Dynamic Programming (Contd.)

Interval Scheduling:

Requests $R_1, \dots, R_n$

Priority $w_1, \dots, w_n$

Objective:

$$\max_{S \subseteq \{R_1,\dots,R_n\}} \left(\sum_{i \in S} w_i \right)$$

$R_i \to start_i$, $R_i \to finish_i$

Optimal $(R_1, \dots, R_n)$

Case-1: R_n may belong to the optimal seq.

$R_1 \quad R_2 \quad \dots \quad R_n$ (Sorted in the order of finish times)

* Remove all incompatible requests w/ R_n.

For a compatible job R_i, we must have $finish_i < start_n$.

$p(j) = \max \{ i \mid R_i$ is compatible with $R_j \}$, $i < j$

max Request whose finish time is less than start time of j

$R_1, \dots, R_{p(n)}$ are compatible with R_n but not $R_{p(n)+1}, \dots, R_{n-1}$.

$p(n)$ may not be $\in [1, n-1]$.

Compute Optimal $(R_1, \dots, R_{p(n)})$ } $w_n +$ Optimal $(R_1, \dots, R_{p(n)})$ or $w_n + \{\}$.

Case-2: R_n may not belong to optimal seq.

Compute Optimal $(R_1, \dots, R_{n-1})$.

$$\text{Optimal}(R_1,\dots,R_n) = \max \begin{cases} \text{Optimal}(R_1,\dots,R_{n-1}) \\ w_n \\ w_n + \text{Optimal}(R_1,\dots,R_{p(n)}) \end{cases}$$

$$\text{Optimal}(R_1) = \max \begin{cases} 0 \\ w_1 \end{cases}$$

W.L.O.G Assume that weights are positive.

0-1 knapsack.

Items	$w_1, w_2, \dots, w_n$ ($\mathbb{Z}$)	
Values	$v_1, v_2, \dots, v_n$	

$$\max_{S \subseteq [n]} \sum_{i \in S} v_i$$

subject to

$$\sum w_i \leq \underline{W}$$

Either $\text{Item}_n \in \text{Opt}$ or not:

$\hookrightarrow$ $\text{Item}_n \in \text{Opt}$: $w_n \overset{?}{\leq} W$

$\hookrightarrow$ Yes $\text{Opt}([1, n-1], W - w_n)$

$\downarrow$

$\text{Opt}([1, n-1], W)$

$$\text{Opt}([1,n], W) = \max \begin{cases} \text{Opt}([1,n-1], W - w_n) + v_n \\ \text{Opt}([1,n-1], W) \end{cases}$$

w' can take 2^n many values.

$\underline{\underline{n \times W}}$

$\text{Opt}([1,i], w')$

i: $1, n$

Say W is very very large

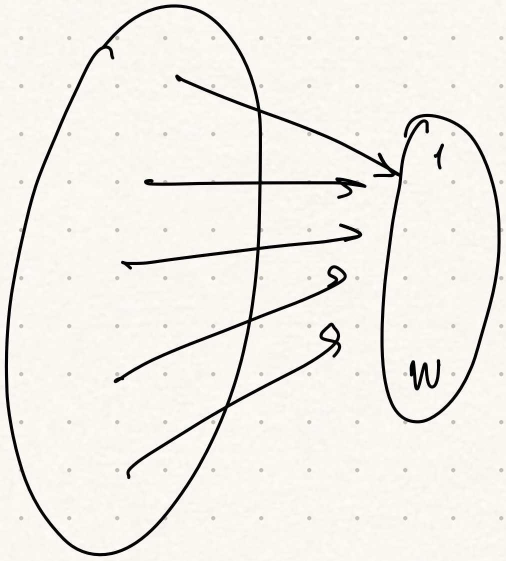
$a_i \in \{0,1\}^n$

$$W' = \sum_{i=1}^{n} a_i w_i$$

If W is small,

$\bar{a}$ s.t.

$$\sum_{i: a_i = 1} w_i \le W$$



$$\mathrm{Opt}\Big(\underline{[1,i]},\ \underbrace{W - \sum_{j=i+1}^{n} \alpha_j \cdot w_j}_{2^{n-i}}\Big)$$

