Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 370 (2007) 19–33

Theoretical Computer Science

www.elsevier.com/locate/tcs

An efficient alignment algorithm for masked sequences[☆]

Jin Wook Kim, Kunsoo Park[*]

School of Computer Science and Engineering, Seoul National University, Seoul, 151-742, Republic of Korea

Received 31 January 2005; received in revised form 23 November 2005; accepted 9 October 2006

Communicated by M. Crochemore

Abstract

We consider the alignment problem where sequences may have masked regions. The bases in masked regions are either unspecified or unknown, and they will be denoted by N. We present an efficient algorithm that finds an optimal local alignment by skipping such masked regions of sequences. Our algorithm works for both the affine gap penalty model and the linear gap penalty model. The time complexity of our algorithm is $O((n-T)(m-S)+vm+wn)$ time, where n and m are the lengths of given sequences A and B, T and S are the numbers of base N in A and B, and v and w are the numbers of masked regions in A and B, respectively.

Keywords: Local alignment; Global alignment; Affine gap penalty; Masked sequence

1. Introduction

A local alignment algorithm finds substrings α and β of sequences A and B, respectively, where α is similar to β [9]. Local alignment algorithms are used in many applications such as DNA sequencing programs. For when the lengths of A and B are n and m, respectively, Smith and Waterman [16] gave a well-known $O(n^2m)$ time local alignment algorithm for the affine gap penalty model and Gotoh [7] improved it to $O(nm)$ time. Crochemore et al. [3] gave a subquadratic time algorithm but it considers only a linear gap penalty model.

In DNA sequencing [1,8,10,11,17], some bases of fragments lose their information for various reasons. RepeatMasker [15] screens DNA sequences for interspersed repeats and low complexity DNA sequences. Low quality bases of DNA sequences are also screened by vector screening [1,8,11]. Some bases of fragments may not be determined by a base caller [4]. These *unspecified* or *unknown* bases will be denoted by N in this paper. Since masked regions lose their base information, alignments with masked regions are meaningless.

In this paper we present an efficient algorithm that finds an optimal local alignment by skipping such masked regions of sequences. In spite of skipping masked regions, the local alignment that our algorithm finds is the same as the one that the Smith–Waterman–Gotoh algorithm finds by computing all entries. To skip masked regions, we

☆ This work was supported by the MOST grant FPR05A2-341.

* Corresponding author. Tel.: +82 2 880 8381; fax: +82 2 885 3141.
E-mail address: kpark@theory.snu.ac.kr (K. Park).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.10.003

first give a new recurrence for *insignificant entries* in a dynamic programming table, and then develop an efficient algorithm based on the new recurrence. Our algorithm runs in $O((n-T)(m-S)+vm+wn)$ time, where T and S are the numbers of base N in A and B, and v and w are the numbers of masked regions in A and B, respectively. We also present an $O((n-T)(m-S)+vm+wn)$ time global alignment algorithm.

The remainder of the paper is organized as follows. In Section 2, we describe the local alignment algorithm due to Smith, Waterman, and Gotoh, and we extend it to include base N. In Section 3, we present a new local alignment algorithm for the affine gap penalty model. In Section 4, we present a new global alignment algorithm and we conclude in Section 5.

2. Preliminaries

We first give some definitions and notation that will be used in our algorithm. A string or a sequence is concatenations of zero or more characters from an alphabet Σ. A space is denoted by $\Delta \notin \Sigma$; we regard Δ as a character for convenience. The length of a string A is denoted by $|A|$. Let a_i denote the ith character of a string A and $A[i..j]$ denote a substring $a_i a_{i+1} \ldots a_j$ of A. When a sequence α is a substring of a sequence A, we denote it by $\alpha \prec A$. Given two strings $A = a_1 a_2 \ldots a_n$ and $B = b_1 b_2 \ldots b_m$, an alignment of A and B is $A^* = a_1^* a_2^* \ldots a_l^*$ and $B^* = b_1^* b_2^* \ldots b_l^*$ constructed by inserting zero or more Δs into A and B so that each a_i^* maps to b_i^* for $1 \le i \le l$. There are three kinds of mappings in a^* and b^* according to the characters of a_i^* and b_i^*:

- match : $a_i^* = b_i^* \ne \Delta$,
- mismatch : $(a_i^* \ne b_i^*)$ and $(a_i^*, b_i^* \ne \Delta)$,
- insertion or deletion (indel for short) : either a_i^* or b_i^* is Δ.

Note that we do not allow the case of $a_i^* = b_i^* = \Delta$.

2.1. Local alignments

Given two sequences A and B, a local alignment of A and B is an alignment of two strings α and β such $\alpha \prec A$ and $\beta \prec B$, and an optimal local alignment of A and B is a local alignment of A and B that has the highest similarity among all local alignments of A and B. We denote the similarity of an optimal local alignment by $SL(A, B)$.

A well-known algorithm for finding an optimal local alignment was given by Smith and Waterman [16], and Gotoh [7], and will be called the Smith–Waterman–Gotoh algorithm (SWG algorithm for short) [16,18,7]. Given two sequences A and B where $|A| = n$ and $|B| = m$, the SWG algorithm computes $SL(A, B)$ using a dynamic programming table (called the *H table*) of size $(n+1)(m+1)$. Let H_{ij} for $0 \le i \le n$ and $0 \le j \le m$ denote $SL(A[1..i], B[1..j])$. Then, H_{ij} can be computed by the following recurrence:

$$\begin{aligned} &H_{i,0} = H_{0,j} = 0 \quad \text{for } 0 \le i \le n, 0 \le j \le m \\ &H_{ij} = \max\left\{H_{i-1,j-1} + s(a_i, b_j), C_{ij}, R_{ij}, 0\right\} \quad \text{for } 1 \le i \le n,\ 1 \le j \le m \end{aligned} \tag{1}$$

where $s(a_i, b_j)$ is the similarity score of elements a_i and b_j such that

$$s(a_i, b_j) = \begin{cases} 1 & \text{if } a_i = b_j \\ -\delta & \text{if } a_i \ne b_j \end{cases}$$

where δ is a non-negative constant, and C_{ij} (or R_{ij}) is the highest similarity among local alignments of $A[1..i]$ and $B[1..j]$ such that the last mapping is insertion (or deletion). C_{ij} and R_{ij} are computed by the following recurrence:

$$\begin{aligned} &C_{0,j} = R_{i,0} = -\infty \quad \text{for } 0 \le i \le n,\ 0 \le j \le m \\ &C_{ij} = \max\left\{H_{i-1,j} - g_1, C_{i-1,j} - \mu\right\} \quad \text{for } 1 \le i \le n,\ 1 \le j \le m \\ &R_{ij} = \max\left\{H_{i,j-1} - g_1, R_{i,j-1} - \mu\right\} \quad \text{for } 1 \le i \le n,\ 1 \le j \le m, \end{aligned} \tag{2}$$

where g_k is the gap penalty for an indel of $k \ge 1$ bases. Among all H_{ij} for $0 \le i \le n$ and $0 \le j \le m$, the highest value is $SL(A, B)$, and the SWG algorithm takes $O(nm)$ time to compute it [7].

The gap penalty g_k for an indel of $k \ge 1$ bases is defined as $g_k = \gamma + k\mu$ where γ and μ are non-negative constants. This is called the *affine gap penalty model*, where γ is the gap initiation penalty and μ is the gap extension penalty. When there is no gap initiation penalty, i.e., $g_k = k\mu$, we call it the *linear gap penalty model*.

Table 1
Long sequences

Organism		**A.** Length of seq.	**B.** No. of base N (**B**/**A**)	**C.** No. of masked regions (avg. length of region: **B**/**C**)
Homo sapiens	*cDNA*	2,027	702 (34.6%)	5 (140.4bp)
	elastase 2	6,672	2,451 (36.7%)	15 (163.4bp)
	chrom. Y	813,231	376,139 (46.3%)	850 (442.5bp)
Apis mellifera	*chrom. MT*	16,343	1,660 (10.2%)	16 (103.8bp)
	chrom. LG15	1,349,409	87,706 (6.5%)	1443 (60.8bp)
Oryza sativa	*chrom. 3*	736,097	16,010 (2.2%)	313 (51.2bp)
	chrom. 12	2,853,905	69,928 (2.5%)	1326 (52.7bp)

Sequences are parts of an organism.

