An Extension of the String-to-String Correction Problem

ROY LOWRANCE AND ROBERT A. WAGNER

Vanderbilt University, Nashville, Tennessee

ABSTRACT. The string-to-string correction problem asks for a sequence S of "edit operations" of minimal cost such that $S(A) = B$, for given strings A and B. The edit operations previously investigated allow changing one symbol of a string into another single symbol, deleting one symbol from a string, or inserting a single symbol into a string. This paper extends the set of allowable edit operations to include the operation of interchanging the positions of two adjacent characters Under certain restrictions on edit-operation costs, it is shown that the extended problem can still be solved in time proportional to the product of the lengths of the given strings.

KEY WORDS AND PHRASES: string correction, string modification, correction, spelling correction, permutations

CR CATEGORIES. 3.79, 4.12, 4.22, 5.23, 5.25

1. *Introduction*

In [1] a problem is described called the string-to-string correction problem. Given strings A and B, a set of "edit operations" taking strings into strings, and a set of weights for those operations, find a sequence of edit operations S such that the sum of the weights of the edit operations is minimal and S converts A to B. The following three edit operations and their weights were considered: (1) insert a character (weight W_I); (2) delete a character (weight W_D); and (3) change a character into any other character (weight W_C). An algorithm is presented in [1] which determines the minimal sum of weights for such an S which runs in time proportional to the product of the lengths of A and B.

We show how the set of edit operations can be extended to include: (4) interchange any two adjacent characters (weight W_S), and we present an algorithm for determining the minimal sum of weights in time proportional to the product of the lengths of the strings. (The time required remains of the same order as that needed by the algorithm in [1], but the constant of proportionality is larger.)

2. *Preliminaries*

For notation, we use A and B for strings, $|A|$ for the length of A (number of characters, possibly zero), $A\langle i:j\rangle$ for the substring $A\langle i\rangle \cdots A\langle j\rangle$ of A, and $A\langle i\rangle$ for $A\langle i:i\rangle$. If S is a sequence of edit operations, $S = S_1S_2 \cdots S_n$, by $S(A)$ we denote the function composition $S_1{}^\circ S_2{}^\circ \cdots {}^\circ S_n(A) = S_n(\cdots(S_1(A))\cdots) = S_1S_2 \cdots S_n(A)$.

A useful model has been developed in [1]. Given strings A and B, consider the diagram shown in Figure 1. A line from $A\langle i\rangle$ to $B\langle j\rangle$ indicates that $A\langle i\rangle$ should be changed to $B\langle j\rangle$, if $A\langle i\rangle \neq B\langle j\rangle$, or that $A\langle i\rangle$ is not to be disturbed, but becomes $B\langle j\rangle$, if $A\langle i\rangle = B\langle j\rangle$. Characters of A not touched by a line are to be deleted and characters of B not

This work was supported by the NSF under Grant GJ-33014.

Authors' present addresses: R. Lowrance, 255 West Squire Drive, Rochester, NY 14623; R. A Wagner, Department of Systems and Information Sciences, Vanderbilt University, Nashville, TN 37235

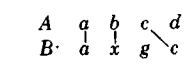


FIG. 1

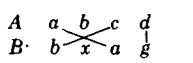


FIG. 2

touched are to be inserted. Such a diagram is called a *trace*. It gives a method for changing string A into string B. We extend this model by allowing lines to cross, as shown in Figure 2. Lines which cross indicate that the characters are to be interchanged. More formally, if A and B are strings, we define $T = [U, A, B]$ to be a *trace* from A to B provided that

(1) $U \subseteq X_A \times X_B$, where $X_C = \{i \mid i \text{ integer, and } 1 \leq i \leq |C|\}$;

(2) if (u_1, j_1) and $(i_2, j_2) \in U$, $i_1 = i_2$ if and only if $j_1 = j_2$.

Thus X_A and X_B are the point sets derived from A and B, and U is a collection of ordered pairs (i, j) interpreted to be line segments joining character $A\langle i\rangle$ to $B\langle j\rangle$. For notation, if $u \in U$, we will usually set $u = (u_1, u_2)$. We say u and v *cross* just when $u_1 < v_1$ if and only if $u_2 > v_2$. Each pair of lines which cross gives rise to one *line crossing*. If $u = (i, j) \in U$, we say that u is a *balanced* line of U provided that $A\langle i\rangle = B\langle j\rangle$; we say u is an *unbalanced* line if $A\langle i\rangle \neq B\langle j\rangle$.

