Theoretical Computer Science 95 (1992) 231–244
Elsevier

On the space complexity of some algorithms for sequence comparison

Yuval Rabani

Department of Computer Science, Tel-Aviv University, Tel-Aviv, Israel

Zvi Galil*

Columbia University, Computer Science Department, New York, NY10027, USA, and Department of Computer Science, Tel-Aviv University, Tel-Aviv, Israel

Communicated by M. Nivat
Received March 1989
Revised February 1990

Abstract

Rabani, Y. and Z. Galil, On the space complexity of some algorithms for sequence comparison, Theoretical Computer Science 95 (1992) 231–244.

Recent algorithms for computing the *modified edit distance* given convex or concave gap cost functions are shown to require $\Omega(n^2)$ space for certain input.

0. Introduction

Galil and Giancarlo [1] described two algorithms for speeding up the computation of

$$E[j] = \min_{0 \leqslant k \leqslant j-1} \{D[k] + w(k, j)\}, \quad j = 1, \ldots, n, \tag{1}$$

where w is a given weight function, $D[0]$ is given and for every $k = 1, \ldots, n$, $D[k]$ is easily computable from $E[k]$. These algorithms handle two special cases: the convex case, where w satisfies the *inverse quadrangle inequality*

$$w(i, k) + w(j, l) \geqslant w(j, k) + w(i, l) \quad \text{for all } i \leqslant j \leqslant k \leqslant l, \tag{2}$$

and the concave case, where w satisfies the *quadrangle inequality*. The algorithms are denoted as Algorithm A and Algorithm B, respectively. Miller and Myers [3] independently described an algorithm, which is essentially the same as Algorithm A. Note that the standard definitions of *convex* and *concave* are interchanged here, following Galil

*The work of the second author was supported in part by NSF Grants DCR-85-11713 and CCR-86-05353.

and Giancarlo [1]. In this paper, functions such as $\log(x)$ or $\sqrt{x}$ are termed *convex* functions and, therefore, the derived weight functions such as $w(i,j)=\log(j-i)$ specify a convex case. Similarly, functions such as x^2 are termed *concave* functions, and the derived weight functions such as $w(i,j)=(j-i)^2$ specify a concave case.

These algorithms were shown useful for computing the *modified edit distance*. Given two strings $x=x_1\ldots x_m$ and $y=y_1\ldots y_n$ over alphabet Σ, the *modified edit distance* is defined as the minimal cost of an edit sequence that changes x into y. An edit sequence consists of operations of the form delete$(x_{i+1}\ldots x_j)$ of cost $w'(i,j)$, insert$(y_{k+1}\ldots y_l)$ of cost $w(k,l)$ and substitute$(x_{i'}, y_{j'})$ of cost $s(x_{i'}, y_{j'})$.

In order to compute the *modified edit distance*, a dynamic programming equation of the form

$$\begin{aligned} D[i,j] &= \min\{D[i-1,j-1]+s(x_i,y_j), E[i,j], F[i,j]\}, \\ E[i,j] &= \min_{0\leqslant k\leqslant j-1}\{D[i,k]+w(k,j)\}, \\ F[i,j] &= \min_{0\leqslant l\leqslant i-1}\{D[l,j]+w'(l,i)\}, \end{aligned} \tag{3}$$

with initial conditions $D[0,0]=0$, $D[i,0]=w'(0,i)$, $1\leqslant i\leqslant m$ and $D[0,j]=w(0,j)$, $1\leqslant j\leqslant n$ is considered.

The computation of a row of E and a column of F are each equivalent to the computation of (1). The speedup achieved for that computation yields a speedup in the computation of (3) from the trivial $O(n^3)$ to $O(n^2\log n)$ or even $O(n^2)$ for simple gap cost functions. In practice, it is the space complexity and not the time complexity which limits the maximum size of problems that can be solved on small machines [3]. The minimum cost edit sequence with affine gap costs has been shown by Hirschberg [2] to be computable in $O(n)$ space. Miller and Myers [3] demonstrated the implementation of Hirschberg's method for the convex case. Unfortunately, this does not mean that space is reduced to $O(n)$. Hirschberg's method enables us to reduce the space required for the maintenance of the matrix D. However, the stack or queue required to compute each row of E (each column of F) consumes on its own $\Omega(n)$ ($\Omega(m)$) space and m rows of E (n columns of F) need be computed simultaneously. Therefore, a bound of $\Omega(mn)$ space can be expected of the application for computing (3). Miller and Myers state that experimental evidence regarding their algorithm implies that the amount of space required is usually linear. We shall hereby demonstrate that the bound of $\Omega(mn)$ is actually achieved for certain inputs in both the convex and the concave cases. For the sake of simplicity we consider $m=n$.

