

백준 11404, 플로이드

키워드 : 플로이드-와샬 알고리즘

인접행렬

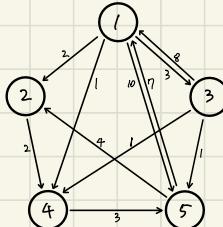
	1	2	3	4	5
1	0	2	3	1	10
2	INF	0	INF	2	INF
3	8	INF	0	1	1
4	INF	INF	INF	0	3
5	7	4	INF	INF	0

$d^{(k=0)}$

	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



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

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

π : 직전 경계점 배열

① 경계 i 를 3 사이에 경로가 존재하지 않으면 $\pi_{ij}=NIL$
 ② π_{ij} 는 경점 j 의 직전 경점이다

$k=1$ 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

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

$k=1$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned} d_{45}^{(k=1)} &= \min(d_{45}^{(k=0)}, d_{41}^{(k=0)} + d_{45}^{(k=0)}) \\ &= \min(1, INF+10)=11 \end{aligned}$$

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

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

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

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

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

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

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

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

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

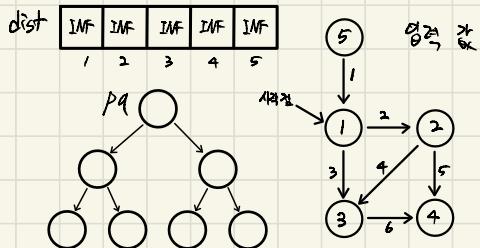
백준 1753, 최단 경로

키워드 : 다익스트라 알고리즘

Step 1. 변수 초기화

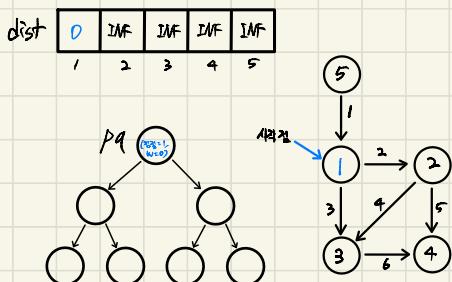
dist: 시작점부터 최단 거리를 저장하는 배열

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



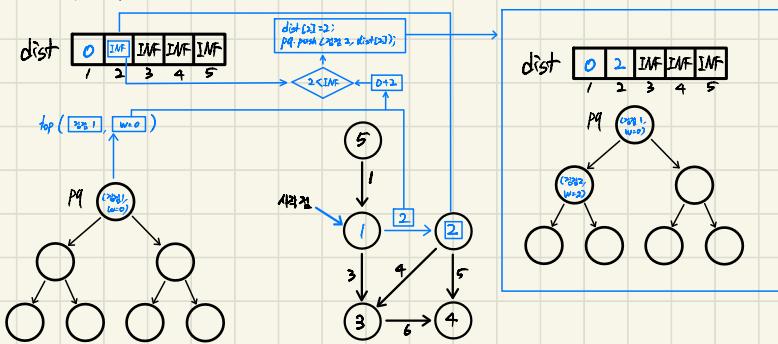
Step 2. 시작점 1의 dist를 0으로 초기화

PQ에 간선 정보 (경유지, w=0) push



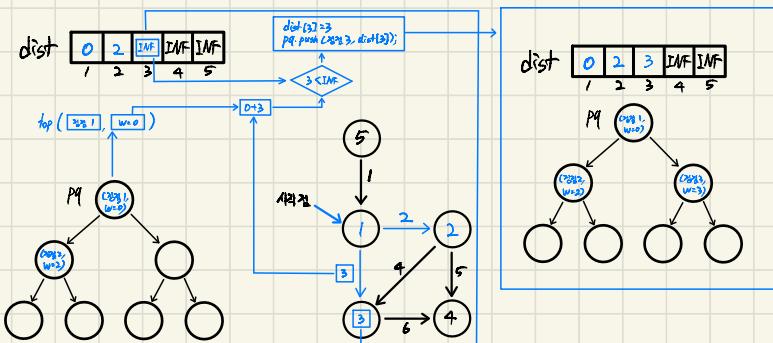
Step 3-1. PQ의 top ([노드, w=0])의 경유지에서 조속히는 경유지를 끌어온다