With each trace $T = [U, A, B]$ from A to B, we associate its cost $C(T)$, the sum of

(1) $W_D \times$ (the number of points in X_A not adjacent to a line of U);

(2) $W_I \times$ (the number of points in X_B not adjacent to a line of U);

(3) $W_C \times$ (the number of unbalanced lines of U); and

(4) $W_S \times$ (the number of line crossings in U).

We wish to convert the search for a minimal weight sequence of edit operations to a search for a minimal cost trace. Section 3 justifies that conversion.

3. *Traces and Edit Operations*

LEMMA 1. *Let T be a trace from A to B. If u and v cross and $|u_1 - v_1| = 1$, then there is a trace $T' = [U', A', B]$ from A' to B such that (1) $C(T') = C(T) - W_S$; and (2) A' is derived from A by interchanging characters u_1 and v_1 of A.*

PROOF. Let $U' = U - \{u, v\} \cup \{(u_1, v_2), (v_1, u_2)\}$. The operations on U amount to detaching u and v from u_1 and v_1 of A and then reattaching to v_1 and u_1 of A', respectively. This decreases the number of line crossings by one and leaves the other three terms in the sum $C(T)$ unchanged. Hence $C(T') = C(T) - W_S$.

Definition. A trace T is *reduced* if it contains no lines which cross.

LEMMA 2. *Let T be a reduced trace from A to B. There is a sequence of edit operations S from $S(A)$ to B such that (1) $W(S) = C(T)$; and (2) $S(A) = B$.*

PROOF. A proof can be found in [1].

LEMMA 3. *Let $T = [U, A, B]$ be a trace from A to B satisfying: (1) each point of X_A is adjacent to some line of U, and (2) T contains N line crossings. There is a sequence S of edit operations and a trace $T' = [U', S(A), B]$ such that (1) $W(S) = N \times W_S$; (2) $C(T') = C(T) - W(S)$; and (3) T' contains no line crossings.*

PROOF. If $N = 0$, then we take $T' = T$ and $S = \varnothing$, so we may assume $N > 0$. Since all points of X_A are adjacent to elements of U and since $N > 0$, there is a pair of lines u and v in U which cross with $|u_1 - v_1| = 1$. Apply Lemma 1 to build a trace T_1. Add to S the edit operation "interchange u_1 and v_1." T_1 satisfies $C(T_1) = C(T) - W_S$ and T_1 contains one less line crossing from T. Continuing in this manner, we can construct traces $T = T_0, T_1, T_2, \cdots, T_N = T'$ such that $C(T_{k+1}) = C(T_k) - W_S$ and T_{k+1} is derived from T_k by the application of an interchange operation.

THEOREM 1. *Let T be a trace from A to B. There is a sequence S of edit operations such that (1) $W(S) = C(T)$, and (2) $S(A) = B$.*

PROOF. Let there be N_D points in X_A not touched by a line, N_I points in X_B not touched by a line, N_C unbalanced lines, and N_S line crossings. Place one delete operation in S_1 for each point of X_A not adjacent to a line and consider the simplified trace T_1 which results from removing these points of X_A. T_1 is a trace from $S_1(A)$ to B and satisfies

$C(T_1) = C(T) - N_D \times W_D$. All points of $X_{S_1(A)}$ are adjacent to a line, and $W(S_1) = N_D \times W_D$. For each of the N_S line crossings in T_1, add an interchange operation to S_2 and consider the reduced trace T_2 which results from operations on T_1 as specified in Lemma 3. T_2 is a trace from $S_2(S_1(A))$ to B, $W(S_2) = N_S \times W_S$, $C(T_2) = C(T_1) - N_S \times W_S$. Since T_2 contains no line crossings, Lemma 2 implies the existence of a set of edit operations S_3 such that $W(S_3) = C(T_2)$ and $S_3(S_2(S_1(A))) = B$. Thus

$$W(S) = W(S_1S_2S_3) = W(S_1) + W(S_2) + W(S_3) = C(T_2) + W(S_2) + N_D \times W_D .$$

Now it is only necessary to show that $W(S_2) = N_S \times W_S$. This follows from Lemma 3. Hence $W(S) = C(T)$.

