

$$\mathbf{EL}_n = \frac{1}{k^n} \sum_{u \in \Sigma^n} \left(\frac{1}{k^n} \sum_{v \in \Sigma^n} \mathbf{L}(u, v) \right) .$$

We can investigate the contribution of each sequence u to the average. This yields the notion of 'adaptability'.

Definition 5.3 For every $u \in \Sigma^*$ we define the *adaptability* of sequence u of length n by

$$A(u) = \frac{1}{k^n} \sum_{v \in \Sigma^n} \mathbf{L}(u, v).$$

It is quite natural that different sequences can have different adaptabilities. The expected length of a longest common subsequence is the average of adaptabilities. We can also raise the questions of what are the the maximal and the minimal adaptabilities.

Definition 5.4 For every $n \geq 0$ we define *minimal adaptability* $\mathbf{A}_n$ and *maximal adaptability* $\mathbf{B}_n$ by

$$\mathbf{A}_n = \min\{A(u) : u \in \Sigma^n\} ,$$

$$\mathbf{B}_n = \max\{A(u) : u \in \Sigma^n\} .$$

Minimal and maximal adaptabilities are strict bounds for the expected length of a longest common subsequence.

Lemma 5.4 *For every $k \geq 2$ and $n \geq 2$ we have*

$$\mathbf{A}_n < \mathbf{EL}_n < \mathbf{B}_n .$$

Proof. Since $\mathbf{EL}_n = \frac{1}{k^n} \sum_{|u|=n} A(u)$, we have directly $\mathbf{A}_n \leq \mathbf{EL}_n \leq \mathbf{B}_n$. The strictness of the inequalities is a consequence of the fact that not all sequences have the same adaptability. □

First we shall investigate minimal adaptability.

Theorem 5.6 *For every $n, k \in \mathbb{N}$, $n > 0$, $k > 1$ we have*

$$\mathbf{A}_n = \frac{n}{k} .$$

Proof. We define an equivalence relation for sequences; two sequences are equivalent, if one can be obtained from the other by a cyclic permutation of symbols of Σ. This splits Σ^n into $K = k^{n-1}$ equivalence classes $S_1, \ldots, S_K$ of size k. If we compare sequences from one class, we can observe, that they have always different symbols in the same position. Therefore for every $u \in \Sigma^n$

$$\sum_{v \in S_i} \mathbf{L}(u, v) \geq n ,$$

hence

$$A(u) = \frac{1}{k^n} \sum_{i=1}^{K} \sum_{v \in S_i} \mathbf{L}(u, v) \geq \frac{1}{k^n} \sum_{i=1}^{K} n \geq \frac{n}{k} .$$

Since $A(00\ldots0) = \frac{n}{k}$, we have immediately $\mathbf{A}_n = \frac{n}{k}$. □

If we define α_k analogously to γ_k (2.4), we get

$$\alpha_k = \lim_{n \to \infty} \frac{\mathbf{A}_n}{n} = \frac{1}{k} .$$

The gap between α_k and γ_k is thus 'quadratic'.

The case of maximal adaptability is more similar to the expected length of a longest common subsequence. Exact values of maximal adaptability are not known, but we shall describe here some upper and lower bounds.

Lemma 5.5 *Maximal adaptability is superadditive, i.e.,*

$$\mathbf{B}_m + \mathbf{B}_n \leq \mathbf{B}_{m+n} .$$

Proof. Let $u \in \Sigma^m$ and $v \in \Sigma^n$ be sequences with maximal adaptabilities. Following the proof of Lemma 2.5 we can easily prove

$$A(u) + A(v) \leq A(uv) .$$

Combining this with the definition of the maximal adaptability we complete the proof of the lemma. □

As a consequence of the superadditivity of maximal adaptability we get the following theorem.

Theorem 5.7 *For every k there is a constant β_k such that*

$$\beta_k = \lim_{n \to \infty} \frac{\mathbf{B}_n}{n} .$$

Moreover $\gamma_k \leq \beta_k$.

Proof. The proof of the first part is the same as the proof of Theorem 2.1. The inequality between γ_k and β_k is a trivial consequence of Lemma 5.4. □

n	$\mathbf{B}_n^{(2)}$	'witness'	n	$\mathbf{B}_n^{(2)}$	'witness'
1	0.5	0	7	0.72321429	010 0110
2	0.625	10	8	0.734375	10 0110 01
3	0.66666667	010	9	0.73763021	010 0110 01
4	0.6875	10 10	10	0.7453125	10 0110 1001
		1001	11	0.74818004	010 0110 1001
5	0.7	010 10	12	0.75423177	10 0110 1001 10
		01001	13	0.75618803	010 0110 1001 10
6	0.71614583	10 0110	14	0.76020595	10 0110 1001 0110

Figure 5.4: Maximal adaptability for $k = 2$ and $n = 1, \ldots, 14$.

The table in Figure 5.4 gives values of $\mathbf{B}_n$ for $n = 1, \ldots, 14$ and $k = 2$ together with a "witness" sequence. This allows us to guess that a sequence with maximal adaptability for $k = 2$ could have form $u(01101001)^* v$ for some 'constant' sequences $u, v \in \Sigma^*$, i.e. u and v depend only on n mod 8. We can build a css machine and thus get a lower bound for the adaptability of $(01101001)^t$. This will serve as a lower bound for β_2, too.

Theorem 5.8 *Let* $C_i = A((01101001)^i)$. *then*

$$0.800350 \leq \lim_{i \to \infty} \frac{C_i}{8i} \leq \beta_2$$

Proof. A css machine giving this lower bound can be found in Appendix C. Since the situation is not symmetric, we have to analyse progress on the top and the bottom tapes separately. The lower bound for β_2 will be the minimum of the lower bounds given by each tape. □

For the alphabet of size k we predict that the adaptability of sequence of the form $u = (0\,1 \ldots k-1)^i$ is quite close to the maximal adaptability. The

following strategy can be used to obtain lower bounds for the adaptability of sequence u. Suppose u is on the top tape and a random input is on the bottom tape. First we read j symbols from the top tape and than we scan the bottom tape until we find a match with one of j symbols from the top tape. We continue in this manner until the input is exhausted.

The waiting time to collect one of the j symbols on the bottom tape is $1 + \frac{k-j}{k} + (\frac{k-j}{k})^2 + \cdots = \frac{k}{j}$. The expected progress on the top tape is $\frac{j+1}{2}$. Hence the expected length of a common subsequence produced is $\min\{\frac{2}{j+1}, \frac{j}{k}\}$. For $j = \frac{1}{2}(\sqrt{8k+1} - 1)$ we have $\frac{2}{j+1} = \frac{j}{k}$, therefore we can choose $j_f = \left\lfloor \frac{1}{2}(\sqrt{8k+1} - 1) \right\rfloor$ or $j_c = \left\lceil \frac{1}{2}(\sqrt{8k+1} - 1) \right\rceil$ depending which gives better bounds.

We can achieve larger progress by alternating j_f and j_c in a suitable ratio r. Then the top tape progress is

$$\frac{j_f + 1}{2} r + \frac{j_c + 1}{2}(1 - r). \tag{5.2}$$

The bottom tape progress corresponding to the ratio r is

$$\frac{k}{j_f} r + \frac{k}{j_c}(1 - r). \tag{5.3}$$

The minimal progress (corresponding to the maximal lower bound) will be achieved when (5.2) and (5.3) are equal, hence

$$r = \frac{j_c j_f (j_c + 1) - 2 j_f k}{(j_c - j_f)(j_c j_f + 2k)}.$$

Having expressed r we can compute lower bounds.

Theorem 5.9 *Let* $j_f = \left\lfloor \frac{1}{2}(\sqrt{8k+1} - 1) \right\rfloor$ *and* $j_c = \left\lceil \frac{1}{2}(\sqrt{8k+1} - 1) \right\rceil$. *Then*

$$\beta_k \geq \frac{j_c j_f + 2k}{k(j_c + j_f + 1)}.$$

The lower bound from Theorem 5.9 is assymptotically $\sqrt{2}/\sqrt{k}$. However, for large alphabets this bound can be improved.

Theorem 5.10 *We have*

$$\lim_{k\to\infty} \beta_k \sqrt{k} \geq 2 .$$

Proof. Let u be a sequence of the form $(0\,1\,\ldots\,k-1)^i$, $|u| = ki$. We shall actually show, that

$$\lim_{k\to\infty} \left(\lim_{i\to\infty} \frac{A(u)}{k\,i} \right) \sqrt{k} \geq 2 .$$

This bound can be achieved by a simple algorithm, that in one cycle reads k symbols from the top and the bottom tapes and computes a longest common subsequence of the read sequences. This cycle is repeated until the input is exhausted. The common subsequence obtained in one cycle is simply the longest increasing subsequence of the sequence read from the bottom tape. From the work of Logan and Shepp [LS77] it is known that the ratio between the expected length of a longest increasing subsequence and $\sqrt{k}$ is at least 2 as $k \to \infty$ (actually, Vershik and Kerov [VK77] have shown this rate to be exactly 2). This means, that $\lim_{k\to\infty} \beta_k \sqrt{k} \geq 2$. □

Methods developed in Chapter 4 can be used to obtain upper bounds for maximal adaptability. In the case of adaptability we have lost the symmetry between tapes, therefore we have to adjust the method.

Lemma 5.6 *Let $H(i,m,n)$ be such that*

$$H(i,m,n) \geq \max_{u\in\Sigma^m} |\{v \in \Sigma^n : \mathbf{L}(u,v) = i\}| .$$

If y is such that $\sum_{i=\lceil yn \rceil}^{n} H(i,n) = o(k^n)$, then $\beta_k \leq y$.

Proof. The proof is analogous to the proof of Lemma 4.1. Let u_n be a sequence from Σ^n with maximal adaptability. Let $F(i,n) = |\{v \in \Sigma^n : \mathbf{L}(u,v) = i\}|$. Since $H(i,n,n) \geq F(i,n)$, we have

$$\begin{aligned}
\frac{\mathbf{B}_n}{n} &= \frac{1}{nk^n}\sum_{i=0}^{n} iF(i,n) \\
&= \frac{1}{nk^n}\left(\sum_{i=0}^{\lceil yn\rceil-1} iF(i,n) + \sum_{i=\lceil yn\rceil}^{n} iF(i,n)\right) \\
&\leq \frac{1}{nk^n}\left(yn\sum_{i=0}^{n} F(i,n) + n\sum_{i=\lceil yn\rceil}^{n} H(i,n,n)\right) = y + o(1)\,.
\end{aligned}$$

□

To prove upper bounds for maximal adaptability we need a notion of 'quasiminimal' match.

Definition 5.5 Let $p = \binom{ua}{va}$ be a match satisfying following conditions:

1. $a \not\sqsubseteq u$,
2. $a \not\sqsubseteq v$,
3. if $|u| > 0$ then $u(1) \not\sqsubseteq v$.

Then match p is *quasiminimal.*

Let u be a fixed sequence of length s. Let $q_{s,t}$ be the number of quasiminimal matches $\binom{u}{v}$ with $|v| = t$. The generating function for the number of quasiminimal matches with $s = 1$ is $x\frac{z}{1-(k-1)z}$. The generating function for the number of quasiminimal matches with $s > 1$ is $\frac{x^2}{1-x}\frac{z}{1-(k-2)z}$. The generating function for quasiminimal matches then is

$$q(x,z) = \sum_{s,t=0}^{\infty} q_{s,t}x^s z^t = \frac{xz}{1-(k-1)z} + \frac{x^2}{1-x}\frac{z}{1-(k-2)z}.$$

The number of quasiminimal matches is dependent only on the length of u and not on the structure of u.

Theorem 5.11 *Let* $q(x,y) = \frac{xz}{1-(k-1)z} + \frac{x^2}{1-x}\frac{z}{1-(k-2)z}$. *Let* x_0, z_0 *be such that* $q(x_0,z_0) < 1$. *Then*

$$\beta_k \leq \frac{\log(kx_0z_0)}{\log q(x_0,z_0)}.$$

Proof. This proof follows the upper bound technique from Chapter 4. Let u be a fixed sequence from Σ^m. Let $\mathbf{p}$ be a collation of order i generating a pair $\binom{u}{v}$ for some $v \in \Sigma^n$ such that every match in $\mathbf{p}$ is quasiminimal. Let $\mathcal{Q}(i,u,n)$ be the set of all such collations. Since for every $v \in \Sigma^n$ with $\mathbf{L}(u,v) = i$ there is a collation in $\mathcal{Q}(i,u,n)$ generating $\binom{u}{v}$, we have

$$|\mathcal{Q}(i,u,n)| \geq |\{v \in \Sigma^n : \mathbf{L}(u,v) = i\}|$$

for every $u \in \Sigma^m$. The number of collations in $\mathcal{Q}(i,u,n)$ is dependent only on the length of u and we can put $Q(i,m,n) = |\mathcal{Q}(i,u,n)|$. The generating function for $Q(i,m,n)$ is $s(x,z)(q(x,z))^i$.

If $y > \frac{\log kx_0z_0}{\log q(x_0,z_0)}$, then $\frac{(q(x_0,z_0))^y}{x_0z_0} < k$ and

$$\begin{aligned}\sum_{i=\lceil yn\rceil}^{n} H_{i,n,n} &\leq \sum_{i=\lceil yn\rceil}^{n} \frac{(q(x_0,z_0))^i}{x_0z_0} \\ &\leq n\left(\frac{(q(x_0,z_0))^y}{x_0z_0}\right)^n = o(k^n).\end{aligned}$$

Therefore $\beta_k \leq y$ for every $y > \frac{\log kx_0z_0}{\log q(x_0,z_0)}$. □

Bounds obtained by Theorems 5.9 and 5.11 are given in the table in Figure 5.5. For $k = 2$ we have better bounds from Theorem 5.8.

5.4 Longest common substrings

Let u and v be two sequences from Σ^*. Sequence w is a common substring of u and v if w is a substring of both u and v. A *longest common substring* of

k	lower bound		upper bound	k	lower bound	upper bound
2	0.75000	(0.80035)	0.88693	9	0.41667	0.58566
3	0.66667		0.80638	10	0.40000	0.56569
4	0.58333		0.74760	11	0.38182	0.54791
5	0.53333		0.70205	12	0.36667	0.53194
6	0.50000		0.66522	13	0.35385	0.51747
7	0.46429		0.63453	14	0.34286	0.50427
8	0.43750		0.60836	15	0.33333	0.49216

Figure 5.5: Maximal adaptability for $k = 2, \ldots, 15$.

u and v is a common substring with maximal possible length. We shall denote this length by $\mathbf{W}(u, v)$. The expected length of a longest common substring is the average of $\mathbf{W}(u, v)$ over all pairs $u, v \in \Sigma^n$, i.e.,

$$\mathbf{EW}_n = \frac{1}{k^{2n}} \sum_{u,v \in \Sigma^n} \mathbf{W}(u, v) . \tag{5.4}$$

While the expected length of a longest common subsequence is linear, the expected length of a longest common substring is logarithmic. Arratia and Waterman [AW85] have proved following theorem.

Theorem 5.12 (Arratia and Waterman [AW85]) *For every $k \geq 2$*

$$\lim_{n \to \infty} \frac{\mathbf{EW}_n}{\log_k n} = 2 .$$

Alignments with mismatches are especially of interest to molecular biologists. Let $u, v \in \Sigma^n$. Sequence w is a common substring of u and v with m mismatches if there are $S \subseteq \{0, 1, \ldots, |w| - 1\}$ and $i, j \in \{1, \ldots, n - |w| + m\}$ such that $|S| \geq |w| - m$ and $w(s) = u(i + s) = v(j + s)$ for all $s \in S$. Let

$\mathbf{M}^m(u,v)$ be the length of a longest common substring of u and v with m mismatches. Let $\mathbf{EM}_n^m$ be the expected value of $\mathbf{M}^m(u,v)$, defined analogously to (5.4). Arratia et al. [AGW86] have evaluated the $\mathbf{EM}_n^m$.

Theorem 5.13 (Arratia et al. [AGW86]) *For a fixed m and large n the expected value of $\mathbf{M}^m(u,v)$ is*

$$\mathbf{EM}_n^m = \log_k(n^2) + m \log_k \log_k(n^2) + O(1)\,. \tag{5.5}$$

If we allow the number of mismatches to be large, the expression (5.5) is not sufficient. For every pair $u, v \in \Sigma^n$ and $r \in \mathbb{R}$ we define $\mathbf{N}^r(u,v)$ to be the length of a longest common substring of u and v with m mismatches such that $m \leq r\mathbf{N}^r(u,v)$. Let $\mathbf{EN}_n^r$ be the expected value of $\mathbf{N}^r(u,v)$ for random sequences u and v from Σ^n. Arratia and Waterman [AW89] consider the expected value of $\mathbf{N}^r(u,v)$.

Theorem 5.14 (Arratia and Waterman [AW89]) *For $r < 1 - \frac{1}{k}$ the expected value of $\mathbf{N}^r(u,v)$ is approximately*

$$\lim_{n\to\infty} \frac{\mathbf{EN}_n^r}{\log n} = \frac{2}{H(1-r, 1/k)}\,,$$

where $H(x,y) = x \log \frac{x}{y} + (1-x) \log \frac{1-x}{1-y}$ is the relative entropy function.

