Announcements

- Homework #7 is due next Monday, Dec. 10, 2005
- Final in SN014, 4:00-7:00pm, on Thursday, Dec. 15, 2005
- Lin's Office Hours Final Week:

  3:00pm - 4:00pm on Dec. 14 and Dec. 15

Steps in DP: Step 1

- Think what decision is the "last piece in the puzzle"
  - Where to place the outermost parentheses in a matrix chain multiplication

$$(A_1)\ (A_2\ A_3\ A_4)$$
$$(A_1\ A_2)\ (A_3\ A_4)$$
$$(A_1\ A_2\ A_3)\ (A_4)$$

DP Step 2

- Ask what subproblem(s) would have to be solved to figure out how good your choice is
  - How to multiply the two groups of matrices, e.g., this one (A_1) (trivial) and this one ($A_2 A_3 A_4$)

DP Step 3

- Write down a formula for the "goodness" of the best choice

$$m[i, j] = \min_{i \le k < j} (m[i, k] + m[k+1, j] + p_{i-1} p_k p_j)$$

DP Step 4

- Arrange subproblems in order from small to large and solve each one, keeping track of the solutions for use when needed
- Need 2 tables
  - One tells you value of the solution to each subproblem
  - Other tells you last option you chose for the solution to each subproblem

Matrix-Chain-Order(p)

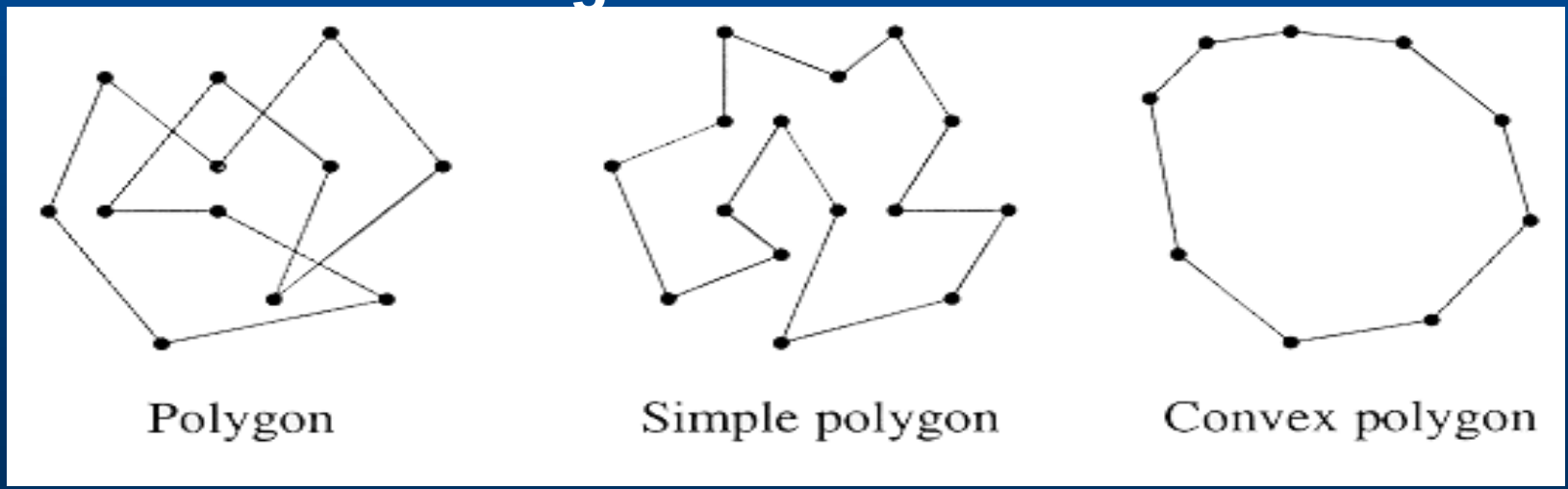
1. $n \leftarrow length[p] - 1$
2. for $i \leftarrow 1$ to n // initialization: $O(n)$ time
3. do $m[i, i] \leftarrow 0$
4. for $L \leftarrow 2$ to n // L = length of sub-chain
5. do for $i \leftarrow 1$ to $n - L+1$
6. do $j \leftarrow i + L - 1$
7. $m[i, j] \leftarrow \infty$
8. for $k \leftarrow i$ to $j - 1$
9. do $q \leftarrow m[i, k] + m[k+1, j] + p_{i-1} p_k p_j$
10. if $q < m[i, j]$
11. then $m[i, j] \leftarrow q$
12. $s[i, j] \leftarrow k$
13. return m and s

