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

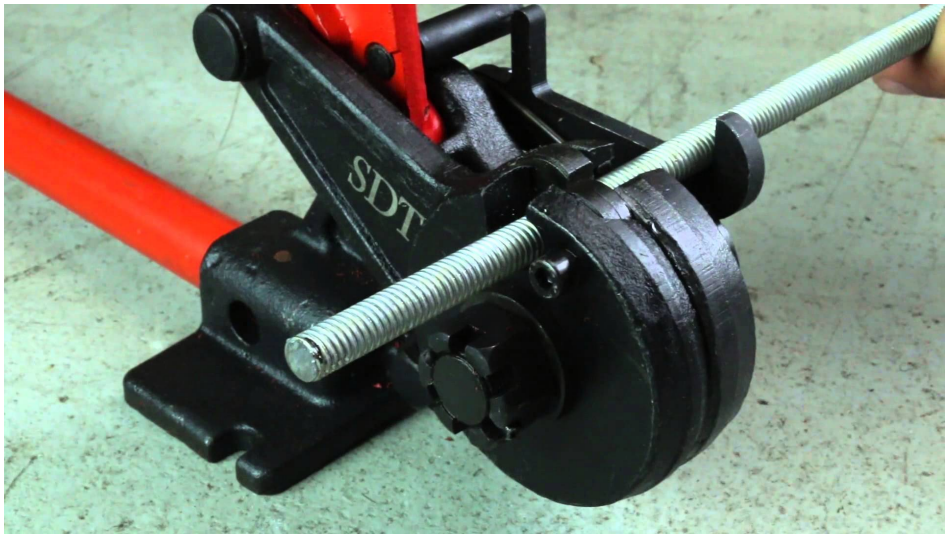
Make Choices by asking yourself the right question

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

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

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

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

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

$$R(n)$$

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

$$R(n)$$

Where is the first cut?

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

$$R(n)$$

Where is the first cut?

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

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

$$R(n)$$

Where is the first cut?

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

$$R(0) = 0$$

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

$$R(n)$$

Where is the first cut?

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

$$R(0) = 0$$

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

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.

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

Printing Neatly (Problem 15-4)

A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

Line width M

$$\mathit{extra}[i, j] = M - (j - i) - \sum_{k=i}^{j} l_k$$

Printing Neatly (Problem 15-4)

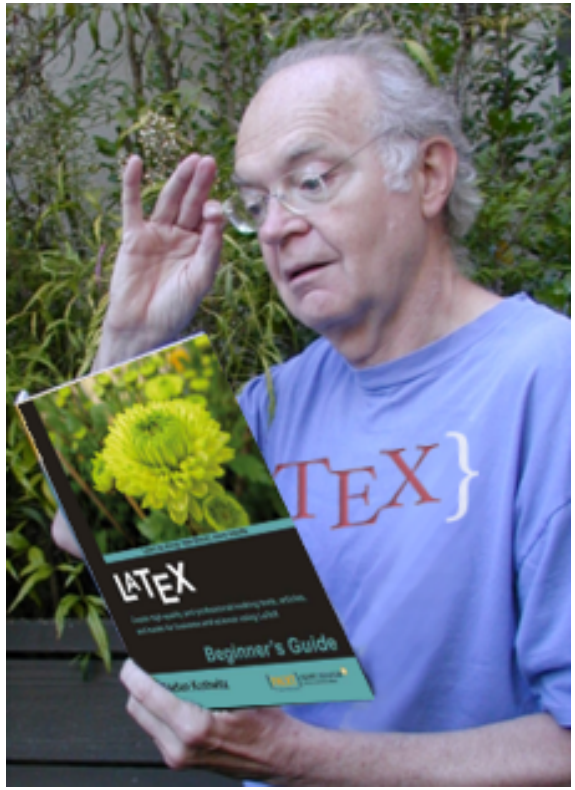
A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

Line width M

$$\mathit{extra}[i,j] = M - (j-i) - \sum_{k=i}^{j} l_k$$

To minimize the sum, over all lines *except the last*, of the cubes of the numbers of extra space characters at the ends of lines.

$$c(n) = \min_{L} \sum_{l_{[i,j]} \in L \wedge j \neq n} \Big(\mathit{extra}[i,j]\Big)^3$$


TEX}
LATEX
Beginner's Guide

A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

$C(i)$: min cost of neatly printing the *first* i words

$C(i)$: min cost of neatly printing the *last* words i through n

$C(i, j)$: min cost of neatly printing *words i through j*

$C(i)$: $\min$ cost of neatly printing the *last* words i through n

$C(i)$: $\min$ cost of neatly printing the *last* words i through n

How many words to place on the first line?