Since for sequences of length n we can expect $\frac{n}{k}$ symbols to be matched, we have $\mathbf{EN}_n^r = n$ for $r \geq 1 - \frac{1}{k}$. To obtain more realistic results, we shall consider insertions and deletions together with mismatches.

We define $m(p) = \min\{|t(p)|, |b(p)|\}$ and $M(p) = \max\{|t(p)|, |b(p)|\}$ for every pair p. The number of mismatches in match p is then $n(p) = m(p) - 1$

and the number of deletions/insertions is $N(p) = M(p) - m(p)$. Let $r, s \geq 0$ be real parameters. For every collation $\mathbf{p}$ of order i and weights r, s we define a *weight of collation* by

$$H(\mathbf{p}; r, s) = \begin{cases} i - r\sum_{j=1}^{i} n(p_j) - rm(p_{i+1}) - s\sum_{j=1}^{i+1} N(p_j) & \text{if } r \leq 2s \\ i - s\sum_{j=1}^{i} (|p_j| - 2) - s|p_{i+1}| & \text{if } r > 2s \end{cases}$$

Let $\mathcal{H}(u, v)$ be the the set of all collations generating substrings of $u, v \in \Sigma^n$, i.e.,

$$\mathcal{H}(u, v) = \left\{ \mathbf{p} : \text{cat}(\mathbf{p}) = \binom{u(i..i')}{v(j..j')} \text{ for some } 1 \leq i \leq i' \leq n, 1 \leq j \leq j' \leq n \right\}.$$

For $u, v \in \Sigma^n$ and $r, s \in \mathbb{R}$ we can defined the length of a *heaviest common substring* of u and v by

$$\mathbf{H}(u, v; r, s) = \max \{ H(\mathbf{p}; r, s) : \mathbf{p} \in \mathcal{H}(u, v) \}.$$

$\mathbf{H}(u, v; 0, 0)$ is then the length of a longest common subsequence of u and v and $\mathbf{H}(u, v; \infty, \infty)$ is the length of a longest common substring. Let $\mathbf{EH}_n(r, s)$ be the expected value of $\mathbf{H}(u, v; r, s)$ for random sequences $u, v \in \Sigma^n$. Waterman et al [WGA87] have shown that, similarly as in the case of $\mathbf{EM}_n^r$, there is a strict boundary separating parameters leading to logarithmic or linear expected length. This is illustrated by the diagram in Figure 5.6.

Theorem 5.15 (Waterman et al [WGA87]) *There is a set $C \subset \mathbb{R}^2$ of critical points such that for every $(r_c, s_c) \in C$*

$$\mathbf{EH}_n(r, s) = \begin{cases} \rho_{r,s} n & \text{for every } r, s, \ 0 \leq r < r_c \text{ and } 0 \leq s < s_c \\ \sigma_{r,s} \log n & \text{for every } r, s, \ r_c < r \leq \infty \text{ and } s_c < s \leq \infty. \end{cases}$$

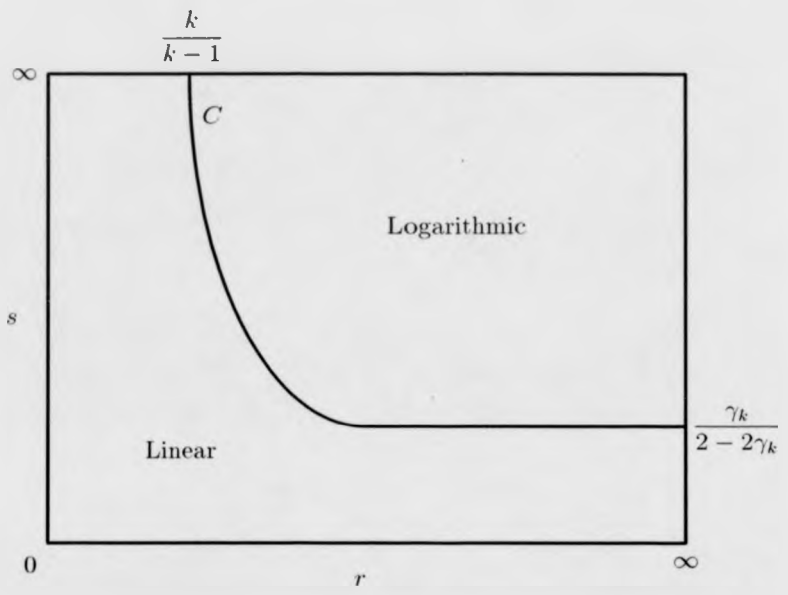

Figure 5.6: Diagram for $\mathbf{EH}_n(r, s)$.

The proofs of Theorems 5.12–5.15 are enhanced versions of the proof of the Erdös-Rényi law. For every $u \in \Sigma^n$ we define the longest run of 'heads' in u by

$$\mathbf{T}(u) = \max\{j : u(i\,..\,i+j-1) = 0^j \text{ for some } i, 1 \le i \le n+1-j\}.$$

Let $\mathbf{ET}_n$ be the expected value of $\mathbf{T}(u)$ for random $u \in \Sigma^n$. Erdös and Rényi [ER70] have proved that

$$\lim_{n\to\infty} \frac{\mathbf{ET}_n}{\log_k n} = 1\,.$$

Longest common subsequences and shortest common supersequences are dual in the sense that $\mathbf{S}(u, v) = |u| + |v| + \mathbf{L}(u, v)$. There is no similar evident

duality between the longest common substring and the *shortest common superstring* problems. It can be shown that the expected length of a shortest common superstring is $2n - O(1)$. The problem of finding a shortest common superstring for more sequences is more interesting. Common superstrings have applications in data compresson [Sto88]. Gallant at al. [GMS80] has shown that the shortest common superstring problem is NP-complete. Middendorf [Mid94] have shown that the shortest common superstring problem over the binary alphabet remains NP-complete even if each given sequence contains exactly three ones. Approximation algorithms for this problem are given by Tarhio and Ukkonen [TU88] and Turner [Tur89]. Blum at al. [BJL+91] have shown that the shortest common supersting problem is also MAX SNP-hard. Thus even approximate algorithms are not likely to be efficient. The expected length of a shortest common superstring of more random sequences is considered by Alexander [Ale94].

Chapter 6

Conclusion

In this thesis we have concentrated on the longest common subsequence problem. Longest common subsequences are often used as the measure of similarity for two (or more) sequences. Knowledge of the expected length of a longest common subsequence is essential for a judgement about the similarity of sequences. It is known that the expected length of a longest common subsequence is proportional to the length of the given sequences. The exact value of this proportion is not known even for a binary alphabet. This proportion is dependent on the alphabet size and has been denoted by γ_k.

We have obtained better upper bounds for γ_k using a new method and, especially for small alphabets, the improvement is substantial. We have also described an algorithm for automated production of lower bounds and using it we have improved lower bounds for γ_2. We have used these methods to obtain bounds in the case of common supersequences and adaptabilities.

It seems that known methods are exhausted and to determine the constants γ_k or substantially improve the bounds for them a new method is needed.

Many open problems remain unsolved and some new ones have been raised. Here is a list of the most challenging of them.

1. What are the exact values of γ_k? What is the value of $\lim_{k\to\infty} \gamma_k k^{1/2}$? What is the convergence rate of $\frac{\mathbf{EL}_n}{n}$, i.e. how large is $\gamma_k n - \mathbf{EL}_n$? Is $\lim_{k\to\infty} \gamma_k k^{1/2} = \lim_{k\to\infty} \gamma_k^{(l)} k^{1-1/l}$? How large is the variance $\mathrm{Var}(\mathbf{L}_n)$?

2. Is there a (uniform) algorithm for computing longest common subsequences that is faster that Masek and Paterson's algorithm? Is it possible to obtain nonlinear lower bounds for the longest common subsequence problem?

3. Can nondominated collations be used to compute the exact values of the constants γ_k?

4. What is the asymptotic behaviour of the expected length of a shortest common supersequence of several sequences?

5. If a longest common nonsupersequence exists, is its length smaller then the total length of the input sequences? Is the longest common nonsupersequence problem decidable? What is the complexity of the shortest maximal supersequence problem? What can be said about longest minimal common nonsubsequences and shortest maximal common nonsupersequences?

6. What are the maximal adaptabilities? Do the sequences with the maximal adaptabilities have the form $u(01101001)^* v$ for some 'constant' sequences $u, v \in \Sigma^*$? What are the proportions of maximal adaptabilities β_k? Is $\lim_{k\to\infty} \gamma_k \sqrt{k} = \lim_{k\to\infty} \beta_k \sqrt{k}$?

Appendix A

Css machine with 931 states

n	s	$T(s,0)$	$T(s,1)$	$d(s)$	n	s	$T(s,0)$	$T(s,1)$	$d(s)$
1	$\left[\begin{smallmatrix}0\\ \lambda\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}\lambda\\ \lambda\end{smallmatrix}\right]1$	$\left[\begin{smallmatrix}0\\ 1\end{smallmatrix}\right]$	39.295	2	$\left[\begin{smallmatrix}0\bullet\\ 1\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00\\ 1\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}01\\ 1\end{smallmatrix}\right]$	31.544
3	$\left[\begin{smallmatrix}00\\ 1\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00\\ 10\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00\\ 11\end{smallmatrix}\right]$	30.133	4	$\left[\begin{smallmatrix}00\bullet\\ 10\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}000\\ 10\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001\\ 10\end{smallmatrix}\right]$	28.362
5	$\left[\begin{smallmatrix}000\\ 10\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}000\\ 100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}000\\ 101\end{smallmatrix}\right]$	25.100	6	$\left[\begin{smallmatrix}01\\ 1\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}1\\ 0\end{smallmatrix}\right]1$	$\left[\begin{smallmatrix}01\\ 11\end{smallmatrix}\right]$	22.712
7	$\left[\begin{smallmatrix}\lambda\bullet\\ \lambda\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0\\ \lambda\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right]$	20.125	8	$\left[\begin{smallmatrix}00\bullet\\ 11\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}000\\ 11\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001\\ 11\end{smallmatrix}\right]$	17.914
9	$\left[\begin{smallmatrix}00\\ \lambda\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0\\ \lambda\end{smallmatrix}\right]1$	$\left[\begin{smallmatrix}00\\ 1\end{smallmatrix}\right]$	17.858	10	$\left[\begin{smallmatrix}000\bullet\\ 110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0000\\ 110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0001\\ 110\end{smallmatrix}\right]$	16.907
11	$\left[\begin{smallmatrix}000\\ 11\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}000\\ 110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}000\\ 111\end{smallmatrix}\right]$	15.796	12	$\left[\begin{smallmatrix}011\bullet\\ 111\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0110\\ 111\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0111\\ 111\end{smallmatrix}\right]$	15.693
13	$\left[\begin{smallmatrix}001\\ 10\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001\\ 100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001\\ 101\end{smallmatrix}\right]$	15.009	14	$\left[\begin{smallmatrix}000\bullet\\ 101\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0000\\ 101\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0001\\ 101\end{smallmatrix}\right]$	15.001
15	$\left[\begin{smallmatrix}001\\ 11\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001\\ 110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001\\ 111\end{smallmatrix}\right]$	12.368	16	$\left[\begin{smallmatrix}001\bullet\\ 100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0010\\ 100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0011\\ 100\end{smallmatrix}\right]$	11.314
17	$\left[\begin{smallmatrix}001\bullet\\ 101\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0010\\ 101\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0011\\ 101\end{smallmatrix}\right]$	11.246	18	$\left[\begin{smallmatrix}0111\\ 1110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}01111\\ 1110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0111\\ 1111\end{smallmatrix}\right]$	10.319
19	$\left[\begin{smallmatrix}0000\\ 1101\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00\\ \lambda\end{smallmatrix}\right]2$	$\left[\begin{smallmatrix}0000\\ 1101\end{smallmatrix}\right]$	9.833	20	$\left[\begin{smallmatrix}0000\\ 101\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0000\\ 110\end{smallmatrix}\right]1$	$\left[\begin{smallmatrix}0000\\ 1011\end{smallmatrix}\right]$	8.787
21	$\left[\begin{smallmatrix}0001\\ 110\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0001\\ 1100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0001\\ 1101\end{smallmatrix}\right]$	8.622	22	$\left[\begin{smallmatrix}0110\\ 111\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0110\\ 1110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0110\\ 1111\end{smallmatrix}\right]$	8.390
23	$\left[\begin{smallmatrix}000\bullet\\ 111\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0000\\ 111\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0001\\ 111\end{smallmatrix}\right]$	8.253	24	$\left[\begin{smallmatrix}0001\\ 101\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0001\\ 1010\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0001\\ 1011\end{smallmatrix}\right]$	7.764
25	$\left[\begin{smallmatrix}0011\\ 100\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0011\\ 1000\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0011\\ 1001\end{smallmatrix}\right]$	7.564	26	$\left[\begin{smallmatrix}0010\\ 100\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0010\\ 1000\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0010\\ 1001\end{smallmatrix}\right]$	6.986
27	$\left[\begin{smallmatrix}01\\ \lambda\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}1\\ \lambda\end{smallmatrix}\right]1$	$\left[\begin{smallmatrix}01\\ 1\end{smallmatrix}\right]$	6.924	28	$\left[\begin{smallmatrix}0011\\ 101\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0011\\ 1010\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0011\\ 1011\end{smallmatrix}\right]$	6.807
29	$\left[\begin{smallmatrix}0000\\ 111\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0000\\ 1110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0000\\ 1111\end{smallmatrix}\right]$	6.698	30	$\left[\begin{smallmatrix}0010\\ 101\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0010\\ 1010\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0010\\ 1011\end{smallmatrix}\right]$	6.553
31	$\left[\begin{smallmatrix}001\bullet\\ 110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0010\\ 110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0011\\ 110\end{smallmatrix}\right]$	6.502	32	$\left[\begin{smallmatrix}00000\bullet\\ 1110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00000\\ 1110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00001\\ 1110\end{smallmatrix}\right]$	6.253
33	$\left[\begin{smallmatrix}000\\ 1\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}000\\ 10\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}000\\ 11\end{smallmatrix}\right]$	5.972	34	$\left[\begin{smallmatrix}0000\bullet\\ 1000\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00\\ \lambda\end{smallmatrix}\right]3$	$\left[\begin{smallmatrix}0000\\ 1000\end{smallmatrix}\right]$	5.921
35	$\left[\begin{smallmatrix}0001\\ 111\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0001\\ 1110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0001\\ 1111\end{smallmatrix}\right]$	5.600	36	$\left[\begin{smallmatrix}0001\bullet\\ 1000\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00010\\ 1000\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00011\\ 1000\end{smallmatrix}\right]$	5.526
37	$\left[\begin{smallmatrix}00000\bullet\\ 1101\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0000\\ \lambda\end{smallmatrix}\right]1$	$\left[\begin{smallmatrix}00001\\ 1101\end{smallmatrix}\right]$	5.270	38	$\left[\begin{smallmatrix}0011\bullet\\ 1001\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}10\\ \lambda\end{smallmatrix}\right]3$	$\left[\begin{smallmatrix}00111\\ 1001\end{smallmatrix}\right]$	5.022
39	$\left[\begin{smallmatrix}011\\ 11\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}011\\ 110\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}011\\ 111\end{smallmatrix}\right]$	4.948	40	$\left[\begin{smallmatrix}00000\bullet\\ 1001\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}000\\ \lambda\end{smallmatrix}\right]2$	$\left[\begin{smallmatrix}00001\\ 1001\end{smallmatrix}\right]$	4.940
41	$\left[\begin{smallmatrix}00110\bullet\\ 1000\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001100\\ 1000\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001101\\ 1000\end{smallmatrix}\right]$	4.878	42	$\left[\begin{smallmatrix}01000\bullet\\ 1111\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}01000\\ 1111\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0011\\ 111\end{smallmatrix}\right]1$	4.851
43	$\left[\begin{smallmatrix}00110\bullet\\ 1001\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001100\\ 1001\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001101\\ 1001\end{smallmatrix}\right]$	4.849	44	$\left[\begin{smallmatrix}000\\ \lambda\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00\\ \lambda\end{smallmatrix}\right]1$	$\left[\begin{smallmatrix}000\\ 1\end{smallmatrix}\right]$	4.766
45	$\left[\begin{smallmatrix}0011\\ 110\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}11\\ 00\end{smallmatrix}\right]2$	$\left[\begin{smallmatrix}0011\\ 1101\end{smallmatrix}\right]$	4.751	46	$\left[\begin{smallmatrix}00110\bullet\\ 10100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001100\\ 1000\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001101\\ 1000\end{smallmatrix}\right]$	4.706
47	$\left[\begin{smallmatrix}0001\bullet\\ 1101\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00010\\ 1101\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00011\\ 1101\end{smallmatrix}\right]$	4.665	48	$\left[\begin{smallmatrix}0001\bullet\\ 1001\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00010\\ 1001\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00011\\ 1001\end{smallmatrix}\right]$	4.545
49	$\left[\begin{smallmatrix}0001\bullet\\ 1100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00010\\ 1100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00011\\ 1100\end{smallmatrix}\right]$	4.519	50	$\left[\begin{smallmatrix}00110\bullet\\ 1100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001100\\ 1100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001101\\ 1100\end{smallmatrix}\right]$	4.507
51	$\left[\begin{smallmatrix}0001\bullet\\ 1011\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00010\\ 1011\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00011\\ 1011\end{smallmatrix}\right]$	4.340	52	$\left[\begin{smallmatrix}0001\bullet\\ 1010\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00010\\ 1010\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0011\\ 10\end{smallmatrix}\right]1$	4.293
53	$\left[\begin{smallmatrix}0011\bullet\\ 1011\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00110\\ 1011\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00111\\ 1011\end{smallmatrix}\right]$	4.186	54	$\left[\begin{smallmatrix}010\\ 1\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}10\\ 0\end{smallmatrix}\right]1$	$\left[\begin{smallmatrix}010\\ 11\end{smallmatrix}\right]$	4.176
55	$\left[\begin{smallmatrix}00110\bullet\\ 1011\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001100\\ 1011\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001101\\ 1011\end{smallmatrix}\right]$	4.176	56	$\left[\begin{smallmatrix}0000\\ 1\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0000\\ 10\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}0000\\ 11\end{smallmatrix}\right]$	4.107
57	$\left[\begin{smallmatrix}0010\\ 110\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00100\\ 1100\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00101\\ 1101\end{smallmatrix}\right]$	4.055	58	$\left[\begin{smallmatrix}0011\bullet\\ 1010\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}00110\\ 1010\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}01111\\ 10\end{smallmatrix}\right]1$	3.821
59	$\left[\begin{smallmatrix}00100\bullet\\ 1010\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001000\\ 1010\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}001001\\ 1010\end{smallmatrix}\right]$	3.701	60	$\left[\begin{smallmatrix}010\\ \lambda\bullet\end{smallmatrix}\right]$	$\left[\begin{smallmatrix}10\\ \lambda\end{smallmatrix}\right]1$	$\left[\begin{smallmatrix}010\\ 1\end{smallmatrix}\right]$	3.519