1. The convex case

Typical computations of the *modified edit distance* with convex weight functions use gap cost functions of the form

$$w(i,j)=f^1(x_i,x_{i+1})+f^2(x_j,x_{j+1})+g(j-i), \tag{4}$$

where g is a convex function. The functions f^1 and f^2 represent the cost of breaking the links at the edges of the gap in the string x.

Three examples are given for input for the Galil–Giancarlo algorithm that require $\Omega(n^2)$ space for the computation of (3). Example 1.1 assumes a substitution cost function s which can have $O(n)$ different values. Let l be the size of the alphabet Σ over which the input strings x and y are defined. Then s may have at most l^2 different values. This implies that l is unbounded, because for input of length n, an alphabet of size $l=O(\sqrt{n})$ is required. Example 1.2 assumes a function s with a constant number of different values, but still uses an unbounded alphabet, because the expression for w' contains a function f^1 with $O(n)$ different values. Because f^1 may have at most l^2 different values, an alphabet of size $l=O(\sqrt{n})$ is needed. Example 1.3 uses a bounded alphabet. However, the gap cost function w is not represented in the form (4) and, therefore, the example may not apply in practice. Furthermore, in all examples w and w' are different functions, which may not be applicable in practice. All examples assume that the matrix D is evaluated by columns. Some different evaluation order will yield a lower space bound.

1.1. An unbounded alphabet example

Assume a nonnegative weight function W that satisfies condition (2) and

$$W(i,k) > W(j,k) \quad \text{for every } i<j<k,$$

$$W(i,k)+W(j,l) > W(j,k)+W(i,l) \quad \text{for every } i<j<k<l. \tag{5}$$

Given such a function, we shall construct weight functions w and w' satisfying condition (2), so that applying Algorithm A for computing (3) requires $\Omega(n^2)$ space.

Definition 1.1. Let

$$K=\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor}[W(k-1,n-k+1)-W(k,n-k+1)]. \tag{6}$$

We define s, w and w' by

$$s(x_i,y_j)=\begin{cases} K & (i,j)=(1,1), \\ W(j-1,n-j+1)-W(j,n-j+1) & 1\leqslant j<i\leqslant \lfloor \frac{n}{2} \rfloor, \\ \text{any positive value} & \text{otherwise,} \end{cases}$$

$$w(i,j)=W(i,j)+K, \tag{7}$$

$$w'(i,j)=K.$$

For actual demonstration of this example, one may consider the function $W(i,j)=\log(j-i)$. It suffices to take $K=n$ in this case.

Lemma 1.2. (i) *For* $(i,j)\neq(0,0)$, $D[i,j]\geqslant K$; (ii) *For* $i\neq 0$ *and* $j\neq 0$, $E[i,j], F[i,j]\geqslant 2K$.

Proof (*by induction over* i,j). For $(i,0)$ and $(0,j)$ we have

$$D[i,0]=w'(0,i)=K\geqslant K, \qquad D[0,j]=w(0,j)=W(0,j)+K\geqslant K.$$

Assume that the lemma holds for each (i',j'), where $i'\leqslant i$, $j'<j$ or $i'<i$, $j'\leqslant j$. Now

$$D[i,j]=\min\{D[i-1,j-1]+s(x_i,y_j), E[i,j], F[i,j]\},$$

$$D[i-1,j-1]+s(x_i,y_j)\geqslant\begin{cases} D[i-1,j-1]\geqslant K & (i,j)\neq(1,1),\\ K & (i,j)=(1,1).\end{cases}$$

On the other hand, we have for some $j'<j$, $i'<i$,

$$E[i,j]=D[i,j']+w(j',j)\geqslant K+K=2K,$$

$$F[i,j]=D[i',j]+w'(i',i)\geqslant K+K=2K. \qquad \square$$

Lemma 1.3. *For* i,j *such that* $0\leqslant j\leqslant i\leqslant\lfloor\frac{n}{2}\rfloor$,

$$D[i,j]=w'(0,i-j)+\sum_{k=1}^{j} s(x_{i-j+k},y_k)=K+\sum_{k=1}^{j} s(x_{i-j+k},y_k). \tag{8}$$