$C(i)$: min cost of neatly printing the *last* words i through n

How many words to place on the first line?

$$C(i) = \min_{\substack{0 < k \le (n-i+1) \\ extra[i, i+k-1] \ge 0}} \Big(extra[i, i+k-1]\Big)^3 + C(i+k)$$

$C(i)$: min cost of neatly printing the *last* words i through n

How many words to place on the first line?

$$C(i) = \min_{\substack{0 < k \le (n-i+1) \\ extra[i, i+k-1] \ge 0}} \Big(extra[i, i+k-1]\Big)^3 + C(i+k)$$

$$C(i) = 0, \quad \text{if } extra[i, n] \ge 0$$

$C(i)$: min cost of neatly printing the *last* words i through n

How many words to place on the first line?

$$C(i) = \min_{\substack{0 < k \le (n-i+1) \\ extra[i, i+k-1] \ge 0}} \Big(extra[i, i+k-1]\Big)^3 + C(i+k)$$

$$C(i) = 0, \quad \text{if } extra[i, n] \ge 0$$

$$O(nW)$$

1: **procedure** PRINTING-NEATLY(n)
2: **for** $i \leftarrow n \downarrow 1$ **do**
3: **if** $\mathit{extra}[i, n] \geq 0$ **then** ▷ put w_i through w_n on a line
4: $C[i] \leftarrow 0$
6: **else**
7: $$C(i) = \min_{\substack{0 < k \leq (n-i+1) \\ \mathit{extra}[i, i+k-1] \geq 0}} \Big(\mathit{extra}[i, i+k-1]\Big)^3 + C(i+k)$$

1: **procedure** PRINTING-NEATLY(n)
2: **for** $i \leftarrow n \downarrow 1$ **do**
3: **if** $\mathit{extra}[i, n] \geq 0$ **then** ▷ put w_i through w_n on a line
4: $C[i] \leftarrow 0$
5: $B[i] \leftarrow n$
6: **else**
7: $C(i) = \min_{\substack{0 < k \leq (n-i+1) \\ \mathit{extra}[i, i+k-1] \geq 0}} \left(\mathit{extra}[i, i+k-1]\right)^3 + C(i+k)$
8: $B[i] = i + k$ ▷ line break

1: **procedure** PRINTING-NEATLY(n)
2: **for** $i \leftarrow n \downarrow 1$ **do**
3: **if** $\mathit{extra}[i, n] \geq 0$ **then** ▷ put w_i through w_n on a line
4: $C[i] \leftarrow 0$
5: $B[i] \leftarrow n$
6: **else**
7: $C(i) = \min_{\substack{0 < k \leq (n-i+1) \\ \mathit{extra}[i, i+k-1] \geq 0}} \left(\mathit{extra}[i, i+k-1]\right)^3 + C(i+k)$
8: $B[i] = i + k$ ▷ line break

$$B[1],$$

1: **procedure** PRINTING-NEATLY(n)
2: **for** $i \leftarrow n \downarrow 1$ **do**
3: **if** $\mathit{extra}[i, n] \geq 0$ **then** ▷ put w_i through w_n on a line
4: $C[i] \leftarrow 0$
5: $B[i] \leftarrow n$
6: **else**
7: $C(i) = \min_{\substack{0 < k \leq (n-i+1) \\ \mathit{extra}[i, i+k-1] \geq 0}} \left(\mathit{extra}[i, i+k-1]\right)^3 + C(i+k)$
8: $B[i] = i + k$ ▷ line break

$$B[1], \quad B[B[1]+1], \quad \cdots$$

A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

A set of n words of lengths $l_1, l_2, \cdots, l_n$

A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

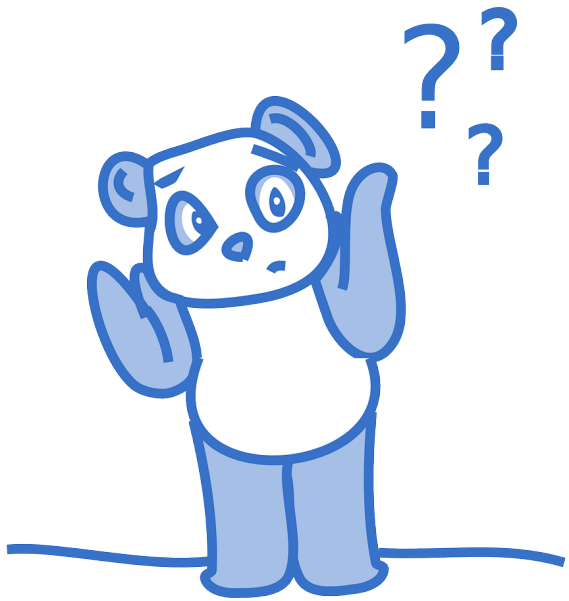
A set of n words of lengths $l_1, l_2, \cdots, l_n$

Greedy? *DP?* *NP-hard?*

A sequence of n words of lengths $l_1, l_2, \cdots, l_n$

A set of n words of lengths $l_1, l_2, \cdots, l_n$

Greedy? *DP?* *NP-hard?*



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 $A[1 \cdots i]$?