n	s	$T(s,0)$	$T(s,1)$	$d(s)$	n	s	$T(s,0)$	$T(s,1)$	$d(s)$
61	[011 / 1•]	[11 / 0]1	[011 / 11]	3.508	62	[0000• / 1111]	[00000 / 1111]	[00001 / 1111]	3.453
63	[010 / 11•]	[010 / 110]	[010 / 111]	3.343	64	[00000 / 1110•]	[000 / λ]2	[00000 / 11101]	3.330
65	[00001 / 1110•]	[00001 / 11100]	[00001 / 11101]	3.127	66	[00011 / 1000•]	[1 / 0]3	[1 / λ]4	3.107
67	[00100 / 1001•]	[00100 / 10010]	[00100 / 10011]	3.064	68	[00001 / 1000•]	[00001 / 10000]	[00001 / 10001]	3.031
69	[00100 / 1000•]	[00100 / 10000]	[00100 / 10001]	2.992	70	[00100 / 1100•]	[00100 / 11000]	[00100 / 11001]	2.872
71	[01000 / 1111•]	[000 / 1110]1	[000 / 1111]1	2.800	72	[0001• / 1110]	[00010 / 1110]	[00011 / 1110]	2.800
73	[00101 / 1001•]	[1 / λ]4	[00101 / 10011]	2.785	74	[00010 / 1000•]	[00010 / 10000]	[00010 / 10001]	2.774
75	[00100 / 1011•]	[00100 / 10110]	[00100 / 10111]	2.682	76	[00001 / 1101•]	[00001 / 11010]	[00001 / 11011]	2.635
77	[00111 / 1001•]	[11 / 0]3	[1 / λ]4	2.629	78	[00011 / 1001•]	[00011 / 10010]	[011 / 11]2	2.617
79	[0111 / 10•]	[111 / 00]1	[111 / 01]1	2.610	80	[0000 / 10•]	[0000 / 100]	[0000 / 101]	2.572
81	[00111 / 1000•]	[111 / 00]2	[111 / 01]2	2.557	82	[00010 / 1011•]	[00010 / 10110]	[00010 / 10111]	2.527
83	[00110 / 1000•]	[00110 / 10000]	[00110 / 10001]	2.504	84	[00010 / 1010•]	[00010 / 10100]	[00010 / 10101]	2.503
85	[00001 / 1001•]	[00001 / 10010]	[01 / 11]2	2.495	86	[00100 / 1010•]	[00100 / 10100]	[0100 / 101]1	2.445
87	[00010 / 1101•]	[00010 / 11010]	[00010 / 11011]	2.398	88	[00101 / 1000•]	[00101 / 10000]	[101 / 01]2	2.356
89	[010 / 10•]	[10 / 00]1	[10 / 01]1	2.356	90	[0000 / 11•]	[0000 / 110]	[0000 / 111]	2.342
91	[0011 / 10•]	[0011 / 100]	[0011 / 101]	2.334	92	[00011 / 1101•]	[00011 / 11010]	[00011 / 11011]	2.333
93	[00010 / 1100•]	[00010 / 11000]	[00010 / 11001]	2.325	94	[00010 / 1001•]	[00010 / 10010]	[00010 / 10011]	2.284
95	[00011 / 1100•]	[00011 / 11000]	[00011 / 11001]	2.259	96	[00101 / 1100•]	[00101 / 11000]	[00101 / 11001]	2.254
97	[00110 / 1011•]	[0110 / 110]1	[00110 / 10111]	2.237	98	[0111 / 1100•]	[0111 / 11000]	[0111 / 11001]	2.173
99	[0100 / 101•]	[100 / 010]1	[100 / 011]1	2.172	100	[00011 / 1011•]	[0011 / 110]1	[00011 / 10111]	2.170
101	[00101 / 1010•]	[00101 / 10100]	[00101 / 10101]	2.166	102	[0010• / 1101]	[00100 / 1101]	[00101 / 1101]	2.132
103	[00111 / 1011•]	[0111 / 110]1	[0111 / 111]1	2.093	104	[00101 / 1011•]	[00101 / 10110]	[λ / 11]3	2.091
105	[00110 / 1010•]	[00110 / 10100]	[0110 / 101]1	2.054	106	[0110 / 110•]	[0110 / 1100]	[0110 / 1101]	2.043
107	[010• / 100]	[100 / 00]1	[101 / 00]1	2.037	108	[00000 / 1111•]	[00000 / 11110]	[00000 / 11111]	1.930
109	[00001• / 11111]	[0000010 / 11111]	[0000011 / 11111]	1.828	110	[011• / 100]	[110 / 00]1	[111 / 00]1	1.802
111	[0110 / 101•]	[110 / 010]1	[110 / 011]1	1.736	112	[00001 / 1111•]	[00001 / 11110]	[00001 / 11111]	1.727
113	[00100 / 1101•]	[00100 / 11010]	[00100 / 11011]	1.684	114	[00100• / 10111]	[001000 / 10111]	[001001 / 10111]	1.669
115	[00000• / 11101]	[00000 / 1]1	[000001 / 11101]	1.665	116	[00001• / 10001]	[010 / 1]3	[011 / 1]3	1.664
117	[00010• / 10111]	[000100 / 10111]	[000101 / 10111]	1.659	118	[0010 / 10•]	[0010 / 100]	[0010 / 101]	1.638
119	[00001• / 10000]	[000010 / 10000]	[11 / λ']4	1.630	120	[00100• / 10001]	[001000 / 10001]	[001001 / 10001]	1.599
121	[00001• / 11100]	[000010 / 11100]	[000011 / 11100]	1.563	122	[00001• / 11101]	[000010 / 11101]	[000011 / 11101]	1.563
123	[00010• / 10001]	[00 / λ]4	[01 / λ]4	1.549	124	[00100• / 10011]	[001000 / 10011]	[001001 / 10011]	1.532
125	[00100• / 10010]	[00 / λ]4	[001001 / 10010]	1.532	126	[00100• / 10000]	[001000 / 10000]	[001001 / 10000]	1.496
127	[00011• / 10111]	[000110 / 10111]	[00111 / 111]1	1.481	128	[0000 / 100•]	[0000 / 1000]	[0000 / 1001]	1.468
129	[00010• / 10000]	[000100 / 10000]	[101 / 0]3	1.468	130	[00010 / 1110•]	[00010 / 11100]	[00010 / 11101]	1.465
131	[00010• / 10011]	[000100 / 10011]	[0101 / 11]2	1.459	132	[00110• / 10111]	[01100 / 111]1	[001101 / 10111]	1.446
133	[0101 / 11•]	[0101 / 110]	[0101 / 111]	1.444	134	[00100• / 11000]	[001000 / 11000]	[001001 / 11000]	1.436
135	[00100• / 11001]	[001000 / 11001]	[001001 / 11001]	1.436	136	[00010• / 11100]	[000100 / 11100]	[000101 / 11100]	1.433
137	[01101• / 11100]	[010 / 100]2	[011 / 100]2	1.432	138	[00010 / 10•]	[010 / λ]2	[00010 / 101]	1.427
139	[0100 / 11•]	[0100 / 110]	[0100 / 111]	1.424	140	[00000 / 1•]	[00000 / 10]	[00000 / 11]	1.423
141	[01101• / 11110]	[010 / 110]2	[011011 / 11110]	1.418	142	[00011 / 1110•]	[00011 / 11100]	[00011 / 11101]	1.400
143	[00010• / 11000]	[000100 / 11000]	[101 / λ]3	1.398	144	[00101• / 10011]	[001010 / 10011]	[011 / 1]3	1.393
145	[00100• / 11010]	[01000 / 10]1	[001001 / 11010]	1.375	146	[0000 / λ•]	[000 / λ]1	[0000 / 1]	1.372
147	[00101• / 10101]	[001010 / 10101]	[01011 / 101]1	1.368	148	[00011• / 11000]	[000110 / 11000]	[11 / λ]3	1.365
149	[0010 / 11•]	[0010 / 110]	[0010 / 111]	1.357	150	[00100• / 10110]	[01000 / 110]1	[001001 / 10110]	1.341
151	[00001• / 11010]	[00010 / 10]1	[000011 / 11010]	1.318	152	[00001• / 11011]	[000010 / 11011]	[000011 / 11011]	1.318
153	[0111 / 11•]	[0111 / 110]	[0111 / 111]	1.314	154	[01000 / 110•]	[01000 / 1100]	[01000 / 1101]	1.313
155	[0010 / 1•]	[0010 / 10]	[0010 / 11]	1.278	156	[00100 / 100•]	[00100 / 1000]	[00100 / 1001]	1.278
157	[001 / 1•]	[001 / 10]	[001 / 11]	1.268	158	[01101• / 11101]	[010 / 101]2	[011011 / 11101]	1.266

n	s	$T(s,0)$	$T(s,1)$	$d(s)$	n	s	$T(s,0)$	$T(s,1)$	$d(s)$
159	[00010•][10110]	[00100][110]1	[000101][10110]	1.263	160	[00010•][10100]	[00100][100]1	[000101][10100]	1.262
161	[01000][10•]	[1000][00]1	[1000][01]1	1.261	162	[00110•][10000]	[001100][10000]	[1101][00]2	1.252
163	[00110•][10001]	[001100][10001]	[1101][01]2	1.252	164	[00010•][10101]	[00100][101]1	[000101][10101]	1.252
165	[011][λ•]	[11][λ]1	[011][1]	1.237	166	[00010•][11001]	[000100][11001]	[0101][1]2	1.234
167	[00100•][10100]	[01000][100]1	[001001][10100]	1.222	168	[00010•][11011]	[000100][11011]	[000101][11011]	1.219
169	[00011•][11001]	[000110][11001]	[0111][1]2	1.201	170	[00010•][11010]	[00100][10]1	[000101][11010]	1.199
171	[01•][10]	[10][0]1	[11][0]1	1.189	172	[00100][101•]	[00100][1010]	[00100][1011]	1.189
173	[00011•][11011]	[000110][11011]	[000111][11011]	1.186	174	[0100][110•]	[0100][1100]	[0100][1101]	1.184
175	[00101•][10000]	[001010][10000]	[1011][00]2	1.178	176	[00011•][11010]	[000110][11010]	[000111][11010]	1.166
177	[00101•][11000]	[001010][11000]	[1011][0]2	1.159	178	[00101•][11001]	[001010][11001]	[001011][11001]	1.159
179	[00100][10•]	[00100][100]	[00100][101]	1.126	180	[01011][101•]	[1011][010]1	[1011][011]1	1.116
181	[01100][111•]	[01100][1110]	[01100][1111]	1.108	182	[00100][110•]	[00100][1100]	[00100][1101]	1.108
183	[00101•][10100]	[001010][10100]	[001011][10100]	1.083	184	[00101][1101•]	[00101][11010]	[00101][11011]	1.066
185	[00101•][10110]	[001010][10110]	[001011][10110]	1.046	186	[0111][1•]	[111]1[0]	[0111][11]	1.030
187	[00110•][10100]	[01100][100]1	[001101][10100]	1.027	188	[0010][λ•]	[010]1[λ]	[0010][1]	1.021
189	[001][λ•]	[01][λ]1	[001][1]	1.016	190	[000000•][11111]	[000000][11111]	[000001][11111]	0.965
191	[0101][1•]	[101]1[0]	[0101][11]	0.960	192	[000010][111110•]	[000010][111110]	[0[11111]]1	0.946
193	[0100][100•]	[100]1[000]	[100]1[001]	0.935	194	[000011][111110•]	[000011][111110]	[λ[1111]]2	0.914
195	[0100][1•]	[100]1[0]	[0100][11]	0.872	196	[000010•][11110]	[000010][11110]	[000011][11110]	0.863
197	[000010][10000•]	[10]4[0]	[0]5[λ]	0.854	198	[00100•][11011]	[001000][11011]	[001001][11011]	0.842
199	[001001][101111•]	[001001][101110]	[01[1111]]2	0.835	200	[001000][10111•]	[01000]1[1110]	[001000][101111]	0.835
201	[000001][111101•]	[000001][111010]	[000001][111011]	0.833	202	[000101][101110•]	[000101][101110]	[λ[111]]3	0.830
203	[000100][10111•]	[000100][101110]	[000100][101111]	0.830	204	[0100][111•]	[0100][1110]	[0100][1111]	0.825
205	[0101][111•]	[0101][1110]	[01[111]]1	0.822	206	[010•][100]	[100]1[01]	[101]1[01]	0.820
207	[0101][110•]	[0101][1100]	[0101][1101]	0.817	208	[00000][11•]	[0000]1[λ]	[00000][111]	0.815
209	[00000][10•]	[000]2[λ]	[0000]1[1]	0.815	210	[001001][10001•]	[1001][010]2	[1001][011]2	0.800
211	[001000][10001•]	[001000][100010]	[001000][100011]	0.800	212	[01000][1110•]	[01000][11100]	[01000][11101]	0.792
213	[000011][11100•]	[011]3[λ]	[000011][111001]	0.782	214	[000011][111101•]	[000011][111010]	[000011][111011]	0.782
215	[000010][11100•]	[010]3[λ]	[0010]2[1]	0.782	216	[000010][111101•]	[00010]1[10]	[000010][111011]	0.782
217	[011][10•]	[1]1[00]	[1]1[01]	0.780	218	[001000][10011•]	[000]3[10]	[001000][100111]	0.766
219	[001001][10010•]	[001001][100100]	[001001][100101]	0.766	220	[001001][10011•]	[001001][100110]	[001001][100111]	0.766
221	[00111][111•]	[00111][1110]	[00111][1111]	0.763	222	[001010][10011•]	[010]3[10]	[001010][100111]	0.757
223	[001001][100000•]	[001001][1000000]	[1001]2[001]	0.748	224	[001000][100000•]	[001000][1000000]	[001000][1000001]	0.748
225	[000110][10111•]	[000110][101110]	[00110]1[1111]	0.740	226	[000100][100000•]	[000100][1000000]	[000100][1000001]	0.734
227	[001010][10101•]	[001010][101010]	[001010][101011]	0.733	228	[00010•][11101]	[000100][11101]	[000101][11101]	0.733
229	[000100][10011•]	[0100]2[110]	[000100][100111]	0.729	230	[001010][100001•]	[001010][1000000]	[001010][1000001]	0.728
231	[000110][11000•]	[000110][1100000]	[000110][1100001]	0.724	232	[001101][10111•]	[001101][101110]	[001101][101111]	0.723
233	[001000][11000•]	[001000][1100000]	[001000][1100001]	0.718	234	[001001][11001•]	[001001][110010]	[001001][110011]	0.718
235	[001001][11000•]	[001001][1100000]	[001001][1100001]	0.718	236	[001000][11001•]	[00]4[λ]	[001000][110011]	0.718
237	[000101][11100•]	[101]3[λ]	[000101][111001]	0.716	238	[000100][111100•]	[000100][1111000]	[000100][111001]	0.716
239	[00010][101•]	[00010][1010]	[00010][1011]	0.713	240	[011011][111110•]	[011[1100]]2	[011[1101]]2	0.709
241	[0010][111•]	[0010][1110]	[0010][1111]	0.708	242	[0001][1000•]	[0001][1000]	[0001][1001]	0.700
243	[00011•][11100]	[000110][111100]	[000111][111100]	0.700	244	[000100][1100000•]	[000100][1100000]	[00]4[λ]	0.699
245	[0100][10•]	[100]1[00]	[100]1[01]	0.690	246	[001001][11010•]	[01001]1[100]	[01001]1[101]	0.687
247	[0110][1•]	[110]1[0]	[0110][11]	0.683	248	[001001][101100•]	[001001][1011000]	[001001][1011001]	0.671
249	[000011][110110•]	[000011][1101100]	[000011][1101101]	0.659	250	[000010][110110•]	[000010][1101100]	[000010][1101101]	0.659
251	[000011][11010•]	[00011]1[100]	[000011][110101]	0.659	252	[001010][110000•]	[001010][1100000]	[001010][1100001]	0.658
253	[001010][110010•]	[001010][1100100]	[001010][1100101]	0.658	254	[01000][1100•]	[000]1[1000]	[01000][11001]	0.656
255	[01000][1101•]	[00]3[λ]	[01000][11011]	0.656	256	[000110][110010•]	[000110][1100100]	[000110][1100101]	0.642

