A SPECIAL CASE OF THE MAXIMAL COMMON SUBSEQUENCE PROBLEM

Thomas G. Szymanski

Technical Report #170
January 1975

PRINCETON UNIVERSITY
Department of Electrical Engineering
Computer Science Laboratory

Part of this work was supported by NSF Grant GJ-1052.

A SPECIAL CASE OF THE MAXIMAL COMMON SUBSEQUENCE PROBLEM

Thomas G. Szymanski

Princeton University

Abstract

We present an algorithm for finding the maximal common subsequence of two sequences under the restriction that each element of either sequence occurs at most once in the other sequence. The running time of the algorithm is shown to be proportional to n log n where n is the length of the sequences in question. If the elements of the sequences are selected from the first n integers, then the algorithm can be modified to run in time O(n log log n).

Introduction

Let A be a finite sequence. We denote the length of A by $|A|$. A[i] is the ith element of A and A[i:j] denotes the sequence A[i], A[i+1], ..., A[j].

If U and V are finite sequences, then U is said to be a subsequence of V if there exist integers $1 \le r_1 < r_2 < \ldots < r_{|U|} \le |V|$ such that $U[i] = V[r_i]$ for $1 \le i \le |U|$. U is a common subsequence of A and B if U is a subsequence of both A and B. A maximal common subsequence is a common subsequence of greatest possible length. The best known algorithms [1,2,3] for solving the general maximal common subsequence problem require time proportional to the product of the sequences in question.

In this paper we shall consider the special case in which each element of either sequence occurs at most once in the other. This restricted version of the problem can be shown to be equivalent to the following combinatorial problems.

1) Given a permutation of the integers 1 thru n, what is the maximal ascending subsequence of this permutation?

2) Let C be a finite collection of vectors in 2-space, partially ordered in the natural fashion. What is the largest linearly ordered subset of C?

Preliminary Results

Throughout this paper we shall use A and B to denote the sequences in question. We shall assume without loss of generality that each element of either sequence occurs exactly once in the other sequence (after all, an element of one sequence which does not occur in the other certainly cannot be a member of a common subsequence). The lengths of the sequences will then be denoted by n. The map j:{1,...,n} -> {1,...,n} will be used to associate elements of A with their "mates" in B, that is, A[i] = B[j(i)] for 1 ≤ i ≤ n.

We shall next introduce some notation for the length of the shortest prefix of B which contains a subsequence of length k in common with the first i elements of A.

Definition 1:

For 0 ≤ i ≤ n and 0 ≤ k, define

$$T_{i,k} = \begin{cases} 0 & \text{if } k = 0 \\ \min \{j \mid A[1{:}i] \text{ and } B[1{:}j] \text{ have a common subsequence of length } k\} & \text{if } k \neq 0. \end{cases}$$

We adopt the convention that min {} = ∞. Thus if A = 'abcd' and B = 'bdca', we would have $T_{1,1} = 4$ and $T_{1,2} = \infty$.

The efficient computation of $T_{i,k}$ for all i and k is a key step in our algorithm. Let us therefore derive several properties of these values.

Proposition 2:

For $1 \le i \le n$ and $0 \le k$, $T_{i,k} \le T_{i-1,k}$.

Proof: Immediate from the observation that every common subsequence of A[1:i-1] and B[1:j] is also a common subsequence of A[1:i] and B[1:j]. Thus it is impossible that $T_{i,k}$ be greater than $T_{i-1,k}$.

Lemma 3:

For $0 \le i \le n$ and $1 \le k$,

$T_{i,k} < \infty$ implies $T_{i,k-1} < T_{i,k}$.

Proof:

Case 1: k = 1.

Then we must have $i \ge 1$ which implies that $T_{i,k}$ is non-zero. Since $T_{i,k-1} = T_{i,0} = 0$, we have $T_{i,k-1} < T_{i,k}$.

Case 2: k > 1.

Since $T_{i,k}$ is finite, A[1:i] and B[1:$T_{i,k}$] contain a common subsequence of length k but A[1:i] and B[1:$T_{i,k}$-1] do not. Since this latter pair of sequences quite clearly contain a common subsequence of length k-1, we must therefore have $T_{i,k-1} \le T_{i,k}-1$, which establishes the lemma.

Lemma 4:

For $1 \le i \le n$ and $1 \le k$,

$$T_{i,k} = \begin{cases} \min \{T_{i-1,k}, j(i)\} & \text{if } j(i) > T_{i-1,k-1} \\ T_{i-1,k} & \text{otherwise.} \end{cases}$$

Proof: We consider the two cases.

