

# CS 331

## DESIGN AND ANALYSIS OF ALGORITHMS

### DYNAMIC PROGRAMMING

Dr. Daisy Tang

# Dynamic Programming

2

## □ Idea:

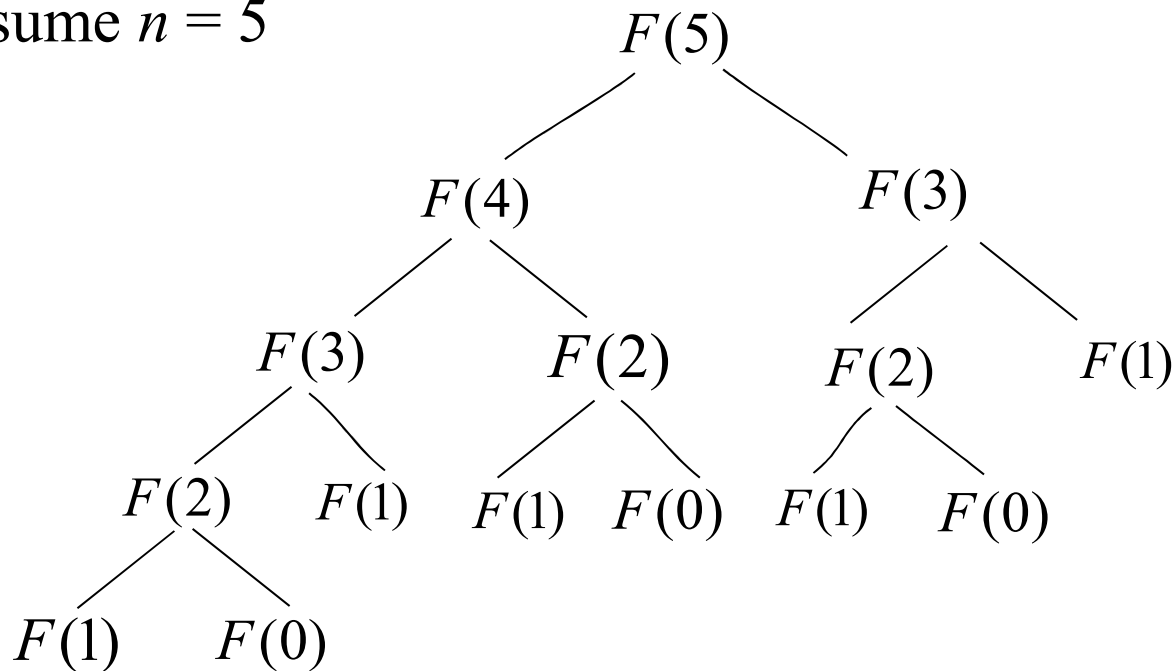
- ▣ Problems can be divided into stages
- ▣ Solution is a sequence of decisions, and the decision at the current stage is based on the results of the previous stage

# Example: Fibonacci Numbers

3

$$\begin{cases} F(0) = 0 \\ F(1) = 1 \\ F(n) = F(n-1) + F(n-2) \quad \forall n \geq 2 \end{cases}$$

Assume  $n = 5$



# If We Use Divide-and-Conquer

4


- Divide the problem into subproblems. Solve each subproblem independently and then combine the solutions.
- For example,  $F(3)$  is first calculated in the left subtree and it will be calculated again in the right subtree
  - ▣ Subproblems are solved independently

# But Dynamic Programming

5

- Works on phases

0 1 1 2 3 5 8 . . .



Based on the results in previous phase, we can calculate result for the current phase

Can be solved in a for-loop

$$F[0] = 0$$

$$F[1] = 1$$

For  $i \leftarrow 2$  to  $n$

$$F[i] = F[i-1] + F[i-2];$$

# Greedy vs. Dynamic Programming

6

- Consider shortest path problem
  - Suppose we wish to find a shortest path from vertex  $i$  to vertex  $j$ . Let  $A_i$  be the vertices adjacent from vertex  $i$ . Which of the vertices in  $A_i$  should be the second vertex on the path? There is no way to make a decision at this time and guarantee that future decisions leading to an optimal sequence can be made.
  - If on the other hand we wish to find a shortest path from vertex  $i$  to all other vertices in  $G$ , then at each step, a correct decision can be made based on the **principle of optimality**.