Table 2
Fragments sets

Organism	No. of frag.	% Frag. contain. N	**A.** Avg. length of seq.	**B.** Avg. no. of base N (**B**/**A**)	**C.** Avg. no. of regions (**B**/**C**)
Bison bison	10,310	54.7%	1029.8	262.0 (25.4%)	1.6 (163.8bp)
Alligator mississippiensis	39,004	88.1%	1015.3	96.6 (9.1%)	4.6 (21.0bp)
Mus musculus	277,490	79.5%	809.1	235.9 (29.2%)	3.6 (65.5bp)

A, **B** and **C** are computed over the fragments containing N.

2.2. Global alignments

Given two sequences A and B, an optimal global alignment of A and B is an alignment of A and B that has the highest similarity. We denote the similarity of an optimal global alignment by $SG(A, B)$.

$SG(A, B)$ also can be computed using the H table. The initial conditions are $H_{i,0} = -g_i$ for $0 \leq i \leq n$ and $H_{0,j} = -g_j$ for $0 \leq j \leq m$, and the recurrence for H_{ij} is $H_{ij} = \max\{H_{i-1,j-1} + s(a_i, b_j), C_{ij}, R_{ij}\}$ for $1 \leq i \leq n, 1 \leq j \leq m$. Then the value H_{nm} is $SG(A, B)$ and it is computed in $O(nm)$ time.

2.3. Scoring matrix

In case of DNA sequences, there are four bases, A, C, G, and T. In addition, IUB ambiguity codes [12] show several other bases which are for incompletely specified bases. In practice, however, only bases N and X are used. N means 'unspecified' and X means 'unknown'. These bases are used when the original bases are screened by repeat masking or vector screening, or when the original bases are not determined by a base caller. In this paper we will use only N for an unspecified or unknown base.

Tables 1 and 2 show the distributions of base N in DNA sequences. We obtained sequences randomly from NCBI [13,14] and they are screened out using RepeatMasker [15]. The ratio of the number of base N to the length of a sequence is various: from 2.2% up to 46.3%. The average length of each masked region is also various: from 21.0bp up to 442.5bp. In general, the ratio of base N to a sequence is significant, and the number of masked regions is quite small.

We extend the definition of similarity score $s(a_i, b_j)$. Because N might be any base, it is impossible to determine whether an alignment with N is a match or a mismatch. Hence, similarity score $s(a_i, b_j)$ is newly defined as follows:

$$s(a_i, b_j) = \begin{cases} 1 & \text{if } a_i = b_j \neq \text{N} \\ -\delta & \text{if } a_i \neq b_j \text{ and } a_i, b_j \neq \text{N} \\ \sigma & \text{if either } a_i \text{ or } b_i \text{ is N} \end{cases}$$

	A	C	G	T	N
A	1	–2	–2	–2	0
C	–2	1	–2	–2	0
G	–2	–2	1	–2	0
T	–2	–2	–2	1	0
N	0	0	0	0	0

Fig. 1. Scoring matrix for a DNA sequence where $\delta = 2$ and $\sigma = 0$.

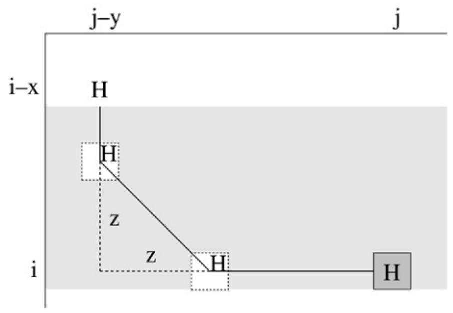

Fig. 2. Proof of Lemma 1.

where σ is a constant (which may not be 0) [6,8]. Fig. 1 shows an example of the similarity score (also known as the scoring matrix) for DNA sequence alignments where $\delta = 2$ and $\sigma = 0$. We call an entry H_{ij} where $a_i = \text{N}$ or $b_j = \text{N}$ an *insignificant entry*.

3. Local alignment algorithm

In this section we present an algorithm that finds an optimal local alignment without computing insignificant entries. We first give a new recurrence for insignificant entries, and then present a new local alignment algorithm for the affine gap penalty model that is based on the new recurrence. We also describe a simpler algorithm for the linear gap penalty model.

3.1. New recurrence for an insignificant entry

Now we present a new recurrence for an insignificant entry H_{ij}. It can be defined as the maximum of all the paths from all its previously defined entries to H_{ij}. Our goal is to select the smallest possible number of entries that must be considered to compute H_{ij} and define a new recurrence using only these entries.

We begin with some lemmas that are needed to get the new recurrence. Lemma 1 provides a relationship between an insignificant entry H_{ij} and its previously defined entries and Lemmas 2–9 provide the range of entries that must be considered in the new recurrence based on Lemma 1.

Let A and B be two DNA sequences where $|A| = n$ and $|B| = m$. We define $C_{0j} = R_{i0} = -\infty$ for $1 \le i \le n$ and $1 \le j \le m$ and define $g_0 = 0$. Suppose that $A[i-t+1..i]$ is a masked region of length t, i.e., $A[i-t+1..i] = \text{NN}\ldots\text{N}$.

Lemma 1. *$H_{ij} \ge H_{i-x,j-y} - g_{y-z} + z\sigma - g_{x-z}$ for some constants x, y and z such that $0 \le x \le t$, $0 \le y \le j$, and $0 \le z \le \min\{x, y\}$.*

Proof. From recurrence (1), H_{ij} is larger than or equal to C_{ij}, and C_{ij} is larger than or equal to $H_{i-x,j} - g_x$ for $0 \le x \le t$. Thus $H_{ij} \ge H_{i-x,j} - g_x$. Similarly, we can obtain that $H_{ij} \ge H_{i,j-y} - g_y$ for $0 \le y \le j$. On the other hand, from recurrence (1), H_{ij} is larger than or equal to $H_{i-1,j-1} + s(a_i, b_j)$. Because a_i is N, it is rewritten that $H_{ij} \ge H_{i-1,j-1} + \sigma$ and it can be easily expanded as $H_{ij} \ge H_{i-z,j-z} + z\sigma$ for $0 \le z \le \min\{j, t\}$. From these three inequalities, we can deduce that $H_{ij} \ge H_{i-x,j-y} - g_{y-z} + z\sigma - g_{x-z}$. See Fig. 2. □

The above lemma provides a relation between H_{ij} and its previously defined entries. Note that Lemma 1 is true when $x - z = 0$ or $y - z = 0$. If equality holds in Lemma 1, we say that H_{ij} *comes from* $H_{i-x,j-y}$. We will use 'come from' for some more cases.

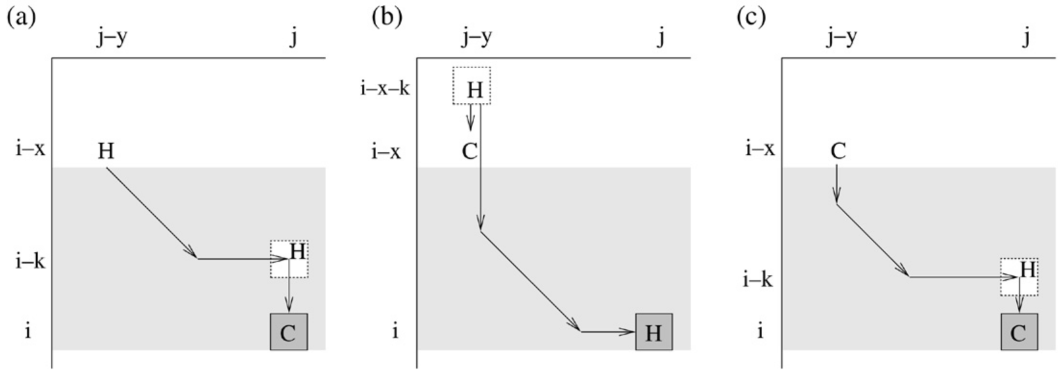

Fig. 3. Three cases of 'come from'.

- We say that C_{ij} comes from $H_{i-x,j-y}$ if there exists $H_{i-k,j}$ for $1 \le k \le x$ such that $C_{ij} = H_{i-k,j} - g_k$ and $H_{i-k,j}$ comes from $H_{i-x,j-y}$. See Fig. 3(a).
- We say that H_{ij} comes from $C_{i-x,j-y}$ if there exists $H_{i-x-k,j-y}$ for $k \ge 1$ such that $C_{i-x,j-y} = H_{i-x-k,j-y} - g_k$ and H_{ij} comes from $H_{i-x-k,j-y}$ via entry $(i-x, j-y)$, i.e., the path from $H_{i-x-k,j-y}$ to H_{ij} passes entry $(i-x, j-y)$. See Fig. 3(b). Unlike the case where C_{ij} comes from $H_{i-x,j-y}$, the intermediate entry $H_{i-x-k,j-y}$ of the case where H_{ij} comes from $C_{i-x,j-y}$ does not lie between $C_{i-x,j-y}$ and H_{ij}.
- We say that C_{ij} comes from $C_{i-x,j-y}$ if there exists $H_{i-k,j}$ for $1 \le k \le x$ such that $C_{ij} = H_{i-k,j} - g_k$ and $H_{i-k,j}$ comes from $C_{i-x,j-y}$. See Fig. 3(c).