THEOREM 2. *Let S be a sequence of edit operations on A. Let $B = S(A)$. There is a trace T from A to B such that $C(T) \leq W(S)$.*

PROOF. We construct T. Imagine that each character $A\langle i\rangle$ of A is written on a slip of paper together with its position i. Apply the sequence of edit operations to the ordered slips of paper as follows:

(1) a delete operation discards a slip of paper;

(2) an insert operation introduces a new slip of paper on which appear the inserted character and the number 0;

(3) a change operation replaces a character by another character;

(4) an interchange operation interchanges the positions of two slips of paper.

After the edit operations are performed, a sequence $C_1, \cdots, C_n$ of numbered slips remains.

Let $U = \{(\text{number}(C_k), k) \mid 1 \leq k \leq n, \text{number}(C_k) \neq 0\}$, where number (C) is the number appearing on slip C. If $T = [U, A, B]$, $C[T] \leq W(S)$, since:

(1) any character $A\langle i\rangle$ of A not touched by a line $(i, j) \in U$ satisfies $\forall_j \in \{1, \cdots, n\}$, $\text{number}(C_j) \neq i$, and hence $A\langle i\rangle$ must have been deleted;

(2) any character $B\langle i\rangle$ of B not touched by a line $(i,j) \in U$ satisfies $\forall_i \in \{1, \cdots, n\}$, $\text{number}(C_i) \neq j$, and $B\langle j\rangle$ must therefore have been inserted;

(3) any line (i, j) for which $A\langle i\rangle \neq B\langle j\rangle$ results from at least one change operation effecting $A\langle i\rangle$; and

(4) a pair of lines which cross must correspond to at least one interchange operation on A.

Thus each component of $C(T)$ is individually less than or equal to the number of occurrences of the corresponding component of $W(S)$. Hence, $C(T) \leq W(S)$.

4. *Properties of Traces*

Definition. If $[T_1, A_1, B_1]$ and $[T_2, A_2, B_2]$ are traces, then we say that $\langle T_1, T_2\rangle$ constitutes a *partition* of the trace $[T, A_1A_2, B_1B_2]$ just when $T = T_1 \cup (T_2 + (|A_1|, |B_1|))$, where $R + (i, j)$ means $\{(u + i, v + j) \mid (u, v) \in R\}$.

Since no lines of $T_2 + (|A_1|, |B_1|) = T_2'$ cross any lines of T_1 in T, and each line of T_1 and T_2' touches exactly the same characters of A_i and B_i that the corresponding line of $[T_i, A_i, B_i]$ touches, we have

$$C(T) = \sum_{i=1}^{2} C(T_i). \tag{1}$$

An obvious generalization allows a partition of any number k of components $T_1, \cdots, T_k$ to be constructed satisfying (1). Of course, given $[T, A, B]$, we may be able to find T_1, T_2, A_1, A_2, B_1, B_2 such that $T = T_1 \cup T_2'$, $T_1 \cap T_2' = \varnothing$, $T_2' = T_2 + (|A_1|, |B_1|)$, $A = A_1A_2$, $B = B_1B_2$, and $[T_i, A_i, B_i]$ is a trace, $i = 1, 2$. This is possible nontrivially whenever T can be partitioned into two nonempty sets T_1 and T_2' such that no line of T_1 crosses a line of T_2'. (When T cannot be so partitioned, T cannot, apparently, be partitioned into *traces* T_1 and T_2' with property (1).)

For certain choices of weights W_S, W_I, W_D, and W_C, any minimal trace may be par-

titioned into components containing at most two lines. One such choice of weights is

$$2W_S \geq W_I + W_D . \tag{2}$$

In what follows, we will investigate the consequence of this choice of weights and develop an algorithm which finds a minimal trace quickly, whenever the weights satisfy (2).

THEOREM 3. *If* $2W_S \geq W_I + W_D$, *there is at least one minimal trace* T *from* A *to* B *such that no line of* T *crosses more than one other line.*

PROOF. Suppose $[T, A, B]$ is a minimal trace containing a line u which crosses $n \geq 2$ other lines. Let T' be $T - \{u\}$. Then $C(T') = C(T) - nW_S + W_I + W_D$, since $nW_S \geq 2W_S \geq W_I + W_D$, $C(T') \leq C(T)$.

