[CSE3081(2반)] 알고리즘 설계와 분석

2020학년도 2학기

강의자료

(2020.10.15 목요일)

서강대학교 공과대학 컴퓨터공학과
임 인 성 교수

[주제 4]
Dynamic Programming

The Manhattan Tourist Problem

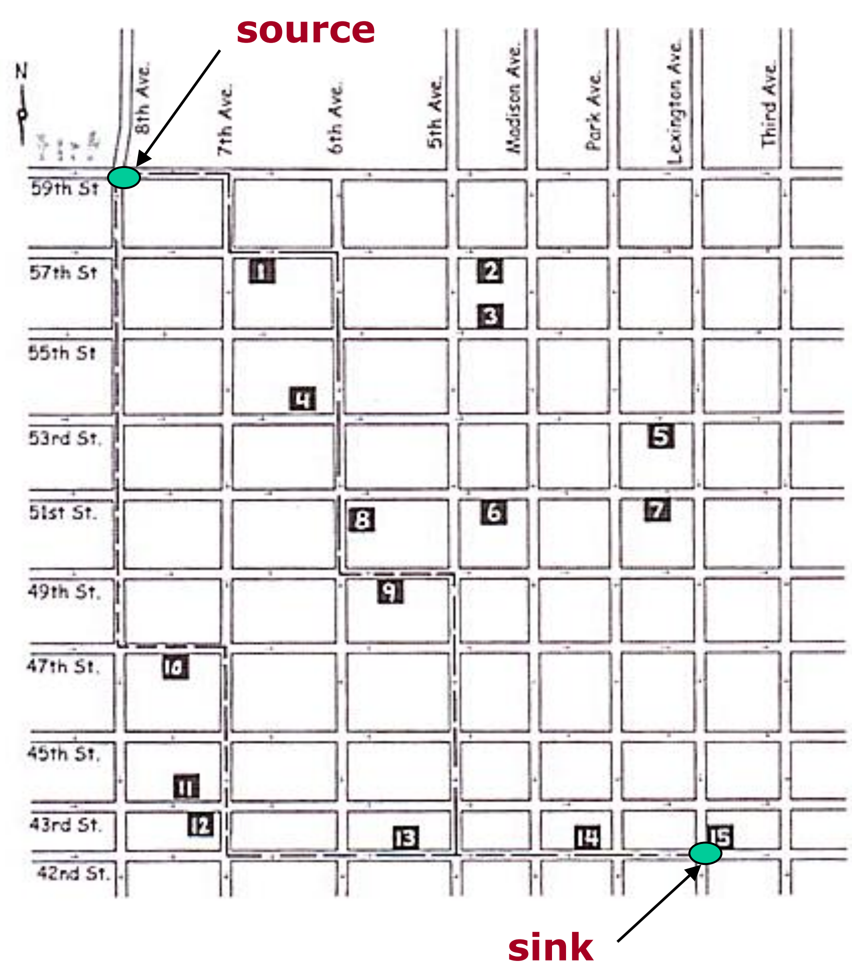
- **Problem:**
  - Given two street corners in the borough of Manhattan in New York City, find the path between them with the maximum number of attractions, that is, a path of maximum overall weight.

  ✓ Assume that a tourist may **move either to east or to south only**.

- **A brute force approach**
  - Search among all paths in the grid for the longest path!

