Updating a table of bounds on the minimum distance of binary linear codes

TECHNISCHE HOGESCHOOL EINDHOVEN

NEDERLAND

ONDERAFDELING DER WISKUNDE

EN INFORMATICA

EINDHOVEN UNIVERSITY OF TECHNOLOGY

THE NETHERLANDS

DEPARTMENT OF MATHEMATICS AND

COMPUTING SCIENCE

UPDATING A TABLE OF BOUNDS ON THE MINIMUM DISTANCE OF BINARY LINEAR CODES

by

Tom Verhoeff

AMS Subject Classification: 94B05

EUT Report 85-WSK-01

ISSN 0167-9708

Coden: TEUEDE

Eindhoven, January 1985

November 1984, Revised January 1985

Updating a Table of Bounds on the Minimum Distance of Binary Linear Codes

Tom Verhoeff

Department of Mathematics and
Computing Science,
Eindhoven University of Technology

Abstract--This paper presents an algorithm for updating a table of bounds on the minimum distance of binary linear codes. Using a PASCAL program implementing this algorithm an updated table of bounds has been generated for codeword lengths less than 128.

CONTENTS

1. Introduction

In this paper a [n,k,d]-code is a binary linear code with codeword length n, dimension k and minimum distance at least d; n, k and d are called the code's parameters, and [n,k,d] its type. Let $d_{max}(n,k)$ be defined for integers n and k with $1 \leq k \leq n$ by

$$d_{max}(n,k) := \text{MAX} \{ d \mid \text{there exists a } [n,k,d]\text{-code} \} \qquad (1)$$

No practical general method to determine $d_{max}(n,k)$, given n and k, is currently known. The existence of a [n,k,d]-code implies $d_{max}(n,k) \geq d$; similarly, the nonexistence of [n,k,d]-codes means $d_{max}(n,k) \leq d-1$. Using the best codes known and the sharpest nonexistence results, Helgert and Stinaff compiled a table of upper and lower bounds on $d_{max}(n,k)$ for $n < 128$, which was published as [A] in 1973. Many new codes and nonexistence proofs tightening their bounds, however, have appeared since then.

We were assigned the task of producing an updated table of bounds incorporating these new results. This task reduces to two subtasks: collecting new results, and updating the table. The first subtask is a matter of scanning the available literature, extracting actual bounds, and checking their usefulness. Usually this raises no additional problems, because articles on bounds (existence and nonexistence of binary linear codes) are readily identifiable as such, and often the table in [A] is explicitly referred to.

The second subtask, which is the topic of this report, turns out to be somewhat more complicated than may at first seem. Updating, of course, simply consists in replacing some values by new ones. Consider, however, the improvement of a lower bound by the discovery of a new code. From this new code many others can be derived by applying general code construction techniques, possibly involving some previously known codes as well. A few standard techniques involving only the new code are: adding a parity check, shortening, and puncturing; among the techniques combining two codes are: concatenation, and the, so called, (u,u+v)-construction (for details see section 2). This construction process can be repeated, i.e. from the newly derived codes still more codes can be derived using the same techniques (like repeatedly shortening). Not all of these (indirect) derivatives are necessarily good codes, but some may improve the currently known lower bounds.

In general, whenever a promising new code is found, most derivatives will not yet have been constructed (a notable exception being the addition of a parity check). When compiling a table of bounds, however, one is obviously interested in the best that is achievable. So, if easily derived codes could improve a bound, they will have to be constructed and taken into consideration. The situation for upper bounds is analogous, because the nonexistence of codes of one type leads--via similar constructions--to the nonexistence of others. To put all of this in a

somewhat different language: the improvement of one bound can result in the improvement of others. You might say that improvements can propagate according to certain "propagation rules." It is this erratic--but desirable--propagation of improvements that complicates the updating task and makes the "pencil-and-eraser" approach unworkable.

We set ourselves the goal of producing a table of bounds that is "closed" or "invariant" under a set of propagation rules, i.e. the final table has to be such that no more improvement is possible by applying any propagation rule (the table in [A] is not closed, see section 5). This still leaves us the freedom of choosing our propagation rules. The more rules the better the bounds, but also the more effort it will demand to compute the closure. We shall come back to the choice of propagation rules in section 3. A second goal is to maintain a reference with each bound, indicating where it originated, like the tables in [A]. This reference tells whether a bound resulted from a propagation rule (and if so which one) or whether it was obtained otherwise (and if so in what article). The references allow one to trace each bound to its ultimate origins and thus should improve the usability of the table, ease its checking, and thereby increase its credibility.

We shall aim at an algorithm that will restore the table's invariance after the improvement of a single bound, by systematically applying propagation rules until no more change is possible. This algorithm should also keep the references up to date. Making one improvement and applying the algorithm will be called making an update based on this improvement. Compiling a complete table (8128 pairs of bounds for $n < 128$) is now a matter of starting with a trivial invariant table and making an update based on each improvement found in the literature. The table in [A] can roughly be thought of as constructed from a little over 150 of such updates (but, again, it is not closed, as opposed to any table resulting from the suggested updating algorithm). Our new table embodies more than 80 additional updates (making over 30 previous improvements superfluous).

Implementing this algorithm as a computer program has several advantages over manually updating the table. Automatic updating is less prone to error, it takes care of future updates on the table, and it enables us to check the table in [A] (this table has to be reconstructed anyway, since typing it into the computer would defeat the first advantage).

We have developed a package of programs, written in PASCAL on the university's mainframe computer, for the maintenance of a table of upper and lower bounds on $d_{max}(n,k)$ with $1 \leq k \leq n < 128$. Besides the updating algorithm this package includes such features as generating an initial (not too trivial) table, maintaining statistics, generating reports on updates, and pretty-printing (part of) a table.

The following sections expose the underlying ideas. We have tried to separate several levels of concerns. Section 2 deals with theoretical aspects: propagation rules, how to obtain them from general construction techniques, some of their properties, how to use them in computing the closure after the improvement of a bound, and the placement of associated references. Section 3 is about the more practical (program design) aspects: our choice of propagation rules, the updating algorithm, the generation of an initial invariant table, this includes the problems due to the finiteness of the table. Section 4 is concerned with implementation details of the actual programs, it also explains how to use our table (interpret the references as justification for an entry). Finally, section 5 summarizes some of our experiences after using the package. The appendices contain the results, especially the updated version of the table of bounds, and complete source listings of the two most important programs.

2. Theoretical Discussion

In this section we shall give a more detailed treatment of improvement propagation. In order to describe the notion of a propagation rule and some related concepts, we need a suitable context and a few notations. A complete formalization of all the concepts involved would lead us too far astray and wouldn't make for better reading, therefore we shall sometimes rely on intuition, if it does not seem to harm understanding. PASCAL-like constructs will be used for some descriptions.

2.1. General Code Construction Techniques

First we deal with a familiar aspect of coding theory: general techniques (for binary linear codes) to construct a new code from one or more known codes. These general construction techniques form the basis of the propagation rules. The adjective "general" is intended to restrict attention to techniques that (a) allow codes of (almost) any type to serve as input code, and (b) need not know more about the structure of the input codes than their type. We are not so much interested in the construction techniques themselves as in their effect. By the effect of a construction we mean the resulting code's parameters expressed as a function of the parameters of the input codes (this is possible because of assumption (b)). We use the same identification for constructions as in [A]; Property 1 will be abbreviated P1, etc. Construction B (discussed below) shows how a technique that is not general, can be transformed into a (somewhat weaker) general technique.

Below we describe the effects of several general construction techniques. The proposition "there exists a [n,k,d]-code" is abbreviated to [n,k,d] (the code's type). The effect of a construction is in the first place written as a logical implication (==>, which can also be read as "yields"), with the type of the input code(s) on the left-hand side and the type of the resulting code on the right-hand side. Universal quantification over free variables (often n, k and d) is intended, but not explicitly mentioned; sometimes a restriction on their range will be included ($1 \leq k \leq n$ is always required).

Each code construction technique can also be used to derive the nonexistence of codes of one type from that of another, by contraposition or the standard *reductio ad absurdum* argument (if the one would exist then so would the other by the construction under discussion, but the other does not exist, contradiction, so the one does not exist). The following relationship between code (non)existence and bounds on d_{max} allows other equivalent formulations of the effect of a construction technique:

[n,k,d]	is equivalent to	$d_{max}(n,k) \geq d$	(2)
not [n,k,d]	is equivalent to	$d_{max}(n,k) \leq d-1$	(3)

For each construction technique, therefore, we give a number of equivalent propositions describing its effect, these are true propositions by virtue of the construction. This enables one to switch from one mode of thinking to another, without too much interruption. Eventually we derive the propagation rules from them.

Property 1 is a code construction without input codes, and thus is not useful for improvement propagation. It is mentioned, because it is used when generating an initial table of bounds (see below), and because a "static" bound on d_{max} depends on it ((20), see remarks below P4). All these equivalences require proofs, which we have omitted, since they are all very similar. Property 2 serves as an example of how the proofs run.

INTERMEZZO: Some functions and a few of their properties.

Integer valued functions defined for a real argument:

```
ceil(x)       := smallest integer not less than x
floor(x)      := largest integer not greater than x
evenceil(x)   := smallest even integer not less than x
evenfloor(x)  := largest even integer not greater than x
oddceil(x)    := smallest odd integer not less than x
oddfloor(x)   := largest odd integer not greater than x
```

Boolean valued functions defined for an integer argument:

```
even(n) := n is even
odd(n)  := n is odd
```

Some properties (for real x and y, and integer n):

the 6 ceil/floor functions are idempotent: f(f(x)) = f(x), and monotonous: if $x \geq y$ then $f(x) \geq f(y)$ (4)

$evenceil(x) = 2 * ceil(x/2)$ (5)

$x \geq evenceil(y)$ is equivalent to $evenfloor(x) \geq y$ (6)

value of	*evenceil*(n)	*evenfloor*(n)	*oddceil*(n)	*oddfloor*(n)
if *even*(n)	n	n	n + 1	n - 1
if *odd*(n)	n + 1	n - 1	n	n

(7)

$evenceil(oddfloor(n)) = oddfloor(n)+1 = evenceil(n) \geq n$ (8)

$evenfloor(n) = oddfloor(n+1) - 1$ (9)

Using PASCAL constructs (on the right-hand side) we can write:

```
ceil(n/2)    ~ (n+1) DIV 2
floor(x)     = trunc(x)
evenceil(n)  = n + ord(odd(n))
             = n + n MOD 2
```

(End of Intermezzo)

P1: Boundary cases

[n,1,n] and not [n,1,n+1] and [n,n,1] and not [n,n,2]
$d_{max}(n,1) = n$ and $d_{max}(n,n) = 1$

P2: Adding a parity check

[n,k,d] ==> [n+1,k,*evenceil*(d)] (10)
if $d_{max}(n,k) \geq d$ then $d_{max}(n+1,k) \geq$ *evenceil*(d) (11)
$d_{max}(n+1,k) \geq$ *evenceil*($d_{max}(n,k)$) (12)

not [n,k,d] ==> not [n-1,k,*oddfloor*(d)] (13)
if $d_{max}(n,k) \leq d$ then $d_{max}(n-1,k) \leq$ *evenfloor*(d) (14)
$d_{max}(n-1,k) \leq$ *evenfloor*($d_{max}(n,k)$) (15)

In (13) to (15) n > 1 is supposed. Propositions (10) to (15) are equivalent. The equivalence proofs are fairly straightforward, relying on (2) and (3), and often some substitutions.

(10) <==> (11)		on account of (2)
(11) ==> (12)		take $d = d_{max}(n,k)$ in (11)
(11) <== (12)		due to (4) (monotonicity of *evenceil*)
(10) ==> (13)		substitute n-1 for n and *oddfloor*(d) for d in (10), use (8) and contraposition
(10) <== (13)		similar
(13) <==> (14)		substitute d+1 for d in (13), use (3) and (9)
(14) ==> (15)		take $d = d_{max}(n,k)$ in (14)
(14) <== (15)		monotonicity of *evenfloor*
(12) <==> (15)		substitute n-1 for n in (12) and use (6)

REMARKS:
--P2 is slightly different from Property 2 in [A], although, no doubt, the formulation we have given was intended (the parity check is not supposed to be added only to *optimal* codes with *odd* minimum distance). From P2 and P3 (see below) follows:
if *odd*($d_{max}(n,k)$) then $d_{max}(n+1,k) = d_{max}(n,k)+1$ (16)
--P2 implies: $d_{max}(n+1,k) \geq d_{max}(n,k)$.

P3: Puncturing (deleting a coordinate)

[n,k,d] ==> [n-1,k,d-1] \
if $d_{max}(n,k) \geq d$ then $d_{max}(n-1,k) \geq d-1$ > (k<n)
$d_{max}(n-1,k) \geq d_{max}(n,k)-1$ /

not [n,k,d] ==> not [n+1,k,d+1]
if $d_{max}(n,k) \leq d$ then $d_{max}(n+1,k) \leq d+1$
$d_{max}(n+1,k) \leq d_{max}(n,k)+1$

P4: Shortening
(selection on and subsequent deletion of a coordinate)

$$[n,k,d] \Longrightarrow [n-1,k-1,d] \quad \text{if} \quad d_{max}(n,k) \geq d \text{ then } d_{max}(n-1,k-1) \geq d \quad \backslash$$
$$d_{max}(n-1,k-1) \geq d_{max}(n,k) \quad / \quad > (k>1)$$

$$\text{not } [n,k,d] \Longrightarrow \text{not } [n+1,k+1,d] \quad \text{if} \quad d_{max}(n,k) \leq d \text{ then } d_{max}(n+1,k+1) \leq d$$
$$d_{max}(n+1,k+1) \leq d_{max}(n,k)$$

REMARKS:
--P2, P3 and P4 are monotonicity properties of d_{max}. These are needed for some of the equivalences below. By induction we get from

P2: if $n \geq m$ then $d_{max}(n,k) \geq d_{max}(m,k)$ (17)

P3: if $n \geq m$ then $d_{max}(n,k) \leq d_{max}(m,k)+n-m$ (18)

P4: if $s \geq 0$ then $d_{max}(n,k) \geq d_{max}(n+s,k+s)$ (19)

--Using (19) and P1 we get:

$$d_{max}(n,k) \leq d_{max}(n-k+1,1) = n-k+1 \quad (20)$$

--Combining (19) and (17), taking $s \geq 0$, yields:

$d_{max}(n,k) \geq d_{max}(n+s,k+s) \geq d_{max}(n,k+s)$, or equivalently

$$\text{if } j \geq k \text{ then } d_{max}(n,k) \geq d_{max}(n,j) \quad (21)$$

A: Helgert and Stinaff construction ([A], residual code)

$$[n,k,d] \Longrightarrow [n-d,k-1,ceil(d/2)] \quad \text{if} \quad d_{max}(n,k) \geq d \text{ then } d_{max}(n-d,k-1) \geq ceil(d/2) \quad \backslash$$
$$d_{max}(n-d_{max}(n,k),k-1) \geq ceil(d_{max}(n,k)/2) \quad > (k>1)$$
$$2*d_{max}(n-d_{max}(n,k),k-1) \geq evenceil(d_{max}(n,k)) \quad /$$
$$2*d_{max}(n-d_{max}(n,k),k-1) \geq d_{max}(n,k) \quad /$$

E: One-step Griesmer bound (equivalent to A)

$$\text{not } [n,k,d] \Longrightarrow \text{not } [n+2d,k+1,2d]$$
$$\text{not } [n,k,d] \Longrightarrow \text{not } [n+2d-1,k+1,2d-1]$$
$$\text{if} \quad d_{max}(n,k) \leq d \text{ then } d_{max}(n+2d+1,k+1) \leq 2d$$
$$d_{max}(n+2*d_{max}(n,k)+1,k+1) \leq 2*d_{max}(n,k)$$

REMARKS:
--There is only a historical reason for using different identifiers (A and E) to denote these equivalant properties.
--The last line of A and of E are quite similar, but it is not so trivial to prove their equivalence using (17) and (18).
--From $k > 1$ and (20) follows $d_{max}(n,k) < n$, so that the first argument of d_{max} in the last three lines of A is positive.

B: Construction using dual code (Yi and repeated shortening)

$[n,k,d]$ and not $[n,n-k,s+1]$ ==> $[n-s,k-s+1,d]$
if $d_{max}(n,k) \geq d$ and $d_{max}(n,n-k) \leq s$
then $d_{max}(n-s,k-s+1) \geq d$
$d_{max}(n-d_{max}(n,n-k),k-d_{max}(n,n-k)+1) \geq d_{max}(n,k)$

not $[n,k,d]$ and not $[n+s,n-k+1,s+1]$ ==> not $[n+s,k+s-1,d]$
if $d_{max}(n,k) \leq d$ and $d_{max}(n+s,n-k+1) \leq s$
then $d_{max}(n+s,k+s-1) \leq d$
$d_{max}(n+s_o,k+s_o-1) \leq d_{max}(n,k)$
whenever $s_o = MIN\ \{\ s > 0\ ;\ d_{max}(n+s,n-k+1) \leq s\ \}$

REMARKS:

--Construction Yi (see [M]) is not a general technique in our sense, since it involves the minimum distance of the dual code as well. Construction Yi states: if there exists a $[n,k,d]$-code C and t is the minimum distance of the dual code of C, then there exists a $[n-t,k-t+1,d]$-code. Repeated application of P4 yields a $[n-s,k-s+1,d]$-code for any $s \geq t$. Obviously $d_{max}(n,n-k) \geq t$ holds, so it is fine if $s \geq d_{max}(n,n-k)$. This results in a general technique that constructs a new code from one code and the nonexistence of another.

--The existence of s_o is not asserted. In fact, it does not exist for $k = n$, and rightly so.

--Other equivalences might have been added, e.g.:
$[n,k,d]$ ==> $[n,n-k,s+1]$ or $[n-s,k-s+1,d]$

C: Concatenation of codes

$[n,k,d]$ and $[m,k,c]$ ==> $[n+m,k,d+c]$ \
if $d_{max}(n,k) \geq d$ and $d_{max}(m,k) \geq c$ > $(k \leq m)$
then $d_{max}(n+m,k) \geq d+c$ /
$d_{max}(n+m,k) \geq d_{max}(n,k) + d_{max}(m,k)$ /

not $[n,k,d]$ and $[m,k,c]$ ==> not $[n-m,k,d-c]$ \
if $d_{max}(n,k) \leq d$ and $d_{max}(m,k) \geq c$ > $(k \leq m < n)$
then $d_{max}(n-m,k) \leq d-c$ /
$d_{max}(n-m,k) \leq d_{max}(n,k) - d_{max}(m,k)$ /

D: (u,u+v)-construction

$[n,k,d]$ and $[n,j,c]$ ==> $[2n,k+j,MIN(d,2c)]$ \
if $d_{max}(n,k) \geq d$ and $d_{max}(n,j) \geq c$ > $(j \leq n)$
then $d_{max}(2n,k+j) \geq MIN(d,2c)$ /
$d_{max}(2n,k+j) \geq MIN\ \{\ d_{max}(n,k),\ 2*d_{max}(n,j)\ \}$ /

REMARKS:

--We omitted the nonexistence consequences because they turn out to be of little practical value, besides, they do not allow a compact presentation (case analysis required).

2.2 The Table of Bounds and Some Operations on It

Although we may have more than one table of bounds at our disposal, the operations we have in mind work on only one such table (we shall not consider operations combining two or more tables). From now on we shall speak of *the* table of bounds, being the table that is the subject of all operations. We represent that single table of bounds for the theoretical discussion (in the programs a different representation is used) by two triangular lower-left matrices (two-dimensional arrays, mappings) Lb(n,k) and Ub(n,k) with $1 \leq k \leq n$; for the time being we do not bother about their size, take the range of n to be upwardly unbounded. A restriction on the range of n introduces what we shall call an artificial boundary, k = 1 and k = n are called natural boundaries. The matrix elements of Lb and Ub are, respectively, lower and upper bounds on d_{max}, i.e. positive integers, satisfying:

$$Lb(n,k) \leq d_{max}(n,k) \qquad (22.a)$$

$$d_{max}(n,k) \leq Ub(n,k) \qquad (22.b)$$

If Lb(n,k) = Ub(n,k), then $d_{max}(n,k)$ is known and equals the common value. The integer d, $d \leq d_{max}(n,k)$, is an improvement for the lower bound in the table at parameter pair (n,k) if $d > Lb(n,k)$; similarly: if $d \geq d_{max}(n,k)$, then d improves the upper bound at (n,k) when $d < Ub(n,k)$. Using pseudo-PASCAL we want to define the boolean function *IsImprovement* to catch this idea. Before doing so we introduce the notion of a bound kind, in order to distinguish between lower and upper bounds, and the notion of a location triple, indicating a position in the table by specifying a bound kind, a length, and a dimension. Furthermore, we use the term bound quadruple to stand for the quadruple of the bound's kind, the length and dimension of the code type it refers to, and the actual bound on the minimum-distance, i.e. for a triple combined with a bound. From now on we shall often use bound in the sense of bound quadruple instead of a single integer; confusion is not very likely because of the context, but at times we shall say bound quadruple to make the distinction clear. The function *Bound* returns the bound quadruple corresponding to a location in the table. We also introduce the auxiliary function *IsBound* as a concise representation of the proposition "q is a proper bound quadruple", i.e. if q = (b,n,k,d): "d is a bound of kind b for $d_{max}(n,k)$." *IsBound* is a conceptual function only, for if we could program it efficiently in PASCAL, many problems concerning d_{max} would have been solved. So when invoking *IsImprovement*, its pre-condition must be known to hold by some other means than the use of *IsBound*. We also need a way to change the table; we have chosen the end-point modification by assignment defined below as *AssignBound*.

```
TYPE
  boundkind = (lower,upper) ;

  triple    = RECORD { indicating location in the table }
    b       : boundkind ;
    n, k    : integer
    END { triple } ;

  quadruple = RECORD { indicating bound and its location }
    t       : triple ;
    d       : integer
    END { quadruple } ;

VAR
  Lb, Ub: ARRAY [ 1.."inf", 1.."inf" ] OF integer ;
    { both Lb and Ub are lower-left matrices }

FUNCTION Bound(t: triple): quadruple ;
  { Pre-condition: 1 ≤ t.k ≤ t.n.
    Returns the current bound in the table at location t.
  }
  BEGIN
  Bound.t := t ;
  WITH t DO CASE b OF
    lower: Bound.d := Lb(n,k) ;
    upper: Bound.d := Ub(n,k) ;
    END { case }
  END { Bound } ;

FUNCTION IsBound(q: quadruple): boolean ;
  { Is q a proper bound quadruple? Conceptual function! }
  BEGIN
  WITH q, t DO
    IF (1 ≤ k) AND (k ≤ n) THEN
        CASE b OF
          lower: IsBound := (d ≤ dmax(n,k)) ;
          upper: IsBound := (d ≥ dmax(n,k)) ;
          END { case }
      ELSE IsBound := false
  END { IsBound } ;

FUNCTION IsImprovement(q: quadruple): boolean ;
  { Pre-condition: IsBound(q).
    Does q improve the current bound in the table at
    the corresponding position?
  }
  BEGIN
  WITH q, t DO
    CASE b OF
      lower: IsImprovement := (d > Lb(n,k)) ;
      upper: IsImprovement := (d < Ub(n,k)) ;
      END { case }
  END { IsImprovement } ;
```

```
PROCEDURE AssignBound(q: quadruple) ;
  { Pre-condition: IsBound(q).
    Assign q as the new bound in the table.
  }
  BEGIN
  WITH q, t DO
    CASE b OF
      lower: Lb(n,k) := d ;
      upper: Ub(n,k) := d ;
      END { case }
  END { AssignBound } ;
```

REMARKS:

- --We could have defined the table of bounds to be a mapping of triples to integers. This was not done because the bound kind is usually fixed; it would result in longer expressions.
- --Since PASCAL lacks a RECORD constructor, we shall use the conventional mathematical notation for triples and quadruples. Often we shall omit the parentheses around a tuple, when it is the only parameter to a function or procedure, or when tuples are nested.
- --(22) is equivalent to *IsBound*(*Bound*(b,n,k)), for all b,n,k.
- --The pre-condition on *AssignBound* assures that afterwards Lb and Ub still form a table of bounds, i.e. satisfy (22).

There is one way of changing the table that particularly interests us: raising a lower bound or lowering an upper bound, it will be called <u>improving</u> the table (bringing the lower and upper bounds closer together; when they meet, d_{max} is known). Of course we never want to lower a lower bound (or raise an upper bound). In successive tables Lb(n,k) (n and k fixed) should be non-decreasing and Ub(n,k) non-increasing. This kind of change may appropriately be called the one-point <u>Monotonic</u> Modification. It can be defined in pseudo-PASCAL as follows.

```
PROCEDURE MM(q: quadruple) ;
  { Pre-condition: IsBound(q).
    Incorporate the bound quadruple q into the table.
    Post-condition: NOT IsImprovement(q) AND
                    Bound(q,T) is no worse than before.
  }
  BEGIN
  IF IsImprovement(q) THEN AssignBound(q)
  END { MM } ;
```

2.3 The Invariance of the Table of Bounds

In section 2.1 bounds were treated individually, we described how one set of bounds can justify others. Now we can apply this to the ensemble of bounds as collected in the table. For example, combining (22.a) and (2[illegible]), taking [illegible] = (b, n, k) in ([illegible]), gives us

$$d_{max}(n+1,k) \geq evenceil(Lb(n,k)), \tag{23}$$

or equivalently

$$IsBound(lower,n+1,k,evenceil(Lb(n,k))). \tag{24}$$

This means that *MM*'s pre-condition is satisfied, and hence the invocation

$$MM(lower,n+1,k,evenceil(Lb(n,k))) \tag{25}$$

produces a correct and possibly improved table of bounds. Such an improvement will be called an internal improvement. We say that the table is invariant with respect to (or closed under) P2's lower bound (or existence) formulation ($P2_{lower}$ for short), if the table cannot be improved in this way, that is if NOT *IsImprovement*(lower,n+1,k,*evenceil*(Lb(n,k))) holds for all n and k. Applying the definition of *IsImprovement*, this can also be written as: for all (n,k) *evenceil*(Lb(n,k)) $\leq$ Lb(n+1,k). Invariance implies that internal improvements are impossible. We now list the quadruples for which *IsBound* holds, and that can be derived from the construction techniques of section 2.1 using (22). All free variables are shown in parentheses following the construction's name. In general there is a lower and upper quadruple for each construction.

```
P2(n,k):   (lower,n+1,k,evenceil(Lb(n,k)))
           (upper,n-1,k,evenfloor(Ub(n,k)))

P3(n,k):   (lower,n-1,k,Lb(n,k))
           (upper,n+1,k,Ub(n,k))

P4(n,k):   (lower,n-1,k-1,Lb(n,k))
           (upper,n+1,k+1,Ub(n,k))

A(n,k):    (lower,n-Lb(n,k),k-1,ceil(Lb(n,k)/2))
E(n,k):    (upper,n+2*Ub(n,k)+1,k+1,2*Ub(n,k))

B(n,k,s):  (lower,n-Ub(n,n-k),k-Ub(n,n-k)+1,Lb(n,k))
           (upper,n+s,k+s-1,Ub(n,k))  if Ub(n+s,n-k+1) <= s

C(n,m,k):  (lower,n+m,k,Lb(n,k)+Lb(m,k))
           (upper,n-m,k,Ub(n,k)-Lb(m,k))

D(n,k,j):  (lower,2*n,k+j,MIN(Lb(n,k),2*Lb(n,j)))
```

REMARKS:

- --Invariance with respect to $R_{lb}(j,..,s)$ is defined as:
  For all sensible j,..,s: not *IsImprovement*($R_{lb}(j,..,s)$)
- --Each of these quadruples has a length-dimension pair differing from that of all bounds (Lb or Ub) involved in its construction, whatever the values of the free variables.
- --Invariance with respect to $P2_{lower}$, $P3_{lower}$ and $P4_{lower}$ implies monotonicity of Lb similar to that of d_{max} as in (17), (18) and (19). The same holds for Ub in case of

invariance with respect to $P2_{upper}$, $P3_{upper}$ and $P4_{upper}$. In fact d_{max} is possibly equal to Lb or Ub for all n and k, as far as we know. When the table is closed, therefore, Lb and Ub share those properties with d_{max} that depend on the closing constructions.

--A_{upper} == E_{lower}.

--B_{upper} has an exceptional formulation.

If the table is not closed under a set of constructions, then there is at least one construction R_b and a set of values j,..,s for its free variables such that *IsImprovement*(R_b(j,..,s)) holds. That is, an internal improvement is possible, it will be called a variance. Applying *MM*(R_b(j,..,s)) improves the table, and has as a consequence: not *IsImprovement*(R_b(j,..,s)) (this particular variance is removed, see second remark above), although it may cause other new variances. Any finite part T of the table, however, can only take a finite number of improvements, since these are monotonic and reduce

SUM((n,k) in T: Ub(n,k) - Lb(n,k)),

which is a non-negative integer. When the sum equals zero, this means that d_{max} is completely known for that part T of the table.

2.4. Disturbing Invariance and Restoring it

We aim at a table closed under a particular set of constructions (the choice is still to be made). How can invariance be obtained? The method suggested above is repeatedly finding a variance and applying *MM*. For a finite table this process will terminate, and will result in a closed table. The removal of one variance, however, may produce (many) others. A straightforward way of removing all variances is scanning the whole table, making whatever internal improvements that are possible, and rescanning the table until none are found for one complete scan. For a fairly large table this will mean that many constructions are tried without avail every time over and over again. What we need, is a systematic way of keeping track of all variances. This should not be too difficult if it is known what new variances can be introduced by a removal.