n	s	$T(s,0)$	$T(s,1)$	$d(s)$	n	s	$T(s,0)$	$T(s,1)$	$d(s)$
257	[001011] [10110•]	[λ 00]4	[001011] [101101]	0.638	258	[0110] [100•]	[110 000]1	[110] [001]1	0.636
259	[01100] [1111•]	[00 110]2	[00 111]2	0.633	260	[01100] [1110•]	[00 100]2	[01100] [11101]	0.633
261	[011011] [11101•]	[011 1010]2	[011 1011]2	0.633	262	[000101] [10110•]	[000101] [101100]	[000101] [101101]	0.632
263	[000101] [10100•]	[000101] [101000]	[00101] [1001]1	0.631	264	[001100] [10001•]	[001100] [100010]	[1100] [011]2	0.626
265	[001100] [10000•]	[001100] [100000]	[001100] [100001]	0.626	266	[000101] [10101•]	[00101] [1010]1	[000101] [101011]	0.626
267	[000100] [11001•]	[0100] [10]2	[000100] [110011]	0.617	268	[01000] [111•]	[01000] [1110]	[01000] [1111]	0.614
269	[01000] [100•]	[1000] [000]1	[1000]1 [001]	0.611	270	[001001] [10100•]	[001001] [101000]	[01001] [1001]1	0.611
271	[000101] [11011•]	[000101] [110110]	[01] [10111]1	0.609	272	[000100] [11001•]	[00100] [110]1	[000100] [110111]	0.609
273	[000101] [11010•]	[000101] [110100]	[000101] [110101]	0.599	274	[0110] [11•]	[0110] [110]	[0110] [111]	0.594
275	[000111] [11011•]	[000111] [110110]	[000111] [110111]	0.593	276	[000110] [11011•]	[000110] [110110]	[000110] [110111]	0.593
277	[001010] [10100•]	[λ 00]4	[001010] [101001]	0.591	278	[000111] [11010•]	[000111] [110100]	[000111] [110101]	0.583
279	[000110] [11010•]	[000110] [110100]	[000110] [110101]	0.583	280	[001011] [11001•]	[11]4	[011 1]3	0.579
281	[01000] [1•]	[1000] [0]1	[01000] [11]	0.576	282	[01010] [11010•]	[01010] [11010]	[010 1011]1	0.570
283	[0110] [λ•]	[110] [λ]1	[0110] [11]	0.566	284	[001011] [10100•]	[001011] [101000]	[01011] [1001]1	0.542
285	[00101•] [11010]	[001010] [11010]	[001011] [11010]	0.533	286	[00010] [1•]	[00010] [10]	[00010] [11]	0.523
287	[001010] [10110•]	[0 100]3	[01010] [1101]1	0.523	288	[00110] [1111•]	[0 110]2	[0 111]2	0.514
289	[01100] [100•]	[1100] [000]1	[1100] [001]1	0.514	290	[001101] [10100•]	[101 01000]1	[001101] [101001]	0.514
291	[000000•] [111111]	[000000] [111111]	[0000001] [111111]	0.494	292	[000000] [11111•]	[000000] [111110]	[000000] [111111]	0.494
293	[000001•] [111111]	[00000010] [111111]	[λ 1111]2	0.488	294	[000001] [11111•]	[000001] [111110]	[000001] [111111]	0.483
295	[0001] [10•]	[0001] [100]	[0001] [101]	0.480	296	[0000010•] [111110]	[000010] [111110]	[0000101] [111110]	0.473
297	[00111] [1111•]	[2 10]3	[λ 11]3	0.471	298	[00111] [1110•]	[00111] [11100]	[00111] [11101]	0.471
299	[000010] [11110•]	[0010] [λ]2	[000010] [111101]	0.463	300	[000011•] [111110]	[000110] [111110]	[λ 110]3	0.457
301	[0101] [100•]	[101] [000]1	[101] [001]1	0.447	302	[01000] [11•]	[01000] [110]	[01000] [111]	0.442
303	[000011] [11110•]	[000011] [111100]	[λ 1101]2	0.432	304	[001000] [11011•]	[01000] [110]1	[01000] [111]1	0.421
305	[001001] [11011•]	[001001] [110110]	[001 10111]1	0.421	306	[00011] [1000]	[00011] [1000]	[00011] [1001]	0.420
307	[0010011•] [101110]	[010010] [1110]1	[011 1110]2	0.417	308	[0010000•] [101111]	[010000] [1111]1	[001 1111]2	0.417
309	[0000001•] [111011]	[000010] [11]1	[0000011] [111011]	0.416	310	[0000001•] [111010]	[000010] [10]1	[0000011] [111010]	0.416
311	[0011] [1110•]	[0011] [1110]	[λ 11]2	0.415	312	[0001010•] [101110]	[0001010] [101110]	[λ 10]4	0.415
313	[0001000•] [101111]	[0001000] [101111]	[0001001] [101111]	0.415	314	[0001000•] [101110]	[001000] [1110]1	[01 1110]2	0.415
315	[01110] [1110•]	[01110] [1110]	[01110] [1111]	0.413	316	[01110] [110•]	[10 00]2	[01110] [1101]	0.413
317	[00000] [1110•]	[00000] [1110]	[00000] [1111]	0.407	318	[01110] [110•]	[01110] [110]	[01110] [111]	0.403
319	[0010000•] [100011]	[0010000] [100011]	[λ 1]5	0.400	320	[0010000•] [100010]	[0010000] [100010]	[0010001] [100010]	0.400
321	[0000110•] [111001]	[0000110] [111001]	[0000111] [111001]	0.391	322	[0000110•] [111011]	[0000110] [111011]	[0000111] [111011]	0.391
323	[0000110•] [111010]	[0000110] [111010]	[0000111] [111010]	0.391	324	[0000100•] [11]	[000100]1	[0000101] [111011]	0.391
325	[0010000•] [100111]	[0000]3 [11]	[0010001] [100111]	0.383	326	[0010010•] [100110]	[0010]3 [10]	[0010011] [100110]	0.383
327	[0010010•] [100100]	[0010010] [100100]	[11 λ]5	0.383	328	[0010010•] [100101]	[0010010] [100101]	[011 1]4	0.383
329	[0010010•] [100111]	[0010010] [100111]	[λ 1]5	0.383	330	[0000110•] [110111]	[0000110] [110111]	[0000111] [110111]	0.381
331	[0000100•] [110111]	[0000100] [111]1	[01 10111]1	0.381	332	[0010100•] [100111]	[0100]3 [11]	[0010101] [100111]	0.378
333	[0010110•] [10110]	[λ 1]5	[0010111] [101101]	0.374	334	[0010010•] [100000]	[0010010] [100000]	[10011 000]2	0.374
335	[0010000•] [100001]	[0010000] [100001]	[10001 001]2	0.374	336	[0010000•] [100000]	[0010000] [100000]	[10001 000]2	0.374
337	[0000110•] [110111]	[0001100] [110111]	[λ 11]4	0.373	338	[0000111•] [110111]	[0001110] [111]1	[0001111] [110111]	0.373
339	[000010] [10•]	[0010] [λ]2	[00010] [1]1	0.371	340	[0001110•] [101110]	[0011100] [1110]1	[0001101] [101110]	0.370
341	[0001110•] [110010]	[0001100] [110010]	[01101] [10]2	0.367	342	[0001100•] [100000]	[0001000] [100000]	[1001 00]3	0.367
343	[0001000•] [100001]	[0001000] [100001]	[1001] [01]3	0.367	344	[0010110•] [101011]	[010100] [1011]1	[λ 1]5	0.366
345	[0010110•] [101010]	[010100] [1010]1	[0010101] [101010]	0.366	346	[000101] [11101•]	[000101] [111010]	[01 11011]1	0.366
347	[000100] [1110•]	[00100] [10]1	[0001001] [111011]	0.366	348	[0001100•] [100111]	[01000] [111]2	[0001001] [100111]	0.365
349	[01001] [101•]	[1001] [010]1	[1001] [011]1	0.364	350	[01001] [100•]	[1001] [000]1	[1001] [001]1	0.364
351	[0010110•] [1000000]	[0010100] [1000000]	[10101] [000]2	0.364	352	[0010110•] [1000001]	[0010100] [1000001]	[10101 001]2	0.364
353	[01011] [1001•]	[1011] [0010]1	[1011] [0011]1	0.362	354	[0000110•] [1100001]	[0001100] [110001]	[101 λ]4	0.362

n	s	$T(s,0)$	$T(s,1)$	$d(s)$	n	s	$T(s,0)$	$T(s,1)$	$d(s)$
355	[000110•] [110000]	[0001100] [110000]	[1101]3 [0]	0.362	356	[001101•] [101111]	[011010]1 [1111]	[0011011] [101111]	0.362
357	[001101•] [101110]	[011010]1 [1110]	[0011011] [101110]	0.362	358	[01100] [1100•]	[λ]4 [0]	[λ]4 [1]	0.360
359	[01010] [111•]	[010] [110]1	[010] [111]1	0.359	360	[01010] [110•]	[01010] [1100]	[01010] [1101]	0.359
361	[0010000•] [110000]	[0010000] [110000]	[10001]2 [00]	0.359	362	[0010000•] [1]	[0000]3 []	[0010001] [110011]	0.359
363	[0010000•] [110001]	[0010000] [110001]	[0010001] [110001]	0.359	364	[0010001•] [110000]	[0010010] [110000]	[10011]2 [00]	0.359
365	[0010010•] [110010]	[0010010] [110010]	[011] [λ]4	0.359	366	[0010010•] [110001]	[0010010] [110001]	[10011]2 [01]	0.359
367	[0010010•] [110011]	[0010010] [110011]	[0011] [10011]1	0.359	368	[0001000•] [111001]	[01000] [1]2	[0001001] [111001]	0.358
369	[0001010•] [111001]	[01010] [1]2	[0001011] [111001]	0.358	370	[0001000•] [111000]	[0001000] [111000]	[1001] [λ]3	0.358
371	[01010] [11•]	[01010] [110]	[01010] [111]	0.354	372	[0100] [λ•]	[100] [λ]1	[0100] [1]	0.353
373	[000111] [11100•]	[111] [0000]3	[000111] [111001]	0.350	374	[000110] [11100•]	[000110] [111000]	[000110] [111001]	0.350
375	[000110•] [111000]	[0001100] [111000]	[1101] [λ]3	0.350	376	[0001000•] [110000]	[0001000] [110000]	[1001] [0]3	0.349
377	[001010] [11010•]	[010] [10100]1	[0001010] [110101]	0.345	378	[0001110•] [110101]	[0001110] [110101]	[0001111] [110101]	0.343
379	[0001100•] [110101]	[0001100] [110101]	[0001101] [110101]	0.343	380	[00100] [λ•]	[0100] [λ]1	[00100] [1]	0.341
381	[01101] [110•]	[01101] [1100]	[λ] [λ]4	0.337	382	[01101] [111•]	[01101] [1110]	[01101] [1111]	0.337
383	[001001•] [101100]	[010010] [1100]1	[0010011] [101100]	0.335	384	[001001•] [101101]	[010010]1 [1101]	[0010011] [101101]	0.335
385	[000011•] [110101]	[0000110] [110101]	[0000111] [110101]	0.329	386	[000010•] [110]	[000100]1 []	[0000101] [110110]	0.329
387	[000011•] [110110]	[0000110] [110110]	[0000111] [110110]	0.329	388	[0010100•] [110011]	[0010100] [110011]	[0010101] [110011]	0.329
389	[001010•] [110000]	[0010100] [110000]	[10101] [00]2	0.329	390	[0010100•] [110001]	[0010100] [110001]	[10101] [01]2	0.329
391	[001010•] [110010]	[0010100] [110010]	[101] [λ]4	0.329	392	[000100] [111•]	[00100] [λ]1	[000100] [111]	0.324
393	[000110•] [110011]	[0001100] [110011]	[0001101] [110011]	0.321	394	[01100] [11•]	[01100] [110]	[01100] [111]	0.320
395	[00100] [1•]	[000100] [10]	[00100] [11]	0.319	396	[001001•] [101000]	[0010010] [101000]	[0100011] [1000]	0.317
397	[0011] [110]	[0011] [110•]	[0011] [111]	0.316	398	[000101•] [101101]	[001010] [1101]	[0001011] [101101]	0.316
399	[000101•] [101100]	[001010] [1100]1	[0001011] [101100]	0.316	400	[000101•] [101000]	[0001010] [101000]	[001011] [1000]1	0.316
401	[0011000•] [100010]	[0011000] [100010]	[11001]2 [010]	0.313	402	[0011000•] [100001]	[0011000] [100001]	[11001]2 [001]	0.313
403	[0011000•] [100000]	[0011000] [100000]	[11001]2 [000]	0.313	404	[000101•] [101011]	[0001010] [101011]	[001011]1 [1011]	0.313
405	[0001100•] [110011]	[01000]2 [11]	[0001001] [110011]	0.309	406	[001000] [1110•]	[001000] [1110]	[01000]1 [1]	0.306
407	[01001] [1001•]	[1001] [0010]1	[1001] [0011]1	0.306	408	[000101•] [110110]	[0001010] [110110]	[0001011] [110110]	0.305
409	[0001100•] [110111]	[0010000]1 [111]	[0001001] [110111]	0.305	410	[01010] [10•]	[1010] [00]1	[1010]1 [01]	0.304
411	[000101•] [110101]	[0001010] [110101]	[001011]1 [101]	0.300	412	[000101•] [110100]	[0001010] [110100]	[001011]1 [100]	0.300
413	[000100] [110•]	[0100]2 [λ]	[00100]1 [1]	0.298	414	[000110•] [110110]	[001100]1 [110]	[0001101] [110110]	0.297
415	[0001110•] [110110]	[0001110] [110110]	[0001111] [110110]	0.297	416	[001010•] [101001]	[λ]5 [1]	[010101]1 [1001]	0.295
417	[0001110•] [110100]	[0001110] [110100]	[001111]1 [100]	0.292	418	[000110•] [110100]	[0001100] [110100]	[0001101] [110100]	0.292
419	[00110] [101•]	[00110] [1010]	[00110] [1011]	0.287	420	[00110] [111•]	[00110] [1110]	[00110] [1111]	0.287
421	[00110] [110•]	[110]2 [00]	[λ]3 [1]	0.287	422	[0101] [10•]	[101]1 [00]	[101]1 [01]	0.281
423	[01100] [110•]	[01100] [1100]	[0]3 [1]	0.278	424	[01100] [10•]	[1100]1 [00]	[1100]1 [01]	0.275
425	[0010110•] [101000]	[10] [0000]3	[0101111]1 [1000]	0.271	426	[000100] [11•]	[000100] [110]	[000100] [111]	0.267
427	[001011] [11010•]	[01011]1 [100]	[01011]1 [101]	0.266	428	[01101] [1111•]	[01101] [11110]	[01101] [11111]	0.262
429	[00010] [11•]	[00010] [110]	[00010] [111]	0.261	430	[00110] [10•]	[00110] [100]	[00110] [101]	0.261
431	[00110] [11•]	[00110] [110]	[00110] [111]	0.261	432	[001100] [110•]	[1100]2 [00]	[001100] [1101]	0.257
433	[01000] [101•]	[1000]1 [010]	[1000]1 [011]	0.257	434	[0011101•] [101001]	[0011010] [101001]	[0011011]1 [1001]	0.257
435	[0010111] [1000•]	[1011]2 [00]	[1011]2 [01]	0.256	436	[0111] [101•]	[111] [010]1	[111] [011]1	0.255
437	[01010] [1100•]	[010] [1000]1	[010] [1001]1	0.253	438	[000010] [11•]	[000010] [110]	[000010] [111]	0.253
439	[0110] [10•]	[110] [00]1	[110] [01]1	0.253	440	[0000001] [1111111•]	[0000001] [1111110]	[λ] [111111]1	0.247
441	[0011100] [1011•]	[11100] [0110]1	[11100] [0111]1	0.245	442	[0000010] [1111111•]	[0000010] [1111110]	[0] [111111]1	0.244
443	[0000001•] [111110]	[0000010] [111110]	[0000011] [111110]	0.241	444	[0000100] [1111110•]	[0000100] [1111100]	[0000100] [1111101]	0.236
445	[0000101] [1111100•]	[0000101] [1111100]	[01] [111101]1	0.236	446	[00000] [λ•]	[0000]1 [λ]	[00000] [1]	0.234
447	[0000010•] [111101]	[0000100]1 [1]	[01] [111101]1	0.232	448	[0010111] [1011•]	[0010111] [10110]	[λ]4 [1]	0.231
449	[0000110] [1111110•]	[0000110] [1111100]	[0] [11101]2	0.229	450	[0110010] [1110•]	[0110]2 [100]	[010]2 [101]	0.228
451	[0100010] [11000•]	[0010]1 [1000]	[0010]1 [1001]	0.220	452	[0100010] [11001•]	[0100010] [11010]	[0010]1 [1011]	0.220