Proof (*by induction over* j). For $j=0$ we have $D[i,0]=w'(0,i)$ and, thus, the lemma holds. For $j>0$, $D[i,j]=\min\{D[i-1,j-1]+s(x_i,y_j), E[i,j], F[i,j]\}$. We show that the minimum is achieved by $D[i-1,j-1]+s(x_i,y_j)$ and complete the proof:

$$\begin{aligned} &D[i-1,j-1]+s(x_i,y_j)\\ &\quad=K+\sum_{k=1}^{j-1} s(x_{i-j+k},y_k)+s(x_i,y_j)\\ &\quad=K+\sum_{k=1}^{j} s(x_{i-j+k},y_k)\\ &\quad=K+\sum_{k=1}^{j}[W(k-1,n-k+1)-W(k,n-k+1)]\\ &\quad\leqslant K+K=2K\leqslant E[i,j], F[i,j] \qquad \square\end{aligned}$$

Lemma 1.4. *Let* E_i *denote the* i*th row of* E. *For* i,j *such that* $1\leqslant j<i\leqslant\lfloor\frac{n}{2}\rfloor$, *step* j *of Algorithm* A *executed to compute* E_i, *given input satisfying Definition* 1.1, *consists of pushing the new entry* $(j,n-j+1)$ *onto the stack, without previous entries being popped out of the stack.*

Proof. For $1<i\leqslant\lfloor\frac{n}{2}\rfloor$, let E_i be an arbitrary row of E. We prove by induction over j that for $0\leqslant j<i$, the stack for computing E_i at the end of step j is

$$(k_{\text{top}}=j,\ h_{\text{top}}=n-j+1),\ (j-1,n-j+2),\ \ldots,\ (0,n+1) \tag{9}$$

and, thus, prove the lemma. For $j=0$, property (9) obviously holds because the initialization of the stack in Algorithm A pushes $(0,n+1)$. Assume that property (9) holds for $j-1$. The following inequalities, if maintained, ensure the required update of the stack at step j:

$$\begin{aligned} &D[i,j-1]+w(j-1,j+1)>D[i,j]+w(j,j+1),\\ &D[i,j-1]+w(j-1,n-j+1)\leqslant D[i,j]+w(j,n-j+1),\\ &D[i,j-1]+w(j-1,n-j)>D[i,j]+w(j,n-j). \end{aligned} \tag{10}$$

The first inequality means that j is a better candidate than $j-1$ for computing $E_i[j+1]$. The second and third inequalities ensure that $h(j-1,j)=n-j+1$.

By (7) and (8) we have

$$D[i,j]-D[i,j-1]=W(j-1,n-j+1)-W(j,n-j+1) \tag{11}$$

because

$$\begin{aligned} D[i,j]-D[i,j-1]&=K+\sum_{k=1}^{j}s(x_{i-j+k},y_k)-\left[K+\sum_{k=1}^{j-1}s(x_{i-j+k+1},y_k)\right]\\ &=s(x_i,y_j)+\sum_{k=1}^{j-1}[s(x_{i-j+k},y_k)-s(x_{i-j+k+1},y_k)]\\ &=s(x_i,y_j)=W(j-1,n-j+1)-W(j,n-j+1). \end{aligned}$$

As $1\leqslant j<\lfloor\frac{n}{2}\rfloor$, we have $j-1<j<j+1<n-j<n-j+1$. Using (5), (7), (8) and (11), we demonstrate that inequalities (10) hold:

$$\begin{aligned} &w(j-1,j+1)-w(j,j+1)\\ &\quad=W(j-1,j+1)-W(j,j+1)\\ &\quad>W(j-1,n-j+1)-W(j,n-j+1)=D[i,j]-D[i,j-1],\\ &w(j-1,n-j+1)-w(j,n-j+1)\\ &\quad=W(j-1,n-j+1)-W(j,n-j+1)\\ &\quad=D[i,j]-D[i,j-1],\\ &w(j-1,n-j)-w(j,n-j)\\ &\quad=W(j-1,n-j)-W(j,n-j)\\ &\quad>W(j-1,n-j+1)-W(j,n-j+1)=D[i,j]-D[i,j-1]. \end{aligned}$$

□

Theorem 1.5. *Algorithm* A *for computing* (3) *requires* $\Omega(n^2)$ *space for values of* s, w *and* w' *of Definition* 1.1.

