element of a candidate subsequence of length $i-1$. Maintain candidate subsequences by linking them through the input sequence.)

16.4 Optimal polygon triangulation

In this section, we investigate the problem of optimally triangulating a convex polygon. Despite its outward appearance, we shall see that this geometric problem has a strong similarity to matrix-chain multiplication.

A ***polygon*** is a piecewise-linear, closed curve in the plane. That is, it is a curve ending on itself that is formed by a sequence of straight-line segments, called the ***sides*** of the polygon. A point joining two consecutive sides is called a ***vertex*** of the polygon. If the polygon is ***simple***, as we shall generally assume, it does not cross itself. The set of points in the plane enclosed by a simple polygon forms the ***interior*** of the polygon, the set of points on the polygon itself forms its ***boundary***, and the set of points surrounding the polygon forms its ***exterior***. A simple polygon is ***convex*** if, given any two points on its boundary or in its interior, all points on the line segment drawn between them are contained in the polygon's boundary or interior.

We can represent a convex polygon by listing its vertices in counterclockwise order. That is, if $P = \langle v_0, v_1, \ldots, v_{n-1} \rangle$ is a convex polygon, it has n sides $\overline{v_0 v_1}, \overline{v_1 v_2}, \ldots, \overline{v_{n-1} v_n}$, where we interpret v_n as v_0. (In general, we shall implicitly assume arithmetic on vertex indices is taken modulo the number of vertices.)

Given two nonadjacent vertices v_i and v_j, the segment $\overline{v_i v_j}$ is a ***chord*** of the polygon. A chord $\overline{v_i v_j}$ divides the polygon into two polygons: $\langle v_i, v_{i+1}, \ldots, v_j \rangle$ and $\langle v_j, v_{j+1}, \ldots, v_i \rangle$. A ***triangulation*** of a polygon is a set T of chords of the polygon that divide the polygon into disjoint ***triangles*** (polygons with 3 sides). Figure 16.4 shows two ways of triangulating a 7-sided polygon. In a triangulation, no chords intersect (except at endpoints) and the set T of chords is maximal: every chord not in T intersects some chord in T. The sides of triangles produced by the triangulation are either chords in the triangulation or sides of the polygon. Every triangulation of an n-vertex convex polygon has $n-3$ chords and divides the polygon into $n-2$ triangles.

In the ***optimal (polygon) triangulation problem***, we are given a convex polygon $P = \langle v_0, v_1, \ldots, v_{n-1} \rangle$ and a weight function w defined on triangles formed by sides and chords of P. The problem is to find a triangulation that minimizes the sum of the weights of the triangles in the triangulation. One weight function on triangles that comes to mind naturally is

$$w(\triangle v_i v_j v_k) = |v_i v_j| + |v_j v_k| + |v_k v_i| \ ,$$

where $|v_i v_j|$ is the euclidean distance from v_i to v_j. The algorithm we shall develop works for an arbitrary choice of weight function.

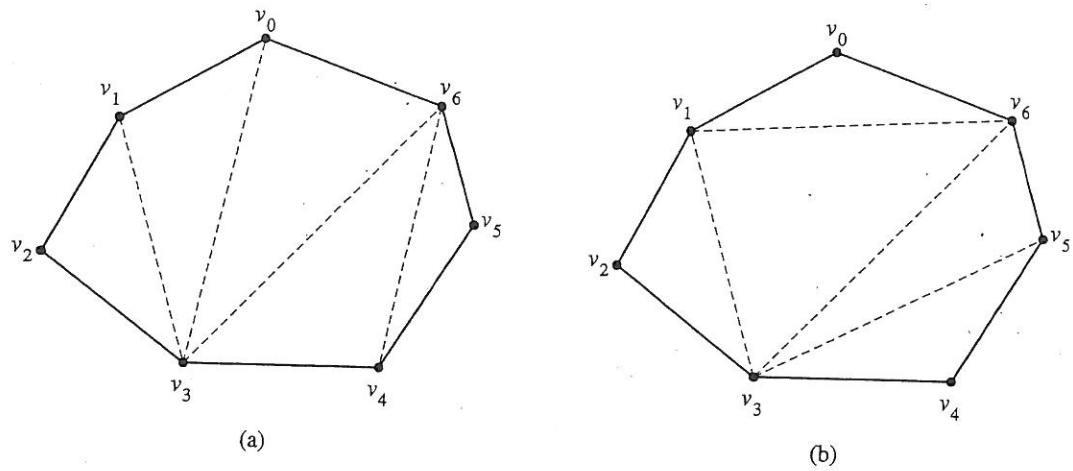

Figure 16.4 Two ways of triangulating a convex polygon. Every triangulation of this 7-sided polygon has $7 - 3 = 4$ chords and divides the polygon into $7 - 2 = 5$ triangles.