Polygons

- A *polygon* is a piecewise linear closed curve in the plane. We form a cycle by joining line segments end to end. The line segments are called the *sides* of the polygon and the endpoints are called the *vertices*.
- A polygon is *simple* if it does not cross itself, i.e. if the edges do not intersect one another except for two consecutive edges sharing a common vertex. A simple polygon defines a region consisting of points it encloses. The points strictly within this region are in the *interior* of this region, the points strictly on the outside are in its *exterior*, and the polygon itself is the *boundary* of this region.

Convex Polygons

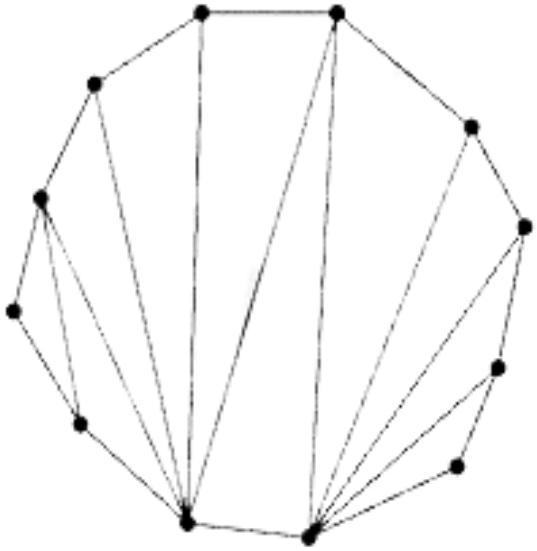
- A simple polygon is said to be convex if given any two points on its boundary, the line segment between them lies entirely in the union of the polygon and its interior.
- Convexity can also be defined by the interior angles. The interior angles of vertices of a convex polygon are at most 180 degrees.



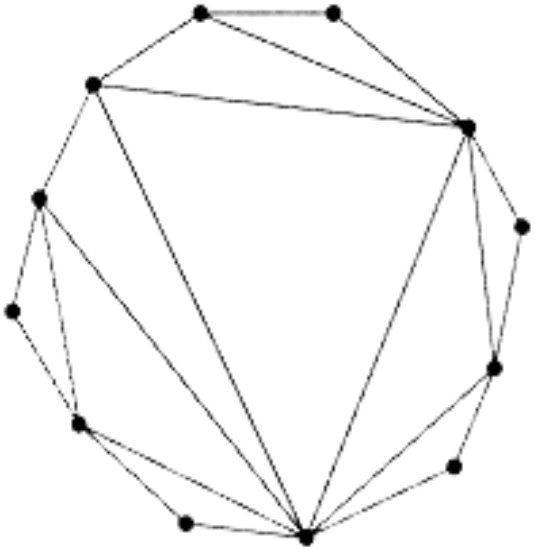
Triangulations

- Given a convex polygon, assume that its vertices are labeled in counterclockwise order $P=\langle v_0,\ldots,v_{n-1}\rangle$. Assume that indexing of vertices is done modulo n, so $v_0 = v_n$. This polygon has n sides, (v_{i-1},v_i).
- Given two nonadjacent v_j, where $i < j$, the line segment (v_i,v_j) is a ***chord***. (If the polygon is simple but not convex, a segment must also lie entirely in the interior of P for it to be a chord.) Any chord subdivides the polygon into two polygons.
- A ***triangulation*** of a convex polygon is a maximal set T of chords. Every chord that is not in T intersects the interior of some chord in T. Such a set of chords subdivides interior of a polygon into set of triangles.

