Palo Alto Research Center

Efficient Dynamic Programming Using Quadrangle Inequalities

By F. Frances Yao

XEROX

Efficient Dynamic Programming Using Quadrangle Inequalities

F. Frances Yao

CSL-80-4   MARCH 1980

Abstract: See next page.

A version of this paper will appear in *Proceedings of the 1980 ACM Symposium on Theory of Computing*, April 1980.

CR Categories: 5.3, 8.3.

Key words and phrases: Binary search tree, convex polygon, dynamic programming, quadrangle inequality, triangle inequality.

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

ABSTRACT

Dynamic programming is one of several widely used problem-solving techniques in computer science and operation research. In applying this technique, one always seeks to find speed-up by taking advantage of special properties of the problem at hand. However, in the current state of art, ad hoc approaches for speeding up seem to be characteristic; few general criteria are known. In this paper we give a *quadrangle inequality* condition for rendering speed-up. This condition is easily checked, and can be applied to several apparently different problems. For example, it follows immediately from our general condition that the construction of optimal binary search trees may be speeded up from $O(n^3)$ steps to $O(n^2)$, a result that was first obtained by Knuth using a different and more complicated argument.

1. INTRODUCTION.

In the application of a general technique, it is often possible to improve the solution by taking advantage of special properties of the problem at hand. *Dynamic programming* is one of several widely used problem-solving techniques in computer science and operation research (see, e.g.[2]). It finds applications in context-free language parsing [8], constructing optimal binary trees [7], finding shortest paths [4], and in solving various "intractible" combinatorial problems (see the references in [2]). In the construction of optimal binary search trees, for example, Knuth[5][7] showed that an $O(n^2)$ algorithm may be obtained by improving upon the straightforward dynamic programming solution which demanded time $O(n^3)$. Knuth's proof is quite complicated and involves detailed properties of the optimal binary trees. In general, ad hoc approaches for speeding up seem to be characteristic in dynamic programming; few general criteria are known.

In the present paper we will discuss a *quadrangle inequality* condition for the purpose of achieving speed-up in dynamic programming. This condition is easily checked and will be applied to several apparently different problems. In particular, it is used to give a simple proof of Knuth's construction of optimal trees, and applied to optimization problems involving multiway partitions.

2. DYNAMIC PROGRAMMING AND QUADRANGLE INEQUALITIES.

We consider a simple dynamic programming problem for the purpose of illustration.

Example 1. Let $L_1, L_2, ..., L_n$ be n finite, nonempty sets of strings. We wish to compute their *product (concatenation)* $L_1 \cdot L_2 \cdot \cdot L_n$ by using $L \cdot L'$, the product of two sets, as the primitive. To simplify matters, we assume that the product operation is charged a cost of $|L| \cdot |L'|$, and results in $|L| \cdot |L'|$ strings stored in $L \cdot L'$ (i.e., duplicate strings will not be detected).

Let $|L_i| = n_i$ and $w(i,j) = n_i n_{i+1} ... n_j$, then the optimal cost $c(i,j)$ for computing $L_i \cdot L_{i+1} \cdot \cdot L_j$ satisfies the following recurrence relations:

$$c(i,i) = 0;$$
$$c(i,j) = w(i,j) + \min_{i<k\leq j}(c(i,k-1) + c(k,j)) \qquad \textit{for} \quad i<j. \tag{1}$$

We will refer to the function w in the above relations as the *increment function* for c; it determines the cost function c completely. To evaluate c using the obvious procedure suggested by these equations will require total time $O(n^3)$. However, as we will see, the increment function w in *Example 1* satisfies the *quadrangle inequalities (QI)*

$$w(i,j) + w(i',j') \leq w(i',j) + w(i,j') \qquad \textit{for} \quad i \leq i' \leq j \leq j'. \tag{2}$$

This property allows the dynamic programming to be speeded up because of the following general theorem.

Theorem 1. If the increment function w satisfies *QI* and furthermore is monotone on the lattice of intervals (ordered by inclusion), i.e.,

$$w(i,j) \leq w(i',j') \qquad \textit{if} \quad [i,j] \subseteq [i',j'],$$

then the function c defined by (1) can be computed in time $O(n^2)$.*

We now verify these conditions for the w in *Example 1*. The monotonicity is obvious. For the QI, let $a = n_i \cdots n_{i'-1}, b = n_{i'} \cdots n_j$, and $c = n_{j+1} \cdots n_{j'}$. Then the QI becomes

$$ab + bc \leq b + abc.$$

This is true since

$$0 \leq b(a-1)(c-1).$$

Theorem 1 is proved by establishing the following two lemmas.

