3-1 Dynamic Programming

(Part I: Examples)

Hengfeng Wei

hfwei@nju.edu.cn

September 10, 2018

Taolu

Steps for Applying DP:

Steps for Applying DP:

(I) Define subproblems

(II) Set the goal

(III) Identify the recurrence

- larger subproblem $\leftarrow$ # smaller subproblems
- init. conditions

Steps for Applying DP:

(I) Define subproblems

(II) Set the goal

(III) Identify the recurrence
- larger subproblem $\leftarrow$ # smaller subproblems
- init. conditions

(IV) Write pseudo-code: filling in "tables" in some order

(V) Analyze the time complexity

(VI) Extract the optimal solution (optionally)

Steps for Applying DP:

(I) Define subproblems

(II) Set the goal

(III) Identify the recurrence
- larger subproblem $\leftarrow$ # smaller subproblems
- init. conditions

(IV) Write pseudo-code: filling in "tables" in some order

(V) Analyze the time complexity

(VI) Extract the optimal solution (optionally)

Steps for Applying DP:

(I) Define subproblems

(II) Set the goal

(III) Identify the recurrence

- larger subproblem $\leftarrow$ # smaller subproblems
- init. conditions

(IV) Write pseudo-code: filling in "tables" in some order

(V) Analyze the time complexity

(VI) Extract the optimal solution (optionally)

1D Subproblems

Input: $x_1, x_2, \ldots, x_n$ (array, sequence, string)

Subproblems: $x_1, x_2, \ldots, x_i$ (prefix/suffix)

#: $\Theta(n)$

Examples:

- Rod cutting
- Maximum-sum subarray
- Longest increasing subsequence
- Text justification (LaTeX)

2D Subproblems

(I) Input: $x_1, x_2, \ldots, x_m$; $y_1, y_2, \ldots, y_n$

Subproblems: $x_1, x_2, \ldots, x_i$; $y_1, y_2, \ldots, y_j$

#: $\Theta(mn)$

Examples: Edit distance, Longest common subsequence

2D Subproblems

(I) Input: $x_1, x_2, \ldots, x_m$; $y_1, y_2, \ldots, y_n$

Subproblems: $x_1, x_2, \ldots, x_i$; $y_1, y_2, \ldots, y_j$

#: $\Theta(mn)$

Examples: Edit distance, Longest common subsequence

(II) Input: $x_1, x_2, \ldots, x_n$

Subproblems: $x_i, \ldots, x_j$

#: $\Theta(n^2)$

Examples: Matrix chain multiplication, Optimal BST

3D Subproblems

- Floyd-Warshall algorithm

$$\mathsf{d}(i,j,k) = \min\Big(\mathsf{d}(i,j,k-1), \mathsf{d}(i,k,k-1) + \mathsf{d}(k,j,k-1)\Big)$$

DP on Graphs

(I) On rooted tree

Subproblems: rooted subtrees

(II) On DAG

Subproblems: nodes after/before in the topo. order

DP on Graphs

(I) On rooted tree

Subproblems: rooted subtrees

(II) On DAG

Subproblems: nodes after/before in the topo. order

Knapsack Problem

Subset sum problem, Change-making problem

And Others . . .

And Others . . .

How to identify the recurrence?

How to identify the recurrence?

GUESS

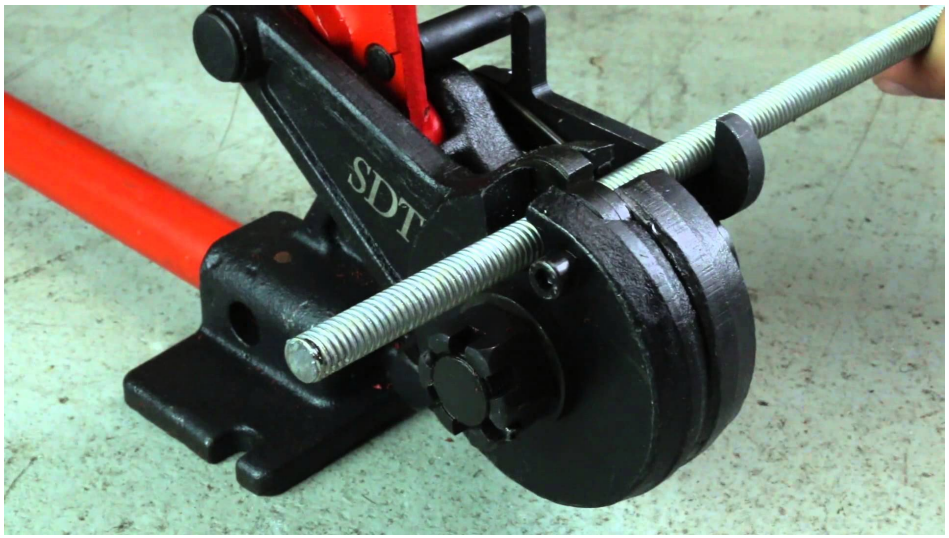
Make Choices by asking yourself the right question