# Greedy vs. Dynamic Programming

7

## □ Greedy:

- For problems that an optimal solution can be found by making the decisions one at a time and never making an erroneous decision [e.g., Knapsack]
- Consider only 1 decision sequence

## □ Dynamic programming:

- For problems that it is not possible to make stepwise decisions based only on local information
- Consider many decision sequences
- Try all possibilities but eliminates impossible cases based on the principle of optimality

# Principle of Optimality

8

- ***Principle of optimality:*** An optimal sequence of decisions has the property that whatever the initial decision is, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision
  - → eliminates “not optimal” subsequences



# Examples

9

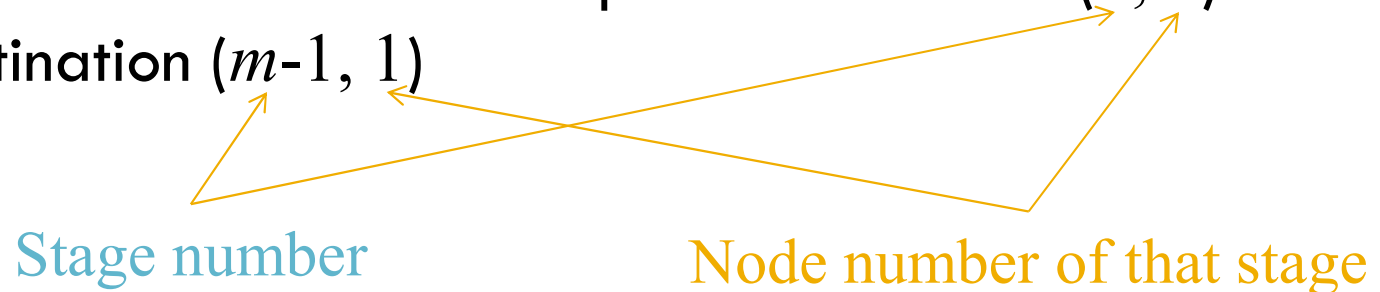
- The shortest path problem satisfies the Principle of Optimality
  - ▣ Subpath of a shortest path is also a shortest path
  
- The longest path problem does not satisfy the Principle of Optimality
  - ▣ Consider a ring graph
  - ▣ The subpath of a longest path might not be a longest path

# 1. Multistage Single-Source Single-Destination Shortest Path Problem

10

## □ The Problem:

- Given:  $m$  columns (or stages) of  $n$  nodes each, a source node, a destination node. Assume edges only exist between stages.
- Goal: to find the shortest path from source  $(0, 1)$  to the destination  $(m-1, 1)$



# Straightforward vs. Dynamic Programming

11

- Straightforward way: Try all possible paths


- ▣ Complexity:  $O(n^{m-2})$

- Dynamic programming:

Let  $M(i, j)$  = length of the shortest path from the  
source  $(0,1)$  to  $(i, j)$

$c(i, k, j)$  = distance from  $(i, k)$  to  $(i+1, j)$

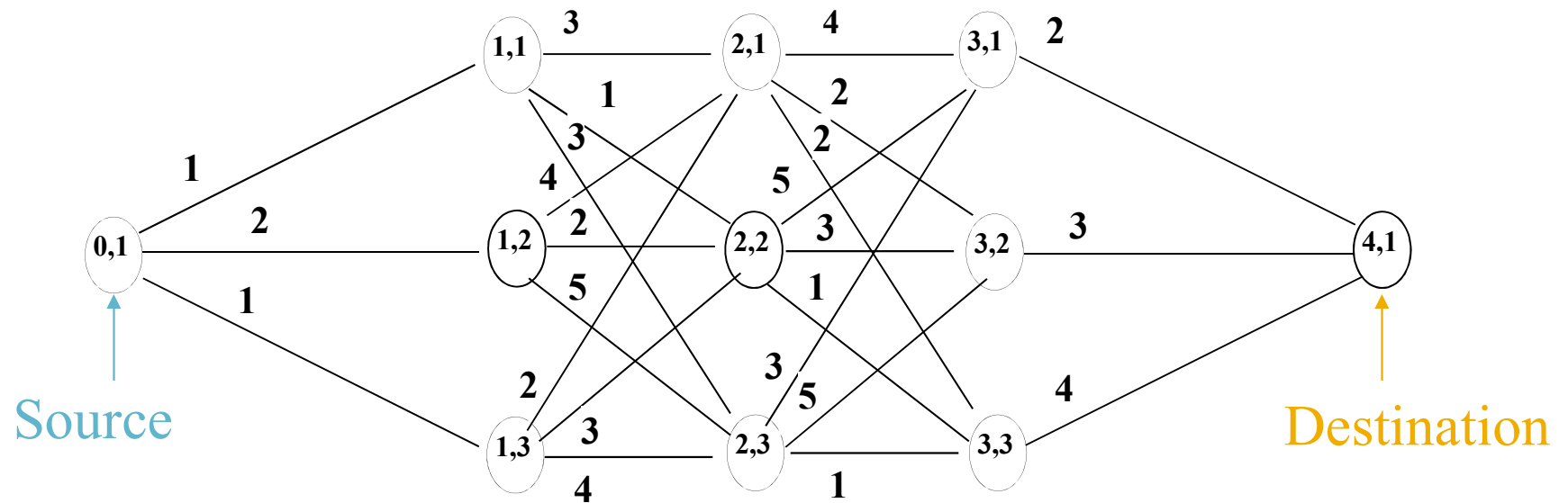
want to find  $M(m-1, 1)$   Shortest path from  
source  $(0,1)$  to destination  $(m-1,1)$