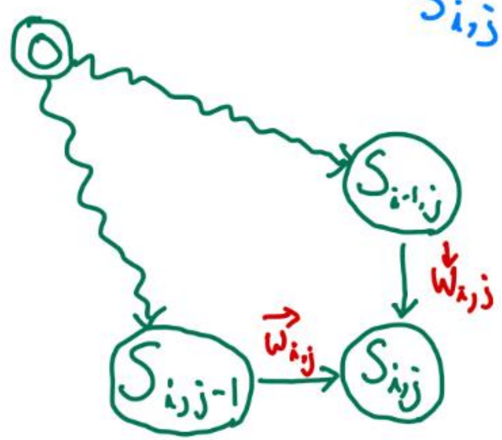
- **A greedy approach**
  - 다음 강의 주제

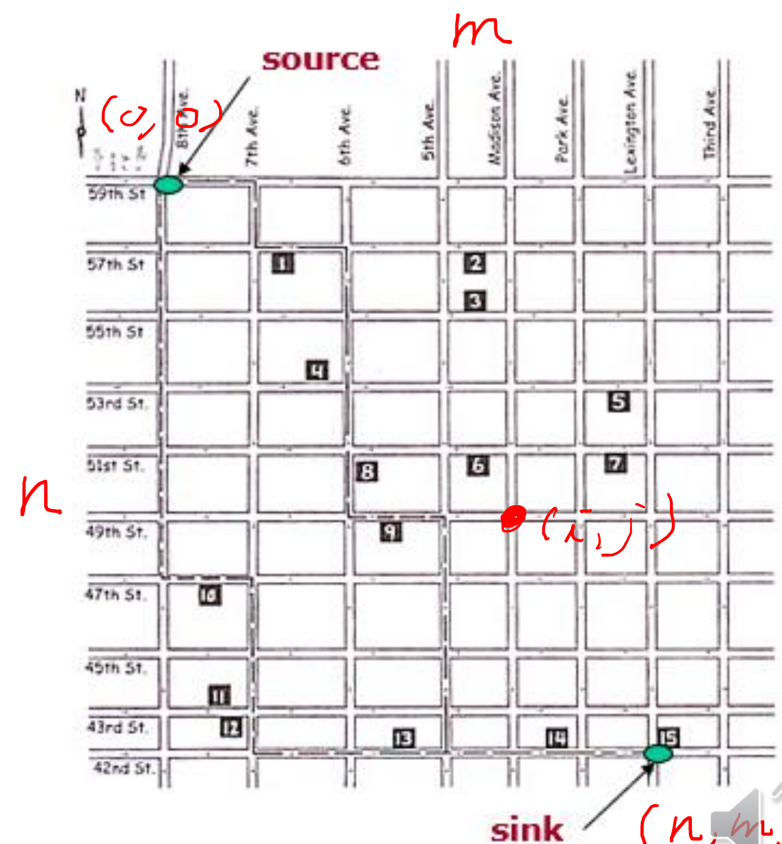
Courtesy of [Jones & Pevzner 6.3]



- **Basic idea**
  - How can you use the solutions of smaller problems to build a solution of a problem?

$$S_{i,j} = \max \{ S_{i-1,j} + \overset{\downarrow}{w}_{i,j} ,\ S_{i,j-1} + \vec{w}_{i,j} \}$$





A given optimization problem can be constructed efficiently from optimal solutions of its subproblems.

→ **optimal substructure**

- **Optimal substructure** $\boxed{s_{n,m} = ?}$

$s_{i,j} \equiv$ the length of the longest path from (0, 0) to (i, j)

1. $i, j \geq 1$

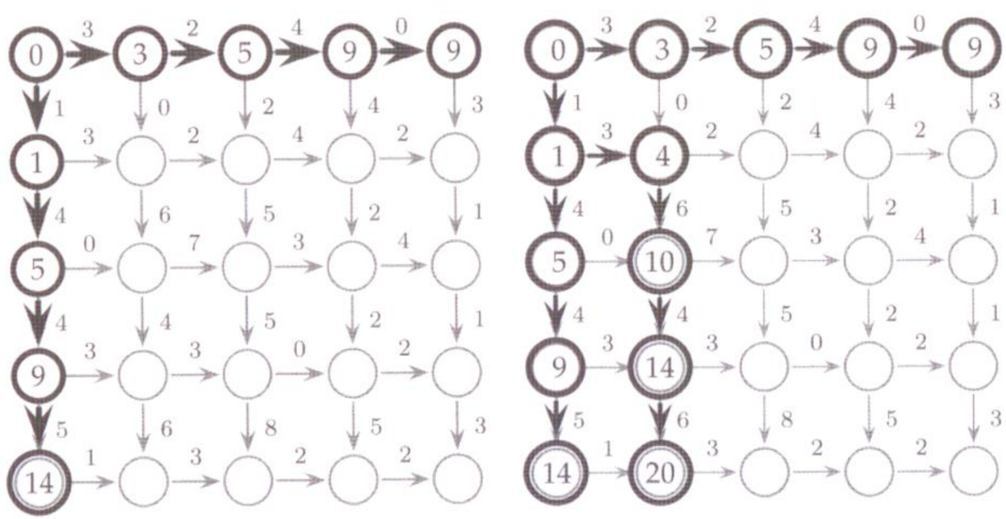
$$\boxed{s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{weight between } (i-1, j) \text{ and } (i, j) \\ s_{i,j-1} + \text{weight between } (i, j-1) \text{ and } (i, j) \end{cases}}$$

