

백준 1005, ACM Craft

키워드 : 위상 정렬

Step 1 변수 초기화

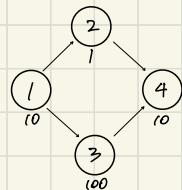
$\text{degree}[v]$: 정점 v 를 가리키는 경계 수

q : $v = 0$ 인 경계를 정렬한 Queue

$\text{node}[v]$: 정점 v 에서 건물을 만들었을 때 걸리는 시간

$\text{times}[v]$: 정점 v 에서 최종 건물 만들 완료 시간

9	1				node	10	1	100	10
	1	2	3	4		1	2	3	4
degree	0	1	1	2	times	10	1	100	10
	1	2	3	4		1	2	3	4



Step 2 q 의 front 점점 1이 가리키는

경계 $V = \{2, 3\}$ 의 degree 와 times 를 다음과 같이 정신

$\text{degree}[v] = \text{degree}[v] - 1$;

$\text{times}[v] = \max(\text{times}[v], \text{times}[0] + \text{node}[v])$;

CASE $v == 2$:

$\text{degree}[2] = \text{degree}[2] - 1$;
= 0

$\text{times}[2] = \max(\text{times}[2], \text{times}[1] + \text{node}[2])$;

= $\max(1, 10 + 1)$;

= 11

CASE $v == 3$:

$\text{degree}[3] = \text{degree}[3] - 1$;
= 0

$\text{times}[3] = \max(\text{times}[3], \text{times}[1] + \text{node}[3])$;

= $\max(1, 10 + 100)$;

= 110

9	1	2	3	
	1	2	3	4
degree	0	0	0	2

node	10	1	100	10
	1	2	3	4
times	10	11	110	10

Step 3 q 의 front 점점 2가 가리키는

경계 $V = \{4\}$ 의 degree 와 times 를 다음과 같이 정신

$\text{degree}[v] = \text{degree}[v] - 1$;

$\text{times}[v] = \max(\text{times}[v], \text{times}[2] + \text{node}[v])$;

CASE $v == 4$:

$\text{degree}[4] = \text{degree}[4] - 1$;
= 1

$\text{times}[4] = \max(\text{times}[4], \text{times}[2] + \text{node}[4])$;

= $\max(1, 11 + 10)$;

= 21

9	2	3		
	1	2	3	4
degree	0	0	0	1
	1	2	3	4

node	10	1	100	10
	1	2	3	4
times	10	11	110	21

Step 4 q 의 front 점점 3이 가리키는

경계 $V = \{4\}$ 의 degree 와 times 를 다음과 같이 정신

$\text{degree}[v] = \text{degree}[v] - 1$;

$\text{times}[v] = \max(\text{times}[v], \text{times}[3] + \text{node}[v])$;

CASE $v == 4$:

$\text{degree}[4] = \text{degree}[4] - 1$;
= 0

$\text{times}[4] = \max(\text{times}[4], \text{times}[3] + \text{node}[4])$;

= $\max(21, 110 + 10)$;

= 120

9	3			
	1	2	3	4
degree	0	0	0	0
	1	2	3	4

node	10	1	100	10
	1	2	3	4
times	10	11	110	120

백준 11404, 플로이드

키워드 : 플로이드-와샬 알고리즘

인접행렬

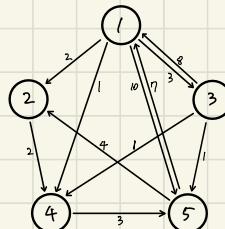
	1	2	3	4	5
1	0	2	3	1	10
2	INF	0	INF	2	INF
3	8	INF	0	1	1
4	INF	INF	INF	0	3
5	7	4	INF	INF	0

$d^{(k=0)}$

	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

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

$k=1$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned} d_{34}^{(k=1)} &= \min(d_{34}^{(k=0)}, d_{31}^{(k=0)} + d_{34}^{(k=0)}) \\ &= \min(1, 8+1)=1 \end{aligned}$$

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned} d_{45}^{(k=1)} &= \min(d_{45}^{(k=0)}, d_{41}^{(k=0)} + d_{45}^{(k=0)}) \\ &= \min(3, INF+10)=13 \end{aligned}$$