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

$$L(n)$$

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

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

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

$$L(n)$$

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

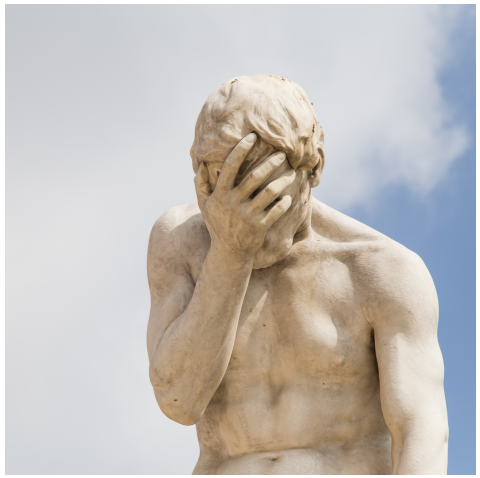
$$L(i) = \max\Big(\underbrace{L(i-1)}_{\text{NO}}, \underbrace{1 + \max_{j<i \,\wedge\, A[j] \le A[i]} L(j)}_{\text{YES}} \Big)$$

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

$L(n)$

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

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



$L(i)$: 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)

$$O(n \log n)$$

Longest Increasing Subsequence (Problem 15.4-6)

$$O(n \log n)$$

Longest Increasing Subsequence (Problem 15.4-6)

$$O(n \log n)$$

Answer by Eugene Yarovoi @ Quora

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

1: **procedure** *LIS*(n)
2: $E[l] \leftarrow \emptyset, \quad \forall 1 \leq i \leq n$
3: **for** $i \leftarrow 1 \uparrow n$ **do**
4: $\forall 1 \leq l \leq i : E[l] \leftarrow$
5: $\{\textit{all inc. subseq. of length } l\}$
6: **return**

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

1: **procedure** *LIS*(n)
2: $E[l] \leftarrow \emptyset, \quad \forall 1 \le i \le n$
3: **for** $i \leftarrow 1 \uparrow n$ **do**
4: $\forall 1 \le l \le i : E[l] \leftarrow$
5: $\{\textit{all inc. subseq. of length } l\}$
6: **return** $\max \{l \mid E[l] \neq \emptyset\}$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

1: **procedure** *LIS*(n)
2: $E[l] \leftarrow \emptyset, \quad \forall 1 \le i \le n$
3: **for** $i \leftarrow 1 \uparrow n$ **do**
4: $\forall 1 \le l \le i : E[l] \leftarrow$ ▷ *By extending each shorter subseq.*
5: $\{\textit{all inc. subseq. of length } l\}$
6: **return** $\max\{l \mid E[l] \neq \emptyset\}$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$$\langle 3 \quad 9 \quad 11 \quad 13 \rangle$$

$$\langle 4 \quad 6 \quad 10 \quad 15 \rangle$$

$$\langle 2 \quad 5 \quad 12 \quad 18 \rangle$$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$$\langle 3 \quad 9 \quad 11 \quad 13 \rangle$$

$$\langle 4 \quad 6 \quad 10 \quad 15 \rangle$$

$$\langle 2 \quad 5 \quad 12 \quad 18 \rangle$$

$\left|E[l]\right| = 1$: *the one with the smallest ending number*

1: **procedure** *LIS*(n)

2: $E[l] \leftarrow \emptyset, \quad \forall 1 \leq i \leq n$

3: **for** $i \leftarrow 1 \uparrow n$ **do**

4: $\forall 1 \leq l \leq i : E[l] \leftarrow$ ▷ *By extending each shorter subseq.*

5: *the inc. subseq. of length* l *with the smallest ending number*

6: ***return*** $\max\{l \mid E[l] \neq \emptyset\}$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$$\left|E[l]\right| = 1 : \textit{the one with the smallest ending number}$$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$$\left|E[l]\right| = 1 : \textit{the one with the smallest ending number}$$

$$\langle 2 \quad 8 \quad 15 \rangle$$

$$\langle 4 \quad 6 \quad 11 \quad 13 \rangle$$

$$\langle 3 \quad 9 \quad 11 \quad 12 \quad 13 \rangle$$

$$E[l] = \{\textit{all increasing subsequences of length } l\}$$

$$\left|E[l]\right| = 1 : \textit{the one with the smallest ending number}$$

$$\langle 2 \quad 8 \quad 15 \rangle$$

$$\langle 4 \quad 6 \quad 11 \quad 13 \rangle$$

$$\langle 3 \quad 9 \quad 11 \quad 12 \quad 13 \rangle$$

$$\forall i < j : \textit{the ending number of } E[i] < \textit{the ending number of } E[j]$$