2. $i = 0, j = 1, 2, \ldots, n$ $\quad \boxed{s_{0,j} = s_{0,j-1} + \text{weight between } (0, j-1) \text{ and } (0, j)}$

3. $j = 0, i = 1, 2, \ldots, m$ $\quad \boxed{s_{i,0} = s_{i-1,0} + \text{weight between } (i-1, 0) \text{ and } (i, 0)}$

4. $i = j = 0$ $\quad \boxed{s_{0,0} = 0}$

- **Table setup and fill**



MANHATTANTOURIST($\downarrow\!\mathrm{w}, \vec{\mathrm{w}}, n, m$)

$s_{0,0} \leftarrow 0$
for $i \leftarrow 1$ to n
  $s_{i,0} \leftarrow s_{i-1,0} + \overset{\downarrow}{w}_{i,0}$
for $j \leftarrow 1$ to m
  $s_{0,j} \leftarrow s_{0,j-1} + \vec{w}_{0,j}$
for $i \leftarrow 1$ to n
  for $j \leftarrow 1$ to m
    $s_{i,j} \leftarrow \max \begin{cases} s_{i-1,j} + \overset{\downarrow}{w}_{i,j} \\ s_{i,j-1} + \vec{w}_{i,j} \end{cases}$
return $s_{n,m}$

- Given a (n, m) grid, what is the time complexity T(n, m)?

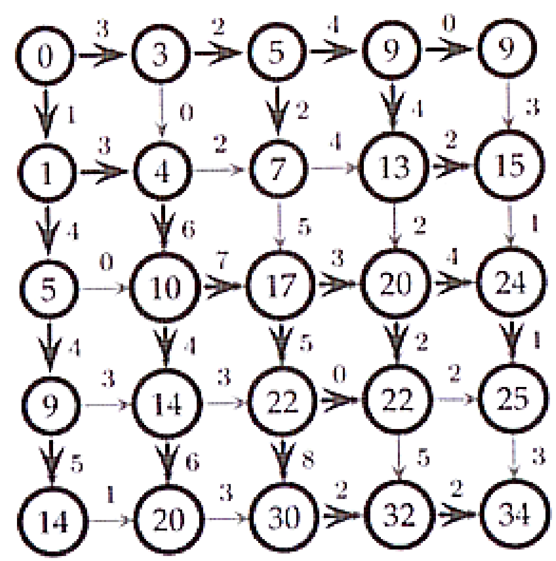
$$\boxed{O(???)}$$

- So far, we have found **the cost of the longest path** from source to each vertex in the grid.

- Then, how can you print out **the actual optimal path** from source to sink?

$\text{ManhattanTourist}(\overset{\downarrow}{\mathbf{w}}, \overset{\rightarrow}{\mathbf{w}}, n, m)$

```
s_{0,0} ← 0
for i ← 1 to n
        s_{i,0} ← s_{i-1,0} + ↓w_{i,0}
for j ← 1 to m
        s_{0,j} ← s_{0,j-1} + →w_{0,j}
for i ← 1 to n
        for j ← 1 to m
                s_{i,j} ← max { s_{i-1,j} + ↓w_{i,j}
                                s_{i,j-1} + →w_{i,j}
return s_{n,m}
```



Chained Matrix Multiplication

[Neapolitan 3.4]

- In general, to multiply an a x b matrix with a b x c matrix using the standard method, it is necessary to do abc elementary multiplications.

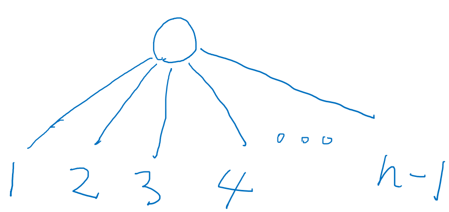
$$\underset{d_0 \times d_1}{A_1} \overset{1}{\times} \underset{d_1 \times d_2}{A_2} \overset{2}{\times} \underset{d_2 \times d_3}{A_3} \overset{3}{\times} \cdots \overset{i-1}{\times} \underset{d_{i-1} \times d_i}{A_i} \overset{i}{\times} \cdots \overset{n-1}{\times} \underset{d_{n-1} \times d_n}{A_n}$$

- **Problem**
  - Determine the minimum number of elementary multiplications, needed to multiply n matrices $A_1 A_2 A_3 \cdots A_n$ where $A_i \in R^{d_{i-1} \times d_i}$.
  - Ex: A_1 x A_2 x A_3 x A_4

