Analysis and Design of Algorithms

National University of Singapore

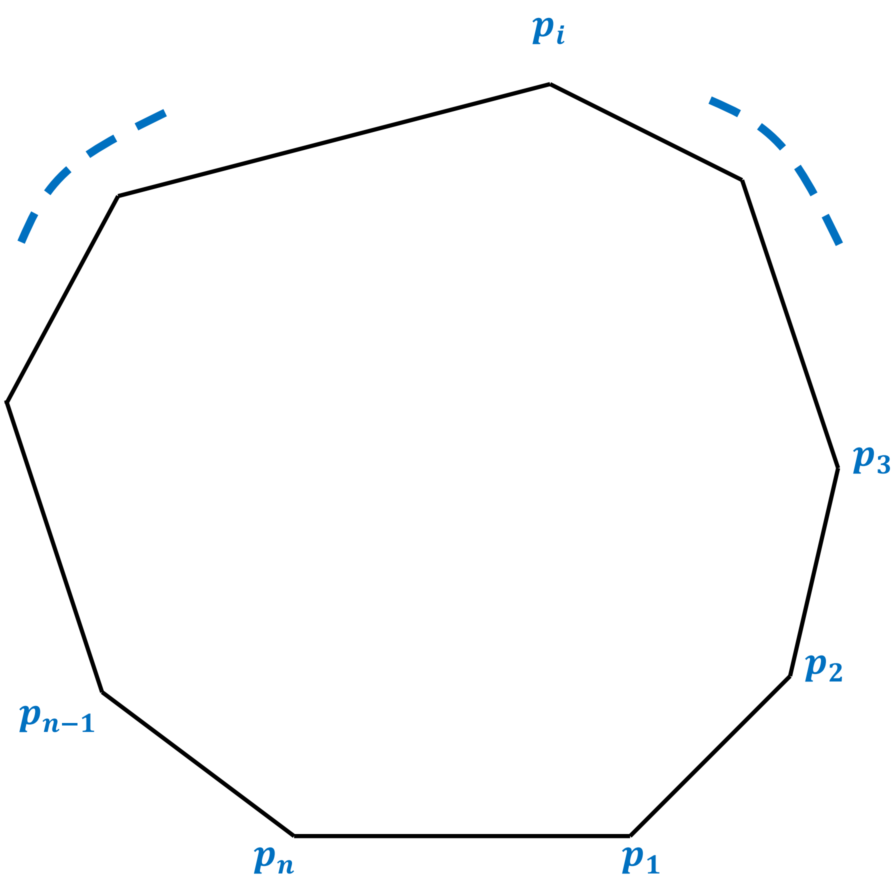
Algorithms

CS3230

Tutorial

Week 9

A Convex Polygon



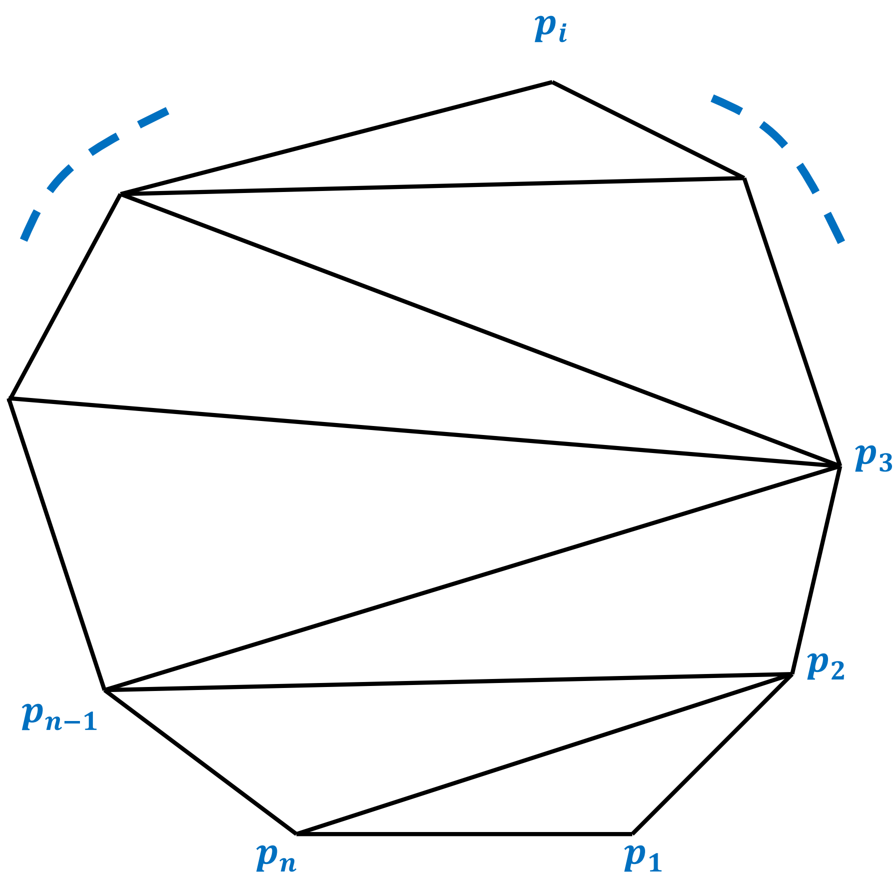
Representation:

< p_1 ,..., p_n > Stored in an array.

< p_i ,..., p_j > :

Polygon consisting of points p_i ,..., p_j

Triangulation of A Convex Polygon

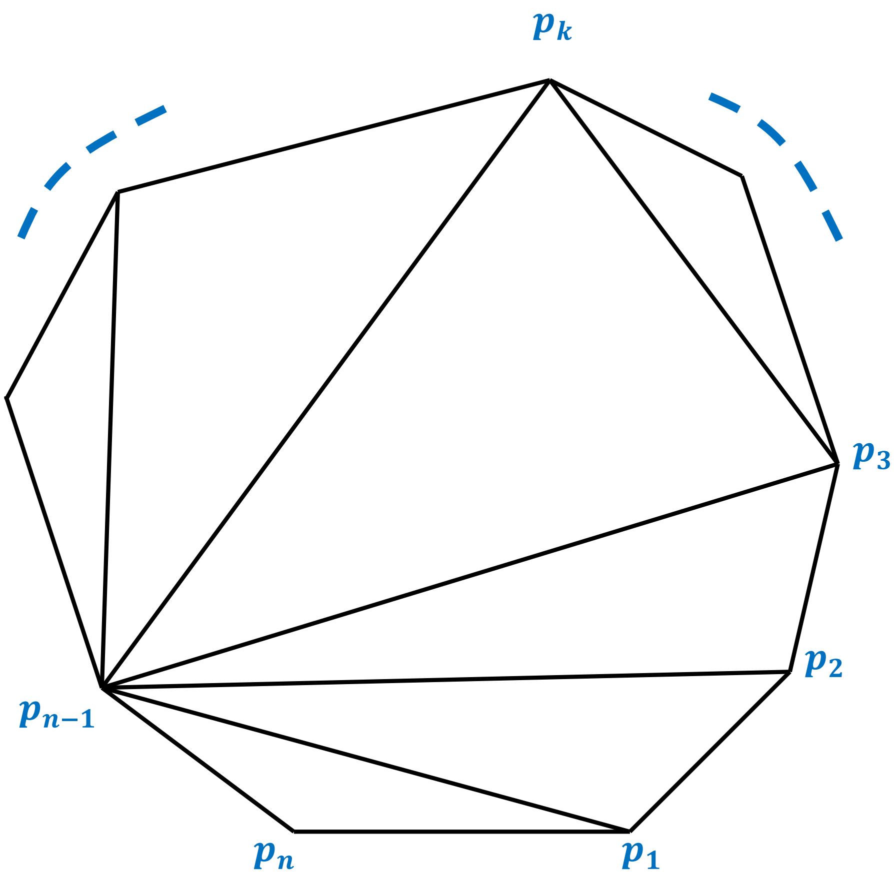


$\omega(i,j,k)$: Weight of triangle formed by p_i, p_j, p_k.