Make Choices by asking yourself the right question

(I) Binary choice
- whether ...

(II) Multi-way choices
- where to ...
- which one ...

Rod Cutting



Rod Cutting Problem

Rod of length n

Rod Cutting Problem

Rod of length n

length i	1	2	3	4	5	$\cdots$
price p_i	1	5	8	9	10	$\cdots$

Rod Cutting Problem

Rod of length n

length i	1	2	3	4	5	$\cdots$
price p_i	1	5	8	9	10	$\cdots$

$$n = i_1 + i_2 + \cdots + i_k$$

$$r_n = p_{i_1} + p_{i_2} + \cdots + p_{i_k}$$

Subproblem: $R(i)$: max revenue obtained from cutting a rod of length i

Goal: $R(n)$

Subproblem: $R(i)$: max revenue obtained from cutting a rod of length i

Goal: $R(n)$

Make Choice: Where is the *first* cut?

Recurrence:

$$R(i) = \max_{1 \le j \le i} \Big(p_j + R(i-j) \Big)$$

Subproblem: $R(i)$: max revenue obtained from cutting a rod of length i

Goal: $R(n)$

Make Choice: Where is the *first* cut?

Recurrence:

$$R(i) = \max_{1 \le j \le i} \Big(p_j + R(i-j) \Big)$$

Init:

$$R(0) = 0$$

Subproblem: $R(i)$: max revenue obtained from cutting a rod of length i

Goal: $R(n)$

Make Choice: Where is the *first* cut?

Recurrence:

$$R(i) = \max_{1 \le j \le i} \Big(p_j + R(i-j) \Big)$$

Init:

$$R(0) = 0$$

Time:

$$O(n^2) = \Theta(n) \cdot O(n)$$

Rod Cutting Problem (Problem 15.1-3)

Each cut incurs a fixed cost of c.

Rod Cutting Problem (Problem 15.1-3)

Each cut incurs a fixed cost of c.

$$R(i) = \max_{1 \le j \le i} \Big(p_j - c + R(i-j) \Big)$$

Longest Increasing Subsequence (Problem 15.4-5)

$$A[1 \ldots n]$$

$$5,\ 2,\ 8,\ 6,\ 3,\ 6,\ 9,\ 7$$

Find (the length of) a longest increasing (non-decreasing) subseqence.

$$5,\ 2,\ 8,\ 6,\ 3,\ 6,\ 9,\ 7$$

$L(i)$: the length of an LIS of $A[1 \ldots i]$

$L(n)$

$L(i)$: the length of an LIS of $A[1 \ldots i]$

$L(n)$

Is $A[i]$ in this LIS of $A[1 \cdots i]$?

$L(i)$: the length of an LIS of $A[1 \ldots i]$

$L(n)$

Is $A[i]$ in this LIS of $A[1 \cdots i]$?

$$L(i) = \max\Big(\underbrace{L(i-1)}_{\text{NO}}, \qquad \Big)$$

$L(i)$: the length of an LIS of $A[1 \dots i]$

$L(n)$

Is $A[i]$ in this LIS of $A[1 \cdots i]$?