마지막 경이 INF를 생성한다



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

마지막 경이 INF를 생성한다



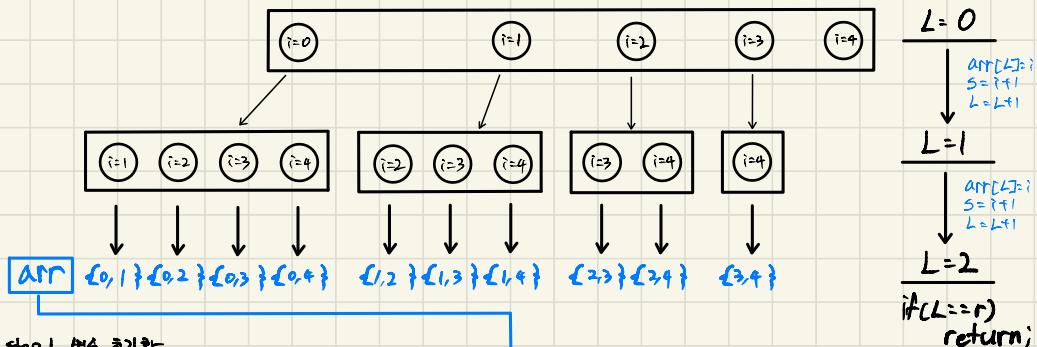
Step 4. PQ가 모두 비어질 때까지 step3 반복

Result: dist [0 2 3 7 INF]

백준 15686, 치킨 배달

키워드 : 조합 구하기, DFS

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



step 1 변수 초기화

h : 집의 위치를 담은 vector

C : 치킨집의 위치를 담은 vector

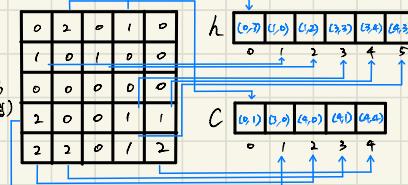
arr : C의 모든 조합의 인덱스 배열

(지금은 경비를 위해 예상 점수로 바꾸면 것이고,
실제는 다른 점수를 DFS에서 따로로 계산)

out: 5시작 치킨집의 최솟값(출발점)

(0: 본인, 1: 경, 2: 치킨집)
집(집):

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 돌아서 최소값을 찾는다.

pseudo code

```

int out = INT_MAX; // 출발점 초기화
for(int i=0; i<arr.size(); i++) // 전체 시작 순회
{
    int sum=0; // 현재 조합의 가치의 총
    // 정복 4
    for(int j=0; j<arr[i].size(); j++)
    {
        int min_dist = INT_MAX; // 각 치킨집까지 거리를 만족
        // 충족 시켜야 하는 조건은 4
        for(int k=0; k<arr[i][j].size(); k++)
        {
            pair<int, int> h-point = arr[i][j][k]; // 집 위치
            pair<int, int> C-point = arr[i][j][k]; // 치킨집 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); // 최소 거리 갱신
        }
        sum += min_dist;
    }
    out = min(out, sum); // 출발 값 갱신
}
    
```

arr : $\{0, 1\} \{0, 2\} \{0, 3\} \{0, 4\} \{1, 2\} \{1, 3\} \{1, 4\} \{2, 3\} \{2, 4\} \{3, 4\}$

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

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

백준 1197, 최소 스패닝 트리

키워드 : 최소 스패닝 트리, Union-Find

Union-Find 알고리즘 / pseudo-code

```

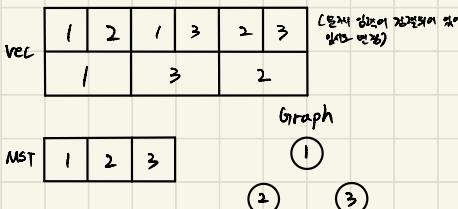
int find(int a) // 서로 다른 원인
{
    if(MST[a]==a)
        return MST[a];
    else
        return MST[a] = find(MST[a]);
}

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
    }
}

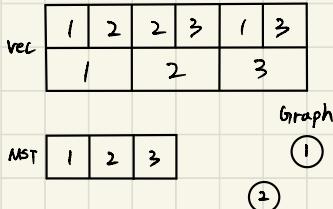
```