Assumption: It takes **O**(1) time to compute $\omega(i,j,k)$

Cost of a triangulation : Sum of the weight of $n-2$ triangles formed.

Triangulation of A Convex Polygon



$\omega(i, j, k)$: Weight of triangle formed by p_i, p_j, p_k.

Assumption: It takes **O**(1) time to compute $\omega(i, j, k)$

Cost of a triangulation : Sum of the weight of $n-2$ triangles formed.

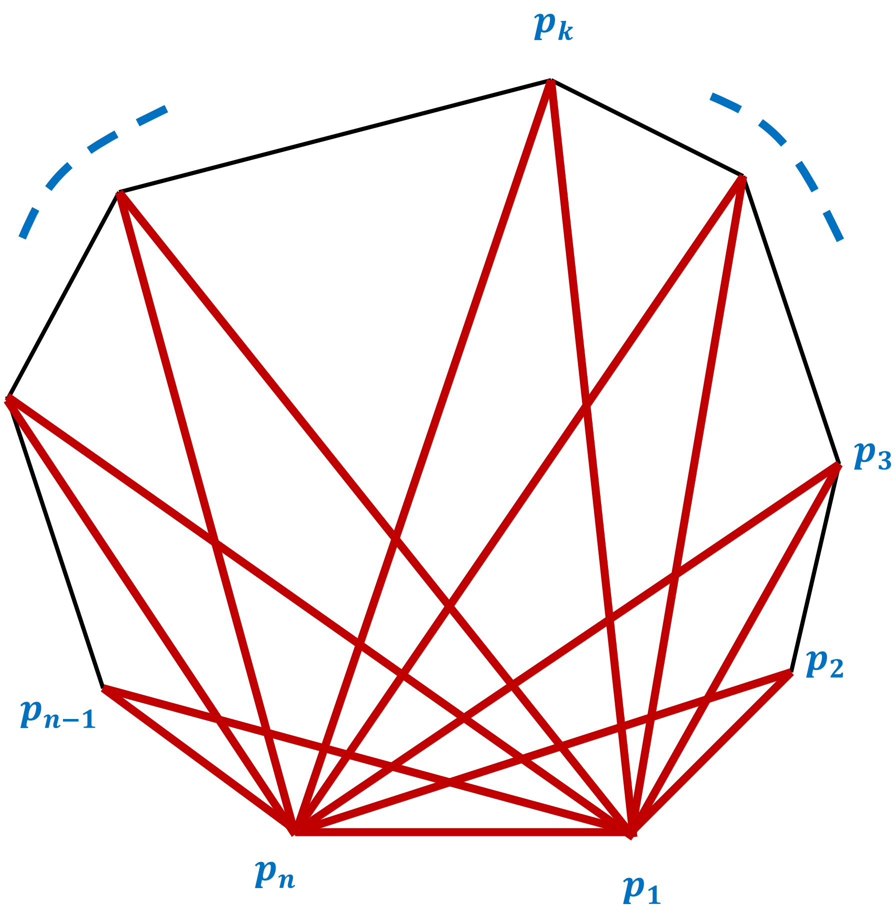
Question 1

Problem: Given a convex polygon represented by < $\boldsymbol{p_1}$,…, $\boldsymbol{p_n}$ >, the objective is to find a triangulation with minimum cost.