    20 x 2   2 x 30   30 x 12   12 x 8

$$a\begin{bmatrix} a \times b \end{bmatrix}\begin{bmatrix} b \times c \end{bmatrix} = a\begin{bmatrix} a \times c \end{bmatrix}$$

- $A_1(A_2(A_3A_4))$: 30x12x8 + 2x30x8 + 20x2x8 = 3,680 multiplications
- $(A_1A_2)(A_3A_4)$: = 8,880 multiplications
- $A_1((A_2A_3)A_4)$: = 1,232 multiplications
- $((A_1A_2)A_3)A_4$: = 10,320 multiplications
- $(A_1(A_2A_3))A_4$: = 3,120 multiplications



✓ The order of multiplication is very important!

$(a \times b) \times c = a \times (b \times c)$

Dynamic programming approach

- **Definition**

  $M(i, j)$: the minimum number of multiplications needed to multiply A_i through A_j $(i \leq j)$.

$$\overbrace{A_i \times A_{i+1} \times \cdots \times A_k}^{d_{i-1} \times d_k} \times \overbrace{A_{k+1} \times \cdots \times A_{j-1} \times A_j}^{d_k \times d_j}$$

$d_{i-1} \times d_i$ $\quad d_i \times d_{i+1}$ $\quad d_{k-1} \times d_k$ $\quad d_k \times d_{k+1}$ $\quad d_{j-2} \times d_{j-1}$ $\quad d_{j-1} \times d_j$

- **Optimal subtructure**

$M(1, n)$

$$\boxed{M(i,j) = \begin{cases} \min_{i \leq k \leq j-1} \{\, M(i,k) + M(k+1,j) + d_{i-1} d_k d_j \,\}, & \text{if } i < j, \\ 0, & \text{if } i = j. \end{cases}}$$

$$M(i,j) = \begin{cases} \min_{i \le k \le j-1} \{ M(i,k) + M(k+1,j) + d_{i-1} d_k d_j \}, & \text{if } i < j, \\ 0, & \text{if } i = j. \end{cases}$$

- **Example:** $M(2, 7)$

M(1, 9)

$$M(2,7) = \min_{2 \le k \le 6} \{ M(2,k) + M(k+1,7) + d_1 d_k d_7 \}$$

$$\underbrace{A_2 \cdot A_3 \cdot A_4}_{d_1 \text{ x } d_4} \cdot \underbrace{A_5 \cdot A_6 \cdot A_7}_{d_4 \text{ x } d_7}$$

$M(2,4) + M(5,7) + d_1$ x d_4 x d_7

(5 cases)

$M(2,2) + M(3,7) + d_1$ x d_2 x d_7
$M(2,3) + M(4,7) + d_1$ x d_3 x d_7
$M(2,4) + M(5,7) + d_1$ x d_4 x d_7
$M(2,5) + M(6,7) + d_1$ x d_5 x d_7
$M(2,6) + M(7,7) + d_1$ x d_6 x d_7

i \ j	1	2	3	4	5	6	7	8	9
1	0								
2		0							
3			0						
4				0					
5					0				
6						0			
7							0		
8								0	
9									0

- **Table fill order**

$$M(2,7) = \min_{2 \le k \le 6} \{ M(2,k) + M(k+1,7) + d_1 d_k d_7 \}$$

$$\underbrace{A_2 \cdot A_3 \cdot A_4}_{d_1 \text{ x } d_4} \cdot \underbrace{A_5 \cdot A_6 \cdot A_7}_{d_4 \text{ x } d_7}$$