Once we have a closed table, invariance can only be disturbed by bound improvements from "outside" (i.e. not based on any of the constructions under which the table is closed), these improvements will be called external. We seek to restore invariance immediately after making a single external improvement. When the bounds in the table are not very tight, it could be more efficient to make all possible external improvements before restoring invariance. But only in the very beginning do we have loose bounds and consequently the opportunity for many improvements. We suggest the following recursive algorithm to restore invariance. It is only an informal attempt, so the terms occurring in it are not well-defined, but it has the right structure.

```
PROCEDURE Restore(t: triple) ;
  { Remove all variances involving t as "input" }
  VAR v: "vector" ; q: quadruple ;
  BEGIN
  FOR "all constructions R" DO BEGIN
    FOR "all occurrences of t in R" DO BEGIN
      v := "vector of values for free variables in R
           corresponding to this occurrence of t" ;
      q := R(v) ;
      IF IsImprovement(q) THEN BEGIN
        AssignBound(q) ;
        Restore(q.t)
        END { if }
      END { for }
    END { for }
  END { Restore } ;
```

When q is a possible external improvement, the calling sequence will be:

```
{ IsBound(q) }
MM(q) ;
{ all variances--if any--involve q.t as "input" }
Restore(q.t)
{ the table is invariant }
```

Notice that *MM* cannot be used in the body of *Restore*. The calling sequence can be rewritten as:

```
{ IsBound(q) }
IF IsImprovement(q) THEN BEGIN
  AssignBound(q) ;
  { all variances--if any--involve q.t as "input" }
  Restore(q.t)
  END { if }
{ the table is invariant }
```

We see that this sequence also occurs inside the body of *Restore*. It can be eliminated in the following way, combining the implied one-point monotonic modification with the restoration of invariance. The resulting recursive procedure is called *Update*.

```
PROCEDURE Update(q: quadruple) ;
  { Pre-condition: IsBound(q).
    Post condition: q is incorporated in the table, AND
      no variances were added, i.e. the ones that were
      introduced have all been removed as well.
  }
  VAR v: "vector" ;
  BEGIN
  IF IsImprovement(q) THEN BEGIN
    AssignBound(q) ;
    FOR "all constructions R" DO BEGIN
      FOR "all occurrences of q.t in R" DO BEGIN
        v := "vector of values for free variables in R
              corresponding to this occurrence of q.t" ;
        Update(R(v))
        END { for }
      END { for }
    END { if }
  END { Update } ;
```

Its calling sequence is:

```
{ the table is closed AND IsBound(q) }
Update(q)
{ bound q has been incorporated AND the table is closed }
```

REMARKS:

--The stacking mechanism involved in the execution of these recursive procedures keeps track of all potential variances. "IF *IsImprovement*" detects true variances (except for the outermost invocation of *Update*, where it checks the usefulness of the new bound), *AssignBound* eliminates one (again excepting the outermost invocation of *Update*, where it makes the external improvement, thereby possibly introducing the first variances), and the FOR-loops (with recursive call) "extend" the list of potential variances with the ones that may have been introduced by *AssignBound*. The initial list is trivial.

--The post-condition of *Update* is to be proved under the assumption that each recursive invocation in the body terminates in the given post-condition (like the step of an induction argument); that is why it is somewhat stronger than what the outermost invocation is to accomplish.

--Because the table is considered unbounded, these procedures may never terminate. But they do not leave any variances, and they do not cycle (get stuck), the only problem is that they may "run away" (see section 3.2 for more on this problem). This can be understood by considering any finite part T of the table, where *IsImprovement* is modified to return false if q.t lies outside T. A termination argument for this situation was given at the end of section 2.3.

2.5 Propagation Rules

Some constructions have more than one input (in particular: B, C and D). When restoring invariance they have to be considered more than once (that is what the second FOR-loop in *Update* does). In constructions C and D the inputs are almost independent, so that the second input can be chosen in many ways even if the first is fixed. The notion of a propagation rule (or function) is a refinement of that of a construction, such that it has only one input. This makes no difference for P2, P3, P4, A, and E. B, C, and D, however, decompose into several propagation rules. The advantage of propagation rules is that they are all alike, this simplifies reasoning about, for example, the effect of applying constructions one after the other (see section 2.6).

We define a propagation rule to be any partial function P, mapping bound quadruples into bound quadruples, possibly involving values of the table Lb and Ub, and which satisfies:

for all q in dom(P): *IsBound*(q) ==> *IsBound*(P(q)) (26)

where dom(P) is P's domain. Since there is only one table, we shall not explicitly write Lb and Ub as arguments or parameters to the propagation function. The function argument will often be called input, the function value sometimes its consequence. By the application of the propagation rule P at the location s we mean the invocation *MM*(P(*Bound*(s))).

Propagation functions can be derived from construction techniques. We use the construction's identifier also as a name for the related propagation function (when there are more functions related to one construction, the distinction will be made by subscripting or priming (')). The if-then formulation of the construction techniques is most suitable for the derivation, because of the implication in (26), and the occurrence of d_{max} in the definition of *IsBound*. The derivation is straightforward. For instance, C'_m was obtained by taking d = Ub(n,k) in the fifth proposition for the effect of construction C (see section 2.1), and afterwards substituting n and m for resp. m and n.

```
P2(b,n,k,d) := (lower,n+1,k,evenceil(d))      case b=lower
               (upper,n-1,k,evenfloor(d))     case b=upper

P3(b,n,k,d) := (lower,n-1,k,d-1)  case b=lower
               (upper,n+1,k,d+1)  case b=upper

P4(b,n,k,d) := (lower,n-1,k-1,d)  case b=lower
               (upper,n+1,k+1,d)  case b=upper

A(b,n,k,d)  := (lower,n-d,k-1,ceil(d/2))      if b=lower
E(b,n,k,d)  := (upper,n+2*d+1,k+1,2*d)        if b=upper
```

$B_1(b,n,k,d)$:= (lower, n-Ub(n,n-k), k-Ub(n,n-k)+1, d) if b=lower
$B_2(b,n,k,s)$:= (lower, n-s, n-k-s+1, Lb(n,n-k)) if b=upper
$B_3(b,n,k,d)$:= (upper, n+s, k+s-1, d) if b=upper
where s = MIN (t | Ub(n+t,n-k+1) $\leq$ t)
$B_4(b,n,k,s)$:= (upper, n, n-k, Ub(n-s,n-k-s+1)) if b=upper

$C_m(b,n,k,d)$:= (lower, n+m, k, d+Lb(m,k)) case b=lower
(upper, n-m, k, d-Lb(m,k)) case b=upper
$C'_m(b,n,k,c)$:= (upper, m-n, k, Ub(m,k)-c)) if b=lower

$D_j(b,n,k,d)$:= (lower, 2*n, k+j, MIN(d, 2*Lb(n,j))) if b=lower
$D'_j(b,n,k,d)$:= (lower, 2*n, k+j, MIN(Lb(n,j), 2*d)) if b=lower

REMARKS:

- "Case" indicates a definition by cases, "if" indicates a restriction on the function's domain.
- The domains of these functions were not clearly defined; b, n, k and d are to be restricted to "values that make sense." The domain may be quite irregular, since it can depend on the actual values in the table.
- B_3 requires explanation. Actually s should have been a parameter (so $B_{3,s}$), satisfying s $\geq$ Ub(n+s,n-k+1). But since $B_{3,t}$ follows from $B_{3,s}$ when s < t (by P4), we might as well restrict ourselves to the minimal s.
- Notice that almost always the the bound kind of the consequence is the same as that of the input. There are two exceptions: B_2 (propagation from upper bound to lower bound) and C' (propagation from lower to upper bound).
- In the propagation functions P2, P3, P4, A and E no values of the table occur; they work directly from input to consequence.
- Propagation functions A and E have disjoint domains and could therefore have been combined into one function.
- For the same reason B_1 could have been combined with any of the other B_i. This was not done for the sake of clarity.

$B_1(b,n,k,Lb(n,k))$ == $B_2(b,n,n-k,Ub(n,n-k))$
$B_3(b,n,k,Ub(n,k))$ == $B_4(b,n+s,n-k+1,s)$ (s as in B_3)
$C_m(upper,n,k,Ub(n,k))$ == $C'_n(upper,m,k,Lb(m,k))$
$D_j(b,n,k,Lb(n,k))$ == $D'_k(b,n,j,Lb(n,j))$

Using propagation rules the notion of invariance can be redefined in an obvious way. Invariance of lower bounds w.r.t. construction D corresponds to invariance w.r.t. propagation rules D_j and D'_j, for all j. With the new notion it is easier to pinpoint what a variance is, it is completely specified by a propagation rule R and a triple s, for which we have *IsImprovement*(R(*Bound*(s))), allowing an internal improvement. Using propagation rules, the procedure *Update* can be rewritten as follows.

```
PROCEDURE Update(q: quadruple) ;
  { Pre-condition: IsBound(q).
    Produce a table that incorporates the bound q, and
    that has no new variances.
  }
  BEGIN
  IF IsImprovement(q) THEN BEGIN
    AssignBound(q) ;
    FOR "all propagation rules R" DO
      IF "q in dom(R)" THEN Update(R(q))
    END { if }
  END { Update } ;
```

A new problem is that propagation rules are not independent. That is, in order for *Update*--as defined above--to work correctly (indeed restoring invariance), the set of propagation rules cannot be chosen arbitrarily. Invariance w.r.t. B_1 is equivalent to invariance w.r.t. B_2, as is readily seen from the last REMARK above. But if invariance w.r.t. B_1 is desired, then B_2 has to be included as well. We need a complete decomposition of a construction into propagation rules. It is obtained by deriving a propagation rule for each occurrence of Lb (or Ub) in the construction, by performing a coordinate transformation such that we get Lb(n,k) (or Ub(n,k)), and then replacing it by d. Construction C, for example, gives two propagation rules for lower bounds that happen to be equivalent. The propagation rules above form complete decompositions.

2.6 Combining Propagation Rules

Propagation rules can be combined by simple functional composition. We shall write P;R to denote the function that maps q onto R(P(q)) (the semicolon was inspired by the sequential composition operator for statements as used in many programming languages). When P and R are propagation functions, then so is P;R, i.e. P;R satisfies (26). We shall now discuss some relationships between the propagation rules given above and some of their composites. We introduce the binary relation -> on bound quadruples, indicating that one bound is at least as useful as another.

$$q1 \rightarrow q2 \text{ is defined as } q1.t = q2.t \text{ AND } q1.d \geq q2.d \text{ case } q1.t.b = \text{lower}; \; q1.d \leq q2.d \text{ case } q1.t.b = \text{upper} \qquad (27)$$

For example, a post-condition of *MM*(q) is: *Bound*(q.t) -> q, expressing that *MM* does not weaken the table. If q1 -> q2 holds, then *IsBound*(q1) implies *IsBound*(q2) (but not necessarily the other way round; for instance, take q1 = (24,15,5) and q2 = (25,15,6)). We define the identity propagation function Id by: Id(q) := q. We shall list a number of relationships between propagation functions in the following format: P r R, where r is ->, <-, or == (equality of quadruples). This is meant to be read as: for all q=(b,n,k,d) in the intersection of dom(P) and dom(R), P(q) r R(q) holds. The postfix operator * on a propagation

function is used to indicate "a sufficient number of times ($\geq$0) composed with itself," it is an existential quantification inside the universal quantification over the argument q, i.e. the number may depend on q. Of course, proofs are required, but in most cases they are trivial, and have been omitted. For lines with # in front a proof is given below.

```
     P3;P2 == Id            if even(d)
     P3;P2 <- Id            in general

     P2;P3 == Id            if odd(d)
     P2;P3 <- Id            in general

     P2;P4 == P4;P2
     P3;P4 == P4;P3

        A  -> P3*;P4
#    P2;A  == A             if odd(d)
#    P2;A  <- A;P2          if even(d)
     P2;A  <- A;P2*         in general
     P3;A  <- A
     P4;A  == A;P4

        E  -> P3*;P4
     P2;E  == E;P2          if even(d)
     P3;E  -> E;P3;P3;P3
     P4;E  == E;P4

        B1 -> P4*;P3
     P2;B1 == P4*           if even(d)
     P3;B1 == B1;P3;P4*     if P4-closed
     P4;B1 == B1;P4*        if P3-closed

     P2;B2 -> B2            if  odd(d) and P4-closed
     P2;B2 -> B2;P4         if even(d) and P4-closed
     P2;B2 -> B2;P4*        if P4-closed, in general
     P3;B2 <- B2            if P4-closed
     P4;B2 -> B2;P2         if P2-closed

        B3 -> P4*;P3
#    P2;B3 == B3;P2;P4*     if P4-closed
     P2;B3 == P4*           if even(d)
     P3;B3;P2;P4* -> B3     if even(d) and
     P4;B3 == B3

     P2;B4 not reachable from B4 by P2,P3,P4
     P3;B4 == B4;P4
     P4;B4 -> B4;P3         if P3 closed

     P2;Cm == Cm;P2         if even(Lb(m,k))
     P2;Cm <- Cm+1          if  odd(Lb(m,k)) and P2-closed
     P3;Cm == Cm;P3
     P4;Cm -> Cm;P4         if P4-closed
        Cm <- Cm-1;P2       if Lb(m,k) = Lb(m-1,k)
        Cm <- Cm+1;P3       if odd(Lb(m,k)) and P2-closed
```

```
     P2;D_j  -> D_j;P2;P2    if P2-closed
     P3;D_j  -> D_j;P3;P3    if P3-closed
     P4;D_j  no relation given
     P2;D'_j -> D'_j;P2;P2   if P2-closed
     P3;D'_j -> D'_j;P3;P3   if P3-closed
     P4;D'_j -> D'_j;P3;P4   if P3-closed
        D_j  -> D'_j         if Lb(n,j) ≤ d
        D'_j -> D_j          if Lb(n,j) ≥ d

Proofs for relationships prefixed with #:
        (P2;A)(lower,n,k,d)
      { def. of ; and of P2 }
     == A(lower,n+1,k,evenceil(d))
      { def. of A }
     == (lower,n+1-evenceil(d),k-1,ceil(evenceil(d)/2))
      { use (5) and (4) (idempotency of ceil) }
     == (lower,n+1-evenceil(d),k-1,ceil(d/2))
      { case analysis: even/odd d, and (7) }
     == (lower,n-d+1,k-1,ceil(d/2))   case even(d)
     == (lower,n-d  ,k-1,ceil(d/2))   case odd(d)
      { def. of A }
     == A(lower,n,k,d)   case odd(d)

        (A;P2)(lower,n,k,d)
      { def. of ; and of A }
     == P2(lower,n-d,k-1,ceil(d/2))
      { def. of P2 }
     == (lower,n-d+1,k-1,evenceil(ceil(d/2)))
      { (8) and def. of -> }
     -> (lower,n-d+1,k-1,ceil(d/2))
      { above }
     == (P2;A)(lower,n,k,d)   case even(d)

        (P2;B_3)(upper,n,k,d)
      { def. of ; and of P2 }
     == B_3(upper,n-1,k,evenfloor(d))
      { def. of B_3 }
     == (upper,n+s-1,k+s-1,evenfloor(d))
        where s := MIN S,   S := { t | Ub(n+t-1,n-k) ≤ t }

        (B_3;P2;P4*)(upper,n,k,d)
      { def. of ; and of B_3 }
     == (P2;P4*)(upper,n+u,k+u-1,d)
        where u := MIN U,   U := { t | Ub(n+t,n-k+1) ≤ t }
      { def. of ; and of P2 }
     == (P4*)(upper,n+u-1,k+u-1,evenfloor(d))
      { def. of P4, taking * as i times }
     == (upper,n+u+i-1,k+u+i-1,evenfloor(d)), where i ≥ 0

If the table is P4-closed, then Ub(n+t,n-k+1) ≤ Ub(n+t-1,n-k). So
S is contained in U,  and hence u ≤ s.  By taking i = s-u, we get
P2;B_3 == B_3;P2;P4*  if P4-closed.
(End of Proofs)
```

REMARKS:

--Let R(i,j,k) stand for a composition of P2, P3, and P4 occurring resp. i, j, and k times in any order. All P4's can be shifted to the right since they commute with both P2 and P3. By repeatedly "cancelling" adjacent P2's and P3's we get

R(i,j,k) <- i-j times P2 ; k times P4 if i≥j

R(i,j,k) <- j-i times P3 ; k times P4 if j≥i

2.7 Associating References

With each bound we associate a reference, indicating the bound's justification. A bound is said to be internally justified, if it can be derived by a construction, possibly involving other bounds in the table (it is the result of a propagation rule or P1); the construction's name may serve as reference in this case. Other (ad hoc) proofs of bounds will be called external justifications; the article containing the proof, or its originator's name, can be used as reference. A bound can, of course, be justified in many ways, even internal and external justifications do not exclude each other. Circular justifications are not acceptable. For example, [8,4,5] and [9,4,6] justify each other by the addition of a parity check (P2) and by puncturing (P3), hardly satisfactory since neither exists.

The procedure *Update* can be given an extra parameter representing the reference for the bound. When the table is changed (by *AssignBound*), the associated reference can be updated as well, reflecting the justification of the new bound. *Update* will never give rise to internal justification cycles. Cycles involving an external justification are of course still possible: for instance, two articles each referring to the other for some intermediate results. Or even more dangerous: an article that constructs bound u via an ingenious method using, it says, bound v from the table. What if u is incorrect because v does not occur in the table, but the incorporation of u has as an indirect consequence v, so that after the update v does occur in the table? The program can not be expected to detect these flaws. There are, however, some additional problems with references. These will be discussed below and in section 3.2.

We prefer the internal justifications P1 through P4 to others, because they are simpler. The corresponding construction is easier to carry out, and the other bound involved--if any--has length and dimension differing by at most one from those of the bound justified. In the table as presented in appendix I, the references P1 through P4 have been omitted, they are called implicit references, the others explicit. This was done in order to make the table better readable. There are many of these, and P4 "commutes" with P2 and P3 (see section 2.6), so that in most cases the choice would seem arbitrary. E.g., [6,3,3] can be derived from [8,4,4] by P3;P4 (via [7,4,3]), or by P4;P3 (via [7,3,4]), so both P3 and P4 are possible as reference on [6,3,3]. The interpretation of an implicit reference requires a little more attention, see section 4.1 for a recipe. The preference of

implicit references complicates the updating process, because now references may have to be changed not only when a bound is improved, but also when it is equalled by a P2, P3, or P4 consequence. In the latter case we shall want to replace the existing reference by an implicit reference (called wiping a reference for short). Some care is needed with reference wiping, in order not to wipe the external reference that started the update (see section 3.2 Ad (iii)).

Although we are anxious to avoid circular justifications, it appears that reference cycles may have to be accepted. Starting from a bound, repeatedly following its reference to another bound (any of the others, if more than one is referenced, as with B, C and D), should ideally end at a bound that is justified externally, or by P1. Imagine, however, a table that allows an improvement at location s, which by a propagation chain improves itself during the restoration of the table's invariance. What should be the reference at s, the external reference, or the last reference of the improving chain? Both are required to reconstruct what has happened. So the history of a table of bounds cannot be neglected. Self-improvement can be detected, but it is only one manifestation of the problem. What about an improvement u that together with bound v constructs bound w, where a consequence of w makes an improvement (or wipes a reference) somewhere in the justification chain for v. This is much harder to detect dynamically during updating. As far as we know (by visual inspection aided with the reports, like in Appendix III), there are no reference cycles in our table. A program could be written to verify this.

3. Pragmatic Considerations

In this section we shall deal with more practical issues, while postponing implementation details to section 4.

3.1. Our Choice of Propagation Rules

As already mentioned in the introduction, invariance under more propagation rules generally implies tighter bounds, but also requires more effort to compute the closure. It does not pay to include a propagation rule that is no stronger (in the sense of (a) having a larger domain, or (b) the relation ->) than all the others and their composites.

We used the following propagation functions in the construction of our tables: P2, P3, P4, A, B, C_m (restricted to b = lower), D_j, D'_j, and E. The upper bound consequences for constructions C and D were omitted, because they did not seem to be very effective, but would give rise to many additional propagation rules; checking all of them for every improved upper bound seemed a waste of computing time. Their ineffectiveness was not proved, but some unsuccessful attempts at internal improvements using them on tables generated without them convinced us. By the way, inspection of the updated table also reveals that constructions C and D have lost (most of) their power for improving lower bounds as well, since even the maximal increase of a lower bound (by making it equal to the upper bound) does not introduce a variance w.r.t. constructions C and D. The contribution to lower bounds by construction B is not very impressive, neither that of A when compared to the amount of upper bound consequences for B and E.

3.2. The Updating Algorithm Revisited

When we designed the procedure *Update* in section 2, there were several aspects that we ignored, and that turn out to be problematical. (i) The table that we intend to maintain is finite, so the artificial boundary comes into play. (ii) *Update* is inefficient. (iii) The updating of associated references has to be incorporated.

Ad (i).

The finiteness of the table is demanded by out storage facilities, and is required for the termination of *Update*. Therefor we introduce an artificial boundary by restricting the length parameter n to values less than, say, 128; this brings along some new problems. Bounds (codes) that fall outside the artificial boundary can be ignored as far as invariance is concerned, although we should not totally neglect them if we want the best possible bounds. Especially not if they are close to the boundary, because by using a construction technique we might very well derive a bound improvement within the table. How far outside does it make sense to keep this up? The problem is that it may take

several applications of propagation rules to get inside; twice puncturing, for instance, seems worth the trouble.

The artificial boundary also causes a referencing dilemma. What reference do we associate with a bound u that is the result of propagation rule P applied to a bound v outside the table (justified by ZZ)? Neither 'P' nor 'ZZ' alone is sufficient to trace u to its origins, since its justification chain can never be followed further than v (because bounds outside the artificial boundary are not stored, there is no reference attached to them).

Ad (ii).

Update considers all possible chains of improvements, it attempts to extend the chain by using a propagation rule. This means that starting from the place where it makes its first improvement, all possible composites of one or more propagation functions are applied. A composite P;R is not considered if P did not result in an improvement, since the table is assumed invariant before the outermost invocation of *Update*, hence R cannot improve if P didn't. This already cuts the number of interesting composites considerably, but in the light of the properties in section 2.6 many more composites can be dropped. For example, none of P2*;P3*;P4*;A need be considered, if all of A;P2*;P4* are investigated as well.

Ad (iii).

Our preference for implicit references and abhorrence of (too obvious) reference cycles poses another problem. The following version of *Update* is unacceptable.

```
PROCEDURE Update2(q: quadruple; r: reference) ; { no good }
  BEGIN
  IF IsImprovement(q) THEN BEGIN
    AssignBound2(q,r) ; { make improvement }
    FOR "all propagation rules R" DO { remove variance }
      IF "q in dom(R)" THEN Update2(R(q),'R')
    END { improvement }
  ELSE IF "q == Bound(q.t)" THEN
    IF "r is an implicit reference" THEN
      AssignBound2(q,r) { replace reference }
  END { Update2 } ;
```

n,k	6	7
	G	
22	9	8
23	10	8-9
	E	
24	10	9-10

n,k	6	7
22	9	8
23	10	9
	E	T3
24	10	10

It is not acceptable because the following scenario is possible. The bound u = (lower,24,7,10) is to be incorporated with reference 'T3' in the table shown on the left (it was taken from the table in [A]; for an explanation of the table's format see section 4.1). This bound u has as P3-consequence v = (lower,23,7,9), which is also an improvement. The P4-consequence of v is (lower,22,6,9), which only equals the bound with reference 'G.' The scoundrel is, however, v's P2-consequence (lower,24,7,10), which equals the original bound u. When *Update2*(u,'T3') is invoked, the reference 'G' is wiped as desired, but the reference 'T3' soon also disappears after being entered correctly. The table shown on the right is what we would have liked. In its zeal to apply all propagation rules at each improvement *Update2* may introduce an obvious two-reference cycle (involving P2 and P3); notice that the bounds are correctly justified, only the associated references will be circular. In order to prevent this, *Update* will have to know something about its calling history, for instance via an additional parameter, but we propose a different solution.

Our approach consists of restoring invariance w.r.t. P2, P3, and P4 nonrecursively prior to the recursive application of other propagation rules. This is fairly simple owing to their composition properties (see section 2.6), and has several advantages. P234 will be used to stand for P2, P3, and/or P4. P234 invariance can be established much more efficiently when done nonrecursively, furthermore it allows P234 composition properties of other propagation rules to be taken into account, thereby further increasing efficiency. This is all we shall do about (ii). Two-reference cycles can also be easily avoided without strengthening the obligations of *Update*, thereby solving (iii). And it enables us to take a fair stand on bounds outside the artificial boundary: to get inside, P234 will be applied repeatedly (composing them nonrecursively), and of the other propagation rules those that decrease the bound's length will be applied recursively; this deals with one of our concerns expressed in (i).

The obligations of this new procedure *Update* have to be formulated carefully, its implementation is a delicate affair. Instead of giving a fully detailed description of *Update* with all assertions required for a correctness proof, we shall discuss some of the problems encountered, and our way of attacking them. *Update* roughly gets the following structure, which can be viewed as a transformed version of the original procedure.

```
     PROCEDURE Update(q: quadruple; r: reference) ;
(a)    { pre-condition: ..., post-condition: ... }
       VAR V: SET OF quadruple ; { see (c) and (e) }
       BEGIN
       V := [] ; { V becomes the empty set }
(b)    IF "q is interesting" THEN BEGIN
(c)      "make improvements on account of P234, seeing to it that
          P234 invariance is finally restored again, thereby
          possibly introducing other variances; add each such
          improvement to V; also take care of the proper
          references, possibly wiping some" ;
(d)      FOR "all non-P234 propagation rules R" DO
(e)        FOR "all quadruples p in V" DO
(f)          IF "R is applicable to p and required"
(g)            THEN Update(R(p),'R') ;
         END { if }
       END { Update } ;
```

Ad (a).
Update's pre-condition is changed to "the table is closed under P234" AND ("*IsBound*(q) on account of r" OR "q lies outside the natural boundaries"), bounds outside the table simply being discarded. The reason for doing so is that q does not belong to the domain of propagation function R if R(q) falls outside the natural boundaries, part of the test "q in dom(R)" can now be deferred to a recursive call on *Update*, so that propagation functions can be applied naively. *Update*'s post-condition is strengthened to "incorporate q" AND "do not introduce new variances" AND "the table is closed under P234." Closure of the table under P234 is now an invariant of the procedure *Update*, this is helpful in (c) and (f).

Ad (b).
Because of (i) *Update* only takes action if "q is within all boundaries" AND "q is an improvement, OR "q is within the natural boundaries, but outside the artificial boundary." In the latter case an attempt will be made to derive bounds closer to or within the artificial boundary. No constructions are considered that first lead away from the artificial boundary, even though it is possible that this would later result in something interesting; these detours can not be done for lack of a good criterium to select a finite number of promising composites (termination requirement).

Ad (c).
Due to the composition properties for P234 (see section 2.6, esp. the closing REMARK) and the assumed closure under P234 (see (a)) it is fairly easy to find all interesting P234* consequences of q (* indicating repetition), but the artificial boundary requires some care. We have chosen the following strategy. P3 is applied zero or more times to q until it is useless, to each of these results P4 is applied zero or more times (until useless). P2 is applied once or more until useless, to each result P4 is again applied repeatedly. In this way all interesting P234* consequences of q (q's P234* area for short) are generated, including

q itself. Each improving P234* consequence is to be added to the set V, so that all remaining variances can later on be removed recursively (see (e)). This set V is an admistration local to each invocation of *Update*. Its representation in PASCAL is somewhat problematic, since sets of records are not allowed. Actually, maintaining a set of triples instead of quadruples suffices, the table can be consulted for the associated distance. Such a set of triples can be represented by its characteristic function. Under (e) we discuss a method of doing this in one global variable; it introduces the possibility of interference between these local administrations, but this turns out to be beneficial.

If a P234* consequence is useless but gives the same bound as in the table, then the associated reference can be wiped (*CondWipeRef* in appendix IV, special care is needed when q lies outside the artificial boundary). P2 is never applied after P3, nor the other way round, so the two-reference cycles mentioned in (ii) will not occur. The reference associated with q will be r, q's P234 consequences get an implicit reference. If, however, q lies outside the artificial boundary, but (some of) its P234* consequences improve the table, then we want an explicit reference somewhere, giving at least a hint about how the improvement arose (see (i)). The current version of the updating algortihm is rather naive in this respect, and puts far too many explicit references on the artificial boundary, superfluous ones being removed manually.

In order to deal with the artificial boundary, it is good to have an idea of the P234* area's anatomy. On all the diagonals generated by P4 the bound is constant (equal to q.d); in P2's direction it is also constant (but possibly differing by one from q.d); in the direction of P3 it is steadily decreasing (increasing) by one on each step for lower (cq. upper) bounds. It is not necessary to generate that part of the P234* area outside the artificial boundary, so in our program is incorporated a straightforward method to skip that part.

Ad (d).
This FOR-loop can be unrolled and each non-P234 propagation rule dealt with separately in an ad hoc fashion.

Ad (e).
All the variances introduced by (c) have their inputs in q's P234* area, as registered in the set V. Previously there was only one such input (viz. q itself), therefore the stacking mechanism kept track of the location of all (inputs of) variances during the recursive calls (g), obviating the set V. We want to represent V without introducing too much overhead, and such that it can be easily traversed. Is it feasible to reconstruct q's P234* area from q alone? Well, q is indeed kept intact by the stacking mechanism, but the table has possibly undergone radical changes as a result of the recursive calls. It would be fairly easy to find an encompassing area; it might, however, be much too large, giving rise to many unneeded applications of propagation rules.

We have chosen to mark q's P234* area in the table while doing (c), this amounts to representing V by its characteristic function. With each bound in the table there is--besides the current bound on the minimum distance--a reference, and a mark. At first one might think of a boolean mark, but the recursive nature of *Update* introduces enough problems to drop this idea (the q's alone do not give enough clues to distinguish between the local V's, which may be adjacent). We use integer marks: an invocation of *Update* marks with its recursion depth (1 and up), 0 indicating unmarked. That overlapping V's cannot be represented in this way is no problem, a variance needs to be removed but once. V's grow only at (c) on the deepest recursion level, which takes precedence in case of overlap. The traversal of the set V at (e) is accomplished by starting from q (itself a member of V), using the P234 rules for stepping, and using the marks to decide when to stop. In a certain sense a P234* area is convex and, hence, connected. With the way we finally implemented marking, it is not necessary to remove the marks afterwards (a selected few beyond the P234* area's boundary are removed while marking), but one restriction is that the P234* area has to be traversed in the same way as it was generated under (c). When q lies outside the artificial boundary, no marking is done. Due to our choice of propagation rules only q and not its entire P234* area is required for the application of rules that decrease the length parameter (and for the removal of variances introduced by q's P234* consequences). The associated reference will be that of the propagation rule, the reference with the bound outside the artificial boundary is lost.