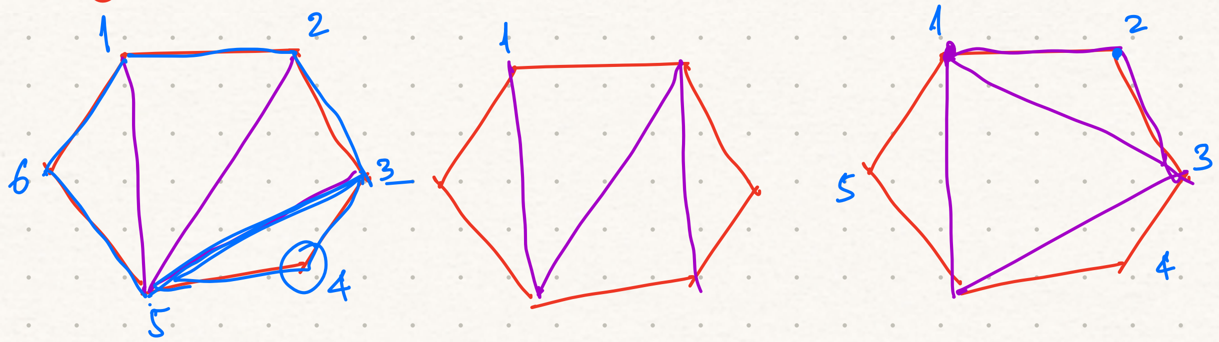
$w_{i+1}, \dots, w_n$

$\underbrace{\bar{\alpha}_{i+1} \quad \bar{\alpha}_n}_{(n-i)} \in \{0,1\}$

Triangulation of Polygons:

$\forall a_1, a_2 \in$ polygon

$\forall \lambda \in [0,1],\ \lambda a_1 + (1-\lambda) a_2 \in$ polygon



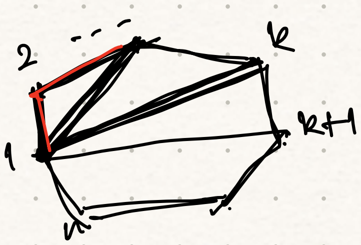
Cost: Sum of the lengths of diagonals introduced $\}$ S

↑ want to minimize this cost.

((Minimize the sum of perimeters of the triangles.

$2S + C$ ← polygon

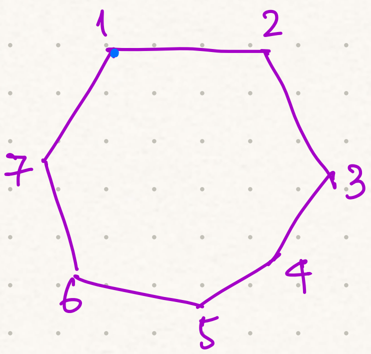
Cost of triangulation

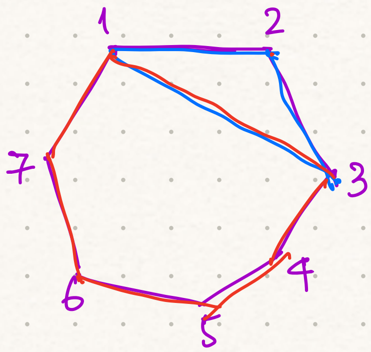


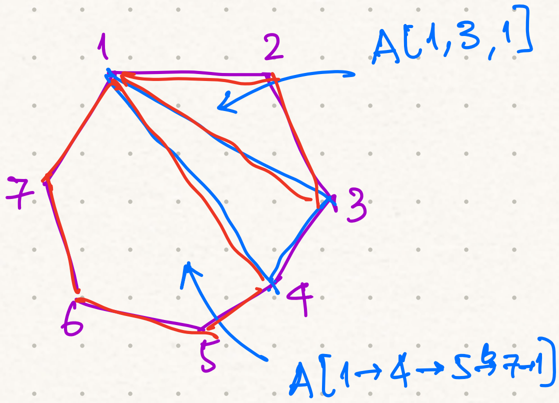
i, j

$A[i,j]$ denote the min cost of triangulation of polygon with vertices $i, i+1, \cdots, j$.

$$A[i,j] = \min_{i \le k \le j} \{ A[i,k] + A[k,j] + P(i,j,k) \}.$$





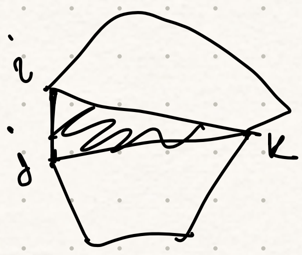


$A[i,j] \leftarrow$ Cost of triangulation with vertices $i, i+1, \ldots, j-1, j$

$i \le k \le j$

$A[i,k]$ $A[k,j]$

$i, i+1, \ldots, k$ $k, \ldots, j$



$i<j$

$$A[i,j] = \min_k \{ A[1,k] + A[k,j] + w_{i \to k} + w_{k \to j} + w_{j \to i} \}.$$

$\rightarrow$ $j = i+1$ $\leftarrow$ zero diagonals in it }

$j = i + 2 \leftarrow$ 1

$j = i + 3$

$i, j \longrightarrow j - i$

$$\sum_{i=1}^{n} \sum_{j>i}^{n} (j - i)$$

$$= \sum_{i} \left(\sum_{j>i}^{n} j \right) - (n-i)i$$

$$= \sum_{i}^{n} \left(\frac{n(n+1)}{2} - \frac{i(i+1)}{2} \right) + i^2 - n \cdot i$$