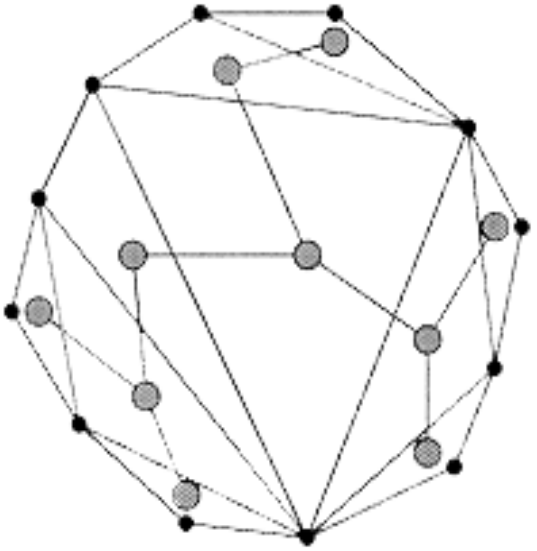
Example: Polygon Triangulation



A triangulation



Another triangulation



The dual tree

Dual graph **of the triangulation is a graph whose vertices are the triangles, and in which two vertices share an edge if the triangles share a common chord. NOTE: the dual graph is a free tree. In general, there are many possible triangulations.**

Minimum-Weight Convex Polygon Triangulation

- The number of possible triangulations is exponential in n, the number of sides. The "best" triangulation depends on the applications.
- **Our problem:** Given a convex polygon, determine the triangulation that minimizes the sum of the perimeters of its triangles.
- Given three distinct vertices, v_i, v_j and v_k, we define the ***weight*** of the associated triangle by the weight function

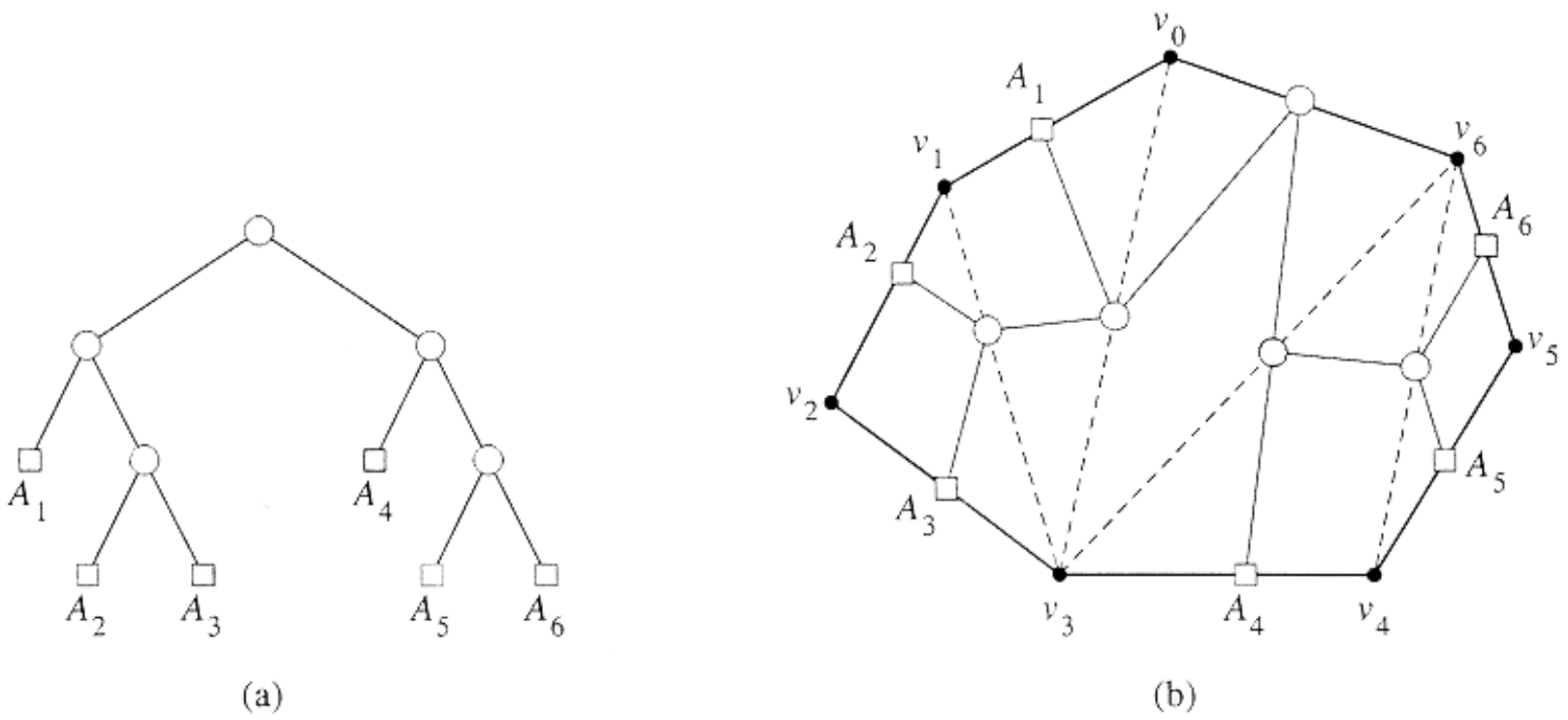
$$w(v_i, v_j, v_k) = |v_i v_j| + |v_j v_k| + |v_k v_i|,$$