Define functional equation:  Define the relation between current stage  
and the result of previous stage

$$\begin{cases} M(i, j) = \min_{k=1}^n \{M(i-1, k) + c(i-1, k, j)\} \\ M(0,1) = 0 \end{cases}$$

# Example

12



$$\begin{aligned}
 M(2,1) &= \min \left\{ \overbrace{1+3}^1, \overbrace{2+4}^2, \overbrace{1+2}^3 \right\} = 3_3 \\
 M(2,2) &= \min \left\{ \overbrace{1+1}^1, \overbrace{2+2}^2, \overbrace{1+3}^3 \right\} = 2_1 \\
 M(2,3) &= \min \left\{ \overbrace{1+3}^1, \overbrace{2+5}^2, \overbrace{1+4}^3 \right\} = 4_1 \\
 M(3,1) &= \min \left\{ \overbrace{3+4}^1, \overbrace{2+5}^2, \overbrace{4+3}^3 \right\} = 7_{1,2,3} \\
 M(3,2) &= \min \left\{ \overbrace{3+2}^1, \overbrace{2+3}^2, \overbrace{4+5}^3 \right\} = 5_{1,2} \\
 M(3,3) &= \min \left\{ \overbrace{3+2}^1, \overbrace{2+1}^2, \overbrace{4+1}^3 \right\} = 3_2 \\
 M(4,1) &= \min \left\{ \overbrace{7+2}^1, \overbrace{5+3}^2, \overbrace{3+4}^3 \right\} = 7_3
 \end{aligned}$$

Remember the previous node of this shortest path

Can be used for constructing the shortest path at the end

The diagram illustrates the backtracking process for finding the shortest path. Arrows point from the minimum value in each row to the corresponding row above it, indicating the previous node in the path. For example, from  $M(4,1) = 7_3$ , an arrow points to  $M(3,3) = 3_2$ , and from  $M(3,3) = 3_2$ , an arrow points to  $M(2,2) = 2_1$ . A dashed purple arrow points from the text 'Can be used for constructing the shortest path at the end' to the sequence of nodes  $(0,1) \rightarrow (1,1) \rightarrow (2,2) \rightarrow (3,3) \rightarrow (4,1)$ .

The shortest path is:  $(0,1) \rightarrow (1,1) \rightarrow (2,2) \rightarrow (3,3) \rightarrow (4,1)$

$M$ :

$node(j) \setminus stage(i)$	0	1	2	3	4
1	0	1	3	7	7
2		2	2	5	
3		1	4	3	

Algorithm:

assign initially  $M(0,1)=0$ ,  $M(1,1) = 1$ ,  $M(1,2) = 2$ ,  $M(1,3)=1$ ;

for  $i = 2$  to  $(m - 2)$  // stages

for  $j = 1$  to  $n$  // nodes

$$M(i, j) = \min_{k=1}^n \{M(i-1, k) + c(i-1, k, j)\}$$

calculate and return  $M(m - 1, 1)$ ;

# Time Complexity

15

There are  $O(mn)$   $M(i, j)$ 's to compute,  
each  $M(i, j)$  takes  $O(n)$  computation.  
So, total complexity is  $O(mn^2)$ .

total # of nodes in the graph



Let  $N = m \times n$  nodes

max # of stages

max # of nodes in each stage

$$mn^2 = mn \times n = N \times n$$

Note: if use the Dijkstra's algorithm, the complexity is  $O(N^2)$

# Revisit Dynamic Programming

16

- The problem can be divided into stages.
- Need to make decisions at each stage.
- The decision for the current stage (based on the results of the previous stage) is independent from that of the previous stages.

e.g. node 3,1 of the multistage shortest path problem, it is calculated from  $m(2,1)$ ,  $m(2,2)$ , and  $m(2,3)$  but it is independent from how node 2,1 (2,2 or 2,3) makes its decision.

Note that we find shortest paths from the source to all three nodes 2,1 and 2,2 and 2,3, but some of them may not be used on the shortest path at all.

- But every decision must be optimal, so that the overall result is optimal. This is called “**Principle of Optimality**”.



# Problem 2: All Pairs Shortest Paths

17

*The problem:*

Given  $G = (V, E)$  a directed graph with  $n$  nodes

$Cost$ , a cost adjacency matrix for  $G$ :

$$Cost_{ii} = 0$$

$$Cost_{ij} = \begin{array}{ll} \text{cost of } \langle i, j \rangle & \text{if } \langle i, j \rangle \in E(G) \\ \infty & \text{otherwise} \end{array}$$

Goal: to determine a matrix  $A$ ,

s.t.  $A_{ij}$  = the length of the shortest path from  $i$  to  $j$

$$\forall 1 \leq i, j \leq n$$

*Use Dijkstra's algorithm, call it for  $n$  times:*

Time complexity is  $O(n^3)$

*The Dynamic Programming way:*

Let  $A_{ij}^k$  = the length of the shortest path  
from  $i$  to  $j$  going through no vertex of  
index greater than  $k$