Let $\tau(i,j)$: cost of an optimal triangulation of polygon ($\boldsymbol{p_i}$,…,$\boldsymbol{p_j}$)

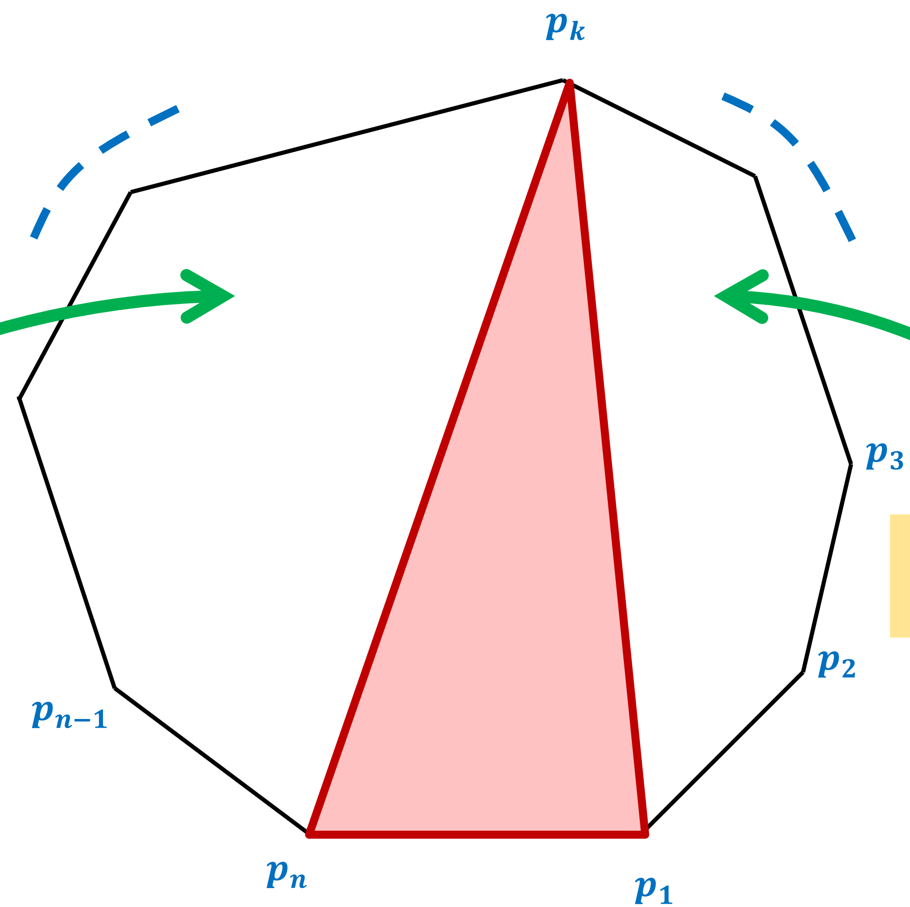
Write down a recursive formula for the above problem, i.e., express $\tau(i,j)$ in terms of $\tau(i',j')$'s where $j' - i' < j - i$ and $j' \leq j$, $i' \geq i$.

How to compute optimal triangulation ?



How to compute optimal triangulation ?