Proof. Straightforward from Lemma 1.4.

1.2. An example with constant possible costs for substitution

Definition 1.6. Let $g(l)$, $l=1,\ldots,n$ be any nonnegative monotonically increasing convex function. We define values for s, w and w' as follows:

$$s(x_i, y_j)=s=g(2)-g(1),$$

$$w(i,j)=g(j-i)+2ns,$$

$$w'(i,j)=f^1(i)=(n-i)s+\sum_{k=1}^{i}[g(n-2k+2)-g(n-2k+1)]. \tag{12}$$

We shall use below the following fact: for $k\geqslant 1$, $g(k+1)-g(k)<s(=g(2)-g(1))$. It follows from the strict convexity of g.

Lemma 1.7. *For each* j *such that* $0\leqslant j\leqslant \lfloor \frac{n}{2} \rfloor$, $D[j,j]=js$,

Proof (*by induction over* j). For $j=0$ the lemma obviously holds. Suppose that the lemma holds for $j-1$. From (3) we have

$$D[j,j]=\min\{D[j-1,j-1]+s(x_j, y_j), E[j,j], F[j,j]\}.$$

Now

$$D[j-1,j-1]+s(x_j, y_j)=(j-1)s+s=js.$$

On the other hand, we have

$$E[j,j]=D[j,j']+w(j',j)\geqslant w(j',j)\geqslant 2ns>js \qquad \text{for some } j'<j,$$

$$F[j,j]=D[j',j]+w'(j',j)\geqslant w'(j',j)\geqslant (n-j')s\geqslant \lfloor \tfrac{n}{2} \rfloor s>js \qquad \text{for some } j'<j.$$

□

Lemma 1.8. *For* i', j *such that* $0\leqslant i'<j<\lfloor \frac{n}{2} \rfloor$, $D[i',j]\geqslant js$.

Proof (*by induction over* j). For $j=1$, i' must be 0 and $D[0,1]=w(0,1)=g(1)+2ns>js$. Assume that the lemma holds for $j-1$. If $D[i',j]=D[i'-1,j-1]+s(x_{i'}, y_j)$, then we have

$$D[i'-1,j-1]+s(x_{i'}, y_j)\geqslant (j-1)s+s=js.$$

If $D[i',j]=D[i',j']+w(j',j)$ for some $j'<j$, then we have

$$D[i',j']+w(j',j)\geqslant w(j',j)\geqslant 2ns>js.$$

If $D[i', j] = D[i'', j] + w'(i'', i')$ for some $i'' < i' < j < \lfloor \frac{n}{2} \rfloor$, then we have

$$D[i'', j] + w'(i'', i') \geqslant w'(i'', i') \geqslant (n - i'')s \geqslant \lfloor \tfrac{n}{2} \rfloor s > js. \qquad \square$$

Lemma 1.9. *For i, j such that $1 \leqslant j < i \leqslant \lfloor \frac{n}{2} \rfloor$,*

$$D[i, j] = D[j, j] + w'(j, i). \tag{13}$$

Proof (*by induction over i*). For $i = 2$, j must be 1. There are four possibilities for obtaining the value of $D[2, 1]$:

$$D[2, 1] = D[1, 1] + w'(1, 2) = ns + g(n) - g(n-1),$$

$$D[2, 1] = D[0, 1] + w'(0, 2) = w(0, 1) + w'(0, 2)$$

$$= 3ns + g(1) > ns + g(n) - g(n-1),$$

$$D[2, 1] = D[2, 0] + w(0, 1) = w'(0, 2) + w(0, 1)$$

$$= 3ns + g(1) > ns + g(n) - g(n-1),$$

$$D[2, 1] = D[1, 0] + s = w'(0, 1) + s = (n+1)s > ns + g(n) - g(n-1);$$

so, the lemma holds for $i = 2$, $j = 1$. Suppose that the lemma holds for each $i' < i$. First we prove that $F[i, j] = D[j, j] + w'(j, i)$. We then show that $F[i, j] \leqslant E[i, j]$ and $F[i, j] \leqslant D[i-1, j-1] + s$, by which we complete the proof.