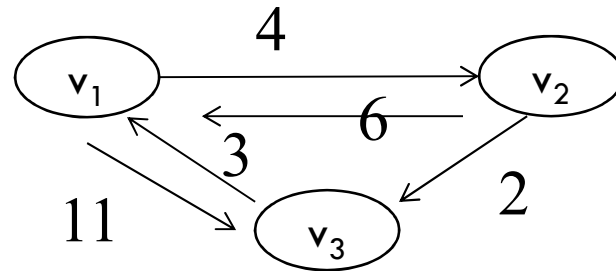
Want to find  $A_{ij}^n$

Observe that  $A_{ij}^0 = \text{Cost}$

Define functional equation:

$$A_{ij}^k = \min \{ A_{ij}^{k-1}, A_{ik}^{k-1} + A_{kj}^{k-1} \}$$

Example:



$$A^0 = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix}$$

$$A_{12}^1 = \min\{A_{12}^0, A_{11}^0 + A_{12}^0\} = 4$$

$$A_{13}^1 = \min\{A_{13}^0, A_{11}^0 + A_{13}^0\} = 11$$

$$A_{21}^1 = \min\{A_{21}^0, A_{21}^0 + A_{11}^0\} = 6$$

$$A_{23}^1 = \min\{A_{23}^0, A_{21}^0 + A_{13}^0\} = 2$$

$$A_{31}^1 = \min\{A_{31}^0, A_{31}^0 + A_{11}^0\} = 3$$

$$A_{32}^1 = \min\{A_{32}^0, A_{31}^0 + A_{12}^0\} = 7$$

$$A^1 = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \underline{7} & 0 \end{bmatrix}$$

$$A^2 = \begin{bmatrix} 0 & 4 & \underline{6} \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

$$A_{12}^2 = \min \{A_{12}^1, A_{12}^1 + A_{22}^1\} = 4$$

$$A_{13}^2 = \min \{A_{13}^1, A_{12}^1 + A_{23}^1\} = 6$$

$$A_{21}^2 = \min \{A_{21}^1, A_{22}^1 + A_{21}^1\} = 6$$

$$A_{23}^2 = \min \{A_{23}^1, A_{22}^1 + A_{23}^1\} = 2$$

$$A_{31}^2 = \min \{A_{31}^1, A_{32}^1 + A_{21}^1\} = 3$$

$$A_{32}^2 = \min \{A_{32}^1, A_{32}^1 + A_{22}^1\} = 7$$

$$A_{12}^3 = \min \{A_{12}^2, A_{13}^2 + A_{32}^2\} = 4$$

$$A_{13}^3 = \min \{A_{13}^2, A_{13}^2 + A_{33}^2\} = 6$$

$$A_{21}^3 = \min \{A_{21}^2, A_{23}^2 + A_{31}^2\} = 5$$

$$A_{23}^3 = \min \{A_{23}^2, A_{23}^2 + A_{33}^2\} = 2$$

$$A_{31}^3 = \min \{A_{31}^2, A_{33}^2 + A_{31}^2\} = 3$$

$$A_{32}^3 = \min \{A_{32}^2, A_{33}^2 + A_{32}^2\} = 7$$

$$A^3 = \begin{bmatrix} 0 & 4 & 6 \\ \underline{5} & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

*Algorithm:*

```
procedure AllPairs ( $n, Cost, A$ ) //  $n, Cost$  are input,  $A$  is output
{
    for  $i \leftarrow 1$  to  $n$  do
        for  $j \leftarrow 1$  to  $n$  do
             $A(i,j) \leftarrow Cost(i,j)$ ;
    for  $k \leftarrow 1$  to  $n$  do
        for  $i \leftarrow 1$  to  $n$  do
            for  $j \leftarrow 1$  to  $n$  do
                 $A(i,j) \leftarrow \min \{A(i,j), A(i,k) + A(k,j)\}$ ;
}
```

There are  $O(n)$   $A^k$ 's to compute, each takes  $O(n^2)$  computation, total complexity of AllPair is  $O(n^3)$ .

*In order to print shortest paths for all pairs of vertices, we modify the algorithm:*

```

procedure AllPaths ( $n$ ,  $Cost$ ,  $A$ ,  $P$ )
//  $n$ ,  $Cost$  are input,  $A$  and  $P$  are output
//  $P(i,j)$  contains the intermediate node from  $i$  to  $j$ 
{
    for  $i = 1$  to  $n$  do
        for  $j = 1$  to  $n$  do
            [  $A(i,j) = Cost(i,j)$ ;
               $P(i,j) = j$ ;
            ]
        for  $k = 1$  to  $n$  do
            for  $i = 1$  to  $n$  do
                for  $j = 1$  to  $n$  do
                    if ( $A(i,j) > A(i,k) + A(k,j)$ )
                        then [  $P(i,j) = k$ ;
                               $A(i,j) = A(i,k) + A(k,j)$  ;
                            ]
        for  $i = 1$  to  $n$  do
            for  $j = 1$  to  $n$  do
                [ print( $i$ );
                  Call PrintOut( $P$ ,  $i$ ,  $j$ );
                ]
    }