Ad (f).
Applicability of a propagation rule R to bound p should together with (b) cover the test "p in dom(R)." Only undefined computations have to be prevented by (f), like accessing a bound outside the table; resulting in a negative dimension is alright, and is eliminated by (b) one recursion level deeper. The properties of composition (see 2.6) allow us to reduce the number of applications, because *Update* (on the next level) always applies P234 repeatedly. For example, rule A need only be applied to q and not to the other bounds in its P234* area. It is also nice to know that the table is closed under P234 prior to the application (see (a) and (c)), so that even more properties can be used. Invariance w.r.t. Besides, P234 is a more natural state of the table since it expresses certain monotonicity properties. For instance, in the light of the properties for C_m and closure under P234, C_m only has to be applied to q and its P4 consequences, and only for those m that satisfy *odd*(Lb(m-1,k)).

Ad (g).
We have usually put (e), (f) and (g) together in one (or more) ad hoc procedure(s) for each non-P234 rule, the supplied reference in (g) is simply a constant, and is always an explicit reference.

3.3. Generating an Initial Invariant Table

At first we thought of using a trivial initial table (one that is easily generated), and using *Update* to turn it into a table closed under our selection of propagation rules. *Update* as described above requires the table to be closed under P234, so that is also a prerequisite of any initial table. Two trivial tables suggest themselves. The most trivial table has not even incorporated P1: Lb(n,k) = 0, and Ub(n,k) = "infinite," for all n and k. This table is also invariant under all the other propagation rules. It should be turned into an initial table by incorporating P1, which can be done by using *Update* to introduce the bounds: (lower,MaxN,1,MaxN), (lower,MaxN,MaxN,1), and (upper,1,1,1). The representation of "infinite" is troublesome, but the representation for the table as used in the programs makes it even less interesting (see section 4.2), since the bounds on the natural boundary n = k would initially not be equal. Another quite trivial table is:

$$Lb(n,k) = \begin{cases} n & (k = 1) \\ 2 & (1 < k < n) \\ 1 & (k = n) \end{cases} \qquad Ub(n,k) = n-k+1 \quad (1 \leq k \leq n)$$

It is closed under P234, even P1 is incorporated, and it does not have the representation problems of the previous one, but it takes a careful analysis to find out which improvements have to be made for it to become closed under the other propagation rules. Since we did not use this table either, we shall not go through the search for a (minimal) set of improvements by considering all the variances in this table. The reason for discarding this alternative as well is that it takes *Update* an enormous amount of time to compute the closure. This is not so strange because the table allows for a lot of improvement, the point is that each tiny improvement causes an avalanche of other small improvements. *Update* does not wait for the big catch, and consequently is running through a huge number of propagation chains, piling one improvement up another, the final value at a particular location being obtained only after what seems to be the maximal number of minimal improvements.

So we abandoned the idea of a trivial initial table. We would have to invest some effort in the initial table itself, instead of letting *Update* do all the work. We came up with a one-pass dynamic initialization, determining once for each location in the table what would be the best value from the other values that were already available, by applying the constructions at hand. Any variances--when known--can be removed afterwards by *Update*.

The order of this single pass over all locations leaves some freedom. We have chosen for the natural writing order, starting with length equal 1 and working downwards (increasing length), while for each length doing all dimensions in increasing order; all the lower bounds are done first. For each location all bounds with the same length and smaller dimension, and all bounds with

smaller length (any dimension) are available. So, for lower bounds we can consider the constructions P2, C, and D (followed by P3 or P4 if the length is odd); for upper bounds P3, P4, B (partly), and E. This implies invariance of the initial table w.r.t. these constructions. It is not so difficult to prove (by induction on the number of computed entries) that the lower bounds will be closed under P3 and P4 as well, similarly for upper bounds under P2.

While filling in the lower bounds any variances with respect to A can be detected and written to a file, so that they can be subjected to an *Update* afterwards. During the generation of upper bounds variances under B for lower bounds and some upper bounds are detectable in the same way. It turns out, however, that the table generated in this way is immediately closed under our selection of propagation rules, so the separate *Update*-pass is not needed. We have not proved that this is generally the case.

4. Implementation Details

The programming language PASCAL was used to write a bound table maintenance package. This package of programs was implemented in Burroughs Pascal (version 3.4.750) to run on a Burroughs B7900 mainframe computer system. In section 4.1 we discuss the outside of the package, explaining such user aspects as the format of a table printout. Section 4.2 reveals inside details, and serves as additional program documentation. Appendix IV contains the listings of the two most important programs in the package.

4.1 Interpreting a Table (and other user aspects)

Since the table is fairly large, it is impossible to find out by simple visual inspection what has and what has not been changed as result of a series of updates. As an aid to the interpretation of an updated table, some statistics are maintained, and during each update session a report is generated, indicating major events. These events include: d_{max} completely known for a particular parameter pair, along an entire column, or along an entire row; previous bound with external reference has been indirectly equalled (the reference is wiped out), has been superseded. See Appendix III for the report generated while producing the updated table.

Each new bound is checked before it is incorporated in the table. This is done to safeguard the table's integrity, because it is next to impossible to undo a faulty update. It is always possible, of course, to corrupt the table by feeding invalid but likely bounds. Ideally *IsBound* should hold before *AssignBound* is done, but in its stead we enforce the more computable NOT *IsViolation*. In case of a violation the program is aborted if there is reason to believe the internal table to be corrupted (i.e. if the table was modified since the outermost invocation of the *Update* that caused the violation), otherwise the violating bound is discarded.

```
FUNCTION IsViolation(q: quadruple): boolean ;
  { Is bound quadruple q in conflict with the other bounds ?
  }
  BEGIN
  WITH q, t DO CASE b OF
    lower: IsViolation := (d > Ub(n,k)) ;
    upper: IsViolation := (d < Lb(n,k)) ;
    END { case }
  END { IsViolation } ;
```

The table is partitioned in blocks for printing. Each block has 25 (or 27) rows and 20 columns of bound pairs. This is about the maximum amount that fits on one page of our lineprinter (66 lines of 132 columns), when a block has been formatted with horizontal and vertical separating lines every 5 bound pairs and each

bound is accompanied by its reference. After an update session there is an option to print all modified blocks with modifications "highlighted." A block is identified by a pair of block coordinates, these are shown in the upper and lower right hand corners. The code's length is plotted vertically, its dimension horizontally.

A bound pair is printed as follows. If the lower and upper bound are equal then d_{max} is known, its value is shown centered; otherwise both lower and upper bound are printed separated by a dash (-). A reference consists of a two-character string, an implicit reference is shown as two blanks. The reference associated with the lower bound is shown over the bound's value (and possibly a bit to the left); the upper bound's reference is similarly shown as a (right-hand) superscript. Between these references there is a high-lighting symbol, a blank indicates that neither bound was modified during the update session; a less-than (greater-than) sign (<,> think of arrows) indicates that only the lower (upper) bound was modified; an asterisk (*) means that both bounds were improved.

How can the references in a table of bounds be interpreted? An external reference (an explicit reference other than A through E) directly refers to the REFERENCE section. Keep in mind that any reference on the artificial boundary (n = 127) may have been produced by a P234* consequence of a bound outside that boundary (see section 3.2 (c)). The list below should help in interpreting internal references. Notice that such a reference does not always uniquely determine the other bound(s) involved, furthermore, it may require a search to find them. An implicit reference--one of P1 through P4--has to be reconstructed since it was omitted from the table. We now give a recipe for the reconstruction of an implicit reference. It is not unique, indeed that was one of the reasons for their omission. If the bound lies on a natural boundary (n = 1, or n = k), its missing reference is P1. Otherwise, first try P4, next try P3 (if the bound is odd it must work, since P2 always results in an even bound), then try P2; take whichever worked first. If none worked, then the table is not closed under P234.

Once the internal reference is known it can be traced to other bounds. If this tracing is repeated--by taking any of the other bounds involved as new bound--it should always lead to P1, an explicit reference, or outside the artificial boundary. If it does not, then there seems to be something amiss with the references. In fact, the tracing process can only cycle for ever in that case, so there is a reference cycle. In general, these are hard to avoid, but in practice they have--as far as we know--not appeared in our tables. For an example of a reference cycle try to trace the bound (upper,83,40,20) in the table in [A], actually the explicit reference 'K' is missing there. In our table such a two-reference cycle cannot occur, implicit references can always be traced to P1 or an explicit reference.

Internal reference at (lower,n,k)	Locations of bounds referred to
P1	none
P2	(lower,n-1,k)
P3	(lower,n+1,k)
P4	(lower,n+1,k+1)
A	(lower,n+2*Lb(n,k),k+1)
B	(lower,n+s,k+s-1), (upper,n+s,n-k+1) for some s with s ≥ Ub(n+s,n-k+1)
C	(lower,m,k), (lower,n-m,k) for some m ≥ k
D	(lower,*ceil*(n/2),j), (lower,*ceil*(n/2),k-j) for some j < n

at (upper,n,k)	
P1	none
P2	(upper,n+1,k)
P3	(upper,n-1,k)
P4	(upper,n-1,k-1)
E	(upper,n-Ub(n,k)-1,k-1)
B	(upper,n,n-k), (upper,n-s,k-s+1) where s = Ub(n,n-k)
C	(lower,m,k), (upper,m+n,k) for some m

Examples of interpretation based on the table in Appendix I

```
(upper, 12,  5, 4) H: see [H] for nonexistence of [12,5,5]
(lower,127, 16,51) We: punctured [128,16,52] from [We]

(lower, 38,  9,15) A: residual code of [68,10,30]
(lower, 85,  8,40) B: construction B using
                      [89,11,40] & (upper,89,78,4)
(lower, 24,  3,13) C: concatenation of [ 6,3,3] & [18,3,10],
                                    or of [10,3,5] & [14,3, 8],
                                    or of [11,3,6] & [13,3, 7]
(lower, 90,  7,44) D: (u,u+v) on [45,1,45] & [45,6,22]

(upper, 65,  8,30) E: Griesmer bound using (upper,34,7,15)
(upper, 18, 10, 4) B: via construction B using
                      (upper,18,8,6) & (upper,12,5,4)

(lower, 63, 12,24): recipe (repeated) leads to [66,18,23] O,
                    equally good are [66,14,25] L, and
                    [63,16,23] I (their P234* areas overlap)
(upper, 23, 12, 7): recipe (repeated) leads to (upper,16,6,6)
                    E, equally good is (upper,18,10,4) B
```

4.2 Inside Details of Programs

The programs *BoundUpdater* and *InitialBt* are listed in Appendix IV. A table of bounds is stored in two ways: during program execution it is in an (internal) ARRAY (TYPE BtArray), for more permanent storage a FILE is used (TYPE BtFile). *BoundUpdater* reads in a BtFile, prompts via standard OUTPUT, reads commands

via standard INPUT, performs the specified operations on its internal copy of the table, and afterwards writes it out to a BtFile; it also produces a report via the FILE Report and a list of applied updates via the FILE List; table printout is via the FILE Pr. *InitialBt* generates a BtFile and a list of improvements on it, that can be used by *BoundUpdater* to obtain a closed table of bounds. *BoundUpdater* interprets commands one line at a time, the first non-blank on the line determines the type of operation requested, the rest of the line constitutes a parameter list. The following commands are available:

L: incorporate specified Lower bound and restore invariance
U: idem for Upper bound
S: Save internal table on file and print statistics
P: Print the block with specified coordinates
A: print All blocks
M: print all Modified-but-not-yet-printed blocks
W: print all blocks in Wallpaper format
Q: do S, finish off, and Quit program

The limitiations of PASCAL and the limited resources available on the computer system running the software compelled us to do some things in a less straightforward way. In what way this influenced the design will be discussed next, along with other aspects that are in want of explanation. To begin with, the source text of *BoundUpdater* is divided in three parts, viz. BoundUpdater, UpdateBt, and PrintBt; the latter two are textually included in the first by the compiler. With each bound in the table there is associated a two-character reference, an integer mark, and a boolean indicating whether the bound was changed during the update session (see the TYPE BtElement). The latter two are only used during program execution, but are nonetheless saved on the BtFile as well. Notice that the reference is not of TYPE Reference, for otherwise BtElement would not be packed in one word (6 bytes) because of alignment restrictions.

The TYPEs triple and quadruple were not used since PASCAL lacks a RECORD constructor and does not allow a RECORD to be returned as function value. Hence the function *Bound* becomes useless, and was accordingly eliminated. The function *IsBound* is only a conceptual function, never intended to be put in the program. The procedure *AssignBound* does not explicitly occur by that name, it was swallowed by *MakeImprovement*, a procedure that also does some other things besides modifying the bound (e.g., checking for violation, updating the statistics, and reporting some major events).

PASCAL does not know triangular arrays, and two square arrays each half-filled require too much space (for $n < 128$, there are 16256 bounds, each occupying one word (6 bytes) of storage, that is about 92K bytes; doubling this would go too far). We decided to put upper and lower bounds in one square array Bt, lower bounds in the lower left triangle, upper bounds in the upper right triangle. That is, for $1 \leq k \leq n$ we have Bt[n,k].dist = Lb(n,k), and Bt[k,n].dist = Ub(n,k). The diagonal

is shared by both triangles, this is possible since the value of d_{max} is known on the boundary n = k (viz. equal 1), so upper and lower bounds are the same. This sharing does call for extra care when using the array Bt.

Several nonstandard PASCAL features have been used. These will now be listed accompanied by a short comment.

Compiler Directives: a line starting with a dollar sign ($) is an instruction for the compiler, and not part of the program as such; these are used to enable or disable certain compile time options (like listing or range checking), and such things as conditional compilation ($ SET OMIT = and $ POP OMIT) and source file inclusion ($ INCLUDE).

Identifiers: upper and lower case are equivalent; all characters are significant; may contain (significant) underscores.

PACKED: our program relies on effective packing; without it the table would increase in size appreciably; this is of course implementation dependent, the (?) Pascal standard does not prescribe anything on packing.

FILE: associated with a file are a number of attributes (like its title if it is a disk resident file), the values of which are determined by: defaults, the program heading, file attribute equations at program invocation, and special procedures (see below).

STRING(n): predefined type; a variable length string of at most n characters.

MIN, MAX: functions with a variable number of parameters all of the same scalar type, returning the minimum resp. maximum.

RUNTIME: parameterless function returning the amount of process time (in seconds) elapsed since the program started; result is of type real.

IORES: pseudo function that can serve as a constant indicating an I/O result; its actual parameter is restricted to a number of identifiers that only have a special meaning in this context (like Ok and DataErr).

ABORT: parameterless procedure that immediately aborts the program's execution.

GETATTRIBUTE, SETATTRIBUTE: procedures to inspect and modify file attributes; they have a special syntax.

CLOSE: procedure to close a file; its first parameter is the file to be closed; its second parameter is optional and indicates a special action, it is restricted to a number of identifiers that only have this special meaning in this context (like SAVE and CRUNCH).

READ: when used as procedure an I/O error aborts the program; can optionally be used as a function that returns an integer indicating the I/O result without aborting.

OTHERWISE: an extension of the CASE statement; this reserved word is not followed by a colon; it is syntactically the last alternative; it can be followed by a statement list that will be terminated by the standard END; this alternative is semantically chosen iff none of the others apply.

5. Concluding Remarks

We used our bound table maintenance package to reconstruct Helgert and Stinaff's table in [A], which was to serve as the starting point for a more up to date table of bounds. For the reconstruction we extracted all bounds from their table that had external references (i.e. 'F' through 'Z', see Appendix II), and used these to update our initial invariant table (which is based solely on P1 and all possible constructions by P2, P3, P4, and A through E). We did not recheck the correctness of all these externally determined bounds, but we could not trace their lower bound 26 on $d_{max}(59,8)$, which has reference 'U.' It is the only 'U' in their table, and it refers to construction Y1, which is the basis of the general construction B; it might be that there is a [63,11,26]-code with a dual code of type [63,52,4] (we know $4 \leq d_{max}(63,52) \leq 5$, but that is not enough). Two external references are missing in their table: the lower bound 23 on $d_{max}(127,57)$ is on account of 'I,' and the upper bound 20 on $d_{max}(83,40)$ is from 'K.'

Our reconstruction of the table in [A] revealed some of its shortcomings. First, our program showed that the table in [A] is not closed under any of the constructions A through E, substantiating the concluding remark in [A] that the effects of B had not been fully evaluated. Using the same data, the program found the following (internal) improvements, written as (bound kind, length, dimension, bound on distance); their P2, P3 and P4 consequences are not mentioned, (**) indicates a second order improvement:

(lower, 38, 9,15) A	(upper, 88,16,36) B
(upper, 62,12,26) E	(upper, 91,16,38) B
(upper, 66, 9,30) E	(upper, 91,25,32) B (**)
(upper, 68,17,26) E	(upper, 94,25,34) B (**)
(upper, 71,13,30) E	(upper, 98,25,36) B (**)
(lower, 72,14,28) A	(lower, 98,31,23) D
(upper, 76,24,26) E	(upper,101,25,38) B (**)
(lower, 79, 7,37) C	(upper,125,25,50) E
(upper, 81,16,32) B	(upper,125,36,44) E
(lower, 82, 7,39) C	(upper,127,10,60) E
(upper, 84,16,34) B	

Second, there are some minor differences in the references, that were associated with each bound. Most remarkable are: (a) the omission of the superfluous references 'C' and 'I' for the lower bounds on resp. $d_{max}(105,18)$ and $d_{max}(31,21)$ (both bounds are also justifiable by P3 and P4, going back to resp. (lower,117,20,43) 'X,' and (lower,33,22,6) 'N'); (b) the avoidance of external justification for a number of bounds (see list below: the lower bounds were justified by D and P3; the upper bound by E). It must be noted that a different order of incorporating external improvements will, generally, result in different references, so (b) is not very telling.

(lower, 7, 4, 3) I	(lower, 63,57, 3) I
(lower, 15,11, 3) I	(lower,115,12,47) X
(lower, 31, 6,15) I	(lower,119, 7,59) F
(lower, 31,26, 3) I	(lower,127, 8,63) I
(lower, 63, 7,31) I	(lower,127,120,3) I
(upper, 63, 9,28) J	

In our initial invariant table there is no contribution to lower bounds by constructions A and B, nor to upper bounds by construction B with $2k < n$. As a result the initial table generated in one pass from top to bottom (increasing length), and left to right (increasing dimension) is immediately invariant. We have no proof for this, it just happened to be the case.

The new table of upper and lower bounds on the minimum-distance of binary linear codes, as it appears in appendix I, is based on many improvements found in the literature (see REFERENCES and Appendix II). The updates on the table were made in chronological order of appearance of the improvement. The results in [Su] were improved by those in [PT], the reference 'Su,' therefore, disappeared again after a short while. It was impossible to give full credit to all researchers involved, many results were found independently. At least one source is mentioned for each improvement incorporated. It appeared that some results claimed to be new, were derivable from older improvements by constructions P2 thru E, this may explain some unexpected or missing references.

Most noticeable in the updated table is that d_{max} has been completely determined for dimensions 6 and 7 (due to [T3]). The new table is the result of 89 single updates, of which about 80 lasted. This resulted in 110 additional parameter pairs for which d_{max} is now known. It is remarkable that (roughly) in the range $8 \leq k \leq 20$, $d_{max}(n,k)$ is most often known, when its value is a multiple of 4 (ranging from 8 to 40). For comparison we give the

Computing times (in seconds on Burroughs B7900)
- 3 for initial invariant table
- 264 for reconstruction of table in [A]
- 19 for updating table (includes conversion for print-out)

Now that the table of bounds is available in computer readable form, it can be used for other purposes than automatic updating and printing.

Acknowledgments

Chris Vos did most of the searching through the available literature of the past ten years. Henk van Tilborg dug up some interesting articles from his files, that might otherwise have been overlooked. Cees Hemerik suggested some improvements in the presentation of the programs, any imperfections, however, fall under my responsibility.

REFERENCES

[A] H.J. Helgert and R.D. Stinaff, "Minimum-distance bounds for binary linear codes," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 344-356, May 1973.
The Helgert-Stinaff residual code construction is also discussed in [MS] p.593.

[B] Dual code construction Y1 (see [M], and [MS] p.592), followed by repeated shortening.

[C] Concatenation (see [MS] p.76).

[D] The u,u+v construction (see [MS] p.76).

[MS] F.J. MacWilliams and N.J.A. Sloane, *The theory of error-correcting codes*, Amsterdam-New York-Oxford: North-Holland, 1977.

[P2] Parity check (see [MS] p.27).

[P3] Puncturing (see [MS] p.28).

[P4] Shortening (see [MS] p.29).

Selected references from [A]:

[Bk] E.R. Berlekamp, *Algebraic coding theory*, New York: McGraw-Hill, 1968.

[E] J.H. Griesmer, "A bound for error-correcting codes," *IBM J. Res. Develop.*, vol. 4, pp. 532-542, 1960.

[F] Griesmer codes, see [E].

[G] T.J. Wagner, "A remark concerning the minimum distance of binary group codes," *IEEE Trans. Inform. Theory*, vol. IT-11, p. 458, July 1965.

[H] A.B. Fontaine and W.W. Peterson, "Group code equivalence and optimum codes," *IEEE Trans. Inform. Theory*, vol. IT-5, pp. 60-70, May 1959.

[I] Primitive BCH codes: W.W. Peterson, *Error-correcting codes*, pp. 166-167, Cambridge, Mass.: M.I.T. Press, 1961.

[J] Improvement on Johnson bound: S.M. Johnson, private communication, see [A] [27].

[K] S.M. Johnson, "On upper bounds for unrestricted binary error-correcting codes," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 466-478, July 1978.

[L] M. Karlin, "New binary coding results by circulants," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 81-92, Jan. 1969.

[M] N.J.A. Sloane, S.M. Reddy, and C.L. Chen, "New binary codes," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 503-510, July 1972.

[N] C.L. Chen, "Computer results on the minimum distance of some binary cyclic codes," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 359-360, May 1970.

[O] Adryanov-Saskovets construction: [Bk] p. 333.

[P] T.J.Wagner, "A searching technique for quasi-perfect codes," *Inform. Contr.*, vol. 9, pp. 94-99, 1966.

[Q] H.J. Helgert, "Srivastava codes," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 292-297, Mar. 1972.

[R] Improvement on Johnson bound: R.J. McEliece and L.R. Welch, private communication, see [A] [29].

[S] Cyclic codes: [Bk] p. 433.

[T] J.M. Goethals, "Analysis of weight distribution in binary cyclic codes," *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 401-402, July 1966.
[U] (?) Construction Y1, see [M].
[V] T. Kasami, S. Lin, and W.W. Peterson, "Polynomial codes," *IEEE Trans. Inform. Theory*, IT-14, pp. 807-814, Nov. 1969.
[W] T. Kasami and N. Tokura, "Some remarks on BCH bounds and minimum weights of binary primitive BCH codes," *IEEE Trans. Inform. Theory*, vol IT-15, pp. 408-413, May 1969.
[X] H.J. Helgert and R.D. Stinaff, "Shortened BCH codes," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 818-820, Nov. 1973.
[Y] Rao code and shortened Rao code.
[Z] M. Karlin, private communication, see [A] [30].

Additional references:

[Al] W.O. Alltop, "Binary codes with improved minimum weights," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 241-243 Mar. 1976.
[BM] L.D. Baumert and R.J. McEliece, "A note on the Griesmer bound," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 134-135, Jan. 1973.
[Bv] B.I. Belov, "A conjecture on the Griesmer bound," *Optimization Methods and their Applications* (Russian), 1974.
[Fa] P.G. Farrell, "An introduction to anticodes," *CISM Summer School: Algebraic coding theory and applications*, ed. Longo, 1978.
[Ha] A.A. Hashim, "Some new results on binary linear block codes," *Electronics Letters*, vol. 10, pp. 31-33, Feb. 1974.
[He] T. Helleseth and H. van Tilborg, "A new class of codes meeting the Griesmer bound," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 548-555, Sep. 1981.
[Ka] M. Kasahara e.a., "A new class of binary codes constructed on the basic of BCH codes," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 582-585, Sep. 1975.
[Lc] V.W. Logacev, "An improvement on the Griesmer bound in the case of small code distances," *Optimization Methods and their Applications* (Russian), 1974.
[Pi] P. Piret, "Good linear codes of lengths 27 and 28," *IEEE Trans. Inform. Theory*, vol IT-26, p. 227, Mar. 1980.
[PT] G. Promhouse and S. Tavares, "The minimum distance of all binary cyclic codes of odd lengths from 69 to 99," *IEEE Trans. Inform. Theory*, vol IT-24, pp. 438-442, July 1978.
[Pu] C.L.M. van Pul, "On bounds on codes," Master's Thesis, p. 236, *Dept. of Math. and Comp. Sc., Univ. of Techn., Eindhoven*, Aug. 1982.
[Ro] G. Roelofsen, "On Goppa and generalised Srivastava codes," Master's Thesis, p. 41, *Dept. of Math. and Comp. Sc., Univ. of Techn., Eindhoven*, Aug. 1982.
[SS] G. Solomon and J.J. Stiffler, "Algebraicly punctured cyclic codes," *Inform. Control*, vol. 8, pp. 170-179, 1965.
[Su] Y. Sugiyama e.a., "Some efficient binary codes constructed using Srivastava codes," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 581-582, Sep. 1975.

[T1] H. van Tilborg, "On quasi-cyclic codes with rate 1/m," *IEEE Trans. Inform. Theory*, vol IT-24, pp.628-630, Sep. 1978.
[T2] H. van Tilborg, "On the uniqueness resp. non-existence of certain codes meeting the Griesmer bound," *Inform. Control*, vol. 44, pp. 16-35, 1980.
[T3] H. van Tilborg, "The smallest lengths of binary 7-dim. linear codes with prescribed minimum distance," *Discr. Math.*, vol. 33, pp. 197-207, 1981.
[We] L. Weng, "Concatenated codes with large minimum distance," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 613-615, Sep. 1977.
[Wi] J.A. Wiseman, "Constructing nonlinear codes," *Inform. and Control*, vol. 45, pp. 70-79, 1981.