Lemma 2.1. If w satisfies *QI* and is monotone on the lattice of intervals, then the function c defined by (1) also satisfies *QI*.

Proof. The proof is by induction on the length $l = |j' - i|$ of the "long side" of the quadrangle inequality

$$c(i,j) + c(i',j') \leq c(i',j) + c(i,j') \qquad \textit{for} \quad i \leq i' \leq j \leq j'. \tag{3}$$

*We assume that $w(i,j)$ is given; in all our examples, $w(i,j)$ is computable in $O(n^2)$ time from the input arguments of the problem.

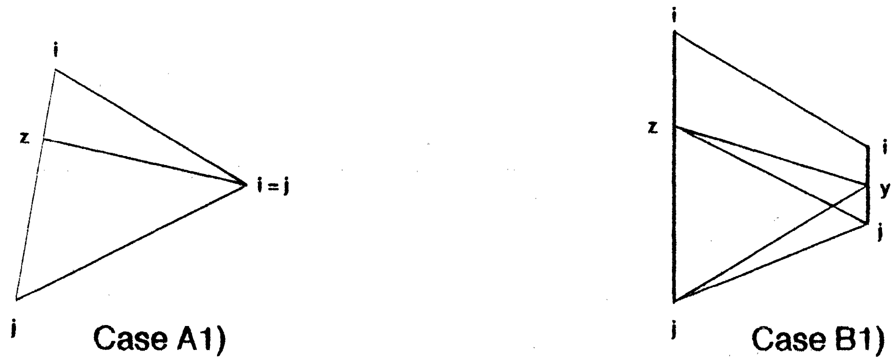

Figure 1. Proof of Lemma 2.1

First note that (3) is trivial when $i = i'$ or $j = j'$. Therefore (3) is true when $l \leq 1$. Inductively, consider two cases: A)$i < i' = j < j'$, and B)$i < i' < j < j'$. (See Figure 1).

Case A). $i < i' = j < j'$.

In this case, (3) becomes the (inverse) triangle inequality:

$$c(i,j) + c(j,j') \leq c(i,j') \qquad \text{for} \quad i < j < j'. \tag{4}$$

Suppose $c(i,j')$ is minimized at $k = z$; that is, $c(i,j') = c_z(i,j')$ where we use $c_k(i,j)$ to denote $w(i,j) + c(i,k-1) + c(k,j)$. There are two symmetric subcases.

Case A1). $z \leq j$.

We have $c(i,j) \leq c_z(i,j) = w(i,j) + c(i,z-1) + c(z,j)$. Therefore,

$$\begin{aligned} c(i,j) + c(j,j') &\leq w(i,j) + c(i,z-1) + c(z,j) + c(j,j') \\ &\leq w(i,j') + c(i,z-1) + c(z,j') \\ &= c(i,j'), \end{aligned}$$

where we used the monotonicity of w, and the induction hypothesis (4) at $z \leq j \leq j'$.

Case A2). $z \geq j$. This is symmetric with *A1)*, with all the intervals reversed.

Case B). $i < i' < j < j'$.

Assume the two terms on the right hand side of (3) achieve their values at $k = y$ and $k = z$ respectively. That is,

$$c(i', j) = c_y(i', j), \quad \text{and} \quad c(i, j') = c_z(i, j').$$

We again look at two symmetric subcases.

Case B1). $z \leq y$.

We have

$$c(i', j') \leq c_y(i', j')$$

and

$$c(i, j) \leq c_z(i, j).$$

Adding them up, we obtain

$$\begin{aligned} c(i, j) + c(i', j') &\leq c_z(i, j) + c_y(i', j') \\ &= w(i, j) + w(i', j') + c(i, z-1) + c(z, j) + c(i', y-1) + c(y, j') \end{aligned} \tag{5}$$

Applying the QI of w, and the induction hypothesis (3) at the points $z \leq y < j < j'$, Equ(5) becomes

$$\begin{aligned} c(i, j) + c(i', j') &\leq w(i', j) + w(i, j') + c(i, z-1) + c(i', y-1) + c(y, j) + c(z, j') \\ &= c_y(i', j) + c_z(i, j') \\ &= c(i', j) + c(i, j') \end{aligned}$$

Case B2). $z \geq y$. This is again symmetric with *B1)* .∎

Let us use $K_c(i, j)$ to denote $max\{k | c_k(i, j) = c(i, j)\}$; so $K_c(i, j)$ is the largest index k where the minimum is achieved in (1). (We define $K_c(i, i) = i$.)

