백준 11404, 플로이드

인접행렬

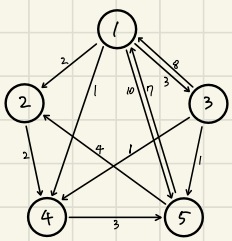
	1	2	3	4	5
1	0	2	3	1	10
2	INF	0	INF	2	INF
3	8	INF	0	1	1
4	INF	INF	INF	0	3
5	7	4	INF	INF	0

$d^{(k=0)}$

	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



$d_{ij}^{(k)}$: 정점 $i \to j$ 까지 최단경로

$\{1,2,3,\cdots,k\}$: $i \to j$ 까지의 출발 경로

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

Π : 직전 정점행렬

① 정점 i 와 j 사이에 경로가 존재하지 않으면 $\pi_{ij} = NIL$

② π_{ij} 는 정점 j 의 직전 정점 이다

$k=1$ 일 때 d, Π 값 변화

$d^{(k=0)}$

	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

$d^{(k=1)}$

	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

$\Pi^{(k=1)}$

	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

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

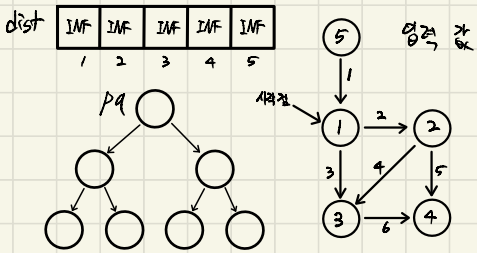
$d_{11}^{(k=1)} = \min(d_{11}^{(0)}, d_{11}^{(0)}+d_{11}^{(0)}) = \min(0, 0+0) = 0$	$d_{12}^{(k=1)} = \min(d_{12}^{(0)}, d_{11}^{(0)}+d_{12}^{(0)}) = \min(2, 0+2) = 2$	$d_{13}^{(k=1)} = \min(d_{13}^{(0)}, d_{11}^{(0)}+d_{13}^{(0)}) = \min(3, 0+3) = 3$	$d_{14}^{(k=1)} = \min(d_{14}^{(0)}, d_{11}^{(0)}+d_{14}^{(0)}) = \min(1, 0+1) = 1$	$d_{15}^{(k=1)} = \min(d_{15}^{(0)}, d_{11}^{(0)}+d_{15}^{(0)}) = \min(10, 0+10) = 10$
$d_{21}^{(k=1)} = \min(d_{21}^{(0)}, d_{21}^{(0)}+d_{11}^{(0)}) = \min(INF, INF+0) = INF$	$d_{22}^{(k=1)} = \min(d_{22}^{(0)}, d_{21}^{(0)}+d_{12}^{(0)}) = \min(0, INF+2) = 0$	$d_{23}^{(k=1)} = \min(d_{23}^{(0)}, d_{21}^{(0)}+d_{13}^{(0)}) = \min(INF, INF+3) = INF$	$d_{24}^{(k=1)} = \min(d_{24}^{(0)}, d_{21}^{(0)}+d_{14}^{(0)}) = \min(2, INF+1) = 2$	$d_{25}^{(k=1)} = \min(d_{25}^{(0)}, d_{21}^{(0)}+d_{15}^{(0)}) = \min(INF, INF+10) = INF$
$d_{31}^{(k=1)} = \min(d_{31}^{(0)}, d_{31}^{(0)}+d_{11}^{(0)}) = \min(8, 8+0) = 8$	$d_{32}^{(k=1)} = \min(d_{32}^{(0)}, d_{31}^{(0)}+d_{12}^{(0)}) = \min(INF, 8+2) = 10$	$d_{33}^{(k=1)} = \min(d_{33}^{(0)}, d_{31}^{(0)}+d_{13}^{(0)}) = \min(0, 8+3) = 0$	$d_{34}^{(k=1)} = \min(d_{34}^{(0)}, d_{31}^{(0)}+d_{14}^{(0)}) = \min(1, 8+1) = 1$	$d_{35}^{(k=1)} = \min(d_{35}^{(0)}, d_{31}^{(0)}+d_{15}^{(0)}) = \min(1, 8+10) = 1$
$d_{41}^{(k=1)} = \min(d_{41}^{(0)}, d_{41}^{(0)}+d_{11}^{(0)}) = \min(INF, INF+0) = INF$	$d_{42}^{(k=1)} = \min(d_{42}^{(0)}, d_{41}^{(0)}+d_{12}^{(0)}) = \min(INF, INF+2) = INF$	$d_{43}^{(k=1)} = \min(d_{43}^{(0)}, d_{41}^{(0)}+d_{13}^{(0)}) = \min(INF, INF+3) = INF$	$d_{44}^{(k=1)} = \min(d_{44}^{(0)}, d_{41}^{(0)}+d_{14}^{(0)}) = \min(0, INF+1) = 0$	$d_{45}^{(k=1)} = \min(d_{45}^{(0)}, d_{41}^{(0)}+d_{15}^{(0)}) = \min(3, INF+10) = 3$
$d_{51}^{(k=1)} = \min(d_{51}^{(0)}, d_{51}^{(0)}+d_{11}^{(0)}) = \min(7, 7+0) = 0$	$d_{52}^{(k=1)} = \min(d_{52}^{(0)}, d_{51}^{(0)}+d_{12}^{(0)}) = \min(4, 7+2) = 4$	$d_{53}^{(k=1)} = \min(d_{53}^{(0)}, d_{51}^{(0)}+d_{13}^{(0)}) = \min(INF, 7+3) = 10$	$d_{54}^{(k=1)} = \min(d_{54}^{(0)}, d_{51}^{(0)}+d_{14}^{(0)}) = \min(INF, 7+1) = 8$	$d_{55}^{(k=1)} = \min(d_{55}^{(0)}, d_{51}^{(0)}+d_{15}^{(0)}) = \min(0, 7+10) = 0$