The following lemmas provide the range of entries of row $i-t$ which H_{ij} can come from.

Lemma 2. *If H_{ij} comes from $H_{i-t,j-u}$ for $u \ge 0$, there exists at least one $H_{i-t,j-s}$ which H_{ij} comes from for some constant $0 \le s \le t$.*

Proof. If $u \le t$, we are done. We give a proof for $u > t$. See Fig. 4(a). Because H_{ij} comes from $H_{i-t,j-u}$, $H_{ij} = H_{i-t,j-u} - g_{u-s} + s\sigma - g_{t-s}$ for some constant $0 \le s \le t$ by the equality case of Lemma 1. Since $H_{i-t,j-u+(u-s)} \ge H_{i-t,j-u} - g_{u-s}$ by Lemma 1, $H_{ij} \le H_{i-t,j-s} + s\sigma - g_{t-s}$. On the other hand, $H_{ij} \ge H_{i-t,j-s} + s\sigma - g_{t-s}$ by Lemma 1. Hence, $H_{ij} = H_{i-t,j-s} + s\sigma - g_{t-s}$, which means that H_{ij} comes from $H_{i-t,j-s}$. $\square$

Lemma 3. *If H_{ij} comes from $C_{i-t,j-u}$ for $u \ge 0$, there exists at least one $C_{i-t,j-s}$ which H_{ij} comes from for some constant $0 \le s \le t-1$.*

Proof. If $u < t$, we are done. We give a proof for $u \ge t$. See Fig. 4(b). Because H_{ij} comes from $C_{i-t,j-u}$, there exists $H_{i-t-k,j-u}$ for $k \ge 1$ such that $C_{i-t,j-u} = H_{i-t-k,j-u} - g_k$ and H_{ij} comes from $H_{i-t-k,j-u}$ via entry $(i-t, j-u)$. Let s be the number of σs in the path from $H_{i-t-k,j-u}$ to H_{ij}. Note that $0 \le s \le t-1$. First we will show that H_{ij} comes from $H_{i-t-k,j-s}$ and then show that H_{ij} comes from $C_{i-t,j-s}$.

Since H_{ij} comes from $H_{i-t-k,j-u}$ and there are s σs in the path, $H_{ij} = H_{i-t-k,j-u} - g_{k+t-s} + s\sigma - g_{u-s}$. Since $H_{i-t-k,j-s} \ge H_{i-t-k,j-u} - g_{u-s}$ by Lemma 1, $H_{ij} \le H_{i-t-k,j-s} - g_{k+t-s} + s\sigma$. On the other hand, $H_{ij} \ge H_{i-t-k,j-s} - g_{k+t-s} + s\sigma$ by Lemma 1. Hence, $H_{ij} = H_{i-t-k,j-s} - g_{k+t-s} + s\sigma$. That is, H_{ij} comes from $H_{i-t-k,j-s}$.

Now we will show that H_{ij} comes from $C_{i-t,j-s}$. To do this, we need to show that $C_{i-t,j-s} = H_{i-t-k,j-s} - g_k$ and that H_{ij} comes from $H_{i-t-k,j-s}$ via entry $(i-t, j-s)$. By definition, $C_{i-t,j-s} \ge H_{i-t-k,j-s} - g_k$. Suppose that $C_{i-t,j-s} > H_{i-t-k,j-s} - g_k$. Then, there exists $H_{i-t-q,j-s}$ for $q \ne k$ such that $C_{i-t,j-s} = H_{i-t-q,j-s} - g_q$. By Lemma 1, $H_{ij} \ge H_{i-t-q,j-s} - g_{q+t-s} + s\sigma$. Because $H_{ij} = H_{i-t-k,j-s} - g_{k+t-s} + s\sigma$, $H_{i-t-k,j-s} - g_{k+t-s} \ge H_{i-t-q,j-s} - g_{q+t-s}$. Since $g_{k+t-s} = \gamma + (k+t-s)\mu = g_k + (t-s)\mu$ and $g_{q+t-s} = g_q + (t-s)\mu$, we get

$$H_{i-t-k,j-s} - g_k \ge H_{i-t-q,j-s} - g_q. \tag{3}$$

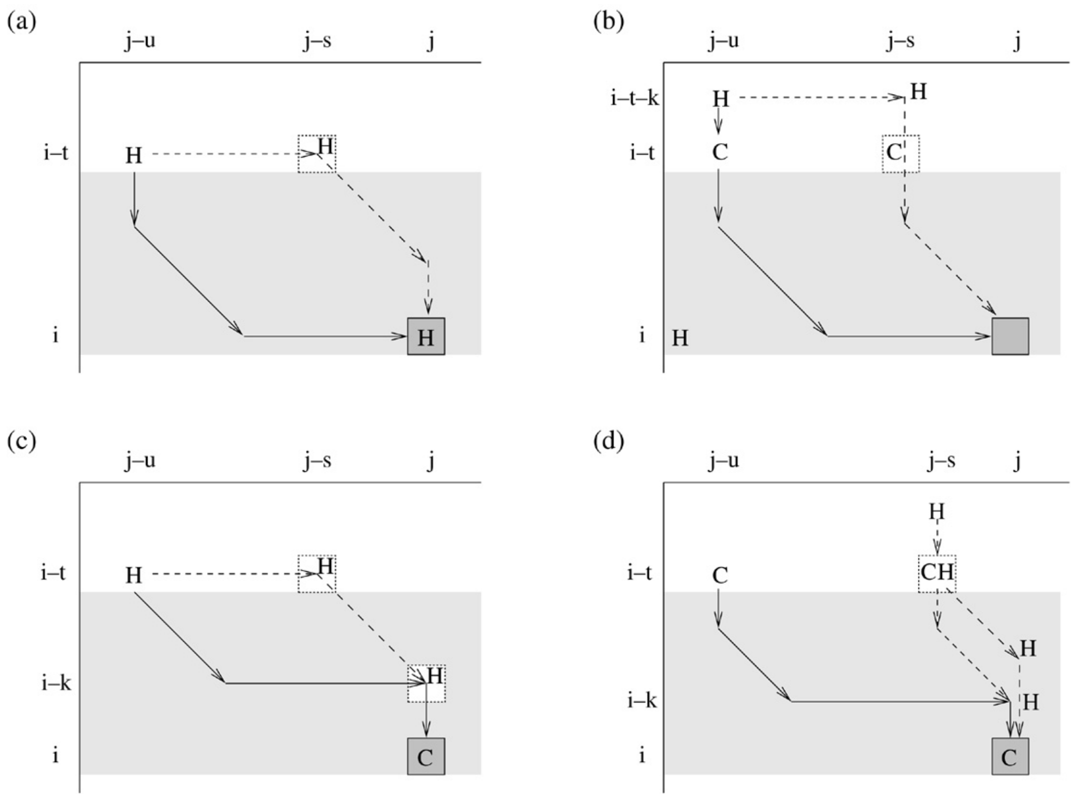

Fig. 4. Proofs of Lemmas 2–5.

By our hypothesis $C_{i-t,j-s} > H_{i-t-k,j-s} - g_k$ and the right hand side of (3) is $C_{i-t,j-s}$. Therefore $C_{i-t,j-s} > C_{i-t,j-s}$, a contradiction. Hence, $C_{i-t,j-s} = H_{i-t-k,j-s} - g_k$. Since H_{ij} comes from $H_{i-t-k,j-s}$ via entry $(i-t, j-s)$, H_{ij} comes from $C_{i-t,j-s}$. $\square$

Note that s in Lemma 3 is the number of σs in the path from $C_{i-t,j-u}$ to H_{ij}. The following lemmas provide the range of entries of row $i-t$ which C_{ij} can come from.

Lemma 4. *If C_{ij} comes from $H_{i-t,j-u}$ for $u \geq 0$, there exists at least one $H_{i-t,j-s}$ which C_{ij} comes from for some constant* $0 \leq s \leq t-1$.