Lemma 2.2. If the function c defined in (1) satisfies QI, then we have

$$K_c(i, j) \leq K_c(i, j+1) \leq K_c(i+1, j+1) \qquad for \quad i \leq j. \tag{6}$$

Proof. It is trivially true when $i = j$, therefore assume $i < j$. To prove the first inequality $K_c(i, j) \leq K_c(i, j+1)$, we show that for $i < k \leq k' \leq j$,

$$[c_{k'}(i, j) \leq c_k(i, j)] \Rightarrow [c_{k'}(i, j+1) \leq c_k(i, j+1)]. \tag{7}$$

Take the quadrangle inequality of c at $k \leq k' \leq j < j+1$

$$c(k,j) + c(k',j+1) \leq c(k',j) + c(k,j+1).$$

Adding $w(i,j) + w(i,j+1) + c(i,k-1) + c(i,k'-1)$ to both sides, we get

$$c_k(i,j) + c_{k'}(i,j+1) \leq c_{k'}(i,j) + c_k(i,j+1),$$

from which (7) follows. Similarly, the second inequality $K_c(i,j+1) \leq K_c(i+1,j+1)$ follows from the QI of c at $i < i+1 \leq k \leq k'$. ∎

Lemma 2.2 says that the matrix $K_c(i,j)$ is nondecreasing along each row and column. As a consequence, when we compute $c(i,j)$ for $\delta = j - i = 0, 1, 2, ..., n-1$, only $K_c(i+1, j+1) - K_c(i,j)$ minimization operations need to be carried out for $c(i,j+1)$. Hence for a fixed δ, the total amount of work is $O(n)$ since

$$\sum_{\substack{j-i=\delta-1 \\ 1 \leq i,j \leq n}} (K_c(i+1,j+1) - K_c(i,j)) \leq K_c(n-\delta+1,n) - K_c(1,\delta)$$
$$\leq n.$$

The overall computation time is therefore $O(n^2)$. This proves *Theorem 1.* ∎

We remark that the monotonicity assumption on w in *Lemma 2.1* is necessary for the QI of c. For example, if we let $(i,i',j,j') = (1,2,2,3)$, then the QI of c becomes

$$c(1,2) + c(2,3) \leq c(1,3),$$

which is equivalent to

$$w(1,2) + w(2,3) \leq w(1,3) + \min(w(1,2), w(2,3)),$$

or

$$\max(w(1,2), w(2,3)) \leq w(1,3).$$

3. OPTIMAL BINARY SEARCH TREES.

The construction of optimal binary search trees is a well known example of dynamic programming. The statement of the problem is as follows[5][7].

Example 2. We are given $2n+1$ probabilities $p_1, p_2, \cdots, p_n$ and $q_0, q_1, \cdots, q_n$ where

$$p_i = \text{probability that Key}_i \text{ is the search argument;}$$
$$q_i = \text{probability that the search argument lies between Key}_i \text{ and Key}_{i+1}.$$

We wish to find a binary tree which minimizes the expected number of comparisons in the search, namely,

$$\sum_{1\le j\le n} p_j(1 + \text{level of jth internal node in symmetric order}) + \sum_{0\le k\le n} q_k(\text{level of the (k+1)st external node})$$

where the root has level zero.

Let $c(i,j)$ be the cost of an optimal subtree with weights $(p_{i+1}, \cdots, p_j; q_i, \cdots, q_j)$. Since all subtrees of an optimal tree are optimal, it follows that $c(i,j)$ satisfies the same recurrences as given by Equ.(1) with w now defined by

$$w(i,j) = p_{i+1} + \cdots + p_j + q_i + \cdots + q_j. \tag{8}$$

This increment function is monotone, and it satisfies the quadrangle inequalities in fact as equalities. It therefore follows from *Theorem 1* that we can have an $O(n^2)$ time construction of an optimal tree by dynamic programming. In [5], the monotone property (6) is derived by a more complex argument.

Note that the question asked in Knuth [7, Section 6.2.2 ex.30] is whether the cost function c satisfies a special case of the quadrangle inequalities, namely

$$c(i,j) + c(i+1,j+1) \le c(i+1,j) + c(i,j+1), \tag{9}$$

and is therefore answered in the affirmative by *Lemma 2.1.* In fact, (9) is equivalent to the general *QI* since (3) can be derived from (9) by induction on $|i'-i|$ and $|j'-j|$.

4. MAXIMIZATION PROBLEMS IN A CONVEX POLYGON .

We look at an example where the quadrangle inequalities have a most intuitive interpretation, and where binary partitions generalize easily to multiway partitions.

