Algorithms

Dana Shapira

Lesson #4: Dynamic programming

Fibonacci Series

- **F(0)=0**
- **F(1)=1**
- **F(n)=F(n-1)+F(n-2)**

- Write a Divide and Conquer Algorithm!
- What is its running time?

Binomial Coefficients

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

- Recursive equation:

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

- Write a Divide and Conquer Algorithm!
- What is its running time?

Dynamic Programming Approach to Optimization Problems

1. Characterize structure of an optimal solution.
2. Recursively define value of an optimal solution.
3. Compute value of an optimal solution in bottom-up fashion.
4. Construct an optimal solution from computed information.

World Series Odds

- Two teams A and B play to see who is the first to win *n* games. In world series games *n*=4.
- Assumption: A and B are equally competent, each has a 50% chance to win any particular game.
- `P(i,j)` - the probability that A needs *i* extra games to win and B needs *j* extra games to win.

World Series Odds

```
P(0,j)=1     j>0
P(i,0)=0     i>0
P(i,j)=½P(i-1,j)+½P(i,j-1)   i>0 and j>0
```


Recursive Solution

```
P(i,j){
  if i=0 return 1
  elseif j=0 return 0
  else return ½P(i-1,j)+½P(i,j-1)
}
```

- What is its running time?

Dynamic Programming

```
P(n,m){
  for(i=0; i≤n; i++)
        T[0,i] = 1
  for(j=1; j≤m; j++)
        T[j,0] = 0
  for(i=1; i≤n; i++)
     for(j=1; j≤m; j++)
        T[i,j]= ½T[i-1,j]+ ½T[i,j-1];
return T[n,m]
}
```

	0	1	2	3	4
0	1	1	1	1	1
1	0	1/2	3/4	7/8	15/16
2	0	1/4	1/2		
3	0	1/8			
4	0	1/16			

- What is its running time?

Matrix Chain Multiplication

-Problem:

Given n matrices $M_1, M_2, \ldots, M_n$, compute the product $M_1 M_2 M_3 \ldots M_n$, where M_i has dimension $d_{i-1} \times d_i$, for $i = 1, \ldots, n$.

-objective

compute $M_1, M_2, \ldots, M_n$ with the minimum number of scalar multiplications.

-Given matrices A with dimension p x q and B with dimension q x r, multiplication AB takes pqr scalar multiplications

Matrix Chain Multiplication

-Problem: Parenthesize the product $M_1 M_2 \ldots M_n$ in a way to minimize the number of scalar multiplications.

-Example: M_1 --- 2 x 5

M_2 --- 5 x 3

M_3 --- 3 x 4

$M_1(M_2M_3)$ --- 60 + 40 = 100 multiplications

$(M_1M_2)M_3$ --- 30 + 24 = 54 multiplications

Matrix Chain Multiplication

- Let m(i,j) be the number of multiplications performed using optimal parenthesis of $M_i M_{i+1} \ldots M_{j-1} M_j$.

$$m(i,j) = \begin{cases} 0 & i = j \\ \min\limits_{i \le k < j} \{m(i,k) + m(k+1,j) + d_{i-1} d_k d_j\} & i < j \end{cases}$$

Matrix Chain Multiplication

```
MUL(i,j){
  if (i= =j) return 0;
  else{
      x←∞
      for k=i to j-1
              y←MUL(i,k)+MUL(k+1,j)+d_{i-1}d_k d_j
              if y<x {
                          x←y
                          S(i,j)←k
              }
      }
}
```

- What is its running time?

Recursive Running Time

$$T(n)=\sum_{k=1}^{n-1}\left(T(k)+T(n-k)+1\right)$$
$$T(n)=2\sum_{k=1}^{n-1}T(k)+(n-1)$$
$$T(n-1)=2\sum_{k=1}^{n-2}T(k)+(n-2)$$
$$T(n)-T(n-1)=2T(n-1)+1$$
$$T(n)=3T(n-1)+1>3T(n-1)>3^2T(n-2)>\cdots>3^kT(n-k)$$

Dynamic Programming

-Example: M_1 --- 2 x 5

M_2 --- 5 x 3

M_3 --- 3 x 4

m	1	2	3
1	0	30	54
2		0	60
3			0

$$m(i,j)=\begin{cases}0 & i=j\\ \min\limits_{i\le k<j}\{m(i,k)+m(k+1,j)+d_{i-1}d_kd_j\} & i<j\end{cases}$$