Proof. See Fig. 4(c). Because C_{ij} comes from $H_{i-t,j-u}$, there exists $H_{i-k,j}$ for $1 \leq k \leq t$ such that $C_{ij} = H_{i-k,j} - g_k$ and $H_{i-k,j}$ comes from $H_{i-t,j-u}$. By Lemma 2, there exists at least one $H_{i-t,j-s}$ which $H_{i-k,j}$ comes from for some constant $0 \leq s \leq t-k \leq t-1$. Thus, C_{ij} comes from $H_{i-t,j-s}$ for $0 \leq s \leq t-1$. $\square$

Lemma 5. *If C_{ij} comes from $C_{i-t,j-u}$ for $u \geq 1$, there exists at least one $H_{i-t,j-s}$ which C_{ij} comes from for some constant* $0 \leq s \leq t-2$.

Proof. See Fig. 4(d). Because C_{ij} comes from $C_{i-t,j-u}$, there exists $H_{i-k,j}$ for $1 \leq k \leq t$ such that $C_{ij} = H_{i-k,j} - g_k$ and $H_{i-k,j}$ comes from $C_{i-t,j-u}$. Let s be the number of σs in the path from $C_{i-t,j-u}$ to $H_{i-k,j}$. Note that $0 \leq s \leq t-2$. First we will show that C_{ij} comes from $C_{i-t,j-s}$ and then show that C_{ij} comes from $H_{i-t,j-s}$.

Since $H_{i-k,j}$ comes from $C_{i-t,j-u}$ and there are s σs in the path from $C_{i-t,j-u}$ to $H_{i-k,j}$, $H_{i-k,j}$ comes from $C_{i-t,j-s}$ by Lemma 3. Since C_{ij} comes from $H_{i-k,j}$ (i.e., $C_{ij} = H_{i-k,j} - g_k$), C_{ij} comes from $C_{i-t,j-s}$.

Now we will show that C_{ij} comes from $H_{i-t,j-s}$. To do this, we need to show that C_{ij} comes from $H_{i-t+s,j}$ and that $H_{i-t+s,j}$ comes from $H_{i-t,j-s}$. Since $H_{i-k,j}$ comes from $C_{i-t,j-s}$, there exists $H_{i-t-p,j-s}$ for $p \geq 1$ such that

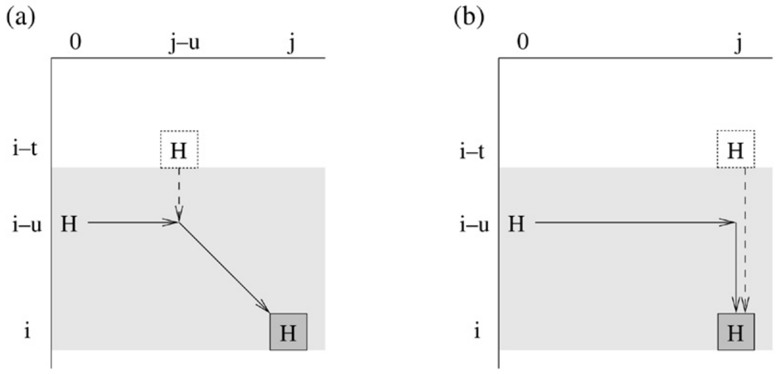

Fig. 5. Proof of Lemma 6.

$C_{i-t,j-s} = H_{i-t-p,j-s} - g_p$ and $H_{i-k,j}$ comes from $H_{i-t-p,j-s}$ via entry $(i-t, j-s)$. Since $C_{ij} = H_{i-k,j} - g_k$, we get

$$C_{ij} = H_{i-t-p,j-s} - g_{t+p-k-s} + s\sigma - g_k. \tag{4}$$

On the other hand, by definition, $C_{ij} \geq H_{i-t+s,j} - g_{t-s}$ and $H_{i-t,j-s} \geq C_{i-t,j-s} = H_{i-t-p,j-s} - g_p$. By Lemma 1, $H_{i-t+s,j} \geq H_{i-t,j-s} + s\sigma$. Thus,

$$C_{ij} \geq H_{i-t-p,j-s} - g_p + s\sigma - g_{t-s}. \tag{5}$$

Since $g_p + g_{t-s} = \gamma + p\mu + \gamma + (t-s)\mu = \gamma + (t+p-k-s)\mu + \gamma + k\mu = g_{t+p-k-s} + g_k$, the right hand side of (5) is equal to the right hand side of (4) and thus all inequalities that lead to (5) become equalities. Hence, $C_{ij} = H_{i-t+s,j} - g_{t-s}$ and $H_{i-t+s,j}$ comes from $H_{i-t,j-s}$. Therefore C_{ij} comes from $H_{i-t,j-s}$. □

The following lemmas handle the case where H_{ij} can come from the leftmost column.

Lemma 6. *If $t < j \leq m$ and H_{ij} comes from $H_{i-u,0}$ for $0 \leq u \leq t$, H_{ij} comes from $H_{i-t,j-s}$ for $0 \leq s \leq t$.*

Proof. We prove the lemma in two cases.

(i) $H_{ij} = H_{i-u,0} - g_{j-u} + u\sigma$: see Fig. 5(a). $H_{ij} \geq H_{i-t,j-u} - g_{t-u} + u\sigma$ by Lemma 1 and $H_{i-u,0} = 0$ from recurrence (1). Hence, $-g_{j-u} + u\sigma \geq H_{i-t,j-u} - g_{t-u} + u\sigma$, i.e., $0 \geq H_{i-t,j-u} + g_{j-u} - g_{t-u}$. However, $H_{i-t,j-u} \geq 0$ and $g_{j-u} - g_{t-u} \geq 0$, and thus the equality must hold. Therefore H_{ij} comes from $H_{i-t,j-u}$.

(ii) $H_{ij} = H_{i-u,0} - g_j - g_u$: see Fig. 5(b). $H_{ij} \geq H_{i-t,j} - g_t$ by Lemma 1 and $H_{i-u,0} = 0$ from recurrence (1). Hence, $-g_j - g_u \geq H_{i-t,j} - g_t$, i.e., $0 \geq H_{i-t,j} + g_j + g_u - g_t$. However, $H_{i-t,j} \geq 0$, $g_u \geq 0$, and $g_j - g_t \geq 0$, and thus the equality must hold. Therefore H_{ij} comes from $H_{i-t,j}$. □

Lemma 7. *If $0 < j \leq t$ and H_{ij} comes from $H_{i-u,0}$ for $0 \leq u \leq t$, H_{ij} comes from $H_{i-j,0}$.*

Proof. Similar to the proof of Lemma 6. □

The following lemmas handle the case where C_{ij} can come from the leftmost column. Since the proofs of Lemmas 8 and 9 are similar to that of Lemma 6, we omit them.

Lemma 8. *If $t \leq j \leq m$ and C_{ij} comes from $H_{i-u,0}$ for $1 \leq u \leq t$, C_{ij} comes from $H_{i-t,j-s}$ for $0 \leq s \leq t-1$.*

Lemma 9. *If $0 < j < t$ and C_{ij} comes from $H_{i-u,0}$ for $1 \leq u \leq t$, C_{ij} comes from $H_{i-j-1,0}$.*

By the lemmas above, we derive the following theorem which provides a new recurrence for an insignificant entry H_{ij}.

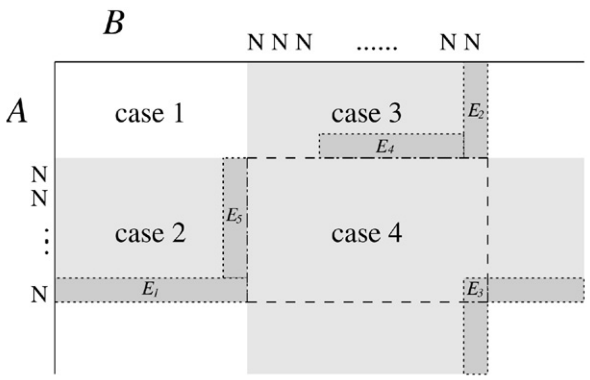

Fig. 6. An example of four cases in the H table. In Case 1, we compute all the entries. In Case 2, we compute only row E_1. We compute only column E_2 in Case 3. In Case 4, we compute an entry E_3 and compute row E_4 and column E_5.