INDEX

Upper and Lower Bounds on $d_{max}(n,k)$ for Binary Linear Codes

n,k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	0,0
1	1																				1
2	2	1																			2
3	3	2	1																		3
4	4	E 2	2	1																	4
5	5	3	2	2	1																5
6	6	4	3	2	2	1															6
7	7	E 4	4	3	2	2	1														7
8	8	C 5	4	D 4	B 2	2	2	1													8
9	9	6	E 4	4	3	2	2	2	1												9
10	10	E 6	5	4	4	3	2	2	2	1											10
11	11	C 7	6	5	4	4	3	2	2	2	1										11
12	12	8	E 6	D 6	H 4	4	4	3	2	2	2	1									12
13	13	E 8	7	6	5	4	4	4	3	2	2	2	1								13
14	14	C 9	8	7	6	5	4	4	4	3	2	2	2	1							14
15	15	10	8	8	7	6	5	4	4	4	3	2	2	2	1						15
16	16	E 10	E 8	8	D 8	E 6	6	5	4	4	D 4	B 2	2	2	2	1					16
17	17	C 11	C 9	E 8	8	7	6	6	N 5	4	4	3	2	2	2	2	1				17
18	18	E 12	E 10	8	8	8	7	6	6	B 4	4	4	3	2	2	2	2	1			18
19	19	C 12	C 10	9	F 8	8	8	7	6	5	4	4	4	3	2	2	2	2	1		19
20	20	13	11	10	9	8	8	8	7	6	5	4	4	4	3	2	2	2	2	1	20
21	21	E 14	12	E 10	10	E 8	8	8	8	7	6	5	4	4	4	3	2	2	2	2	21
22	22	E 14	12	11	10	9	8	8	8	8	7	6	5	4	4	4	3	2	2	2	22
23	23	C 15	E 12	12	11	10	9	8	8	8	8	N 7	6	G 5	4	4	4	3	2	2	23
24	24	16	C 13	12	D 12	E 10	T3 10	8-9	8	8	8	8	E 6	6	4-5	4	4	4	3	2	24
25	25	E 16	14	E 12	12	11	10	9-10	8-9	8	8	8	6-7	6	5-6	4-5	4	4	4	3	25
n,k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	0,0

n,k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1,0
26	26	17	14	13	12	12	11	10	9-10	8-9	8	8	7-8	6-7	6	5-6	4-5	4	4	4	26
27	27	18	15	14	13	12	12	10-11	10	9-10	8	8	8	7-8	6	6	5-6	4-5	4	4	27
28	28	18	16	14	14	12	12	11-12	10-11	10	8-9	8	8	8	6-7	6	6	5-6	4-5	4	28
29	29	19	16	15	14	13	12	12	11-12	10-11	9-10	8-9	8	8	7-8	6-7	6	6	5-6	4-5	29
30	30	20	16	16	15	14	12	12	12	11-12	10	9-10	8-9	8	8	7-8	6-7	6	6	5-6	30
31	31	20	17	16	16	15	13	12	12	12	11	10	8-10	8-9	8	8	6-8	6-7	6	6	31
32	32	21	18	16	16	16	14	12-13	12	12	12	10-11	9-10	8-10	8-9	8	7-8	6-8	6-7	6	32
33	33	22	18	16	16	16	14	12-14	12-13	12	12	10-12	10-11	9-10	8-10	8-9	8	7-8	6-8	6-7	33
34	34	22	19	17	16	16	15	13-14	12-14	12-13	12	11-12	10-12	10-11	9-10	8-10	8-9	8	7-8	6-8	34
35	35	23	20	18	16	16	16	14-15	13-14	12-14	12-13	12	11-12	10-12	10-11	8-10	8-10	8-9	8	7-8	35
36	36	24	20	18	17	16	16	14-16	14-15	13-14	12-14	12	12	10-12	10-12	9-11	8-10	8-10	8-9	8	36
37	37	24	20	19	18	17	16	15-16	14-16	14-15	13-14	12-13	12	11-12	10-12	10-12	9-11	8-10	8-10	8	37
38	38	25	21	20	18	18	16	16	15-16	14-16	14-15	12-14	12-13	12	11-12	10-12	10-12	9-10	8-10	8-9	38
39	39	26	22	20	19	18	17	16	16	14-16	14-16	12-14	12-14	12-13	12	11-12	10-12	10-11	9-10	8-10	39
40	40	26	22	20	20	18	18	16-17	16	15-16	14-16	13-15	12-14	12-14	12-13	12	11-12	10-12	10-11	9-10	40
41	41	27	23	21	20	19	18	16-18	16-17	16	15-16	14-16	13-15	12-14	12-14	12-13	12	11-12	10-12	10-11	41
42	42	28	24	22	20	20	19	16-18	16-18	16	16	14-16	14-16	13-15	12-14	12-14	12-13	12	11-12	10-12	42
43	43	28	24	22	21	20	20	17-19	16-18	16-17	16	14-16	14-16	14-16	13-15	12-14	12-14	12-13	12	11-12	43
44	44	29	24	23	22	21	20	18-20	17-18	16-18	16-17	15-16	14-16	14-16	14-16	12-14	12-14	12-14	12-13	12	44
45	45	30	25	24	22	22	20	18-20	18-19	17-18	16-18	16-17	15-16	14-16	14-16	12-15	12-14	12-14	12-14	12-13	45
46	46	30	26	24	23	22	21	19-20	18-20	18-19	16-18	16-18	16-17	15-16	14-16	12-16	12-15	12-14	12-14	12-14	46
47	47	31	26	24	24	23	22	20-21	19-20	18-20	16-19	16-18	16-18	16-17	15-16	13-16	12-16	12-15	12-14	12-14	47
48	48	32	27	24	24	24	22	21-22	20-21	19-20	17-20	16-19	16-18	16-18	16-17	14-16	13-16	12-16	12-15	12-14	48
49	49	32	28	25	24	24	23	22	20-22	20	18-20	17-20	16-19	16-18	16-18	15-17	14-16	13-16	12-16	12-15	49
50	50	33	28	26	24	24	24	23	20-22	20-21	19-20	18-20	16-20	16-18	16-18	16-18	15-16	14-16	13-16	12-16	50
n,k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1,0

n,k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	2,0
51	51	34	28	26	25	24	24	24	21-23	20-22	20-21	19-20	16-20	16-19	16-18	16-18	16-17	14-16	14-16	12-16	51
52	52	34	29	27	26	25	24	24	22-24	21-22	20-22	20-21	17-20	16-20	16-19	16-18	16-18	15-17	14-16	13-16	52
53	53	35	30	28	26	26	24	24	22-24	22-23	20-22	20-22	18-21	17-20	16-20	16-19	16-18	16-18	15-17	14-16	53
54	54	36	30	28	27	26	24	24	22-24	22-24	20-23	20-22	18-22	18-21	16-20	16-20	16-19	16-18	16-18	15-17	54
55	55	36	31	28	28	27	25	24	23-24	22-24	21-24	20-23	18-22	18-22	17-21	16-20	16-20	16-19	16-18	16-18	55
56	56	37	32	29	28	28	26	24-25	24	23-24	22-24	20-24	19-23	18-22	18-22	17-21	16-20	16-20	16-19	16-18	56
57	57	38	32	30	28	28	26	24-26	24-25	24	23-24	21-24	20-24	19-23	18-22	18-22	16-20	16-20	16-20	16-19	57
58	58	38	32	30	29	28	27	25-26	24-26	24	24	22-24	21-24	20-24	19-22	18-22	17-21	16-20	16-20	16-20	58
59	59	39	33	31	30	29	28	26-27	24-26	24-25	24	23-24	22-24	21-24	20-23	19-22	18-22	17-21	16-20	16-20	59
60	60	40	34	32	30	30	28	26-28	25-27	24-26	24-25	24	23-24	22-24	21-24	20-23	19-22	18-22	17-21	16-20	60
61	61	40	34	32	31	30	29	27-28	26-28	25-26	24-26	24-25	24	23-24	22-24	21-24	20-23	19-22	18-22	17-21	61
62	62	41	35	32	32	31	30	28	27-28	26-27	25-26	24-26	24-25	24	23-24	22-24	21-24	20-23	18-22	18-22	62
63	63	42	36	32	32	32	31	28-29	28	27-28	25-27	24-26	24-26	24-25	24	23-24	22-24	21-24	19-23	18-22	63
64	64	42	36	33	32	32	32	28-30	28-29	28	25-28	25-27	24-26	24-26	24	24	22-24	22-24	20-24	18-23	64
65	65	43	36	34	32	32	32	29-30	28-30	28-29	25-28	26-28	25-27	24-26	24-25	24	23-24	22-24	20-24	19-24	65
66	66	44	37	34	32	32	32	30-31	29-30	28-30	25-28	26-28	26-28	25-27	24-26	24-25	24	23-24	20-24	20-24	66
67	67	44	38	35	33	32	32	30-32	30-31	29-30	25-29	26-28	26-28	26-28	24-26	24-26	24-25	24	20-24	20-24	67
68	68	45	38	36	34	32	32	31-32	30-32	30-31	27-30	26-29	26-28	26-28	24-27	24-26	24-26	24-25	20-24	20-24	68
69	69	46	39	36	34	33	32	32	31-32	30-32	28-30	27-30	26-29	26-28	25-28	24-27	24-26	24-26	20-25	20-24	69
70	70	46	40	36	35	34	33	32	32	30-32	28-31	28-30	27-30	26-29	26-28	25-28	24-27	24-26	20-26	20-25	70
71	71	47	40	37	36	34	34	32	32	30-32	28-32	28-31	28-30	27-30	26-29	26-28	24-28	24-27	20-26	20-26	71
72	72	48	40	38	36	35	34	32-33	32	30-32	28-32	28-32	28-31	28-30	26-30	26-29	25-28	24-28	21-27	20-26	72
73	73	48	41	38	36	36	34	32-34	32-33	30-32	28-32	28-32	28-32	28-31	27-30	26-30	26-28	24-28	22-28	21-27	73
74	74	49	42	39	37	36	35	32-34	32-34	30-33	29-32	28-32	28-32	28-32	28-31	27-30	26-29	24-28	22-28	22-28	74
75	75	50	42	40	38	36	36	33-35	32-34	31-34	30-32	29-32	28-32	28-32	28-32	28-30	26-30	24-29	22-28	22-28	75
n,k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	2,0

n,k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	3,0
76	76	E 50 C	43	40	E 38	37	36	34-36	Ka 33-35	32-34	31-33	30-32	29-32	28-32	28-32	28-31	Ka 27-30	24-30	23-29	22-28	76
77	77	C 51	44	40	C 39	C 38	E 36	34-36	34-36	B 32-34	32-34	31-33	30-32	29-32	28-32	28-32	28-31	24-30	24-30	23-29	77
78	78	52	E 44	E 40	40	E 38	Fa 37	35-36	34-36	32-35	32-34	32-34	31-33	30-32	28-32	28-32	28-32	24-31	24-30	24-30	78
79	79	E 52	C 44	41	40	C 39	38	36-37	Pu 35-36	32-36	32-35	32-34	32-34	31-33	28-32	28-32	28-32	24-32	24-31	24-30	79
80	80	C 53	C 45	42	40	40	E 38	T1 37-38	36-37	32-36	32-36	32-35	32-34	Pu 32-34	Ka 29-33	28-32	28-32	25-32	24-32	24-31	80
81	81	54	E 46	E 42	E 40	40	Fa 39	38	36-38	B 33-36	32-36	32-36	32-35	32-34	30-34	B 28-32	28-32	26-32	25-32	24-32	81
82	82	E 54	E 46	C 43	C 41	40	40	38-39	36-38	34-37	33-36	32-36	32-36	32-35	30-34	29-33	28-32	27-32	26-32	25-32	82
83	83	C 55	C 47	44	42	E 40	40	38-40	36-39	35-38	34-37	32-36	32-36	32-36	B 31-34	30-34	28-33	28-32	27-32	26-32	83
84	84	56	48	E 44	E 42	C 41	40	B 39-40	37-40	B 36-38	35-38	PT 33-37	32-36	32-36	32-35	X 31-34	X 28-34	28-33	28-32	PT 27-32	84
85	85	E 56	48	44	C 43	42	40	40	38-40	37-39	36-38	34-38	32-37	32-36	32-36	X 32-35	29-34	28-34	28-33	PT 28-32	85
86	86	C 57	E 48	C 45	44	E 42	T2 41	40	39-40	38-40	37-39	34-38	32-38	32-37	B 32-36	32-36	30-35	28-34	28-34	28-33	86
87	87	C 58	C 49	46	44	C 43	Al 42	40-41	40	39-40	38-40	35-39	32-38	32-38	32-36	32-36	30-36	28-35	28-34	28-34	87
88	88	E 58	E 50	E 46	E 44	44	42	40-42	40-41	40	39-40	We 36-40	33-39	32-38	32-37	32-36	30-36	29-36	28-35	28-34	88
89	89	C 59	E 50	C 47	C 45	44	T2 43	40-42	40-42	B 40	S 40	36-40	34-40	32-39	32-38	32-37	31-36	30-36	29-36	28-35	89
90	90	C 60	51	48	46	E 44	D 44	41-43	40-42	40-41	40	36-40	35-40	32-40	B 32-38	32-38	32-37	31-36	30-36	29-36	90
91	91	E 60	52	48	E 46	45	44	42-44	40-43	40-42	40-41	36-40	T 36-40	32-40	32-39	32-38	32-38	32-37	31-36	30-36	91
92	92	C 61	52	48	C 47	46	Bv 45	43-44	40-44	B 40-42	40-42	36-41	36-40	32-40	32-40	32-39	32-38	32-38	32-37	31-36	92
93	93	E 62	E 52	E 48	48	E 46	46	Al E 44	40-44	40-43	40-42	37-42	36-41	32-40	32-40	32-40	32-39	32-38	32-38	PT 32-37	93
94	94	E 62	C 53	C 49	48	47	46	44-45	Pu 41-44	40-44	40-43	38-42	36-42	C 33-41	32-40	32-40	32-40	32-39	32-38	32-38	94
95	95	C 63	54	50	48	48	47	45-46	42-45	40-44	40-44	39-43	36-42	34-42	B 32-40	32-40	32-40	32-40	32-39	32-38	95
96	96	64	E 54	E 50	48	D 48	48	T1 E 46	42-46	40-45	40-44	We 40-44	36-43	34-42	33-41	32-40	32-40	32-40	32-40	32-39	96
97	97	E 64	C 55	C 51	E 48	48	48	46-47	43-46	40-46	B 40-44	40-44	36-44	35-43	34-42	33-41	32-40	32-40	32-40	32-40	97
98	98	C 65	56	52	C 49	48	48	46-48	44-47	C 41-46	40-45	40-44	37-44	36-44	B 35-42	34-42	33-41	32-40	32-40	32-40	98
99	99	66	E 56	E 52	E 50	E 48	48	47-48	45-48	42-47	40-46	40-45	38-44	37-44	36-43	35-42	34-42	33-41	32-40	32-40	99
100	100	E 66	56	52	50	49	48	D 48	46-48	42-48	B 41-46	40-46	39-45	38-44	37-44	36-43	35-42	34-42	33-41	32-40	100
n,k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	3,0

n,k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	4,0
101	101	C 67	C 57	C 53	C 51	F 50	49	48	47-48	43-48	42-47	41-46	40-46	39-45	38-44	37-44	36-43	35-42	34-42	33-41	101
102	102	E 68	E 58	E 54	52	E 50	50	E 48	D 48	44-48	43-48	42-47	40-46	40-46	39-45	B 58-44	36-44	36-43	35-42	34-42	102
103	103	C 68	C 58	C 54	E 52	51	50	48	48	44-48	44-48	We 43-48	40-47	40-46	40-46	We 39-44	36-44	36-44	36-43	We 35-42	103
104	104	69	59	55	C 52	52	SS 51	48-49	48	44-48	44-48	44-48	40-48	40-47	40-46	40-45	36-44	36-44	36-44	36-43	104
105	105	70	60	56	53	52	52	48-50	48-49	44-48	44-48	44-48	40-48	40-48	40-47	40-46	36-45	36-44	36-44	36-44	105
106	106	E 70	60	56	54	E 52	52	E 48-50	48-50	45-49	44-48	44-48	41-48	40-48	40-48	B 40-46	37-46	36-45	36-44	36-44	106
107	107	C 71	E 60	56	E 54	53	52	49-51	48-50	46-50	Pu B 45-48	44-48	42-48	41-48	40-48	40-47	38-46	37-46	36-45	36-44	107
108	108	72	C 61	E 56	C 55	54	53	50-52	48-51	46-50	46-49	44-48	43-48	42-48	41-48	40-48	39-47	38-46	37-46	36-45	108
109	109	E 72	C 62	E 57	C 56	E 54	SS 54	51-52	PJ 49-52	47-51	46-50	45-49	44-48	43-48	42-48	41-48	40-48	39-47	38-46	37-46	109
110	110	73	62	58	56	55	54	E 52	50-52	48-52	B 47-50	46-50	44-49	44-48	43-48	42-48	40-48	40-48	39-47	38-46	110
111	111	C 74	E 63	58	56	56	55	53	50-52	48-52	48-51	47-50	44-50	44-49	44-48	43-48	40-48	40-48	40-48	39-47	111
112	112	E 74	64	C 59	E 56	D 56	56	T1 54	50-53	48-52	48-52	We 48-51	44-50	44-50	44-49	We 44-48	41-43	40-48	40-48	We 40-48	112
113	113	C 75	64	60	C 57	56	56	E 54	50-54	48-53	48-52	48-52	44-51	44-50	44-50	B 44-48	42-48	41-48	40-48	40-48	113
114	114	76	E 64	60	58	E 56	56	54-55	50-54	48-54	B 48-52	48-52	45-52	44-51	44-50	44-49	43-48	42-48	41-48	40-48	114
115	115	E 76	C 65	E 60	E 58	57	56	54-56	51-55	49-54	48-53	48-52	46-52	45-52	44-51	44-50	44-49	43-48	42-48	41-48	115
116	116	C 77	66	C 61	C 59	F 58	57	54-56	52-56	50-55	49-54	48-53	47-52	46-52	45-52	B 44-50	44-50	44-49	43-48	42-48	116
117	117	78	E 66	62	60	E 58	58	E 55-56	53-56	51-56	B 50-54	49-54	48-53	47-52	46-52	45-51	44-50	44-50	44-49	X 43-48	117
118	118	E 78	C 67	E 62	60	59	58	D 56-57	54-56	52-56	51-55	50-54	48-54	48-53	47-52	46-52	44-51	44-50	44-50	44-49	118
119	119	C 79	68	C 63	E 60	60	59	56-58	E 55-56	53-56	52-56	51-55	49-54	48-54	48-53	47-52	44-52	44-51	44-50	44-50	119
120	120	80	68	64	C 61	60	D 60	AL E 57-58	AL 56-57	54-56	52-56	We 52-56	50-55	49-54	48-54	We B 48-52	45-52	44-52	44-51	44-50	120
121	121	E 80	E 68	64	62	E 60	60	58-59	56-58	E 55-57	B 53-56	52-56	51-56	50-55	49-54	48-53	46-52	45-52	44-52	44-51	121
122	122	C 81	C 69	64	E 62	61	60	58-60	E 56-58	56-58	54-57	53-56	52-56	51-56	50-55	48-54	47-53	46-52	45-52	44-52	122
123	123	82	E 70	E 64	C 63	E 62	D 61	59-60	56-59	56-58	55-58	B 54-56	53-56	52-56	51-56	B 48-54	48-54	47-53	46-52	45-52	123
124	124	C 82	C 70	65	64	62	62	Lc 60	56-60	56-59	56-58	55-57	54-56	53-56	52-56	48-55	48-54	48-54	47-53	46-52	124
125	125	83	71	E 66	64	63	62	61	56-60	E 56-60	56-59	56-58	B 55-57	54-56	53-56	49-56	48-55	48-54	48-54	47-53	125
126	126	E 84	E 72	C 66	E 64	64	D 63	62	56-60	56-60	56-60	56-58	56-58	55-57	I 54-56	We 50-56	48-56	48-55	48-54	48-54	126
127	127	84	72	67	64	64	64	63	56-61	56-60	56-60	56-59	56-58	56-58	55-57	51-56	48-56	48-56	48-55	48-54	127
n,k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	4,0

Appendix I. The Updated Table of Bounds

n,k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	0,1
1																					1
2																					2
3																					3
4																					4
5																					5
6																					6
7																					7
8																					8
9																					9
10																					10
11																					11
12																					12
13																					13
14																					14
15																					15
16																					16
17																					17
18																					18
19																					19
20																					20
21	1																				21
22	2	1																			22
23	2	2	1																		23
24	2	2	2	1																	24
25	2	2	2	2	1																25
n,k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	0,1

n,k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	1,1
26	3	2	2	2	2	1															26
27	4	3	2	2	2	2	1														27
28	4	4	3	2	2	2	2	1													28
29	4	4	4	3	2	2	2	2	1												29
30	B 4	4	4	4	3	2	2	2	2	1											30
31	5	4	4	4	4	3	2	2	2	2	1										31
32	6	5	4	4	4	D 4	B 2	2	2	2	2	1									32
33	6	N 6	4-5	4	4	4	3	2	2	2	2	2	1								33
34	6-7	6	5-6	4-5	4	4	4	3	2	2	2	2	2	1							34
35	6-8	R 6	6	5-6	4-5	4	4	4	3	2	2	2	2	2	1						35
36	6-8	6-7	6	6	5-6	4-5	4	4	4	3	2	2	2	2	2	1					36
37	7-8	6-8	6-7	6	6	5-6	4-5	4	4	4	3	2	2	2	2	2	1				37
38	8	7-8	6-8	6-7	6	6	5-6	4-5	4	4	4	3	2	2	2	2	2	1			38
39	8-9	8	7-8	6-8	6-7	6	6	5-6	4-5	4	4	4	3	2	2	2	2	2	1		39
40	8-10	8-9	8	7-8	6-8	6-7	6	6	5-6	4-5	4	4	4	3	2	2	2	2	2	1	40
41	N 9-10	8-10	8-9	8	7-8	6-8	6-7	6	6	Ha 5-6	4-5	4	4	4	3	2	2	2	2	2	41
42	10-11	8-10	8-10	8-9	8	7-8	6-8	6-7	6	6	4-6	4-5	4	4	4	3	2	2	2	2	42
43	10-12	9-11	8-10	8-10	8-9	8	7-8	6-8	6-7	6	5-6	4-6	R 4	4	4	4	3	2	2	2	43
44	11-12	10-12	9-11	8-10	8-10	8-9	8	7-8	6-8	6-7	6	5-6	4-5	4	4	4	4	3	2	2	44
45	12	11-12	10-12	9-11	8-10	8-10	8-9	8	7-8	6-8	R 6	6	5-6	4-5	4	4	4	4	3	2	45
46	12	12	11-12	10-12	8-11	8-10	8-10	8-9	8	7-8	6-7	6	6	5-6	4-5	4	4	4	4	3	46
47	B 12-13	12	12	N 11-12	Ha 9-12	R 8-10	8-10	8-10	8-9	8	7-8	6-7	6	6	5-6	4-5	4	4	4	4	47
48	12-14	12-13	12	12	10-12	8-11	8-10	8-10	8-10	8-9	Y 8	6-8	6-7	6	6	5-6	4-5	4	4	4	48
49	12-14	12-14	12-13	12	10-12	9-12	8-11	8-10	8-10	8-10	R 8	7-8	6-8	6-7	6	6	5-6	4-5	4	4	49
50	12-15	12-14	12-14	12-13	10-12	10-12	9-12	8-11	8-10	8-10	8-9	8	7-8	6-8	6-7	6	6	5-6	4-5	4	50
n,k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	1,1

n,k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	2,1
51	12-16	12-15	12-14	12-14	10-13	10-12	10-12	9-12	8-11	8-10	8-10	8-9	8	7-8	6-8	6-7	6	6	5-6	4-5	51
52	12-16	12-16	12-15	12-14	11-14	10-13	10-12	10-12	9-12	8-11	8-10	8-10	8-9	8	7-8	6-8	6-7	6	6	5-6	52
53	13-16	12-16	12-16	12-15	12-14	11-14	10-13	10-12	10-12	8-12	8-11	8-10	8-10	8-9	8	7-8	6-8	6-7	6	6	53
54	14-16 N	12-16	12-16	12-16	12-15	12-14 K	11-14	10-13	10-12	9-12	8-12	8-11	8-10	8-10	8-9	8	7-8	6-8	6-7	6	54
55	15-17	13-16	12-16	12-16	12-16	12-14	12-14	11-14	10-13	10-12	9-12	8-12	8-11	8-10	8-10	8-9	8	7-8	6-8	6-7	55
56	16-18	14-17	13-16	12-16	12-16	12-15	12-14	12-14	11-14	10-13	10-12	9-12	8-12	8-10 K	8-10	8-10	8-9	8	7-8	6-8	56
57	16-18	15-18	14-17	13-16	12-16	12-16	12-15	12-14	12-14	11-14	10-13	10-12 E	9-12	8-11	8-10	8-10	8-10	8-9	8	7-8	57
58	16-19	16-18	15-18	14-17	13-16	12-16	12-16	12-15	12-14	12-14	11-14	10-12	10-12	9-12	8-11	8-10	8-10	8-10	8-9	8	58
59	16-20	16-19	16-18	15-18	14-17	13-16	12-16	12-16	12-15	12-14	12-14	11-13	10-12	10-12	9-12	8-11	8-10	8-10	8-10	8-9	59
60	16-20	16-20	16-19	16-18	15-18	14-17	13-16	12-16	12-15	12-15	12-14	12-14	11-13	10-12	10-12	9-12	8-11	8-10	8-10	8-10	60
61	16-20	16-20	16-20	16-19	16-18 B	15-18	14-17	13-16	12-16	12-16	12-15	12-14 J	12-14	11-13	10-12	10-12 I	9-12	8-11	8-10	8-10	61
62	17-21 N	16-20	16-20	16-20	16-18	16-18	15-18	14-17 N	13-16	12-16 I	12-16	12-14	12-14	12-14	11-13	10-12	10-12	9-12	8-11	8-10	62
63	18-22	16-21	16-20	16-20	16-19	16-18	16-18	15-18	14-17	13-16 B	12-16	12-15	12-14	12-14	12-14	11-13	10-12	10-12	9-12	8-11	63
64	18-22	17-22	16-21	16-20	16-20	16-19	16-18	16-18	14-18	14-16	12-16	12-16	12-15	12-14	12-14	12-14	10-13	10-12	10-12	9-12	64
65	18-23	18-22	17-22	16-21 M	16-20	16-20	16-19	16-18	15-18	14-17	13-16	12-16	12-16	12-15	12-14	12-14	11-14	10-13	10-12	10-12	65
66	19-24	18-23	18-22	17-22	16-21	16-20	16-20	16-19	16-18	15-18	14-17	13-16	12-16	12-16	12-15	12-14	12-14	11-14	10-13 O	10-12	66
67	20-24	19-24	18-23	18-22	16-22	16-21	16-20 E	16-20	16-19	16-18	15-18	14-17	13-16	12-16	12-16	12-15	12-14	12-14	11-14	10-13	67
68	20-24	20-24	19-24	18-23	17-22	16-22	16-20	16-20	16-20	16-19	16-18	15-18	14-17	13-16	12-16	12-16	12-15	12-14	12-14	10-14	68
69	20-24	20-24	20-24	19-24	18-23	17-22 B	16-21	16-20	16-20	16-20	16-19	16-18	15-18	14-17	13-16	12-16	12-16	12-15	12-14 J	10-14	69
70	20-24	20-24	20-24	20-24	19-24	18-22	17-22	16-21 M	16-20	16-20	16-20	16-19	16-18	15-18	14-17	13-16	12-16	12-16	12-14	10-14	70
71	20-25	20-24	20-24	20-24	20-24	19-23	18-22	17-22	16-21	16-20	16-20	16-20	16-19	16-18 K	15-18	14-17	12-16	12-16	12-15	10-14	71
72	20-26	20-25	20-24	20-24	20-24	20-24	19-23 PT	18-22	16-22	16-21	16-20	16-20	16-20	16-18	16-18	15-18 PT	13-17	12-16 Ka	12-16 K	11-15	72
73	20-26	20-26	20-25	20-24	20-24	20-24	20-24	18-23	17-22	16-22	16-21	16-20	16-20	16-19	16-18	16-18	14-18	13-16	12-16	12-16	73
74	21-27	20-26	20-26	20-25	20-24	20-24	20-24	18-24	18-23	17-22	16-22	16-21	16-20	16-20	16-19	16-18	14-18	14-17	12-16	12-16	74
75	22-28	21-27	20-26	20-26	20-25	20-24	20-24	18-24	18-24	18-23	17-22	16-22	16-21	16-20	16-20	16-19	14-18	14-18	12-17	12-16	75
n,k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	2,1

n,k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	3,1
76	22-28	22-28	21-27	20-26 M	20-26	20-24 B	20-24	18-24	18-24	18-24	18-23	17-22	16-22	16-20 K	16-20	16-20	15-19	14-18	13-18	12-17	76
77	22-28	22-28	22-28	21-27	20-26	20-25	20-24	18-24	18-24	18-24	18-24	18-23	17-22	16-21	16-20	16-20	16-20	15-19	14-18	13-18	77
78	23-29	22-28	22-28	22-28	20-27	20-26 B	20-25	19-24	18-24	18-24	18-24	18-24	18-23	17-22	16-21	16-20	16-20	16-20	15-19	14-18 L	78
79	24-30	23-29	22-28	22-28	20-28	20-26	20-26	20-25	19-24	18-24	18-24	18-24	18-24	18-23	17-22 E	16-21	16-20	16-20	16-20	15-19	79
80	24-30	24-30	23-29	22-28	21-28	20-27	20-26	20-26	20-25	19-24	18-24	18-24	18-24	18-24	18-22	17-22	16-21	16-20	16-20	16-20	80
81	24-31	24-30	24-30	23-29	22-28	21-28	20-27	20-26	20-26	20-25	19-24	18-24	18-24	18-24	18-23	18-22	17-22	16-21	16-20	16-20	81
82	24-32	24-31	24-30	24-30	23-29	22-28	21-28	20-27	20-26	20-26	20-25	19-24	18-24 M	18-24	18-24	18-23	18-22	17-22	16-21	16-20	82
83	24-32	24-32	24-31	24-30	24-30	23-29	22-28 E	21-28	20-27	20-26	20-26	20-25	19-24	18-24	18-24	18-24	18-23	18-22	17-22	16-20	83
84	25-32	24-32	24-32	24-31	24-30	24-30	23-28	22-28	21-28	20-27	20-26	20-26	20-25	18-24	18-24	18-24	18-24	18-23	18-22	17-21	84
85	26-32	25-32	24-32	24-32	24-31	24-30	24-29	23-28	22-28	21-28	20-27	20-26	20-26	19-25	18-24	18-24	18-24	18-24	18-23	18-22	85
86	27-32	26-32	25-32	24-32	24-32	24-31	24-30 E	24-29	23-28	22-28	21-28 PT	20-27	20-26	20-26	19-25	18-24	18-24	18-24	18-24	18-23	86
87	28-33	27-32	26-32	24-32	24-32	24-32	24-30	24-30	24-29	23-28 L	22-28	20-28	20-27	20-26	20-26	19-25	18-24	18-24	18-24	18-24	87
88	28-34	28-33	27-32 X	24-32	24-32	24-32	24-31	24-30	24-30	24-29	22-28	20-28	20-28	20-27	20-26	20-26	19-25	18-24	18-24	18-24	88
89	28-34	28-34	28-33	24-32	24-32	24-32	24-32	24-31	24-30	24-30	22-29	20-28	20-28	20-28	20-27	20-26	20-26	19-25	18-24	18-24	89
90	28-35	28-34	28-34	24-33	24-32	24-32	24-32	24-32	24-31	24-30	22-30	20-29	20-28	20-28	20-28	20-27	20-26	20-26	19-25	18-24	90
91	29-36	28-35	28-34	24-34	24-32 B	24-32	24-32	24-32	24-32	24-31	22-30	21-30	20-28 E	20-28	20-28	20-28	20-27	20-26	20-26	19-25	91
92	30-36	29-36	28-35 PT	24-34 B	24-33	24-32	24-32	24-32	24-32	24-32	23-31 PT	22-30	21-29 PT	20-28	20-28	20-28	20-28	20-27	20-26	20-25	92
93	30-36	30-36	29-36	24-34	24-34	24-33	24-32	24-32	24-32	24-32	24-32	22-31	22-30 E	20-29	20-28	20-28	20-28	20-28	20-27	20-25	93
94	30-37	30-36	30-36	24-35	24-34	24-34	24-33	24-32	24-32	24-32	24-32	22-32	22-30	20-30	20-29	20-28	20-28	20-28	20-28	20-27	94
95	30-38	30-37	30-36	24-36	24-35	24-34	24-34	24-33	24-32	24-32	24-32	22-32	22-31	20-30	20-30	20-29	20-28	20-28	20-28	20-28	95
96	30-38	30-38	30-37	24-36 B	24-36	24-35	24-34	24-34	24-33	24-32	24-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	96
97	30-39	30-38	30-38	25-36	24-36	24-36	24-35	24-34	24-34	24-33	24-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	97
98	30-40	30-39	30-38	26-37	25-36	24-36	24-36	24-35	24-34	24-34	24-33	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-29	98
99	30-40	30-40	30-39	27-38 B	26-37	25-36	24-36	24-36	24-35	24-34	24-34	22-33	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	99
100	31-40	30-40	30-40	28-38	27-38	26-37	25-36	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-32	21-31	20-30	20-29	100
n,k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	3,1