n	s	$T(s,0)$	$T(s,1)$	$d(s)$	n	s	$T(s,0)$	$T(s,1)$	$d(s)$
453	[0001] [1•]	[0001] [10]	[0001] [11]	0.218	454	[001110] [111•]	[λ]4 [λ]	[0]3 [1]	0.217
455	[01101] [11•]	[01101] [110]	[01101] [111]	0.216	456	[000011•] [111100]	[00001110] [111100]	[00001111] [111100]	0.216
457	[01011] [100•]	[1011]1 [000]	[1011]1 [001]	0.214	458	[00010] [λ•]	[0010]1 [λ]	[00010] [1]	0.213
459	[01110] [1111•]	[01110] [11110]	[01110] [11111]	0.212	460	[01110] [110•]	[ ]4 [0]	[01110] [11101]	0.212
461	[0010011•] [110110]	[010010]1 [110]	[0011]1 [10110]	0.210	462	[01101] [100•]	[1]3 [00]	[1]3 [01]	0.210
463	[010010] [110•]	[0010]1 [100]	[0010]1 [101]	0.209	464	[010000] [111•]	[0000]1 [110]	[0000]1 [111]	0.209
465	[00000111] [1110111•]	[0000011] [1110110]	[0000011] [1110111]	0.208	466	[0000011] [111010•]	[000011]1 [100]	[0000111] [1110101]	0.208
467	[01010] [1•]	[1010]1 [0]	[01010] [11]	0.208	468	[0001001] [101111•]	[01] [11110]2	[01] [11111]2	0.207
469	[0001010] [101110•]	[001010]1 [11100]	[0] [1101]3	0.207	470	[0001000] [101111•]	[001000]1 [11110]	[001] [11111]2	0.207
471	[01110] [1101•]	[10] [010]2	[01110] [11011]	0.207	472	[0010000] [1000100]	[0010000] [1000100]	[0] [101]4	0.200
473	[00100000] [100011•]	[0] [110]4	[0] [111]4	0.200	474	[0010001] [100011•]	[0010001] [1000100]	[λ] [01]5	0.200
475	[00110] [1•]	[00110] [10]	[00110] [11]	0.200	476	[001000] [111•]	[001000] [1110]	[000] [111]	0.198
477	[001011] [100•]	[001011] [1000]	[011]3 [λ]	0.196	478	[0000110] [1110111•]	[0000110] [1110110]	[0] [10111]2	0.195
479	[0000111] [1110111•]	[λ] [0110]3	[00000111] [11101111]	0.195	480	[0000110] [111010•]	[000110]1 [100]	[00000110] [1110101]	0.195
481	[0000101] [1110111•]	[0000101] [1110110]	[01] [110111]1	0.195	482	[0000110] [111001•]	[00110]2 [10]	[00110]2 [11]	0.195
483	[0000111] [1110011•]	[000011] [1110010]	[00000111] [11100111]	0.195	484	[0000111] [111010•]	[000011] [1110100]	[λ] [0101]3	0.195
485	[00100010] [100100•]	[10] [0]5	[0] [λ]6	0.191	486	[0010001] [100111•]	[0001]3 [110]	[01] [1111]3	0.191
487	[0010011] [100110•]	[0010011] [1001100]	[λ] [01]5	0.191	488	[00100010] [100101•]	[010] [10]4	[λ] [11]5	0.191
489	[00100010] [100111•]	[0010] [110]3	[0] [111]4	0.191	490	[001111] [100•]	[111]2 [0]	[11] [λ]3	0.191
491	[00000111] [1101111•]	[0000111] [1101110]	[000111]1 [1111]	0.191	492	[0000110] [1101111•]	[0000110] [1101110]	[λ] [1111]3	0.191
493	[0010101] [100111•]	[0101]3 [110]	[0101]3 [111]	0.189	494	[01101] [101•]	[1101]1 [010]	[1101]1 [011]	0.188
495	[00101111] [1011101•]	[ ] [010]4	[00101111] [10110111]	0.187	496	[00100000] [100001•]	[00100000] [1000010]	[λ] [11]5	0.187
497	[0010010] [100000•]	[10010]2 [0000]	[0010010] [1000001]	0.187	498	[0010000] [100000•]	[10000]2 [0000]	[λ] [01]5	0.187
499	[0001100] [1110001•]	[0001100] [1110000]	[0001100] [1110001]	0.187	500	[0001111] [110111•]	[11] [01110]2	[0001111] [1101111]	0.187
501	[0001100] [110111•]	[0001100] [1101110]	[0] [1111]3	0.187	502	[000010] [110•]	[0010]2 [λ]	[00010]1 [1]	0.186
503	[0001101] [101110•]	[001101]1 [11100]	[00110]1 [11101]	0.185	504	[001100] [110•]	[00] [100]2	[0] [101]2	0.185
505	[000000] [1•]	[00000]1 [λ]	[000000] [11]	0.184	506	[0001100] [110010•]	[0001100] [1100100]	[0001100] [1100101]	0.184
507	[01101] [10•]	[1101]1 [00]	[1101]1 [01]	0.184	508	[0001000] [100000•]	[1000]3 [000]	[1000]3 [001]	0.183
509	[0001000] [100001•]	[0001000] [1000010]	[λ] [011]4	0.183	510	[0010101] [101010•]	[λ] [00]5	[010101]1 [10101]	0.183
511	[010100] [1011•]	[10100]1 [0110]	[10100]1 [0111]	0.183	512	[010100] [1010•]	[10100]1 [0100]	[10100]1 [0101]	0.183
513	[000101•] [111010]	[0001010] [1110101]	[0001011] [1110101]	0.183	514	[0001000•] [111011]	[001000]1 [11]	[001] [11011]1	0.183
515	[0001001] [100111•]	[λ] [110]4	[λ] [111]4	0.182	516	[0010100] [1000001•]	[10100]2 [0000]	[00101001] [1000001]	0.182
517	[0010100] [1000010•]	[0010100] [1000010]	[100] [00011]2	0.182	518	[0001110] [110101•]	[10] [01010]2	[0001110] [1101011]	0.182
519	[0001100] [110001•]	[0001100] [1100010]	[00] [λ]5	0.181	520	[0001100] [1100000•]	[0001100] [1100000]	[0001100] [1100001]	0.181
521	[0011011] [101110•]	[1011]1 [011100]	[0011011] [10111101]	0.181	522	[0011011] [101111•]	[0011011] [1011110]	[011011]1 [11111]	0.181
523	[011010] [1111•]	[010]2 [110]	[010]2 [111]	0.181	524	[00100000] [1100000•]	[00100000] [1100000]	[0000]1 [1000011]	0.179
525	[00100010] [110010•]	[10]5 [λ]	[010]4 [1]	0.179	526	[00100001] [1100011•]	[00100000] [1100010]	[0000]1 [1000111]	0.179
527	[0010010] [110001•]	[10010]2 [010]	[0010] [10001]1	0.179	528	[0010001] [110011•]	[0001]3 [10]	[0001]1 [100111]	0.179
529	[0010010] [110000•]	[10010]2 [000]	[10010]2 [001]	0.179	530	[0010010] [100111•]	[0010]3 [10]	[0010]1 [100111]	0.179
531	[00100001] [1100011•]	[0010001] [1100010]	[0001]1 [1000111]	0.179	532	[0001011] [111001•]	[01011]2 [10]	[0001011] [1110011]	0.179
533	[0001001] [111001•]	[01001]2 [10]	[001] [11001]1	0.179	534	[0001000] [111000•]	[0001000] [1110000]	[000]4 [λ]	0.179
535	[0101] [λ•]	[101]1 [λ]	[0101] [1]	0.178	536	[011011] [11111•]	[011] [1110]2	[011] [1111]2	0.176
537	[000110•] [111001]	[0001100] [1110011]	[01101]2 [1]	0.175	538	[0001000] [1100000•]	[1000]3 [00]	[0001000] [1100001]	0.175
539	[001010•] [110101]	[010100]1 [101]	[0101]1 [10101]	0.173	540	[0001101] [110101•]	[λ]4 [010]	[0001101] [1101011]	0.172
541	[00001111] [110101•]	[11] [01010]2	[0001111] [1101011]	0.172	542	[0001100] [110101•]	[0] [1010]3	[0001100] [1101011]	0.172
543	[01101•] [1111]	[010]2 [111]	[011011] [1111]	0.171	544	[01101] [1110•]	[01101] [11100]	[01101] [11101]	0.168
545	[0010011] [101100•]	[011] [11000]2	[010011]1 [11001]	0.168	546	[0010011] [101101•]	[0010011] [1011010]	[011] [11011]2	0.168
547	[011110] [1111•]	[λ]5 [λ]	[0]4 [1]	0.167	548	[0000110] [110110•]	[000110]1 [1100]	[0000110] [1101101]	0.165
549	[0000110] [110101•]	[0000110] [1101010]	[0000110] [1101011]	0.165	550	[0000101] [110110•]	[000101]1 [1100]	[0000101] [1101101]	0.165

n	s	$T(s,0)$	$T(s,1)$	$d(s)$	n	s	$T(s,0)$	$T(s,1)$	$d(s)$
551	[0000111] [1101011]	[0000111] [1101010]	[0000111] [1101011]	0.165	552	[0000111] [1101100]	[0000111] [1101100]	[0000111] [1101101]	0.165
553	[0010100] [1100000]	[0010100] [1100000]	[0010100] [1100001]	0.165	554	[0010101] [1100111]	[0101]3	[0101]3	0.165
555	[0010100] [1100111]	[0100]3	[0100 1001111]1	0.165	556	[0010100] [1100010]	[0010100] [1100010]	[0100 1000011]1	0.165
557	[0010100] [1100100]	[0010100] [1100100]	[00]5	0.165	558	[01011] [10]	[1011]1	[1011]1	0.164
559	[0001] [1]	[001]1	[0001] [1]	0.163	560	[0001100] [1111]	[00100]1	[00] [1111]1	0.162
561	[0111] [100]	[111]1	[111]1	0.162	562	[0101100] [101]	[10100]1	[10100]1	0.161
563	[0001100] [1100111]	[01100]2	[1]4	0.160	564	[0001101] [1100111]	[01101]2	[01101]2	0.160
565	[00100] [11]	[00100] [110]	[00100] [111]	0.160	566	[0001110] [1101100]	[10 01100]2	[0001110] [1101101]	0.159
567	[0010010] [1010000]	[0100010]1	[0100010]1	0.158	568	[0100011] [1000]	[10011]1	[10011]1	0.158
569	[001010] [11010]	[001010] [11010]	[010]1	0.158	570	[0001011] [1101100]	[1]4	[0001011] [1011011]	0.158
571	[001010] [1100]	[001010] [11000]	[001010] [11001]	0.158	572	[0001011] [101100]	[1]4	[0001011] [1011001]	0.158
573	[0001010] [1010000]	[001010]1 [10000]	[001010]1 [10001]	0.158	574	[0011000] [1000010]	[0011000] [1000010]	[1000 0000011]1	0.157
575	[0011000] [1000100]	[0011000] [1000100]	[1000 0001101]1	0.157	576	[0011000] [1000000]	[11000]2 [0000]	[11000]2 [0001]	0.157
577	[0001010] [1010110]	[0001010] [1010110]	[1]4	0.156	578	[0001110] [101000]	[10 01000]2	[001110]1 [1001]	0.156
579	[0001001] [1100111]	[01001]2 [110]	[001 1001111]1	0.154	580	[001000] [11100]	[001000] [111000]	[001000] [111001]	0.153
581	[0001011] [1101100]	[011 1011001]1	[011 1011101]1	0.152	582	[0001001] [1101110]	[001 1011110]1	[001 1011111]1	0.152
583	[0001010] [1101100]	[0001010] [1101100]	[010 1011101]1	0.152	584	[0001010] [1101000]	[001010]1 [1000]	[001010]1 [1001]	0.150
585	[0001010] [1101010]	[001010]1 [1010]	[010 1010111]1	0.150	586	[001011] [101]	[01011]1 [10]	[001011] [1011]	0.150
587	[0001101] [1101100]	[1]4 [100]	[0001101] [1101101]	0.148	588	[0001111] [1101100]	[11 01100]2	[11 01101]2	0.148
589	[0101101] [1001]	[10101] [0010]1	[10101]1 [0011]	0.148	590	[0001] [11]	[0001] [110]	[0001] [111]	0.146
591	[0001101] [1101000]	[1]4 [000]	[0001101] [1101001]	0.146	592	[0001100] [1101000]	[0001100] [1101000]	[0001100]1 [1001]	0.146
593	[01001] [1100]	[001 1000]1	[001 1001]1	0.144	594	[001110] [1110]	[00110] [11100]	[00110] [11101]	0.144
595	[01111] [1111]	[01111] [11110]	[01111] [11111]	0.143	596	[0001100] [1]	[00100]1 [1]	[0001100] [11]	0.143
597	[0010010] [10001]	[1010]2 [010]	[1010]2 [011]	0.139	598	[0110000] [1101]	[00]3 [10]	[000]3 [1]	0.139
599	[01001] [10]	[1001]1 [00]	[1001]1 [01]	0.138	600	[000001] [10]	[001]2 [1]	[0001]1 [1]	0.136
601	[0101111] [1000]	[10111]1 [0000]	[10111]1 [0001]	0.135	602	[0011100] [1001]	[100]3 [0]	[00]4 [1]	0.134
603	[00111] [1000]	[00111] [10000]	[00111] [10001]	0.132	604	[000010] [1110]	[00010] [11100]	[0 111]1	0.131
605	[00010] [110]	[00010] [1100]	[00010] [1101]	0.131	606	[001110] [100]	[00110] [1000]	[10]3 [1]	0.131
607	[0000111] [1000]	[11]4 [1]	[011]3 [1]	0.130	608	[0001110] [100]	[110]3 [1]	[0110]2 [1]	0.129
609	[0011100] [1101]	[0 10]3	[0 11]3	0.129	610	[0011010] [1010010]	[1010 010010]1	[011010]1 [10011]	0.128
611	[0110111] [1001]	[11011]1 [0010]	[11011]1 [0011]	0.128	612	[0000010] [111]	[00010]1 [1]	[00010] [1111]	0.126
613	[0000000] [1]	[00000]1 [0]	[0000000] [0]	0.124	614	[0000001] [1111110]	[0000010]1 [1]	[1 11110]2	0.124
615	[0110] [1011]	[1100]1 [011]	[1101]1 [011]	0.123	616	[0000010] [1111110]	[0000100]1 [1]	[01 111110]1	0.122
617	[0101] [1011]	[1010]1 [011]	[1011]1 [011]	0.122	618	[001010] [10000]	[001010] [10000]	[001010] [10001]	0.121
619	[001010] [10011]	[10]4 [1]	[001010] [10011]	0.121	620	[0000010] [111110]	[00010]2 [1]	[000010]1 [1]	0.121
621	[0000011] [1111100]	[0000011] [1111100]	[1 11101]2	0.121	622	[0111] [1]	[111]1 [1]	[0111] [1]	0.118
623	[0000101] [1111100]	[001010]2 [1]	[00001011] [1111100]	0.118	624	[0000100] [1111101]	[0001000]1 [1]	[001 111101]1	0.118
625	[0000100] [1111100]	[001000]2 [1]	[001001]2 [1]	0.118	626	[010010] [100]	[10010]1 [0]	[10010]1 [01]	0.117
627	[010010] [101]	[10010]1 [010]	[10010]1 [011]	0.117	628	[01110] [1]	[1110]1 [0]	[01110] [11]	0.116
629	[01011] [1011]	[1011]1 [0110]	[1011]1 [0111]	0.115	630	[01011] [1010]	[1011]1 [0100]	[1011]1 [0101]	0.115
631	[0000110] [1111100]	[00001100] [1111100]	[00001101] [1111100]	0.114	632	[01011] [1111]	[011 1110]1	[011 1111]1	0.112
633	[011101] [10111]	[11101]1 [01110]1	[11101]1 [01111]	0.111	634	[0100010] [110100]	[010010] [1101000]	[0010 10101]1	0.110
635	[01010] [1010]	[1010]1 [0100]	[1010]1 [0101]	0.109	636	[0000111] [1111000]	[0000111] [1111000]	[λ 1001]3	0.108
637	[0000110] [1111100]	[0110]3 [λ]	[00110]2 [1]	0.108	638	[011100] [1111]	[00]3 [10]	[00]3 [11]	0.106
639	[011100] [1101]	[100]2 [010]	[100]3 [λ]	0.106	640	[011100] [1110]	[λ]5 [λ]	[0]4 [1]	0.106
641	[000000] [10001]	[000]3 [1]	[000001] [10001]	0.106	642	[0110000] [1011]	[11000]1 [0110]	[11000]1 [0111]	0.105
643	[010010] [110]	[010010] [1100]	[010010] [1101]	0.105	644	[000011] [100]	[000011] [1000]	[0011]2 [1]	0.104
645	[0000011] [1110111]	[0 10111]2	[00000111] [1110111]	0.104	646	[0000011] [1110101]	[0000110]1 [101]	[λ 0101]3	0.104
647	[0000011] [1110110]	[0000110]1 [110]	[λ 0110]3	0.104	648	[001010] [111000]	[001010] [111000]	[001010] [111001]	0.104