Theorem 10. *Let $A[i-t+1..i]$ be a masked region of length t. Then H_{ij} can be computed by the following recurrence:*

$$C_{ij} = \begin{cases} \max\left\{ \begin{array}{l} \max\limits_{0\le s\le j-1}\{H_{i-t,j-s} + s\sigma - \gamma - (t-s)\mu\}, \\ C_{i-t,j} - t\mu,\ j\sigma - \gamma - \mu \end{array} \right\} & \textit{for } 1 \le j < t \\ \max\left\{ \begin{array}{l} \max\limits_{0\le s\le t-1}\{H_{i-t,j-s} + s\sigma - \gamma - (t-s)\mu\}, \\ C_{i-t,j} - t\mu \end{array} \right\} & \textit{for } t \le j \le m. \end{cases}$$

$$H_{ij} = \begin{cases} \max\left\{ \begin{array}{l} \max\limits_{1\le s\le j-1}\{C_{i-t,j-s} + s\sigma - (t-s)\mu\}, \\ j\sigma,\ C_{ij},\ 0 \end{array} \right\} & \textit{for } 1 \le j \le t \\ \max\left\{ \begin{array}{l} \max\limits_{1\le s\le t-1}\{C_{i-t,j-s} + s\sigma - (t-s)\mu\}, \\ H_{i-t,j-t} + t\sigma,\ C_{ij},\ 0 \end{array} \right\} & \textit{for } t < j \le m. \end{cases}$$

Proof. If $t \le j \le m$, C_{ij} comes from $H_{i-t,j-s}$ for $0 \le s \le t-1$ and $C_{i-t,j}$ by Lemmas 4, 5 and 8. We need to consider $C_{i-t,j}$ separately since Lemma 5 does not cover the case where C_{ij} comes from $C_{i-t,j}$. If $1 \le j < t$, C_{ij} comes from $H_{i-t,j-s}$ for $0 \le s \le j-1$, $C_{i-t,j}$, and $H_{i-j-1,0}$ by Lemmas 4, 5 and 9.

If $t < j \le m$, H_{ij} comes from $H_{i-t,j-s}$ for $0 \le s \le t$ and $C_{i-t,j-s}$ for $0 \le s \le t-1$ by Lemmas 2, 3 and 6. In the recurrence of H_{ij}, the elements from $H_{i-t,j-s}$ for $0 \le s \le t-1$ were replaced by C_{ij} because the recurrence of C_{ij} includes them. If $1 \le j \le t$, H_{ij} comes from $H_{i-t,j-s}$ for $0 \le s \le j-1$, $C_{i-t,j-s}$ for $0 \le s \le j-1$, and $H_{i-j,0}$ by Lemmas 2, 3 and 7. □

Suppose that $B[i-t+1..i]$ is a masked region of length t. The symmetric version of the above lemmas and Theorem 10 hold. In other words, they hold if we replace H_{ij} by H_{ji} and C_{ij} by R_{ji}.

3.2. New algorithm for the affine gap penalty model

A straightforward method for finding an optimal local alignment of masked sequences A and B where $|A| = n$ and $|B| = m$ is running the SWG algorithm in $O(nm)$ time. We want to find the local alignment without computing insignificant entries.

We present an algorithm for the affine gap penalty model. Our algorithm consists of the following four cases:

Case 1: Neither a_i nor b_j is N. Compute all these entries by the SWG algorithm.
Case 2: Only a_i is N. Compute the bottom row of a block of insignificant entries.
Case 3: Only b_j is N. Compute the rightmost column of a block of insignificant entries.
Case 4: Both a_i and b_j are N. Compute the bottom-right corner entry of a block of insignificant entries. In addition, compute a column to the left of the block and a row on top of the block. See Fig. 6.

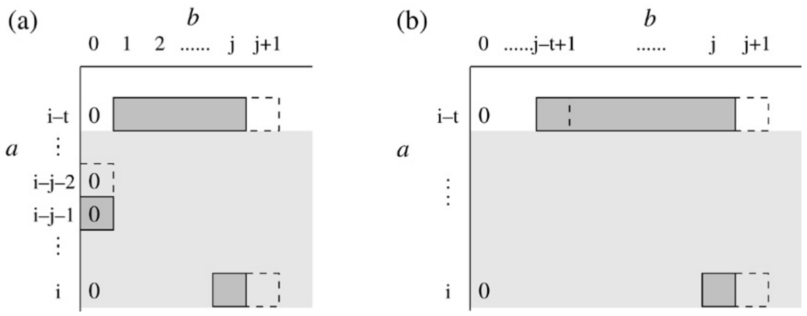

Fig. 7. (a) $1 \le j \le t$. To compute C_{ij}, we need $H_{i-j-1,0}$ and $H_{i-t,j-s}$ for $0 \le s \le j-1$ and to compute $C_{i,j+1}$, we need $H_{i-j-2,0} = 0$ and $H_{i-t,j+1-s}$ for $0 \le s \le j$. (b) $t < j \le m$. To compute C_{ij} and $C_{i,j+1}$, we need $H_{i-t,j-s}$ and $H_{i-t,j+1-s}$, respectively, for $0 \le s \le t-1$.

Fig. 6 shows an example of four cases in the H table. We explain our algorithm for Case 2 and for Case 4, since Case 3 is symmetric to Case 2.

3.2.1. Case 2

Let $A[i-t+1..i]$ be a masked region of length t, i.e., $A[i-t+1..i] = \text{NN}\ldots\text{N}$. We will show how to compute the bottom row $H_{i,j-q+1..j}$ of a $t \times q$ block in $O(q)$ time. To do this, we first find the relation between two adjacent insignificant entries. Then, we define a new data structure, MQUEUE, and give an algorithm of $O(q)$ time.

Computing the bottom row without computing other insignificant entries is possible by using the recurrences for an insignificant entry in Theorem 10. But it still takes $O(tq)$ time for a $t \times q$ block because it needs $O(t)$ computations for each C_{ij} and H_{ij}. However, we can reduce the time complexity to $O(q)$ using some properties of the recurrences.

Two adjacent entries of C are very similar. See Fig. 7. For $t < j \le m$, C_{ij} and $C_{i,j+1}$ are computed as

$$C_{ij} = \max\left\{\max_{0\le s\le t-1}\{H_{i-t,j-s} + s\sigma - \gamma - (t-s)\mu\}, C_{i-t,j} - t\mu\right\},$$

$$C_{i,j+1} = \max\left\{\max_{0\le s\le t-1}\{H_{i-t,j+1-s} + s\sigma - \gamma - (t-s)\mu\}, C_{i-t,j+1} - t\mu\right\}.$$

The changes from C_{ij} to $C_{i,j+1}$ are that (1) the element $H_{i-t,j-t+1} + (t-1)\sigma - \gamma - \mu$ is deleted, (2) value $\sigma + \mu$ is added to all elements except the last of C_{ij}, (3) a new element $H_{i-t,j+1} - \gamma - t\mu$ is inserted, (4) the element $C_{i-t,j} - t\mu$ is replaced by $C_{i-t,j+1} - t\mu$. Thus $C_{i,j+1}$ can be rewritten as

$$C_{i,j+1} = \max\left\{\begin{array}{l}\max\limits_{-1\le s\le t-2}\{H_{i-t,j-s} + s\sigma - \gamma - (t-s)\mu\},\\ C_{i-t,j+1} - \sigma - (t+1)\mu\end{array}\right\} + \sigma + \mu, \tag{6}$$

where $\max_{0\le s\le t-1}\{H_{i-t,j-s} + s\sigma - \gamma - (t-s)\mu\}$ also appears in C_{ij}. An insignificant entry $H_{i,j+1}$ can also be rewritten as follows:

$$H_{i,j+1} = \max\left\{\begin{array}{l}\max\limits_{0\le s\le t-2}\{C_{i-t,j-s} + s\sigma - (t-s)\mu\},\\ H_{i-t,j+1-t} + (t-1)\sigma - \mu, C_{i,j+1} - \sigma - \mu, -\sigma - \mu\end{array}\right\} + \sigma + \mu,$$

where $\max_{1\le s\le t-1}\{C_{i-t,j-s} + s\sigma - (t-s)\mu\}$ also appears in H_{ij}. Similarly, for $1 \le j \le t$, we can compute $C_{i,j+1}$ and $H_{i,j+1}$ using C_{ij} and H_{ij}, respectively.

We need a new data structure to find maximum values in amortized $O(1)$ time. Because the element $H_{i-t,j-t+1} + (t-1)\sigma - \gamma - \mu$ is deleted in recurrence (6), $\max_{0\le s\le t-1}\{H_{i-t,j-s} + s\sigma - \gamma - (t-s)\mu\}$ which is already computed in C_{ij} cannot be directly used in computing $C_{i,j+1}$. However, if we know the maximum value for the range of s such that $0 \le s \le t-2$, it can be used in $C_{i,j+1}$. We will find the maximum value using MQUEUE.

We define a specialized queue data structure, MQUEUE, that supports the following three operations:

`max()` : Return the maximum score among all elements in MQUEUE.