1: **procedure** $LIS(n)$
2: $E[l] \leftarrow \emptyset, \quad \forall 1 \leq i \leq n$
3: **for** $i \leftarrow 1 \uparrow n$ **do**
4: $\forall 1 \leq l \leq i : E[l] \leftarrow$ ▷ *By extending each shorter subseq.*
5: *the inc. subseq. of length* l *with the smallest ending number*
6: ***return*** $\max\{l \mid E[l] \neq \emptyset\}$

1: **procedure** $LIS(n)$
2: $E[l] \leftarrow \emptyset, \quad \forall 1 \leq i \leq n$
3: **for** $i \leftarrow 1 \uparrow n$ **do**
4: $\forall 1 \leq l \leq i : E[l] \leftarrow$ ▷ *Extending only one subseq using binary search*
5: *the inc. subseq. of length* l *with the smallest ending number*
6: **return** $\max\{l \mid E[l] \neq \emptyset\}$

Extending

Extending

by looking at only the ending number of the inc. subseq

Extending

by looking at only the ending number of the inc. subseq

1: **procedure** *LIS*(n)
2: $E[l] \leftarrow$ $\forall 1 \leq i \leq n$
3: **for** $i \leftarrow 1 \uparrow n$ **do**
4: $E[l] \leftarrow$ ▷ *Extending only one subseq using binary search*
5: *the smallest ending number for the inc. subseq. of length* l
6: **return**

Extending

by looking at only the ending number of the inc. subseq

1: **procedure** *LIS*(n)
2: $E[l] \leftarrow \infty \quad \forall 1 \leq i \leq n$
3: **for** $i \leftarrow 1 \uparrow n$ **do**
4: $E[l] \leftarrow$ ▷ *Extending only one subseq using binary search*
5: *the smallest ending number for the inc. subseq. of length* l
6: **return**

Extending

by looking at only the ending number of the inc. subseq

1: **procedure** $LIS(n)$
2: $E[l] \leftarrow \infty \quad \forall 1 \le i \le n$
3: **for** $i \leftarrow 1 \uparrow n$ **do**
4: $E[l] \leftarrow$ ▷ *Extending only one subseq using binary search*
5: *the smallest ending number for the inc. subseq. of length* l
6: **return** $\max\{l \mid E[l] < \infty\}$

Matrix-chain Multiplication

$m[i, j]$: min cost to compute the matrix $A_i \cdots A_j$

$$m[1, n]$$

$m[i, j]$: min cost to compute the matrix $A_i \cdots A_j$

$$m[1, n]$$

Where is the last parentheses?

$m[i,j]$: min cost to compute the matrix $A_i \cdots A_j$

$$m[1,n]$$

Where is the last parentheses?

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

$m[i,j]$: min cost to compute the matrix $A_i \cdots A_j$

$$m[1,n]$$

Where is the last parentheses?

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

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

$m[i,j]$: min cost to compute the matrix $A_i \cdots A_j$

$$m[1,n]$$

Where is the last parentheses?

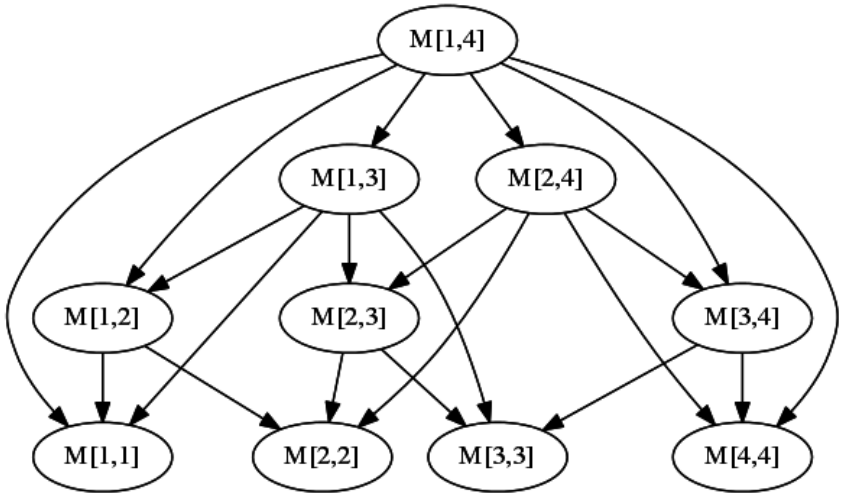
$$m[i,j] = \min_{i \le k < j} \left(m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \right)$$