$M(2,4) + M(5,7) + d_1 \text{ x } d_4 \text{ x } d_7$

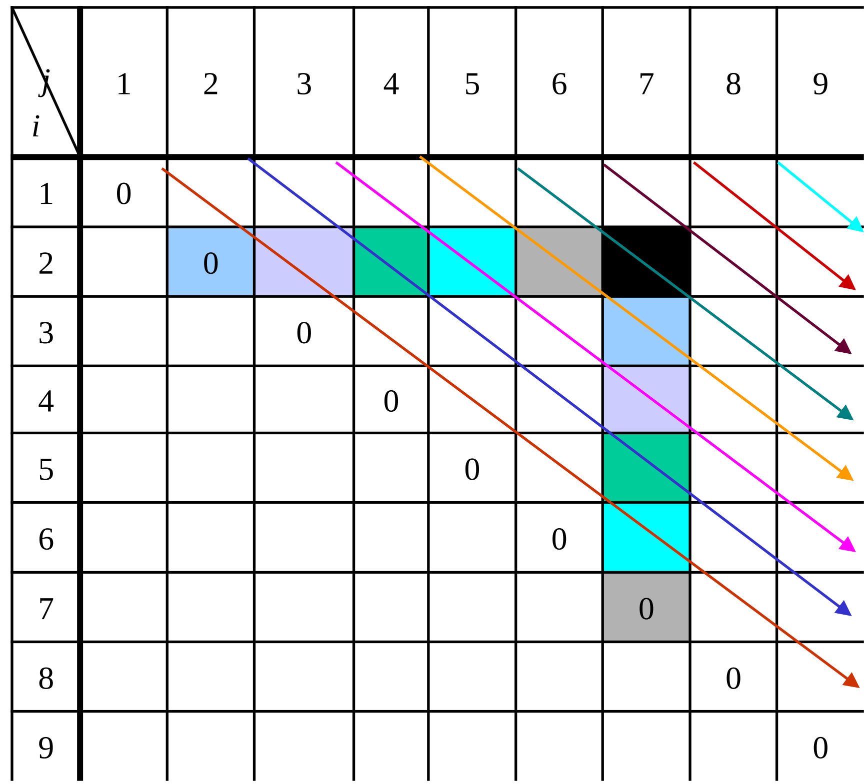
i \ j	1	2	3	4	5	6	7	8	9
1	0								
2		0							
3			0						
4				0					
5					0				
6						0			
7							0		
8								0	
9									0

```
for (i = 1; i <= n; i++) M[i][i] = 0;
for (g = 1; g <= n-1; g++)
  for (i = 1; i <= n-g; i++) {
    j = i + g;
    M[i][j] = BIG_NUM;
    for (k = i; k <= j-1; k++)
      if ((tmp = M[i][k] + M[k+1][j] + d[i-1]*d[k]*d[j]) < M[i][j]) {
        M[i][j] = tmp; P[i][j] = k;
      }
  }
```

$$\underbrace{A_1}_{d_0 \times d_1} \overset{1}{\times} \underbrace{A_2}_{d_1 \times d_2} \overset{2}{\times} \underbrace{A_3}_{d_2 \times d_3} \overset{3}{\times} \cdots \overset{i-1}{\times} \underbrace{A_i}_{d_{i-1} \times d_i} \overset{i}{\times} \cdots \overset{n-1}{\times} \underbrace{A_n}_{d_{n-1} \times d_n}$$

- **Time complexity**

$$
\begin{aligned}
& n+(n-1)\cdot 1+(n-2)\cdot 2+(n-3)\cdot 3+\cdots+(n-(n-1))\cdot(n-1) \\
= \quad & n+\sum_{g=1}^{n-1}(n-g)g \\
= \quad & O(n^3)
\end{aligned}
$$



```
Chained matrix multiplication problem
- O(n^3) by Godbole (1973)
- O(n^2) by Yao (1972)
- O(n log n) by Hu and Shing (1982, 1984)
```

$$\boxed{M(2,7) = \min_{2 \le k \le 6} \{ M(2,k) + M(k+1,7) + d_1 d_k d_7 \}}$$

- **Printing optimal order**

$A_1 \overset{1}{\times} A_2 \overset{2}{\times} A_3 \overset{3}{\times} A_4 \overset{4}{\times} A_5 \overset{5}{\times} A_6$

	1	2	3	4	5	6
1		1	1	1	1	1
2			2	3	4	5
3				3	4	5
4					4	5
5						5

P(1, 6) = 1

```
void order(int i, int j) {
  int k;
  if (i == j)
    printf("A_%d", i);
  else {
    k = P[i][j];
    printf("(");
    order(i, k);
    order(k+1, j);
    printf(")");
  }
}
```

P(1, 1) = ×

P(2, 6) = 5

→ $O(n)$ time

```
          (1,6)
         /     \
     (1,1)     (2,6)
              /     \
          (2,5)     (6,6)
         /     \
     (2,4)     (5,5)
    /     \
 (2,3)    (4,4)
 /    \
(2,2) (3,3)
```

$$(A_1((((A_2 A_3) A_4) A_5) A_6))$$