n,k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	n,k
101	32-40	31-40	30-40	29-39	28-38	27-38	26-37	25-36	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-32	21-31	20-30 E	101
102	32-41	32-40	31-40	30-40	28-39	28-38	27-38	26-37 X	24-36	24-36	24-36	24-35	24-34	23-34	22-33 X	22-32	22-32	22-32	22-32	21-30 E	102
103	32-42	32-41	32-40	31-40 We	28-40	28-39	28-38	27-38	25-37	24-36	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-31	103
104	32-42	32-42	32-41	32-40 B	28-40	28-40	28-39	28-38	26-38	25-37	24-36	24-36	24-36	24-35	24-34	22-34	22-33	22-32	22-32	22-32	104
105	32-43	32-42	32-42	32-40	28-40	28-40	28-40	28-39	27-38	26-38	25-37	24-36	24-36	24-36	24-35	23-34	22-34	22-33	22-32	22-32	105
106	33-44	32-43	32-42	32-41	29-40	28-40	28-40	28-40	28-39	27-38	26-38	25-37	24-36	24-36	24-36	24-35	23-34	22-34	22-32 E	22-32	106
107	34-44	33-44	32-43	32-42 B	30-41	29-40	28-40	28-40	28-40	28-39	27-38	26-38	25-37	24-36	24-36	24-36	24-35	23-34	22-33	22-32	107
108	35-44	34-44	33-44	32-42	31-42	30-41	29-40	28-40	28-40	28-40	28-39	27-38	26-38	25-37	24-36 E	24-36	24-36	24-35	23-34	22-33	108
109	36-45	35-44	34-44	33-43	32-42	31-42	30-41	29-40	28-40	28-40	28-40	28-39	27-38	26-38	25-36	24-36	24-36	24-36	24-34 E	23-34	109
110	36-46	36-45	35-44	34-44	32-43	32-42	31-42	30-41	29-40	28-40	28-40	28-40	28-39	27-38	26-37	24-36	24-36	24-36	24-35	24-34	110
111	36-46	36-46	36-45	35-44 We	32-44	32-43	32-42	31-42	30-41	29-40	28-40	28-40	28-40	28-39	27-38 X	25-37	24-36	24-36	24-36	24-35	111
112	36-47	36-46	36-46	36-45	32-44 B	32-44	32-43	32-42	31-42	30-41 X	28-40	28-40	28-40	28-40	28-38 E	26-38	25-37	24-36	24-36	24-36	112
113	37-48	36-47	36-46	36-46	33-44	32-44	32-44	32-43	32-42	31-42	29-41	28-40	28-40	28-40	28-39	27-38	26-38	25-37	24-36	24-36	113
114	38-48	37-48	36-47	36-46	34-45	33-44	32-44	32-44	32-43	32-42	30-42	29-41	28-40	28-40	28-40	28-39	27-38	26-38	25-37	24-36	114
115	39-48	38-48	37-48	36-47	35-46	34-45	33-44	32-44	32-44	32-43	31-42	30-42	29-41	28-40	28-40	28-40	28-39	27-38	26-38	25-37	115
116	40-48	39-48	38-48	37-48	36-46 B	35-46	34-45	33-44	32-44	32-44	32-43	31-42	30-42	29-41	28-40	28-40	28-40	28-39	27-38	26-38	116
117	41-48	40-48	39-48	38-48	37-47	36-46	35-46	34-45	33-44	32-44	32-44	32-43	31-42	30-42	29-40 E	28-40	28-40	28-40	28-39	27-38	117
118	42-48	41-48	40-48	39-48	38-48	37-47	36-46	35-46	34-45	32-44	32-44	32-44	32-43	31-42	30-41	29-40	28-40	28-40	28-40	28-39	118
119	43-49	42-48	41-48	40-48	39-48	38-48	37-47	36-46	35-46	32-45	32-44	32-44	32-44	32-43	31-42 E	30-41	29-40	28-40	28-40	28-40	119
120	44-50	43-49	42-48	41-48	40-48	39-48	38-48	37-47	36-46	32-46	32-45	32-44	32-44	32-44	32-42	31-42	30-41	29-40	28-40	28-40	120
121	44-50	44-50	43-49	42-48	41-48	40-48	39-48	38-48	37-47	32-46	32-46	32-45	32-44	32-44	32-43	32-42	31-42	30-41	29-40	28-40	121
122	44-51	44-50	44-50	43-49	42-48 B	41-48	40-48	39-48	38-48	32-47	32-46	32-46	32-45	32-44	32-44	32-43	32-42	31-42	30-41	29-40	122
123	44-52	44-51	44-50	44-50	43-48	42-48	41-48	40-48	39-48	33-48	32-47	32-46	32-46	32-45	32-44	32-44	32-43	32-42	31-42	30-41	123
124	45-52	44-52	44-51	44-50	44-49	43-48	42-48	41-48	40-48	34-48	33-48	32-47	32-46	32-46	32-45	32-44 E	32-44	32-43	32-42	31-42	124
125	46-52	45-52	44-52	44-51	44-50 B	44-49	43-48	42-48	41-48	35-48	34-48	33-48	32-47	32-46	32-46	32-44	32-44	32-44	32-43	32-42	125
126	47-53	46-52 I	44-52	44-52	44-50	44-50	44-49	43-48	42-48 I	36-48	35-48	34-48 We	32-48	32-47	32-46	32-45	32-44	32-44	32-44	32-43	126
127	48-54	47-53	44-52	44-52	44-51	44-50	44-50	44-49	43-48	36-48	36-48	35-48	32-48	32-48	32-47	32-46	32-45	32-44	32-44	32-44	127
n,k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	n,k

n,k	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	1,2
26																					26
27																					27
28																					28
29																					29
30																					30
31																					31
32																					32
33																					33
34																					34
35																					35
36																					36
37																					37
38																					38
39																					39
40																					40
41	1																				41
42	2	1																			42
43	2	2	1																		43
44	2	2	2	1																	44
45	2	2	2	2	1																45
46	2	2	2	2	2	1															46
47	3	2	2	2	2	2	1														47
48	4	3	2	2	2	2	2	1													48
49	4	4	3	2	2	2	2	2	1												49
50	4	4	4	3	2	2	2	2	2	1											50
n,k	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	1,2

n,k	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	2,2
51	4	4	4	4	3	2	2	2	2	2	1										51
52	4-5	4	4	4	4	3	2	2	2	2	2	1									52
53	5-6	4-5	4	4	4	4	3	2	2	2	2	2	1								53
54	6	5-6	4-5	4	4	4	4	3	2	2	2	2	2	1							54
55	6	6	5-6	4-5	4	4	4	4	3	2	2	2	2	2	1						55
56	6-7	6	6	5-6	4-5	4	4	4	4	3	2	2	2	2	2	1					56
57	6-8	6-7	6 K	6	5-6	4-5	4	4	4	4	3	2	2	2	2	2	1				57
58	7-8	7-8	6	6	6	5-6	4-5	4	4	4	4	3	2	2	2	2	2	1			58
59	8	7-8	6-7	6	6	6	5-6	4-5	4	4	4	4	3	2	2	2	2	2	1		59
60	8 K	8	7-8	6-7	6	6	6	5-6	4-5	4	4	4	4	3	2	2	2	2	2	1	60
61	8-9	8	8	7-8	6-7	6	6	6	5-6	4-5	4	4	4	4	3	2	2	2	2	2	61
62	8-10	8-9	8	8	7-8	6-7 N	6	6	6	5-6	4-5	4	4	4	4	3	2	2	2	2	62
63	8-10 K	8-10	8-9	8	8	7-8	6-7	6	6	6	5-6	4-5	4 K	4	4	4	3 D	2 B	2	2	63
64	8-10	8-10	8-10	8-9	8	8	6-8	6-7	6	6	6	5-6	4 N	4	4	4	4	2	2	2	64
65	9-11	8-10	8-10	8-10	8-9	8	7-8	6-8	6-7	6	6	6	5	4	4	4	4	3	2	2	65
66	10-12	9-11	8-10	8-10	8-10	8-9	8	7-8	6-8 M	6-7	6	6	6	4-5	4	4	4	4	3	2	66
67	10-12	10-12	9-11	8-10	8-10	8-10	8-9	8	7-8	6-8	6-7	6	6	5-6	4-5	4	4	4	4	3	67
68	10-13	10-12	10-12	9-11	8-10	8-10	8-10	8-9	8	6-8	6-8	6-7	6	6	5-6	4-5	4	4	4	4	68
69	10-14	10-13	10-12 K	10-12	9-11	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	5-6	4-5	4	4	4	69
70	10-14	10-14	10-12	10-12	10-12	9-11	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	5-6	4-5	4	4	70
71	10-14	10-14	10-13	10-12	10-12	10-12	9-11	8-10	8-10	8-10	8-9	7-8 K	6-8 Ha	6-8	6-7	6	6	5-6	4-5	4	71
72	10-14	10-14	10-14	10-13	10-12	10-12	10-12	9-11	8-10 V	8-10	8-10	8	7-8	6-8	6-8	6-7	6	6	5-6	4-5 Q	72
73	11-15	10-14	10-14	10-14	10-13	10-12	10-12	10-12	9-11	8-10	8-10	8-9	8	6-8	6-8	6-8	6-7	6	6	5-6	73
74	12-16	11-15	10-14	10-14	10-14	10-13	10-12	10-12	10-12	8-11	8-10	8-10	8-9	7-8	6-8	6-8	6-8	6-7	6	6	74
75	12-16	12-16	11-15	10-14	10-14	10-14	10-13	10-12	10-12	8-12	8-11	8-10	8-10	8-9	7-8	6-8	6-8	6-8	6 K	6	75
n,k	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	2,2

Appendix I. The Updated Table of Bounds

n,k	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	3,2
76	12-16	12-16	12-16	Ka 11-15	10-14	10-14	10-14	10-13	10-12	Ka 9-12	8-12	8-11	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	76
77	12-17	12-16	12-16	12-16	10-15	10-14	10-14	10-14	10-13	10-12	8-12	8-12	8-11	8-10	8-10	8-9	7-8	6-8	6-8	6-7	77
78	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	9-12	8-12	8-12	8-11	8-10	8-10	8-9	7-8	6-8	6-8	78
79	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	9-12	8-12	8-12	8-11	8-10	8-10	8-9	7-8	6-8	79
80	13-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	9-12	8-12	8-12	8-11	8-10	8-10	8-9	7-8	80
81	14-20	13-19	12-18	12-18	K 12-16	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	9-12	8-12	8-12	8-11	8-10	8-10	8-9	81
82	14-20	14-20	K 13-18	12-18	12-17	12-16	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	9-12	8-12	8-12	8-11	8-10	8-10	82
83	15-20	14-20	14-19	13-18	12-18	12-17	12-16	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	9-12	8-12	8-12	8-11	8-10	83
84	16-20	15-20	14-20	14-19	13-18	12-18	12-17	12-16	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	9-12	8-12	8-12	8-11	84
85	17-21	16-20	15-20	14-20	J 14-18	13-18	12-18	12-17	12-16	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	9-12	8-12	8-12	85
86	18-22	17-21	16-20	15-20	14-19	14-18	13-18	12-18	12-17	12-16	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	9-12	8-12	86
87	18-23	18-22	17-21	16-20	15-20	14-19	14-18	13-18	12-18	12-17	12-16	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	9-12	87
88	18-24	E 18-22	18-22	17-21	16-20	14-20	14-19	14-18	13-18	12-18	12-17	12-16	12-16	12-16	11-16	10-15	10-14	10-14	10-14	E 10-12	88
89	18-24	18-23	18-22	18-22	L K 17-20	14-20	14-20	14-19	14-18	13-18	12-18	12-17	12-16	12-16	12-16	PT 11-16	10-15	10-14	10-14	10-13	89
90	18-24	18-24	18-23	18-22	18-21	14-20	14-20	14-20	14-19	14-18	13-18	12-18	12-17	12-16	12-16	12-16	10-16	K 10-14	10-14	10-14	90
91	18-24	18-24	18-24	18-23	18-22	14-21	14-20	14-20	14-20	14-19	PT 14-18	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	91
92	19-25	18-24	18-24	18-24	18-23	14-22	14-21	14-20	14-20	14-20	14-19	12-18	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	92
93	20-26	19-25	18-24	18-24	18-24	15-23	14-22	14-21	14-20	14-20	14-20	13-19	12-18	12-18	12-18	12-17	12-16	12-16	11-16	10-15	93
94	20-26	20-26	19-25	18-24	18-24	16-24	15-23	14-22	14-21	14-20	14-20	14-20	13-19	12-18	12-18	12-18	12-17	12-16	12-16	11-16	94
95	20-27	20-26	20-26	E 19-24	18-24	17-24	16-24	15-23	14-22	14-21	14-20	14-20	14-20	13-19	12-18	12-18	12-18	12-17	12-16	12-16	95
96	20-28	20-27	20-26	20-25	19-24	18-24	17-24	16-24	15-23	14-22	14-21	14-20	14-20	14-20	13-19	12-18	12-18	12-18	12-17	12-16	96
97	20-28	20-28	20-27	20-26	20-25	19-24	18-24	17-24	16-24	15-23	14-22	14-21	14-20	14-20	14-20	13-19	12-18	12-18	12-18	12-17	97
98	20-28	20-28	20-28	E 20-26	20-26	20-25	19-24	18-24	17-24	16-24	15-23	14-22	14-21	14-20	14-20	14-20	13-19	12-18	12-18	12-18	98
99	20-28	20-28	20-28	20-27	20-26	20-26	20-25	19-24	18-24	17-24	16-24	15-23	14-22	14-21	14-20	14-20	14-20	13-19	12-18	12-18	99
100	20-28	20-28	20-28	20-28	20-27	20-26	20-26	20-25	19-24	18-24	17-24	16-24	14-23	14-22	14-21	14-20	14-20	14-20	13-19	12-18	100
n,k	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	3,2

n,k	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	4,2
101	20-29	20-28	20-28	20-28	20-28	20-27	20-26	20-26	20-25	19-24	18-24	17-24	15-24	14-23	14-22	14-21	14-20	14-20	14-20	Ro 13-19	101
102	20-30	20-29	20-28	20-28	20-28	20-28	20-27	20-26	20-26	20-25	19-24	18-24 L	16-24	15-24	14-23	14-22	14-21	14-20	14-20	14-20	102
103	21-30	20-30	20-29	20-28	20-28	20-28	20-28	20-27	20-26	20-26	20-25	19-24	16-24	15-24	15-24	14-23	14-22	14-21	14-20	14-20	103
104	22-31	21-30	20-30 X	20-29	20-28	20-28	20-28	20-28	20-27	20-26	20-26	20-25	17-24	15-24	16-24	15-24	14-23	14-22	14-21	14-20	104
105	22-32	22-31	21-30	20-30	20-29	20-28	20-28	20-28	20-28	20-27	20-26	20-26	18-25	17-24	16-24	16-24	15-24 Ro	14-23	14-22	14-21	105
106	22-32	22-32	22-31	20-30	20-30	20-29	20-28	20-28	20-28	20-28	20-27	20-26	19-26	18-25	16-24	16-24	16-24	14-24	14-23	14-22	106
107	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	20-28	20-27	20-26	19-26 L	16-25	16-24	16-24	15-24	14-24	14-23	107
108	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	20-28	20-27	20-26	16-26	16-25	16-24	16-24	15-24	14-24	108
109	22-33	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	20-28	20-27	17-26	16-26	16-25	16-24	16-24	15-24	109
110	23-34	22-33	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	20-28	18-27	17-26	16-26	16-25	16-24	16-24	110
111	24-34	23-34 X	22-33	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	19-28	18-27	17-26	16-26	16-25	16-24	111
112	24-35	24-34	22-34	22-33	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	19-28	18-27	17-26	16-26	16-25	112
113	24-36	24-35	23-34	22-34	22-33	22-32 E	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	19-28	18-27	17-26	16-26	113
114	24-36	24-36	24-35	23-34	22-34	22-32	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	19-28	18-27	17-26	114
115	24-36	24-36	24-36	24-35	23-34	22-33	22-32	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	19-28	18-27	115
116	25-37	24-36	24-36	24-36	24-35	23-34 E	22-33	22-32	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	19-28	116
117	26-38	25-37	24-36	24-36	24-36	24-34	23-34	22-33	22-32	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28	20-28	20-28	117
118	27-38	26-38	25-37	24-36 E	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-29	20-28 E	20-28	118
119	28-39	27-38	26-38	25-36	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-32	22-32	21-31	20-30	20-30	20-28	20-28	119
120	28-40	28-39	27-38	26-37	25-36	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-32	22-32	21-31	20-30	20-29	20-28	120
121	28-40	28-40	28-39	27-38 E	26-37	25-36 E	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-32	22-32	21-31	20-30 E	20-29	121
122	28-40	28-40	28-40	28-38	27-38	26-36	25-36	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-32	22-32	21-30	20-30	122
123	29-40 E	28-40	28-40	28-39	28-38	27-37	26-36	25-36	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-32	22-31	21-30	123
124	30-40	29-40	28-40	28-40	28-39	28-38 E	27-37	26-36	25-36	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-32	22-31	124
125	31-41	30-40	29-40	28-40	28-40	28-38	28-38	27-37	26-36	25-36	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	22-32	22-32	125
126	32-42 E	31-41	30-40 W	29-40	28-40	28-39	28-38	28-38	27-37	26-36 I	24-36	24-36	24-36	24-35	24-34	23-34	22-33 I	22-32	22-32	22-32	126
127	32-42	32-42	31-41	28-40	28-40	28-40	28-39	28-38	28-38	27-37	24-36	24-36	24-36	24-36	24-35	24-34	23-34	22-33	22-32	22-32	127
n,k	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	4,2

Appendix I. The Updated Table of Bounds

n,k	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	2,3
51																					51
52																					52
53																					53
54																					54
55																					55
56																					56
57																					57
58																					58
59																					59
60																					60
61	1																				61
62	2	1																			62
63	2	2	1																		63
64	2	2	2	1																	64
65	2	2	2	2	1																65
66	2	2	2	2	2	1															66
67	2	2	2	2	2	2	1														67
68	3	2	2	2	2	2	2	1													68
69	4	3	2	2	2	2	2	2	1												69
70	4	4	3	2	2	2	2	2	2	1											70
71	4	4	4	3	2	2	2	2	2	2	1										71
72	4	4	4	4	3	2	2	2	2	2	2	1									72
73	4-5	4	4	4	4	3	2	2	2	2	2	2	1								73
74	4-6	4-5	4	4	4	4	3	2	2	2	2	2	2	1							74
75	5-6	4-6	4-5	4	4	4	4	3	2	2	2	2	2	2	1						75
n,k	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	2,3

n,k	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	3,3
76	6	5-6	4-6	4-5	4	4	4	4	3	2	2	2	2	2	2	1					76
77	6	6	5-6	4-6	4-5	4	4	4	4	3	2	2	2	2	2	2	1				77
78	6-7	6	6	5-6	4-6	4-5	4	4	4	4	3	2	2	2	2	2	2	1			78
79	6-8	6-7	6	6	5-6	4-6	4-5	4	4	4	4	3	2	2	2	2	2	2	1		79
80	6-8	6-8	6-7	6	6	5-6	4-6	4-5	4	4	4	4	3	2	2	2	2	2	2	1	80
81	7-8	6-8	6-8	6-7	6	6	5-6	4-6	4-5	4	4	4	4	3	2	2	2	2	2	2	81
82	8-9	7-8	6-8	6-8	6-7	6	6	5-6	4-6	4-5	4	4	4	4	3	2	2	2	2	2	82
83	8-10	8-9	7-8	6-8	6-8	6-7	6	6	5-6	4-6	4-5	4	4	4	4	3	2	2	2	2	83
84	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	5-6	4-6	4-5	4	4	4	4	3	2	2	2	84
85	8-11	8-10	8-10	8 K	7-8	6-8	6-8	6-7	6	6	5-6	4-6	4-5	4	4	4	4	3	2	2	85
86	8-12	8-11	8-10	8-9	8	7-8 Ha	6-8	6-8	6-7	6	6	5-6	4-6	4-5	4	4	4	4	3	2	86
87	8-12	8-12	8-11	8-10	8-9	8	6-8	6-8	6-8	6-7	6	6	5-6	4-6	4-5	4	4	4	4	3	87
88	9-12	8-12	8-12	8-10	8-10	8-9	7-8	6-8	6-8	6-8	6-7	6	6	5-6	4-6	4-5	4	4	4	4	88
89	10-12 K	9-12	8-12 Q	8-11 J	8-10 K	8-10	8-9	7-8	6-8	6-8	6-8	6-7	6	6	5-6	4-6	4-5	4 J	4	4	89
90	10-12	10-12	9-12	8-12	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-8	6-7	6	6	5-6	4-6	4	4	4	90
91	10-13	10-12	10-12	8-12	8-11	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-8	5-7	6 J	6	5-6	4-5	4	4	91
92	10-14	10-13	10-12	9-12	8-12	8-11	8-10	8-10	8-10	8-9	7-8	6-8	6-8	5-8	6	6	6	5-6	4-5	4	92
93	10-14	10-14	10-13	10-12	9-12	8-12	8-11	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	93
94	10-15	10-14	10-14	10-13	10-12	9-12	8-12	8-11	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	94
95	11-16	10-15	10-14	10-14	10-13	10-12	9-12	8-12	8-11	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	95
96	12-16	11-16	10-15	10-14 K	10-14	10-13	10-12	9-12	8-12	8-11	8-10 K	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	96
97	12-16	12-16	11-16	10-14	10-14	10-14	10-13	10-12	9-12	8-12	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	97
98	12-17	12-16	12-16	11-15	10-14 Ro	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	98
99	12-18	12-17	12-16	12-16	11-15	10-14	10-14	10-14	10-13	10-12	9-12	9-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	99
100	12-18	12-18	12-17	12-16	12-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	100
n,k	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	3,3

n,k	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	4,3
101	12-18	12-18	12-18	12-17	12-16 E	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	101
102	12-19	12-18	12-18	12-18	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	102
103	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	103
104	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	104
105	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	105
106	14-21	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	106
107	14-22	14-21	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	107
108	14-23	14-22	14-21	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	108
109	14-24	14-23	14-22	14-21	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	109
110	15-24	14-24	14-23	14-22	14-21	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	110
111	16-24	15-24	14-24	14-23	14-22 E	14-20 E	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	111
112	16-24	16-24	15-24	14-24	14-22 E	14-21	14-20	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	112
113	16-24 E	16-24	16-24	15-24	14-23	14-22	14-21	14-20	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	113
114	16-25	16-24 E	16-24	16-24	15-24	14-22 E	14-22	14-21	14-20	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	114
115	17-26	16-24	16-24	16-24	16-24	15-23	14-22	14-22	14-21	14-20	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	115
116	18-26 E	17-25	16-24	16-24	16-24	16-24	15-23	14-22	14-22	14-21	14-20	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	116
117	19-27	18-26 E	17-25	16-24	16-24	16-24	16-24	15-23	14-22	14-22	14-21	14-20 E	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	117
118	20-28	19-26	18-26	17-25	16-24	16-24	16-24	16-24	15-23	14-22	14-22	14-20	14-20	14-20	14-20	13-20	12-19	12-18	12-18	12-17	118
119	20-28	20-27	19-26	18-26	17-25	16-24	16-24	16-24	16-24	15-23	14-22	14-21	14-20	14-20	14-20	14-20	13-20	12-19	12-18	12-18	119
120	20-28	20-28	20-27	19-26	18-26	17-25	16-24	16-24	16-24	16-24	15-23	14-22	14-21	14-20	14-20	14-20	14-20	13-20	12-19	12-18	120
121	20-28	20-28	20-28	20-27	19-26	18-26	17-25	16-24	16-24	16-24	16-24	15-22 E	14-22	14-21	14-20	14-20	14-20	14-20	13-20	12-19	121
122	20-29	20-28	20-28	20-28	20-27	19-26	18-26	17-25	16-24	16-24	16-24	16-23	15-22	14-22	14-21	14-20	14-20	14-20	14-20	13-20	122
123	20-30	20-29	20-28	20-28	20-28	20-27	19-26	18-26	17-25	16-24	16-24	16-24	16-23	15-22	14-22	14-21	14-20	14-20	14-20	14-20	123
124	21-30	20-30	20-29	20-28	20-28	20-28	20-27	19-26	18-26	17-25	16-24	16-24	16-24	16-23	15-22	14-22	14-21	14-20	14-20	14-20	124
125	22-31	21-30	20-30	20-29	20-28 E	20-28	20-28	20-27	19-26	18-26	17-25	16-24	16-24	16-24	16-23	15-22	14-22	14-21	14-20	14-20	125
126	22-32	22-31	21-30	20-30 I	20-28	20-28	20-28	20-28	20-27	19-26	18-26 I	16-25	16-24	16-24	16-24	16-23	15-22	14-22 I	14-21	14-20	126
127	22-32	22-32	22-31	21-30	20-29	20-28	20-28	20-28	20-28	20-27	19-26	16-26	16-25	16-24	16-24	16-24	16-23	15-22	14-22	14-21	127
n,k	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	4,3

	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	3,4
																					76
																					77
																					78
																					79
																					80
	1																				81
	2	1																			82
	2	2	1																		83
	2	2	2	1																	84
	2	2	2	2	1																85
	2	2	2	2	2	1															86
	2	2	2	2	2	2	1														87
	3	2	2	2	2	2	2	1													88
	4	3	2	2	2	2	2	2	1												89
	4	4	3	2	2	2	2	2	2	1											90
	4	4	4	3	2	2	2	2	2	2	1										91
	4	4	4	4	3	2	2	2	2	2	2	1									92
	4	4	4	4	4	3	2	2	2	2	2	2	1								93
	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1							94
	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1						95
	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1					96
	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1				97
	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1			98
	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1		99
	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1	100
	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	3,4

Appendix I. The Updated Table of Bounds

n,k	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	n,k
101	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	101
102	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	102
103	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	103
104	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	104
105	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	105
106	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	106
107	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	107
108	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	108
109	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	109
110	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	110
111	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	111
112	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	112
113	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	113
114	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	114
115	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	115
116	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	116
117	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	117
118	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	118
119	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	6-8	119
120	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	6-8	120
121	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	7-8	121
122	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	8-9	122
123	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	8-10	123
124	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	8-10	124
125	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	8-10	125
126	14-20	14-20	14-20	13-20	12-19	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	8-10	126
127	14-20	14-20	14-20	14-20	13-20	12-18	12-18	12-18	12-17	12-16	12-16	11-16	10-15	10-14	10-14	10-14	10-13	10-12	9-12	8-11	127
n,k	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	n,k

n,k	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	4,5
101	1																				101
102	2	1																			102
103	2	2	1																		103
104	2	2	2	1																	104
105	2	2	2	2	1																105
106	2	2	2	2	2	1															106
107	2	2	2	2	2	2	1														107
108	3	2	2	2	2	2	2	1													108
109	4	3	2	2	2	2	2	2	1												109
110	4	4	3	2	2	2	2	2	2	1											110
111	4	4	4	3	2	2	2	2	2	2	1										111
112	4	4	4	4	3	2	2	2	2	2	2	1									112
113	4	4	4	4	4	3	2	2	2	2	2	2	1								113
114	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1							114
115	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1						115
116	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1					116
117	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1				117
118	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1			118
119	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1		119
120	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	1	120
121	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	2	121
122	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	2	122
123	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	2	123
124	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	2	124
125	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4-5	4	4	4	4	4	3	2	2	125
126	8-10	8-10	8-10	8-9	7-8	6-8	6-8	6-7	6	6	6	5-6	4 K	4	4	4	4	4	3	2	126
127	8-10	8-10	8-10	8-10	8-9	7-8 I	6-8	6-8	6-7	6	6	6	5 I	4	4	4	4	4	4	3 D	127
n,k	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	4,5

n,k	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	4,6
101																					101
102																					102
103																					103
104																					104
105																					105
106																					106
107																					107
108																					108
109																					109
110																					110
111																					111
112																					112
113																					113
114																					114
115																					115
116																					116
117																					117
118																					118
119																					119
120																					120
121	1																				121
122	2	1																			122
123	2	2	1																		123
124	2	2	2	1																	124
125	2	2	2	2	1																125
126	2	2	2	2	2	1															126
127	2	2	2	2	2	2	1														127
n,k	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	4,6

External Improvements Used for Reconstructed Table

(lower, 7, 4, 3) I
(upper, 12, 5, 4) H
(lower, 15, 11, 3) I
(lower, 17, 9, 5) N
(lower, 20, 5, 9) F
(lower, 22, 6, 9) G
(lower, 23, 12, 7) N
(lower, 23, 14, 5) G
(lower, 27, 14, 7) L
(upper, 27, 15, 6) R
(lower, 31, 6,15) I
(lower, 31, 11,11) I
(lower, 31, 16, 8) L
(lower, 31, 21, 5) I
(lower, 31, 26, 3) I
(lower, 33, 13,10) N
(lower, 33, 22, 6) N
(upper, 34, 8,14) R
(upper, 34, 11,12) R
(upper, 35, 22, 6) R
(lower, 36, 20, 8) Y
(lower, 37, 6,17) F
(lower, 37, 11,13) O
(upper, 38, 18,10) K
(upper, 39, 12,14) R
(lower, 39, 28, 5) P
(lower, 41, 11,15) M
(lower, 41, 21, 9) N
(lower, 43, 15,13) N
(upper, 43, 33, 4) R
(upper, 44, 16,14) R
(upper, 45, 31, 6) R
(lower, 47, 15,15) M
(lower, 47, 24,11) N
(upper, 47, 26,10) R
(lower, 48, 31, 8) Y
(upper, 49, 31, 8) R
(lower, 51, 8,24) N
(lower, 51, 17,16) N
(lower, 51, 19,14) N
(lower, 52, 6,25) F
(lower, 52, 12,20) Z
(lower, 52, 29, 9) Q
(lower, 53, 14,17) L
(lower, 55, 21,15) N
(upper, 55, 26,14) K
(lower, 56, 16,17) M
(upper, 56, 34,10) K
(lower, 57, 11,23) M
(upper, 58, 43, 6) K
(lower, 59, 8,26) U
(upper, 60, 41, 8) K
(upper, 62, 32,14) J
(lower, 63, 7,31) I
(upper, 63, 9,28) J
(lower, 63, 10,27) I
(lower, 63, 11,26) N
(lower, 63, 16,23) I
(lower, 63, 18,21) I
(lower, 63, 19,19) N
(lower, 63, 21,18) N
(lower, 63, 28,15) N
(lower, 63, 30,13) I
(lower, 63, 36,11) I
(lower, 63, 46, 7) N
(lower, 63, 57, 3) I
(upper, 64, 41,10) K
(upper, 64, 53, 4) K
(lower, 65, 53, 5) N
(lower, 66, 14,25) L
(lower, 66, 18,23) O
(lower, 66, 20,19) M
(lower, 66, 24,17) M
(upper, 66, 29,18) K
(lower, 66, 30,15) M
(lower, 67, 10,29) O
(lower, 67, 39,11) O
(lower, 67, 49, 7) M
(lower, 68, 21,19) M
(lower, 70, 7,33) F
(lower, 70, 16,25) O
(lower, 70, 22,19) M
(lower, 70, 36,13) O
(upper, 70, 39,14) J
(upper, 70, 43,12) K
(lower, 70, 51, 7) O
(lower, 71, 28,17) M
(upper, 72, 34,18) K
(upper, 72, 52, 8) K
(lower, 73, 17,25) M
(upper, 73, 38,16) K
(lower, 73, 49, 9) V
(lower, 73, 60, 5) Q
(lower, 74, 16,27) M
(lower, 74, 25,19) M
(lower, 74, 42,11) M
(upper, 75, 59, 6) K
(upper, 76, 34,20) K
(lower, 77, 24,21) M
(lower, 78, 13,29) M
(lower, 79, 40,15) L
(lower, 79, 46,11) X
(lower, 80, 12,32) X
(upper, 81, 45,16) K
(upper, 82, 43,18) K
(lower, 83, 33,19) M
(upper, 83, 40,20) K
(lower, 83, 63, 7) Q
(lower, 85, 16,32) X
(lower, 85, 17,29) X
(upper, 85, 45,18) J
(upper, 85, 64, 8) K
(lower, 88, 30,24) L
(lower, 89, 11,40) S
(lower, 89, 23,28) X
(lower, 89, 45,17) L
(upper, 89, 45,20) K
(upper, 89, 64,11) J
(upper, 90, 58,14) K
(upper, 90, 61,12) K
(lower, 90, 63, 9) Q
(upper, 90, 65,10) K
(upper, 90, 78, 4) J
(lower, 91, 13,36) T
(upper, 92, 75, 6) J
(lower, 95, 61,11) X
(lower, 96, 18,32) M
(lower, 96, 55,13) Q
(upper, 97, 64,14) K
(upper, 97, 71,10) K
(lower,101, 7,49) F
(lower,103, 21,31) X
(lower,103, 28,27) X
(lower,103, 35,23) X
(lower,103, 52,19) L
(lower,105, 43,21) X
(lower,107, 54,19) L
(lower,111, 15,43) X
(lower,111, 35,27) X
(lower,111, 42,23) X
(lower,113, 30,31) X
(lower,115, 12,47) X
(lower,116, 7,57) F
(lower,117, 20,43) X
(lower,118, 8,55) X
(lower,119, 7,59) F
(upper,126,113, 4) K
(lower,127, 8,63) I
(lower,127, 15,55) I
(lower,127, 22,47) I
(lower,127, 29,43) I
(lower,127, 43,31) W
(lower,127, 50,27) I
(lower,127, 57,23) I
(lower,127, 64,21) I
(lower,127, 71,19) I
(lower,127, 78,15) I
(lower,127, 85,13) I
(lower,127, 92,11) I
(lower,127, 99, 9) I
(lower,127,106, 7) I
(lower,127,113, 5) I
(lower,127,120, 3) I