Step 1. 연결 초기화

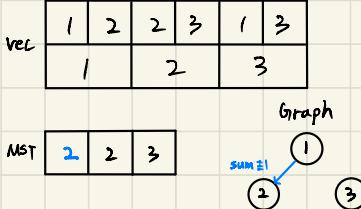
NPV : 최소 스패닝 트리 대상
VEC : 모든 정점 번호들인 Vector
SUM : 최소 스패닝 트리 가중치



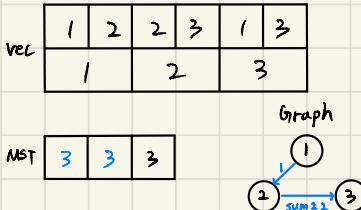
Step 2 모든 간선을 가중치 오름차순으로 정렬



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



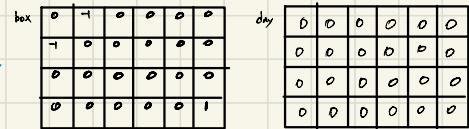
Step 4 그 다음, Vec[1] MST로는 개별로 Union-Find 알고리즘으로 연결한다



백준 7576, 토마토

키워드 : BFS

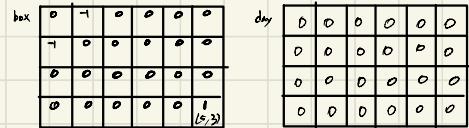
Step1 토마토 배달
days : 토마토 배달에 걸리는 일정
box : 상자 번호와 토마토 수
q : 토마토 배달 상자 번호 큐



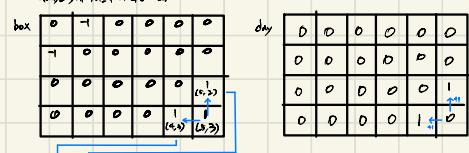
q:

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

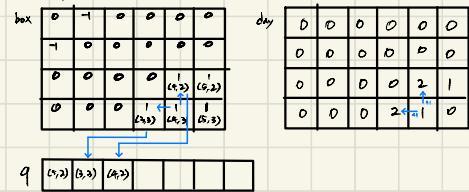
Step2 box 배열을 확인하여 1인 곳의 좌표를 큐에 push



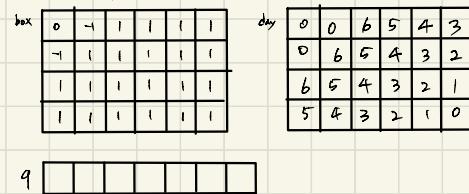
Step3 큐에서 제거한 위치의 상하좌우 4방향을 체크
상자 번호에 해당하는 큐에 넣기
상자 번호가 0이면 큐에 +1씩 더해주기



Step4 step3를 반복, 큐의 사이즈가 0이 되면 멈춰



Step5 큐에 있는 box[0][0]에서 정점(가 3의 주자가 걸려있어 계산하지 않는다.)



step5 마지막 Vec[0][0]에서 정점(가 3의 주자가 걸려있어 계산하지 않는다.)

백준 2178, 미로 탐색

키워드 : BFS

Step1 맵을 초기화

단계 1: 천적 진을 살피자. 같은 배열 (여기 풀기 가능한 칸, (/)이 있는 칸은 끝)

단계 2: 시작점에서 천적 진을 지나가는 배울 (여기 허용)

단계 3: 끝으로 향하는 경로를 같은 queue

arr	<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr></table>	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	
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

dist	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	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
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	<table border="1"><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>							

Step 2 시작점 (1,1)을 큐에 push 한다

큐에 있는 천적 진은 여기도 탐색한다

arr	<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr></table>	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	
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																								
push	<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr></table>	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	
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																								

9	<table border="1"><tr><td>(1,1)</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	(1,1)						
(1,1)								

dist

dist	<table border="1"><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	1	0	0	0	0	0	0	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
1	0	0	0	0	0	0																														
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																														

arr	<table border="1"><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr></table>	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	
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																								
push	<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td></td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr></table>	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	
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																								