Let $i', j \neq i' < i$, be an arbitrary candidate for computing $F[i, j]$. If $j < i' < i$, then

$$D[i', j] = D[j, j] + w'(j, i'),$$

and

$$D[i', j] + w'(i', i) = D[j, j] + w'(j, i') + w'(i', i) \geqslant D[j, j] + w'(j, i).$$

If $0 \leqslant i' < j < i$, then by Definition 1.6 and Lemma 1.8 we have

$$D[i', j] \geqslant js \geqslant i's + \sum_{k=i'+1}^{j} [g(n-2k+2) - g(n-2k+1)],$$

and

$$D[i', j] + w'(i', i) \geqslant ns + \sum_{k=1}^{j} [g(n-2k+2) - g(n-2k+1)] = D[j, j] + w'(j, i)$$

by Lemma 1.7. Therefore, $i' = j$ is the best candidate for computing $F[i, j]$ and $F[i, j] = D[j, j] + w'(j, i)$.

Now we show that $F[i, j] \leqslant E[i, j]$. For some $j' < j$

$$E[i, j] = D[i, j'] + w(j', j)$$

$$\geqslant w(j', j) \geqslant 2ns \geqslant ns + \sum_{k=1}^{j} [g(n-2k+2) - g(n-2k+1)] = F[i, j].$$

To complete the proof we show that $F[i,j] \leqslant D[i-1,j-1]+s(x_i,y_j)$. We have

$$\begin{aligned}
&D[i-1,j-1]+s(x_i,y_j)\\
&\quad = D[j-1,j-1]+w'(j-1,i-1)+s\\
&\quad = (j-1)s+(n-j+1)s+\sum_{k=1}^{j-1}[g(n-2k+2)-g(n-2k+1)]+s\\
&\quad = js+(n-j)s+\sum_{k=1}^{j}[g(n-2k+2)-g(n-2k+1)]\\
&\quad +g(n-2j+1)-g(n-2j+2)+s\\
&\quad = js+w'(j',i)+g(n-2j+1)-g(n-2j+2)+g(2)-g(1)\\
&\quad \geqslant js+w'(j,i)=F[i,j]. \qquad \square
\end{aligned}$$

Lemma 1.10. *Let E_i denote the ith row of E. For i,j such that $1 \leqslant j \leqslant i \leqslant \lfloor \frac{n}{2} \rfloor$, step j of Algorithm* A *executed to compute E_i, given input satisfying Definition* 1.6, *consists of pushing the new entry $(j, n-j+1)$ onto the stack, without previous entries being popped out of the stack.*

Proof. For $1 < i < \lfloor \frac{n}{2} \rfloor$, let E_i be an arbitrary row of E. As in Lemma 1.4, we shall prove this lemma by showing that the stack for computing E_i satisfies property (9) for each $0 \leqslant j < i$.

Given (12) and (13) we obtain that

$$D[i,j]-D[i,j-1]=g(n-2j+2)-g(n-2j+1) \tag{14}$$

since

$$\begin{aligned}
&D[i,j]-D[i,j-1]\\
&\quad = D[j,j]+w'(j,i)-D[j-1,j-1]-w'(j-1,i)\\
&\quad = js+(n-j)s+\sum_{k=1}^{j}[g(n-2k+2)-g(n-2k+1)]-(j-1)s-(n-j+1)s\\
&\quad -\sum_{k=1}^{j-1}[g(n-2k+2)-g(n-2k+1)]=g(n-2j+2)-g(n-2j+1).
\end{aligned}$$

Using (12), (13) and (14) we demonstrate the correctness of inequalities (10):

$$\begin{aligned}
&w(j-1,j+1)-w(j,j+1)\\
&\quad = g(2)-g(1) > g(n-2j+2)-g(n-2j+1)=D[i,j]-D[i,j-1],
\end{aligned}$$

$$w(j-1, n-j+1)-w(j, n-j+1)$$
$$=g(n-2j+2)-g(n-2j+1)$$
$$=D[i, j]-D[i, j-1],$$
$$w(j-1, n-j)-w(j, n-j)$$
$$=g(n-2j+1)-g(n-2j)$$
$$>g(n-2j+2)-g(n-2j+1)=D[i, j]-D[i, j-1]. \quad \square$$

Theorem 1.11. *Algorithm* A *for computing* (3) *requires* $\Omega(n^2)$ *space for values of* s, w *and* w' *given in Definition* 1.6.

Proof. Straightforward from Lemma 1.10. $\square$

1.3. A bounded alphabet example

Claim 1.12. *Let* $g(i)$ *be any concave function. Then* $w(i, j)=g(i+j)$ *satisfies the inverse quadrange inequality* (2).