where $|v_i v_j|$ denotes length of the line segment (v_i, v_j).

Correspondence to Binary Trees

- In MCM, the associated binary tree is the evaluation tree for the multiplication, where the leaves of the tree correspond to the matrices, and each node of the tree is associated with a product of a sequence of two or more matrices.
- Consider an (n+1)-sided convex polygon, $P=<v_0,\ldots,v_n>$ and fix one side of the polygon, (v_0 ,v_n). Consider a rooted binary tree whose root node is the triangle containing side (v_0 ,v_n), whose internal nodes are the nodes of the dual tree, and whose leaves correspond to the remaining sides of the tree. The partitioning of a polygon into triangles is equivalent to a binary tree with n-1 leaves, and vice versa.

Binary Tree for Triangulation



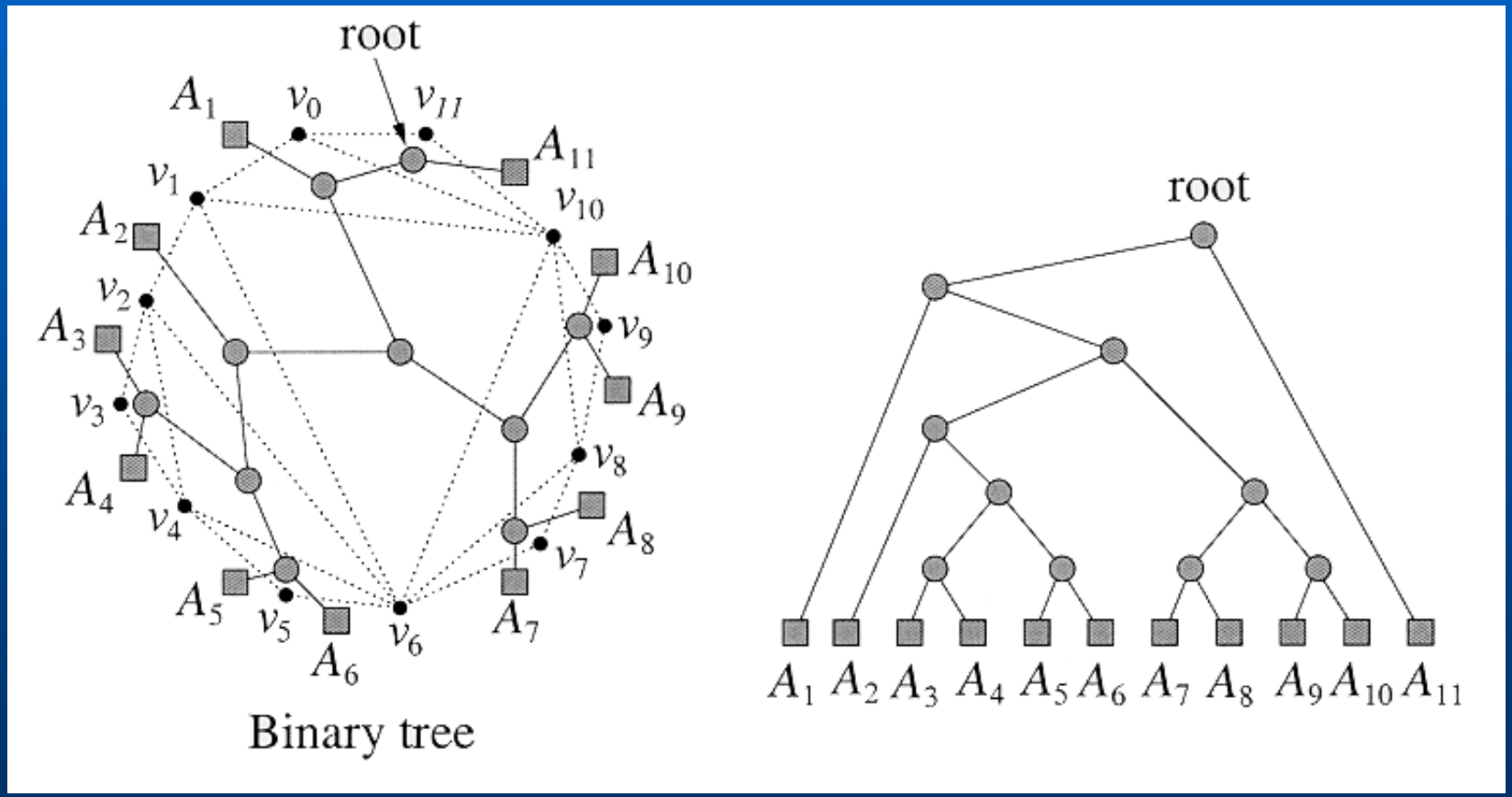
- The associated binary tree has *n* leaves, and hence *n-1* internal nodes. Since each internal node other than the root has one edge entering it, there are *n-2* edges between the internal nodes.

Lemma

- A triangulation of a simple polygon has *n-2* triangles and *n-3* chords.

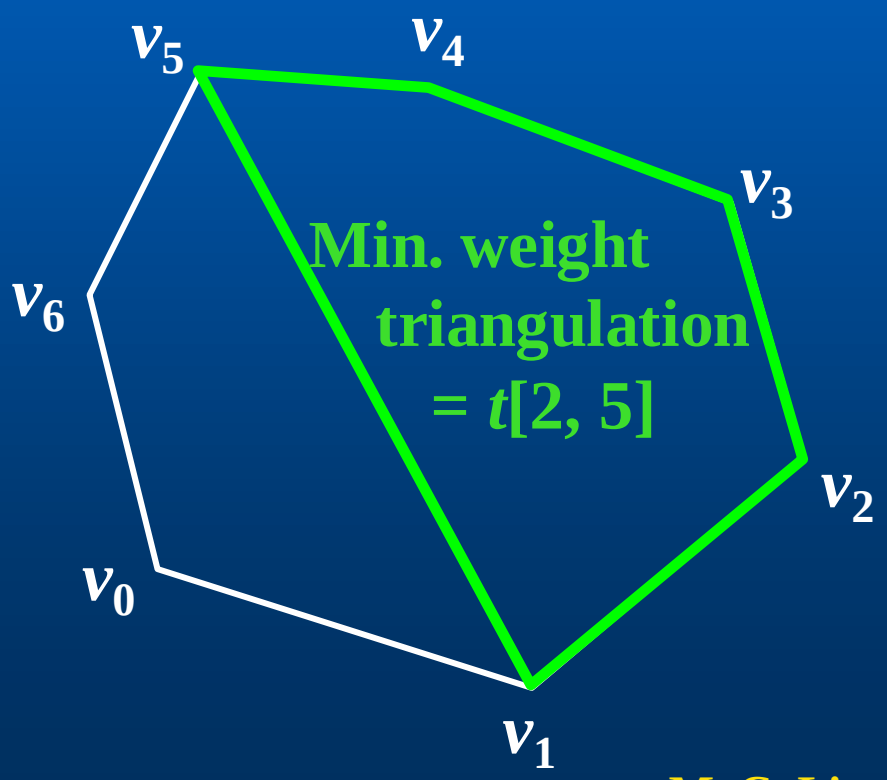
(Proof) The result follows directly from the previous figure. Each internal node corresponds to one triangle and each edge between internal nodes corresponds to one chord of triangulation. If we consider an *n*-vertex polygon, then we'll have *n*-1 leaves, and thus *n*-2 internal nodes (triangles) and *n*-3 edges (chords).