n	s	$T(s,0)$	$T(s,1)$	$d(s)$	n	s	$T(s,0)$	$T(s,1)$	$d(s)$
649	[001000 11110•]	[001000 11110]	[001000 11110]	0.104	650	[01101 1•]	[1101 0]1	[01101 11]	0.102
651	[000101 1000•]	[101 0]3	[01 λ]4	0.101	652	[011000 1110•]	[0(0) 100]2	[000 101]2	0.100
653	[0011 1•]	[0011 10]	[0011 11]	0.100	654	[00100001• 1000100]	[00100010 1000100]	[100011 0100]2	0.100
655	[00100000• 1000100]	[00 100]4	[100001 0100]2	0.100	656	[0001100 111001•]	[00 10010]2	[01100 11]2	0.099
657	[011000 110•]	[0 λ]4	[011000 1101]	0.099	658	[001010 1010•]	[001010 10100]	[001010 10101]	0.098
659	[00000111• 1110011]	[00001110 1110011]	[00001111 1110011]	0.098	660	[0000110• 1110110]	[0001100 110]	[01 10110]2	0.098
661	[0000111• 1110111]	[λ 111]4	[00001111 1110111]	0.098	662	[001000 10•]	[1000 λ]2	[01000 1]1	0.098
663	[0000111• 1110010]	[λ 010]4	[00001111 1110010]	0.098	664	[0000101• 1110110]	[0001010 110]1	[011 110110]1	0.098
665	[0000110• 1110101]	[0 01100 101]1	[01 10101]2	0.098	666	[0000111• 1110100]	[λ 100]4	[1 0100]3	0.098
667	[0101 101•]	[101 010]1	[101 011]1	0.097	668	[0010011• 1001100]	[λ 0]6	[001 111 100]3	0.096
669	[000111 1111•]	[λ 10]3	[λ 11]3	0.095	670	[0000111• 1101110]	[0001110 1110]1	[11 01110]2	0.095
671	[0000110• 1101110]	[00001100 1101110]	[λ 110]4	0.095	672	[0010111• 1011011]	[0101110 11011]1	[11 011]4	0.094
673	[00100000• 1000010]	[0 10]5	[λ 0]6	0.093	674	[0010010• 1000001]	[100 0001]3	[1001101 0001]2	0.093
675	[0001100• 1110000]	[000 10000]2	[11001 0]3	0.093	676	[0001100• 1110001]	[0001 1000 1110001]	[1001 λ]4	0.093
677	[0001100• 1101110]	[0011000 1110]1	[01 1110]3	0.093	678	[0001111• 1101111]	[0011110 1111]1	[00111111 1111]1	0.093
679	[001101 11100•]	[01 1000]2	[001101 111001]	0.093	680	[001101 11101•]	[01 1010]2	[01 10111]2	0.093
681	[000000 11•]	[00000 λ]1	[000000 111]	0.092	682	[0001100• 1100100]	[0001100 1100100]	[λ 00]5	0.092
683	[0001100• 1100101]	[00011000 1100101]	[λ 01]5	0.092	684	[0001000• 1000010]	[0001 0000 1000010]	[10001 010]3	0.092
685	[010101 10101•]	[10101 01010]1	[10101 01011]1	0.092	686	[01011 1000•]	[1011 0000]1	[1011 0001]1	0.092
687	[001000 11•]	[01000 λ]1	[001000 111]	0.092	688	[0001010 111010•]	[001010 100]1	[001010 101]1	0.092
689	[001010 100•]	[001010 1000]1	[001010 1001]1	0.092	690	[0001011 111010•]	[001011 100]1	[011 110101]1	0.092
691	[00101000• 1000001]	[1000 00001]2	[101001 0001]2	0.091	692	[00101000• 1000010]	[00101000 1000010]	[101001 0010]2	0.091
693	[0001110• 1101011]	[00111100 11011]1	[101 01011]2	0.091	694	[0001100• 1100010]	[0001 1000 1100010]	[1001 0]4	0.090
695	[0001100• 1100000]	[00011000 1100000]	[11001 00]3	0.090	696	[0001100• 1100001]	[λ 01]5	[11001 01]3	0.090
697	[0011011• 1011110]	[01101110 11110]1	[10111 011110]1	0.090	698	[0011011• 1011101]	[01101110 11101]1	[10111 011101]1	0.090
699	[01000 λ•]	[1000 λ]1	[01000 1]	0.090	700	[0010000• 1100000]	[00000 100000]1	[100001 000]2	0.090
701	[0010000• 1100010]	[00000 100010]1	[00001 100010]1	0.090	702	[0010001• 1100010]	[00010 100010]1	[00100011 1100010]	0.090
703	[0001011• 1110011]	[0110 110011]1	[010111 11]2	0.090	704	[0001000• 1110000]	[00010000 1110000]	[10001 0]3	0.090
705	[000010 11•]	[000010 10]	[000010 11]	0.089	706	[01111 1110•]	[01111 1110]	[01111 1111]	0.089
707	[001000 λ•]	[01000 λ]1	[000 λ]1	0.089	708	[001110 1001•]	[110 0]3	[10 λ]4	0.088
709	[0001000• 1100001]	[0000 100001]1	[10001 01]3	0.087	710	[011110 10111•]	[11110 01110]1	[11110 01111]1	0.086
711	[01000 1000]	[1000 000]1	[1001 000]1	0.086	712	[0001100• 1101011]	[001100 1011]1	[01 1011]3	0.086
713	[0001111• 1101011]	[0011110 1011]1	[111 01011]2	0.086	714	[0001101• 1101011]	[λ 11]5	[1 011]4	0.086
715	[010011 11001•]	[0011 10010]1	[0011 10011]1	0.084	716	[0010011• 1011010]	[0110 11010]2	[0100111 11010]1	0.084
717	[0000101• 1101101]	[010 1011101]1	[011 1011101]1	0.082	718	[0000110• 1101010]	[0001100 1010]1	[λ 010]4	0.082
719	[000110 1100•]	[000110 11000]	[000110 11001]	0.082	720	[0000111• 1101010]	[0001110 1010]1	[00011111 1010]1	0.082
721	[000101 1100•]	[101 λ]3	[0101 1]2	0.082	722	[0000111• 1101100]	[0001110 1100]1	[00011111 1101100]	0.082
723	[0000111• 1101011]	[0001110 1011]1	[0001111 1011]1	0.082	724	[0000110• 1101101]	[0 1101]3	[λ 101]4	0.082
725	[0000110• 1101011]	[0 1011]3	[λ 011]4	0.082	726	[0000111• 1101101]	[0000 1110 1101101]	[11 01101]2	0.082
727	[0010100• 1100001]	[01000 100001]1	[101001 001]2	0.082	728	[0010100• 1100000]	[0010 1000 1100000]	[101001 000]2	0.082
729	[0010100• 1100010]	[01000 100010]1	[101001 010]2	0.082	730	[0010100• 1100100]	[1000 0]4	[1001 0]4	0.082
731	[00100 111•]	[00100 1110]	[00 111]1	0.080	732	[0001110• 1101101]	[0011100 1101]1	[101 01101]2	0.079
733	[010010 10000•]	[10010 00000]1	[10010 00001]1	0.079	734	[010010 10001•]	[10010 00010]1	[10010 00011]1	0.079
735	[0001011• 1011001]	[λ 01]5	[0010111 11001]1	0.079	736	[0001011• 1011011]	[λ 11]5	[0010111 1011011]	0.079
737	[0011000• 1000100]	[110000 0100]2	[110001 0100]2	0.078	738	[0011000• 1000010]	[10000 000010]1	[110001 0010]2	0.078
739	[0001010• 1010110]	[0010100 10110]1	[λ 10]5	0.078	740	[01001 110•]	[01001 1100]	[001 λ]2	0.077
741	[0001010• 1101100]	[0010100 1100]1	[0101 1011100]1	0.076	742	[010011 1000]	[10011 000]1	[10011 001]1	0.076
743	[010001 110•]	[0001 100]1	[0001 λ]2	0.076	744	[010000 1100]	[0000 100]1	[0000 λ]2	0.076
745	[0001101• 1101101]	[0 101]4	[λ 01]5	0.074	746	[0001100• 1101000]	[00 1000]3	[0011001 1000]1	0.073

n	s	$T(s,0)$	$T(s,1)$	$d(s)$	n	s	$T(s,0)$	$T(s,1)$	$d(s)$
747	[00001101•] [1101001]	[λ] [01] 5	[0011011] [1001] 1	0.073	748	[000000•] [10000]	[00] [λ] 4	[000001] [10000]	0.071
749	[00111110] [1111•]	[λ] [λ] 5	[0] [1] 4	0.069	750	[01100] [λ•]	[1100] [λ] 1	[100] [λ] 1	0.067
751	[0100•] [1011]	[1000] [011] 1	[1001] [011] 1	0.067	752	[011110] [111•]	[10] [0] 3	[011110] [1111]	0.066
753	[010110] [1110•]	[0110] [1100] 1	[0110] [1101] 1	0.066	754	[010110] [1101•]	[110] [λ] 3	[0110] [1011] 1	0.066
755	[00011100] [1010•]	[001100] [100] 1	[001100] [101] 1	0.066	756	[00001] [1•]	[00001] [10]	[00001] [11]	0.065
757	[010010] [1010•]	[100010] [0100] 1	[100010] [0101] 1	0.064	758	[011010] [10011•]	[11010] [00110] 1	[11010] [00111] 1	0.064
759	[0011] [λ•]	[011] [λ] 1	[0011] [1]	0.064	760	[000010] [1111•]	[000010] [11110]	[000010] [11111]	0.063
761	[000001] [10001•]	[001] [10] 3	[001] [11] 3	0.063	762	[000110] [10•]	[000110] [100]	[00110] [1] 1	0.062
763	[00001] [λ•]	[0001] [λ] 1	[00001] [1]	0.062	764	[0000010] [λ•]	[000010] [λ] 1	[0000010] [1]	0.062
765	[0000100] [λ•]	[000100] [λ] 1	[0000100] [1]	0.061	766	[000001] [10•]	[0001] [λ] 2	[00001] [1] 1	0.061
767	[00000011•] [1111100]	[000110] [λ] 2	[λ] [1100] 3	0.060	768	[001010] [λ•]	[01010] [λ] 1	[010] [λ] 1	0.059
769	[00001011] [1111100•]	[01011] [λ] 3	[011] [1111001] 1	0.059	770	[0001000] [1•]	[001000] [λ] 1	[000] [1] 1	0.059
771	[001001] [λ•]	[01001] [λ] 1	[001] [λ] 1	0.059	772	[01101] [100•]	[1101] [000] 1	[1101] [001] 1	0.059
773	[000010] [λ•]	[00010] [λ] 1	[000010] [1]	0.058	774	[00110] [λ•]	[0110] [λ] 1	[00110] [1]	0.058
775	[00001101] [1111100•]	[01101] [λ] 3	[01] [111001] 2	0.057	776	[00001100] [1111100•]	[01100] [λ] 3	[001100] [1] 2	0.057
777	[01010] [λ•]	[1010] [λ] 1	[01010] [1]	0.057	778	[0101•] [1010]	[1010] [010] 1	[1011] [010] 1	0.055
779	[01110] [10•]	[1110] [00] 1	[1110] [01] 1	0.054	780	[00001111•] [1111000]	[01110] [λ] 3	[λ] [000] 4	0.054
781	[000100] [λ•]	[00100] [λ] 1	[000100] [1]	0.053	782	[01110] [λ•]	[1110] [λ] 1	[01110] [1]	0.053
783	[00000111] [1110111•]	[λ] [01110] 3	[00000111] [11011111]	0.052	784	[0000110] [101•]	[000110] [10] 1	[000110] [11] 1	0.052
785	[0000110] [110•]	[00110] [λ] 2	[λ] [1] 3	0.052	786	[011101] [11111•]	[01] [110] 3	[01] [111] 3	0.051
787	[011101] [11110•]	[01] [100] 3	[01] [101] 3	0.051	788	[011101] [11101•]	[λ] [0] 5	[λ] [1] 5	0.051
789	[011110] [1011•]	[1110] [0110] 1	[1110] [0111] 1	0.050	790	[00100010] [1000100•]	[100010] [01000] 2	[λ] [01] 6	0.050
791	[001110] [1011•]	[01110] [110] 1	[01110] [111] 1	0.049	792	[001111] [1011•]	[01111] [110] 1	[01111] [111] 1	0.049
793	[001110] [1101•]	[10] [010] 2	[01110] [111] 1	0.049	794	[01111] [1100]	[λ] [00] 3	[01111] [11101]	0.049
795	[00001111] [1110011•]	[1] [00110] 3	[0001111] [111] 2	0.049	796	[0001100] [110•]	[01100] [λ] 2	[0] [1] 3	0.049
797	[00001110] [1110011•]	[λ] [0110] 4	[λ] [0111] 4	0.049	798	[00001111] [1110111•]	[1] [01110] 3	[00001111] [11101111]	0.049
799	[0001010] [110•]	[01010] [λ] 2	[010] [101] 1	0.049	800	[001001] [100•]	[1001] [0] 2	[001] [λ] 3	0.049
801	[00001111] [1110010•]	[001111] [100] 2	[00001111] [11100101]	0.049	802	[0001100] [101•]	[0001100] [1010]	[001100] [1] 1	0.049
803	[0001110] [1110•]	[λ] [0] 4	[λ] [1] 4	0.048	804	[00001100] [1101100•]	[0001100] [11100] 1	[0] [11101] 3	0.048
805	[0101110] [11011•]	[01110] [10110] 1	[01110] [10111] 1	0.047	806	[00011000] [1100001•]	[000] [100010] 2	[000] [λ] 5	0.047
807	[0011000] [1110•]	[000] [100] 2	[000] [101] 2	0.047	808	[0011111] [1111•]	[1] [0] 4	[λ] [λ] 5	0.047
809	[000001] [10000•]	[λ] [λ] 5	[01] [1] 4	0.046	810	[0000000] [111•]	[00000] [λ] 1	[000000] [1111]	0.046
811	[00011000] [1100101•]	[0] [1010] 4	[011000] [1011] 2	0.046	812	[00011000] [1100100•]	[011000] [1000] 2	[0] [1001] 4	0.046
813	[01110] [101•]	[1110] [010] 1	[1110] [011] 1	0.046	814	[00010000] [1000010•]	[10000] [0100] 3	[λ] [101] 5	0.046
815	[01010•] [1010]	[10100] [0101] 1	[10101] [0101] 1	0.046	816	[01010•] [10100]	[10100] [λ] 1	[10101] [0100] 1	0.046
817	[001010] [101•]	[001010] [1010]	[0] [1] 3	0.046	818	[00101000] [1000010•]	[1010000] [00100] 2	[1000] [0001101] 2	0.046
819	[0011100] [1011•]	[011100] [110] 1	[011100] [111] 1	0.046	820	[0001000] [1100010•]	[1000] [00] 4	[λ] [101] 5	0.045
821	[00011000] [1100000•]	[11000] [000] 3	[λ] [001] 5	0.045	822	[0110110] [11110•]	[0110] [1100] 2	[0110] [1101] 2	0.045
823	[0110110] [11101•]	[0110] [1010] 2	[0110] [1011] 2	0.045	824	[001111] [111•]	[1] [0] 3	[001111] [1111]	0.045
825	[011111] [11111•]	[λ] [0] 5	[λ] [1] 5	0.045	826	[00100011] [1100010•]	[100011] [0100] 2	[00011] [1000101] 1	0.045
827	[010111] [11•]	[0111] [10] 1	[0111] [11] 1	0.045	828	[00010000] [1100000•]	[10000] [00] 3	[0000] [1100001] 1	0.045
829	[0011000] [1011•]	[011000] [110] 1	[011000] [111] 1	0.043	830	[01011] [111•]	[01011] [1110]	[01011] [1111]	0.043
831	[0011110] [1011•]	[011110] [110] 1	[011110] [111] 1	0.043	832	[01111] [110•]	[111] [00] 2	[11] [01] 2	0.042
833	[0100111] [11010•]	[111] [λ] 4	[0111] [1] 3	0.042	834	[0001110] [1010•]	[001110] [100] 1	[001110] [101] 1	0.041
835	[0001111] [1010•]	[001111] [100] 1	[001111] [101] 1	0.041	836	[0001110] [1100•]	[1110] [λ] 3	[01110] [1] 2	0.041
837	[0001110] [1011•]	[001110] [110] 1	[001110] [111] 1	0.041	838	[0001111] [1011•]	[001111] [110] 1	[001111] [111] 1	0.041
839	[00001111] [1101100•]	[0001111] [11000] 1	[11] [011001] 2	0.041	840	[00001110] [1101101•]	[0001110] [11010] 1	[0001110] [11011] 1	0.041
841	[010110] [110•]	[0110] [100] 1	[010110] [101]	0.041	842	[010110] [111•]	[010110] [1110]	[0110] [111] 1	0.041
843	[010110] [101•]	[10110] [010] 1	[10110] [011] 1	0.041	844	[00101000] [1100000•]	[1010000] [0000] 2	[01000] [1000001] 1	0.041