Example 3. Suppose $v_1 v_2 \cdots v_n$ is a convex polygon in E^2. Let $d(i,j) =$ the Euclidean distance between v_i and v_j if $i \leq j$, and $d(i,j) = 0$ if $i > j$. We notice that d satisfies the *inverse quadrangle inequalities*, i.e.,

$$d(i,j) + d(i',j') \geq d(i',j) + d(i,j') \qquad \text{for} \quad i \leq i' \leq j \leq j'. \tag{10}$$

(Inverse *QI's* are what we need in considering maximization problems such as the present one.) We use $A \otimes B$ to denote the $(max, +)$ — *multiplication* of upper triangular matrices A and B. That is, if $A = (a(i,j))$ and $B = (b(i,j))$, then $A \otimes B = (c(i,j))$ where $c(i,j) = \max_{i \leq k \leq j}(a(i,k) + b(k,j))$. We define $D^{(1)} = D = (d(i,j))$, $D^{(t)} = D^{(t-1)} \otimes D$, and write $D^{(t)}$ as $(d^{(t)}(i,j))$. For example, $d^{(2)}(i,j)$ is the length of the longest trajectory from v_i to v_j that allows one bounce off the wall $v_i v_{i+l} \cdots v_j$. We are interested in efficiently computing $D^{(t)}$, and thereby finding a perimeter maximizing t-gon inscribed in the given convex polygon.

By associativity $D^{(t)} = D^{(r)} \otimes D^{(s)}$ for $t = r + s$. This multiplication is a special case of a relation of the following form.

$$c(i,j) = w(i,j) + \max_{i \leq k \leq j}(a(i,k) + b(k,j)) \qquad \text{for} \quad i \leq j. \tag{11}$$

It follows from *Lemma 4.1* below that $d^{(r)}(i,j)$ satisfies the inverse *QI* for any $r \geq 1$ by induction on r. *Lemma 4.2* then tells us that the multiplication $D^{(r)} \otimes D^{(s)}$ can be done in $O(n^2)$ time for any $r \geq 1$ and $s \geq 1$.

Lemma 4.1. If w, a and b all satisfy the inverse *QI*, then the function c defined by (11) also satisfies the inverse *QI*.

Proof. Similar to the proof of *Lemma 2.1*, except that we need not consider *Case A)* seperately from *Case B)*.

Lemma 4.2. If both a and b satisfy the inverse *QI*, then for the function c defined by (11) we have

$$K_c(i,j) \leq K_c(i,j+1) \leq K_c(i+1,j+1) \qquad \text{for} \quad i \leq j.$$

Proof. Similar to the proof of *Lemma 2.2*.

Theorem 2. For any $t > 1$, $D^{(t)}$ can be computed in time $O((\log t)n^2)$.

Proof. Apply a standard binary algorithm for computing powers. (Also cf. proof of *Theorem 3*).∎

Corollary. For any $t > 1$, we can find a perimeter maximizing $(t+1)$-gon inscribed in the given convex polygon in time $O((\log t)n^2)$.

Proof. It is easy to see that the largest entry in the matrix $D^{(t)} + D$ gives the maximum perimeter that we want.∎

Example 3 is reminiscent of the problem studied in [3], where monotonicity properties similar to *Lemma 4.2* are utilized to find an *area* maximizing triangle inscribed in a convex polygon efficiently.

5. OPTIMAL MULTIWAY TREES .

In view of *Theorem 2*, what can we say when Equ(1) is generalized to allow $c(i,j)$ to be partitioned into up to t subproblems? The recurrence becomes

$$\begin{aligned} c(i,j) &= w(i,j) + \min_{i<k_1\le k_2\le\cdots\le k_{t-1}\le j}(c(i,k_1-1)+c(k_1,k_2-1)+\cdots+c(k_{t-1},j)) \\ & \qquad\qquad\qquad\qquad\qquad\qquad \text{if} \quad i<j, \\ c(i,j) &= 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{if} \quad i\ge j. \end{aligned} \tag{12}$$

So when $i < j$, the problem of computing $c(i,j)$ is divided into 2 to t subproblems whose sizes are strictly smaller than that of the original problem. (We say a subproblem $c(k,l)$ is *empty* if $k > l$.) This problem is similar to *Example 3*; it requires a little more care since it involves recurrences.

The main result we have here is the following Theorem. We say that a function w satisfies the *triangle inequalities (TI)* if

$$w(i,j) + w(j,j') \le w(i,j') \qquad \textit{for} \quad i<j<j'.$$

In this section, we shall assume that $w(i,j)$ is nonnegative; then w satisfies *TI* implies that w is monotone.

Theorem 3. If the increment function w satisfies *QI* and *TI*, then the function c defined by (12) can be computed in time $O((\log t)n^2)$.

