

백준 11404, 플로이드

키워드 : 플로이드-와샬 알고리즘

인접행렬

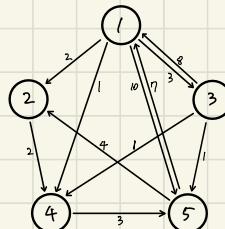
	1	2	3	4	5
1	0	2	3	1	10
2	INF	0	INF	2	INF
3	8	INF	0	1	1
4	INF	INF	INF	0	3
5	7	4	INF	INF	0

$d^{(k=0)}$

	1	2	3	4	5
1	0	2	3	1	10
2	INF	0	INF	2	INF
3	8	INF	0	1	1
4	INF	INF	INF	0	3
5	7	4	INF	INF	0

$\pi^{(k=0)}$

	1	2	3	4	5
1	NIL	1	1	1	1
2	NIL	NIL	N	2	NIL
3	3	NIL	NIL	3	3
4	NIL	NIL	NIL	NIL	4
5	5	5	NIL	NIL	NIL



$d_{ij}^{(k)}$: 경로 $i \rightarrow j$ 까지 최단경로
 $\{1, 2, 3, \dots, k\}$: $i \rightarrow j$ 까지의 경로 경로

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k=0 \\ \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{if } k \geq 1 \end{cases}$$

π : 직전 선정노드

① 경로 { }에 3사이에 경로가 존재하지 않으면 $\pi_{ij}=NIL$
 ② π_{ij} 는 경로 j 의 직전 경로이다

$k=1$ 일 때 d, π 값 변화

	1	2	3	4	5
1	0	2	3	1	10
2	INF	0	INF	2	INF
3	8	INF	0	1	1
4	INF	INF	INF	0	3
5	7	4	INF	INF	0

	1	2	3	4	5
1	0	2	3	1	10
2	INF	0	INF	2	INF
3	8	10	0	1	1
4	INF	INF	INF	0	3
5	7	4	10	8	0

	1	2	3	4	5
1	NIL	1	1	1	1
2	NIL	NIL	N	2	NIL
3	3	1	NIL	3	3
4	NIL	NIL	NIL	NIL	4
5	5	5	1	1	NIL

$$d_{ij}^{(k=1)} = \min(d_{ij}^{(k=0)}, d_{ik}^{(k=0)} + d_{kj}^{(k=0)})$$

$k=1$

$$\begin{aligned} d_{11}^{(k=1)} &= \min(d_{11}^{(k=0)}, d_{11}^{(k=0)} + d_{11}^{(k=0)}) \\ &= \min(0, 0+0)=0 \end{aligned}$$

$d_{12}^{(k=1)}$

$$\begin{aligned} d_{12}^{(k=1)} &= \min(d_{12}^{(k=0)}, d_{11}^{(k=0)} + d_{12}^{(k=0)}) \\ &= \min(2, 0+2)=2 \end{aligned}$$

$d_{13}^{(k=1)}$

$$\begin{aligned} d_{13}^{(k=1)} &= \min(d_{13}^{(k=0)}, d_{11}^{(k=0)} + d_{13}^{(k=0)}) \\ &= \min(3, 0+3)=3 \end{aligned}$$

$d_{14}^{(k=1)}$

$$\begin{aligned} d_{14}^{(k=1)} &= \min(d_{14}^{(k=0)}, d_{11}^{(k=0)} + d_{14}^{(k=0)}) \\ &= \min(1, 0+1)=1 \end{aligned}$$

$d_{15}^{(k=1)}$

$$\begin{aligned} d_{15}^{(k=1)} &= \min(d_{15}^{(k=0)}, d_{11}^{(k=0)} + d_{15}^{(k=0)}) \\ &= \min(10, 0+10)=10 \end{aligned}$$

$$\begin{aligned} d_{21}^{(k=1)} &= \min(d_{21}^{(k=0)}, d_{21}^{(k=0)} + d_{11}^{(k=0)}) \\ &= \min(INF, INF+0)=INF \end{aligned}$$

$d_{22}^{(k=1)}$

$$\begin{aligned} d_{22}^{(k=1)} &= \min(d_{22}^{(k=0)}, d_{21}^{(k=0)} + d_{22}^{(k=0)}) \\ &= \min(0, INF+2)=INF \end{aligned}$$