THEOREM 4. *If* $2W_S \geq W_I + W_D$, *then at least one minimal cost trace* $[T, A, B]$ *exists such that every line of* T *crosses at most one other line and such that every line of* T *which crosses another line is balanced.*

PROOF. Suppose $[T, A, B]$ is a minimal cost trace from A to B satisfying Theorem 3 and that $u \in T$ is an unbalanced line which crosses another line v of T, where $u = (u_1, u_2)$, $v = (v_1, v_2)$, and $v_1 < u_1$. Let $T' = T - \{u, v\}$. Suppose v is unbalanced. Then

$$C(T) = C(T') + 2W_C + W_S - 2W_D - 2W_I .$$

Let $u' = (u_1, v_2)$, $v' = (v_1, u_2)$, and $T'' = T' \cup \{u', v'\}$. Then

$$C(T'') \leq C(T') + 2W_C - 2W_D - 2W_I \leq C(T),$$

so T'' is also minimal, T'' has two fewer unbalanced crossing lines than T, and T'' has no additional crossings.

Suppose v is balanced. Then

$$C(T) = C(T') + W_C + W_S - 2W_D - 2W_I ; \quad C(T'') \leq C(T') + 2W_C - 2W_D - 2W_I .$$

Two cases arise: (1) $W_C \leq W_S$, and (2) $W_C > W_S$. In case (1), $C(T'') \leq C(T)$, so T'' is again minimal, with one fewer unbalanced crossing line than T and no additional crossings. In case (2), $W_C > W_S$.

Consider $T''' = T' \cup \{v\}$.

$$\begin{aligned} C(T''') &= C(T') - W_I - W_D , \\ C(T) &= C(T') + W_C + W_S - 2W_I - 2W_D \\ &\geq C(T') + 2W_S - 2W_I - 2W_D \\ &\geq C(T') - W_I - W_D = C(T''') . \end{aligned}$$

T''' has one fewer unbalanced crossing line than T and no additional crossings. The theorem follows by induction on the number of unbalanced crossings.

THEOREM 5. *If* $2W_S \geq W_I + W_D$, *there is at least one minimal trace* $[T, A, B]$ *satisfying Theorems* 3 *and* 4 *and such that, if* $u = (u_1, u_2)$ *and* $v = (v_1, v_2)$ *are lines in* T *which cross, with* $u_1 < v_1$, *then there are no integers* i *(or* j*) such that*

(1) $u_1 < i < v_1$ *and* $A\langle u_1 \rangle = A\langle i \rangle$ *or, symmetrically,*

(2) $v_2 < j < u_2$ *and* $B\langle v_2 \rangle = B\langle j \rangle$.

PROOF. Suppose T is a trace satisfying Theorems 3 and 4 and suppose i exists as in Theorem 5(1). Let $T' = T - \{u\} \cup \{(i, u_2)\}$.

Since $A\langle i \rangle = A\langle u_1 \rangle$, (i, u_2) is balanced if and only if u is balanced. By Theorem 3, u crosses only v. Therefore, no lines (k, l) exist in T such that $u_1 < k < v_1$, since any such lines must cross either line u or line v. In T', (i, u_2) still crosses only v. So $C(T') = C(T)$. Similarly, if j exists as in Theorem 5(2), v can be replaced by (v_1, j) without changing the cost of T. Repeated application of these operations produces a minimal trace with no lines u or v which cross and satisfy Theorem 5(1) or (2).

Any trace T which satisfies properties (i)–(iii) below is said to be a *restricted* trace.

(i) No line of T crosses more than one other line.

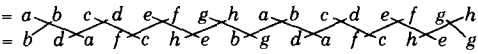
(ii) Every line of T which crosses another line is balanced.

(iii) If $u = (u_1, u_2)$ and $v = (v_1, v_2)$ are lines of T which cross, with $u_1 < v_1$, then there are no integers i (or j) such that (1) $u_1 < i < v_1$ and $A\langle u_1\rangle = A\langle i\rangle$ or (2) $v_2 < j < u_2$ and $B\langle v_2\rangle = B\langle j\rangle$.

If edit-operation weights W_S, W_I, and W_D satisfy (*) $2W_S \geq W_I + W_D$, then, for every minimal cost trace T, there exists a restricted trace T' with $C(T') \leq C(T)$. Since T' is also a trace, $C(T') \geq C(T)$ as well, for $C(T)$ was minimal. Thus, whenever (*) holds, the search for a minimal cost trace need not consider any nonrestricted trace.

We note that (*) holds when $W_I = W_D = W_S = W_C$, and hence the problem of finding a minimal *length* sequence of edit operations reduces to a search for a minimal cost restricted trace whose weights are $W_I = W_D = W_S = W_C = 1$.