백준 1753, 최단 경로

Step1 변수 초기화

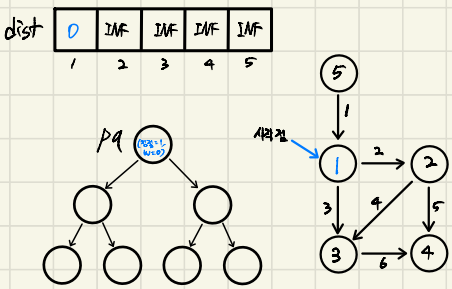
dist : 시작점부터 최소거리를 저장하는 배열

pq : 현재 정점에서 최소 거리로 갈 수 있는 간선을 저장하는 priority_queue (가중치 최소힙)

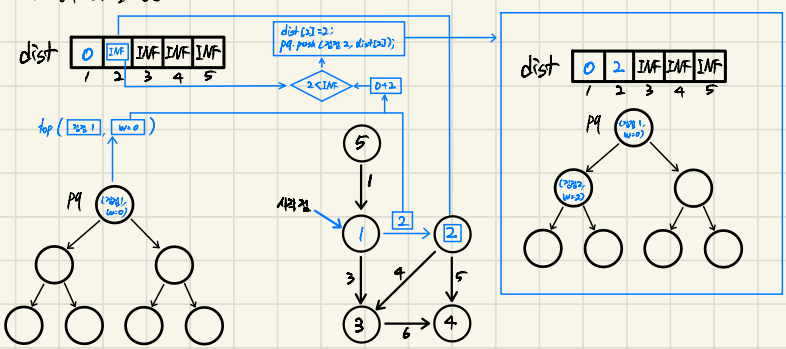


Step2 시작점 1의 dist를 0으로 초기화

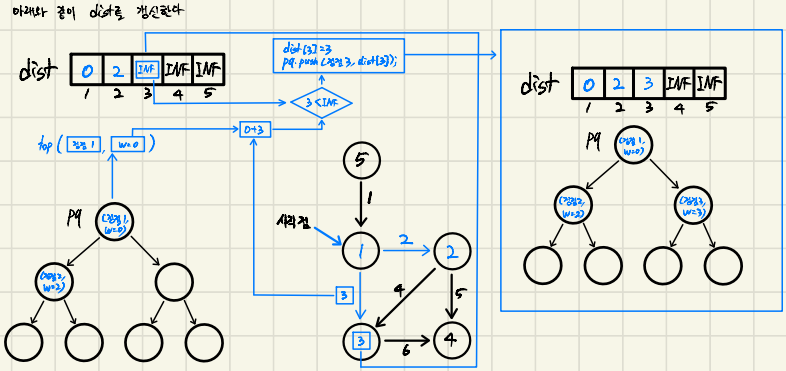
pq에 간선 정보 (정점:1, w:0) push



Step3-1 pq의 top (정점 1, w:0)의 정점 1에서 갈 수 있는 정점 2를 골라 아래와 같이 dist를 갱신한다



Step3-2 pq의 top (정점 1, w:0)의 정점 1에서 갈 수 있는 정점 3를 골라 아래와 같이 dist를 갱신한다

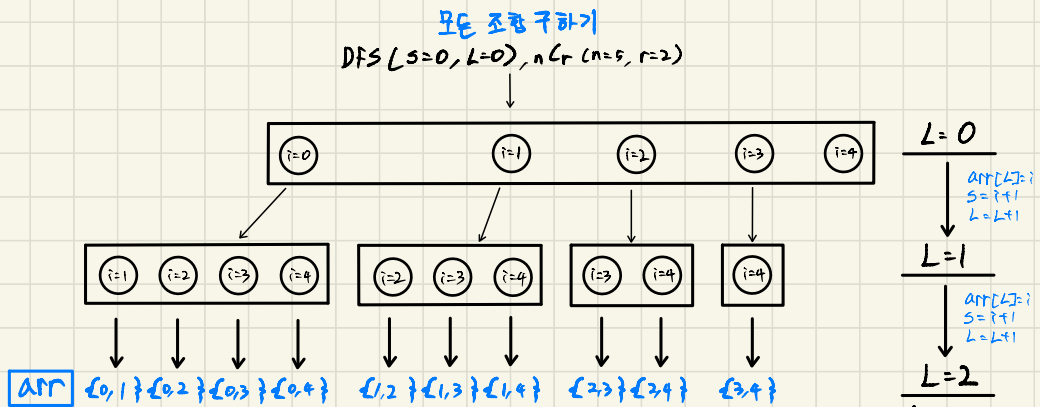


Step4 pq가 모두 비워질 때까지 Step3 반복

Result : dist

0	2	3	7	INF
1	2	3	4	5

백준 15686, 치킨 배달



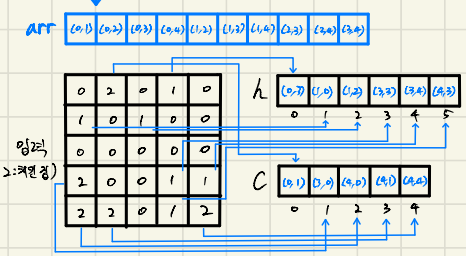
step 1 변수 초기화

h : 집의 위치를 담은 vector

C : 치킨집의 위치를 담은 vector

arr : C의 모든 조합의 인덱스 배열
(지금은 집과 치킨집 배열 형태로 나타낸 것이고, 실제는 나온 조합을 DFS에서 index로 계산)

out : 도시의 치킨집의 최솟값(출력 값)

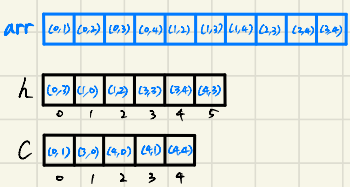


step 2 arr 전체를 돌면서 최소 값을 찾는다.

pseudo code

```
int out=INT_MAX; // 출력값 초기화
for(int i=0; i<arr.size(); ) // 전체 arr 순회
{
    int sum=0 // 현재 조합의 거리의 합
    // 집의 수
    for(int j=0; j<h.size(); )
    {
        int min_dist=INT_MAX; // 각 집마다 최소거리 담을 변수
        // 치킨집의 index 조합의 수