n	s	$T(s,0)$	$T(s,1)$	$d(s)$
845	[00100 11100•]	[00100 11100]	[00100 11101]	0.040
847	[0010111 11001•]	[111 λ]4	[0111 1]3	0.039
849	[0010100 10110•]	[0101100 1100]1	[0101100 1101]1	0.039
851	[0010100 1100•]	[0100 1000]1	[0100 λ]3	0.038
853	[0011001 1000•]	[11001 00]2	[11001 01]2	0.036
855	[01111 11•]	[01111 110]	[01111 111]	0.035
857	[000110 λ•]	[00110 λ]1	[0 1]2	0.034
859	[001100 101•]	[01100 10]1	[01100 11]1	0.033
861	[011100 111•]	[011100 1110]	[011100 1111]	0.033
863	[01101 10111•]	[1101 0110]1	[1101 0111]1	0.032
865	[0000010 1•]	[000010 λ]1	[000010 11]	0.031
867	[000110 λ•]	[00110 λ]1	[00110 1]	0.030
869	[0000110 1100•]	[0110 λ]3	[00110 1]2	0.030
871	[01101 λ•]	[1101 λ]1	[01101 1]	0.029
873	[000001111• 111011111]	[λ 1111]4	[00001111 11101111]	0.026
875	[000011110• 111011111]	[00011110 1111]1	[11 01111]3	0.024
877	[01110• 10110]	[11100 0110]1	[11100 0110]1	0.023
879	[011101 111•]	[011101 1110]	[011101 1111]	0.023
881	[011000 1000•]	[11000 0000]1	[11000 0001]1	0.023
883	[01010 10110•]	[1010 0110]1	[1010 0111]1	0.023
885	[001111 1111•]	[λ 0]4	[λ 1]4	0.023
887	[011000 1110•]	[01100 1110]	[000 11]2	0.021
889	[01011 11100•]	[011 11100]1	[01011 11101]	0.021
891	[001110 1010•]	[01110 10]1	[001110 1011]	0.021
893	[001110 1100•]	[1110 00]2	[001110 1101]	0.021
895	[0001111 110000•]	[1111 0]3	[11 λ]4	0.021
897	[0001110 110100•]	[0001110 1101000]	[0001110 1101001]	0.021
899	[010111 1111•]	[0111 1110]1	[0111 1111]1	0.021
901	[011100 110•]	[011100 1100]	[011100 1101]	0.020
903	[010100 11000•]	[0100 1000]1	[0100 1001]1	0.020
905	[01100 10110•]	[1100 0110]1	[1100 0111]1	0.019
907	[0000010 11•]	[000010 λ]1	[000010 111]	0.015
909	[000110 1•]	[00110 10]	[00110 11]	0.015
911	[01001 1•]	[1001 0]1	[001 1]1	0.015
913	[000011111 111001010•]	[11 001010]3	[001111 1011]2	0.012
915	[000111110 1111•]	[λ λ]5	[0 1]4	0.012
917	[011101 1110•]	[1 0]4	[11101 1110]	0.012
919	[01000 11000•]	[1000 0000]1	[1000 0001]1	0.011
921	[010001 10000•]	[1001 0000]1	[10001 0001]1	0.010
923	[0000010 111•]	[000010 λ]1	[0 111]1	0.008
925	[00001111 11111•]	[λ 10]4	[λ 11]4	0.007
927	[0011111 10110•]	[011111 110]1	[011111 111]1	0.006
929	[00101 101•]	[00101 1010]	[00101 1011]	0.006
931	[011111 1110•]	[11 0]3	[λ 1]4	0.003

n	s	$T(s,0)$	$T(s,1)$	$d(s)$
846	[0011100 1101•]	[100 010]2	[011100 11]1	0.040
848	[00010111 10110110•]	[1 0110]4	[0010111 110111]1	0.039
850	[001100 111•]	[001100 110]	[00 1]2	0.039
852	[010011 1010•]	[10011 0100]1	[10011 0101]1	0.038
854	[00110011 1001•]	[1011 0]3	[011 λ]4	0.036
856	[0110• 1010]	[1100 010]1	[1101 010]1	0.034
858	[001100 100•]	[1100 0]2	[001100 1001]	0.033
860	[011100 110•]	[100 00]2	[011100 1101]	0.033
862	[00001 11•]	[0001 λ]1	[λ 11]1	0.032
864	[01101 10110•]	[1101 0100]1	[1101 0101]1	0.032
866	[0000100 1•]	[000100 λ]1	[000100 111]	0.031
868	[01011 λ•]	[1011 λ]1	[01011 1]	0.030
870	[01001 λ•]	[1001 λ]1	[01001 1]	0.030
872	[001100 1•]	[01100 λ]1	[001100 11]	0.029
874	[00001111• 11100101]	[10 00101]3	[0000111111 11100101]	0.024
876	[0001100 11000•]	[0001100 111000]	[0001100 111001]	0.024
878	[01110• 10111]	[11100 0111]1	[11101 0111]1	0.023
880	[000000 1111•]	[00000 λ]1	[000000 11111]	0.023
882	[01111 1•]	[1111 0110]1	[1111 0111]1	0.023
884	[010111 11011•]	[010111 110110]	[01111 10111]1	0.023
886	[01110101 11100•]	[10 00]4	[10 01]4	0.022
888	[011110 110•]	[110 00]2	[110 01]2	0.021
890	[001110 100•]	[1110 0]2	[001110 1001]	0.021
892	[001111 101•]	[01111 10]1	[001111 1011]	0.021
894	[001111 110•]	[1111 00]2	[11 01]2	0.021
896	[011001 11•]	[001 λ]3	[001 1]2	0.021
898	[0001110 110111•]	[0001110 1101110]	[001110 111]1	0.021
900	[0111110 10111•]	[111110 01110]1	[111110 01111]1	0.021
902	[00101011 1101110•]	[0111 1011110]1	[0111 1011111]1	0.020
904	[0101100 1101•]	[0100 1010]1	[0100 1011]1	0.020
906	[011001 1010•]	[1100 0101]1	[1100 0101]1	0.019
908	[0000100 11•]	[000100 λ]1	[00 11]1	0.015
910	[01011 1•]	[1011 0]1	[011 1]1	0.015
912	[000001111 1110111110•]	[1 01110]3	[0000111 11111]1	0.013
914	[00101 110•]	[00101 1100]	[00101 101]	0.012
916	[011101 11110•]	[011101 11110]	[011101 11111]	0.012
918	[0101• 1000]	[1010 0000]1	[1011 0000]1	0.011
920	[01111 10•]	[1111 00]1	[1111 01]1	0.010
922	[010000 10000•]	[10000 0000]1	[10000 0001]1	0.010
924	[011110 1•]	[11110 0]1	[λ10]2	0.007
926	[001010 11•]	[01010 λ]1	[010 11]1	0.006
928	[00101 1000•]	[00101 1000]	[00101 1001]	0.006
930	[011111 1110•]	[11 00]2	[11 01]2	0.003

Appendix B

Recurrences for $\mathcal{H}_1(i), \ldots, \mathcal{H}_{52}(i)$

		$\binom{0}{0}$	$\binom{1}{1}$	$\binom{10}{0}$	$\binom{0}{10}$	$\binom{01}{1}$	$\binom{1}{01}$	$\binom{1+10}{0}$	$\binom{0}{1+10}$	$\binom{0+01}{1}$	$\binom{1}{0+01}$
1	$\mathcal{U}[\lambda, 0]$	1	1	2	2	2	2	3	3	3	3
2	$\mathcal{U}[10, 1]$	2	4	5	×	6	7	18	×	23	20
3	$\mathcal{U}[110, 1]$	8	29	10	×	45	41	52	×	19	20
4	$\mathcal{U}[110, 2]$	21	25	2	11	2	2	3	×	3	3
5	$\mathcal{U}[010, 1]$	3	2	12	×	5	×	12	×	18	×
6	$[\mathcal{U}_t(10, 1) \cap \mathcal{U}_t(1001, 2)]$	2	24	5	×	47	32	18	×	23	×
7	$[\mathcal{U}_t(10, 1) \cap \mathcal{U}_b(00, 1)]$	14	33	15	×	26	7	35	×	×	20
8	$\mathcal{U}[100, 1]$	5	24	13	×	47	32	13	×	23	×
9	$\mathcal{U}[00, 1]$	9	21	8	7	2	11	19	20	3	×
10	$\mathcal{U}[0100, 1]$	18	2	13	×	5	×	13	×	18	×
11	$[\mathcal{U}_t(10, 1) \cap \mathcal{U}_b(01, 1)]$	16	16	15	×	×	15	35	×	×	35
12	$\mathcal{U}[0110, 1]$	19	5	12	×	13	×	12	×	13	×
13	$\mathcal{U}[1010, 1]$	13	3	13	×	12	×	13	×	12	×
14	$[\mathcal{U}_t(10, 1) \cap \mathcal{U}_b(11, 1)]$	38	27	17	×	40	39	×	×	20	19
15	$[\mathcal{U}_t(010, 1) \cap \mathcal{U}_b(11, 1)]$	3	14	17	×	15	×	×	×	35	×
16	$[\mathcal{U}_t(10, 1) \cap \mathcal{U}_b(110, 2)]$	38	4	17	×	6	7	×	×	23	20
17	$[\mathcal{U}_t(010, 1) \cap \mathcal{U}_b(10, 1)]$	51	16	×	×	15	×	×	×	35	×
18	$\mathcal{U}[0010, 1]$	3	8	12	×	10	×	12	×	52	×
19	$\mathcal{U}[1100, 1]$	12	29	13	×	45	32	13	×	19	×
20	$[\mathcal{U}_t(110, 1) \cap \mathcal{U}_b(00, 1)]$	31	29	49	×	30	41	35	×	×	20

		$\binom{0}{0}$	$\binom{1}{1}$	$\binom{10}{0}$	$\binom{0}{10}$	$\binom{01}{1}$	$\binom{1}{01}$	$\binom{1+10}{0}$	$\binom{0}{1+10}$	$\binom{0+01}{1}$	$\binom{1}{0+01}$
21	$\mathcal{U}[100,2]$	22	1	2	34	2	28	3	3	3	20
22	$\mathcal{U}[010,2]$	4	1	6	7	2	34	23	20	3	3
23	$[\mathcal{U}_t(110,1)\cap\mathcal{U}_t(1001,2)]$	8	29	10	×	45	32	52	×	19	×
24	$\mathcal{U}[1100,2]$	42	25	2	26	2	28	3	×	3	20
25	$\mathcal{U}[1110,3]$	43	1	2	36	2	2	3	32	3	3
26	$[\mathcal{U}_t(10,1)\cap\mathcal{U}_t(1001,2)\cap\mathcal{U}_b(01,1)]$	16	16	15	×	×	17	15	×	×	×
27	$[\mathcal{U}_t(00,1)\cap\mathcal{U}_b(001,2)]$	9	21	8	7	11	11	19	20	×	×
28	$[\mathcal{U}_t(10,1)\cap\mathcal{U}_b(001,2)]$	2	33	5	×	26	7	18	×	×	20
29	$[\mathcal{U}_t(00,1)\cap\mathcal{U}_t(0001,2)]$	9	21	8	7	2	30	19	20	3	×
30	$[\mathcal{U}_t(100,1)\cap\mathcal{U}_b(01,1)]$	5	16	13	×	×	17	13	×	×	×
31	$[\mathcal{U}_t(100,1)\cap\mathcal{U}_b(11,1)]$	5	27	17	×	40	30	×	×	20	×
32	$[\mathcal{U}_t(110,1)\cap\mathcal{U}_b(01,1)]$	50	14	49	×	×	15	52	×	×	35
33	$[\mathcal{U}_t(110,2)\cap\mathcal{U}_b(100,2)]$	22	25	34	11	28	2	3	×	20	3
34	$[\mathcal{U}_t(10,1)\cap\mathcal{U}_b(010,2)]$	16	4	15	×	6	7	35	×	23	20
35	$[\mathcal{U}_t(0010,1)\cap\mathcal{U}_b(11,1)]$	3	31	12	×	49	×	×	×	52	×
36	$[\mathcal{U}_t(10,1)\cap\mathcal{U}_b(011,2)]$	2	4	5	×	6	7	35	×	23	20
37	$[\mathcal{U}_t(100,1)\cap\mathcal{U}_b(10,1)]$	5	33	×	×	40	32	×	×	20	×
38	$[\mathcal{U}_t(10,1)\cap\mathcal{U}_b(100,2)]$	2	4	5	×	6	7	18	×	20	20
39	$[\mathcal{U}_t(100,1)\cap\mathcal{U}_b(00,1)]$	5	24	13	×	26	32	13	×	×	×
40	$[\mathcal{U}_t(10,1)\cap\mathcal{U}_t(1001,2)\cap\mathcal{U}_b(00,1)]$	14	33	15	×	26	32	35	×	×	×
41	$[\mathcal{U}_t(10,1)\cap\mathcal{U}_b(000,1)]$	14	33	15	×	30	7	35	×	×	20
42	$\mathcal{U}[0110,2]$	24	22	47	32	2	34	23	×	3	3
43	$\mathcal{U}[1000,3]$	1	1	2	2	2	2	3	3	3	44
44	$[\mathcal{U}_t(110,1)\cap\mathcal{U}_b(001,2)]$	8	29	10	×	30	41	52	×	×	20
45	$[\mathcal{U}_t(100,1)\cap\mathcal{U}_t(10001,2)]$	5	24	13	×	6	46	13	×	23	×
46	$[\mathcal{U}_t(1100,1)\cap\mathcal{U}_b(01,1)]$	12	14	13	×	×	17	13	×	×	×
47	$[\mathcal{U}_t(10,1)\cap\mathcal{U}_t(10011,2)]$	2	48	5	×	47	32	18	×	23	×
48	$\mathcal{U}[11001,2]$	42	24	2	26	47	32	3	×	23	×
49	$[\mathcal{U}_t(0100,1)\cap\mathcal{U}_b(11,1)]$	18	14	17	×	15	×	×	×	35	×
50	$[\mathcal{U}_t(100,1)\cap\mathcal{U}_b(110,2)]$	5	24	17	×	47	32	×	×	23	×
51	$[\mathcal{U}_t(110,1)\cap\mathcal{U}_b(10,1)]$	37	27	×	×	39	40	×	×	19	20
52	$\mathcal{U}[00100,1]$	18	8	13	×	10	×	13	×	52	×

Appendix C

Css machine for adaptability

s	$T(s,0)$	$T(s,1)$	d
$s_1 = \binom{0110}{1}$	$\binom{.1101}{.0} +$	$\binom{0110}{11}$	0.105083
$s_2 = \binom{0100}{1}$	$\binom{.1001}{.0} +$	$\binom{0100}{11}$	0.104168
$s_3 = \binom{0100}{\lambda}$	$\binom{1001}{\lambda} +$	$\binom{0100}{1}$	0.100481
$s_4 = \binom{0010}{1}$	$\binom{0010}{10}$	$\binom{0010}{11}$	0.079724
$s_5 = \binom{0110}{\lambda}$	$\binom{.1101}{.\lambda} +$	$\binom{0110}{1}$	0.055223
$s_6 = \binom{0110}{11}$	$\binom{0110}{110}$	$\binom{...0100}{..1} ++$	0.055033
$s_7 = \binom{0100}{11}$	$\binom{0100}{110}$	$\binom{0100}{111}$	0.054731
$s_8 = \binom{0010}{\lambda}$	$\binom{.0101}{.\lambda} +$	$\binom{0010}{1}$	0.049381
$s_9 = \binom{0010}{11}$	$\binom{0010}{110}$	$\binom{0010}{111}$	0.042353
$s_{10} = \binom{0010}{10}$	$\binom{..1011}{...\lambda} ++$	$\binom{0010}{101}$	0.039862
$s_{11} = \binom{0101}{\lambda}$	$\binom{.1011}{.\lambda} +$	$\binom{0101}{1}$	0.035927
$s_{12} = \binom{0010}{111}$	$\binom{0010}{1110}$	$\binom{.....0100}{...1} +++$	0.034859

s	$T(s,0)$	$T(s,1)$	d
$s_{13} = \binom{0110}{110}$	$\binom{....1001}{...0} + ++$	$\binom{0110}{1101}$	0.027516
$s_{14} = \binom{0100}{110}$	$\binom{....1011}{....\lambda} + ++$	$\binom{..0010}{....\lambda} + +$	0.027366
$s_{15} = \binom{0100}{111}$	$\binom{0100}{1110}$	$\binom{..0010}{.111} +$	0.027366
$s_{16} = \binom{0010}{110}$	$\binom{..1011}{....\lambda} + +$	$\binom{0010}{1101}$	0.021177
$s_{17} = \binom{0010}{101}$	$\binom{0010}{1010}$	$\binom{0010}{1011}$	0.019931
$s_{18} = \binom{0101}{1}$	$\binom{.1011}{.0} +$	$\binom{..0110}{.1} +$	0.017964
$s_{19} = \binom{0010}{1110}$	$\binom{..1011}{.....\lambda} + +$	$\binom{0010}{11101}$	0.017430
$s_{20} = \binom{0110}{1101}$	$\binom{.....0101}{.....\lambda} + + + + ++$	$\binom{...0100}{.....\lambda} + ++$	0.013758
$s_{21} = \binom{0100}{1110}$	$\binom{....1011}{.....\lambda} + ++$	$\binom{..0010}{.....\lambda} + +$	0.013683
$s_{22} = \binom{0010}{1101}$	$\binom{.0101}{...10} +$	$\binom{.....0100}{.....\lambda} + + + +$	0.010588
$s_{23} = \binom{0010}{1010}$	$\binom{0010}{10100}$	$\binom{.0101}{..101} +$	0.009965
$s_{24} = \binom{0010}{1011}$	$\binom{.....1001}{.....\lambda} + + + + ++$	$\binom{.....0100}{.....\lambda} + + + +$	0.009965
$s_{25} = \binom{0010}{11101}$	$\binom{.....0101}{.....\lambda} + + + + + +$	$\binom{.....0100}{.....\lambda} + + + +$	0.008715
$s_{26} = \binom{0101}{10}$	$\binom{.1011}{.00} +$	$\binom{..0110}{..1} + +$	0.005294
$s_{27} = \binom{0101}{101}$	$\binom{...1101}{...0} + ++$	$\binom{..0110}{..11} + +$	0.004983
$s_{28} = \binom{0010}{10100}$	$\binom{....1101}{....00} + ++$	$\binom{0010}{101001}$	0.004983
$s_{29} = \binom{0010}{101001}$	$\binom{....1101}{.....\lambda} + + + +$	$\binom{.....0100}{.....\lambda} + + + + ++$	0.002491

