Floyd-Warshall

플로이드-워셜 알고리즘

Contents

Definition

'모든 지점에서 다른 모든 지점까지의
최단 경로를 모두 구해야 하는 경우'
사용하는 알고리즘

Comparison

그렇다면 다익스트라 알고리즘과의 차이점은?

Comparison

	다익스트라	플로이드 워셜
	임의의 노드로부터 모든 노드의 최단거리	모든 노드로부터 모든 노드의 최단거리
	양의 간선만 가능	음의 간선도 가능
	그리디 알고리즘	DP - 점화식을 사용
	1차원 배열	2차원 배열
	O(ElogV)~O(V**2)	O(N**3)

https://algorithms.discrete.ma.tum.de/graph-algorithms/spp-floyd-warshall/index_en.html

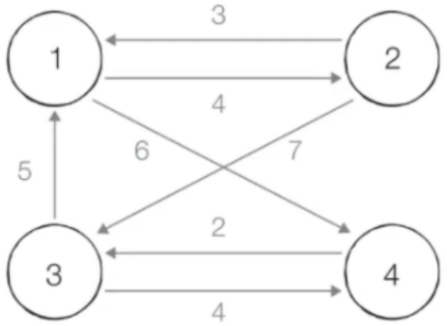
Algorithm

https://www.cs.usfca.edu/~galles/visualization/Floyd.html

$$D_{ab} = \min\left(D_{ab}, D_{ak} + D_{kb}\right)$$

[step 0] 그래프의 노드와 간선에 따라 최단 거리 테이블을 갱신한다.

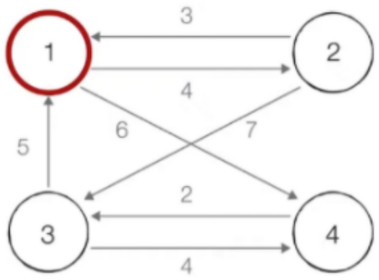
- **기본 점화식**: $D_{ab} = \min(D_{ab}, D_{ak} + D_{kb})$



출발 \ 도착	1번	2번	3번	4번
1번	0	4	무한	6
2번	3	0	7	무한
3번	5	무한	0	4
4번	무한	무한	2	0

[step 1] 1번 노드를 거쳐 가는 경우를 고려하여 테이블을 갱신한다.

- **점화식**: $D_{ab} = \min(D_{ab}, D_{a1} + D_{1b})$



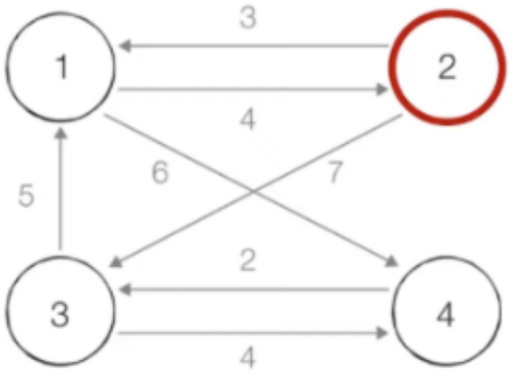
$D_{23} = \min(D_{23}, D_{21} + D_{13})$
$D_{24} = \min(D_{24}, D_{21} + D_{14})$
$D_{32} = \min(D_{32}, D_{31} + D_{12})$
$D_{34} = \min(D_{34}, D_{31} + D_{14})$
$D_{42} = \min(D_{42}, D_{41} + D_{12})$
$D_{43} = \min(D_{43}, D_{41} + D_{13})$

갱신된 최단 거리 테이블

0	4	무한	6
3	0	7	9
5	9	0	4
무한	무한	2	0

[step 2] 2번 노드를 거쳐 가는 경우를 고려하여 테이블을 갱신한다.