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

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

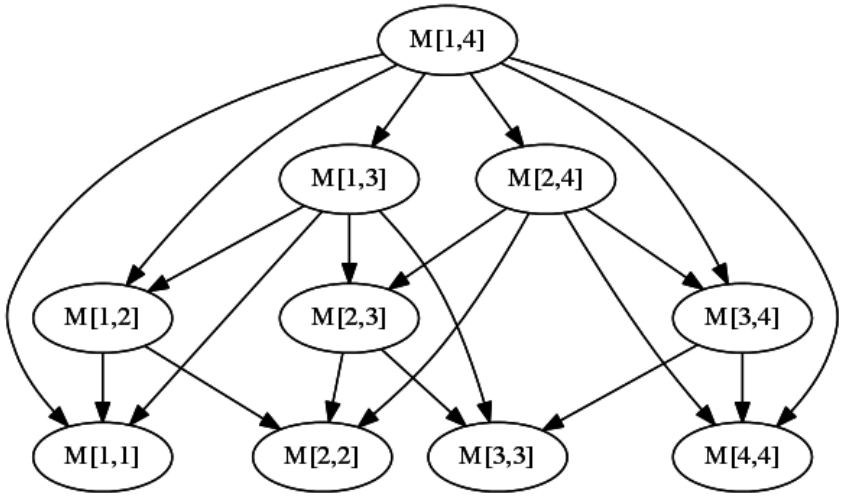
Subproblem Graph for Matrix-chain Multiplication (Problem 15.2-4)

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



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

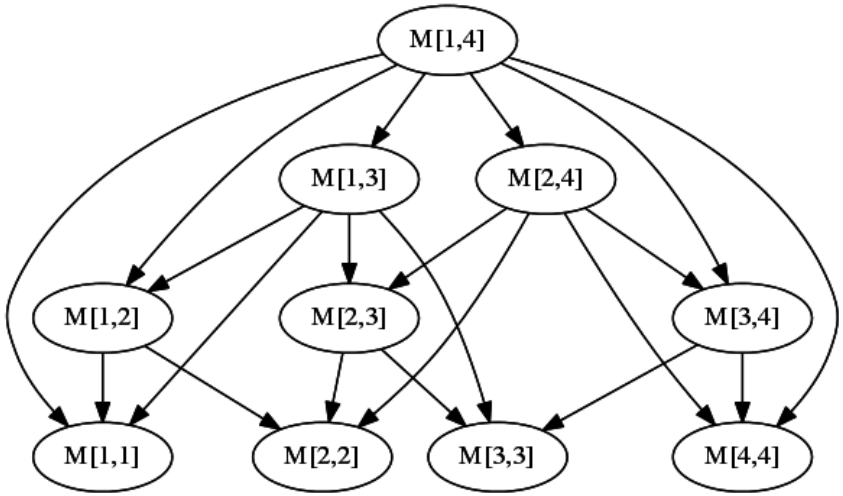
$$m[i,j] = \min_{i \le k < j} \left(m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \right)$$



$$\left| \{(i,j) | 1 \le i \le j \le n\} \right| = \frac{n(n+1)}{2}$$

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

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



$$\Big| \{(i,j) | 1 \le i \le j \le n\} \Big| = \frac{n(n+1)}{2}$$

$$\sum_{1 \le i \le j \le n} 2(j-i) \overset{k \triangleq j-i}{=} \sum_{0 \le k \le n-1} 2k(n-k)$$

Construct-Optimal-BST(root) (Problem 15.5-1)

$$e[i,j] = \begin{cases} q_{i-1} & \text{if } j = i-1, \\ \min\limits_{i \le r \le j} \{e[i,r-1] + e[r+1,j] + w(i,j)\} & \text{if } i \le j. \end{cases}$$

CONSTRUCT-OPTIMAL-BST(root) (Problem 15.5-1)

$$e[i,j] = \begin{cases} q_{i-1} & \text{if } j = i-1, \\ \min\limits_{i \le r \le j} \{e[i,r-1] + e[r+1,j] + w(i,j)\} & \text{if } i \le j. \end{cases}$$

1: **procedure** CONSTRUCT-OPTIMAL-BST($root$, i, j)
2: **if** $j = i-1$ **then**
3: **return** node with d_{i-1}
4: $r \leftarrow$ node with key $k_{root[i,j]}$
5: $r.left \leftarrow$ CONSTRUCT-OPTIMAL-BST$(root, i, root[i,j]-1)$
6: $r.right \leftarrow$ CONSTRUCT-OPTIMAL-BST$(root, root[i,j]+1, j)$
7: **return** r

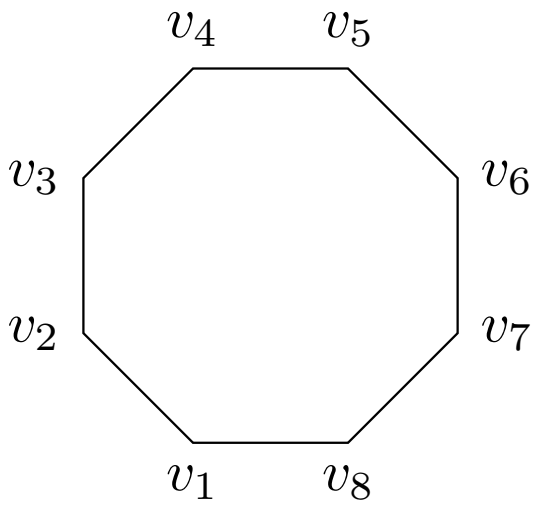
CONSTRUCT-OPTIMAL-BST(root) (Problem 15.5-1)

$$e[i,j] = \begin{cases} q_{i-1} & \text{if } j = i-1, \\ \min\limits_{i \le r \le j} \{e[i,r-1] + e[r+1,j] + w(i,j)\} & \text{if } i \le j. \end{cases}$$