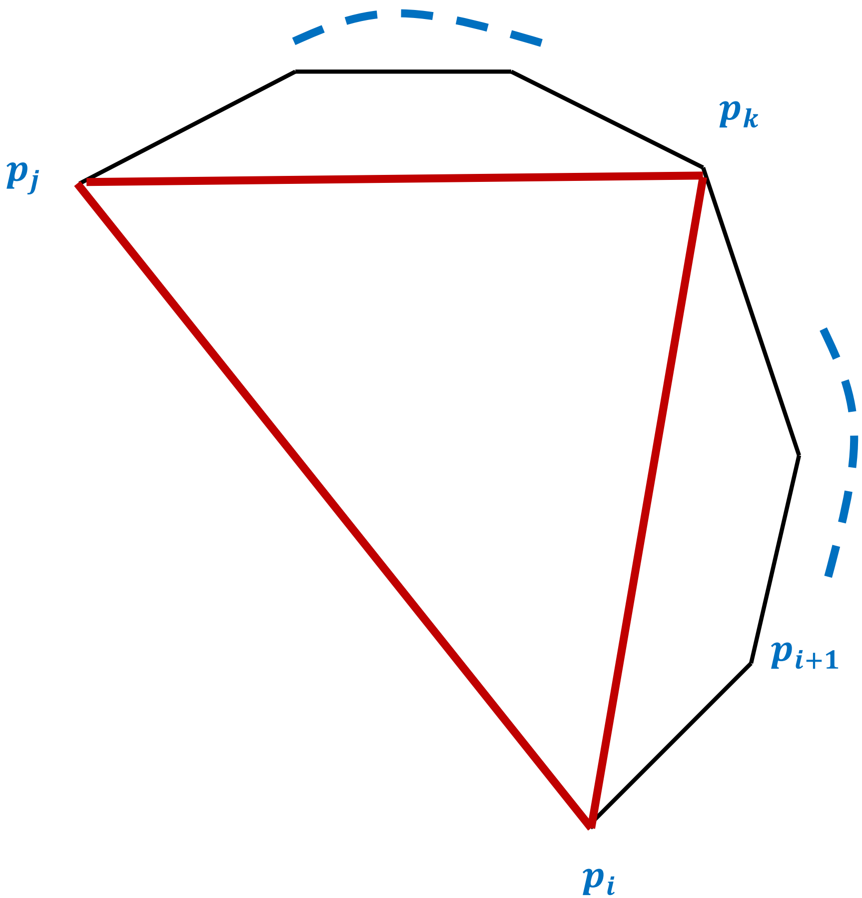
If the opt. triangulation has triangle (p_1, p_k, p_n), what can we infer ?



Polygon (p_k,…,p_n) is optimally triangulated

Polygon (p_1,…,p_k) is optimally triangulated

Recursive formulation for $\tau(i,j)$



$\tau(i,i+1) = 0$

For $j > i+1$

$$\tau(i,j) = \min_{i<k<j} \left(\tau(i,k) + \tau(k,j) + \omega(i,k,j)\right)$$

Question 2

Consider the following algorithm to find the value of $\tau(i,j)$

```
Find-τ(i,j)
{If (j = i + 1)
     return 0;
 Else
 {  t ← ∞;
    For (i < k < j)
    {
       temp ←Find-τ(i,k) + Find-τ(k,j) + ω(i,k,j);
       If (t > temp)
          t ← temp;
    }
 }
 return t;
}
```

What is the running time?

1. $2^{O(j-i)}$
2. $O((j-i)^2)$
3. $O((j-i)^3)$

Question 2

Answer: A

n is the number of points

$\boldsymbol{T}(n)$: worst case running time of $\tau(1,n)$

$\boldsymbol{T}(2) = c$

$$\begin{aligned}\boldsymbol{T}(n) = \quad & \boldsymbol{T}(2) + \boldsymbol{T}(n-1) + c \\ & + \boldsymbol{T}(3) + \boldsymbol{T}(n-2) + c \\ & \vdots \\ & + \boldsymbol{T}(n-1) + \boldsymbol{T}(2) + c\end{aligned}$$

$\Rightarrow T(n) = 2\sum_{i=2}^{n-1} T(i) + c(n-2)$

$\Rightarrow \boldsymbol{T}(n) - \boldsymbol{T}(n-1) = 2\,\boldsymbol{T}(n-1) + c$

$\Rightarrow \boldsymbol{T}(n) = 3\,\boldsymbol{T}(n-1) + c$

$\boldsymbol{T}(n) = 2^{O(n)}$

Exponential !!

Question 3

Consider the previous Find−$\tau(1, n)$ algorithm. Which one of the following is/are true.

1. Find−$\tau(1, n)$ computes 2^n different sub-problems

2. Find−$\tau(1, n)$ computes only at most n^2 different sub-problems, but to compute each sub-problem it takes $O(\frac{2^n}{n^2})$ time

3. Find−$\tau(1, n)$ computes only at most n^2 different sub−problems, but each sub-problem multiple times.

Question 3

Answer: 3

Draw the recursion tree.