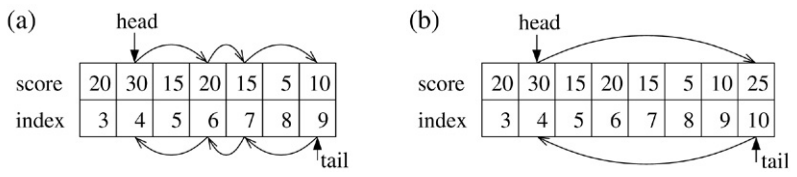

Fig. 8. An example of MQUEUE. (a) The list of maximum candidates is $30 \to 20 \to 15 \to 10$. (b) After `insert(25)`, the list is modified to $30 \to 25$.

`insert(`e`)` : Add an element e to MQUEUE.
`delete()` : Remove the oldest element in MQUEUE.

The `max()` operation is added to the traditional queue. Instead of MQUEUE, we can use a deque with heap order [5]. Each operation of it takes $O(1)$ worst case time. However, a simple $O(1)$ amortized time operation is sufficient for this case.

To obtain $O(1)$ amortized time for each operation, MQUEUE maintains a list of maximum candidates. Maximum candidates are defined as follows: Given a set of ordered elements, the first maximum candidate is an element that has the maximum score in the list. If the ith maximum candidate is defined, then the $(i+1)$st maximum candidate is an element that has the maximum score in the range from the next element of the ith maximum candidate to the last element of the list. Fig. 8(a) shows a list of maximum candidates.

The list of maximum candidates is maintained as a doubly linked list. A pointer *head* points to the maximum element of MQUEUE and *tail* points to the last one. The important property of the list is that the scores of the elements are in descending order.

Operation `insert(`e`)` needs to maintain the list property. After adding an element e, the score of e is compared with the maximum candidates from *tail* until it meets the score that is larger than the score of e or the list becomes empty. When it meets a larger one, they link each other and *tail* points to e. When the list becomes empty, we make a new list which has one element e and *head* and *tail* point to e. Fig. 8(b) shows an example of `insert(`e`)`.

Lemma 11. *Operation* `insert(`e`)` *takes amortized* $O(1)$ *time.*

Proof. If there are c comparisons in one `insert(`e`)`, the number of elements in the list of maximum candidates is reduced by $c-2$. Hence, each `insert` takes amortized $O(1)$ time. □

Operation `delete()` removes the oldest element from MQUEUE. If *head* points to the oldest element, then *head* is set to the next element in the list and we remove the oldest element. Operation `max()` returns the score of the element which is pointed to by *head*.

Fig. 9 shows the algorithm for Case 2 with $A[i-t+1..i]$ and $B[1..q]$ where $q > t$. The algorithm consists of two **for** loops. The first **for** loop computes H_{ij} for $1 \le j \le t$ and the second **for** loop computes H_{ij} for $t+1 \le j \le q$. We use two MQUEUEs, one for C_{ij} and the other for H_{ij}. In addition, we compute R_{ij} for $1 \le j \le q$ using recurrence (2) because R_{iq} can be used in Case 4 of the block which is to the right of the current block.

3.2.2. *Case 4*

We will show how to compute the bottom-right corner entry H_{ij} of a $t \times s$ block and how to compute the row on top of the block and the column to the left of the block. See Fig. 6. Let $A[i-t+1..i]$ and $B[j-s+1..j]$ be masked regions, i.e., $A[i-t+1..i] = \text{NN}\ldots\text{N}$ and $B[j-s+1..j] = \text{NN}\ldots\text{N}$.

Lemma 12. *If H_{ij} comes from $C_{i-t,j-u}$ for $1 \le u \le \min\{s,t\}-1$, H_{ij} comes from $C_{i-t,j}$ or $H_{i-t,j-l}$ for $0 \le l \le \min\{s,t\}-2$.*

Proof. Because H_{ij} comes from $C_{i-t,j-u}$, there exists $H_{i-t-k,j-u}$ for $k \ge 1$ such that $C_{i-t,j-u} = H_{i-t-k,j-u} - g_k$ and $H_{ij} = H_{i-t-k,j-u} + u\sigma - g_{k+t-u}$. If $k > u$, then $H_{i-t-k,j-u} + u\sigma \le H_{i-t-k+u,j}$ and $H_{i-t-k+u,j} - g_{k-u} \le C_{i-t,j}$, and thus $H_{ij} \le C_{i-t,j} - t\mu$. However, $H_{ij} \ge C_{i-t,j} - t\mu$ by definition, and hence H_{ij} comes from $C_{i-t,j}$. See Fig. 10(a). If $1 \le k \le u$, then $H_{i-t-k,j-u} + k\sigma \le H_{i-t,j-u+k}$ and thus $H_{ij} \le H_{i-t,j-u+k} + (u-k)\sigma - g_{t+k-u}$. Because $H_{ij} \ge H_{i-t,j-u+k} + (u-k)\sigma - g_{t+k-u}$ by Lemma 1, H_{ij} comes from $H_{i-t,j-u+k}$. Note that $0 \le u-k \le \min\{s,t\}-2$. See Fig. 10(b). □

```
MQUEUE $cmq$, $hmq$
$cut \leftarrow -\sigma - \mu$
$le \leftarrow \sigma + \mu$
for $j \leftarrow 1$ to $t$
    $cut \leftarrow cut + \sigma + \mu$
    $cmq$.insert($H_{i-t,j} - \gamma - t\mu - cut$),
    $hmq$.insert($C_{i-t,j} - t\mu - cut$)
    $le \leftarrow le - \mu$
    $C_{ij} \leftarrow \max(cmq.\mathtt{max}(), C_{i-t,j} - t\mu - cut) + cut$
    $H_{ij} \leftarrow \max(\max(le, hmq.\mathtt{max}(), -cut) + cut, C_{ij})$
    $R_{ij} \leftarrow \max(R_{i,j-1} - \mu, H_{i,j-1} - \gamma - \mu)$
end for
for $j \leftarrow t + 1$ to $q$
    $cut \leftarrow cut + \sigma + \mu$
    $cmq$.delete(),
    $cmq$.insert($H_{i-t,j} - \gamma - t\mu - cut$),
    $hmq$.delete()
    $hmq$.insert($C_{i-t,j} - t\mu - cut$)
    $C_{ij} \leftarrow \max(cmq.\mathtt{max}(), C_{i-t,j} - t\mu - cut) + cut$
    $H_{ij} \leftarrow \max(\max(hmq.\mathtt{max}(), -cut) + cut, H_{i-t,j-t} + t\sigma, C_{ij})$
    $R_{ij} \leftarrow \max(R_{i,j-1} - \mu, H_{i,j-1} - \gamma - \mu)$
end for
```

Fig. 9. Algorithm for Case 2.

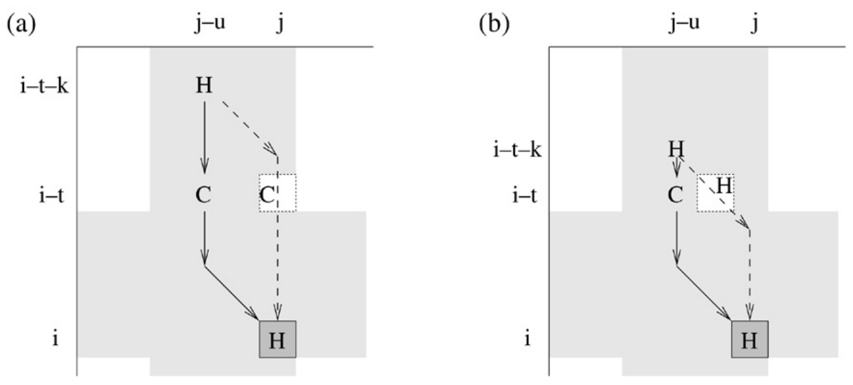

Fig. 10. Proof of Lemma 12.

Lemma 13. *If $H_{i,j+r}$ for $1 \le r \le t-2$ comes from $C_{i-t,j+r-u}$ for $r+1 \le u \le \min\{\min\{s,t\}-1+r, t-1\}$, $H_{i,j+r}$ comes from $C_{i-t,j}$ or $H_{i-t,j+r-l}$ for $r \le l \le \min\{\min\{s,t\}-2+r, t-2\}$. (See Fig. 11.)*

Proof. Similar to the proof of Lemma 12. □

The symmetric versions of the above lemmas hold.