External Improvements Used for Updated Table

(lower, 45, 6,22) SS
(lower, 73, 6,36) SS
(lower, 77, 6,38) SS
(lower,105, 7,52) SS
(lower,109, 7,54) SS
(upper, 28, 6,12) BM
(upper, 40, 6,18) BM
(lower, 26, 9, 9) Ha
(lower, 30, 12, 9) Ha
(lower, 34, 15, 9) Ha
(lower, 47, 25, 9) Ha
(lower, 72, 53, 7) Ha
(lower, 86, 66, 7) Ha
(lower, 41, 30, 5) Ha
(lower, 92, 7,45) Bv
(upper, 61, 7,29) Lc
(upper, 29, 6,13) Lc
(upper,103, 8,49) Lc
(upper,106, 8,51) Lc
(upper,110, 8,53) Lc
(upper,113, 8,55) Lc
(upper,118, 8,57) Lc
(upper,121, 8,59) Lc
(upper,125, 8,61) Lc
(lower, 80, 47,11) Su
(lower, 86, 46,13) Su
(lower, 76, 50, 9) Ka
(lower, 76, 44,11) Ka
(lower, 73, 38,13) Ka
(lower, 72, 17,25) Ka
(lower, 73, 9,31) Ka
(lower, 76, 17,27) Ka
(lower, 80, 15,29) Ka
(lower, 76, 9,33) Ka
(lower, 45, 6,22) Al
(lower,105, 7,52) Al
(lower,109, 7,54) Al
(lower, 93, 7,46) Al
(lower, 87, 7,42) Al
(lower, 90, 7,44) Al
(lower, 81, 7,38) Al
(lower, 56, 7,26) Al
(lower, 93, 8,44) Al
(lower,112, 8,52) Al
(lower,120, 9,56) Al
(lower,120, 8,57) Al
(lower, 70, 9,32) We
(lower, 88, 12,36) We
(lower, 96, 12,40) We
(lower, 96, 20,32) We
(lower,104, 12,44) We
(lower,104, 16,40) We
(lower,104, 20,36) We
(lower,104, 24,32) We
(lower,112, 12,48) We
(lower,112, 16,44) We
(lower,112, 20,40) We
(lower,112, 24,36) We
(lower,120, 12,52) We
(lower,120, 16,48) We
(lower,128, 16,52) We
(lower,128, 32,36) We
(lower, 73, 27,20) PT
(lower, 73, 36,16) PT
(lower, 85, 12,34) PT
(lower, 85, 20,28) PT
(lower, 87, 31,22) PT
(lower, 89, 56,11) PT
(lower, 91, 51,14) PT
(lower, 93, 20,32) PT
(lower, 93, 23,29) PT
(lower, 93, 31,24) PT
(lower, 93, 33,22) PT
(lower, 74, 7,35) Fa
(lower, 78, 7,37) Fa
(lower, 81, 7,39) Fa
(lower, 86, 7,41) Fa
(lower, 92, 7,45) Fa
(lower,108, 7,53) Fa
(lower, 35, 7,16) T1
(lower, 42, 7,19) T1
(lower, 80, 8,37) T1
(lower, 96, 8,46) T1
(lower,112, 8,54) T1
(lower, 27, 10, 9) Pi
(upper, 74, 7,35) T2
(upper, 78, 7,37) T2
(upper, 81, 7,39) T2
(upper, 86, 7,41) T2
(upper, 89, 7,43) T2
(lower, 24, 7,10) T3
(lower, 32, 7,14) T3
(upper, 31, 7,13) T3
(upper, 34, 7,15) T3
(lower, 39, 7,17) T3
(lower, 47, 7,22) T3
(upper, 55, 7,25) T3
(upper, 58, 7,27) T3
(lower, 51, 9,21) Wi
(lower, 45, 10,17) Pu
(lower, 51, 9,21) Pu
(lower, 52, 10,21) Pu
(lower, 80, 14,32) Pu
(lower, 79, 9,35) Pu
(lower, 94, 9,41) Pu
(lower,107, 11,45) Pu
(lower,109, 9,49) Pu
(lower,120, 9,56) Pu
(lower, 99, 65,11) Ro
(lower,101, 60,13) Ro
(lower,105, 57,15) Ro
(lower, 59, 7,28) He