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

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

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

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

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

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

$$\begin{aligned} d_{54}^{(k=1)} &= \min(d_{54}^{(k=0)}, d_{51}^{(k=0)} + d_{54}^{(k=0)}) \\ &= \min(INF, 7+1)=8 \end{aligned}$$

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

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

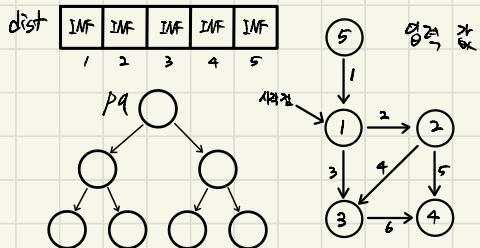
백준 1753, 최단 경로

키워드 : 다익스트라 알고리즘

Step 1. 변수 초기화

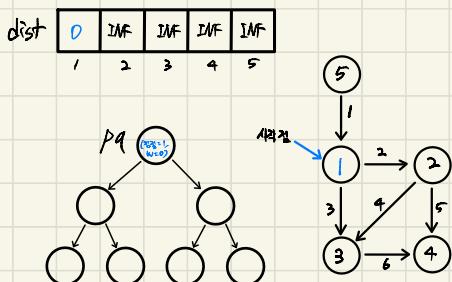
dist: 시작점부터 최단 거리를 저장하는 배열

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



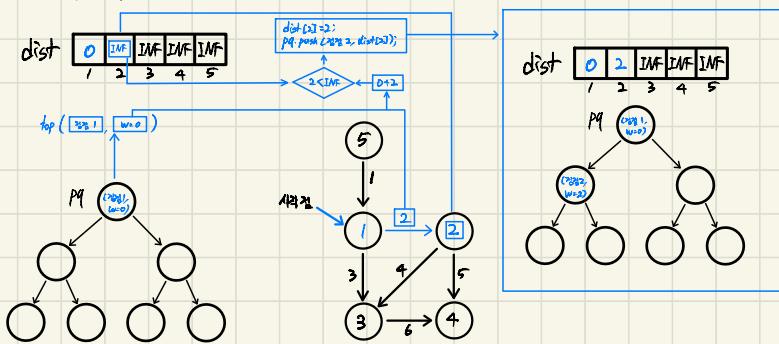
Step 2. 시작점 1의 dist를 0으로 초기화

PQ에 간선 정보 (경유지, w=0) push



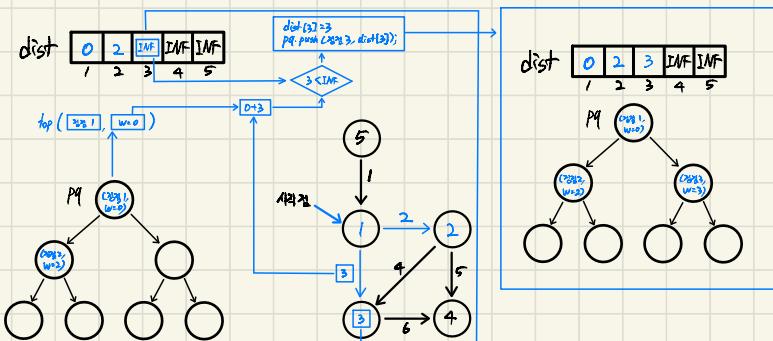
Step 3-1. PQ의 top ([경유지], [w=0])의 경유지에서 조속히는 경유지 2를 끌어