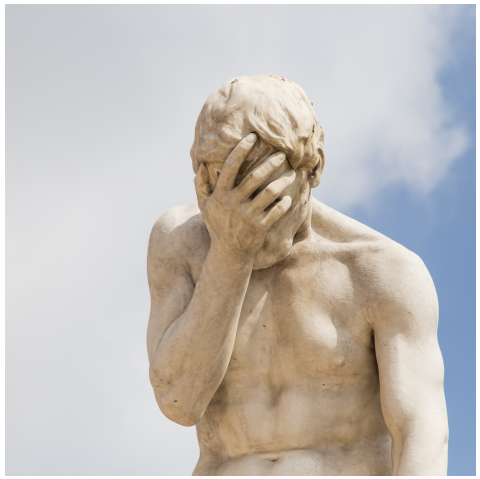
$$L(i) = \max\Big(\underbrace{L(i-1)}_{\text{NO}}, \underbrace{1 + \max_{j<i \,\wedge\, A[j] \le A[i]} L(j)}_{\text{YES}} \Big)$$

$L(i)$: the length of an LIS of $A[1 \dots i]$

$L(n)$

Is $A[i]$ in this LIS of $A[1 \cdots i]$?

$$L(i) = \max\Big(\underbrace{L(i-1)}_{\text{NO}}, \underbrace{1 + \max_{j<i \,\wedge\, A[j] \le A[i]} L(j)}_{\text{YES}} \Big)$$



$L(i)$: the length of an LIS *ending with* $A[i]$

$$\max_i L(i)$$

$L(i)$: the length of an LIS *ending with* $A[i]$

$$\max_i L(i)$$

What is the previous element?

$L(i)$: the length of an LIS *ending with* $A[i]$

$$\max_i L(i)$$

What is the previous element?

$$L(i) = 1 + \max_{j<i \wedge A[j] \leq A[i]} L(j)$$

$L(i)$: the length of an LIS *ending with* $A[i]$

$$\max_i L(i)$$

What is the previous element?

$$L(i) = 1 + \max_{j<i \wedge A[j] \leq A[i]} L(j)$$

$$L(1) = 1$$

$L(i)$: the length of an LIS *ending with* $A[i]$

$$\max_i L(i)$$

What is the previous element?

$$L(i) = 1 + \max_{j<i \wedge A[j] \leq A[i]} L(j)$$

$$L(1) = 1$$

$$O(n^2) = \Theta(n) \cdot O(n)$$

$$\text{LIS(A)} = \text{LCS}\Big(A, \textsc{Sort}(A)\Big)$$

$$\text{LIS(A)} = \text{LCS}\Big(A, \textsc{Sort}(A)\Big)$$

$$O(n^2) = O(n \log n) + O(n^2)$$

Longest Increasing Subsequence (Problem 15.4-6)

$$O(n \log n)$$

LIS$[i]$: all increasing subsequences of length $1 \leq l \leq i$ using $A[1 \cdots i]$

Longest Increasing Subsequence (Problem 15.4-6)

$$O(n \log n)$$

LIS$[i]$: all increasing subsequences of length $1 \leq l \leq i$ using $A[1 \cdots i]$

$E : E[l] =$ all increasing subsequences of length $l, \quad 1 \leq l \leq n$

Longest Increasing Subsequence (Problem 15.4-6)

$$O(n \log n)$$

LIS$[i]$: all increasing subsequences of length $1 \leq l \leq i$ using $A[1 \cdots i]$

$E : E[l] =$ all increasing subsequences of length $l, \quad 1 \leq l \leq n$

$$\max\{k \mid E[k] \neq \emptyset\}$$

$E[l] =$ all increasing subsequences of length l, $\quad 1 \leq l \leq n$

$E[l]$ = all increasing subsequences of length l, $\quad 1 \leq l \leq n$

$$\{3 \quad 9 \quad 11 \quad 13\}$$

$$\{4 \quad 6 \quad 10 \quad 15\}$$

$E[l]$ = all increasing subsequences of length l, $\quad 1 \leq l \leq n$

$$\{3 \quad 9 \quad 11 \quad 13\}$$

$$\{4 \quad 6 \quad 10 \quad 15\}$$

Keep only the *smallest* ending number per distinct length l

LIS$[i]$: $E[l]$ = the smallest ending number for the increasing subsequence of

Matrix-chain Multiplication

Subproblem: $m[i, j]$: min cost to compute the matrix $A_{i...j}$

Goal: $m[1, n]$

Subproblem: $m[i,j]$: min cost to compute the matrix $A_{i...j}$

Goal: $m[1,n]$

Make Choice: Where is the *last* parentheses?