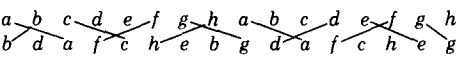
When (*) does not hold, a much more complicated situation exists. In particular, for proper choices of weights (take $W_C > W_I + W_D > 2W_S$), the minimal cost trace T may be a "daisy-chain," containing lines which touch every character of A and of B, but such that no partition of T exists. For example,



(3)

With $W_S = 1$, $W_I = 1$, $W_D = 2$, $W_C = 4$, the cost of the above trace is $15W_S = 15$.

A minimal cost restricted trace[1] is



(4)

Its cost is $5W_S + 5(W_I + W_D) = 20$. At present, we know of no algorithm of speed comparable to the algorithm presented in Section 5, which can find a minimal cost trace when such a trace can consist of arbitrarily long "chains" of crossed lines.

5. *An Algorithm*

We present an algorithm which finds a minimum cost restricted trace. This algorithm solves the extended string-to-string correction problem whenever $2W_S \geq W_I + W_D$.

Let $T[i, j]$ denote a trace from $A\langle 1{:}i\rangle$ to $B\langle 1{:}j\rangle$ and let $T^*[i, j]$ denote a restricted trace of minimal cost. Define $H[i, j] = C(T^*[i, j])$. Suppose that for $0 \leq k \leq i$ and $0 \leq n \leq j$, $H[k, n]$ are known when $k < i$ or $n < j$, and we wish to calculate $H[i, j]$. There are four possibilities.

(1) $A\langle i\rangle$ is not touched by a line of $T^*[i, j]$. Then $H[i, j] = H[i - 1, j] + W_D$.

(2) $B\langle j\rangle$ is not touched by a line of $T^*[i, j]$. Then $H[i, j] = H[i, j - 1] + W_I$.

(3) $A\langle i\rangle$ and $B\langle j\rangle$ are touched by the same line of $T^*[i, j]$. Then $H[i, j] = H[i - 1, j - 1] + d$, where $d = W_C$ if $A\langle i\rangle \neq B\langle j\rangle$ and $d = 0$ otherwise.

(4) $A\langle i\rangle$ and $B\langle j\rangle$ are touched by different lines of $T^*[i, j]$. Let (x, j) and (i, y) be the lines. Then $H[i, j] = H[x - 1, y - 1] + (i - x - 1) \times W_D + W_S + (j - y - 1) \times W_I$, since by Theorem 3, no other lines of $T^*[i, j]$ cross these (necessarily mutually crossing) lines.

Consider case (4). By Theorem 4, we know that we may take $A\langle x\rangle = B\langle j\rangle$ and $A\langle i\rangle = B\langle y\rangle$. Also, by Theorem 5, x must be the largest integer less than or equal to i such that $A\langle x\rangle = B\langle j\rangle$. Likewise, y is the largest integer less than or equal to j such that $B\langle y\rangle = A\langle i\rangle$. These considerations suggest recording, for each pair (i, a), where i is an integer and a is an alphabet character, the largest $x \leq i$ such that $A\langle x\rangle = a$, and likewise for string B. This function can be computed in time proportional to $|A| + |B|$. To conserve storage space, however, some of the function values will be recomputed each time they are needed. In the algorithm, we will need to represent each character by an integer. We use $A[i]$ for the integer representing $A\langle i\rangle$ and $B[j]$ likewise. The range of $A[i]$ and $B[j]$ lies in $\{1, 2, \cdots, C\}$.

[1] Trace (4) was discovered by the algorithm of Section 5 and thus is known to be a minimal cost restricted trace.

ALGORITHM S

```
 1. INF ← | A | * W_D + | B | * W_I + 1;
 2. for i ← 0 step 1 until | A | do
 3.   begin H[i, 0] ← i * W_D;
 4.   H[i, −1] ← INF end;
 5. for j ← 1 step 1 until | B | do
 6.   begin H[0, j] ← j * W_I ;
 7.   H[−1, j] ← INF end;
 8. for d ← 1 step 1 until C do DA[d] ← 0;
 9 for i ← 1 step 1 until | A | do
10.   begin DB ← 0;
11.   for j ← 1 step 1 until | B | do
12.     begin i1 ← DA[B[j]];
13.     j1 ← DB·
```