Question 4

Consider the following algorithm

Iterative-opt-traingulation $(1,n)$

{**for** $(i = 1$ to $n-1)$ $\boldsymbol{T}[i,i+1] \leftarrow 0$;

for $(k = i+1$ to $j-1)$

{

}

}

} return $\boldsymbol{T}[1,n]$;

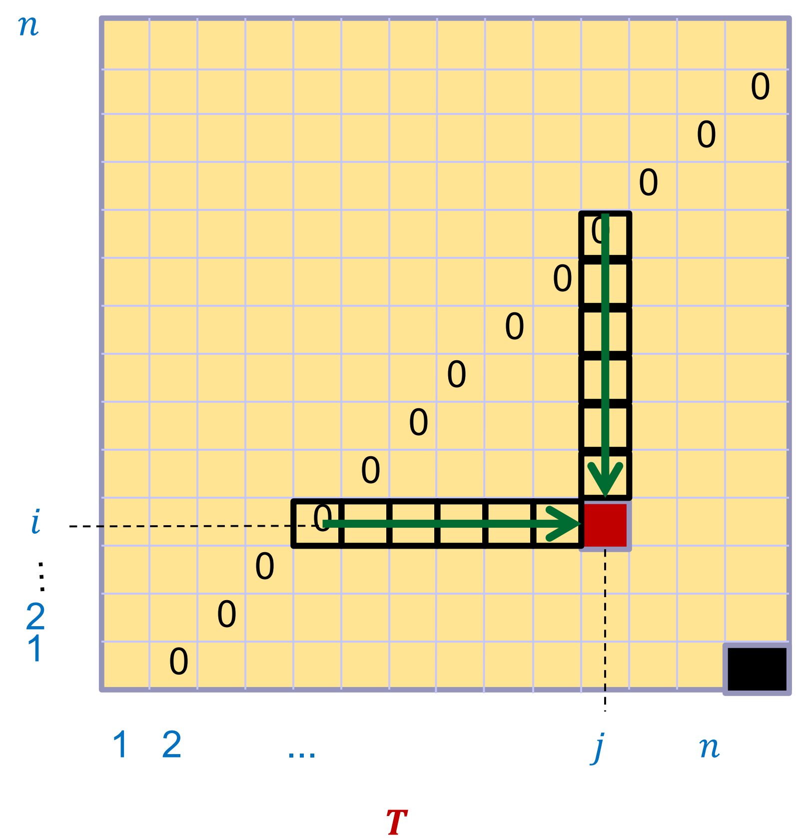
Fill the blocks so that the following are true:

1. This algorithm finds the value of $\tau(i,j)$
2. This algorithm runs in time $O(n^3)$time
3. This algorithm computes only at most n^2 different sub−problems, each exactly once

Recursive algorithm for $\tau(i,j)$

```
Find-τ(i,j)
{If (j = i + 1)
      return 0;
 Else
 {  t ← ∞;
    For (i < k < j)
    {
       temp ←Find-τ(i,k) + Find-τ(k,j)
+ ω(i,k,j);
       If (t > temp)
           t ← temp;
    }
 }
 return t;
}
```

$T[i,j] = \tau(i,j)$



Iterative algorithm for $\tau(i,j)$

Iterative-opt-traingulation $(1,n)$

{**for** $(i = 1$ to $n-1)$ $\quad T[i,i+1] \leftarrow 0$;

 for $(\Delta = 2$ to $n-1)$

 { **for** $(i = 1$ to $n-1-\Delta)$

  { $j \leftarrow i + \Delta$;

   $T[i,j] \leftarrow \infty$;

   for $(k = i+1$ to $j-1)$

   { $temp \leftarrow T[i,k] + T[k,j] + \omega(i,k,j)$;

    If $(T[i,j] > temp)$

     $T[i,j] \leftarrow temp$;

   }

  }

 }

return $T[1,n]$;

$T[i,j] = \tau(i,j)$