Correspondence to parenthesization

There is a surprising correspondence between the triangulation of a polygon and the parenthesization of an expression such as a matrix-chain product. This correspondence is best explained using trees.

A full parenthesization of an expression corresponds to a full binary tree, sometimes called the ***parse tree*** of the expression. Figure 16.5(a) shows a parse tree for the parenthesized matrix-chain product

$$((A_1(A_2A_3))(A_4(A_5A_6))) \ . \tag{16.6}$$

Each leaf of a parse tree is labeled by one of the atomic elements (matrices) in the expression. If the root of a subtree of the parse tree has a left subtree representing an expression E_l and a right subtree representing an expression E_r, then the subtree itself represents the expression (E_lE_r). There is a one-to-one correspondence between parse trees and fully parenthesized expressions on n atomic elements.

A triangulation of a convex polygon $\langle v_0, v_1, \ldots, v_{n-1}\rangle$ can also be represented by a parse tree. Figure 16.5(b) shows the parse tree for the triangulation of the polygon from Figure 16.4(a). The internal nodes of the parse tree are the chords of the triangulation plus the side $\overline{v_0v_6}$, which is the root. The leaves are the other sides of the polygon. The root $\overline{v_0v_6}$ is one side of the triangle $\triangle v_0v_3v_6$. This triangle determines the children of the root: one is the chord $\overline{v_0v_3}$, and the other is the chord $\overline{v_3v_6}$. Notice that this triangle divides the original polygon into three parts: the triangle $\triangle v_0v_3v_6$ itself, the polygon $\langle v_0, v_1, \ldots, v_3\rangle$, and the polygon $\langle v_3, v_4, \ldots, v_6\rangle$. Moreover, the two subpolygons are formed entirely by sides of the original polygon, except for their roots, which are the chords $\overline{v_0v_3}$ and $\overline{v_3v_6}$.

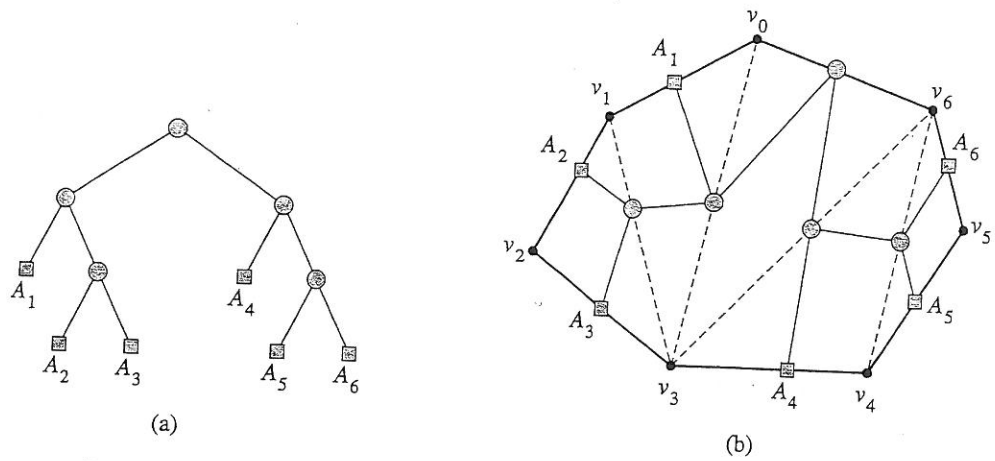

Figure 16.5 Parse trees. **(a)** The parse tree for the parenthesized product $((A_1(A_2A_3))(A_4(A_5A_6)))$ and for the triangulation of the 7-sided polygon from Figure 16.4(a). **(b)** The triangulation of the polygon with the parse tree overlaid. Each matrix A_i corresponds to the side $\overline{v_{i-1}v_i}$ for $i = 1, 2, \ldots, 6$.

In recursive fashion, the polygon $\langle v_0, v_1, \ldots, v_3 \rangle$ contains the left subtree of the root of the parse tree, and the polygon $\langle v_3, v_4, \ldots, v_6 \rangle$ contains the right subtree.

In general, therefore, a triangulation of an n-sided polygon corresponds to a parse tree with $n - 1$ leaves. By an inverse process, one can produce a triangulation from a given parse tree. There is a one-to-one correspondence between parse trees and triangulations.