S	1	2	3
1		1	2
2			2
3			

Matrix Chain Multiplication

```
MUL(n){
   for (i=1;i≤n;i++) m[i,i]=0;
   for (diff=1;diff≤n-1;diff++){
       for (i=1;i≤n-diff;i++){
             j←i+diff
             x←∞
             for (k=i;k≤j-1;k++)
                   y←m[i,k]+m[k+1,j]+d_{i-1}d_k d_j
                   if y<x {
                         x←y
                         S(i,j)←k
                   }
             m[i,j]=x;
   }
}
```


Longest Common Subsequence

- Let $x = x_1 \cdot x_2 \cdots x_n$

  $y = y_1 \cdot y_2 \cdots y_m$
- A common subsequence of X and Y of length k exists if there are indices $i_1 \le i_2 \le \ldots \le i_k$ and $j_1 \le j_2 \le \ldots \le j_k$ such that for every $1 \le \ell \le k$ $x_{i_\ell} = y_{j_\ell}$

- objective

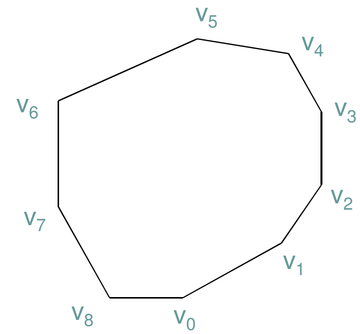
Find the longest Common Subsequence of x and y (LCS)

LCS

```
LCS(X,Y){
  for (i=1;i≤m;i++) C[i,0]=0;
  for (j=0;j≤n;i++) C[0,j]=0;
  for (i=1;i≤m;i++){
     for (j=1;j≤n;i++){
          if (x_i=y_j)
               C[i,j]←C[i-1,j-1]+1
          else C[i,j]← max(C[i,j-1],C[i-1,j])
  }
  return C[m,n]
}
```

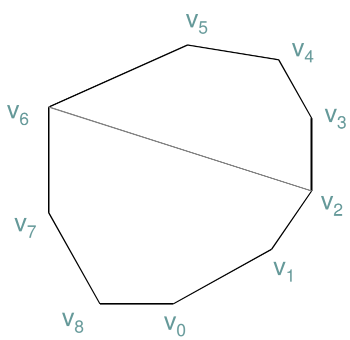

Optimal Polygon Triangulation

Given a convex polygon P=< $v_0,v_1,\ldots,v_{n-1}$ >, a chord v_iv_j divides P into two polygons <$v_i,v_{i+1},\ldots,v_j$> and <$v_j,v_{j+1},\ldots,v_i$> (assume $v_n=v_0$ or more generally, $v_k=v_{k \bmod n}$).



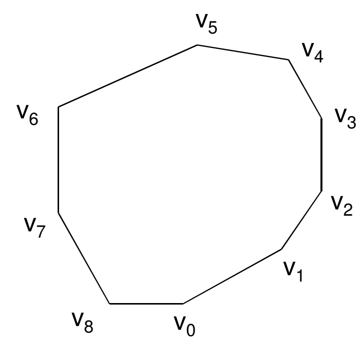
Optimal Polygon Triangulation

Given a convex polygon P=< $v_0,v_1,\ldots,v_{n-1}$ >, a chord v_iv_j divides P into two polygons <$v_i,v_{i+1},\ldots,v_j$> and <$v_j,v_{j+1},\ldots,v_i$> (assume $v_n=v_0$ or more generally, $v_k=v_{k \bmod n}$).



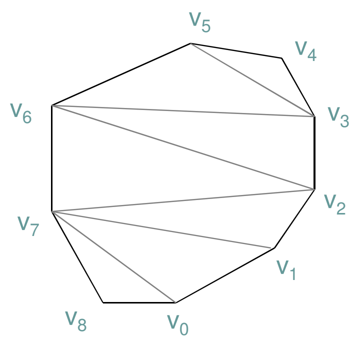
Example 3. Optimal Polygon Triangulation

Given a convex polygon P=< $v_0,v_1,\ldots,v_{n-1}$ >, it can always be divided into n-2 non-overlapping triangles using n-3 chords.