Notation

Σ – fixed alphabet $\{0, 1, \ldots, k-1\}$, 4

k – size of alphabet Σ, 4

Σ^n – set of sequences of length n over the Σ, 4

λ – empty sequence, 4

$|u|$ – the length of sequence u, 4

$\|u\|$ – the number of different symbols in sequence u, 4

Σ^* – set of all sequences over the Σ, 4

Σ^+ – set of all nonempty sequences over the Σ, 4

$u \sqsubseteq v$ – u is subsequence of v, 5

$u \trianglelefteq v$ – u is a prefix of v, 5

$u \triangleleft v$ – u is a proper prefix of v, 5

Π – the set of all pairs of sequences, 7

$l\binom{u}{v}$ – total length of the pair, 7

$t\binom{u}{v}$ – top sequence of the pair, 7

$b\binom{u}{v}$ – bottom sequence of the pair, 7

$\mathrm{cat}(p_1, p_2, \ldots, p_n)$ – the concatenation of the sequence of pairs, 7

$\mathbf{L}(u, v)$ – length of lcss, 8

$\mathbf{EL}_n$ – expected length of lcss of two sequences of length n, 14

γ_k – $\lim_{n\to\infty} \frac{\mathbf{EL}_n}{n}$, 15

$\mathbf{D}(u, v)$ – diagonal lcss of u and v, 18

$\mathbf{ED}_n$ – expected length of diagonal less of two sequences of length n, 18

δ_k – $\lim_{n\to\infty} \frac{\mathbf{ED}_n}{n}$, 19

$\mathrm{Var}(\mathbf{L}_n)$ – variance of less, 19

$\mathfrak{M}$ – finite state machine, 21

$I(s)$ – tape switch function, 21

$T(s,a)$ – transition function, 21

$O(s,a)$ – output function, 21

$s \xrightarrow{a} T(s,a)$ – transition, 21

$T^m(s,u,v)$ – extended transition function, 21

$T^*(s,u,v)$ – $T^{|u|+|v|}(s,u,v)$, 21

$\mathbf{C}_{\mathfrak{M}}(s,u,v,m)$ – computation function, 22

$\boldsymbol{T}$ – transition matrix, 27

$[p]$ – set of pairs equivalent to p, 31

$\mathcal{F}(i,n)$ – set of all pairs of sequences of length n with less of length i, 45

$F(i,n)$ – number of pairs in $\mathcal{F}(i,n)$, 45

Δ – set of all matches, 48

$\lfloor p \rfloor$ – pair with top and bottom sequence swapped, 48

$\mathcal{C}(m)$ – set of all collations generating a pair of total length m, 49

$\mathcal{N}(i)$ – set of all nondominated collations of order i, 49

$\mathcal{H}(i)$ – set of accepted collations of order i, 56

$\mathcal{U}(u)$ – set of potentially dominated collations, 62

$\mathcal{U}_t(u,i), \mathcal{U}_b(u,i)$ – sets of potentially dominated collations, 67

Ψ^n – set $(\Sigma^n)^l$ of l-tuples of sequences of length n, 79

Ψ^* – set $(\Sigma^*)^l$ of l-tuples of sequences, 79

$\gamma_k^{(l)}$ – γ_k for l sequences, 80

$\mathbf{S}(u,v)$ – length of a shortest common supersequence of u and v, 84

$\sigma_k^{(l)}$ – proportional expected length of a shortest common supersequence of l sequences, 85

$A(u)$ – adaptability of a sequence u, 93

$\mathbf{A}_n$ – minimal adaptability, 93

$\mathbf{B}_n$ – maximal adaptability, 93

α_k – proportional minimal adaptability, 94

β_k – proportional maximal adaptability, 95

$\mathbf{W}(u, v)$ – longest common substring of u and v, 101

$\mathbf{M}^m(u, v)$ – the length of a longest common substring of u and v with m mismatches, 102

$\mathbf{N}^r(u, v)$ – the length of a longest common substring of u and v with proportion of mismatches $\leq r$, 102

$\mathbf{H}(u, v; r, s)$ – heaviest common substring of u and v, 103

Index

Bibliography

[ABG92] A. Apostolico, S. Browne, and C. Guerra. Fast linear-space computations of longest common subsequences. *Theoretical Computer Science*, 92:3–17, 1992.

[AG87] A. Apostolico and C. Guerra. The longest common subsequence problem revisited. *Algorithmica*, 2:315–336, 1987.

[AGW86] Richard Arratia, Louis Gordon, and Michael S. Waterman. An extremal value theory for sequence matching. *The Annals of Statistics*, 14(3):971–993, 1986.

[Ale92] Kenneth. S. Alexander. The rate of convergence of the mean length of the longest common subsequences. Unpublished manuscript, August 1992.

[Ale94] Kenneth S. Alexander. Shortest common superstrings of random strings. To appear CPM'94.

[And86] Gabriela Andrejková. Systolic systems for the longest common subsequence problem. *Computers and Artificial Intelligence*, 5(3):199–212, 1986.

[Apo86] Alberto Apostolico. Improving the worst-case performance of the Hunt-Szymanski strategy for the longest common subsequence of two strings. *Information Processing Letters*, 23:63–69, 1986.

[Apo87] Alberto Apostolico. Remark on the Hsu-Du new algorithm for the longest common subsequence problem. *Information Processing Letters*, 25:235–236, 1987.

[AW85] Richard Arratia and Michael S. Waterman. An Erdős-Rényi law with shifts. *Advances in Mathematics*, 55:13–23, 1985.

[AW89] Richard Arratia and Michael S. Waterman. The Erdős-Rényi strong law for pattern matching with a given proportion of mismatches. *The Annals of Probability*, 17(3):1152–1168, 1989.

[BJ91] James H. Bradford and T. A. Jenkyns. On the inadequacy of tournament algorithms for the *n*-scs problem. *Information Processing Letters*, 38(4):169–171, 1991.

[BJL+91] Avrim Blum, Tao Jiang, Ming Li, John Tromp, and Mihalis Yannakakis. Linear approximation of shortest superstrings. In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 328–336, New York, 1991. Assoc. Computing Machinery.

[BY91] Ricardo A. Baeza-Yates. Searching subsequences. *Theoretical Computer Science*, 78:363–376, 1991.

[CL92] William I. Chang and Jordan Lampe. Theoretical and empirical comparisons of approximate string matching algorithms. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Combinatorial Pattern Matching. Proceedings*, pages 175–184. Lecture Notes in Computer Science 644, Springer-Verlag, 1992.

[CP90] Francis Y. L. Chin and C. K. Poon. A fast algorithm for computing longest common subsequences of small alphabet size. *Journal of*

Information Processing, 13(4):463–469, 1990.

[CS75] Václav Chvátal and David Sankoff. Longest common subsequence of two random sequences. *Journal of Applied Probability*, 12:306–315, 1975.

[CS83] Václav Chvátal and David Sankoff. An upper-bound technique for lengths of common subsequences. In D. Sankoff and J. B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: The theory and practice of sequence comparison*, chapter 16, pages 359–362. Addison-Wesley, Reading, Mass, 1983.

[Dek79] Joseph G. Deken. Some limit results for longest common subsequences. *Discrete Mathematics*, 26:17–31, 1979.

[Dek83] Joseph G. Deken. Probabilistic behavior of longest-common-subsequence length. In D. Sankoff and J. B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: The theory and practice of sequence comparison*, chapter 16, pages 359–362. Addison-Wesley, Reading, Mass, 1983.

[DP94] Vlado Dančík and Mike Paterson. Upper bounds for the expected length of a longest common subsequence of two binary sequences. In P. Enjalbert, E. W. Mayr, and K.W.Wagner, editors, *11th Annual Symposium on Theoretical Aspects of Computer Science, Proceedings*, pages 669–678. Lecture Notes in Computer Science 775, Springer-Verlag, 1994.

[EGGI92] David Eppstein, Zvi Galil, Raffaele Giancarlo, and Giuseppe F. Italiano. Sparse dynamic programming I: Linear cost functions. *Journal of the Association for Computing Machinery*, 39(3):519–545, 1992.

[ER70] Paul Erdös and Alfred Rényi. On a new law of large numbers. *J. Analyse Math.*, 22:103–111, 1970.

[Fel68] William Feller. *An Introduction to Probability Theory and its Applications*, volume I. John Wiley & Sons, New York, third edition, 1968.

[FLY92] David E. Foulser, Ming Li, and Qiang Yang. Theory and algorithms for plan merging. *Artificial Intelligence*, 57:143–181, 1992.

[GMS80] John Gallant, David Maier, and James Storer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, 20:50–58, 1980.

[HD80] Patrick A. V. Hall and Geoff R. Dowling. Approximate string matching. *Computing Surveys*, 12(4):381–402, 1980.

[HD84] W. J. Hsu and M. W. Du. New algorithms for the LCS problem. *Journal of Computer and System Sciences*, 19:133–152, 1984.

[Heb91] Jean-Jacques Hebrard. An algorithm for distinguishing efficiently bit-strings by their subsequences. *Theoretical Computer Science*, 82:35–49, 1991.

[HI92] Koji Hakata and Hiroshi Imai. The longest common subsequence problem for small alphabet size between many strings. In T. Ibaraki, Y. Inagaki, K. Iwama, T. Nishizeki, and M. Yamashita, editors, *Algorithms and Computation, Proceedings*, pages 469–478. Lecture Notes in Computer Science 650, Springer-Verlag, 1992.

[Hir75] Daniel S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the Association for Computing Machinery*, 18(6):341–343, 1975.

[Hir77] Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the Association for Computing Machinery*, 24(4):664–675, 1977.

[Hir78] Daniel S. Hirschberg. An information-theoretic lower bound for the longest common subsequence problem. *Information Processing Letters*, 7(1):40–41, 1978.

[HS77] James W. Hunt and Thomas G. Szymanski. A fast algorithm for computing longest common subsequences. *Communications of the Association for Computing Machinery*, 20(5):350–353, 1977.

[IF92] Robert W. Irving and Campbell B. Fraser. Two algorithms for the longest common subsequence of three (or more) strings. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Combinatorial Pattern Matching, Proceedings*, pages 214–229. Lecture Notes in Computer Science 644, Springer-Verlag, 1992.

[IF94] Robert W. Irving and Campbell B. Fraser. Maximal common subsequences and minimal common supersequences. To appear CPM'94.

[ITH92] O. H. Ibarra, J. A. Tao, and W. Hui. String editing on a one-way linear-array of finite-state machines. *IEEE Transactions on Computers*, 41(1):112–118, 1992.

[JL94] Tao Jiang and Ming Li. On the approximation of shortest common supersequences and longest common subsequences. To appear ICALP'94.

[KR87] S. Kiran Kumar and C. Pandu Rangan. A linear space algorithm for the LCS problem. *Acta Informatica*, 24:353–362, 1987.

[KS60] John G. Kemeny and J. Laurie Snell. *Finite Markov Chains.* D. Van Nostrand, Princeton, New Jersey, 1960.

[Liu85] Chung L. Liu. *Elements of Discrete Mathematics.* McGraw-Hill, New-York, 1985.

[LLF91] Hua Lin, Mi Lu, and Jesse Fang. An optimal algorithm for the longest common subsequence problem. In *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing*, pages 630–639, Los Alamitos, 1991. IEEE Comput. Soc. Press.

[LS77] B. F. Logan and L. A. Shepp. A variational problem for random Young tableaux. *Advances in Mathematics*, 26:206–222, 1977.

[Lu90] Mi Lu. Parallel computation of longest-common-subsequence. In S. G. Akl, F. Fiala, and W. W. Koczkodaj, editors, *Advances in Computing and Information - ICCI 90, Proceedings*, pages 383–394. Lecture Notes in Computer Science 468, Springer-Verlag, 1990.

[Mai78] David Maier. The complexity of some problems on subsequences and supersequences. *Journal of the Association for Computing Machinery*, 25(2):322–336, 1978.

[Mid93] Martin Middendorf. The shortest common nonsubsequence problem is np-complete. *Theoretical Computer Science*, 108:365–369, 1993.

[Mid94] Martin Middendorf. More on the complexity of common superstring and supersequence problems. *Theoretical Computer Science*, 125:205–228, 1994.

[MP80] William J. Masek and Michael S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, 1980.

[NKY82] Narao Nakatsu, Yahiko Kambayashi, and Shuzo Yajima. A longest common subsequence algorithm suitable for similar text strings. *Acta Informatica*, 18:171–179, 1982.

[PY88] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 229–234, New York, 1988. Assoc. Computing Machinery.

[Rio78] John Riordan. *An Introduction to Combinatorial Analysis*. Princeton University Press, Princeton, New Jersey, 1978.

[RU81] Kari-Jouko Räihä and Esko Ukkonen. The shortest common supersequence problem over binary alphabet is np-complete. *Theoretical Computer Science*, 16:187–198, 1981.

[Sim89] Imre Simon. Sequence comparison: Some theory and some practice. In M. Gross and D. Perrin, editors, *Electronic Dictionaries and Automata in Computational Linguistics, Proceedings*, pages 79–92. Lecture Notes in Computer Science 377, Springer-Verlag, 1989.

[SK83] D. Sankoff and J. B. Kruskal. *Time Warps, String Edits, and Macromolecules: The theory and practice of sequence comparison*. Addison-Wesley, Reading, Mass, 1983.

[Ste82] J. Michael Steele. Long common subsequences and the proximity of two random strings. *SIAM Journal on Applied Mathematics*, 14(4):731–737, 1982.

[Ste86] J. Michael Steele. An Efron-Stein inequality for nonsymmetric statistics. *The Annals of Statistics*, 14(2):753–758, 1986.

[Ste92] Graham A. Stephen. String search. Techinal report tr-92-gas-01, School of Electronic Engineering Science, University College of North Wales, Bangor, October 1992.

[Sto77] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1977.

[Sto88] James A. Storer. *Data Compression: Methods and Theory*. Computer Science Press, Rockville, Maryland, 1988.

[Tim89] В. Г. Тимковский. Сложность поиска общих подпоследовательностей, надпоследовательностей и сходных задач. *Кибернетика*, 25(5):1–13, 1989. (English translation: V. G. Timkovskii. Complexity of Common Subsequence and Supersequence Problems and Related Problems. *Cybernetics*, 25:565-580, 1990).

[TU88] Jorma Tarhio and Esko Ukkonen. A greedy approximation algorithm for constructing shortest common superstrings. *Theoretical Computer Science*, 57:131–145, 1988.

[Tur89] Jonathan S. Turner. Approximation algorithms for the shortest common superstring problem. *Information and Computation*, 83:1–20, 1989.

[Ukk85] Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64:100–118, 1985.

[VK77] A. M. Vershik and S. V. Kerov. Asymptotics of the plancherel measure of the symmetric group and the limiting form of Young tables. *Soviet Math. Doklady*, 18:527–531, 1977.

[WC76] C. K. Wong and A. K. Chandra. Bounds for the string editing problem. *Journal of the Association for Computing Machinery*, 23(1):13–

16, 1976.

[WF74] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

[WGA87] Michael S. Waterman, Louis Gordon, and Richard Arratia. Phase-transitions in sequence matches and nucleic acid structure. *Proceedings of the National Academy of Sciences of the USA*, 84:1239–1243, 1987.

[WMMM90] Sun Wu, Udi Manber, Gene Myers, and Webb Miller. An $O(NP)$ sequence comparison algorithm. *Information Processing Letters*, 35:317–323, 1990.