        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]] // 치킨 집 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); // 최소거리 갱신
        }
    }
    out = min(out, sum); // 출력 값 갱신
}
```



백준 1197, 최소 스패닝 트리

step1 변수초기화
MST : 최소 스패닝 트리 배열
vec : 모든 간선 정보들어간 vector
sum : 최소 스패닝 트리 가중치

vec

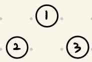
1	2	1	3	2	3
1		3		2	

(문제시 입력이 정렬되어 있어 임시로 변경)

MST

1	2	3

Graph



step2 모든 간선을 가중치 오름차순으로 정렬

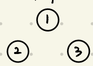
vec

1	2	2	3	1	3
1		2		3	

MST

1	2	3

Graph



step3 먼저, vec[0]에서 정점 1과 2를 Union-Find 알고리즘으로 연결한다
이때, sum에 1→2 의 가중치 1을 더한다

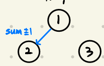
vec

1	2	2	3	1	3
1		2		3	

MST

2	2	3

Graph



Union-Find 알고리즘

```
int Find(int a) // a의 루트 찾기
{
    if(MST[a]==a)
    {
        return MST[a];
    }
    return MST[a] = Find(MST[a]);
}

void Union(int a, int b)
{
    a=Find(a);
    b=Find(b);
    if(a!=b) // a와 b가 연결 되지 않았으면
    {
        MST[a]=b; // a와 b를 연결
    }
}
```

step4 그다음, vec[1]에서도 두 정점을 Union-Find 알고리즘으로 연결한다

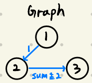
vec

1	2	2	3	1	3
1		2		3	

MST

3	3	3

Graph



step4 마지막 vec[2]에서 정점 1과 3의 루트가 같으므로 계산하지 않는다.