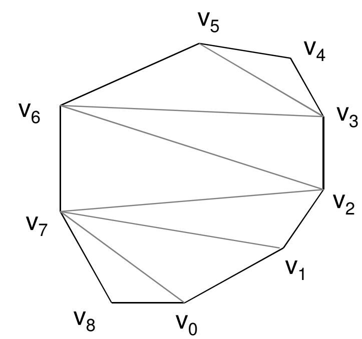
Example 3. Optimal Polygon Triangulation

Given a convex polygon P=< $v_0, v_1, \ldots, v_{n-1}$ >, it can always be divided into n-2 non-overlapping triangles using n-3 chords.



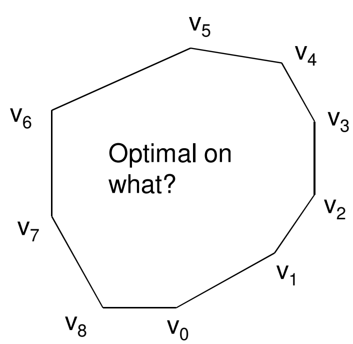
Example 3. Optimal Polygon Triangulation

Given a convex polygon P=< $v_0, v_1, \ldots, v_{n-1}$ >, there could be a lot of triangulations (in fact, an exponential number of them).



Example 3. Optimal Polygon Triangulation

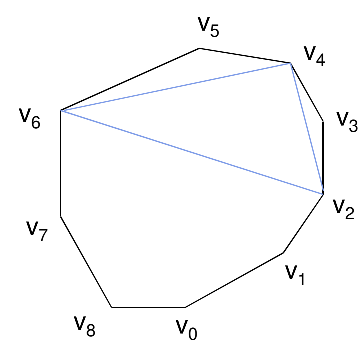
Given a convex polygon P=< $v_0, v_1, \ldots, v_{n-1}$ >, we want to compute an optimal triangulation.



Example 3. Optimal Polygon Triangulation

Given a convex polygon P=< $v_0, v_1, \ldots, v_{n-1}$ >, we want to compute an optimal triangulation whose weight is minimized.

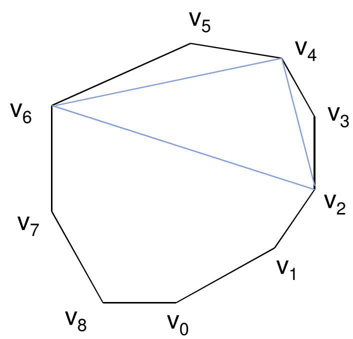
The weight of a triangulation is the weight of all its triangles and the weight of a triangle is the sum of its 3 edge lengths.



Example 3. Optimal Polygon Triangulation

Given a convex polygon P=< $v_0, v_1, \ldots, v_{n-1}$ >, we want to compute an optimal triangulation whose weight is minimized.

The weight of a triangulation is the weight of all its triangles and the weight of a triangle is the sum of its 3 edge lengths.

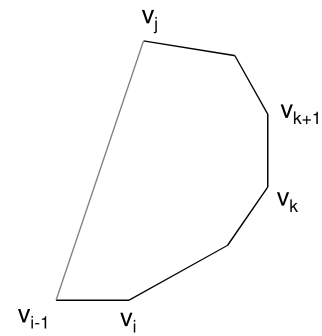


The weight of $\Delta v_2 v_4 v_6$ is

$|v_2v_4|+|v_4v_6|+|v_2v_6|$

Example 3. Optimal Polygon Triangulation

Dynamic Solution: Let t[i,j] be the weight of an optimal triangulation of polygon <$v_{i-1}, v_i, \ldots, v_j$>.

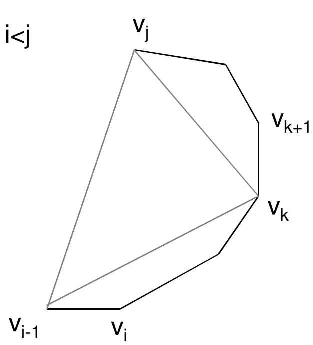


Example 3. Optimal Polygon Triangulation

Dynamic Solution: Let t[i,j] be the weight of an optimal triangulation of polygon <$v_{i-1}, v_i, \ldots, v_j$>.

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

$t[i,i] = 0$



Example 3. Optimal Polygon Triangulation

Dynamic Solution: Let t[i,j] be the weight of an optimal triangulation of polygon <$v_{i-1}, v_i, \ldots, v_j$>.

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

$t[i,i] = 0$

Almost identical to the matrix chain multiplication problem!