Lemma 12 shows that we do not need $C_{i-t,j-k}$ for $1 \le k \le \min\{s,t\}-1$ to compute the bottom-right corner entry H_{ij}. Thus, by Theorem 10 and Lemma 12, in order to compute H_{ij} we need $H_{i-t,j-k}$ for $0 \le k \le t$, $C_{i-t,j}$ and $C_{i-t,j-k}$ for $\min\{s,t\} \le k \le t-1$. Since $C_{i-t,j}$ is already computed in Case 3 of the block which is on top of the current block, and $H_{i-t,j-k}$ for $\min\{s,t\} \le k \le t$ and $C_{i-t,j-k}$ for $\min\{s,t\} \le k \le t-1$ are also already computed in Case 2 of the block which is to the left of the current block, we will show below how to compute $H_{i-t,j-k}$ for $1 \le k \le \min\{t,s\}-1$.

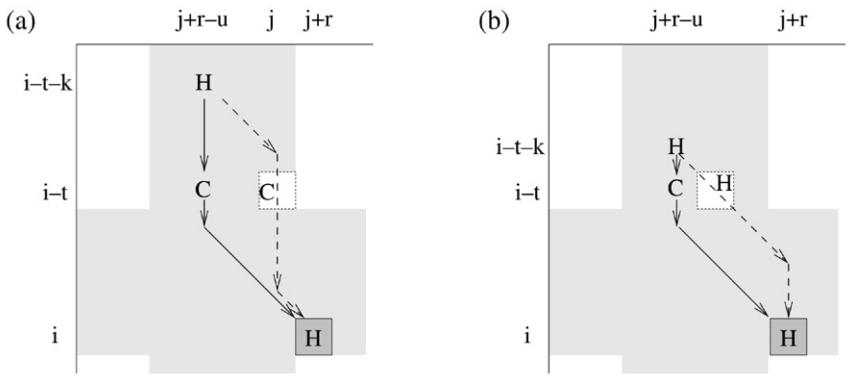

Fig. 11. Lemma 13.

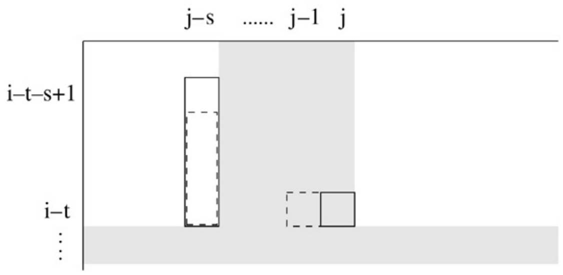

Fig. 12. Two MQUEUEs for $H_{i-t,j}$ and $H_{i-t,j-1}$.

For $1 \leq k \leq \min\{t, s\} - 1$, we compute $H_{i-t,j-k}$ (row E_4 in Fig. 6) using the MQUEUE that was constructed to compute $H_{i-t,j}$ in Case 3, i.e., we compute $H_{i-t,j-k}$ from right to left. See Fig. 12. The difference between two MQUEUEs for $H_{i-t,j}$ and $H_{i-t,j-1}$ is that the oldest element, $R_{i-t-s+1,j-s}$, in the MQUEUE for $H_{i-t,j}$ is deleted. Thus, after one `delete`, we get the MQUEUE for $H_{i-t,j-1}$. Of course, to compute $H_{i-t,j-1}$, μ is added to `max`. Similarly, we can compute $H_{i-k,j-s}$ for $1 \leq k \leq \min\{t, s\} - 1$ (column E_5 in Fig. 6).

The entries in row E_4 of Fig. 6 will be used in Case 2 of the block which is to the right of the current block. To compute the entry $H_{i,j+1}$ in Case 2, we need $H_{i-t,j+1-t..j+1}$ and $C_{i-t,j+1-t+1..j+1}$ among which $H_{i-t,j+1-t..j}$ and $C_{i-t,j+1-t+1..j}$ are the entries that lie outside of the block of Case 2. We already have computed $H_{i-t,j+1-t..j}$ in Case 4 of the current block. For C, we do not need $C_{i-t,j+1-k}$ for $2 \leq k \leq \min\{\min\{s, t\}, t - 1\}$ by Lemma 13 and already have others. Similarly, the entries in column E_5 of Fig. 6 will be used in Case 3 of the block which is below the current block.

3.3. Analysis

Consider the time complexity for each of the four cases. In Case 1, the algorithm takes $O(pq)$ time for a $p \times q$ block because we use the SWG algorithm to compute H_{ij}. Since Case 3 is symmetric to Case 2, we consider Cases 2 and 4.

In Case 2, the algorithm takes $O(q)$ time for a $t \times q$ block. There are q `insert`, q `max` and $q - t$ `delete` operations. Because `max` and `delete` take constant time, q `max` and $q - t$ `delete` operations take $O(q)$ time. Since `insert` takes amortized $O(1)$ time by Lemma 11, q `insert` operations also take $O(q)$ time. Hence, the total time for Case 2 is $O(q)$.

In Case 4, the algorithm takes $O(\min\{t, s\})$ time for a $t \times s$ block. Because it computes one row and one column with $\min\{t, s\}$ entries and the operations of MQUEUE take amortized $O(1)$ time, it takes $O(\min\{t, s\})$ time.

Now consider the worst case time complexity of our algorithm. The worst case is that the lengths of all masked regions of A and B are the same (say, t). Let v and w be the numbers of masked regions of A and B, respectively, and let T and S be the total numbers of N in A and B, respectively (i.e., $T = vt$, $S = wt$). Case 1 covers $(n - T)(m - S)$ entries and it takes $O((n - T)(m - S))$ time. For one masked region of A, Case 2 covers $t \times (m - S)$ entries and Case 4 covers $t \times S$ entries. Thus we compute $(m - S) + S = m$ entries and it takes $O(m)$ time. Similarly, it takes $O(n)$

time for one masked region of B. Since there are v masked regions in A and w in B, the time complexity for Cases 2, 3 and 4 is $O(vm + wn)$. Therefore, the total time complexity is $O((n - T)(m - S) + vm + wn)$.

3.4. Algorithm for the linear gap penalty model

Because the linear gap penalty model is a special case of the affine gap penalty model, it can be solved by the algorithm for the affine gap penalty model. From Lemmas 2, 6 and 7, however, we derive the following theorem which provides a simpler recurrence for an insignificant entry H_{ij}.

Theorem 14.

$$H_{ij} = \begin{cases} \max\left\{ j\sigma, \max\limits_{0\le s\le j-1}\{H_{i-t,j-s} + s\sigma - (t-s)\mu\}, 0\right\} & \textit{for } 1 \le j \le t \\ \max\left\{ \max\limits_{0\le s\le t}\{H_{i-t,j-s} + s\sigma - (t-s)\mu\}, 0\right\} & \textit{for } t < j \le m. \end{cases}$$

Using the new recurrence of Theorem 14, we can give a local alignment algorithm for the linear gap penalty model that is similar to the algorithm for the affine gap penalty model, and the time complexity is also $O((n - T)(m - S) + vm + un)$.

4. Global alignment algorithm

In this section we present an algorithm that finds an optimal global alignment without computing insignificant entries. We first give a new recurrence for insignificant entries H_{ij}, and then give an $O((n - T)(m - S) + vm + wn)$ time algorithm.

We provide some lemmas to make a new recurrence for an insignificant entry H_{ij} for an optimal global alignment. Lemmas 1–5 still hold for a global alignment. However, because there are some differences between the recurrences for a local alignment and a global alignment, we cannot use Lemmas 6 to 9. The following lemmas handle the case where H_{ij} can come from the leftmost column.

Lemma 15. *If $t < j \le m$ and H_{ij} comes from $H_{i-u,0}$ for $0 \le u \le t$, H_{ij} comes from $H_{i-t,j-t}$ or $H_{i,0}$.*

Proof. We prove the lemma in three cases.

(i) $H_{ij} = H_{i-u,0} - g_{j-u} + u\sigma$ and $\sigma + 2\mu \ge 0$: see Fig. 13(a). $H_{ij} \ge H_{i-t,0} - g_{j-t} + t\sigma$ by Lemma 1 and $H_{i-u,0} = H_{i-t,0} - (t-u)\mu$ by definition. Hence, $-(t-u)\mu - g_{j-u} + u\sigma \ge -g_{j-t} + t\sigma$, i.e., $0 \ge (t-u)(\sigma + 2\mu)$. However, $t \ge u$ and $\sigma + 2\mu \ge 0$, and thus the equality must hold. Therefore $H_{ij} = H_{i-t,0} - g_{j-t} + t\sigma$. By the way, by Lemma 1, $H_{i-t,j-t} \ge H_{i-t,0} - \gamma - (j-t)\mu$ and $H_{ij} \ge H_{i-t,j-t} + t\sigma$. Then, $H_{ij} \ge H_{i-t,0} - \gamma - (j-t)\mu + t\sigma$. Thus the equality must hold and therefore H_{ij} comes from $H_{i-t,j-t}$.