Proof. Given $i \leqslant j \leqslant k \leqslant l$, we have $i+l \leqslant j+l$. Now $(i+l)-(i+k)=l-k=(j+l)-(j+k)$. Since g is concave, $g(i+l)-g(i+k) \leqslant g(j+l)-g(j+k)$ or $w(i, l)-w(i, k) \leqslant w(j, l)-w(j, k)$, which proves the claim. $\square$

Definition 1.13. Let $g(l), l=1, \ldots, 2n-1$ be any nonnegative monotonically decreasing concave function. We define values for s, w and w' by

$$s(x_i, y_j)=s=g(n)-g(n+1),$$
$$w(i, j)=g(i+j)+2ns, \tag{15}$$
$$w'(i, j)=ns.$$

Lemma 1.14. *For* j *such that* $0 \leqslant j < \lfloor \frac{n}{2} \rfloor$, $D[j, j]=js$.

Proof (*by induction over* j). For $j=0$ the lemma obviously holds. Suppose that the lemma holds for $j-1$. From (3) we have

$$D[j, j]=\min\{D[j-1, j-1]+s(x_i, y_j), E[j, j], F[j, j]\}.$$

Now

$$D[j-1, j-1]+s(x_i, y_j)=(j-1)s+s=js.$$

On the other hand, we have

$$E[j, j]=D[j, j']+w(j', j) \geqslant w(j', j) \geqslant 2ns > js \quad \text{for some } j' < j,$$
$$F[j, j]=D[j', j]+w'(j', j) \geqslant w'(j', j)=ns > js \quad \text{for some } j' < j. \quad \square$$

Lemma 1.15. *For i', j such that $0 \leqslant i' < j < \lfloor \frac{n}{2} \rfloor$, $D[i', j] \geqslant js$.*

Proof (*by induction over j*). For $j=1$, i' must be 0 and $D[0,1]=w(0,1)=g(1)+2ns \geqslant 2ns > js$. Assume that the lemma holds for $j-1$. If $D[i', j]=D[i'-1, j-1]+s(x_{i'}, y_j)$, then we have

$$D[i'-1, j-1]+s(x_{i'}, y_j) \geqslant (j-1)s+s=js.$$

If $D[i', j]=D[i', j']+w(j', j)$ for some $j' < j$, then we have

$$D[i', j']+w(j', j) \geqslant w(j', j) \geqslant 2ns > js.$$

If $D[i', j]=D[i'', j]+w'(i'', i')$ for some $i'' < i' < j < \lfloor \frac{n}{2} \rfloor$, then we have

$$D[i'', j]+w'(i'', i') \geqslant w'(i'', i')=ns > js. \qquad \square$$

Lemma 1.16. *For i, j such that $1 \leqslant j < i \leqslant \lfloor \frac{n}{2} \rfloor$, $D[i, j]=(n+j)s$.*

Proof (*by induction over i*). For $i=2$, j must be 1. There are four possibilities for obtaining the value of $D[2,1]$:

$$D[2,1]=D[1,1]+w'(1,2)=(n+1)s,$$

$$D[2,1]=D[0,1]+w'(0,2)=g(1)+3ns > (n+1)s,$$

$$D[2,1]=D[2,0]+w(0,1)=3ns+g(1) > (n+1)s,$$

$$D[2,1]=D[1,0]+s=(n+1)s;$$

so, the lemma holds for $i=2, j=1$. Suppose that the lemma holds for each $i' < i$ and let $1 \leqslant j < i \leqslant \lfloor \frac{n}{2} \rfloor$. Then we have

$$D[i-1, j-1]+s(x_i, y_j)=(n+j-1)s+s=(n+j)s.$$

Therefore,

$$E[i, j]=D[i, j']+w(j', j) \geqslant w(j', j) \geqslant 2ns > (n+j)s \quad \text{for some } j' < j,$$

$$F[i, j]=D[i', j]+w'(i', i) \quad \text{for some } i' < i.$$

If $i' > j$, then $D[i', j]=(n+j)s$; therefore, $F[i, j] \geqslant (n+j)s$. If $i' \leqslant j$, then by Lemmas 11 and 12, $D[i', j] \geqslant js$ and, therefore, $F[i, j] \geqslant js+ns=(n+j)s$. $\square$

Lemma 1.17. *Let E_i denote the ith row of E. For i, j such that $1 \leqslant j < i \leqslant \lfloor \frac{n}{2} \rfloor$, step j of Algorithm* A *executed to compute E_i, given input satisfying Definition* 1.13, *consists of pushing the new entry $(j, n-j+1)$ onto the stack, without previous entries being popped out of the stack.*