Another Example of Binary Tree for Triangulation



DP Solution (I)

- For $1 \le i \le j \le n$, let $t[i, j]$ denote the minimum weight triangulation for the subpolygon $\langle v_{i-1}, v_i, \ldots, v_j \rangle$.
- We start with v_{i-1} rather than v_i, to keep the structure as similar as possible to the matrix chain multiplication problem.



DP Solution (II)

- Observe: if we can compute $t[i, j]$ for all i and j ($1 \leq i \leq j \leq n$), then the weight of minimum weight triangulation of the entire polygon will be $t[1, n]$.

- For the basis case, the weight of the trivial 2-sided polygon is zero, implying that $t[i, i] = 0$ (line (v_{i-1}, v_i)).

DP Solution (III)

- In general, to compute $t[i, j]$, consider the subpolygon $\langle v_{i-1}, v_i, \ldots, v_j \rangle$, where $i \leq j$. One of the chords of this polygon is the side (v_{i-1}, v_j). We may split this subpolygon by introducting a triangle whose base is this chord, and whose third vertex is any vertex v_k, where $i \leq k \leq j-1$. This subdivides the polygon into 2 subpolygons $\langle v_{i-1}, \ldots v_k \rangle$ & $\langle v_{k+1}, \ldots v_j \rangle$, whose minimum weights are $t[i, k]$ and $t[k+1, j]$.

- We have following recursive rule for computing $t[i, j]$:

  $t[i, i] = 0$

  $t[i, j] = \min_{i \leq k \leq j-1} (t[i, k] + t[k+1, j] + w(v_{i-1} v_k v_j))$ for $i < k$