(ii) $H_{ij} = H_{i-u,0} - g_{j-u} + u\sigma$ and $\sigma + 2\mu < 0$: see Fig. 13(b). $H_{ij} \ge H_{i,0} - g_j$ by Lemma 1 and $H_{i,0} = H_{i-u,0} - u\mu$ by definition. Hence, $-g_{j-u} + u\sigma \ge -u\mu - g_j$, i.e., $u(\sigma + 2\mu) \ge 0$. However, $\sigma + 2\mu < 0$ and $u \ge 0$, and thus the equality holds only when $u = 0$. Therefore H_{ij} comes from $H_{i,0}$.

(iii) $H_{ij} = H_{i-u,0} - g_{j-u} + u\sigma$: see Fig. 13(c). $H_{ij} \ge H_{i,0} - g_j$ by Lemma 1 and $H_{i,0} = H_{i-u,0} - u\mu$ by definition. Hence, $-g_u \ge -u\mu$, i.e., $0 \ge \gamma$. However, $\gamma \ge 0$ and thus the equality must hold. Therefore H_{ij} comes from $H_{i,0}$. □

Lemma 16. *If $0 < j \le t$ and H_{ij} comes from $H_{i-u,0}$ for $0 \le u \le t$, H_{ij} comes from $H_{i-j,0}$ or $H_{i,0}$.*

Proof. Similar to the proof of Lemma 15. □

The following lemmas handle the case where C_{ij} can come from the leftmost column. Since the proofs of Lemmas 17 and 18 are similar to that of Lemma 15, we omit them.

Lemma 17. *If $t \le j \le m$ and C_{ij} comes from $H_{i-u,0}$ for $1 \le u \le t$, C_{ij} comes from $H_{i-t,j-k}$ for $0 \le k \le t - 1$.*

Lemma 18. *If $0 < j < t$ and C_{ij} comes from $H_{i-u,0}$ for $1 \le u \le t$, C_{ij} comes from $H_{i-j-1,0}$ or $H_{i-t,j-k}$ for $0 \le k < j$.*

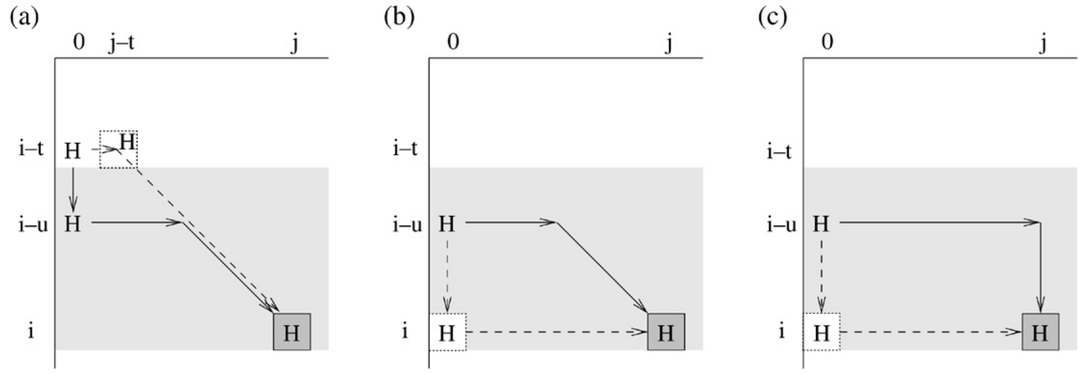

Fig. 13. Proof of Lemma 15.

By the above lemmas, we derive the following theorem which provides a new recurrence for an insignificant entry H_{ij}.

Theorem 19. *Let $A[i-t+1..i]$ be a masked region of length t. Then H_{ij} can be computed by the following recurrence:*

$$C_{ij} = \begin{cases} \max\left\{ \begin{array}{l} \max\limits_{0\le s\le j-1}\{H_{i-t,j-s} + s\sigma - \gamma - (t-s)\mu\}, \\ C_{i-t,j} - t\mu,\ H_{i-j-1,0} + j\sigma - \gamma - \mu \end{array} \right\} & \textit{for } 1 \le j < t \\ \max\left\{ \begin{array}{l} \max\limits_{0\le s\le t-1}\{H_{i-t,j-s} + s\sigma - \gamma - (t-s)\mu\}, \\ C_{i-t,j} - t\mu \end{array} \right\} & \textit{for } t \le j \le m. \end{cases}$$

$$H_{ij} = \begin{cases} \max\left\{ \begin{array}{l} \max\limits_{1\le s\le j-1}\{C_{i-t,j-s} + s\sigma - (t-s)\mu\}, \\ H_{i-j,0} + j\sigma,\ C_{ij},\ H_{i,0} - \gamma - j\mu \end{array} \right\} & \textit{for } 1 \le j \le t \\ \max\left\{ \begin{array}{l} \max\limits_{1\le s\le t-1}\{C_{i-t,j-s} + s\sigma - (t-s)\mu\}, \\ H_{i-t,j-t} + t\sigma,\ C_{ij},\ H_{i,0} - \gamma - j\mu \end{array} \right\} & \textit{for } t < j \le m. \end{cases}$$

Using the recurrence of Theorem 19, we can give a global alignment algorithm for the affine gap penalty model. The algorithm is essentially the same as the local alignment algorithm, and the time complexity is $O((n-T)(m-S)+vm+un)$.

5. Conclusion

In this paper we have presented an efficient algorithm that finds an optimal local alignment by skipping masked regions of sequences. The algorithm is based on a new recurrence for an insignificant entry and a data structure MQUEUE. We also gave an efficient global alignment algorithm.

Our technique of skipping some regions using recurrences for insignificant entries can be applicable to other problems [2] where dynamic programming tables are divided into blocks.

References

[1] A. Batzoglou, D.B. Jaffe, K. Stanley, J. Butler, et al., ARACHNE: A whole-genome shotgun assembler, Genome Research 12 (2002) 177–189.
[2] M. Crochemore, G.M. Landau, B. Schieber, M. Ziv-Ukelson, in: C.S. Iliopoulos, T. Lecroq (Eds.), Re-use Dynamic Programming for Sequence Alignment: An Algorithmic Toolkit, String Algorithmics, King's College London Publications, 2005, pp. 19–59.
[3] M. Crochemore, G.M. Landau, M. Ziv-Ukelson, A subquadratic sequence alignment algorithm for unrestricted scoring matrices, SIAM Journal on Computing 32 (6) (2003) 1654–1673.
[4] B. Ewing, L. Hillier, M.C. Wendl, P. Green, Base-calling of automated sequencer traces using Phred. I. Accuracy assessment, Genome Research 8 (1998) 175–185.
[5] H. Gajewska, R.E. Tarjan, Deques with heap order, Information Processing Letters 22 (1986) 197–200.
[6] GCG Documentation, http://www-igbmc.u-strasbg.fr/BioInfo/GCGdoc/Doc.html.
[7] O. Gotoh, An improved algorithm for matching biological sequences, Journal of Molecular Biology 162 (1982) 705–708.

[8] P. Green, PHRAP, http://www.phrap.org.
[9] D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, New York, 1997.
[10] J.W. Kim, K. Roh, K. Park, et al., MLP: Mate-based Layout with PHRAP, in: 7th Annual Inter. Conference on Research in Computational Molecular Biology — Currents in Computational Molecular Biology, 2003, pp. 65–66.
[11] E.W. Myers, G.G. Sutton, A.L. Delcher, I.M. Dew, D.P. Fasulo, et al., A whole-genome assembly of Drosophila, Science 287 (2000) 2196–2204.
[12] NC-UIB, Nomenclature for incompletely specified bases in nucleic acid sequences. Recommendations 1985, The European Journal of Biochemistry 150 (1985) 1–5.
[13] NCBI, http://www.ncbi.nlm.nih.gov.
[14] NCBI, ftp://ftp.ncbi.nlm.nih.gov/pub/TraceDB.
[15] A.F.A. Smit, R. Hubley, P. Green, RepeatMasker Open-3.0, 1996–2004, http://www.repeatmasker.org.
[16] T.F. Smith, M.S. Waterman, Identification of common molecular subsequences, Journal of Molecular Biology 147 (1981) 195–197.
[17] J. Wang, G.K. Wong, P. Ni, et al., RePS: A sequence assembler that masks exact repeats identified from the shotgun data, Genome Research 12 (2002) 824–831.
[18] M.S. Waterman, Introduction to Computational Biology, Chapman & Hall, London, 1995.