Comment: Each time execution reaches this point, $DA[c]$ holds the largest $x \leq i - 1$ such that $A[x] = c$, for every character c. Also, DB holds the largest $y \leq j - 1$ such that $B[y] = A[i]$. These computations permit the only possible x and y to be calculated in constant time when the pair of crossing lines (x, j) and (i, y) are to be examined as candidates for inclusion in $T^*[i, j]$;

```
14.     d ← if A[i] = B[j] then 0 else W_C ;
15.     if A[i] = B[j] then DB ← j;
16  L1. H[i, j] = min(H[i−1, j−1] + d,
17.                    H[i, j−1] + W_I ,
18.                    H[i−1, j] + W_D ,
19                     H[i1−1, j1−1] + (i−i1−1) × W_D + W_S + (j−j1−1) × W_I)
20      end;
21.   DA[A[i]] ← i
22.   end;
```

We note that the value chosen for INF exceeds the cost of the "empty" trace (which contains no lines) from A to B. The value of INF therefore exceeds the cost of the minimum cost restricted trace from A to B and also the cost of any trace $T^*[i, j]$. Thus, the *min* operation in lines 16–19 will never assign a value greater than or equal to INF to $H[i, j]$.

To show that the assignment statement labeled L1 is correct, we define an auxiliary function G as

$$G(X, c, i) = \begin{cases} 0, & \text{if } i = 0; \\ i, & \text{if } i > 0 \text{ and } X[i] = c; \\ G(X, c, i - 1), & \text{if } i > 0 \text{ and } X[i] \neq c. \end{cases}$$

Thus, $G(X, c, i)$ is the largest $k \leq i$ such that $X[k] = c$. We claim that just before the line labeled L1 is executed,

(1) $DB = G(B, A[i], j)$;
(2) $DA[d] = G(A, d, i - 1)$ for each $d \in \{1, 2, \cdots, C\}$,
(3) $d = \begin{cases} 0, & \text{if } A[i] = B[j], \\ W_C, & \text{if } A[i] \neq B[j]; \end{cases}$
(4) $i1 = G(A, B[j], i - 1)$;
(5) $j1 = G(B, A[i], j - 1)$.

To show (1), let DB_k be the value of DB after line 15 has been executed exactly k times following an execution of line 10. Since line 15 is executed for increasing values of j for fixed i, and since $DB_0 = 0$ (line 10), when line 15 has been executed k times, following an execution of line 10 we have

$$DB_k = \begin{cases} 0, & \text{if } k = 0; \\ k, & \text{if } A[i] = B[k] \text{ and } k > 0; \\ DB_{(k-1)}, & \text{if } A[i] \neq B[k] \text{ and } k > 0. \end{cases}$$

This is the definition of $G(B, A[i], k)$.

We show (2) by induction on i. When i is 1, DA is a vector of zeros since it is initialized in line 8 and not altered in lines 9–20 (we have used the fact that the algorithm contains no GOTOs). Suppose that for $i = k$, we know $DA[d] = G(A, d, k - 1)$ for each $d \in \{1, 2, \cdots, C\}$. The assignment to DA in line 21 which occurs when $i = k$ alters DA so that $DA[A[k]] = k$. On the next iteration of the for $-i$ loop, i is $k + 1$ and

$$DA[d] = \begin{cases} G(A, d, k - 1), & \text{if } d \neq A[k]; \\ k, & \text{if } d = A[k]. \end{cases}$$

This is the defining formula for $G(A, d, k)$.

To show (3), note that only line 14 changes d, after line 10; it is apparent that (3) is satisfied.

To show (4), recall that $DA[d]$ is $G(A, d, i - 1)$ and note that the only assignment to $i1$ (line 12) forces $i1$ to become $G(A, B[j], i - 1)$.

To show (5), recall that we have shown $DB = G(B, A[i], j)$ whenever L1 is executed. It is easy to modify the argument to show that just before line 15 is executed, $DB = G(B, A[i], j - 1)$. This relation also holds at line 13.

6. *Summary*

We have derived an algorithm which computes the smallest edit-operation distance between two strings, when given two strings A and B and a set of weights for edit operations such that $2W_S \geq W_I + W_D$. The algorithm uses space and time proportional to $|A| \times |B|$. If the actual sequence of edit operations is needed, it may be derived by looking back through H and finding the choices made at L1 by the algorithm.

REFERENCE

1. WAGNER, R. A., AND FISCHER, M. J. The string-to-string correction problem. *J. ACM 21*, 1 (Jan. 1974), 168–173

RECEIVED SEPTEMBER 1973; REVISED JULY 1974