Example 4. Consider the construction of optimal search trees as in *Example 2*, but allowing each node to have degree at most t. In the special case that all the q's are zero, we have $w(i,j) = p_{i+1} + \cdots + p_j$, which satisfies the condition of *Theorem 3*, and such optimal t-way trees can be constructed in time $O((\log t)n^2)$.

Let us denote the 'min' term in Equ(12) by $f^{(t)}(i,j)$. Thus, (12) can be rewritten as

$$\begin{aligned} c(i,j) &= w(i,j) + f^{(t)}(i,j) && \text{if } i < j, \\ c(i,j) &= 0 && \text{if } i \geq j, \end{aligned} \tag{12a}$$

where

$$\begin{aligned} f^{(t)}(i,j) &= \min_{i<k_1 \leq k_2 \leq \cdots \leq k_{t-1} \leq j} (c(i,k_1-1) + c(k_1,k_2-1) + \cdots + c(k_{t-1},j)) \\ & \qquad\qquad\qquad\qquad \text{if } i < j, \\ f^{(t)}(i,j) &= 0 \qquad\qquad\qquad\qquad \text{if } i \geq j. \end{aligned} \tag{13}$$

Furthermore, define $f^{(1)}(i,j) = c(i,j)$; and for $2 \leq q \leq t-1$, define $f^{(q)}(i,j)$ to be the optimal sum of $\leq q$ subproblems:

$$\begin{aligned} f^{(q)}(i,j) &= \min_{i \leq k_1 \leq k_2 \leq \cdots \leq k_{q-1} \leq j} (c(i,k_1-1) + c(k_1,k_2-1) + \cdots + c(k_{q-1},j)) \\ & \qquad\qquad\qquad\qquad \text{if } i \leq j, \\ f^{(q)}(i,j) &= 0 \qquad\qquad\qquad\qquad \text{if } i \geq j. \end{aligned} \tag{14}$$

Note that, in a partition for $f^{(q)}(i,j)$, only one subproblem is required to be nonempty.

Fact A. $f^{(1)}(i,j) \geq f^{(2)}(i,j) \geq \cdots \geq f^{(t)}(i,j)$.

Proof. All except the last inequality follows immediately from the definition of $f^{(q)}$. If $f^{(t-1)}(i,j)$ is obtained by a decomposition into two or more subproblems, then we have $f^{(t-1)}(i,j) \geq f^{(t)}(i,j)$; otherwise $f^{(t-1)}(i,j) = c(i,j) \geq f^{(t)}(i,j)$ by Equ(12a) since $w(i,j)$ is nonnegative. ∎

Fact B.

$$f^{(q)}(i,j) = \min_{i \leq k \leq j} (f^{(r)}(i,k-1) + f^{(s)}(k,j)) \quad \text{for} \quad q = r+s, \quad 2 \leq q \leq t-1 \quad \text{and} \quad r \geq 1, s \geq 1. \tag{15}$$

Proof. We will show that lefthand side≥righthand side since the other direction is obvious. If in Equ(14) the minimum value of $f^{(q)}(i,j)$ is achieved with division points $k_1, \cdots, k_r, \cdots, k_{q-1}$, we can achieve the same value on the righthand side of (15) by choosing $k = k_r$. ∎

Fact C. $$f^{(t)}(i,j) = \min_{i<k\leq j}(f^{(r)}(i,k-1) + f^{(s)}(k,j)) \quad \textit{for} \quad t = r+s, \quad \textit{and} \quad r \geq 1, s \geq 1. \tag{16}$$

Proof. Similar to the proof of *Fact B.* Again choose k on the righthand side to be the k_r of the lefthand side.∎

Lemma 5.1. In (15), if $f^{(r)}(i,j)$ and $f^{(s)}(i,j)$ satisfy *QI* for $j-i \leq \delta$, then $f^{(q)}(i,j)$ satisfies *QI* for $j-i \leq \delta$.

Proof. Similar to the proof of *Lemma 2.1.*In the case corresponding to *Case A1),* we need the *TI*

$$f^{(s)}(z,j) + f^{(q)}(j,j') \leq f^{(s)}(z,j'),$$

which follows from the *QI* of $f^{(s)}$, and the fact $f^{(q)} \leq f^{(s)}$. Similarly, *Case A2)* follows from the *QI* of $f^{(r)}$, *Case B1)* from the *QI* of $f^{(s)}$, and *B2)* from the *QI* of $f^{(r)}$. (See Figure 2).∎

Lemma 5.2. In (15), if $f^{(r)}(i,j)$ and $f^{(s)}(i,j)$ satisfy *QI* for $j-i \leq \delta$, then $f^{(t)}(i,j)$ satifies *QI* for $j-i \leq \delta+1$.