Recurrence:

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

Subproblem: $m[i,j]$: min cost to compute the matrix $A_{i\ldots j}$

Goal: $m[1,n]$

Make Choice: Where is the *last* parentheses?

Recurrence:

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

Init:

$$m[i,i] = 0$$

Subproblem: $m[i,j]$: min cost to compute the matrix $A_{i\ldots j}$

Goal: $m[1,n]$

Make Choice: Where is the *last* parentheses?

Recurrence:

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

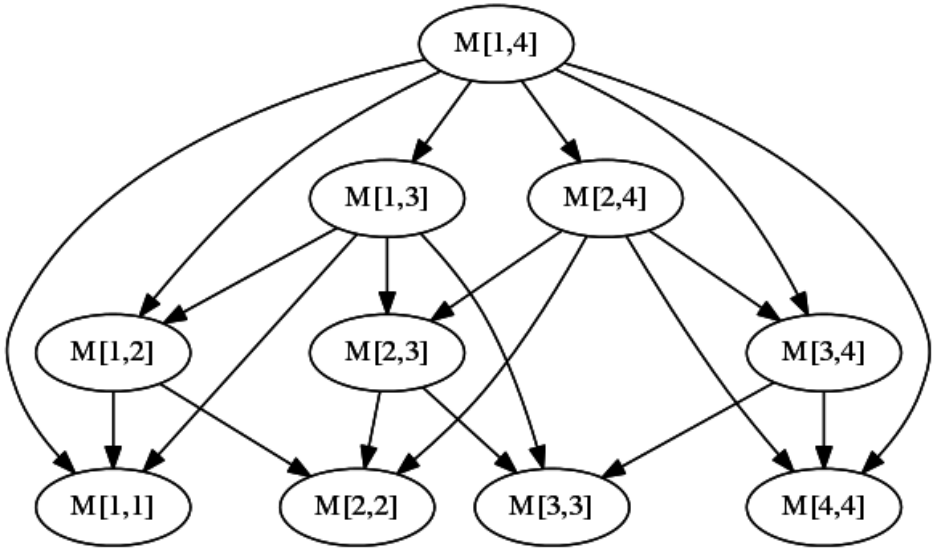
Init:

$$m[i,i] = 0$$

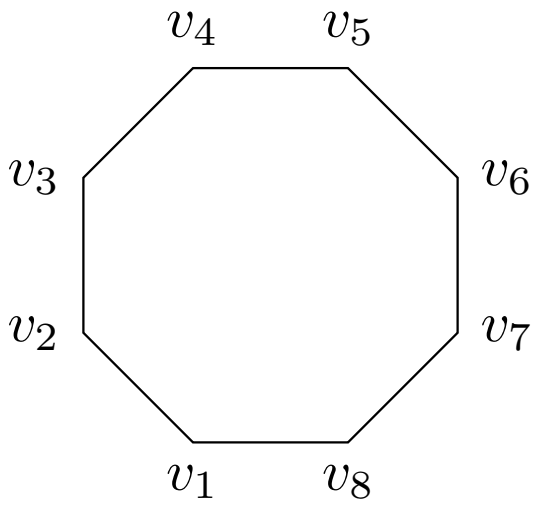
Time:

$$O(n^3) = \Theta(n^2) \cdot O(n)$$

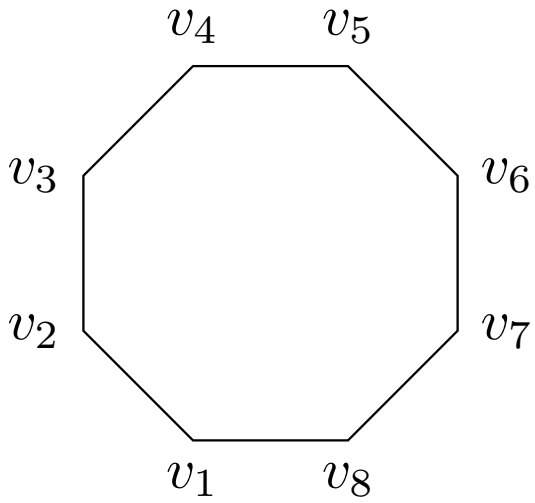
Subproblem Graph for Matrix-chain Multiplication (Problem 15.2-4)