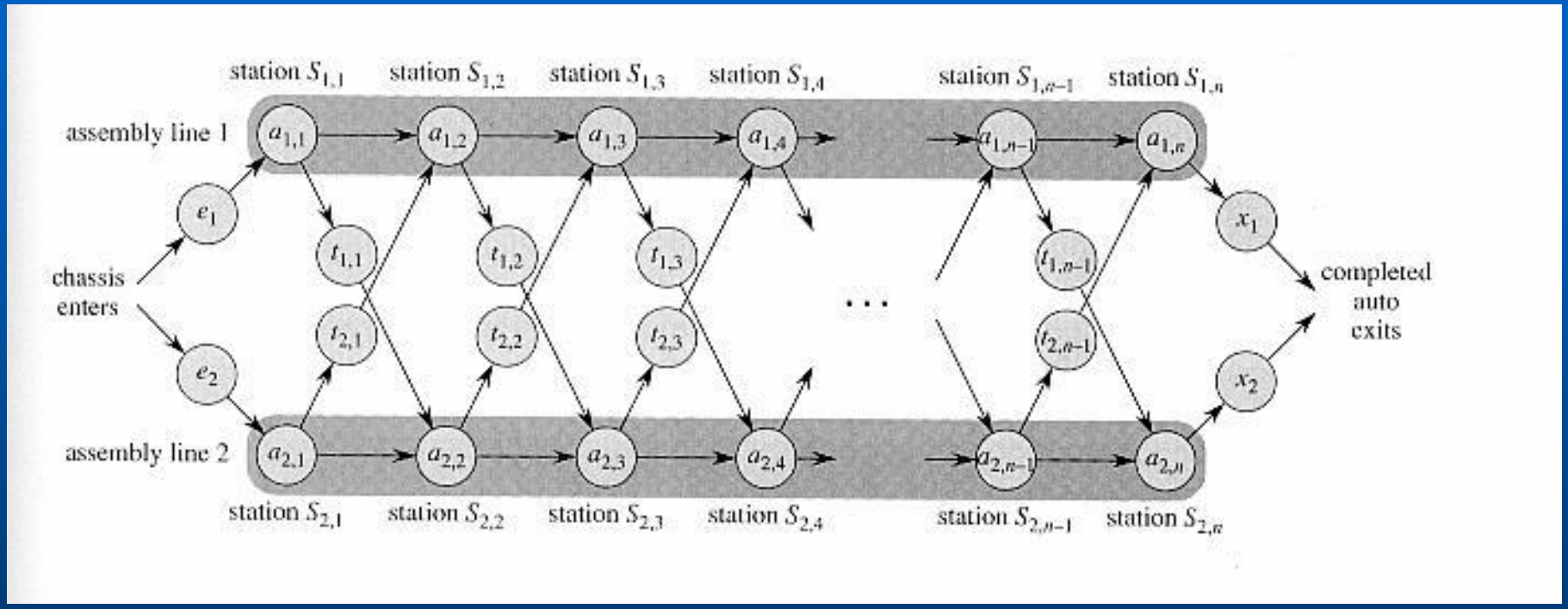
Weighted-Polygon-Triangulation(V)

```
1.  n ← length[V] - 1                    // V = <v_0, v_1, ..., v_n>
2.  for i ← 1 to n                       // initialization: O(n) time
3.       do t[i, i] ← 0
4.    for L ← 2 to n                     // L = length of sub-chain
5.       do for i ← 1 to n-L+1
6.            do j ← i + L - 1
7.               t[i, j] ← ∞
8.                  for k ← i to j - 1
9.                        do q ← t[i, k] + t[k+1, j] + w(v_{i-1}, v_k, v_j)
10.                          if q < t[i, j]
11.                          then t[i, j] ← q
12.                               s[i, j] ← k
13.  return t and s
```

Assembly-Line Scheduling

- Two parallel assembly lines in a factory, lines 1 and 2
- Each line has n stations $S_{i,1} \ldots S_{i,n}$
- For each j, $S_{1,j}$ does the same thing as $S_{2,j}$, but it may take a different amount of assembly time $a_{i,j}$
- Transferring away from line i after stage j costs $t_{i,j}$
- Also entry time e_i and exit time x_i at beginning and end

Assembly Lines



Finding Subproblem

- Pick some convenient stage of the process
  - Say, just before the last station
- What's the next decision to make?
  - Whether the last station should be $S_{1,n}$ or $S_{2,n}$
- What do you need to know to decide which option is better?
  - What the fastest times are for $S_{1,n}$ & $S_{2,n}$

Recursive Formula for Subproblem

Fastest time to any given station $= \min($ Fastest time through prev station (same line), Fastest time through prev station (other line) $+$ Time it takes to switch lines $)$

Recursive Formula (II)

- Let $f_i[\,j]$ denote the fastest possible time to get the chassis through $S_{i,j}$
- Have the following formulas:

$$f_1[\,1] = e_1 + a_{1,1}$$

$$f_1[\,j] = \min(\, f_1[\,j\text{-}1] + a_{1,j},\ \ f_2[\,j\text{-}1] + t_{2,j\text{-}1} + a_{1,j}\,)$$

- Total time:

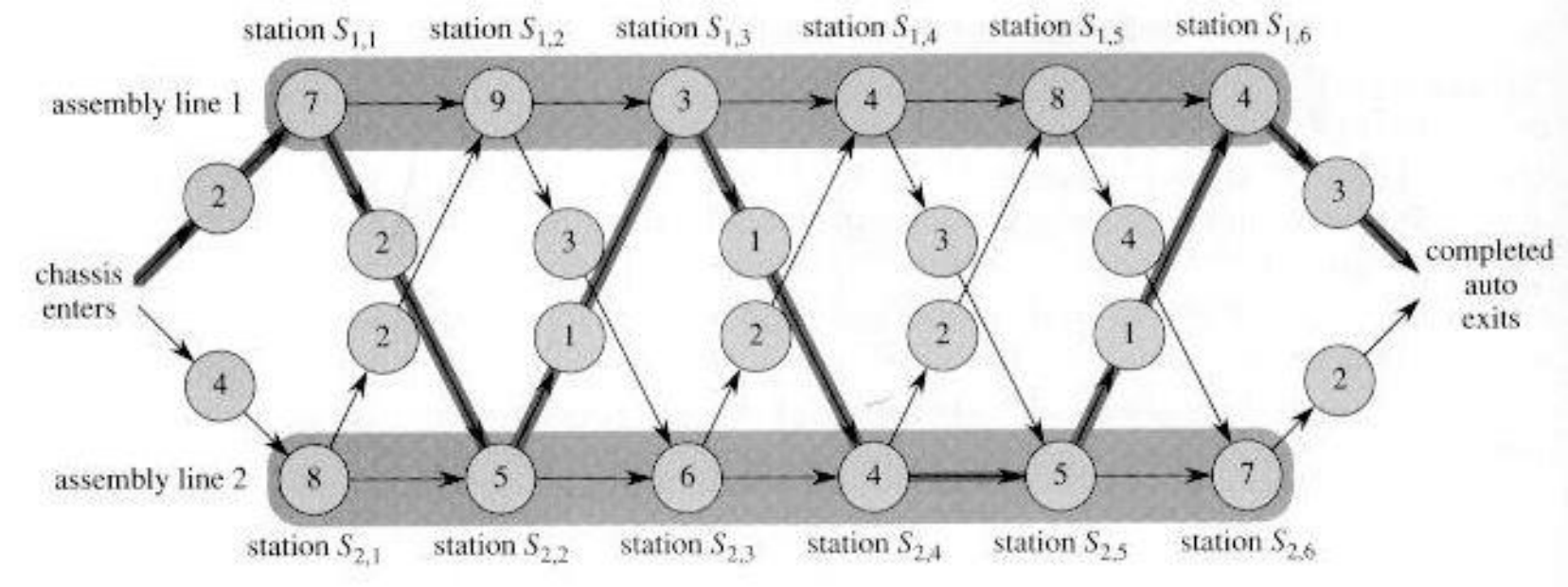
$$f^* = \min(\, f_1[n] + x_1,\ \ f_2[\,n] + x_2)$$

```
FASTEST-WAY(a, t, e, x, n)
 1  f1[1] ← e1 + a1,1
 2  f2[1] ← e2 + a2,1
 3  for j ← 2 to n
 4       do if f1[j − 1] + a1,j ≤ f2[j − 1] + t2,j−1 + a1,j
 5            then f1[j] ← f1[j − 1] + a1,j
 6                 l1[j] ← 1
 7            else f1[j] ← f2[j − 1] + t2,j−1 + a1,j
 8                 l1[j] ← 2
 9          if f2[j − 1] + a2,j ≤ f1[j − 1] + t1,j−1 + a2,j
10            then f2[j] ← f2[j − 1] + a2,j
11                 l2[j] ← 2
12            else f2[j] ← f1[j − 1] + t1,j−1 + a2,j
13                 l2[j] ← 1
14  if f1[n] + x1 ≤ f2[n] + x2
15     then f* = f1[n] + x1
16          l* = 1
17     else f* = f2[n] + x2
18          l* = 2
```

Analysis

- Only loop is lines 3-13 which iterate n-1 times: Algorithm is $O(n)$.
- The array l records which line is used for each station number

Example


station $S_{1,1}$
station $S_{1,2}$
station $S_{1,3}$
station $S_{1,4}$
station $S_{1,5}$
station $S_{1,6}$
assembly line 1
chassis enters
assembly line 2
completed auto exits
station $S_{2,1}$
station $S_{2,2}$
station $S_{2,3}$
station $S_{2,4}$
station $S_{2,5}$
station $S_{2,6}$