미래와 같이 dist를 갱신한다



Step 3-2. PQ의 top ([경유지], [w=0])의 경유지에서 조속히는 경유지 3을 끌어

미래와 같이 dist를 갱신한다



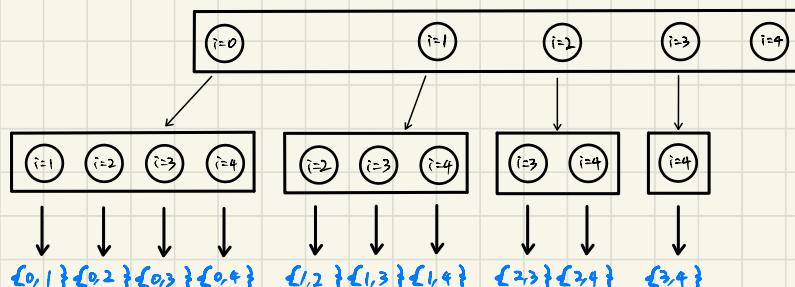
Step 4. PQ가 모두 비어질 때까지 step3 반복

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

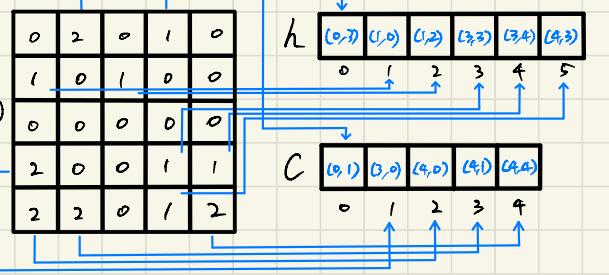
ARR : C 의 모든 조합의 원소로 배열

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

DATA: 5시작 치킨집의 좌표값 (총 4개)

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

ARR [0,1] [0,2] [0,3] [0,4] [1,2] [1,3] [1,4] [2,3] [2,4] [3,4]



step 2 ARR 전체를 돌면서 최소값을 찾는다.

pseudo code

```

int out = INT_MAX; // 출력값 초기화
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 값을 변수
        // 예전에 만난 친구면 경비 0
        for(int k=0; k<ARR[i].size(); )
        {
            pair<int, int> h-point = h[j]; // 집 위치
            pair<int, int> c-point = C[ARR[i][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;
        j++;
    }
    out = min(out, sum); // 출력 값 갱신
}
    
```

ARR [0,1] [0,2] [0,3] [0,4] [1,2] [1,3] [1,4] [2,3] [2,4] [3,4]

h [0,1] [1,0] [1,2] [3,2] [3,4] [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

```

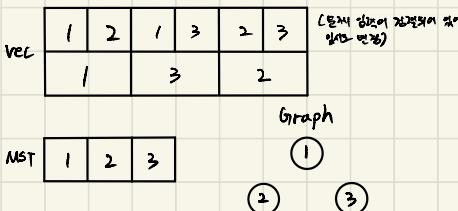
int find(int a) // 서로 다른 원인
{
    if(MST[a]==a)
        return MST[a];
    else
        return MST[a] = find(MST[a]);
}

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
}

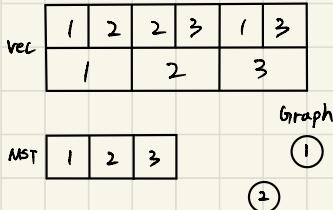
```

Step 1. 연결 초기화

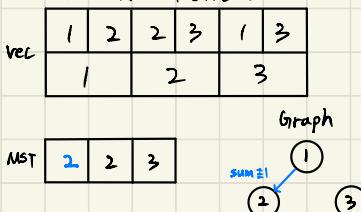
NPV : 최소 스패닝 트리 대상
VEC : 모든 정점 번호들인 Vector
SUM : 최소 스패닝 트리 가중치



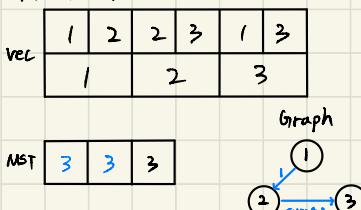
Step 2 모든 간선을 가중치 오름차순으로 정렬



Step 3 모든 VEC[i]에서 차집합(각 2를 Union-Find 알고리즘으로 연결한다)
이후 sum에 1→2 이 차집합 1을 더한다



Step 4 그 다음, VEC[i] MST로는 차집합을 Union-Find 알고리즘으로 연결한다

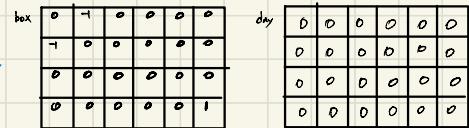


Step 5 마지막, VEC[i] 사이에 차집합(가 3의 주자가 같으므로 계산하지 않는다).