```

```

procedure PrintOut ( $P$ ,  $i$ ,  $j$ )
{
    if  $P(i,j) = j$ 
        then print( $j$ );
    else [ let  $k = P(i,j)$ ;
           call PrintOut( $P$ ,  $i$ ,  $k$ );
           call PrintOut( $P$ ,  $k$ ,  $j$ );
         ]
}

```

# In-Class Exercise #1

23

- Solve the all pairs shortest path problem for the graph with the following weight matrix. Find A and P and shortest paths for all pairs of vertices. Show steps.

$$A^0 = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & \infty & 1 & \infty \\ \infty & 0 & 1 & 1 \\ 1 & 1 & 0 & 5 \\ \infty & 1 & 5 & 0 \end{bmatrix} \end{matrix}$$

# Problem 3: Chained Matrix Multiplication

24

*The problem:*

Given  $M_1 \times M_2 \times \dots \times M_r$  ( $r$  matrices)  
 $[d_0 \times d_1] \quad [d_1 \times d_2] \quad \dots \quad [d_{r-1} \times d_r]$

Goal: find the minimum cost of multiplying  $r$  matrices



*The idea:*

Define

$C(i, j)$  = minimum cost of computing  $M_i \times M_{i+1} \times \dots \times M_j$

$$C(i, i) = 0 \quad \forall i$$

$C(i, i+1) = d_{i-1} \times d_i \times d_{i+1}$  (the total # of elementary multiplications )

Want to find  $C(1, r)$

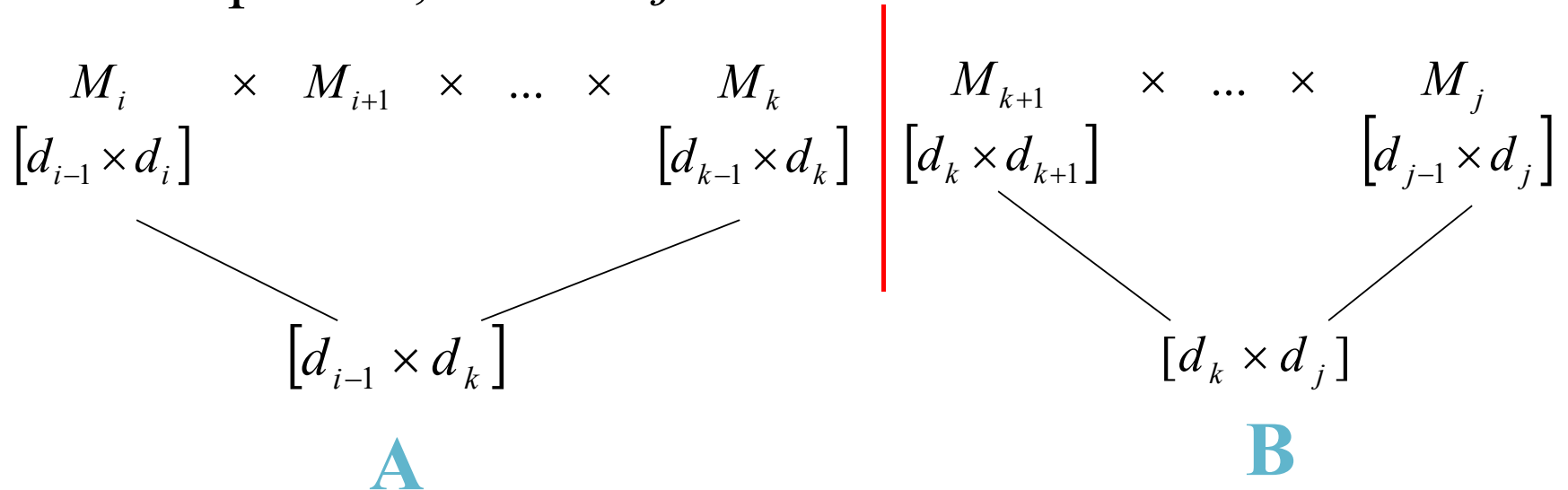
Consider matrices from  $i$  to  $j$  :

$$k = \begin{array}{ccccccccccc} M_i & \times & M_{i+1} & \times & M_{i+2} & \times & \dots & \dots & \times & M_{j-1} & \times & M_j \\ & & i & & i+1 & & i+2 & \dots & \dots & & & j-1 \end{array}$$

$k$ : divide-point,  $i \leq k \leq j-1$

We can divide the sequence of matrices from  $i$  to  $j$  into two parts with dividing point  $k$ , where  $i \leq k \leq j-1$  .