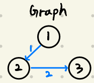
vec

1	2	2	3	1	3
1		2		3	

MST

3	3	3

Graph



백준 7576, 토마토

step1 변수 초기화
day : 토마토가 익는 데 걸리는 시간, 0으로 초기화
box : 초기 토마토 상태, 문제 입력
q : 검사할 토마토 좌표 값을 담는 Queue

box

0	-1	0	0	0	0
-1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	1

day

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

q

step2 box 배열을 확인하여 1인 곳의 좌표를 q에 push

box

0	-1	0	0	0	0
-1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	1 (5,3)

day

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

q

(5,3)						

step3 q에서 좌표를 꺼내서 상하좌우를 검사
box 값이 0인것을 q에 넣고 값을 1로 변경
이때, day에 각 좌표에 +1씩 증가시킨다

box

0	-1	0	0	0	0
-1	0	0	0	0	0
0	0	0	0	0	1 (5,2)
0	0	0	0	1 (4,3)	1 (5,3)

day

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

q

(4,3)	(5,2)					

q.front ; (5,3)

step4 step3를 반복 (q의 size 가 0일때 까지)

box

0	-1	0	0	0	0
-1	0	0	0	0	0
0	0	0	0	1 (4,2)	1 (5,2)
0	0	0	1 (3,3)	1 (4,3)	1 (5,3)

day

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

q

(5,2)	(3,3)	(4,2)				

q.front : (4,3)

최종출력
step5 q가 모두 비워 졌을 때, box 값이 0 인곳은 갈 수없는곳이므로 -1을 출력
box에 0이 없으면 day중 가장 큰 값을 출력

box

0	-1	1	1	1	1
-1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	1	1

day

0	0	6	5	4	3
0	6	5	4	3	2
6	5	4	3	2	1
5	4	3	2	1	0

q

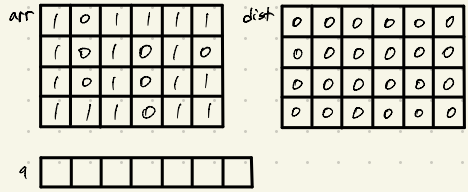
백준 2178, 미로탐색

Step1 변수 초기화

arr : 현재 칸의 상태를 담을 배열 (0: 이동할 수 있는 칸 / 1: 이동 불가능한 칸)

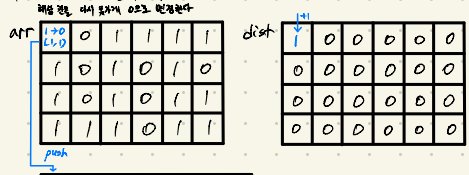
dist : 시작 위치에서 현재까지 거리를 저장하는 배열 (0으로 초기화)

q : 갈 수 있는 칸의 좌표를 담은 Queue



Step 2 시작 좌표 (1,1)를 q에 push 한다

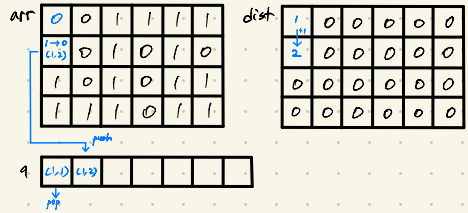
해당 좌표 dist 값을 0으로 변경한다



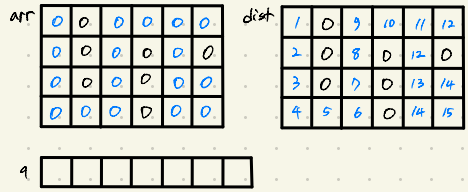
Step3 q.front의 좌표 값으로 갈 수 있는 칸의 좌표를 q에 push 한다

이때, dist[갈 수 있는 칸의 좌표] = dist[q.front 좌표]+1 을 한다.

그 좌표를 q.front 는 pop 한다.



Step4 q가 모두 비워질 때까지 Step 3을 계속한다



백준 11675, 타임머신

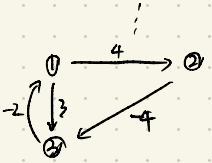
키워드

벨만-포드 알고리즘

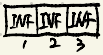
영역

3(정점 3) 4(간선 3)

1(정점 A) 2(정점 B) 4(가중치)



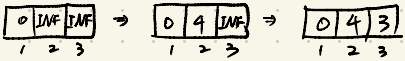
step 1 dist 배열 (1부터 n 정점 까지 갈리는 시간)을 무한대로 초기화



step 2 dist[시작정점,] 을 0으로 초기화

step 3 간선 수-1 x 정점 수 만큼 가면 dist와 가중치를 더한 값을 적은 값으로 계속 업데이트

단, dist가 INF는 갈 수 없으므로 계산에서 제외



step 4 마지막으로 정점 수 만큼 업데이트 하는데 기존보다 작아지면 사이클이 존재