- **점화식**: $D_{ab} = \min(D_{ab}, D_{a2} + D_{2b})$



$D_{13} = \min(D_{13}, D_{12} + D_{23})$

$D_{14} = \min(D_{14}, D_{12} + D_{24})$

$D_{31} = \min(D_{31}, D_{32} + D_{21})$

$D_{34} = \min(D_{34}, D_{32} + D_{24})$

$D_{41} = \min(D_{41}, D_{42} + D_{21})$

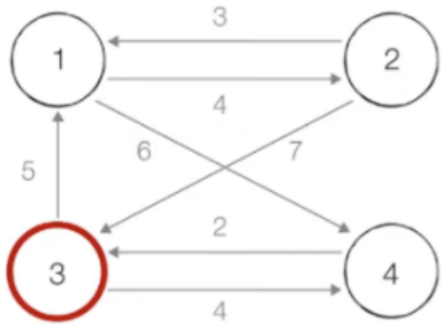
$D_{43} = \min(D_{43}, D_{42} + D_{23})$

갱신된 최단 거리 테이블

0	4	11	6
3	0	7	9
5	9	0	4
무한	무한	2	0

[step ~] 3번, 4번, ... 노드를 거쳐 가는 경우를 고려하여 테이블을 갱신한다.

- **점화식**: $D_{ab} = \min(D_{ab}, D_{a3} + D_{3b})$

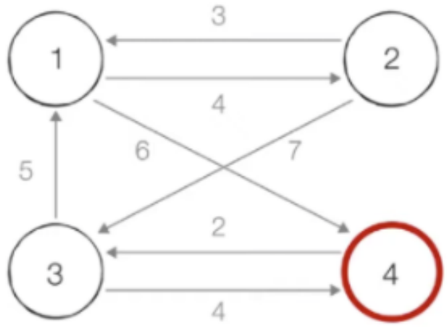


$$D_{12} = \min(D_{12}, D_{13} + D_{32})$$
$$D_{14} = \min(D_{14}, D_{13} + D_{34})$$
$$D_{21} = \min(D_{21}, D_{23} + D_{31})$$
$$D_{24} = \min(D_{24}, D_{23} + D_{34})$$
$$D_{41} = \min(D_{41}, D_{43} + D_{31})$$
$$D_{42} = \min(D_{42}, D_{43} + D_{32})$$

갱신된 최단 거리 테이블

0	4	11	6
3	0	7	9
5	9	0	4
7	11	2	0

- **점화식**: $D_{ab} = \min(D_{ab}, D_{a4} + D_{4b})$



$$D_{12} = \min(D_{12}, D_{14} + D_{42})$$
$$D_{13} = \min(D_{14}, D_{14} + D_{43})$$
$$D_{21} = \min(D_{21}, D_{24} + D_{41})$$
$$D_{23} = \min(D_{23}, D_{24} + D_{43})$$
$$D_{31} = \min(D_{31}, D_{34} + D_{41})$$
$$D_{32} = \min(D_{32}, D_{34} + D_{42})$$

갱신된 최단 거리 테이블

0	4	8	6
3	0	7	9
5	9	0	4
7	11	2	0

```python
INF = int(1e9)

# 노드의 개수(n)과 간선의 개수(m) 입력
n = int(input())
m = int(input())

# 2차원 리스트 만들고, 무한대로 초기화
graph = [[INF] * (n + 1) for _ in range(n + 1)]

# 자신 -> 자신으로 가능 비용 (0으로 초기화)
for a in range(1, n + 1):
    for b in range(1,  n + 1):
        if a == b:
            graph[a][b] = 0

# 각 간선에 대한 정보 입력받기
for _ in range(m):
    a, b, c = map(int, input().split())
    graph[a][b] = c

# 점화식에 따라 플로이드 워셜 알고리즘을 수행
for k in range(1, n + 1):
    for a in range(1, n + 1):
        for b in range(1, n + 1):
            graph[a][b] = min(graph[a][b], graph[a][k] + graph[k][b])

# 수행된 결과를 출력
for a in range(1, n + 1):
    for b in range(1, n + 1):
        if graph[a][b] == INF:
            print('INFINITY', end = " ")
        else:
            print(graph[a][b], end= " ")
        print()
```

시간복잡도: O(N**3)

- 노드의 개수가 N개 일 때, N번의 단계를 수행하며, 단계마다 O(N**2)의 연산을 통해 '현재 노드를 거쳐가는 모든 경로'를 고려하기 때문에

Example