
“본 강의 동영상 및 자료는 대한민국 저작권법을 준수합니다. 본 강의 동영상 및 자료는 상명대학교 재학생들의 수업목적으로 제작·배포되는 것이므로, 수업목적으로 내려받은 강의 동영상 및 자료는 수업목적 이외에 다른 용도로 사용할 수 없으며, 다른 장소 및 타인에게 복제, 전송하여 공유할 수 없습니다. 이를 위반해서 발생하는 모든 법적 책임은 행위 주체인 본인에게 있습니다.”

Contents

6.0 Introduction

6.1 0/1-Knapsack

6.2 Weighted interval scheduling

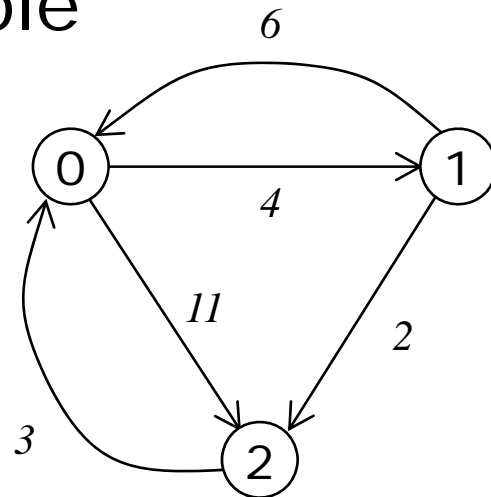
6.3 Multistage graph

6.4 All pairs shortest path

6.4 All Pairs Shortest path

- Problem: All-pairs shortest path (feat. 4.9)
 - The problem of finding shortest paths for every two vertices v to u .
 - Solving single-source shortest path for all vertices in G

- Example



	0	1	2
0			
1			
2			

6.4 All Pairs Shortest path

- Algorithm: Floyd's algorithm
 - Finding the all-pair's shortest path.
 - Input: adjacency matrix of a graph.
 - Output: minimum cost distance + path.
 - The weight of a path between two vertices is the sum of the weights of the edges along that path.
 - Negative weight is allowed.
 - Negative cycle is not allowed.

$$A^k[i][j] = \min \{ A^{k-1}[i][j], A^{k-1}[i][k] + A^{k-1}[k][j] \}$$

$$A^{-1}[i][j] = w[i][j]$$

6.4 All Pairs Shortest path

- Strategy
 - Suppose we wish to find a shortest path from vertex i to vertex j .
 - Let A_i be the vertices adjacent to i .
 - Which of the vertices in A_i should be the second vertex?
-

6.4 All Pairs Shortest path

- Strategy
 - Principle of optimality

Whatever the initial state and decision are, the remaining decisions must constitute an optimal decision sequence

- Let $A_1 = \{i_x\}$.
- The optimal path will be $i \rightarrow i_x \rightarrow \dots \rightarrow j$.
- Whatever i_x will be, $i_x \rightarrow j$ must be optimal.

$$Cost(i, i_1, i_2, \dots, j) = \min(Cost(i, i_x) + Cost(i_x, \dots, j))$$

6.4 All Pairs Shortest path

- Strategy
 - Let k be an intermediate vertex on a shortest path from i to j such that $\{i \rightarrow \dots \rightarrow k \rightarrow \dots \rightarrow j\}$.
 - k is max $(i_1, i_2, \dots, k, p_1, p_2, \dots)$
 - Path $\{i \rightarrow i_1 \rightarrow i_2 \rightarrow \dots \rightarrow k\}$ is shortest from i to k
 - Path $\{k \rightarrow p_1 \rightarrow p_2 \rightarrow \dots \rightarrow j\}$ is shortest from k to j

$$\begin{aligned} & \text{Cost}(i, i_1, i_2, \dots, k, p_1, p_2, \dots, j) \\ &= \text{Cost}(i, i_1, i_2, \dots, k,) + \text{Cost}(k, p_1, p_2, \dots, j) \end{aligned}$$

6.4 All Pairs Shortest path

- Algorithm: Floyd's algorithm
 - $A^k[i][j]$:
 - The cost of the shortest path from vertex i to j , using only those intermediate vertices with an index $\leq k$.
 - $A^{-1}[i][j]$: the weight of an edge connecting vertex i and vertex j

Minimum cost with those
vertices less than k

Minimum cost from i to k
those vertices less than k

Minimum cost from k to j
those vertices less than k

$$A^k[i][j] = \min \{ A^{k-1}[i][j], A^{k-1}[i][k] + A^{k-1}[k][j] \}$$

$$A^{-1}[i][j] = w[i][j]$$

6.4 All Pairs Shortest path

- Algorithm: Floyd's algorithm
 - Answer: $A^n[i][j]$:
 - The cost of the shortest path from vertex i to j , using only those intermediate vertices with an index $\leq n$.
 - n : number of vertices
-

6.4 All Pairs Shortest path

- Algorithm: Floyd's algorithm
 - Answer: $A^n[i][j]$:
 - The cost of the shortest path from vertex i to j , using only those intermediate vertices with an index $\leq n$.
 - n : number of vertices

$$A^n[i][j] = \min \{ A^{n-1}[i][j], A^{n-1}[i][n] + A^{n-1}[n][j] \}$$

$$A^{n-1}[i][j] = \min \{ A^{n-2}[i][j], A^{n-2}[i][n-1] + A^{n-2}[n-1][j] \}$$

...

$$A^1[i][j] = \min \{ A^0[i][j], A^0[i][1] + A^0[1][j] \}$$

$$A^0[i][j] = \min \{ A^{-1}[i][j], A^{-1}[i][0] + A^{-1}[0][j] \}$$

$$A^{-1}[i][j] = w[i][j]$$

6.4 All Pairs Shortest path

- Algorithm: Floyd's algorithm

```
void Floyd ( int cost[][ ], int dist[][ ], int n )
{
    int i, j, k;
    for ( i = 0; i < n; i++ )
        for ( j = 0; j < n; j++ )
            dist[i][j] = cost[i][j];

    for ( k = 0; k < n; k++ ) {
        for ( i = 0; i < n; i++ )
            for ( j = 0; j < n; j++ )
                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);
    }
}
```