$d_{23}^{(k=1)}$

$$\begin{aligned} d_{23}^{(k=1)} &= \min(d_{23}^{(k=0)}, d_{21}^{(k=0)} + d_{23}^{(k=0)}) \\ &= \min(INF, INF+2)=INF \end{aligned}$$

$d_{24}^{(k=1)}$

$$\begin{aligned} d_{24}^{(k=1)} &= \min(d_{24}^{(k=0)}, d_{21}^{(k=0)} + d_{24}^{(k=0)}) \\ &= \min(2, INF+2)=INF \end{aligned}$$

$d_{25}^{(k=1)}$

$$\begin{aligned} d_{25}^{(k=1)} &= \min(d_{25}^{(k=0)}, d_{21}^{(k=0)} + d_{25}^{(k=0)}) \\ &= \min(INF, INF+10)=INF \end{aligned}$$

$$\begin{aligned} d_{31}^{(k=1)} &= \min(d_{31}^{(k=0)}, d_{31}^{(k=0)} + d_{11}^{(k=0)}) \\ &= \min(8, 8+0)=8 \end{aligned}$$

$d_{32}^{(k=1)}$

$$\begin{aligned} d_{32}^{(k=1)} &= \min(d_{32}^{(k=0)}, d_{31}^{(k=0)} + d_{12}^{(k=0)}) \\ &= \min(INF, 8+2)=10 \end{aligned}$$

$d_{33}^{(k=1)}$

$$\begin{aligned} d_{33}^{(k=1)} &= \min(d_{33}^{(k=0)}, d_{31}^{(k=0)} + d_{13}^{(k=0)}) \\ &= \min(0, 8+3)=8 \end{aligned}$$

$d_{34}^{(k=1)}$

$$\begin{aligned} d_{34}^{(k=1)} &= \min(d_{34}^{(k=0)}, d_{31}^{(k=0)} + d_{14}^{(k=0)}) \\ &= \min(0, INF+1)=0 \end{aligned}$$

$d_{35}^{(k=1)}$

$$\begin{aligned} d_{35}^{(k=1)} &= \min(d_{35}^{(k=0)}, d_{31}^{(k=0)} + d_{15}^{(k=0)}) \\ &= \min(1, 8+10)=11 \end{aligned}$$

$$\begin{aligned} d_{41}^{(k=1)} &= \min(d_{41}^{(k=0)}, d_{41}^{(k=0)} + d_{11}^{(k=0)}) \\ &= \min(INF, INF+0)=INF \end{aligned}$$

$d_{42}^{(k=1)}$

$$\begin{aligned} d_{42}^{(k=1)} &= \min(d_{42}^{(k=0)}, d_{41}^{(k=0)} + d_{12}^{(k=0)}) \\ &= \min(INF, INF+2)=INF \end{aligned}$$

$d_{43}^{(k=1)}$

$$\begin{aligned} d_{43}^{(k=1)} &= \min(d_{43}^{(k=0)}, d_{41}^{(k=0)} + d_{13}^{(k=0)}) \\ &= \min(INF, INF+3)=INF \end{aligned}$$

$d_{44}^{(k=1)}$

$$\begin{aligned} d_{44}^{(k=1)} &= \min(d_{44}^{(k=0)}, d_{41}^{(k=0)} + d_{14}^{(k=0)}) \\ &= \min(0, INF+4)=0 \end{aligned}$$

$d_{45}^{(k=1)}$

$$\begin{aligned} d_{45}^{(k=1)} &= \min(d_{45}^{(k=0)}, d_{41}^{(k=0)} + d_{15}^{(k=0)}) \\ &= \min(INF, 8+10)=18 \end{aligned}$$

$$\begin{aligned} d_{51}^{(k=1)} &= \min(d_{51}^{(k=0)}, d_{51}^{(k=0)} + d_{11}^{(k=0)}) \\ &= \min(7, 7+0)=7 \end{aligned}$$

$d_{52}^{(k=1)}$

$$\begin{aligned} d_{52}^{(k=1)} &= \min(d_{52}^{(k=0)}, d_{51}^{(k=0)} + d_{12}^{(k=0)}) \\ &= \min(4, 7+2)=9 \end{aligned}$$