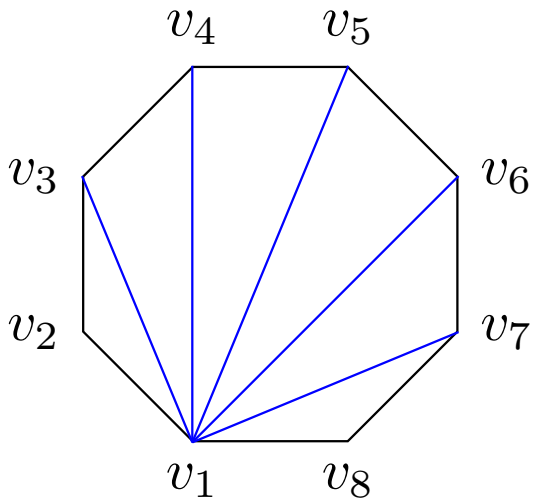


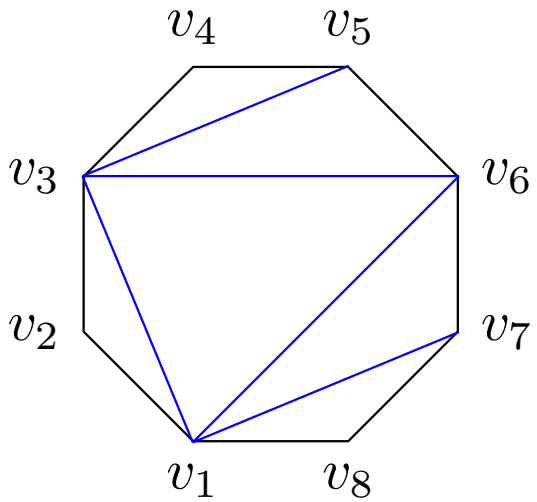
Minimum Weight Triangulation

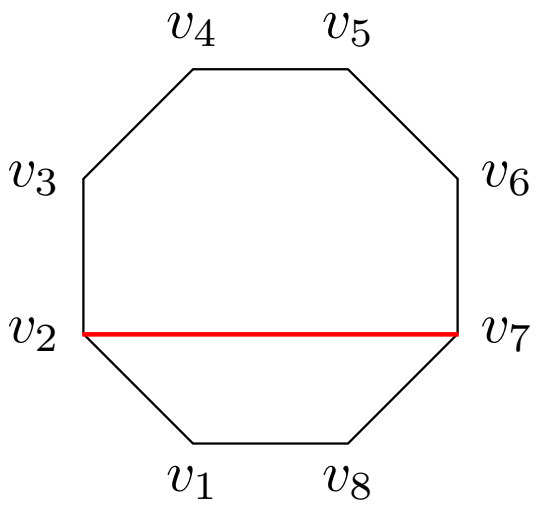


Minimum Weight Triangulation

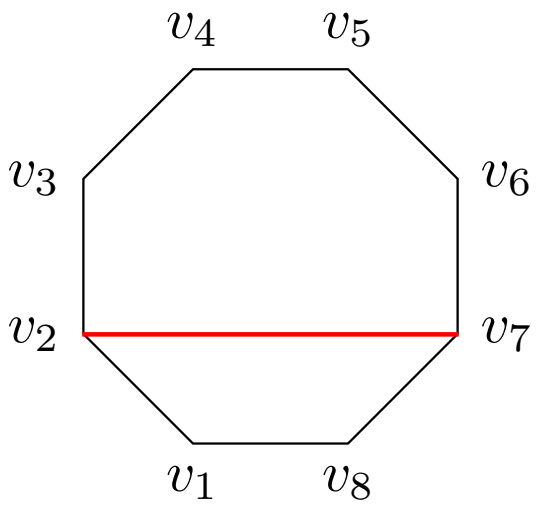






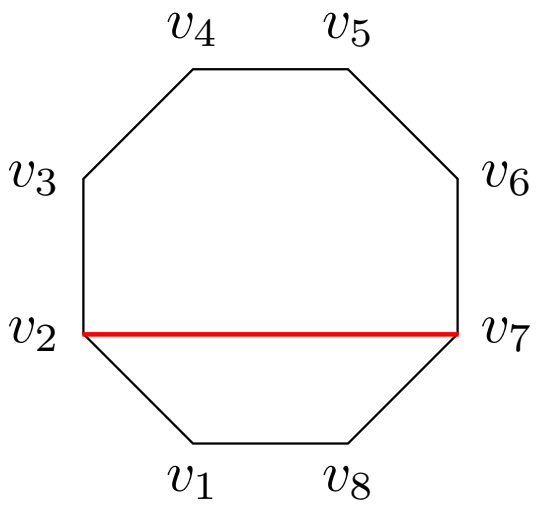


$T(i,j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), *clockwise*



$T(i,j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), *clockwise*