We divide it into two parts with  
divide-point  $k$ ,  $i \leq k \leq j - 1$



Define the functional equation:

$$C(i, j) = \min_{i \leq k \leq j-1} \left\{ C(i, k) + C(k+1, j) + \underbrace{d_{i-1} \times d_k \times d_j}_{\text{Cost of multiplying A and B}} \right\}$$

Cost of multiplying A and B

Example:

$$\begin{array}{ccccccc} M_1 & \times & M_2 & \times & M_3 & \times & M_4 \\ [3 \times 8] & & [8 \times 2] & & [2 \times 5] & & [5 \times 4] \end{array}$$

$$\text{size 1} \quad \leftarrow [C(i,i) = 0 \quad \forall i]$$

$$\text{size 2} \quad \leftarrow \left[ \begin{array}{l} C(1,2) = 3 \times 8 \times 2 = 48 \\ C(2,3) = 8 \times 2 \times 5 = 80 \\ C(3,4) = 2 \times 5 \times 4 = 40 \end{array} \right]$$

$$\text{size } 3 \leftarrow \begin{cases} C(1,3) = \min_{k=1,\underline{2}} \{ (C(1,1) + C(2,3) + 3 \times 8 \times 5), \underbrace{(C(1,2) + C(3,3) + 3 \times 2 \times 5)}_{\text{Smallest}} \} \\ \quad = 78 \\ C(2,4) = \min_{k=\underline{2},3} \{ \underbrace{(C(2,2) + C(3,4) + 8 \times 2 \times 4)}_{\text{Smallest}}, (C(2,3) + C(4,4) + 8 \times 5 \times 4) \} \\ \quad = 104 \end{cases}$$

$$\text{size } 4 \leftarrow C(1,4) = \min_{k=1,\underline{2},3} \left\{ \begin{array}{l} (C(1,1) + C(2,4) + 3 \times 8 \times 4), \\ \boxed{(C(1,2) + C(3,4) + 3 \times 2 \times 4)}, \\ (C(1,3) + C(4,4) + 3 \times 5 \times 4) \end{array} \right\} = 112$$

48 + 40 + 24

↓

It has the smallest cost when  $k = 2$ .

Remember this  $k$ -value and it can be used for obtaining the optimal solution. The optimal multiplication sequence for this instance is  $(M_1 \times M_2) \times (M_3 \times M_4)$ .

***Algorithm:***

```
procedure MinMult ( $r, D, P$ ) //  $r, D$  are inputs,  $P$  is output
{
    for  $i \leftarrow 1$  to  $r$  do
         $C(i, i) = 0$ 

    for  $d \leftarrow 1$  to  $(r-1)$  do
        for  $i \leftarrow 1$  to  $(r-d)$  do
            {
                 $j \leftarrow i + d$ ;
                 $C(i, j) = \min_{i \leq k \leq j-1} \{C(i, k) + C(k+1, j) + D(i-1) \times D(k) \times D(j)\};$ 
            }

     $P \leftarrow C(1, r);$ 
}
```

Analysis:

Matrix product of		# of $C(i,j)$ 's with this size	Cost of computing $C(i,j)$
size 2	<i>i.e.</i> $j = i + 1$	$r - 1$	1
size 3	<i>i.e.</i> $j = i + 2$	$r - 2$	2
...	...	...	...
size $r$	$j = i + (r - 1)$	1	$r - 1$

Total # of  $C(i, j)$ 's to compute is  $O(r^2)$

each  $C(i, j)$  takes  $O(r)$  time in the worst case

Total complexity is  $O(r^3)$

# Problem 4: 0/1 Knapsack

31

*The problem:*

Given  $n$  objects,  $p_1, p_2, \dots, p_n$  profit,

$w_1, w_2, \dots, w_n$  weight

$M$  = the capacity of the knapsack

Goal: find  $x_1, x_2, \dots, x_n$

s.t.  $\sum_{i=1}^n p_i x_i$  is maximized subject to  $\sum_{i=1}^n w_i x_i \leq M$

where  $x_i = 0$  or  $1$

*The straight forward way:*

Try all possibilities. The complexity is  $O(2^n)$ .

Example:  $n = 3$

$$(p_1, p_2, p_3) = (1, 2, 5)$$

$$(w_1, w_2, w_3) = (2, 3, 4)$$

$$M = 6$$

$x_1$	$x_2$	$x_3$	$\sum w_i x_i$	$\sum p_i x_i$
0	0	0	0	0
0	0	1	4	5
0	1	0	3	2
0	1	1	—	—
1	0	0	2	1
1	0	1	6	<u>6</u> $\leftarrow$ solution
1	1	0	5	3
1	1	1	—	—



*Does Greedy work?*

This problem can't be solved by Greedy ( $P_i / W_i$  ↓ order)

Counterexample:  $M = 6$ ,  $P = 5 \ 3 \ 3$

$W = 4 \ 3 \ 3$

Greedy  $X = (1, 0, 0)$ , total profit = 5.

Optimal  $Y = (0, 1, 1)$ , total profit = 6.