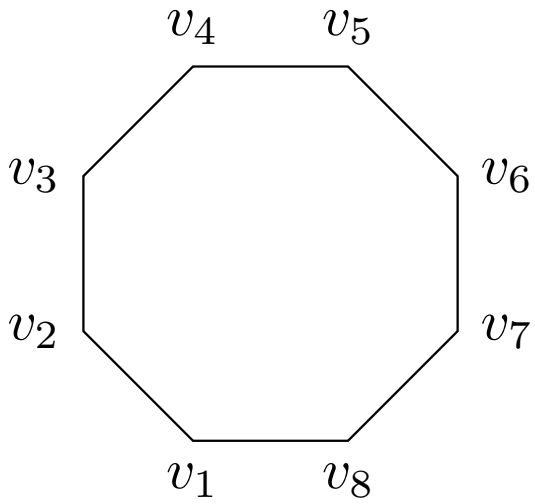
1: **procedure** CONSTRUCT-OPTIMAL-BST($root$, i, j)
2: **if** $j = i-1$ **then**
3: **return** node with d_{i-1}
4: $r \leftarrow$ node with key $k_{root[i,j]}$
5: $r.left \leftarrow$ CONSTRUCT-OPTIMAL-BST$(root, i, root[i,j]-1)$
6: $r.right \leftarrow$ CONSTRUCT-OPTIMAL-BST$(root, root[i,j]+1, j)$
7: **return** r