$d_{53}^{(k=1)}$

$$\begin{aligned} d_{53}^{(k=1)} &= \min(d_{53}^{(k=0)}, d_{51}^{(k=0)} + d_{13}^{(k=0)}) \\ &= \min(INF, 7+3)=10 \end{aligned}$$

$d_{54}^{(k=1)}$

$$\begin{aligned} d_{54}^{(k=1)} &= \min(d_{54}^{(k=0)}, d_{51}^{(k=0)} + d_{14}^{(k=0)}) \\ &= \min(INF, 7+4)=11 \end{aligned}$$

$d_{55}^{(k=1)}$

$$\begin{aligned} d_{55}^{(k=1)} &= \min(d_{55}^{(k=0)}, d_{51}^{(k=0)} + d_{15}^{(k=0)}) \\ &= \min(0, 7+10)=0 \end{aligned}$$

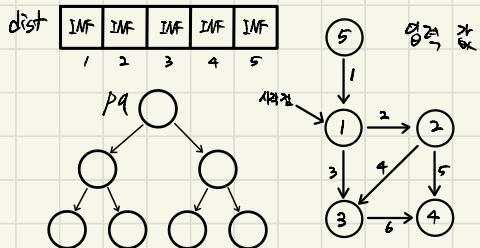
백준 1753, 최단 경로

키워드 : 다익스트라 알고리즘

Step 1. 변수 초기화

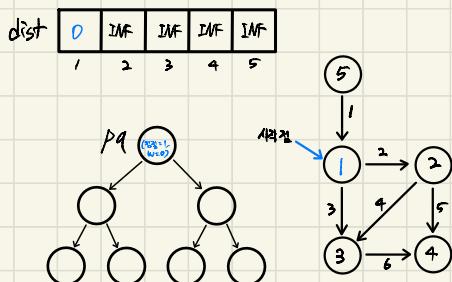
dist: 시작점부터 최단 거리를 저장하는 배열

PQ: 현재 경유지에서 최단 거리로 갈 수 있는 간선을 저장하는 priority-queue (가중치 힙구조)



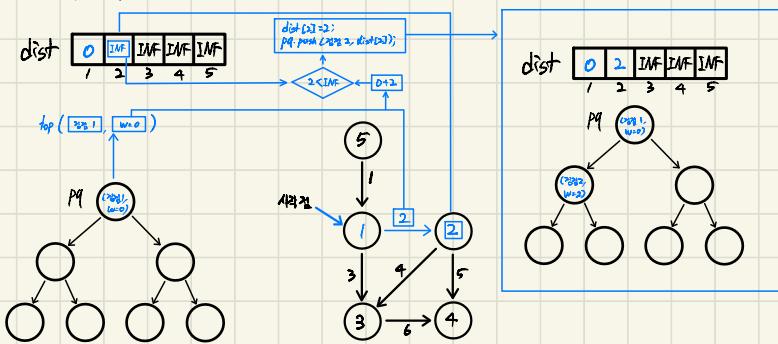
Step 2. 시작점 1의 dist를 0으로 초기화

PQ에 간선 정보 (경유지, w=0) push



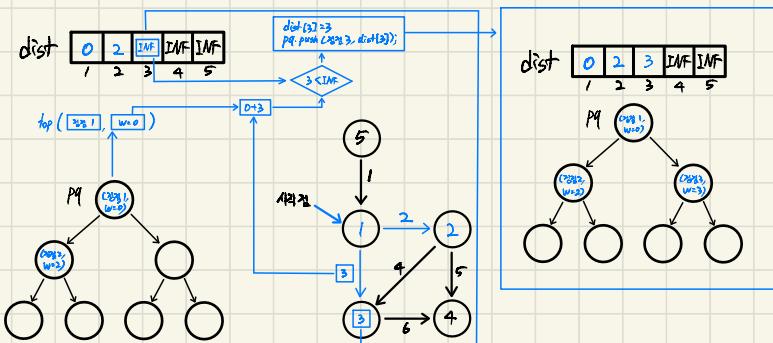
Step 3-1. PQ의 top ([노드, w=0])의 경유지에서 조속히는 경유지를 끌어온다