9	<table border="1"><tr><td>(1,1) (2,2)</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	(1,1) (2,2)						
(1,1) (2,2)								

Step 3 arr에서 천적 진을 찾을 수 있는 경로의 거리를 사이 1이나 한다

단계 4 arr에서 찾은 천적 진은 dist[front]에 1로 한다

단계 5 arr에서 찾은 천적 진은 arr[front]에 pop된다

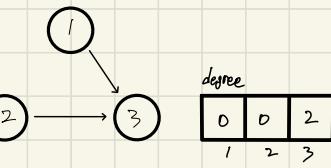
dist

dist	<table border="1"><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>2</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	1	0	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
1	0	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	<table border="1"><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>							

백준 2252, 줄 세우기

키워드 : 위상 정렬



Step1 degree 값이 0인 정점을 q에 push



Step2 q의 front 정점(1)에서 친일가능한 정점(3)의 degree를 1만큼 뺀다. q.front()를 pop한다



Step3 q의 front 정점(2)에서 친일가능한 정점(3)의 degree를 1만큼 뺀다. q.front()를 pop한다. 정점(3)의 degree가 0이므로 push한다



Step4 q의 front 정점(3)에서 친일가능한 정점이 없으므로 q.front()를 pop한다



Step5 q가 모두 비워졌을 때까지 Step 3을 계속한다

arr	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	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																							
0	0	0	0	0	0	0																							
0	0	0	0	0	0	0																							
0	0	0	0	0	0	0																							
push	<table border="1"><tr><td>1</td><td>0</td><td>9</td><td>10</td><td>11</td><td>12</td><td></td></tr><tr><td>2</td><td>0</td><td>8</td><td>0</td><td>12</td><td>0</td><td></td></tr><tr><td>3</td><td>0</td><td>7</td><td>0</td><td>13</td><td>14</td><td></td></tr><tr><td>4</td><td>5</td><td>6</td><td>0</td><td>14</td><td>15</td><td></td></tr></table>	1	0	9	10	11	12		2	0	8	0	12	0		3	0	7	0	13	14		4	5	6	0	14	15	
1	0	9	10	11	12																								
2	0	8	0	12	0																								
3	0	7	0	13	14																								
4	5	6	0	14	15																								

dist

dist	<table border="1"><tr><td>1</td><td>0</td><td>9</td><td>10</td><td>11</td><td>12</td><td></td></tr><tr><td>2</td><td>0</td><td>8</td><td>0</td><td>12</td><td>0</td><td></td></tr><tr><td>3</td><td>0</td><td>7</td><td>0</td><td>13</td><td>14</td><td></td></tr><tr><td>4</td><td>5</td><td>6</td><td>0</td><td>14</td><td>15</td><td></td></tr></table>	1	0	9	10	11	12		2	0	8	0	12	0		3	0	7	0	13	14		4	5	6	0	14	15	
1	0	9	10	11	12																								
2	0	8	0	12	0																								
3	0	7	0	13	14																								
4	5	6	0	14	15																								

백준 11675, 탐색 알고리즘

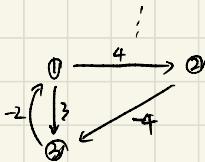
기회드

벨만-포드 알고리즘

정렬

3(정점 S) 4(전선 A)

1(정점 A) 2(정점 B) 4(가중치)



step 1 dist 배열 (1부터 각 정점 까지 걸리는 시간)을 무한 대로 초기화

INF	INF	INF
1	2	3

step 2 dist[1(정점 A), 1]은 0으로 초기화

0	INF	INF
1	2	3

step 3 간선 수 - 1 × 정점 수 만큼 거리를 초기화 가중치를 더한 값을 차운 것으로 계속 업데이트
문, dist가 INF는 몇 수 없으므로 계산이 끝나면 제외

0	INF	INF
1	2	3

 \Rightarrow

0	4	INF
1	2	3

 \Rightarrow

0	4	3
1	2	3

step 4 마지막으로 정점 수 만큼 업데이트 하는데 기존값보다 값이 적으면 사이를 더하거나

0	4	3
1	2	3

 \Rightarrow

0	0	3
1	2	3

같이 초기화하고 초기 사이를 끝내기