*The Dynamic Programming way:*

Let  $f_i(X)$  = max profit generated from  $x_1, x_2, \dots, x_i$   
subject to the capacity  $X$

want to find  $f_n(M)$

Define functional equation:

$$\begin{cases} f_i(X) = \max \left\{ \underbrace{f_{i-1}(X)}_{x_i = 0}, \underbrace{p_i + f_{i-1}(X - W_i)}_{x_i = 1} \right\} \\ f_0(X) = 0 \end{cases}$$

Example:  $n = 6$ ,

$$\begin{aligned} (p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6) &= (w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6) \\ &= (100 \quad 50 \quad 20 \quad 10 \quad 7 \quad 3) \end{aligned}$$

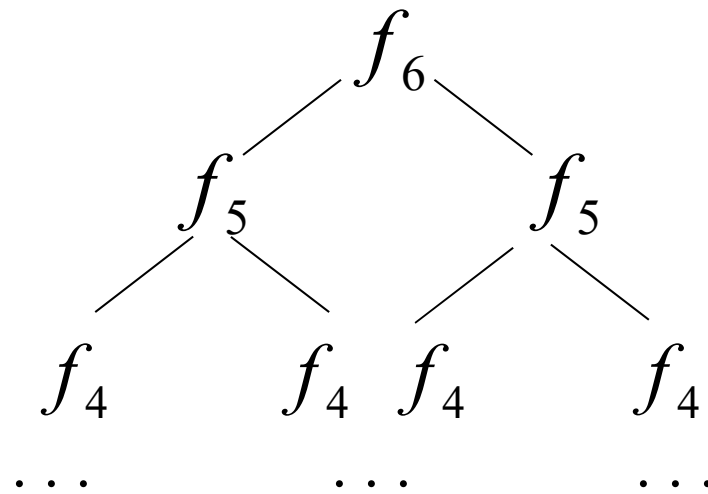
$$M = 165$$

To find  $f_6(165)$ :

$$\begin{aligned} f_6(165) &= \max \{f_5(165), f_5(162) + 3\} = \max \{160, 163\} = 163 \\ f_5(165) &= \max \{f_4(165), f_4(158) + 7\} = \max \{160, 157\} = 160 \\ f_4(165) &= \max \{f_3(165), f_3(155) + 10\} = \max \{150, 160\} = 160 \\ f_3(165) &= \max \{f_2(165), f_2(145) + 20\} = \max \{150, 120\} = 150 \\ f_2(165) &= \max \{f_1(165), f_1(115) + 50\} = \max \{100, 150\} = 150 \\ f_1(165) &= \max \{f_0(165), f_0(65) + 100\} = 100 \\ f_1(115) &= \max \{f_0(115), f_0(15) + 100\} = 100 \\ f_2(145) &= \max \{f_1(145), f_1(95) + 50\} = \max \{100, 50\} = 100 \\ &\dots \dots \\ f_5(162) &= \max \{f_4(162), f_4(155) + 7\} = \max \{160, 157\} = 160 \\ &\dots \dots \end{aligned}$$

The result:

$$(x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6) = (1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1)$$



Therefore, the complexity of 0/1 Knapsack is  $O(2^n)$

$\longleftrightarrow M+1 \longrightarrow$

	0	1	...	$j-w_i$	...	$j$	...	$M$
0	0	0	0	0	0	0	0	0
1								
...								
$i-1$				$f_{i-1}(j-w_i)$		$f_{i-1}(j)$		
$i$						$f_i(j)$		
...								
$n$								

$\longleftrightarrow$

$\longleftrightarrow$

$\longleftrightarrow$

$(M+1)(n+1)$  entries and constant time each  $\Rightarrow O(Mn)$

Example:  $M = 6$

	Object 1	Object 2	Object 3	Object 4
$p_i$	3	4	8	5
$w_i$	2	1	4	3

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3
2	0	4	4	7	7	7	7
3	0	4	4	7	8	12	12
4	0	4	4	7	9	12	12

*Max profit  
is 12*

*By tracing which entries lead to this max-profit solution,  
we can obtain the optimal solution - Object 1,2 and 4 (or 2 and 3).*

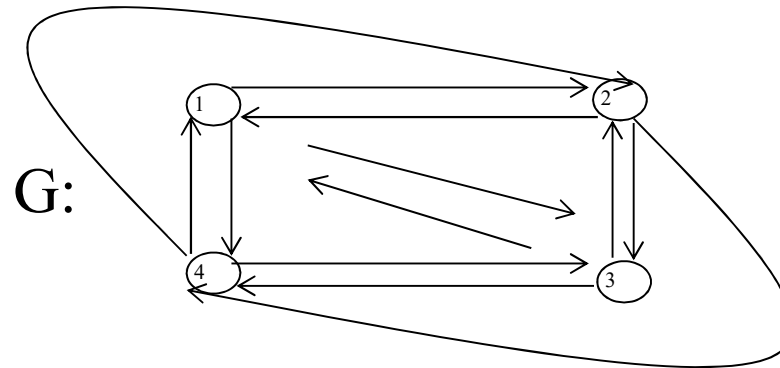
## Problem 5: Traveling Salesman Problem (TSP)