마지막 경이 INF를 생성한다



Step 3-2. PQ의 top ([노드, w=0])의 경유지에서 조속히는 경유지를 끌어온다

마지막 경이 INF를 생성한다



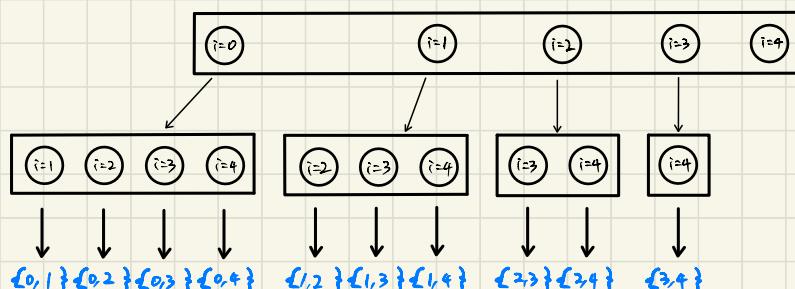
Step 4. PQ가 모두 비어질 때까지 step3 반복

Result: dist [0 2 3 7 INF]

백준 15686, 치킨 배달

키워드 : 조합 구하기, DFS

모든 조합 구하기
DFS ($s=0, L=0$), $nCr (n=5, r=2)$



$L=0$
 $arr[L]=i$
 $s=i+1$
 $L=L+1$
 $L=1$
 $arr[L]=i$
 $s=i+1$
 $L=L+1$
 $L=2$
 $if(L==r)$
return;

step 1 변수 초기화

h : 집의 위치를 담은 vector

C : 치킨집의 위치를 담은 vector

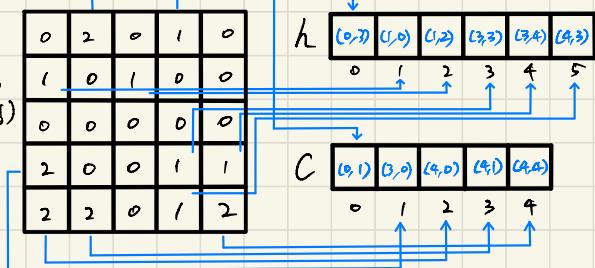
ARR : C 의 모든 조합의 원소로 배열

(지금은 경비를 위해 예상 형태로 바꾸었지만,
실제는 다른 형태로 DFS에서 원형으로 계산)

out: 5시작 치킨집의 최솟값(출처: 강)

ARR $\{0,1\} \{0,2\} \{0,3\} \{0,4\} \{1,2\} \{1,3\} \{1,4\} \{2,3\} \{2,4\} \{3,4\}$

입력
(0:본관, 1:점, 2:치킨점)
(0:본관, 1:점, 2:치킨점)



step 2 ARR 전체를 돌면서 최솟값을 찾는다.

pseudo code

```

int out = INT_MAX; // 출발점 정하기
for(int i=0; i<arr.size(); ) // 전체 arr 순회
{
    int sum = 0; // 현재 조합의 거리
    // 경비 4
    for(int j=0; j<h.size(); )
    {
        int min_dist = INT_MAX; // 각 집마다 최초거리 찾을 변수
        // 예상지역의 양수 카운트 4
        for(int k=0; k< ARR[j].size(); )
        {
            pair<int, int> h-point = h[j]; // 집 위치
            pair<int, int> C-point = C[ARR[j][k]]; // 치킨점 위치

            int dist = abs(h-point.first - C-point.first) +
                      abs(h-point.second - C-point.second); // 거리 계산
            min_dist = min(min_dist, dist); // 최초거리 갱신
        }
        sum += min_dist;
    }
    out = min(out, sum); // 출발점 갱신
}
    
```