Case 1: $j(i) > T_{i-1,k-1}$.

Clearly $j(i)-1 \ge T_{i-1,k-1}$, which means that A[1:i-1] and B[1:j(i)-1] contain a common subsequence of length k-1. Therefore, A[1:i] and B[1:j(i)] must contain a common subsequence of length k. We conclude that $T_{i,k} \le j(i)$. Combining this with Proposition 2, we establish

(1) $$T_{i,k} \le \min \{T_{i-1,k}, j(i)\}.$$

Since $T_{i,k}$ has thus been shown to be finite, we see that A[1:i] and B[1:$T_{i,k}$] contain some common subsequence S of length k. Let us suppose that

(2) $$T_{i,k} < \min \{T_{i-1,k}, j(i)\}.$$

By (2), $j(i) > T_{i,k}$ and so B[j(i)] = A[i] is not a member of S. Thus S is also a common subsequence of A[1:i-1] and B[1:$T_{i,k}$]. Since $|S| = k$, we conclude that $T_{i-1,k} \le T_{i,k}$ which clearly violates supposition (2). We conclude then that $T_{i,k} \ge \min \{T_{i-1,k}, j(i)\}$ which combined with (1) above

yields $T_{i,k} = \min \{T_{i-1,k}, j(i)\}$.

Case 2: $j(i) \leq T_{i-1,k-1}$.

Let us suppose that

(3) $$T_{i,k} < T_{i-1,k}.$$

Then A[1:i] and B[1:$T_{i,k}$] contain a common subsequence S of length k, but A[1:i-1] and B[1:$T_{i,k}$] do not. Thus A[i] must be the last element of S. Therefore A[1:i] and B[1:j(i)] contain S and hence A[1:i-1] and B[1:j(i)-1] contain a common subsequence of length k-1. Thus $T_{i-1,k-1} < j(i)$ which violates the hypothesis for this case. Since supposition (3) was incorrect, we must have $T_{i,k} \geq T_{i-1,k}$ which combined with Proposition 2 yields $T_{i,k} = T_{i-1,k}$.

The next theorem implies a simple inductive method for computing all $T_{i,k}$.

Theorem 5:

For $1 \leq i \leq n$ and $1 \leq k$,

$$T_{i,k} = \begin{cases} j(i) & \text{if } k = T_{i-1,k-1} < j(i) < T_{i-1,k} \\ T_{i-1,k} & \text{otherwise.} \end{cases}$$

Proof:

Case 1: $j(i) \leq T_{i-1,k-1}$.

Then $T_{i,k} = T_{i-1,k}$ by Lemma 4.

Case 2: $T_{i-1,k-1} < j(i) < T_{i-1,k}$.

By Lemma 4, we have $T_{i,k} = \min\{T_{i-1,k}, j(i)\} = j(i)$.

Case 3: $T_{i-1,k} \leq j(i)$.

Thus $T_{i-1,k}$ is finite. By Lemma 3, we must therefore have $T_{i-1,k-1} < T_{i-1,k}$ and so $T_{i-1,k-1} < j(i)$. Applying Lemma 4 again, we have $T_{i,k} = \min\{T_{i-1,k}, j(i)\} = T_{i-1,k}$.

Our next piece of notation assigns to each position of the A sequence (and its matching B position) the length of the longest common subsequence appearing to its left.

Definition 6:

Define L_i to be the length of the longest common subsequence of A[1:i] and B[1:j(i)].

As we shall soon see, a maximal common subsequence may easily be constructed if we know the value of L_i for $1 \leq i \leq n$. The next theorem relates the L_i's to the $T_{i,k}$'s.

Theorem 7:

If $T_{i,k} = j(i)$ then $L_i = k$.

Proof: Since j(i) is finite, we can conclude from Lemma 3 that $T_{i,k+1} > j(i)$. That is, A[1:i] and B[1:j(i)] contain a common subsequence of length k but none of length k+1. Hence $L_i = k$.

The Algorithm

We can now present our algorithm.

Algorithm A:

Input: sequences A[1:n] and B[1:n] such that each element of A occurs exactly once in B and vice versa.