Appendix II. Lists of External Improvements

```
Bound Updater [1.1], MaxBtIndex = 127

Table of bounds read from file: (WSADDIOE1)HELGERTSTINAFF/BOUNDS ON USER2
    5590 unsolved cases

All blocks printed.
Considering bound (lower, 45,  6,22) "SS"
  d_max( 45,  6) = 22 "SS" a1
  d_max( 44,  6) = 21 "  " a1
  d_max( 77,  6) = 38 "C " a2
  d_max( 76,  6) = 37 "  " a2
  d_max( 90,  7) = 44 "D " a2
  (PT = 0.10)
Considering bound (lower, 73,  6,36) "SS"
  d_max( 73,  6) = 36 "SS" a1
  d_max( 72,  6) = 35 "  " a1
  (PT = 0.02)
Considering bound (lower, 77,  6,38) "SS": USELESS  (PT = 0.00)
Considering bound (lower,105,  7,52) "SS"
  d_max(105,  7) = 52 "SS" a1
  d_max(104,  7) = 51 "  " a1
  (PT = 0.04)
Considering bound (lower,109,  7,54) "SS"
  d_max(109,  7) = 54 "SS" a1
  d_max(108,  7) = 53 "  " a1
  (PT = 0.02)
Considering bound (upper, 28,  6,12) "BM"
  d_max( 28,  6) = 12 "BM" a1
  d_max( 29,  6) = 13 "  " a1
  d_max( 53,  7) = 24 "E " a2
  d_max(102,  8) = 48 "E " a3
  d_max( 57, 10) = 24 "B " a3
  d_max( 58, 11) = 24 "  " a3
  d_max( 63, 15) = 24 "B " a4
  d_max( 64, 16) = 24 "  " a4
  d_max( 32,  9) = 12 "B " a2
  d_max( 33, 10) = 12 "  " a2
  Bound (upper,34,11,12) "R ": external reference removed
  (PT = 0.32)
Considering bound (upper, 40,  6,18) "BM"
  d_max( 40,  6) = 18 "BM" a1
  d_max( 41,  6) = 19 "  " a1
* DIMENSION 6 HAS BEEN COMPLETELY DETERMINED FOR LENGTH <= 127
  d_max( 77,  7) = 36 "E " a2
  d_max( 80,  7) = 38 "E " a2
  Bound (upper,66,29,18) "K ": external reference removed
  (PT = 0.20)
Considering bound (lower, 26,  9, 9) "Ha"
  d_max( 27,  9) = 10 "  " a1
  d_max( 26,  8) = 10 "  " a1
  (PT = 0.06)
Considering bound (lower, 30, 12, 9) "Ha"
  d_max( 31, 12) = 10 "  " a1
  (PT = 0.04)
Considering bound (lower, 34, 15, 9) "Ha"
  Bound (lower,33,13,10) "N ": external reference removed
  (PT = 0.07)
Considering bound (lower, 47, 25, 9) "Ha": useful  (PT = 0.04)
Considering bound (lower, 72, 53, 7) "Ha"
  Bound (lower,70,51,7) "O ": external reference removed
  d_max( 73, 53) =  8 "  " a1
  d_max( 72, 52) =  8 "  " a1
  (PT = 0.05)
Considering bound (lower, 86, 66, 7) "Ha"
  Bound (lower,83,63,7) "Q ": external reference removed
  d_max( 87, 66) =  8 "  " a1
  d_max( 86, 65) =  8 "  " a1
  d_max( 85, 64) =  8 "  " a1
  (PT = 0.03)
Considering bound (lower, 41, 30, 5) "Ha"
  Bound (lower,39,28,5) "P ": external reference removed
  d_max( 42, 30) =  6 "  " a1
  d_max( 41, 29) =  6 "  " a1
  (PT = 0.05)
Considering bound (lower, 92,  7,45) "Bv"
  d_max( 92,  7) = 45 "Bv" a1
  d_max( 93,  7) = 46 "  " a1
  (PT = 0.03)
Considering bound (upper, 61,  7,29) "Lc"
  d_max( 61,  7) = 29 "Lc" a1
  d_max( 60,  7) = 28 "  " a1
  (PT = 0.05)
Considering bound (upper, 29,  6,13) "Lc": USELESS  (PT = 0.00)
Considering bound (upper,103,  8,49) "Lc": USELESS  (PT = 0.00)
Considering bound (upper,106,  8,51) "Lc": USELESS  (PT = 0.00)
Considering bound (upper,110,  8,53) "Lc": USELESS  (PT = 0.00)
Considering bound (upper,113,  8,55) "Lc": USELESS  (PT = 0.00)
Considering bound (upper,118,  8,57) "Lc": USELESS  (PT = 0.00)
Considering bound (upper,121,  8,59) "Lc": USELESS  (PT = 0.00)
Considering bound (upper,125,  8,61) "Lc"
  d_max(125,  8) = 61 "Lc" a1
  d_max(124,  8) = 60 "  " a1
  (PT = 0.02)
Considering bound (lower, 80, 47,11) "Su"
  Bound (lower,79,46,11) "X ": external reference removed
  (PT = 0.03)
Considering bound (lower, 86, 46,13) "Su": useful  (PT = 0.02)
Considering bound (lower, 76, 50, 9) "Ka": useful  (PT = 0.01)
Considering bound (lower, 76, 44,11) "Ka"
  Bound (lower,74,42,11) "M ": external reference removed
  (PT = 0.03)
Considering bound (lower, 73, 38,13) "Ka": useful  (PT = 0.03)
Considering bound (lower, 72, 17,25) "Ka"
  Bound (lower,73,17,25) "M ": improved to 26
  (PT = 0.02)
Considering bound (lower, 73,  9,31) "Ka": useful  (PT = 0.02)
Considering bound (lower, 76, 17,27) "Ka": useful  (PT = 0.03)
Considering bound (lower, 80, 15,29) "Ka"
  Bound (lower,78,13,29) "M ": external reference removed
  (PT = 0.03)
Considering bound (lower, 76,  9,33) "Ka": useful  (PT = 0.07)
Considering bound (lower, 45,  6,22) "Al": USELESS  (PT = 0.00)
Considering bound (lower,105,  7,52) "Al": USELESS  (PT = 0.00)
Considering bound (lower,109,  7,54) "Al": USELESS  (PT = 0.00)
Considering bound (lower, 93,  7,46) "Al": USELESS  (PT = 0.00)
Considering bound (lower, 87,  7,42) "Al"
  d_max( 87,  7) = 42 "Al" a1
  (PT = 0.03)
```

```
Considering bound (lower, 90,  7,44) "Al": USELESS  (PT = 0.00)
Considering bound (lower, 81,  7,38) "Al": USELESS  (PT = 0.00)
Considering bound (lower, 56,  7,28) "Al"
  d_max( 56,  7) = 26 "Al" a1
  (PT = 0.08)
Considering bound (lower, 93,  8,44) "Al"
  d_max( 93,  8) = 44 "Al" a1
  (PT = 0.03)
Considering bound (lower,112,  8,52) "Al": USELESS  (PT = 0.00)
Considering bound (lower,120,  9,56) "Al"
  Bound (lower,118,8,55) "X ": external reference removed
  (PT = 0.03)
Considering bound (lower,120,  8,57) "Al": useful  (PT = 0.02)
Considering bound (lower, 70,  9,32) "We"
  d_max( 70,  9) = 32 "We" a1
  d_max( 69,  8) = 32 "  " a1
  d_max( 71,  9) = 32 "  " a1
  d_max( 70,  8) = 32 "  " a1
  d_max( 72,  9) = 32 "  " a1
  Bound (lower,73,9,31) "Ka": improved to 32
  (PT = 0.06)
Considering bound (lower, 88, 12,36) "We": useful  (PT = 0.02)
Considering bound (lower, 96, 12,40) "We": useful  (PT = 0.04)
Considering bound (lower, 96, 20,28) "We"
  Bound (lower,96,18,32) "M ": external reference removed
  (PT = 0.09)
Considering bound (lower,104, 12,44  "We": useful  (PT = 0.05)
Considering bound (lower,104, 16,40) "We": useful  (PT = 0.10)
Considering bound (lower,104, 20,36) "We": useful  (PT = 0.09)
Considering bound (lower,104, 24,32) "We"
  Bound (lower,103,21,31) "X ": improved to 32
  (PT = 0.08)
Considering bound (lower,112, 12,48) "We": useful  (PT = 0.05)
Considering bound (lower,112, 16,44) "We"
  Bound (lower,111,15,43) "X ": improved to 44
  (PT = 0.08)
Considering bound (lower,112, 20,40) "We": useful  (PT = 0.06)
Considering bound (lower,112, 24,36) "We": useful  (PT = 0.05)
Considering bound (lower,120, 12,50) "We": useful  (PT = 0.03)
Considering bound (lower,120, 16,48) "We": useful  (PT = 0.05)
Considering bound (lower,128, 16,52) "We": useful  (PT = 0.01)
Considering bound (lower,128, 32,36) "We": useful  (PT = 0.03)
Considering bound (lower, 73, 27,20) "PT"
  Bound (lower,66,20,19) "M ": improved to 20
  Bound (lower,68,21,19) "M ": improved to 20
  Bound (lower,70,22,19) "M ": improved to 20
  Bound (lower,74,25,19) "M ": improved to 20
  (PT = 0.16)
Considering bound (lower, 73, 36,15) "PT"
  Bound (lower,66,30,15) "M ": external reference removed
  Bound (lower,70,36,13) "O ": external reference removed
  (PT = 0.10)
Considering bound (lower, 85, 12,34) "PT": useful  (PT = 0.02)
Considering bound (lower, 85, 20,28) "PT": useful  (PT = 0.03)
Considering bound (lower, 87, 31,22) "PT": useful  (PT = 0.04)
Considering bound (lower, 89, 56,11) "PT"
  Bound (lower,80,47,11) "Su": external reference removed
  (PT = 0.06)
Considering bound (lower, 91, 51,14) "PT"
  Bound (lower,86,46,13) "Su": improved to 14
  (PT = 0.08)
Considering bound (lower, 93, 20,32) "PT"
  Bound (lower,96,20,32) "We": external reference removed
  (PT = 0.08)
Considering bound (lower, 93, 23,29) "PT": useful  (PT = 0.06)
Considering bound (lower, 93, 31,24) "PT": useful  (PT = 0.03)
Considering bound (lower, 93, 33,22) "PT": useful  (PT = 0.03)
Considering bound (lower, 74,  7,35) "Fa"
  d_max( 75,  7) = 36 "  " a1
  (PT = 0.03)
Considering bound (lower, 78,  7,37) "Fa"
  d_max( 78,  7) = 37 "Fa" a1
  d_max( 79,  7) = 38 "  " a1
  (PT = 0.02)
Considering bound (lower, 81,  7,39) "Fa"
  d_max( 81,  7) = 39 "Fa" a1
  d_max( 82,  7) = 40 "  " a1
  (PT = 0.03)
Considering bound (lower, 86,  7,41) "Fa": USELESS  (PT = 0.00)
Considering bound (lower, 92,  7,45) "Fa": USELESS  (PT = 0.00)
Considering bound (lower,108,  7,53) "Fa": USELESS  (PT = 0.00)
Considering bound (lower, 35,  7,16) "T1"
  d_max( 35,  7) = 16 "T1" a1
  (PT = 0.05)
Considering bound (lower, 42,  7,19) "T1"
  d_max( 42,  7) = 19 "T1" a1
  d_max( 41,  7) = 18 "  " a1
  d_max( 43,  7) = 20 "  " a1
  d_max( 44,  7) = 20 "  " a1
  (PT = 0.08)
Considering bound (lower, 80,  8,37) "T1"
  d_max( 81,  8) = 38 "  " a1
  (PT = 0.04)
Considering bound (lower, 96,  8,46) "T1"
  d_max( 96,  8) = 46 "T1" a1
  (PT = 0.04)
Considering bound (lower,112,  8,54) "T1"
  d_max(112,  8) = 54 "T1" a1
  d_max( 59,  7) = 28 "  " a2
  (PT = 0.10)
Considering bound (lower, 27, 10, 9) "Pi"
  Bound (lower,26,9,9) "Ha": external reference removed
  d_max( 28, 10) = 10 "  " a1
  (PT = 0.04)
Considering bound (upper, 74,  7,35) "T2"
  d_max( 74,  7) = 35 "T2" a1
  d_max( 73,  7) = 34 "  " a1
  (PT = 0.06)
Considering bound (upper, 78,  7,37) "T2": USELESS  (PT = 0.00)
Considering bound (upper, 81,  7,39) "T2": USELESS  (PT = 0.00)
Considering bound (upper, 86,  7,41) "T2"
  d_max( 86,  7) = 41 "T2" a1
  d_max( 85,  7) = 40 "  " a1
  d_max( 89, 10) = 40 "B " a2
  (PT = 0.07)
Considering bound (upper, 89,  7,43) "T2"
  d_max( 89,  7) = 43 "T2" a1
  d_max( 88,  7) = 42 "  " a1
```

```
  (PT = 0.06)
Considering bound (lower, 24,  7,10) "T3"
  d_max( 24,  7) = 10 "T3" a1
  d_max( 23,  7) =  9 "  " a1
* LENGTH 23 HAS BEEN COMPLETELY DETERMINED
  Bound (lower,22,6,9) "G ": external reference removed
  (PT = 0.03)
Considering bound (lower, 32,  7,14) "T3"
  d_max( 32,  7) = 14 "T3" a1
  (PT = 0.03)
Considering bound (upper, 31,  7,13) "T3"
  d_max( 31,  7) = 13 "T3" a1
  d_max( 30,  7) = 12 "  " a1
  d_max( 31,  8) = 12 "  " a1
  d_max( 55,  8) = 24 "E " a2
  d_max( 56,  9) = 24 "  " a2
  d_max(104,  9) = 48 "E " a3
  d_max( 61, 13) = 24 "B " a3
  d_max( 62, 14) = 24 "  " a3
  d_max( 36, 12) = 12 "B " a2
  d_max( 37, 13) = 12 "  " a2
  d_max( 38, 14) = 12 "  " a2
  d_max( 39, 15) = 12 "  " a2
  d_max( 46, 21) = 12 "  " a3
  d_max( 47, 22) = 12 "  " a3
  (PT = 0.49)
Considering bound (upper, 34,  7,15) "T3"
  d_max( 34,  7) = 15 "T3" a1
  d_max( 33,  7) = 14 "  " a1
  Bound (upper,34,8,14) "R ": external reference removed
  d_max( 62,  8) = 28 "E " a2
  (PT = 0.18)
Considering bound (lower, 39,  7,17) "T3"
  d_max( 39,  7) = 17 "T3" a1
  d_max( 40,  7) = 18 "  " a1
  (PT = 0.05)
Considering bound (lower, 47,  7,22) "T3"
  d_max( 47,  7) = 22 "T3" a1
  d_max( 46,  7) = 21 "  " a1
  (PT = 0.04)
Considering bound (upper, 55,  7,25) "T3"
  d_max( 55,  7) = 25 "T3" a1
  d_max( 54,  7) = 24 "  " a1
  d_max(103,  8) = 48 "E " a2
  d_max( 58, 10) = 24 "B " a2
  d_max( 59, 11) = 24 "  " a2
  d_max( 60, 12) = 24 "  " a2
  d_max( 64, 15) = 24 "B " a3
  d_max( 65, 16) = 24 "  " a3
  d_max( 66, 17) = 24 "  " a3
  d_max( 67, 18) = 24 "  " a3
  (PT = 0.26)
Considering bound (upper, 58,  7,27) "T3"
  d_max( 58,  7) = 27 "T3" a1
  d_max( 57,  7) = 26 "  " a1
* DIMENSION 7 HAS BEEN COMPLETELY DETERMINED FOR LENGTH <= 127
  d_max(110,  8) = 52 "E " a2
  d_max(111,  8) = 53 "  " a2
  d_max(113,  8) = 54 "E " a2
  (PT = 0.17)
Considering bound (lower, 51,  9,21) "Wi": useful  (PT = 0.04)
Considering bound (lower, 45, 10,17) "Pu": useful  (PT = 0.07)
Considering bound (lower, 51,  9,21) "Pu": USELESS  (PT = 0.00)
Considering bound (lower, 52, 10,21) "Pu"
  Bound (lower,51,9,21) "Wi": external reference removed
  (PT = 0.04)
Considering bound (lower, 80, 14,32) "Pu"
  Bound (lower,80,12,32) "X ": external reference removed
  (PT = 0.10)
Considering bound (lower, 79,  9,35) "Pu": useful  (PT = 0.03)
Considering bound (lower, 94,  9,41) "Pu": useful  (PT = 0.02)
Considering bound (lower,107, 11,45) "Pu": useful  (PT = 0.03)
Considering bound (lower,109,  9,49) "Pu": useful  (PT = 0.02)
Considering bound (lower,120,  9,56) "Pu": USELESS  (PT = 0.00)
Considering bound (lower, 99, 65,11) "Ro"
  Bound (lower,95,61,11) "X ": external reference removed
  (PT = 0.02)
Considering bound (lower,101, 60,13) "Ro"
  Bound (lower,96,55,13) "Q ": external reference removed
  (PT = 0.02)
Considering bound (lower,105, 57,15) "Ro": useful  (PT = 0.02)
Considering bound (lower, 59,  7,28) "He": USELESS  (PT = 0.00)
All (13) blocks with unprinted modifications have been printed.

Cumulative Session Statistics:
   18.68 process time
     112 updates
      89 useful updates
     654 lower bounds improved
     370 upper bounds improved
      22 external references removed
       9 external references superseded
     110 cases solved
    5480 cases left open
Table of bounds saved on file: (WSADDIOE1)NEW/BOUNDS ON USER2

      34 pages printed
Bound Updater Terminated.
```

```
10000    $$ RESET LIST                                                          00010000
10010    $$ SET   LISTINCL                                                      00010010
10020    $$ SET   WARNSUPR                                                      00010020
10030    $$ SET   NOBOUNDS                                                      00010030
10040    $$ RESET TESTOUTPUT                                                    00010040
10050    $$ RESET LSIPRINTER                                                    00010050
10060                                                                           00010060
11000    PROGRAM BoundUpdater (                                                 00011000
11010      input,                                                               00011010
11020      output,                                                              00011020
11030      Pr      : FILE <kind=printer,trainid=EBCDIC96,pagesize=63>,          00011030
11040      Report  : FILE <kind=printer,trainid=EBCDIC96>,                      00011040
11050      List    : FILE <kind=printer,trainid=EBCDIC96>                       00011050
11060      ) ;                                                                  00011060
11070      (*****************************************************************  00011070
11080       *                                                               *  00011080
11090       *  This program maintains a table of known upper and lower      *  00011090
11100       *  bounds on the maximum minumum-distance of binary linear      *  00011100
11110       *  codes with word length less than 128.                        *  00011110
11120       *                                                               *  00011120
11130       *                                                               *  00011130
11140       *  Author: Tom Verhoeff (student)                               *  00011140
11150       *                                                               *  00011150
11160       *          Department of Mathematics and Computing Science,     *  00011160
11170       *          Eindhoven University of Technology.                  *  00011170
11180       *                                                               *  00011180
11190       *  Date:   January 1984 (revised September 1984)                *  00011190
11200       *                                                               *  00011200
11210       *****************************************************************) 00011210
11220                                                                           00011220
12000    CONST                                                                  00012000
12010      Version             = '1.1' ;                                        00012010
12020      MaxBtIndex          =  127  ; { maximum bound table index }          00012020
12030      BtSize              =16384  ; { size in words = sqr(MaxBtIndex+1) }  00012030
12040                                                                           00012040
12050      NoRef               =  '  ' ; { implicit reference }                00012050
12060      A_Ref               =  'A ' ;                                        00012060
12070      B_Ref               =  'B ' ;                                        00012070
12080      C_Ref               =  'C ' ;                                        00012080
12090      D_Ref               =  'D ' ;                                        00012090
12100      E_Ref               =  'E ' ;                                        00012100
12110                                                                           00012110
12120      MaxPage             = 1000  ; { maximum # pages printed to 1 file }  00012120
12130                                                                           00012130
12140    TYPE                                                                   00012140
12150      BoundKind           = ( lower, upper ) ;                             00012150
12160      BtIndex             = 0 .. MaxBtIndex ;                              00012160
12170      Reference           = PACKED ARRAY [ 1..2 ] OF char ;                00012170
12180      BtElementKind       = ( Statistics, BoundInfo ) ;                    00012180
12190                                                                           00012190
12200      BtElement           = PACKED RECORD CASE BtElementKind OF            00012200
12210        Statistics        : (                                              00012210
12220          unsolved        : integer ) ;                                    00012220
12230        BoundInfo         : (                                              00012230
12240          dist            : BtIndex ;                                      00012240
12250          refch1, refch2  : char ; { not ref: Reference because of packing } 00012250
```

```
12260         mark           : 0 .. BtSize ;                                  00012260
12270         unchanged      : boolean ) ;                                    00012270
12280       END { BtElement } ;                                               00012280
12290                                                                        00012290
12300     BtArray            = ARRAY [ BtIndex, BtIndex ] OF BtElement ;      00012300
12310                                                                        00012310
12320     BtFile             = FILE OF BtArray ;                              00012320
12330                                                                        00012330
13000   VAR                                                                  00013000
13010     Bt                 : BtArray ;                                      00013010
13020                          { if 0 < k <= n <= MaxBtIndex then             00013020
13030                              Bt[n,k].dist = current lower bound,        00013030
13040                              Bt[n,k].refch1..2 corresponding reference, 00013040
13050                              Bt[k,n].dist = current upper bound,        00013050
13060                              Bt[k,n].refch1..2 corresponding reference, 00013060
13070                              N.B. Bt[n,n].dist = 1 = lower = upper      00013070
13080                                   Bt[n,n].refch1..2 = NoRef (P1)        00013080
13090                              Bt[n,0].unsolved = # k: case (n,k) is open, 00013090
13100                              Bt[0,k].unsolved = # n: case (n,k) is open, 00013100
13110                              Bt[0,0].unsolved = # n,k: case (n,k) is open 00013110
13120                           }                                            00013120
13130     Id                 : ARRAY [ BoundKind ] OF                         00013130
13140                            PACKED ARRAY [ 1 .. 5 ] OF char ;           00013140
13150     Pr, Report, List   : TEXT ;                                         00013150
13160                                                                        00013160
13170     { session statistics }                                             00013170
13180     BtModified           : boolean ; { Bt modified after last save }   00013180
13190     TotUpdates,                                                        00013190
13200     TotUseful,                                                         00013200
13210     TotExtWiped,                                                       00013210
13220     TotExtImproved,                                                    00013220
13230     Bt00               : integer ;                                     00013230
13240     TotImproved        : ARRAY [ BoundKind ] OF integer ;              00013240
13250     PrevPt             : real ;                                        00013250
13260     PagesPrinted       : integer ;                                     00013260
13270     ToBePrinted        : ARRAY [0..4,0..6] OF boolean ;                00013270
13280                          { identifies modified unprinted blocks }      00013280
13290                                                                        00013290
15000   FUNCTION IsOpenCase(n,k: BtIndex): boolean ;                         00015000
15010     BEGIN { 1 <= k <= n  assumed }                                     00015010
15020     IsOpenCase := (Bt[n,k].dist < Bt[k,n].dist) ;                      00015020
15030     END { IsOpenCase } ;                                               00015030
15040                                                                        00015040
15050   FUNCTION HasExplicitRef(b: BoundKind; n,k: BtIndex): boolean ;       00015050
15060     BEGIN { 1 <= k <= n  assumed }                                     00015060
15070     CASE b OF                                                          00015070
15080       lower: WITH Bt[n,k] DO                                           00015080
15090                HasExplicitRef := (refch1 <> ' ') OR (refch2 <> ' ') ;  00015090
15100       upper: WITH Bt[k,n] DO                                           00015100
15110                HasExplicitRef := (refch1 <> ' ') OR (refch2 <> ' ') ;  00015110
15120       END { case } ;                                                   00015120
15130     END { HasExplicitRef } ;                                           00015130
15140                                                                        00015140
15150   FUNCTION HasExternalRef(b: BoundKind; n,k: BtIndex): boolean ;       00015150
15160     BEGIN { 1 <= k <= n  assumed }                                     00015160
15170       CASE b OF                                                        00015170
15180       lower: WITH Bt[n,k] DO                                           00015180
15190         HasExternalRef :=                                              00015190
15200           NOT (refch1 in [' ','A'..'E']) OR (refch2 <> ' ') ;          00015200
```

```
15210       upper: WITH Bt[k,n] DO                                                    00015210
15220         HasExternalRef :=                                                       00015220
15230           NOT (refch1 in [' ','A'..'E']) OR (refch2 <> ' ') ;                   00015230
15240       END { case } ;                                                            00015240
15250     END { HasExternalRef } ;                                                    00015250
15260                                                                                 00015260
15270   PROCEDURE AssignRef(b: BoundKind; n,k: BtIndex; r: Reference);                00015270
15280     BEGIN                                                                       00015280
15290     CASE b OF                                                                   00015290
15300       lower: WITH Bt[n,k] DO BEGIN                                              00015300
15310                refch1 := r[1] ; refch2 := r[2] END ;                          00015310
15320       upper: WITH Bt[k,n] DO BEGIN                                              00015320
15330                refch1 := r[1] ; refch2 := r[2] END ;                          00015330
15340       END { case } ;                                                            00015340
15350     END { AssignRef } ;                                                         00015350
15360                                                                                 00015360
15370   FUNCTION IsBlock(i,j: integer): boolean ;                                     00015370
15380     BEGIN                                                                       00015380
15390     IsBlock := (0 <= i) AND (i <= 4) AND (0 <= j) AND (j <= 6) AND              00015390
15400                (j*20+1 <= (i+1)*25) ;                                           00015400
15410     END { IsBlock } ;                                                         00015410
15420                                                                                 00015420
15430   FUNCTION ToUpper(c: char): char ;                                           00015430
15440     BEGIN { N.B. EBCDIC character set }                                       00015440
15450     IF c IN ['a'..'i', 'j'..'r', 's'..'z'] THEN                               00015450
15460         ToUpper := chr(ord(c) - ord('a') + ord('A'))                          00015460
15470       ELSE                                                                    00015470
15480         ToUpper := c ;                                                        00015480
15490     END { ToUpper } ;                                                         00015490
15500                                                                               00015500
15510   PROCEDURE SayProcTime ;                                                     00015510
15520     VAR pt, delta: real ;                                                     00015520
15530     BEGIN                                                                     00015530
15540     pt := RUNTIME ; delta := pt - PrevPt ; PrevPt := pt ;                     00015540
15550     writeln('PT = ',pt:1:2,', delta PT = ',delta:1:2) ;                       00015550
15560     END { SayProcTime } ;                                                     00015560
15570                                                                               00015570
19000   PROCEDURE SaveBt ;                                                          00019000
19010     VAR nbf: BtFile ; t: STRING[80] ;                                         00019010
19020     BEGIN                                                                     00019020
19030     writeln(Report) ;                                                         00019030
19040     writeln(Report,'Cumulative Session Statistics:') ;                        00019040
19050     writeln(Report,RUNTIME:8:2,' process time') ;                             00019050
19060     writeln(Report,TotUpdates:8,' updates') ;                                 00019060
19070     IF BtModified THEN BEGIN                                                  00019070
19080       writeln(Report,TotUseful:8,' useful updates') ;                         00019080
19090       writeln(Report,TotImproved[lower]:8,' lower bounds improved') ;         00019090
19100       writeln(Report,TotImproved[upper]:8,' upper bounds improved') ;         00019100
19110       writeln(Report,TotExtWiped:8,' external references removed') ;          00019110
19120       writeln(Report,TotExtImproved:8,' external references superseded') ;    00019120
19130       writeln(Report,Bt00-Bt[0,0].unsolved:8,' cases solved') ;               00019130
19140       writeln(Report,Bt[0,0].unsolved:8,' cases left open') ;                 00019140
19150       SETATTRIBUTE(nbf, AREASIZE, 1) ;                                        00019150
19160       SETATTRIBUTE(nbf, AREAS,    1) ;                                        00019160
19170       rewrite(nbf) ; GETATTRIBUTE(nbf,TITLE,t) ;                              00019170
19180       nbf^ := Bt ; put(nbf) ; CLOSE(nbf,SAVE) ;                               00019180
19190       writeln(Report,'Table of bounds saved on file: ',t) ;                   00019190
19200       BtModified := false ;                                                   00019200
19210       END ELSE writeln(Report,'Table not modified since last save.') ;        00019210
```

```
19220     END { SaveBt } ;                                                                00019220
19230                                                                                     00019230
20000   $$ INCLUDE "UPDATEBT"                                                           00020000
30000   $$ INCLUDE "PRINTBT"                                                            00030000
70000   PROCEDURE ProcessInput ;                                                        00070000
70010     VAR ch,dummy: char ;                                                          00070010
70020         b: BoundKind ;                                                            00070020
70030         n,k,d,ior: integer ;                                                      00070030
70040         r: Reference ;                                                            00070040
70050     BEGIN                                                                         00070050
70060     REPEAT { until (ch = 'Q') }                                                   00070060
70070       SayProcTime ;                                                               00070070
70080       REPEAT { until (ior = IORES(OK)) }                                          00070080
70090         writeln('Give command: L, U, S, P, M, A, W, Q:') ;                        00070090
70100         IF eoln THEN readln ;                                                     00070100
70110         REPEAT read(ch) UNTIL (ch <> ' ') ; ch := ToUpper(ch) ;                   00070110
70120         CASE ch OF                                                                00070120
70130           'L','U': BEGIN                                                          00070130
70140             CASE ch OF 'L': b := lower ; 'U': b := upper END ;                     00070140
70150             ior := read(n,k,d) ;                                                  00070150
70160             IF (ior = IORES(OK)) THEN BEGIN                                       00070160
70170               REPEAT read(dummy) UNTIL (dummy <> ' ') ; r[1] := dummy ;           00070170
70180               IF eoln THEN dummy := ' ' ELSE read(dummy) ; r[2] := dummy ;        00070180
70190         writeln('Bound = (',Id[b],',',n:1,',',k:1,',',d:1,') "',r,'"') ;          00070190
70200               END { then }                                                        00070200
70210             ELSE writeln('Invalid bound format') ;                              00070210
70220           END { L, U } ;                                                        00070220
70230           'P': BEGIN                                                            00070230
70240             ior := read(n,k) ;                                                  00070240
70250             IF (ior = IORES(OK)) THEN BEGIN                                     00070250
70260               IF NOT IsBlock(n,k) THEN BEGIN ior := IORES(DATAERR) ;            00070260
70270                 writeln('(',n:1,',',k:1,') is not a block') END                 00070270
70280               END { then }                                                      00070280
70290               ELSE writeln('Invalid block coordinate format') ;                 00070290
70300             END { P } ;                                                         00070300
70310           'S','M','A','W','Q': ior := IORES(OK) ;                               00070310
70320           OTHERWISE                                                           00070320
70330             writeln('Unrecognized command: ',ch) ; ior := IORES(DATAERR) ;    00070330
70340           END { case ch } ;                                                    00070340
70350         WHILE NOT eoln DO read(dummy) ;                                       00070350
70360         UNTIL (ior = IORES(OK)) ;                                             00070360
70370       CASE ch OF                                                              00070370
70380         'L','U': UpdateBt(b,n,k,d,r) ;                                        00070380
70390         'S': SaveBt ;                                                         00070390
70400         'P': BEGIN PrintBlock(n,k) ;                                          00070400
70410           writeln(Report, 'Block (',n:1,',',k:1,') printed.') END ;           00070410
70420         'A': PrintAll ;                                                       00070420
70430         'M': PrintModified ;                                                  00070430
70440         'W': WallPaper ;                                                      00070440
70450         'Q': writeln('QUIT command') ;                                        00070450
70460         END { case } ;                                                        00070460
70470       UNTIL (ch = 'Q') ;                                                      00070470
70480     END { ProcessInput } ;                                                    00070480
70490                                                                               00070490
80000   PROCEDURE Initialize ;                                                    00080000
80010     VAR i,j: integer ;                                                      00080010
80020         cbf: BtFile ; t: STRING[80] ;                                       00080020
80030     BEGIN                                                                   00080030
80040     reset(input) ; rewrite(output) ;                                        00080040
```

```
80050     rewrite(Pr) ; rewrite(Report) ; rewrite(List) ;                    00080050
80060     writeln('Bound Updater [',Version,'], MaxBtIndex = ',MaxBtIndex:1) ; 00080060
80070     writeln(Report,                                                    00080070
80080       'Bound Updater [',Version,'], MaxBtIndex = ',MaxBtIndex:1) ;     00080080
80090     writeln(Report) ;                                                  00080090
80100     Id[lower] := 'lower' ; Id[upper] := 'upper' ; { constant array }    00080100
80110     PrevPt := 0.0 ; PagesPrinted := 0 ;                                00080110
80120     BtModified := false ; { no modifications yet }                     00080120
80130     TotUpdates := 0 ; TotUseful := 0 ;                                 00080130
80140     TotImproved[lower] := 0 ; TotImproved[upper] := 0 ;                00080140
80150     TotExtWiped := 0 ; TotExtImproved := 0 ;                           00080150
80160     FOR i := 0 TO 4 DO                                                 00080160
80170       FOR j := 0 TO 6 DO ToBePrinted[i,j] := false ;                   00080170
80180     reset(cbf) ; Bt := cbf^ ;                                          00080180
80190     GETATTRIBUTE(cbf,TITLE,t) ;                                        00080190
80200     writeln(Report,'Table of bounds read from file: ',t) ;             00080200
80210     FOR i := 1 TO MaxBtIndex DO                                        00080210
80220       FOR j := 1 TO MaxBtIndex DO Bt[i,j].unchanged := true ;          00080220
80230     Bt00 := Bt[0,0].unsolved ; { initial number of open cases }        00080230
80240     writeln(Report,Bt00:8,' unsolved cases') ; writeln(Report) ;       00080240
80250     END { Initialize } ;                                               00080250
80260                                                                        00080260
80270   PROCEDURE Finalize ;                                                 00080270
80280     BEGIN                                                            00080280
80290     SaveBt ;                                                         00080290
80300     writeln(Report) ;                                                00080300
80310     writeln(Report,PagesPrinted:8,' pages printed') ;                00080310
80320     writeln(Report,'Bound Updater Terminated.') ;                    00080320
80330     END { Finalize } ;                                               00080330
80340                                                                      00080340
90000   BEGIN { BoundUpdater }                                             00090000
90010   Initialize ;                                                       00090010
90020   ProcessInput ;                                                     00090020
90030   Finalize ;                                                         00090030
90040   END { BoundUpdater }.                                              00090040
90050                                                                      00090050
```

```
20000    PROCEDURE UpdateBt(b: BoundKind;                                        00020000
20010                       len, dim, newdist: integer;                          00020010
20020                       r: Reference) ;                                      00020020
20030      LABEL 0 ; { in case of bound violation }                              00020030
20040                                                                            00020040
20050      VAR                                                                   00020050
20060        NoRemarksYet,                                                       00020060
20070        Lmodified: boolean ; { table modified since last entry of UpdateBt } 00020070
20080        pt: real ;                                                          00020080
20090        depth: integer ;                                                    00020090
20100                                                                            00020100
20110    $ SET OMIT = NOT TESTOUTPUT                                             00020110
20120      PROCEDURE indent ; VAR i: integer ;                                   00020120
20130        BEGIN                                                               00020130
20140        FOR i := 1 TO MIN(25,depth) DO write(Report,'.') ;                  00020140
20150        IF (depth > 25) THEN write(Report,'>') ;                            00020150
20160        END { indent } ;                                                    00020160
20170    $ POP OMIT                                                              00020170
20180                                                                            00020180
20190      FUNCTION IsInTable(n,k: integer): boolean ;                           00020190
20200        BEGIN                                                               00020200
20210        IsInTable := (0 < k) AND (k <= n) AND (n <= MaxBtIndex) ;           00020210
20220        END { IsInTable } ;                                                 00020220
20230                                                                            00020230
20240      FUNCTION IsImprovement(b: BoundKind; n,k,newd: integer): boolean ;    00020240
20250        BEGIN                                                               00020250
20260        IF IsInTable(n,k) THEN                                              00020260
20270            CASE b OF                                                       00020270
20280              lower: IsImprovement := (Bt[n,k].dist < newd) ;                00020280
20290              upper: IsImprovement := (Bt[k,n].dist > newd) ;                00020290
20300              END { case b }                                                00020300
20310          ELSE                                                              00020310
20320            IsImprovement := false ;                                        00020320
20330        END { IsImprovement } ;                                             00020330
20340                                                                            00020340
20350      FUNCTION IsSame(b: BoundKind; n,k,d: integer): boolean ;              00020350
20360        BEGIN                                                               00020360
20370        IF IsInTable(n,k) THEN                                              00020370
20380          CASE b OF                                                         00020380
20390            lower: IsSame := (Bt[n,k].dist = d) ;                           00020390
20400            upper: IsSame := (Bt[k,n].dist = d) ;                           00020400
20410            END { case }                                                    00020410
20420          ELSE IsSame := false ;                                            00020420
20430        END { IsSame } ;                                                    00020430
20440                                                                            00020440
20450      FUNCTION IsMarked(b: BoundKind; n,k: integer): boolean ;              00020450
20460        BEGIN                                                               00020460
20470        IF IsInTable(n,k) THEN                                              00020470
20480            CASE b OF                                                       00020480
20490              lower: IsMarked := (Bt[n,k].mark = depth) ;                   00020490
20500              upper: IsMarked := (Bt[k,n].mark = depth) ;                   00020500
20510              END { case }                                                  00020510
20520          ELSE IsMarked := false ;                                          00020520
20530        END { IsMarked } ;                                                  00020530
20540                                                                            00020540
20550      PROCEDURE EnsureNewLine ;                                             00020550
```

```
20560       BEGIN                                                                00020560
20570       IF NoRemarksYet THEN BEGIN                                           00020570
20580         writeln(Report) ; NoRemarksYet := false END ;                      00020580
20590       END { EnsureNewLine } ;                                              00020590
20600                                                                            00020600
20610     PROCEDURE ToReport(b: BoundKind; n,k: BtIndex) ;                       00020610
20620       BEGIN                                                                00020620
20630       EnsureNewLine ;                                                      00020630
20640       write(Report,'  Bound (',Id[b],',',n:1,',',k:1,',') ;                00020640
20650       CASE b OF                                                            00020650
20660         lower: WITH Bt[n,k] DO                                             00020660
20670                  write(Report,dist:1,') "',refch1,refch2,'": ') ;          00020670
20680         upper: WITH Bt[k,n] DO                                             00020680
20690                  write(Report,dist:1,') "',refch1,refch2,'": ') ;          00020690
20700         END { case } ;                                                     00020700
20710       END { ToReport } ;                                                   00020710
20720                                                                            00020720
20730     PROCEDURE CondWipeRef(b: BoundKind; n,k,d: integer) ;                  00020730
20740       BEGIN                                                                00020740
20750       IF IsSame(b,n,k,d) THEN BEGIN                                        00020750
20760         Lmodified := true ;                                                00020760
20770   $ SET OMIT = TESTOUTPUT                                                  00020770
20780         IF HasExternalRef(b,n,k) THEN                                      00020780
20790   $ POP OMIT                                                               00020790
20800           BEGIN                                                            00020800
20810           TotExtWiped := TotExtWiped + 1 ;                                 00020810
20820           ToReport(b,n,k) ;                                                00020820
20830           writeln(Report,'external reference removed') END ;               00020830
20840         AssignRef(b,n,k,NoRef) ;                                           00020840
20850         END { IsSame } ;                                                   00020850
20860       END { CondWipeRef } ;                                                00020860
20870                                                                            00020870
20880     PROCEDURE WipeMark(b: BoundKind; n,k: integer) ;                       00020880
20890       BEGIN                                                                00020890
20900       IF IsInTable(n,k) THEN                                               00020900
20910         CASE b OF                                                          00020910
20920           lower: WITH Bt[n,k] DO IF (mark = depth) THEN mark := 0 ;        00020920
20930           upper: WITH Bt[k,n] DO IF (mark = depth) THEN mark := 0 ;        00020930
20940           END { case } ;                                                   00020940
20950       END { WipeMark } ;                                                   00020950
20960                                                                            00020960
20970     PROCEDURE ViolationExit                                                00020970
20980                 (b: BoundKind ; n,k,d: integer ; r: Reference) ;          00020980
20990       BEGIN EnsureNewLine ;                                                00020990
21000       write(Report,'>>> ',Id[b],' bound ',d:1,' "',r,'" violates') ;        00021000
21010       CASE b OF                                                            00021010
21020         lower: ToReport(upper,n,k) ;                                       00021020
21030         upper: ToReport(lower,n,k) ;                                       00021030
21040         END { case } ;                                                     00021040
21050       writeln(Report,'(depth = ',depth:1,')') ;                            00021050
21060       writeln('>>>  VIOLATION DETECTED  <<<') ;                            00021060
21070       IF Lmodified THEN BEGIN                                              00021070
21080         writeln(Report,'Internal table assumed corrupted.') ;              00021080
21090         writeln(Report,'Bound Updater terminated, table not saved.') ;     00021090
21100         writeln('Aborted w/o saving.') ;                                   00021100
21110         CLOSE(Pr) ; CLOSE(Report) ; CLOSE(output) ;                        00021110
21120         ABORT END                                                          00021120
21130       ELSE goto 0 ;                                                        00021130
21140       END { ViolationExit } ;                                              00021140
```

```
21150
21160      PROCEDURE MakeImprovement(b: BoundKind ;                                  00021150
21170                                n,k,newd: integer ;                             00021160
21180                                r: Reference ) ;                                00021170
21190        BEGIN { IsImprovement(b,n,k,newd) assumed }                           00021180
21200        IF (newd < Bt[n,k].dist) OR (newd > Bt[k,n].dist) THEN                00021190
21210          ViolationExit(b,n,k,newd,r) ;                                       00021200
21220        Lmodified := true ;                                                   00021210
21230      $ SET OMIT = NOT TESTOUTPUT                                             00021220
21240        indent ;                                                            00021230
21250      $ SET OMIT = TESTOUTPUT                                                 00021240
21260        IF HasExternalRef(b,n,k) THEN                                       00021250
21270      $ POP OMIT  POP OMIT                                                    00021260
21280          BEGIN ToReport(b,n,k) ; writeln(Report,'improved to ',newd:1) ;     00021270
21290            TotExtImproved := TotExtImproved + 1 END ;                        00021280
21300        CASE b OF                                                           00021290
21310          lower: WITH Bt[n,k] DO BEGIN                                      00021300
21320                   dist := newd ;                                           00021310
21330                   TotImproved[b] := TotImproved[b] + ord(unchanged) ;      00021320
21340                   unchanged := false ;                                     00021330
21350                   END { with } ;                                           00021340
21360          upper: WITH Bt[k,n] DO BEGIN                                      00021350
21370                   dist := newd ;                                           00021360
21380                   TotImproved[b] := TotImproved[b] + ord(unchanged) ;      00021370
21390                   unchanged := false ;                                     00021380
21400                   END { with } ;                                           00021390
21410        END { case } ;                                                      00021400
21420        ToBePrinted[MIN(4,(n-1) DIV 25),(k-1) DIV 25] := true ;             00021410
21430        AssignRef(b,n,k,r) ;                                                00021420
21440        IF NOT IsOpenCase(n,k) THEN BEGIN { case solved }                   00021430
21450          EnsureNewLine ;                                                   00021440
21460          writeln(Report,'  d_max(',n:3,',',k:3,') = ',newd:2,             00021450
21470                  ' "',r,'" @',depth:1) ;                                   00021460
21480          WITH Bt[0,0] DO unsolved := unsolved - 1 ;                        00021470
21490          WITH Bt[0,k] DO unsolved := unsolved - 1 ;                        00021480
21500          WITH Bt[n,0] DO unsolved := unsolved - 1 ;                        00021490
21510          IF (Bt[n,0].unsolved = 0) THEN BEGIN                              00021500
21520            writeln(Report,                                                 00021510
21530                    '* LENGTH ',n:1,' HAS BEEN COMPLETELY DETERMINED') ;    00021520
21540            END { if } ;                                                    00021530
21550          IF (Bt[0,k].unsolved = 0) THEN BEGIN                              00021540
21560            writeln(Report,'* DIMENSION ',k:1,                              00021550
21570                    ' HAS BEEN COMPLETELY DETERMINED FOR LENGTH <= ',       00021560
21580                    MaxBtIndex:1) ;                                         00021570
21590            END { if } ;                                                    00021580
21600          END { if case solved } ;                                          00021590
21610        END { MakeImprovement } ;                                           00021600
21620                                                                            00021610
21630      PROCEDURE Update(b: BoundKind; n,k,newd: integer; r: Reference) ;     00021620
21640        FORWARD ;                                                           00021630
21650                                                                            00021640
21660      PROCEDURE zm3_E(n,k,newd: integer) ;                                  00021650
21670        { zero or more P3, followed by E }                                  00021660
21680        VAR nn,dd,kk: integer ;                                             00021670
21690        BEGIN                                                               00021680
21700        IF odd(newd) THEN BEGIN n := n - 1 ; newd := newd - 1 END ;         00021690
21710        dd := 2*newd ; nn := n + dd + 1 ; kk := k + 1 ;                     00021700
21720        WHILE (nn <= MaxBtIndex+5) AND IsMarked(upper,nn,kk) DO BEGIN       00021710
21730          Update(upper,nn,kk,dd,E_Ref) ; { E }                              00021720
                                                                                 00021730
```

```
21740         n := n + 1 ; newd := newd + 1 ; nn := nn + 3 ; dd := dd + 2 ;                00021740
21750         END { while } ;                                                             00021750
21760       END { zm3_E } ;                                                               00021760