38

### □ *The problem:*

- Given a directed graph  $G = (V, E)$  with edge cost
- A tour (or Hamiltonian Circuit) of  $G$  is defined as directed cycle from a vertex to itself that passes through every other vertices exactly once.
- Goal: find a tour with minimum cost

Example:



$$C = \begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$$

Tour 1: [ 1 2 3 4 1 ] Cost = 10 + 9 + 12 + 8 = 39

Tour 2: [ 1 3 2 4 1 ] Cost = 15 + 13 + 10 + 8 = 46

### *The straightforward way:*

This is a permutation problem. There are  $(n-1)*(n-2)*\dots*1$  combinations of tours, so the complexity is  $O(n!)$ .

### *The Dynamic Programming way:*

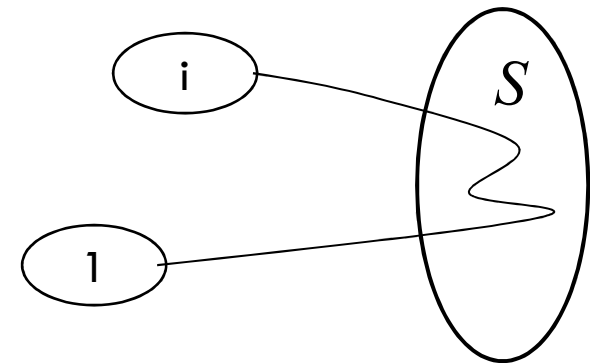
Assume starting vertex is 1, without loss of generality.

Let  $g(i,S)$  = length of the shortest path starting at vertex  $i$ ,  
going through all the vertices in  $S$  and  
terminating at vertex 1

Want to find  $g(1, V - \{1\})$

Define functional equation:

$$\begin{aligned} g(i,S) &= C_{i,1} && \text{if } S = \emptyset \\ g(i,S) &= C_{i,S} + C_{S,1} && \text{if } |S| = 1 \\ g(i,S) &= \min_{k \in S} \{ C_{i,k} + g(k, S - \{k\}) \} \end{aligned}$$





Use the above example:

$$\begin{aligned}
 g(1, \{2,3,4\}) &= \min_{k=2,3,4} \{ (C_{12} + g(2, \{3,4\})), (C_{13} + g(3, \{2,4\})), (C_{14} + g(4, \{2,3\})) \} \\
 &= \min_{k=2,3,4} \{ 10+25, 15+25, 20+23 \} = 35 \\
 g(2, \{3,4\}) &= \min_{k=3,4} \{ (C_{23} + g(3, \{4\})), (C_{24} + g(4, \{3\})) \} \\
 &= \min_{k=3,4} \{ 9 + (12 + 8), 10 + (9 + 6) \} = \underline{25} \\
 g(3, \{2,4\}) &= \min_{k=2,4} \{ (C_{32} + g(2, \{4\})), (C_{34} + g(4, \{2\})) \} \\
 &= \min_{k=2,4} \{ 13 + (10 + 8), 12 + (8 + 5) \} = \underline{25} \\
 g(4, \{2,3\}) &= \min_{k=2,3} \{ (C_{42} + g(2, \{3\})), (C_{43} + g(3, \{2\})) \} \\
 &= \min_{k=2,3} \{ 8 + (9 + 6), 9 + (13 + 5) \} = \underline{23}
 \end{aligned}$$

Tour:  $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$

Cost: 35

There are  $O(n \cdot 2^n)$   $g(i, S)$ 's to compute, each takes  $O(n)$  computation.

Therefore, the complexity of TSP by dynamic programming is  $O(n^2 \cdot 2^n)$ .

*Algorithm:*

```
procedure TSP (C, minlength);      //assume starting vertex is 1
{
  for  $i \leftarrow 2$  to  $n$  do  $g(i, \emptyset) = C(i, 1)$ ;
  for  $k \leftarrow 1$  to  $(n-2)$ 
    for all subsets  $S \subseteq V - \{1\}$  containing  $k$  vertices
      for ( $i$  such that  $i \neq 1$  and  $i \notin S$ )
        
$$\left[ \begin{array}{l} g(i, S) = \min_{j \in S} ( C(i, j) + g(j, S - \{j\}) ); \\ \text{path}(i, S) = \text{value of } j \text{ that gave the minimum;} \end{array} \right.$$

      
$$g(1, V - \{1\}) \leftarrow \min_{2 \leq j \leq n} ( C(1, j) + g(j, V - \{1, j\}) );$$

       $\text{path}(1, V - \{1\}) \leftarrow \text{value of } j \text{ that gave the minimum;}$ 
       $\text{minlength} \leftarrow g(1, V - \{1\});$ 
}
```