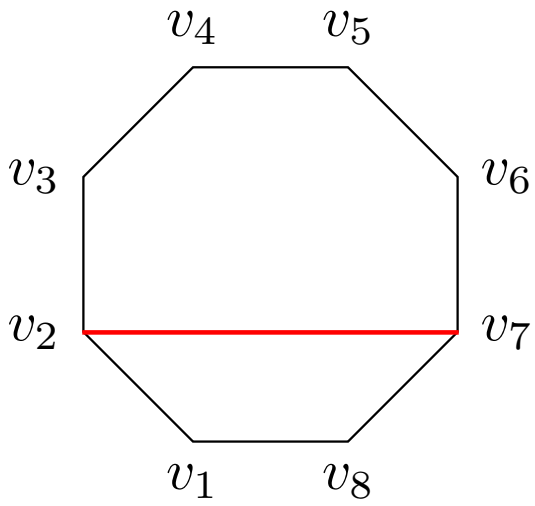
$$T(i,j) = \min_{\substack{i \le k < l-1 \le j \\ (k,l) \neq (i,j)}} \Big(T[k,l] + T[?,?] + d_{ij} \Big)$$

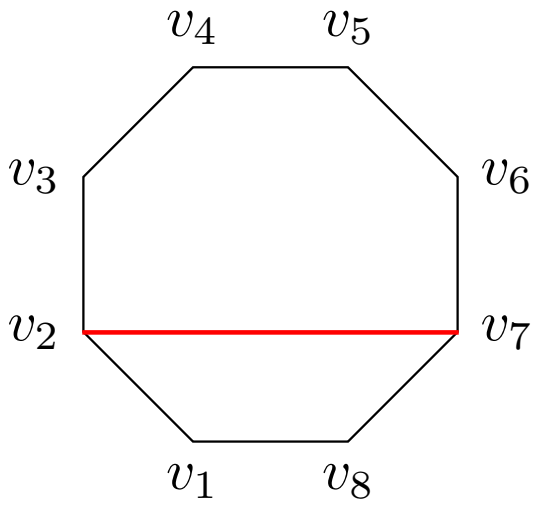


$T(i,j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), *clockwise*

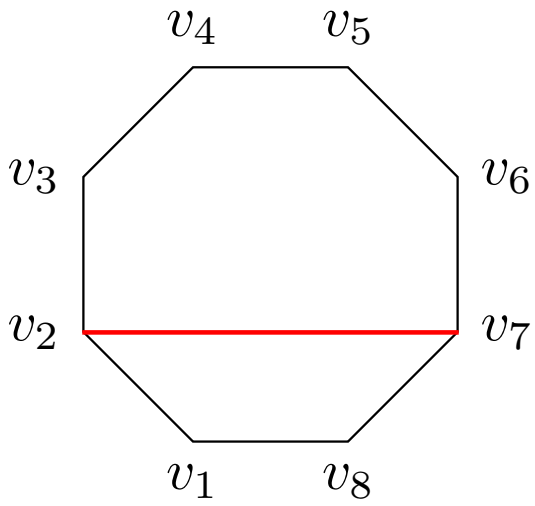
$$T(i,j) = \min_{\substack{i \le k < l-1 \le j \\ (k,l) \neq (i,j)}} \Big(T[k,l] + T[?,?] + d_{ij} \Big)$$