Proof. For $1<i\leqslant\lfloor\frac{n}{2}\rfloor$, let E_i be an arbitrary row of E. As in Lemma 1.4, we prove this lemma by showing that the stack for computing E_i satisfies property (9) for each $0\leqslant j<i$. By Lemma 1.16, it follows that

$$D[i,j]-D[i,j-1]=s. \tag{16}$$

Using (15) and (16) we demonstrate the correctness of inequalities (10). Note that for any $1\leqslant k\leqslant n-1$, $g(n)-g(n+1)<g(k)-g(k+1)$ because of the strict concavity of g:

$$\begin{aligned}
&w(j-1,j+1)-w(j,j+1)\\
&\quad=g(2j)-g(2j+1)>g(n)-g(n+1)\\
&\quad=D[i,j]-D[i,j-1],\\
&w(j-1,n-j+1)-w(j,n-j+1)\\
&\quad=g(n)-g(n+1)=D[i,j]-D[i,j-1],\\
&w(j-1,n-j)-w(j,n-j)\\
&\quad=g(n-1)-g(n)>g(n)-g(n+1)\\
&\quad=D[i,j]-D[i,j-1]. \qquad \square
\end{aligned}$$

Theorem 1.18. *Algorithm* A *for computing* (3) *requires* $\Omega(n^2)$ *space for values of* s, w *and* w' *of Definition* 1.13.

Proof. Straightforward from Lemma 1.17. $\square$

2. The concave case

Definition 2.1. Let $g(l)$, $l=1,\ldots,n$ be any nonnegative monotonically increasing concave function. Let $L=g(\lfloor\frac{n}{2}\rfloor+1)-g(\lfloor\frac{n}{2}\rfloor)$ and $K=\lfloor\frac{n}{2}\rfloor L$. We define values for s, w and w' by

$$s(x_i,y_j)=\begin{cases}K & (i,j)=(1,1),\\ L & \text{otherwise,}\end{cases}$$

$$w(i,j)=g(j-i)+K, \tag{17}$$

$$w'(i,j)=K$$

For actual demonstration of this example, one may consider the function $g(l)=l^2$. This gives $L=2\lfloor\frac{n}{2}\rfloor+1$. In this case K can be taken to be n^2.

Lemma 2.2. (i) *For* $(i,j)\neq(0,0)$, $D[i,j]\geqslant K$;
(ii) *For* $i\neq 0$ *and* $j\neq 0$, $E[i,j], F[i,j]\geqslant 2K$.

Proof. (*by induction over* i,j). For $(i,0)$ and $(0,j)$ we have

$$D[i,0]=w'(0,i)=K\geqslant K, \qquad D[0,j]=w(0,j)=g(j)+K\geqslant K.$$

Assume that the lemma holds for each (i',j'), where $i'\leqslant i$, $j'<j$ or $i'<i$, $j'\leqslant j$. Now

$$D[i,j]=\min\{D[i-1,j-1]+s(x_i,y_j),\ E[i,j], F[i,j]\},$$

$$D[i-1,j-1]+s(x_i,y_j)\geqslant\begin{cases}D[i-1,j-1]\geqslant K & (i,j)\neq(1,1),\\ K & (i,j)=(1,1).\end{cases}$$

On the other hand, we have for some $j'<j$, $i'<i$,

$$E[i,j]=D[i,j']+w(j',j)\geqslant K+g(j-j')+K\geqslant 2K,$$

$$F[i,j]=D[i',j]+w'(i',i)\geqslant K+K=2K. \qquad \square$$

Lemma 2.3. *For* i,j *such that* $0\leqslant j<i\leqslant\lfloor\frac{n}{2}\rfloor$,

$$D[i,j]=w'(0,i-j)+\sum_{k=1}^{j}s(x_{i-j+k},y_k)=K+jL. \tag{18}$$

Proof. By induction over j. For $j=0$ we have $D[i,0]=w'(0,i)$ and, thus, the lemma holds. For $j>0$, $D[i,j]=\min\{D[i-1,j-1]+s(x_i,y_j), E[i,j], F[i,j]\}$. We show that the minimum is achieved by $D[i-1,j-1]+s(x_i,y_j)$ and complete the proof:

$$D[i-1,j-1]+s(x_i,y_j)=K+(j-1)L+L$$

$$=K+jL<K+\left\lfloor\frac{n}{2}\right\rfloor L=2K\leqslant E[i,j], F[i,j]. \qquad \square$$

Lemma 2.4. *Let* E_i *denote the ith row of* E. *For* i,j *such that* $1\leqslant j<i\leqslant\lfloor\frac{n}{2}\rfloor$, *step* j *of Algorithm* B *executed to compute* E_i, *given input satisfying Definition* 2.1, *consists of enqueueing the new entry* $(j,\lfloor\frac{n}{2}\rfloor+j)$, *without previous entries being removed from the queue, then modifying* h_{front} *to be* $j+1$.