ARR $\{0,1\} \{0,2\} \{0,3\} \{0,4\} \{1,2\} \{1,3\} \{1,4\} \{2,3\} \{2,4\} \{3,4\}$

h $\{0,2\} \{1,0\} \{1,2\} \{3,3\} \{4,3\}$
0 1 2 3 4 5

C $\{0,1\} \{3,0\} \{4,0\} \{4,1\} \{4,4\}$
0 1 2 3 4

백준 1197, 최소 스패닝 트리

키워드 : 최소 스패닝 트리, Union-Find

Union-Find 알고리즘 / pseudo-code

```

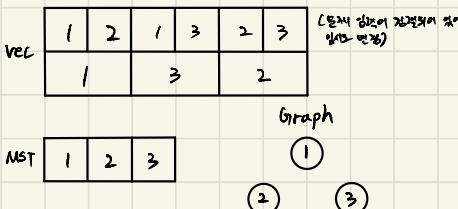
int find(int a) // 서로 다른 원인
{
    if(MST[a]==a)
        return MST[a];
    else
        return MST[a] = find(MST[a]);
}

void Union(int a, int b)
{
    a=find(a);
    b=find(b);
    if(a!=b) // a와 b가 연결되어 있지 않으면
    {
        MST[a]=b; // a와 b를 연결
    }
}

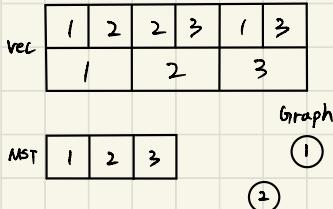
```

Step 1. 연결 초기화

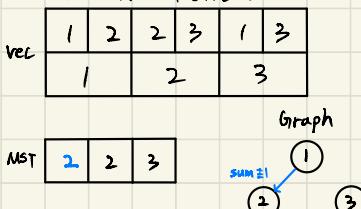
NPV : 최소 스패닝 트리 대상
VEC : 모든 정점 번호들인 Vector
SUM : 최소 스패닝 트리 가중치



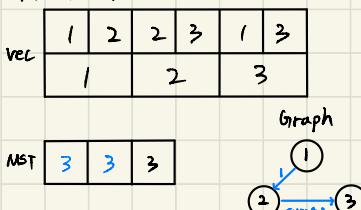
Step 2 모든 간선을 가중치 오름차순으로 정렬



Step 3 Vec[0]에서 첫정점(1과 2를 Union-Find 알고리즘으로 연결한다)
이후 sum += 1 → 2. 이 과정을 1을 반복하



Step 4 그 다음, Vec[1] MST로는 개별로 Union-Find 알고리즘으로 연결한다

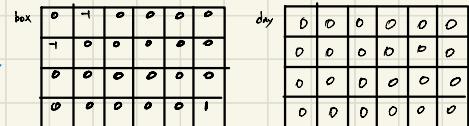


Step 5 마지막, Vec[2]에서 첫정점(가 3개 주어진 경우로 계산하지 않는다).

백준 7576, 토마토

키워드 : BFS

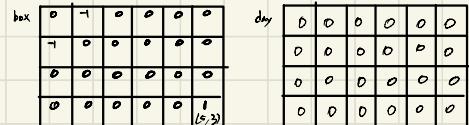
Step1 토마토 배달
days : 토마토 배달에 걸리는 일정
box : 상자에 넣은 토마토 수
q : 토마토 배달 경로를 찾는 queue



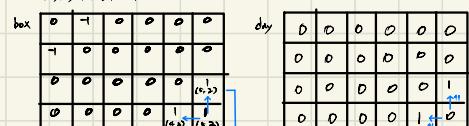
q:

--	--	--	--	--	--	--

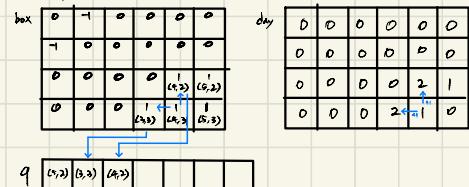
Step2 box 배열을 확인하여 1인 곳에 토마토 1개 push



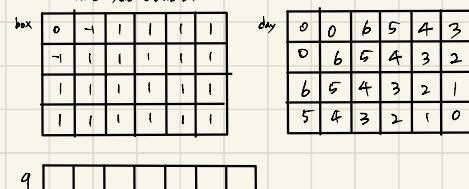
Step3 1에서 2까지, 2에서 3까지, 3에서 4까지, 4에서 5까지, 5에서 6까지, 6에서 7까지, 7에서 8까지, 8에서 9까지, 9에서 10까지 토마토를 더해나간다.



Step4 step3를 반복, L(9x5x2)로 가면 끝까지



Step5 마지막에 box에 넣은 토마토를 day에 넣어야 한다. 즉, day에 있는 box의 토마토 수를 더해나간다.



