Analysis and Design of Algorithms

National University of Singapore

Algorithms

CS3230

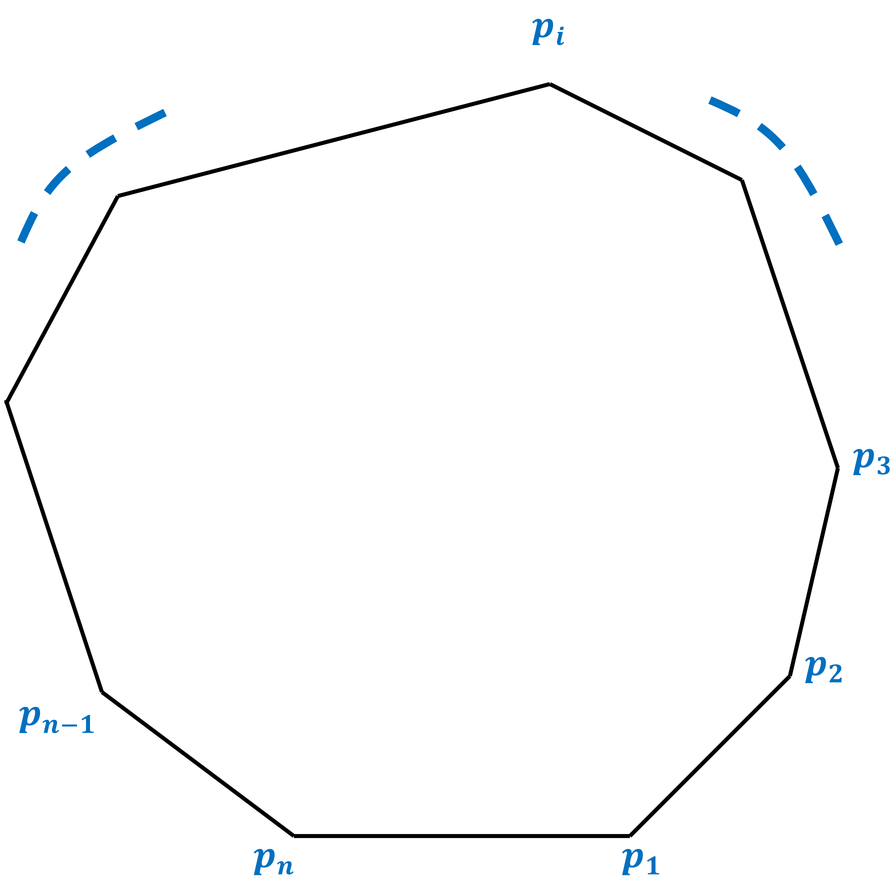
Tutorial

Week 10

Dynamic Programming algorithm paradigm (Recap)

- Expressing the solution recursively
- Overall there are only small (maybe polynomial) number of subproblems
- But there is a huge overlap among the subproblems. So the recursive algorithm takes exponential time (solving same subproblem multiple times)
- So we compute the recursive solution iteratively in a bottom-up fashion. This avoids wastage of computation and leads to an efficient implementation

A Convex Polygon



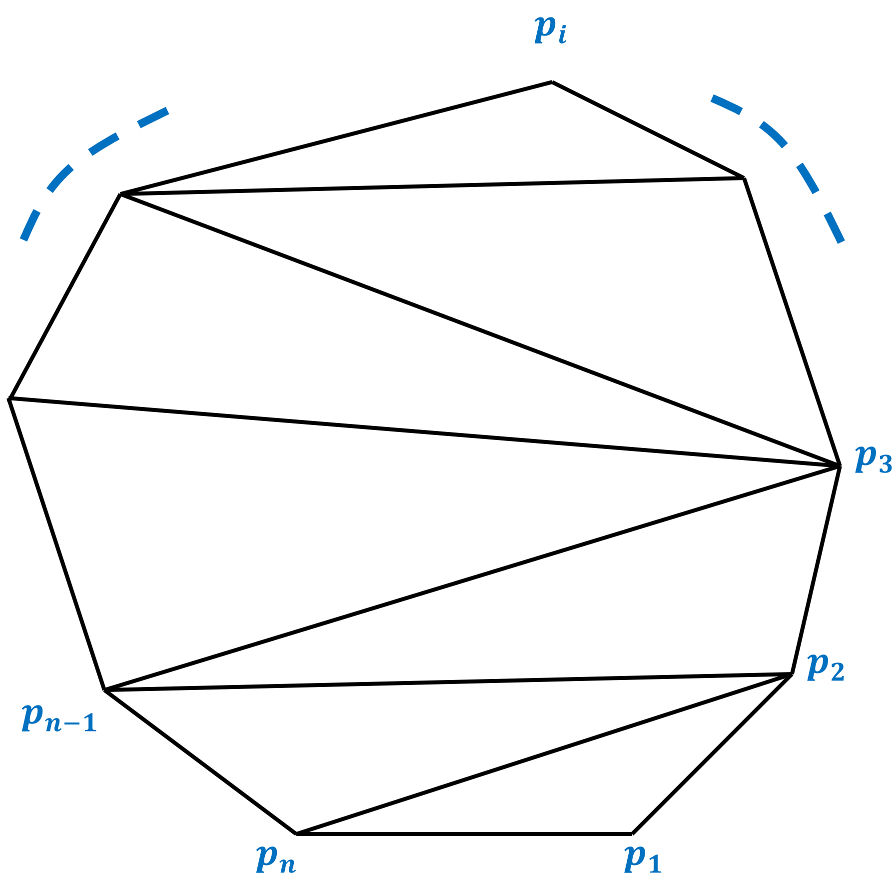
Representation:

$< p_1 ,\ldots, p_n >$ Stored in an array.

$< p_i ,\ldots, p_j >$:

Polygon consisting of points $p_i ,\ldots, p_j$

Triangulation of A Convex Polygon

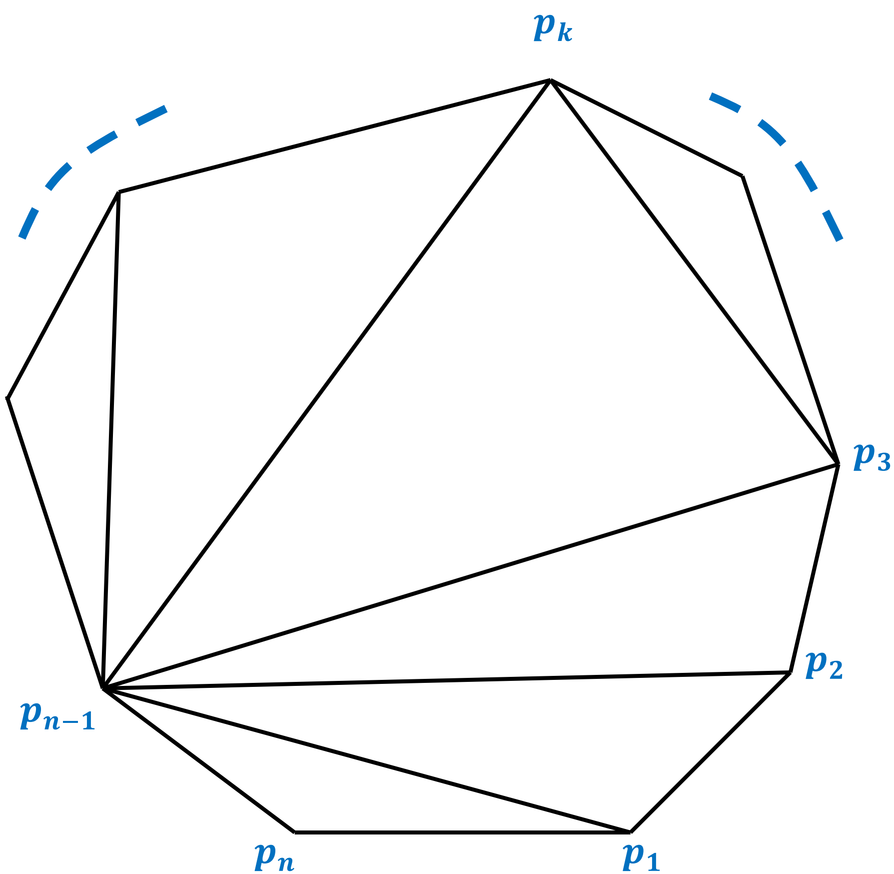


$\omega(i,j,k)$: Weight of triangle formed by p_i, p_j, p_k.

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 $< \boldsymbol{p_1}, \ldots, \boldsymbol{p_n} >$, the objective is to find a triangulation with minimum cost.

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

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 $j' \leq j$, $i' \geq i$.

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 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 (non-recursively) it takes $\Omega(\frac{2^n}{n^2})$ time

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

Question 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