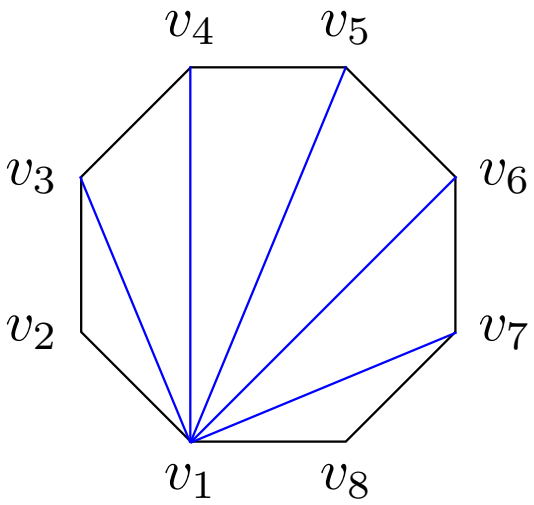
CONSTRUCT-OPTIMAL-BST$(root, 1, n)$

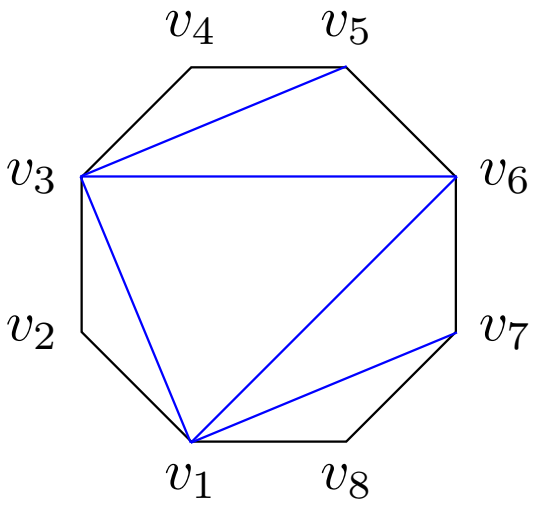
Minimum Weight Triangulation

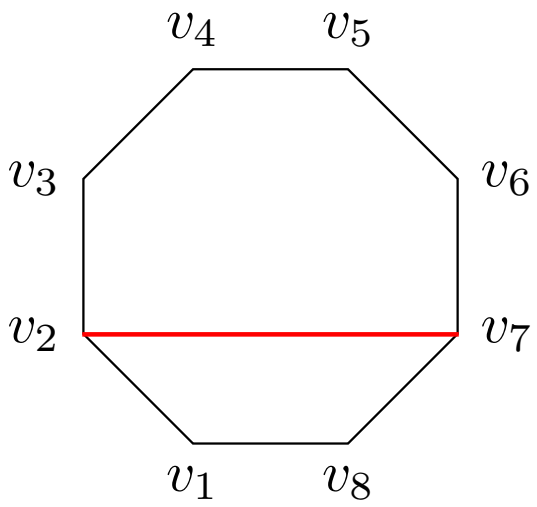


Minimum Weight Triangulation

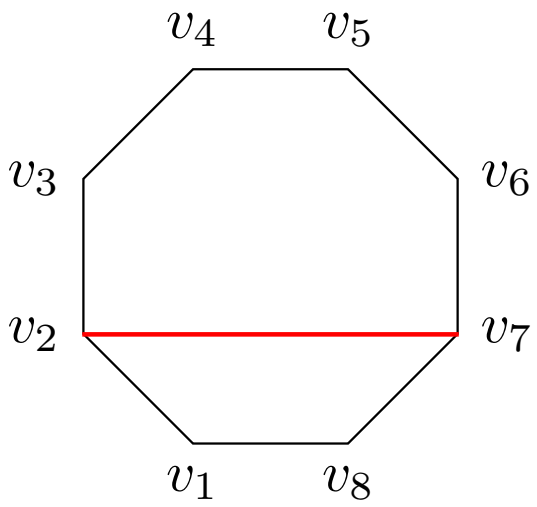






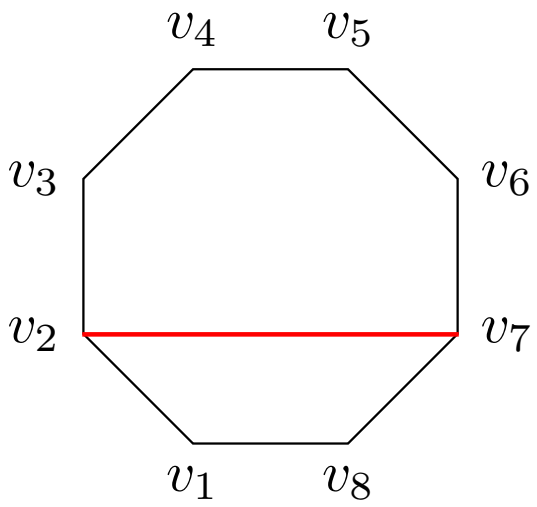


$T(i,j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), *clockwise*



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

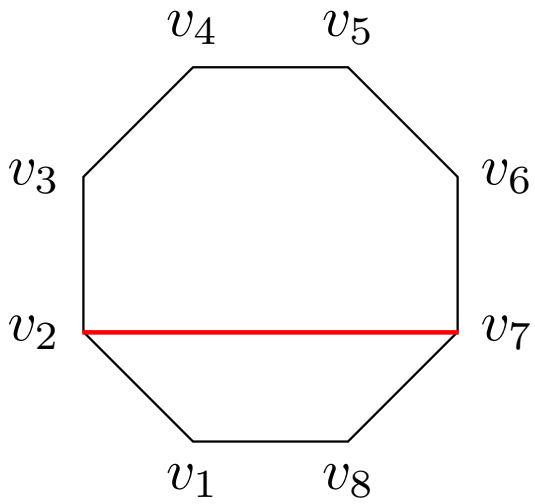
$$T(i,j) = \min_{\substack{i \le k < l-1 \le j \\ (k,l) \neq (i,j)}} \Big(T[k,l] + T[?,?] + d_{ij} \Big)$$

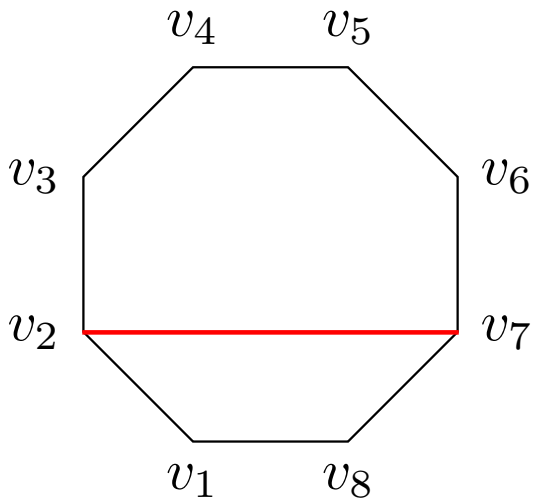


$T(i,j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), *clockwise*

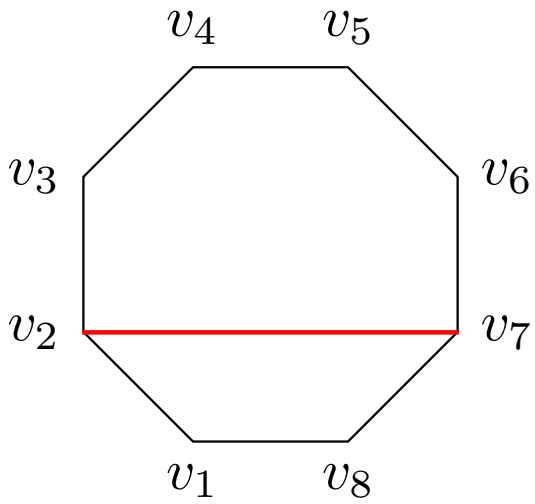
$$T(i,j) = \min_{\substack{i \le k < l-1 \le j \\ (k,l) \neq (i,j)}} \Big(T[k,l] + T[?,?] + d_{ij} \Big)$$