21770                                                                                     00021770
21780     PROCEDURE zm3_B3(n,k,newd: integer) ;                                           00021780
21790       { zero or more P3, followed by B3 }                                           00021790
21800       VAR s: integer ;                                                            00021800
21810       FUNCTION looking: boolean ;                                                 00021810
21820         BEGIN { implements conditional AND }                                      00021820
21830         IF (n+s <= MaxBtIndex) THEN                                               00021830
21840             looking := (Bt[n+1-k,n+s].dist > s)                                   00021840
21850           ELSE looking := false                                                   00021850
21860         END { looking } ;                                                         00021860
21870       BEGIN                                                                       00021870
21880       IF odd(newd) THEN BEGIN n := n - 1 ; newd := newd - 1 END ;                  00021880
21890       WHILE IsMarked(upper,n,k) DO BEGIN                                          00021890
21900         s := 1 ;                                                                  00021900
21910         WHILE looking DO s := s + 1 ;                                             00021910
21920         IF (n+s <= MaxBtIndex) THEN                                               00021920
21930           Update(upper,n+s,k+s-1,newd,B_Ref) ; { B3 }                             00021930
21940         n := n + 1 ; newd := newd + 1 ;                                           00021940
21950         END { while } ;                                                           00021950
21960       END { zm3_B3 } ;                                                            00021960
21970                                                                                   00021970
21980     PROCEDURE zm2_zm4_B284(n,k,newd: integer) ;                                   00021980
21990       { zero or more P2 and P4, followed by either B2, or B4 }                    00021990
22000       VAR m, j: integer ;                                                         00022000
22010       BEGIN                                                                       00022010
22020       IF odd(newd) THEN BEGIN n := n - 1 ; newd := newd - 1 END ;                 00022020
22030       WHILE IsMarked(upper,n,k) DO BEGIN                                          00022030
22040         m := n ; j := k ;                                                         00022040
22050         REPEAT                                                                    00022050
22060           Update(lower,m-newd,m-j-newd+1,Bt[m,m-j].dist,B_Ref) ; { B2 }           00022060
22070           Update(upper,m,m-j,Bt[m-j-newd+1,m-newd].dist,B_Ref) ; { B4 }           00022070
22080           m := m + 1 ; j := j + 1 ;                                               00022080
22090           UNTIL NOT IsMarked(upper,m,j) ;                                         00022090
22100         n := n - 1 ;                                                            00022100
22110         END { while } ;                                                         00022110
22120       END { zm2_zm4_B284 } ;                                                      00022120
22130                                                                                 00022130
22140     PROCEDURE B1(n,k,newd: integer) ;                                           00022140
22150       VAR s: integer ;                                                          00022150
22160       BEGIN                                                                     00022160
22170       IF odd(newd) THEN BEGIN n := n + 1 ; newd := newd + 1 END ;               00022170
22180       { determine upper bound on dmax of dual code of (n,k) }                   00022180
22190       IF IsInTable(n,k) THEN s := Bt[n-k,n].dist                                00022190
22200         ELSE IF (k >= MaxBtIndex) THEN s := k + 1                               00022200
22210         ELSE s := Bt[MaxBtIndex-k,MaxBtIndex].dist ;                            00022210
22220       Update(lower,n-s,k-s+1,newd,B_Ref) ; { B1 }                               00022220
22230       END { B1 } ;                                                              00022230
22240                                                                                 00022240
22250     PROCEDURE zm4_C(n,k,newd: integer) ;                                        00022250
22260       { zero or more P4, followed by C }                                        00022260
22270       VAR m: integer ;                                                          00022270
22280       BEGIN                                                                     00022280
22290       WHILE IsMarked(lower,n,k) DO BEGIN                                        00022290
22300         m := k + 1 ;                                                            00022300
22310         REPEAT                                                                  00022310
22320           WHILE (m < MaxBtIndex) AND NOT odd(Bt[m-1,k].dist) DO                 00022320
```

```
22330             m := m + 1 ;                                                  00022330
22340           IF (n+m <= MaxBtIndex+5) THEN                                   00022340
22350             Update(lower,n+m,k,newd+Bt[m,k].dist,C_Ref) ; { Cm }          00022350
22360           m := m + 1 ;                                                    00022360
22370           UNTIL (n+m > MaxBtIndex + 5) ;                                  00022370
22380         n := n - 1 ; k := k - 1 ;                                         00022380
22390         END { while } ;                                                   00022390
22400       END { zm4_C } ;                                                     00022400
22410                                                                           00022410
22420     PROCEDURE zm4_D(n,k,newd: integer) ;                                  00022420
22430       { zero or more P4, followed by D }                                  00022430
22440       VAR j: integer ;                                                    00022440
22450       BEGIN                                                               00022450
22460       WHILE IsMarked(lower,n,k) DO BEGIN                                  00022460
22470         j := 1 ;                                                          00022470
22480         REPEAT                                                            00022480
22490           Update(lower,2*n,k+j,MIN(2*newd,Bt[n,j].dist),D_Ref) ; { D*j }  00022490
22500           j := j + 1 ;                                                    00022500
22510           UNTIL (Bt[n,j].dist <= newd) ;                                  00022510
22520         REPEAT                                                            00022520
22530           Update(lower,2*n,k+j,MIN(newd,2*Bt[n,j].dist),D_Ref) ; { Dj }   00022530
22540           j := j + 1 ;                                                    00022540
22550           UNTIL (j = n) ;                                                 00022550
22560         n := n - 1 ; k := k - 1 ;                                         00022560
22570         END { while } ;                                                   00022570
22580       END { zm4_D } ;                                                     00022580
22590                                                                           00022590
22600     PROCEDURE zm3_zm4_D(n,k,newd: integer) ;                              00022600
22610       { zero or more P3 and P4, followed by D }                           00022610
22620       BEGIN                                                               00022620
22630       IF (2*n <= MaxBtIndex+5) THEN                                       00022630
22640         WHILE IsMarked(lower,n,k) DO BEGIN                                00022640
22650           zm4_D(n,k,newd) ;                                               00022650
22660           n := n - 1 ; newd := newd - 1 ;                                 00022660
22670           END { while } ;                                                 00022670
22680       END { zm3_zm4_D } ;                                                 00022680
22690                                                                           00022690
22700     PROCEDURE om2_zm4_D(n,k,newd: integer) ;                              00022700
22710       { once or more P2, zero or more P4, followed by D }                 00022710
22720       BEGIN                                                               00022720
22730       n := n + 1 ; IF odd(newd) THEN newd := newd + 1 ;                   00022730
22740       WHILE IsMarked(lower,n,k) AND (2*n <= MaxBtIndex+5) DO BEGIN        00022740
22750         zm4_D(n,k,newd) ;                                                 00022750
22760         n := n + 1 ;                                                    00022760
22770         END { while } ;                                                 00022770
22780       END { zm3_zm4_D } ;                                                 00022780
22790                                                                           00022790
22800     PROCEDURE zm4_lower(n,k,newd: integer; r: Reference) ;                00022800
22810       { zero or more P4 (lower bound) }                                   00022810
22820       BEGIN                                                               00022820
22830       IF NOT IsInTable(n,k) THEN BEGIN                                    00022830
22840         k := k-n+MaxBtIndex ; n := MaxBtIndex END ;                       00022840
22850       WHILE IsImprovement(lower,n,k,newd) DO BEGIN                        00022850
22860         MakeImprovement(lower,n,k,newd,r) ;                             00022860
22870         Bt[n,k].mark := depth ;                                         00022870
22880         r := NoRef ;                                                    00022880
22890         n := n - 1 ; k := k - 1 ;                                       00022890
22900         END { while } ;                                                 00022900
22910       IF (n < MaxBtIndex) THEN BEGIN                                    00022910
```

```
22920         CondWipeRef(lower,n,k,newd) ; WipeMark(lower,n,k) ;                    00022920
22930         END { if } ;                                                           00022930
22940       END { zm4_lower } ;                                                      00022940
22950                                                                              00022950
22960     PROCEDURE zm3_zm4_lower(n,k,newd: integer; r: Reference) ;               00022960
22970       { zero or more P3 and P4 (lower bound) }                               00022970
22980       BEGIN                                                                  00022980
22990       IF (n-k+1 > MaxBtIndex) THEN BEGIN                                     00022990
23000         newd := newd-(n-k+1-MaxBtIndex) ; n := k-1+MaxBtIndex END ;          00023000
23010       WHILE ((n > MaxBtIndex) OR IsImprovement(lower,n,k,newd)) AND          00023010
23020             (k < n) DO BEGIN                                                 00023020
23030         zm4_lower(n,k,newd,r) ;                                            00023030
23040         IF IsInTable(n,k) THEN r := NoRef ;                                00023040
23050         n := n - 1 ; newd := newd - 1 ;                                    00023050
23060         END { while } ;                                                    00023060
23070       IF (n < MaxBtIndex) THEN BEGIN                                       00023070
23080         CondWipeRef(lower,n,k,newd) ; CondWipeRef(lower,n-1,k,newd-1) ;    00023080
23090         WipeMark(lower,n,k) ;                                              00023090
23100         END { if } ;                                                       00023100
23110       END { zm3_zm4_lower } ;                                              00023110
23120                                                                            00023120
23130     PROCEDURE om2_zm4_lower(n,k,newd: integer) ;                           00023130
23140       { once or more P2, zero or more P4 (lower bound) }                   00023140
23150       BEGIN                                                                00023150
23160       n := n + 1 ; IF odd(newd) THEN newd := newd + 1 ;                    00023160
23170       WHILE ((n > MaxBtIndex) OR IsImprovement(lower,n,k,newd)) AND        00023170
23180             (n-k+1 <= MaxBtIndex) DO BEGIN                                 00023180
23190         zm4_lower(n,k,newd,NoRef) ;                                        00023190
23200         n := n + 1 ;                                                       00023200
23210         END { while } ;                                                    00023210
23220       CondWipeRef(lower,n,k,newd) ; WipeMark(lower,n,k) ;                  00023220
23230       END { om2_zm4_lower } ;                                              00023230
23240                                                                            00023240
23250     PROCEDURE zm4_upper(n,k,newd: integer; r: Reference) ;                 00023250
23260       { zero or more P4 (upper bound) }                                    00023260
23270       BEGIN { on entry IsImprovement(upper,n,k,newd) holds }               00023270
23280       REPEAT                                                               00023280
23290         MakeImprovement(upper,n,k,newd,r) ;                                00023290
23300         Bt[k,n].mark := depth ;                                            00023300
23310         r := NoRef ;                                                       00023310
23320         n := n + 1 ; k := k + 1 ;                                          00023320
23330         UNTIL NOT IsImprovement(upper,n,k,newd) ;                          00023330
23340       CondWipeRef(upper,n,k,newd) ; WipeMark(upper,n,k) ;                  00023340
23350       END { zm4_upper } ;                                                  00023350
23360                                                                            00023360
23370     PROCEDURE om2_zm4_upper(n,k,newd: integer; r: Reference) ;             00023370
23380       { once or more P2, zero or more P4 (upper bound) }                   00023380
23390       BEGIN                                                                00023390
23400       IF (n <= MaxBtIndex) THEN r := NoRef ;                               00023400
23410       n := n - 1 ; IF odd(newd) THEN newd := newd - 1 ;                    00023410
23420       IF (n > MaxBtIndex) THEN n := MaxBtIndex ;                           00023420
23430       WHILE IsImprovement(upper,n,k,newd) DO BEGIN                         00023430
23440         zm4_upper(n,k,newd,r) ;                                            00023440
23450         r := NoRef ;                                                       00023450
23460         n := n - 1 ;                                                       00023460
23470         END { while } ;                                                    00023470
23480       CondWipeRef(upper,n,k,newd) ; WipeMark(upper,n,k) ;                  00023480
23490       END { om2_zm4_upper } ;                                              00023490
23500                                                                            00023500
```

```
23510      PROCEDURE zm3_zm4_upper(n,k,newd: integer; r: Reference) ;                00023510
23520        { zero or more P3 and P4 (upper bound) }                                00023520
23530        BEGIN                                                                   00023530
23540        WHILE IsImprovement(upper,n,k,newd) DO BEGIN                            00023540
23550          zm4_upper(n,k,newd,r) ;                                               00023550
23560          r := NoRef ;                                                          00023560
23570          n := n + 1 ; newd := newd + 1 ;                                       00023570
23580          END { while } ;                                                       00023580
23590        CondWipeRef(upper,n,k,newd) ; CondWipeRef(upper,n+1,k,newd+1) ;         00023590
23600        WipeMark(upper,n,k) ;                                                   00023600
23610        END { zm3_zm4_upper } ;                                                 00023610
23620                                                                                00023620
23630      PROCEDURE Update(b: BoundKind; n,k,newd: integer; r: Reference) ;         00023630
23640        { The recursive updating procedure }                                    00023640
23650        BEGIN                                                                   00023650
23660    $ SET OMIT = NOT TESTOUTPUT                                                 00023660
23670        indent ;                                                                00023670
23680    write(Report,'Checking bound (',Id[b],',',n:1,',',k:1,',',newd:1,')',r);    00023680
23690    $ POP OMIT                                                                  00023690
23700        IF ((0 < k) AND (k < n) AND (n > MaxBtIndex) AND (newd > 1))            00023700
23710          OR IsImprovement(b,n,k,newd) THEN BEGIN                               00023710
23720    $ SET OMIT = NOT TESTOUTPUT                                                 00023720
23730        writeln(Report,': interesting') ;                                       00023730
23740    $ POP OMIT                                                                  00023740
23750        depth := depth + 1 ;                                                    00023750
23760        CASE b OF                                                               00023760
23770          lower: BEGIN                                                          00023770
23780            zm3_zm4_lower(n,k,newd,r) ; { also does marking }                   00023780
23790            om2_zm4_lower(n,k,newd) ;   { also does marking }                   00023790
23800            Update(lower,n-newd,k-1,(newd+1) DIV 2,A_Ref) ; { A }               00023800
23810            B1(n,k,newd) ;                                                      00023810
23820            IF IsInTable(n,k) THEN BEGIN                                        00023820
23830              zm4_C(n,k,newd) ;                                                 00023830
23840              zm3_zm4_D(n,k,newd) ;                                             00023840
23850              om2_zm4_D(n,k,newd) ;                                             00023850
23860              END { if } ;                                                      00023860
23870            END { lower bound } ;                                               00023870
23880          upper: BEGIN                                                          00023880
23890            zm3_zm4_upper(n,k,newd,r) ; { also does marking }                   00023890
23900            om2_zm4_upper(n,k,newd,r) ; { also does marking }                   00023900
23910            IF IsInTable(n,k) THEN BEGIN                                        00023910
23920              zm3_E(n,k,newd) ;                                                 00023920
23930              zm3_B3(n,k,newd) ;                                                00023930
23940              zm2_zm4_B2B4(n,k,newd) ;                                          00023940
23950              END { if } ;                                                      00023950
23960            END { upper bound } ;                                               00023960
23970          END { case b } ;                                                      00023970
23980        depth := depth - 1 ;                                                    00023980
23990    $ SET OMIT = NOT TESTOUTPUT                                                 00023990
24000        indent ;                                                                00024000
24010    writeln(Report,'Exit bound (',Id[b],',',n:1,',',k:1,',',newd:1,')',r) ;     00024010
24020    $ POP OMIT                                                                  00024020
24030        END { then }                                                            00024030
24040    $ SET OMIT = NOT TESTOUTPUT                                                 00024040
24050        ELSE writeln(Report,': useless') ;                                      00024050
24060    $ POP OMIT                                                                  00024060
24070        END { Update } ;                                                        00024070
24080                                                                                00024080
24090      BEGIN { UpdateBt }                                                        00024090
```

```
      write(Report,                                                          00024100
        'Considering bound (',Id[b],',',len:3,',',dim:3,',',newdist:2,') "', 00024110
        r,'"') ;                                                             00024120
      writeln(List,'(',Id[b],',',len:3,',',dim:3,',',newdist:2,') ',r) ;     00024130
      Lmodified := false ;                                                   00024140
      NoRemarksYet := true ; { first remark still has to be made }           00024150
      pt := RUNTIME ;                                                        00024160
      depth := 0 ;                                                           00024170
      Update(b,len,dim,newdist,r) ;                                          00024180
    0: { in case of violation exit }                                         00024190
      pt := RUNTIME - pt ; TotUpdates := TotUpdates + 1 ;                    00024200
      IF Lmodified THEN BEGIN                                                00024210
          TotUseful := TotUseful + 1 ;                                       00024220
          BtModified := true ;                                               00024230
          IF NoRemarksYet THEN write(Report,': useful') ;                    00024240
          END                                                                00024250
        ELSE IF NoRemarksYet THEN write(Report,': USELESS') ;                00024260
      writeln(Report,'  (PT = ',pt:1:2,')') ;                               00024270
      END { UpdateBt } ;                                                     00024280
                                                                             00024290
```

```
30000    PROCEDURE StartPage ;                                                  00030000
30010      BEGIN                                                                00030010
30020      PagesPrinted := PagesPrinted + 1 ;                                   00030020
30030      IF (PagesPrinted MOD MaxPage = 0) THEN BEGIN                         00030030
30040        CLOSE(Pr) ; rewrite(Pr) END ;                                      00030040
30050      END { StartPage } ;                                                  00030050
30060                                                                           00030060
30070    PROCEDURE PrintBlock(i,j: integer) ; { i,j assumed in range }          00030070
30080      CONST SqSize = 5 ;                                                   00030080
30090      VAR line,col,height,width: integer ;                                 00030090
30100          startline,endline,startcol,endcol: integer ;                     00030100
30110                                                                           00030110
30120      PROCEDURE VertSep ;                                                  00030120
30130        BEGIN                                                              00030130
30140        IF (col MOD SqSize = 0) THEN write(Pr,'|')                         00030140
30150          ELSE write(Pr,' ') ;                                             00030150
30160        END { VertSep } ;                                                  00030160
30170                                                                           00030170
30180      PROCEDURE HorSep ;                                                   00030180
30190        BEGIN                                                              00030190
30200        IF ((line MOD SqSize = 0) AND (line <> 125)) OR                    00030200
30210           (line = MaxBtIndex) THEN BEGIN                                  00030210
30220          write(Pr,'-----') ;                                              00030220
30230          col := startcol ;                                                00030230
30240          WHILE (col <> endcol) DO BEGIN                                   00030240
30250            VertSep ;                                                      00030250
30260            write(Pr,'------') ;                                           00030260
30270            col := col + 1 ;                                               00030270
30280            END { while } ;                                                00030280
30290          VertSep ; writeln(Pr,'-----') ;                                  00030290
30300          END { if } ;                                                     00030300
30310        END { HorSep } ;                                                   00030310
30320                                                                           00030320
30330      PROCEDURE kValues ;                                                  00030330
30340        BEGIN                                                              00030340
30350        write(Pr,' n,k') ;                                                 00030350
30360        col := startcol ;                                                  00030360
30370        WHILE (col <> endcol) DO BEGIN                                     00030370
30380          VertSep ;                                                        00030380
30390          col := col + 1 ;                                                 00030390
30400          IF (col < 100) THEN write(Pr,' ',col:2,'  ')                     00030400
30410            ELSE write(Pr,' ',col:3,' ') ;                                 00030410
30420          END { while } ;                                                  00030420
30430        vertSep ; write(Pr,' ',i:1,',',j:1) ; { block's coordinates }      00030430
30440        END { kValues } ;                                                  00030440
30450                                                                           00030450
30460      PROCEDURE PrintRef(r1, r2: char; rightjustified: boolean) ;          00030460
30470        BEGIN                                                              00030470
30480        IF rightjustified THEN                                             00030480
30490            IF (r2 = ' ') THEN write(Pr,r2,r1)                             00030490
30500              ELSE write(Pr,r1,r2)                                         00030500
30510          ELSE                                                             00030510
30520            IF (r1 = ' ') THEN write(Pr,r2,r1)                             00030520
30530              ELSE write(Pr,r1,r2)                                         00030530
30540        END { PrintRef } ;                                                 00030540
30550                                                                           00030550
```

```
30560     PROCEDURE BtValues ;                                                                  00030560
30570       VAR x: integer ;                                                                    00030570
30580       BEGIN                                                                               00030580
30590       write(Pr,' ':4) ;                                                                   00030590
30600       col := startcol ;                                                                   00030600
30610       WHILE (col <> endcol) DO BEGIN                                                      00030610
30620         VertSep ; col := col + 1 ;                                                        00030620
30630         IF (col > line) OR (line > MaxBtIndex) THEN write(Pr,' ':5)                        00030630
30640           ELSE BEGIN                                                                      00030640
30650             WITH Bt[line,col] DO BEGIN                                                    00030650
30660               PrintRef(refch1,refch2,                                                     00030660
30670                        NOT IsOpenCase(line,col) AND (dist < 10)) ;                        00030670
30680               x := ord(unchanged) END ;                                                   00030680
30690             WITH Bt[col,line] DO BEGIN                                                    00030690
30700               x := 2*x + ord(unchanged) ;                                               00030700
30710               CASE x OF                                                                 00030710
30720                 0: write(Pr,'*') ;                                                      00030720
30730                 1: write(Pr,'<') ;                                                      00030730
30740                 2: write(Pr,'>') ;                                                      00030740
30750                 3: write(Pr,' ') ;                                                      00030750
30760                 END { case } ;                                                          00030760
30770               PrintRef(refch1,refch2,IsOpenCase(line,col)) ;                            00030770
30780               END { with } ;                                                            00030780
30790             END { else } ;                                                              00030790
30800         END { while } ;                                                                 00030800
30810       VertSep ; writeln(Pr) ;                                                           00030810
30820       write(Pr,line:3,' ') ; col := startcol ;                                          00030820
30830       WHILE (col <> endcol) DO BEGIN                                                    00030830
30840         VertSep ; col := col + 1 ;                                                      00030840
30850         IF (col > line) OR (line > MaxBtIndex) THEN write(Pr,' ':5)                      00030850
30860           ELSE IF IsOpenCase(line,col) THEN BEGIN                                       00030860
30870             write(Pr,Bt[line,col].dist:2,'-') ;                                         00030870
30880             WITH Bt[col,line] DO                                                        00030880
30890               IF (dist < 10) THEN write(Pr,dist:1,' ')                                  00030890
30900                 ELSE write(Pr,dist:2) ;                                                 00030900
30910             END { open case }                                                           00030910
30920           ELSE                                                                        00030920
30930             write(Pr,Bt[line,col].dist:3,'  ') ;                                      00030930
30940         END { while } ;                                                               00030940
30950       VertSep ; writeln(Pr,' ',line:3) ;                                              00030950
30960       END { BtValues } ;                                                              00030960
30970                                                                                       00030970
30980     BEGIN { PrintBlock }                                                              00030980
30990     StartPage ;                                                                       00030990
31000     ToBePrinted[i,j] := false ;                                                       00031000
31010     IF (i = 0) THEN BEGIN                                                             00031010
31020       IF (j = 0) THEN { print table header }                                          00031020
31030           writeln(Pr,' ':35,                                                          00031030
31040           'Upper and Lower Bounds on d_max(n,k) for Binary Linear Codes')             00031040
31050         ELSE writeln(Pr) ;                                                          00031050
31060       writeln(Pr) ;                                                                 00031060
31070       END ;                                                                         00031070
31080     IF (i < 4) THEN height := 25 ELSE height := 27 ;                                00031080
31090     startline := 25*i ; endline := startline + height ;                             00031090
31100     width := 20 ; startcol := 20*j ; endcol := startcol + width ;                   00031100
31110     line := startline ;                                                             00031110
31120     kValues ; writeln(Pr) ;                                                         00031120
31130     WHILE (line <> endline) DO BEGIN                                                00031130
31140       HorSep ; line := line + 1 ;                                                   00031140
```

```
      BtValues ;
      END { while } ;
    HorSep ; kValues ;
  $$ SET OMIT = NOT LSIPRINTER
    IF (height <> 27) THEN
      { an LSI printer automatically pages for height = 27 }
  $$ POP OMIT
      page(Pr) ;
    END { PrintBlock } ;

  PROCEDURE PrintAll ;
    VAR i,j: integer ;
    BEGIN
    FOR j := 0 TO 6 DO
      FOR i := 0 TO 4 DO
        IF IsBlock(i,j) THEN PrintBlock(i,j) ;
    writeln(Report, 'All blocks printed.') ;
    END { PrintAll } ;

  PROCEDURE PrintModified ;
    VAR i,j,count: integer ;
    BEGIN
    count := 0 ; { # blocks printed }
    FOR j := 0 TO 6 DO
      FOR i := 0 TO 4 DO
        IF ToBePrinted[i,j] THEN BEGIN
          PrintBlock(i,j) ; count := count + 1 ;
        END { if } ;
    writeln(Report, 'All (',count:1,') blocks with ',
     'unprinted modifications have been printed.') ;
    END { PrintModified } ;

  PROCEDURE WallPaper ;
    VAR i,j: integer ;
    BEGIN
    FOR j := 0 TO 4 DO
      FOR i := 0 TO 4 DO
        IF IsBlock(i,j) THEN PrintBlock(i,j)
          ELSE IF (i = 0) AND (j >= 3) THEN PrintBlock(4,j+2)
          ELSE BEGIN
            StartPage ;
            writeln(Pr,'FILLER PAGE FOR WALLPAPER') ; page(Pr) ;
            END ;
    writeln(Report, 'Wallpaper printed.') ;
    END { WallPaper } ;
```

```
10000   $$ RESET LIST                                                    00010000
10010   $$ RESET LISTINCL                                                00010010
10020   $$ SET   WARNSUPR                                               00010020
10030   $$ SET   NOBOUNDS                                               00010030
10040                                                                    00010040
10050   PROGRAM InitialBt (                                              00010050
10060     output,                                                        00010060
10070     ibf     : FILE <areasize=1,areas=1,title='INITIAL/BOUNDS'>,    00010070
10080     iuf     : FILE <blockstructure=fixed,areasize=20,              00010080
10090                     title='INITIAL/UPDATES'>                       00010090
10100     ) ;                                                            00010100
10110                                                                    00010110
10120   { Generate initial table of bounds, stored on file INITIAL/BOUNDS.  00010120
10130     This table P2, P3, P4, C, and D invariant, any variances w.r.t.  00010130
10140     other propagation rules are written to the file INITIAL/UPDATES. }  00010140
10150                                                                    00010150
10160   { Include CONST and TYPE section from BoundUpdater }             00010160
10170   $$ INCLUDE "BOUNDUPDATER" 12000 TO 12999                         00010170
10180                                                                    00010180
10190   VAR                                                              00010190
10200     Bt                  : BtArray ;                                00010200
10210     ibf                 : BtFile ; { initial bound file }          00010210
10220     iuf                 : TEXT ;   { initial update file }         00010220
10230                                                                    00010230
10240   FUNCTION IsOpenCase(n,k: BtIndex): boolean ;                     00010240
10250     BEGIN { 1 < k <= n  assumed }                                  00010250
10260     IsOpenCase := (Bt[n,k].dist < Bt[k,n].dist) ;                  00010260
10270     END { IsOpenCase } ;                                           00010270
10280                                                                    00010280
10290   PROCEDURE GenInitialBt ;                                         00010290
10300     { Generate initial table of bounds Bt }                        00010300
10310     VAR n,n1,n2, k,k1,k2, c,dualD: integer ;                       00010310
10320                                                                    00010320
10330   PROCEDURE GenUpdate(b: BoundKind; n,k,newd: integer; r: Reference) ;  00010330
10340       BEGIN                                                        00010340
10350       CASE b OF                                                    00010350
10360         lower: write(iuf,'L') ;                                    00010360
10370         upper: write(iuf,'U') ;                                    00010370
10380         END { case } ;                                             00010380
10390       writeln(iuf,n:4,k:4,newd:4,r:4) ;                          00010390
10400       END { GenUpdate } ;                                          00010400
10410                                                                    00010410
10420   PROCEDURE Improve(b: BoundKind; n,k,newd: BtIndex; r: Reference) ;  00010420
10430       BEGIN                                                        00010430
10440       CASE b OF                                                    00010440
10450         lower: BEGIN                                               00010450
10460           WITH Bt[n,k] DO BEGIN dist := newd ; refch1 := r[1] END ;  00010460
10470           { conditionally wipe some earlier established references }  00010470
10480           WITH Bt[n-1,k-1] DO IF (dist = newd) THEN refch1 := ' ' ; { P4 }  00010480
10490           WITH Bt[n-1,k] DO IF (dist = newd-1) THEN refch1 := ' ' ; { P3 }  00010490
10500           IF (k <= n-2) THEN { twice P3, special because of D below }  00010500
10510             WITH Bt[n-2,k] DO IF (dist = newd-2) THEN refch1 := ' ' ;  00010510
10520           END { lower bound } ;                                    00010520
10530         upper: BEGIN                                               00010530
10540           WITH Bt[k,n] DO BEGIN dist := newd ; refch1 := r[1] END ;  00010540
10550           { conditionally wipe some earlier established references }  00010550
```

```
10560           WITH Bt[k,n-1] DO BEGIN { P2 }                                  00010560
10570             IF odd(newd) THEN newd := newd - 1 ;                          00010570
10580             IF (dist = newd) THEN refch1 := ' ' ;                         00010580
10590             END { with } ;                                                00010590
10600           END { upper bound } ;                                           00010600
10610         END { case } ;                                                    00010610
10620       END { Improve } ;                                                   00010620
10630                                                                           00010630
10640     BEGIN { GenInitialBt }                                                00010640
10650     { Lower bounds }                                                      00010650
10660     FOR n := 1 TO MaxBtIndex DO                                           00010660
10670       FOR k := 1 TO n DO WITH Bt[n,k] DO BEGIN                            00010670
10680         refch1 := ' ' ; refch2 := ' ' ;                                   00010680
10690         IF (k = 1) THEN dist := n                                         00010690
10700           ELSE IF (k = n) THEN dist := 1                                  00010700
10710           ELSE BEGIN { 1 < k < n }                                        00010710
10720           { P2 (parity check) }                                           00010720
10730           dist := Bt[n-1,k].dist ;                                        00010730
10740           IF odd(dist) THEN dist := dist + 1 ;                            00010740
10750           { conditionally wipe Reference on account of P4, but not P3 }   00010750
10760           IF (Bt[n-1,k-1].dist = dist) THEN Bt[n-1,k-1].refch1 := ' ' ;   00010760
10770                                                                           00010770
10780           { C (concatenation) }                                           00010780
10790           n1 := k ; n2 := n - k ; { n1+n2 = n }                           00010790
10800           WHILE (n1 <= n2) DO BEGIN                                       00010800
10810             c := Bt[n1,k].dist + Bt[n2,k].dist ;                          00010810
10820             IF (c > dist) THEN BEGIN                                      00010820
10830               Improve(lower,n,k,c,C_Ref) ;                                00010830
10840               END { if } ;                                                00010840
10850             n1 := n1 + 1 ; n2 := n2 - 1 ;                                 00010850
10860             END { while } ;                                               00010860
10870                                                                           00010870
10880           { D: construction u,u+v }                                       00010880
10890           IF NOT odd(n) THEN BEGIN                                        00010890
10900             n2 := n DIV 2 ; k1 := 1 ; k2 := k - 1 ; { k1+k2 = k }         00010900
10910             WHILE (k1 <= k2) DO BEGIN                                     00010910
10920               c := MIN(Bt[n2,k1].dist, 2*Bt[n2,k2].dist) ;                00010920
10930               IF (c > dist) THEN BEGIN                                    00010930
10940                 Improve(lower,n,k,c,D_Ref) ;                              00010940
10950                 { check [n-1,k-1] (P4) & [n-1,k] (P3) }                   00010950
10960                 IF (dist > Bt[n-1,k-1].dist) THEN                         00010960
10970                   Improve(lower,n-1,k-1,dist,NoRef) ;                     00010970
10980                 IF (dist-1 > Bt[n-1,k].dist) THEN                         00010980
10990                   Improve(lower,n-1,k,dist-1,NoRef) ;                     00010990
11000                 END { if } ;                                              00011000
11010               k1 := k1 + 1 ; k2 := k2 - 1 ;                               00011010
11020               END { while } ;                                             00011020
11030             END { if } ;                                                  00011030
11040                                                                           00011040
11050           { A: residual code }                                            00011050
11060           c := (dist+1) DIV 2 ; { = ceiling(dist/2) }                     00011060
11070           IF (c > Bt[n-dist,k-1].dist) THEN                               00011070
11080             GenUpdate(lower,n-dist,k-1,c,A_Ref) ;                         00011080
11090           END { 1 < k < n } ;                                             00011090
11100       END { for n, for k, with Bt[n,k] } ;                                00011100
11110                                                                           00011110
11120     { exceptional case, MaxBtIndex is odd: 0 followed by P3, P4 or A }    00011120
11130     IF odd(MaxBtIndex) THEN BEGIN                                         00011130
11140       n := MaxBtIndex + 1 ; n2 := n DIV 2 ;                               00011140
```

```
    FOR k := 2 TO n-2 DO BEGIN                                                  00011150
      k2 := MIN(k-1,n2-1) ; k1 := k-k2 ; { k1+k2 = k }                          00011160
      WHILE (k1 <= k2) DO BEGIN                                                 00011170
        c := MIN(Bt[n2,k1].dist, 2*Bt[n2,k2].dist) ; { (lower,n,k,c) }          00011180
        IF (c > Bt[n-1,k-1].dist) THEN { P4 }                                   00011190
          IF odd(c) THEN Improve(lower,n-1,k-1,c,D_Ref)                         00011200
            ELSE Improve(lower,n-1,k-1,c,NoRef) ;                               00011210
        IF (c-1 > Bt[n-1,k].dist) THEN { P3 }                                   00011220
          Improve(lower,n-1,k,c-1,D_Ref) ;                                      00011230
        IF ((c+1) DIV 2 > Bt[n-c,k-1].dist) THEN { A }                          00011240
          GenUpdate(lower,n-c,k-1,(c+1) DIV 2,A_Ref) ;                          00011250
        k1 := k1 + 1 ; k2 := k2 - 1 ;                                           00011260
        END { while } ;                                                         00011270
      END { for k } ;                                                           00011280
    END { if } ;                                                                00011290
                                                                                00011300
  { Upper bounds, N.B. subscripts of Bt interchanged for upper bound! }         00011310
  FOR n := 1 TO MaxBtIndex DO                                                   00011320
    FOR k := 1 TO n DO WITH Bt[k,n] DO BEGIN                                    00011330
      refch1 := ' ' ; refch2 := ' ' ;                                           00011340
      IF (k = 1) THEN dist := n                                                 00011350
        ELSE IF (k = n) THEN dist := 1                                          00011360
        ELSE BEGIN { 1 < k < n }                                                00011370
        { P3 and P4 }                                                           00011380
        dist := MIN(Bt[k-1,n-1].dist, Bt[k,n-1].dist+1) ;                       00011390
                                                                                00011400
        { E: one step Griesmer bound }                                          00011410
        WHILE ((dist+1) DIV 2 > Bt[k-1,n-dist].dist) DO BEGIN                   00011420
          Improve(upper,n,k,dist-1,E_Ref) ;                                     00011430
          END { while } ;                                                       00011440
                                                                                00011450
        { B: upper bound via "dual code" construction Y1,P4* }                  00011460
        IF (n-k <= k) AND (k < n-1) THEN BEGIN                                  00011470
          { N.B. possibly n-k = k }                                             00011480
          duald := Bt[n-k,n].dist ; { upper bound for dual of (n,k) }           00011490
          { duald <= k+1 < n, in fact duald < k+1 }                             00011500
          WHILE (dist > Bt[k-duald+1, n-duald].dist) DO BEGIN                   00011510
            Improve(upper,n,k,Bt[k-duald+1,n-duald].dist,B_Ref) ;               00011520
            duald := Bt[n-k,n].dist ; { when n-k = k: duald = dist }            00011530
            END { while } ;                                                     00011540
          c := Bt[n-k-dist+1, n-dist].dist ;                                    00011550
          IF (duald > c) THEN                                                   00011560
            GenUpdate(upper,n,n-k,c,B_Ref) ;                                    00011570
          END { if } ;                                                          00011580
                                                                                00011590
        { B: lower bound via "dual code" construction Y1,P4* }                  00011600
        IF (k < n-1) THEN BEGIN                                                 00011610
          { dist <= n-k+1 < n, in fact dist < n-k+1 }                           00011620
          c := Bt[n,n-k].dist ;                                                 00011630
          IF (c > Bt[n-dist, n-k-dist+1].dist) THEN                             00011640
            GenUpdate(lower,n-dist,n-k-dist+1,c,B_Ref) ;                        00011650
          END { if } ;                                                          00011660
        END { 1 < k < n } ;                                                     00011670
      END { for n, for k, with Bt[k,n] } ;                                      00011680
                                                                                00011690
  { initialize statistics }                                                     00011700
  Bt[0,0].unsolved := 0 ;                                                       00011710
  FOR k := 1 TO MaxBtIndex DO Bt[0,k].unsolved := 0 ;                           00011720
  FOR n := 1 TO MaxBtIndex DO Bt[n,0].unsolved := 0 ;                           00011730
```

```
11740      FOR n := 1 TO MaxBtIndex DO                                                    00011740
11750        FOR k := 1 TO n DO                                                           00011750
11760          IF IsOpenCase(n,k) THEN BEGIN                                            00011760
11770            WITH Bt[0,0] DO unsolved := unsolved + 1 ;                             00011770
11780            WITH Bt[0,k] DO unsolved := unsolved + 1 ;                             00011780
11790            WITH Bt[n,0] DO unsolved := unsolved + 1 ;                             00011790
11800            END { if } ;                                                           00011800
11810      END { GenInitialBt } ;                                                       00011810
11820                                                                                   00011820
11830    BEGIN { InitialBt }                                                            00011830
11840    rewrite(output) ; rewrite(iuf) ;                                               00011840
11850    writeln('Initial Bound Table Generator, MaxBtIndex = ',MaxBtIndex:1) ;         00011850
11860    GenInitialBt ;                                                                 00011860
11870    write(iuf,'0') ;                                                               00011870
11880    CLOSE(iuf,CRUNCH) ;                                                            00011880
11890    { write Bt to file }                                                           00011890
11900    rewrite(ibf) ;                                                                 00011900
11910    ibf@ := Bt ;                                                                   00011910
11920    put(ibf) ;                                                                     00011920
11930    CLOSE(ibf) ;                                                                   00011930
11940    END { InitialBt }.                                                             00011940
11950                                                                                   00011950
```