Proof. Analogous to the proof of *Lemma 5.1*; here the problem sizes are strictly reduced in the inductive step. ∎

Lemma 5.3. In (12), if w satisfies *QI* and *TI*, then $f^{(1)}(=c), \cdots, f^{(q)}, \cdots, f^{(t)}$ all satisfy *QI*.

Proof. It follows from the preceding two Lemmas by induction on $\delta = j-i$ and induction on q. Note that the facts that w satisfies *QI, TI,* and $f^{(t)}$ satisfies *QI* together imply that c satisfies *QI.*∎

Proof of Theorem 3. For the f's on the lefthand side of (15) and (16), we use $K_f(i,j)$ as before to denote the largest k on the righthand side which allows the minimum value of $f(i,j)$ to be achieved. By the same argument which led to *Lemma 2.2 and 4.2,* we have

$$K_f(i,j) \leq K_f(i,j+1) \leq K_f(i+1,j+1) \quad \textit{for} \quad i \leq j, \quad \textit{and} \quad f = f^{(q)}, 1 \leq q \leq t. \tag{17}$$

Let $q_1, q_2, \cdots, q_h$ be an *addition chain*[6] for t; that is, $q_1 = 1$, $q_h = t$, and for each $i \geq 1$, $q_i = q_j + q_k$ for some $j < i$, $k < i$. It is well known that any $t > 1$ has an addition chain of length $h \leq 2\log t$. The following procedure then employs Equ(12a), (15) and (16) to compute $c(i,j)$.

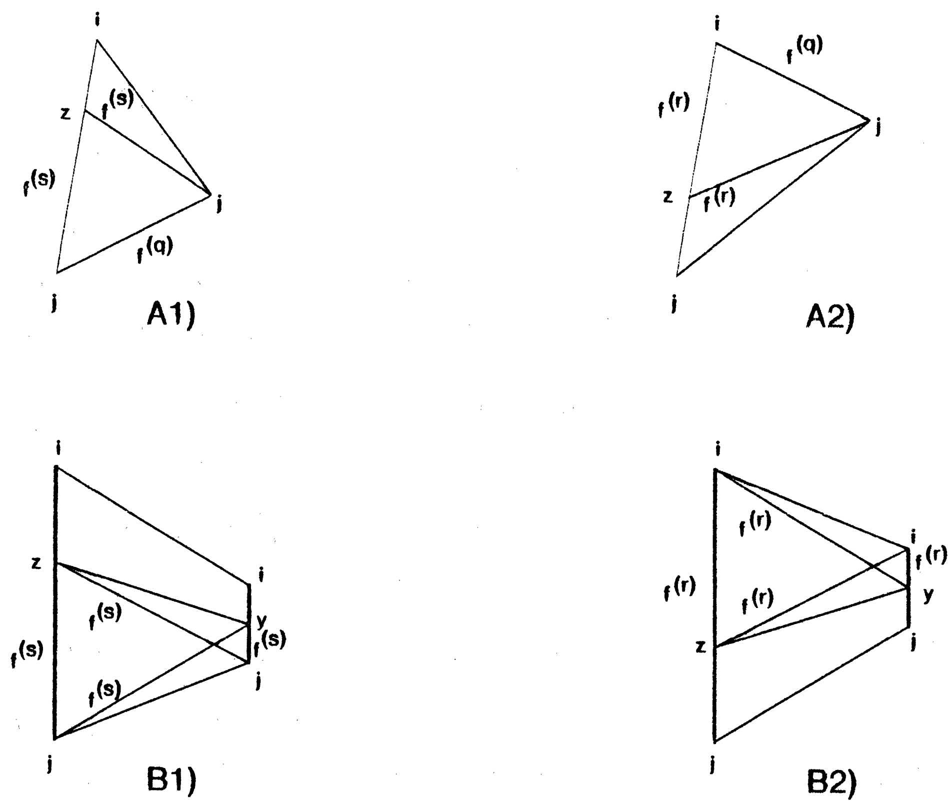

Figure 2. Proof of Lemma 5.1

begin

for $1 \le i \le n, 1 \le m \le h$ **do** $f^{(q_m)}(i,i) \leftarrow 0$;

  for $\delta \leftarrow 1$ **to** n **do**

    for $m \leftarrow 1$ **to** h **do**

      for $j - i = \delta$ **do**

        compute $f^{(q_m)}(i,j)$

end

Because of Equ(17), the innermost loop takes only $O(n)$ steps. Therefore the algorithm uses total time $O(hn^2)$, which is $O((\log t)n^2)$. ∎

6. CONCLUDING REMARKS .

In this paper we have considered a general type of conditions which ensures monotonicity of division points in certain dynamic programming processes. This monotonicity property makes it possible to achieve speed-up by a factor of n or more over the straightforward implementations. We would like to point out some situations where the present results do not apply, and which deserve further study.

