9주차 : 플로이드-워셜

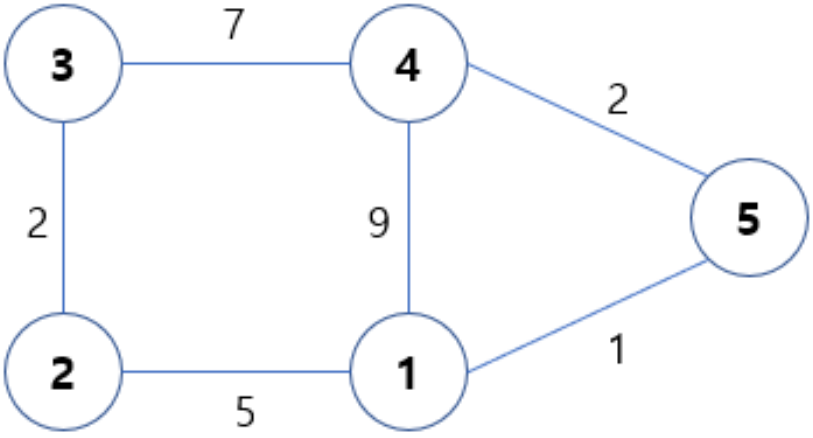
플로이드-워셜 알고리즘은 **모든 정점**에서 다른 **모든 정점**으로 가는 최단 경로를 알려준다.

- 플로이드 워셜은 모든 지점에서 다른 모든 지점까지의 최단 거리를 저장해야 하기 때문에 2차원 테이블에 최단 거리 정보를 저장한다.
- 음의 간선도 사용 가능하다.
- 노드의 개수가 `N`개라고 할 때, `N`번 만큼의 단계를 반복하며 '점화식에 맞게' 2차원 리스트를 갱신하기 때문에 DP라고 볼 수 있다.
    - Q. 다익스트라는 그리디 알고리즘??

플로이드-워셜 알고리즘 점화식

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

플로이드-워셜 알고리즘 진행과정



[0단계] 그래프의 노드와 간선에 따른 인접 행렬 구성

- 모든 노드 간의 최단거리를 구해야 하므로 2차원 인접 행렬을 구성
- 해당 노드에서 특정 노드까지 가는 길이 없는 경우 INF로 초기 인접행렬 구성

초기 인접행렬

	1	2	3	4	5
1	0	5	INF	9	1
2	5	0	2	INF	INF
3	INF	2	0	7	INF
4	9	INF	7	0	2
5	1	INF	INF	2	0

[1단계] 1번 노드를 거치는 경우를 고려하여 인접 행렬 갱신

- 1번 노드를 중간노드로 하여 2 - 1 - 4 (5 + 9 = 14) 로 갈 수 있음
- 1번 노드를 중간노드로 하여 2 - 1 - 5 (5 + 1 = 6) 로 갈 수 있음

	1	2	3	4	5
1	0	5	INF	9	1
2	5	0	2	14	6
3	INF	2	0	7	INF
4	9	14	7	0	2
5	1	6	INF	2	0

▼ 점화식 예시

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

$$
\begin{aligned}
D_{23} &= min(D_{23}, D_{21} + D_{13}) \\
D_{24} &= min(D_{24}, D_{21} + D_{14}) \\
D_{25} &= min(D_{25}, D_{21} + D_{15}) \\
D_{32} &= min(D_{32}, D_{31} + D_{12}) \\
D_{34} &= min(D_{34}, D_{31} + D_{14}) \\
D_{35} &= min(D_{35}, D_{31} + D_{15}) \\
D_{42} &= min(D_{42}, D_{41} + D_{12}) \\
D_{43} &= min(D_{43}, D_{41} + D_{13}) \\
D_{45} &= min(D_{45}, D_{41} + D_{15}) \\
D_{52} &= min(D_{52}, D_{51} + D_{12}) \\
D_{53} &= min(D_{53}, D_{51} + D_{13}) \\
D_{54} &= min(D_{54}, D_{51} + D_{14})
\end{aligned}
$$

[2단계] 2번 노드를 거치는 경우를 고려하여 인접 행렬 갱신

- 2번 노드를 중간노드로 하여 1 - 2 - 3 (5 + 2 = 7) 로 갈 수 있음
- 2번 노드를 중간노드로 하여 3 - 2 - 5 (2 + 6 = 8) 로 갈 수 있음

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

	1	2	3	4	5
1	0	5	7	9	1
2	5	0	2	14	6
3	7	2	0	7	8
4	9	14	7	0	2
5	1	6	8	2	0

[3단계] 3번 노드를 거치는 경우를 고려하여 인접 행렬 갱신

- 3번 노드를 중간노드로 하여 2 - 3 - 4 (2 + 7 = 9) 로 갈 수 있음

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

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

[4단계] 4번 노드를 거치는 경우를 고려하여 인접 행렬 갱신

- 4번 노드를 중간노드로 하여 가중치가 최소가 되는 경우 X

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

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

[5단계] 5번 노드를 거치는 경우를 고려하여 인접 행렬 갱신

- 5번 노드를 중간노드로 하여 1 - 5 - 4 (1 + 2 = 3) 로 갈 수 있음
- 5번 노드를 중간노드로 하여 2 - 5 - 4 (6 + 2 = 8) 로 갈 수 있음

※점화식 : $D_{ab} = min(D_{ab}, D_{a5} + D_{5b})$

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

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

플로이드-워셜 코드

```
# 양승열

max_cost = 100001*100

# 입력부 n = 도시의 개수 / m = 버스의 개수 / a = 시작 도시, b = 도착 도시, c = 1회 탑승 비용
n = int(input())
m = int(input())
floyd = [[max_cost]*n for _ in range(n)]

# 시작 도시 = 도착 도시 0처리
for i in range(n):
    floyd[i][i] = 0

# 입력값 배열 세팅 - 시작 도시와 도착 도시를 연결 하는 노선은 하나가 아닐 수 있기 때문에 min() 사용
for _ in range(m):
    a, b, c = map(int, input().split())
    floyd[a-1][b-1] = min(floyd[a-1][b-1], c)

# Floyd Algorithm
for i in range(n):
    for j in range(n):
        for k in range(n):
            floyd[j][k] = min(floyd[j][k], floyd[j][i] + floyd[i][k])  # i를 거칠 때와 아닐때 비교해 더 짧은 거
리 채택

# 출력부 - 이렇게 하면 98%에서 오류,,, 원인이 뭘까요? -- max_cost설정 너무 낮게함
for i in floyd:
    for j in i:
        if j == max_cost:
            j = 0
        print(j, end=' ')
    print()
```

노드의 개수가 N개 일 때, **시간 복잡도 O(N^3)**

세제곱 시간 알고리즘을 적용해도 문제가 풀릴 때만 사용할 수 있습니다. **(노드의 갯수가 너무 많지 않은 경우)**

문제

플로이드 (Gold4) : https://www.acmicpc.net/problem/11404

플로이드 2 (Gold2) : https://www.acmicpc.net/problem/11780