Since a fully parenthesized product of n matrices corresponds to a parse tree with n leaves, it therefore also corresponds to a triangulation of an $(n + 1)$-vertex polygon. Figures 16.5(a) and (b) illustrate this correspondence. Each matrix A_i in a product $A_1A_2 \cdots A_n$ corresponds to a side $\overline{v_{i-1}v_i}$ of an $(n + 1)$-vertex polygon. A chord $\overline{v_iv_j}$, where $i < j$, corresponds to a matrix $A_{i+1..j}$ computed during the evaluation of the product.

In fact, the matrix-chain multiplication problem is a special case of the optimal triangulation problem. That is, every instance of matrix-chain multiplication can be cast as an optimal triangulation problem. Given a matrix-chain product $A_1A_2 \cdots A_n$, we define an $(n + 1)$-vertex convex polygon $P = \langle v_0, v_1, \ldots, v_n \rangle$. If matrix A_i has dimensions $p_{i-1} \times p_i$ for $i = 1, 2, \ldots, n$, we define the weight function for the triangulation as

$$w(\triangle v_i v_j v_k) = p_i p_j p_k \ .$$

An optimal triangulation of P with respect to this weight function gives the parse tree for an optimal parenthesization of $A_1A_2 \cdots A_n$.

Although the reverse is not true—the optimal triangulation problem is *not* a special case of the matrix-chain multiplication problem—it turns out that our code MATRIX-CHAIN-ORDER from Section 16.1, with minor

modifications, solves the optimal triangulation problem on an $(n+1)$-vertex polygon. We simply replace the sequence $\langle p_0, p_1, \ldots, p_n \rangle$ of matrix dimensions with the sequence $\langle v_0, v_1, \ldots, v_n \rangle$ of vertices, change references to p to references to v, and change line 9 to read:

9 **do** $q \leftarrow m[i,k] + m[k+1,j] + w(\triangle v_{i-1} v_k v_j)$

After running the algorithm, the element $m[1,n]$ contains the weight of an optimal triangulation. Let us see why this is so.

The substructure of an optimal triangulation

Consider an optimal triangulation T of an $(n+1)$-vertex polygon $P = \langle v_0, v_1, \ldots, v_n \rangle$ that includes the triangle $\triangle v_0 v_k v_n$ for some k, where $1 \le k \le n-1$. The weight of T is just the sum of the weights of $\triangle v_0 v_k v_n$ and triangles in the triangulation of the two subpolygons $\langle v_0, v_1, \ldots, v_k \rangle$ and $\langle v_k, v_{k+1}, \ldots, v_n \rangle$. The triangulations of the subpolygons determined by T must therefore be optimal, since a lesser-weight triangulation of either subpolygon would contradict the minimality of the weight of T.

A recursive solution

Just as we defined $m[i,j]$ to be the minimum cost of computing the matrix-chain subproduct $A_i A_{i+1} \cdots A_j$, let us define $t[i,j]$, for $1 \le i < j \le n$, to be the weight of an optimal triangulation of the polygon $\langle v_{i-1}, v_i, \ldots, v_j \rangle$. For convenience, we consider a degenerate polygon $\langle v_{i-1}, v_i \rangle$ to have weight 0. The weight of an optimal triangulation of polygon P is given by $t[1,n]$.

Our next step is to define $t[i,j]$ recursively. The basis is the degenerate case of a 2-vertex polygon: $t[i,i] = 0$ for $i = 1, 2, \ldots, n$. When $j - i \ge 1$, we have a polygon $\langle v_{i-1}, v_i, \ldots, v_j \rangle$ with at least 3 vertices. We wish to minimize over all vertices v_k, for $k = i, i+1, \ldots, j-1$, the weight of $\triangle v_{i-1} v_k v_j$ plus the weights of the optimal triangulations of the polygons $\langle v_{i-1}, v_i, \ldots, v_k \rangle$ and $\langle v_k, v_{k+1}, \ldots, v_j \rangle$. The recursive formulation is thus

$$t[i,j] = \begin{cases} 0 & \text{if } i = j\,, \\ \min\limits_{i \le k \le j-1} \{t[i,k] + t[k+1,j] + w(\triangle v_{i-1} v_k v_j)\} & \text{if } i < j\,. \end{cases} \qquad (16.7)$$

Compare this recurrence with the recurrence (16.2) that we developed for the minimum number $m[i,j]$ of scalar multiplications needed to compute $A_i A_{i+1} \cdots A_j$. Except for the weight function, the recurrences are identical, and thus, with the minor changes to the code mentioned above, the procedure MATRIX-CHAIN-ORDER can compute the weight of an optimal triangulation. Like MATRIX-CHAIN-ORDER, the optimal triangulation procedure runs in time $\Theta(n^3)$ and uses $\Theta(n^2)$ space.