Proof. For $1<i\leqslant\lfloor\frac{n}{2}\rfloor$ let E_i be an arbitrary row of E. We prove the lemma by induction over j. Suppose the lemma holds for $j-1$, so that the queue for computing E_i at step $j-1$ is

$$(k_{\text{front}}=0, h_{\text{front}}=j),\ \left(1,\left\lfloor\frac{n}{2}\right\rfloor+1\right),\ \ldots,\ \left(j-1,\left\lfloor\frac{n}{2}\right\rfloor+j-1\right) \tag{19}$$

(obviously, this holds for $j=1$). The following inequalities, if maintained, ensure the required update of the queue at step j:

$$D[i,0]+w(0,j+1)<D[i,j]+w(j,j+1),$$

$$D[i,j-1]+w\left(j-1,\left\lfloor\frac{n}{2}\right\rfloor+j-1\right)<D[i,j]+w\left(j,\left\lfloor\frac{n}{2}\right\rfloor+j-1\right), \tag{20}$$

$$D[i,j-1]+w\left(j-1,\left\lfloor\frac{n}{2}\right\rfloor+j\right)\geqslant D[i,j]+w\left(j,\left\lfloor\frac{n}{2}\right\rfloor+j\right).$$

The first inequality means that 0 is a better candidate than j for computing $E_i[j+1]$. The second inequality means that at the start of stage j, $j-1$ is a better candidate than j. The second and third inequalities ensure that $h(j,j-1)=\lfloor\frac{n}{2}\rfloor+j$.

By (17) and (18) it follows that

$$D[i,j]-D[i,j-1]=L,$$

$$D[i,j]-D[i,0]=jL. \tag{21}$$

By (17), (18) and (21) we demonstrate the correctness of inequalities (20). Note that for $k\leqslant\lfloor\frac{n}{2}\rfloor$, $g(k)-g(k-1)<L$ because of the strict concavity of g:

$$\begin{aligned} w(0,j+1)-w(j,j+1)&=g(j+1)-g(1)\\ &=g(j+1)-g(j)+\cdots+g(2)-g(1)<jL\\ &=D[i,j]-D[i,0], \end{aligned}$$

$$\begin{aligned} w\left(j-1,\left\lfloor\frac{n}{2}\right\rfloor+j-1\right)-w\left(j,\left\lfloor\frac{n}{2}\right\rfloor+j-1\right)&=g\left(\left\lfloor\frac{n}{2}\right\rfloor\right)-g\left(\left\lfloor\frac{n}{2}\right\rfloor-1\right)<L\\ &=D[i,j]-D[i,j-1], \end{aligned}$$

$$\begin{aligned} w\left(j-1,\left\lfloor\frac{n}{2}\right\rfloor+j\right)-w\left(j,\left\lfloor\frac{n}{2}\right\rfloor+j\right)&=g\left(\left\lfloor\frac{n}{2}\right\rfloor+1\right)-g\left(\left\lfloor\frac{n}{2}\right\rfloor\right)\\ &=L=D[i,j]-D[i,j-1]. \end{aligned}$$

□

Theorem 2.5. *Algorithm* B *for computing* (3) *requires* $\Omega(n^2)$ *space for values of* s, w *and* w' *given in Definition* 2.1.

Proof. Straightforward from Lemma 2.4.

References

[1] Z. Galil and R. Giancarlo, *Speeding up dynamic programming with applications to molecular biology*, *Theoret. Comput. Sci.* **64** (1989) 107–118.

[2] D.S. Hirschberg, A linear space algorithm for computing maximal common subsequences, *Comm. ACM* **18** (1975) 341–343.

[3] W. Miller and E.W. Myers, Sequence comparison with concave weighting functions, *Bull. Math. Biol.* **50**(2) (1988) 97–120.