Output: a maximal common subsequence of A and B.

```
         begin
            integer array L[1:n], T[0:n+1], J[1:n];
            sequence SEQ[1:n];
step1:      for i <- 1 to n do
               J[i] <- the unique j such that A[i] = B[j];
step2:      T[0] <- 0;
            for k <- 1 to n+1 do
               T[k] <- "∞";
step3:      for i <- 1 to n do
               begin
                  k <- unique k such that T[k-1] < J[i] < T[k];
                  T[k] <- J[i];
                  L[i] <- k;
               end;
step4:      lastj <- n+1;
            kmax <- max {L[i] | 1 ≤ i ≤ n};
            k <- kmax;
          * for i <- n downto 1 do
               if L[i] = k and J[i] < lastj then
                  begin
                     SEQ[k] <- A[i];
                     lastj <- J[i];
                     k <- k-1;
                  end;
step5:      for k <- 1 to kmax do
               print SEQ[k];
         end;
```

* *This line is missing in the original and was added during retyping. (Otherwise i is undefined and the subsequent code mis-indented.)*

Correctness Proof and Time Analysis

Theorem 8:

Algorithm A correctly prints a maximal common subsequence of its inputs.

Proof: The key observation is that $T[k] = T_{i-1,k}$ at the start of the ith iteration of the loop of step 3, and $T[k] = T_{i,k}$ at the end of the ith iteration of this same loop. This fact follows from an easy induction on i, using Theorem 5 to relate the value of $T_{i,k}$ to $T_{i-1,k}$ and Lemma 3 to show that exactly one of the T's changes value per iteration. (Note that for all k, $T_{i-1,k}$ is either "∞" or is equal to j(i') for some i' < i. Thus j(i) can never equal $T_{i-1,k}$ for any k).

Given the above behavior of the T's, Theorem 6 then implies that $L[i] = L_i$ for $1 \le i \le n$ at the end of step 3.

In order to print a maximal common subsequence, we observe that if $L_i > 1$ then there exists an i' < i such that $L_{i'} = L_i - 1$ and j(i') < j(i). Thus a maximal common subsequence of A[1:i] and B[1:j(i)] may be formed by concatenating A[i] onto the end of a maximal common subsequence of A[1:i'] and B[1:j(i')]. This is exactly what step 4 does.

Theorem 9:

Algorithm A requires at most O(n log n) units of time to process two input sequences of length n.

Proof: Step 1, the computation of the j function, can be performed by sorting the two input sequences (remembering the original position of each element) and then merging the sequences together. This clearly can be done in O(n log n) steps.

Step 2, the initialization of the T array, takes O(n) steps.

Step 3, the computation of the L array, involves 2n simple assignments plus n searches of the T array. Each such search can be accomplished in time O(log n) using the well-known binary search technique. Thus step 3 requires at most O(n log n) time.

Step 4, the actual determination of a maximal common subsequence, clearly requires O(n) steps as does step 5, the printing of the subsequence.

A similar O(n log n) algorithm for Problem 2 of our introduction has been independently discovered by A. C. Yao and F. F. Yao [5].

Corollary 10:

Algorithm A requires at most O(n log log n) units of time to process two sequences which are permutations of the first n integers.

Proof:

Step 1 can be performed in time O(n) using a distribution sort.

In Step 3, each search of the T array can be performed in time O(log log n) time using the techniques of van Emde Boas[4]. These techniques allow us to efficiently perform certain manipulations on subsets of the first n integers. More specifically, the following operations are available: insert an integer into a set, delete an integer from a set, and determine the greatest integer in a set which is smaller than some specified search argument. Moreover, each of these operations may be performed in time O(log log n). These three operations are exactly what we need to perform Step 3.

Acknowledgement

The author is indebted to Harold Stone, who first suggested this variant of the general problem, and to Jeffrey Ullman for several enlightening conversations.

References

[1] Chvatal, V., D. A. Klarner, and D. E. Knuth, "Selected Combinatorial Research Problems", STAN-CS-72-292, Stanford University, Stanford, Ca. (June 1972) .

[2] Hirschberg, D. S., "A Linear Space Algorithm for Computing Maximal Common Subsequences", to appear in Communications of the Association for Computing Machinary (May, 1975).

[3] Wagner, R. A., and M. J. Fischer, "The String-to-String Correction Problem", Journal of the Association for Computing Machinary 12:1 (January 1974) pp. 168-173.

[4] van Emde Boas, P., "An O(n log log n) On-line Algorithm for the Insert-Extract Min Problem", TR 74-221, Department of Computer Science, Cornell University, Ithaca, New York, (Dec. 1974).

[5] Yao, A. C., and F. F. Yao, "On Computing the Rank Function for a Set of Vectors", UIUCDCS-R-75-699, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois (Feb. 1975).