$$O(n^4) = O(n^2) \cdot O(n^2)$$


v_4
v_5
v_3
v_6
v_2
v_7
v_1
v_8

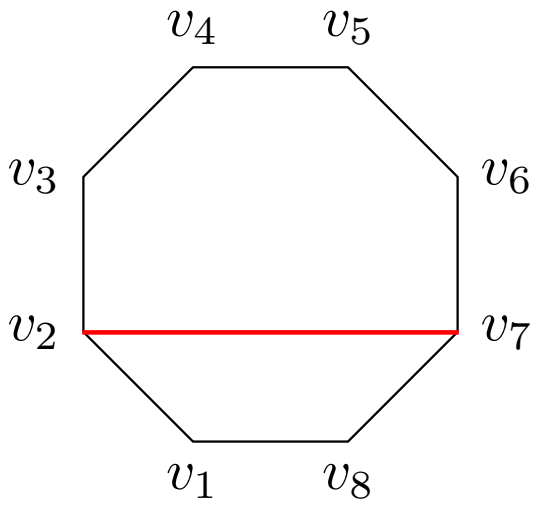


Which vertex k to pair with (i, j)?



Which vertex k to pair with (i, j)?

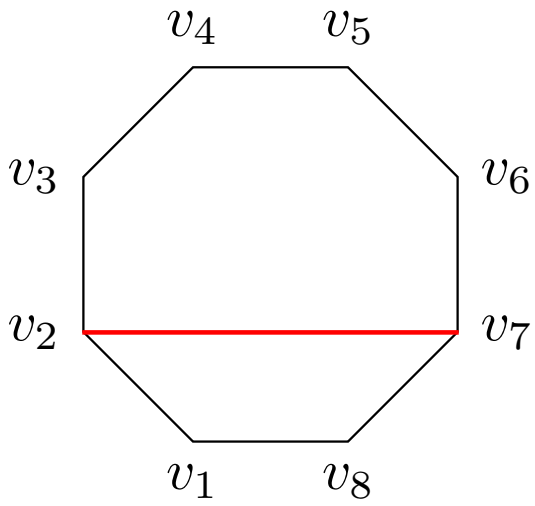
$T(i, j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), $i < j$



Which vertex k to pair with (i, j)?

$T(i, j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), $i < j$

$$T(i, j) = \min_{i<k<j} \Big(T(i, k) + T(k, j) + d_{ik} + d_{kj} \Big)$$

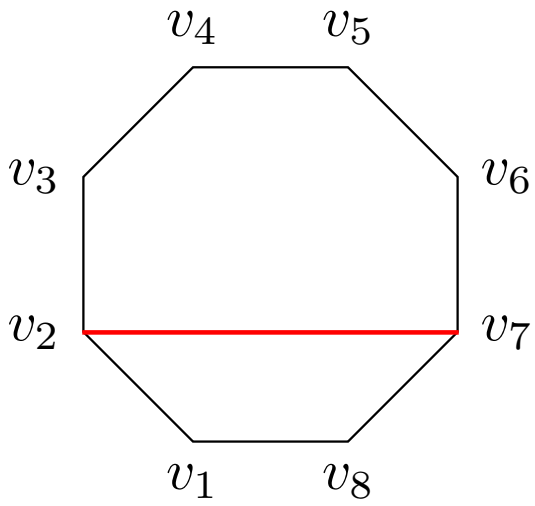


Which vertex k to pair with (i, j)?

$T(i, j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), $i < j$

$$T(i, j) = \min_{i<k<j} \Big(T(i, k) + T(k, j) + d_{ik} + d_{kj} \Big)$$

$$T[i, i + 1] = 0, \quad 1 \le i \le n - 1$$



Which vertex k to pair with (i, j)?

$T(i, j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), $i < j$

$$T(i, j) = \min_{i<k<j} \Big(T(i, k) + T(k, j) + d_{ik} + d_{kj} \Big)$$

$$T[i, i + 1] = 0, \quad 1 \le i \le n - 1$$

$$O(n^3) = O(n^2) \cdot O(n)$$


THANK YOU