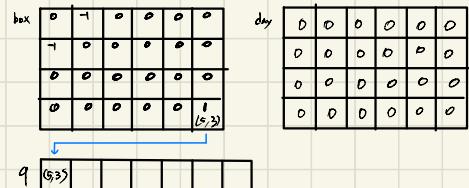
백준 7576, 토마토

키워드 : BFS

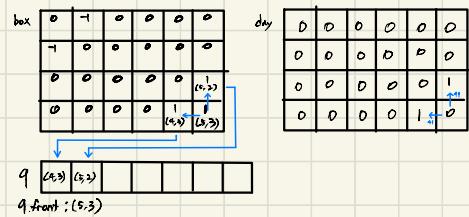
Step 1. 연결 초기화
days : 토마토가 익어가는 날짜
box : 토마토가 있는 상자
q : 토마토가 있는 상자 큐



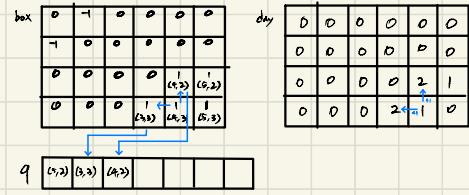
Step 2. box 배열을 확인하여 1인 곳에 좌표를 큐에 push



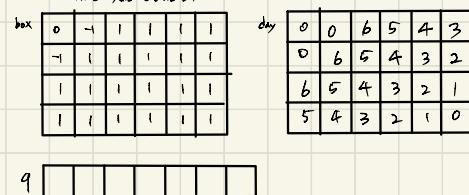
Step 3. 큐에서 꺼내고, 현재 상자에 속하는 모든
상자를 연결하고, 큐에 해당 상자 + 1개 상자를 추가



Step 4. step 3를 반복, 큐가 빈 상태가 되면 끝



Step 5. 큐를 빼면, box에 있는 모든 토마토를 계산하는 과정
큐에 있는 모든 day를 큐에 넣는다.



백준 2178, 미로 탐색

키워드 : BFS

Step1 맵과 초기화

ATM : 본래 칸의 상태를 담은 배열 (여기 풀어야 하는 칸 (:이동 가능한 칸))

dist : 시작 위치에서 출발지까지 거리로 초기화되는 배열 (여기 초기화)

q : 찾을 경로 좌표를 담은 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
출발점에서 목적점으로 이동경로는?

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

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 경우에는 경계 칸들을 큐에 push 한다.
여기서 dist[4][4] = dist[4][front[2]] + 1로 표기
다음은 front[2]를 pop한다.

ATM	0	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

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 3을 계속한다

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	

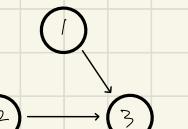
dist

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



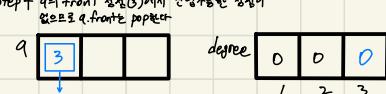
Step2 q의 front(1)에서 칸팅 가능한 정점(3)의
degree를 1만큼 빼고, q.front()을 pop한다



Step3 q의 front(2)에서 칸팅 가능한 정점(3)의
degree를 1만큼 빼고, q.front()을 pop한다
이때 정점(3)의 degree가 0이므로 push한다



Step4 q의 front(3)에서 칸팅 가능한 정점이
없으므로 q.front()을 pop한다



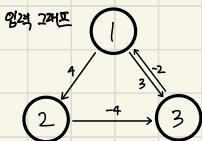
Step 5 q의 size가 0이면 종료



백준 11657, 타임머신

키워드 : 벨만 포드 알고리즘

Step 1 초기화

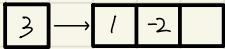


$dist[N]$: 정점 1부터 정점 N 까지 걸리는 최소시간

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

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

case 1 → 2;
 $w = 4$
 $dist[2] = \min(dist[2], dist[1] + w)$
 $= \min(INF, 0 + 4)$
 $= 4$

case 1 → 3;
 $w = 3$
 $dist[3] = \min(dist[3], dist[1] + w)$
 $= \min(INF, 0 + 3)$
 $= 3$

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 아지면 음의 사이클이 존재