The monotonicity property for the division points does not hold for the matrix multiplication chain problem[1], as shown by the following example. Consider the matrices M_1, M_2, M_3, M_4 with dimensions $2 \times 3, 3 \times 2, 2 \times 10$, and 10×1, respectively. As can be easily verified, the proper order to compute $M_1M_2M_3$ is to parenthesize it as $(M_1M_2)M_3$, while the optimal computation of $M_1M_2M_3M_4$ corresponds to $M_1(M_2(M_3M_4))$.

Similarly, optimal t-way search trees in general (when the q's are not zero) do not satisfy the monotonicity property either.The addition of a new leftmost key may force the division points (at the root) to shift rightward! Such an example for ternary trees is shown in Figure 3. As the increment function w defined by (8) fails to satisfy *TI* (for example, $w(1,2) + w(2,3) \ge w(1,3)$), the function c defined by (12) does not satisfy the *QI* (Figure 3a). When Key_1 is added, the division points may change from $\{3,4\}$ to $\{4,5\}$ if the weights are properly chosen.(Figure 3b).

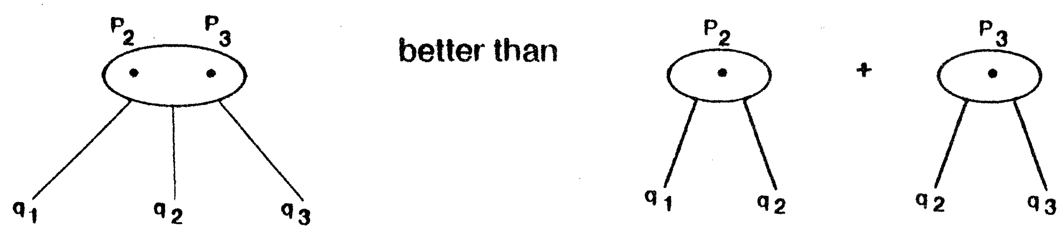

Figure 3a. $c(1,3) + c(2,2) \leq c(1,2) + c(2,3)$.