백준 2178, 미로 탐색

키워드 : BFS

Step1 맵과 초기화

ATM : 현재 칸과 상대를 같은 배열(이동할 가능한 칸) (:이동 가능한 칸)

dist : 시작 위치에서 출발지까지 거리로 초기화하는 배열(:이동 경로)

q : 찾고자 하는 위치를 위한 queue

ATM	1	0	1	1	1	1	1
	1	0	1	0	1	0	
	1	0	1	0	1	1	
	1	1	1	0	1	1	

dist

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

9							
---	--	--	--	--	--	--	--

Step 2 시작 점(1,1)을 큐에 push 한다
출발점은 큐에 있으므로 맨처음이다

ATM	1	0	1	1	1	1
	1	0	1	0	1	0
	1	0	1	0	1	1
	1	1	1	0	1	1

dist

1	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

9	(1,1)						
---	-------	--	--	--	--	--	--

Step 3 큐에서 꺼낸 다음 그 칸으로 갈 수 있는 경로 기록을 위해 mode 전환
여기서 dist[0][0]와 front[0][0] = dist[front[0][0]][front[0][0]]로 표기
다음은 front[0][0]을 pop한다.

ATM	0	0	1	1	1	1
	1	0	1	0	1	0
	1	0	1	0	1	1
	1	1	1	0	1	1

dist

1	1	0	0	0	0	0
2	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

9	(1,1)	(1,2)					
---	-------	-------	--	--	--	--	--

Step 4 큐가 모두 비어있을 때까지 step 3를 계속 한다

ATM	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0

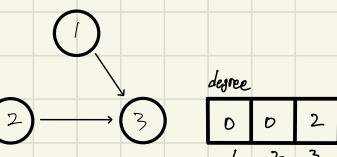
dist

1	0	9	10	11	12
2	0	8	0	12	0
3	0	7	0	13	14
4	5	6	0	14	15

9							
---	--	--	--	--	--	--	--

백준 2252, 줄 세우기

키워드 : 위상 정렬



Step1 degree 값이 0인 정점을 q에 push



Step2 큐의 front(1)에서 친일가능한 정점(3)의
degree를 1만큼 빼고, q.front.pop한다



Step3 큐의 front(2)에서 친일가능한 정점(3)의
degree를 1만큼 빼고, q.front.pop한다
이때 정점(3)의 degree가 0이므로 push한다



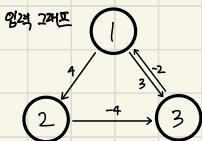
Step 5 큐의 size가 0이면 종료

9							
---	--	--	--	--	--	--	--

백준 11657, 타임머신

키워드 : 벨만 포드 알고리즘

Step 1 초기화

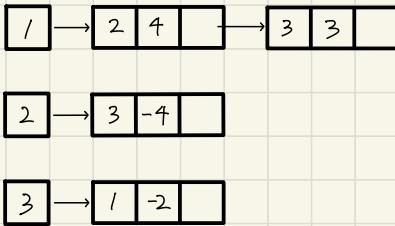


$dist[N]$: 정점 1부터 정점 N 까지 걸리는 최소시간

$$dist[i] = \begin{cases} 0 & \text{if } i=1 \\ \text{INF} & \text{otherwise} \end{cases}$$

dist	0	INF	INF
	1	2	3

Map: 정점, 고려트 일정리스트



Step 2 $dist$ 값 중 INF가 아닌 정점 1에서 찾을 수 있는 정점을 선택, $dist$ 를 업데이트하여 정진

$$\begin{aligned} \text{Case } 1 \rightarrow 2: \\ w &= 4 \\ dist[2] &= \min(dist[2], dist[1] + w) \\ &= \min(INF, 0+4) \\ &= 4 \end{aligned}$$

$$\begin{aligned} \text{Case } 1 \rightarrow 3: \\ w &= 3 \\ dist[3] &= \min(dist[3], dist[1] + w) \\ &= \min(INF, 0+3) \\ &= 3 \end{aligned}$$

dist	0	4	3
	1	2	3

Step 3 Step 2를 간선 9-1 만큼 반복

Step 4 마지막으로 한번 더 Step 2를 실행하는데 거울 같으나 꼭 아기면 음의 사이클이 존재