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


v_4
v_5
v_3
v_6
v_2
v_7
v_1
v_8

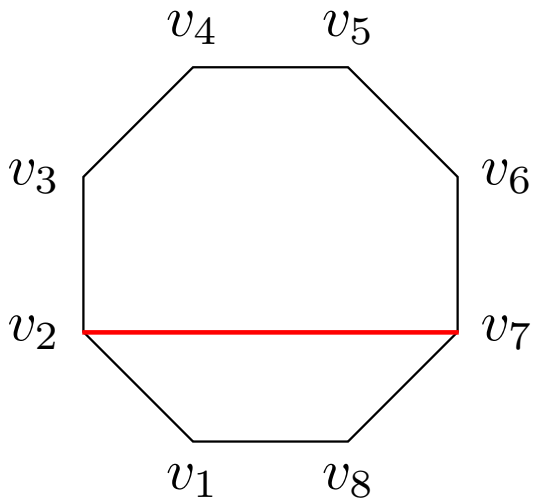


Which vertex k to pair with (i, j)?



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

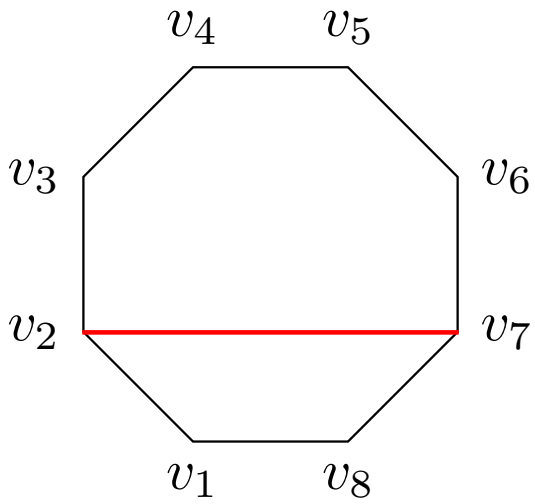
$T(i, j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), $i < j$



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

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

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

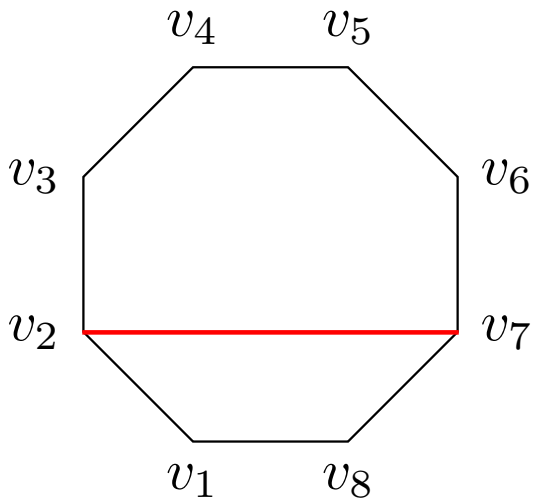


Which vertex k to pair with (i, j)?

$T(i, j)$: 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)?

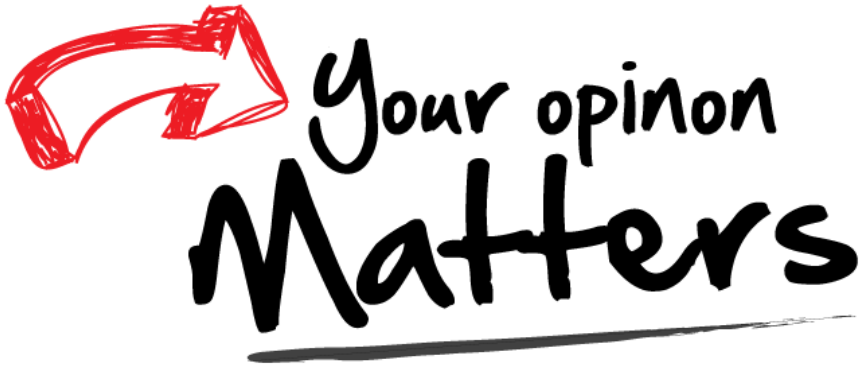
$T(i, j)$: min cost of triangulating $v_i \cdots v_j$ (with $v_j - v_i$), $i < j$

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

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

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


THANK YOU



Office 302

Mailbox: H016

hfwei@nju.edu.cn