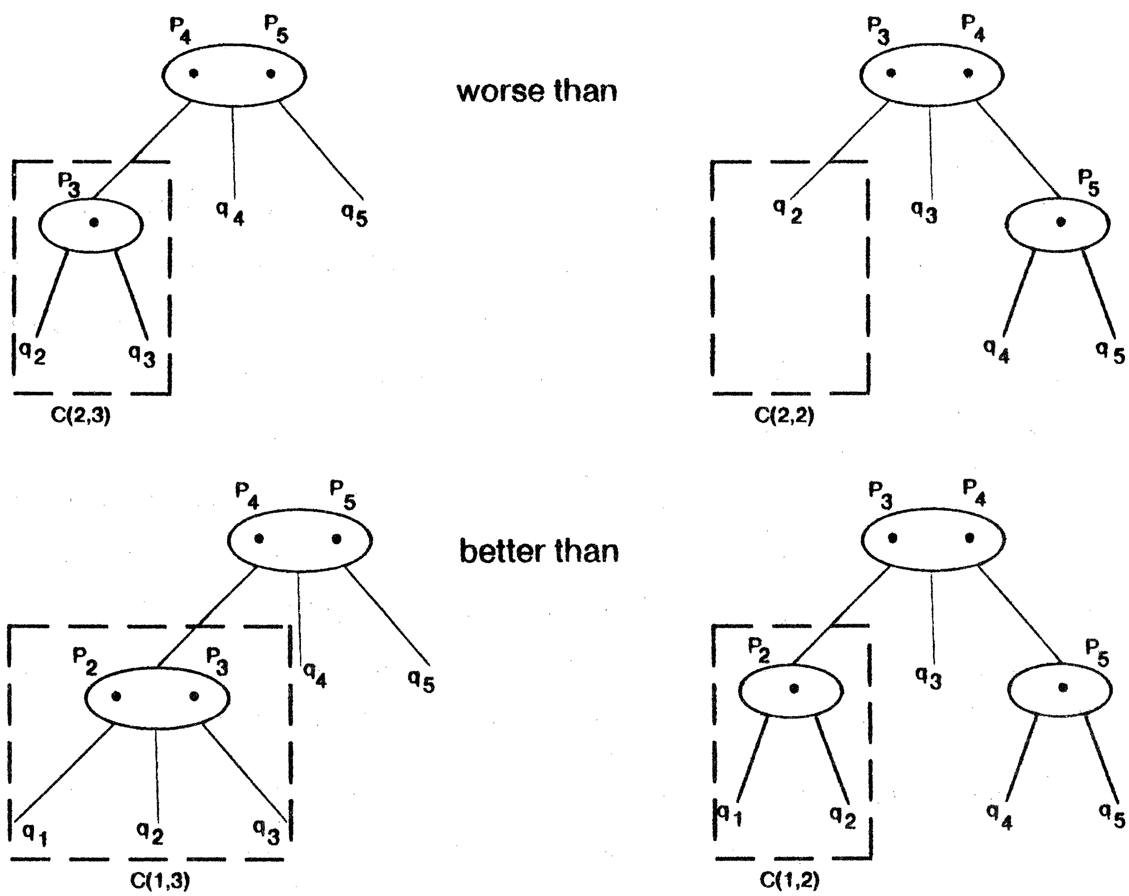

Figure 3b. For the w defined by (8), Equ(17) fails.

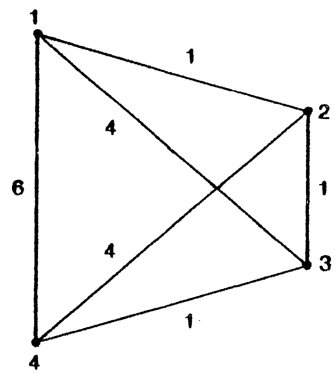

Figure 4. A function $w(i,j)$ for $1 \leq i \leq j \leq 4$ (with $w(i,i) = 0$), which satisfies *TI* but not *QI*.

Finally, we note a few properties of the quadrangle inequalities.

1. *QI* and *TI* are incomparable: the w given by (8) is an example of a function which satisfies *QI* but not *TI*;.the example in Figure 4 is a function that satisfies *TI* but not *QI*.

2. It is easy to show that if $g: R \to R$ is a concave, nondecreasing function, then $g \circ w$ satisfies *QI* if w does. Therefore the conclusions of *Examples 1* and *2* still hold if we substitute $w(i,j)^2$ for $w(i,j)$. Similarly, a convex, nondecreasing function g preserves the *inverse QI*; thus in *Exmaple 3* we may use, say, $\log d(i,j)$ as the distance between v_i and v_j.

3. In *Lemmas 2.2 and 4.2*, a stronger monotone property than that stated in (6) is actually implied by the *QI*. If we define $R(i,j)$ to be the *set* of optimal division points, that is, $R(i,j) = \{k \mid c_k(i,j) = c(i,j)\}$, then the entire set $R(i,j)$ shifts right as either i or j increases. For example, one might have $R(i,j) = \{1,3,4,7\}, R(i,j+1) = \{4,7,9\}$ and $R(i+1,j+1) = \{9,11\}$. (cf. [7, Section 6.2.2 ex.27]).

We recently learned from Don Knuth that Zhu Yongjin and Wang Jianfang [9], in studying algorithms for constructing alphabetic trees with restricted depth, used an approach similar to ours.

REFERENCES

[1] A. Aho, J. Hopcroft and J. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley, Reading Mass., 1974.

[2] K. Q. Brown, *Dynamic programming in computer science*, Computer Science Department Report CMU-CS-79-106, Carnegie-Mellon University, February 1979.

[3] D. P. Dobkin and L. Snyder, *On a general method for maximizing and minimizing among certain geometric problems*, Proc. IEEE 20th Annual Symposium on Foundations of Computer Science, Puerto Rico, 1979, 9-17.

[4] R. W. Floyd, *Algorithm 97 : shortest path*, *Comm. ACM* 5 (1962), 345.

[5] D. E. Knuth, *Optimum binary search trees*, *Acta Informatica* 1 (1971), 14-25.

[6] D. E. Knuth, *The Art of Computer Programming, Vol 2 :Seminumerical Algorithms*, Addison-Wesley, Reading Mass., 1975.

[7] D. E. Knuth, *The Art of Computer Programming, Vol 3 :Searching and Sorting*, Addison-Wesley, Reading Mass., 1973.

[8] D. H. Younger, *Recognition of context-free languages in time n^3*, *Information and Control* 10 (1967), 189-208.

[9] Y. Zhu and J. Wang, *On alphabetic-extended binary trees with restricted path length*, *Scientia Sinica* 22 (1979), 1362-1371.

XEROX

XEROX

Efficient Dynamic Programming Using Quadrangle Inequalities By F. Frances Yao

CSL-80-4

Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California 94304

Printed in U.S.A.