25-1 이니로
알고리즘 멘토링

멘토 - 김수성

음악프로그램 / 2623

백준 2623 / https://www.acmicpc.net/problem/2623

- 1 4 3
- 6 2 5 4
- 2 3

남일이가 할 일은 이 순서들을 모아서 전체 가수의 순서를 정하는 것이다. 남일이는 잠시 생각을 하더니 6 2 1 5 4 3으로 순서를 정했다. 이렇게 가수 순서를 정하면 세 보조 PD가 정해온 순서를 모두 만족한다. 물론, 1 6 2 5 4 3으로 전체 순서를 정해도 괜찮다.

경우에 따라서 남일이가 모두를 만족하는 순서를 정하는 것이 불가능할 수도 있다. 예를 들어, 세 번째 보조 PD가 순서를 2 3 대신에 3 2로 정해오면 남일이가 전체 순서를 정하는 것이 불가능하다. 이번에 남일이는 우리 나라의 월드컵 4강 진출 기념 음악제의 PD를 맡게 되었는데, 출연 가수가 아주 많다. 이제 여러분이 해야 할 일은 보조 PD들이 가져온 순서들을 보고 남일이가 가수 출연 순서를 정할 수 있도록 도와 주는 일이다.

보조 PD들이 만든 순서들이 입력으로 주어질 때, 전체 가수의 순서를 정하는 프로그램을 작성하시오.

입력

첫째 줄에는 가수의 수 N과 보조 PD의 수 M이 주어진다. 가수는 번호 1, 2,...,N 으로 표시한다. 둘째 줄부터 각 보조 PD가 정한 순서들이 한 줄에 하나씩 나온다. 각 줄의 맨 앞에는 보조 PD가 담당한 가수의 수가 나오고, 그 뒤로는 그 가수들의 순서가 나온다. N은 1이상 1,000이하의 정수이고, M은 1이상 100이하의 정수이다.

출력

출력은 N 개의 줄로 이뤄지며, 한 줄에 하나의 번호를 출력한 다. 이들은 남일이가 정한 가수들의 출연 순서를 나타낸다. 답이 여럿일 경우에는 아무거나 하나를 출력 한다. 만약 남일이가 순서를 정하는 것이 불가능할 경우에는 첫째 줄에 0을 출력한다.

음악프로그램 / 2623

각 입력에 대해서 같은 줄에 있는 인원끼리는 순서를 지킬 때 될 수 있는 순서 중 하나를 출력 하는 문제

- 143
- 6254
- 23

1 -> 4 -> 3　　2 -> 3

6 -> 2 -> 5 -> 4

예제 입력 1 복사

```
6 3
3 1 4 3
4 6 2 5 4
2 2 3
```

예제 출력 1 복사

```
6
2
1
5
4
3
```

음악프로그램 / 2623

1 -> 4 -> 3　　2 -> 3

6 -> 2 -> 5 -> 4

같은 줄에 있는 입력끼리 간선을 만들어주면 됨

- 1 4 3
- 6 2 5 4
- 2 3

예제 입력 1 복사

```
6 3
3 1 4 3
4 6 2 5 4
2 2 3
```

예제 출력 1 복사

```
6
2
1
5
4
3
```

음악프로그램 / 2623

또한 위상 정렬이 불가능 할 때 0을 출력

순서를 정하는 것이 불가능할 경우에는 첫째 줄에 0을 출력한다.

위상 정렬이 불가능한 조건 -> 사이클 존재

사이클이 존재하면 위상 정렬이 끝났을 때 indegree가 0이 아닌 어떤 인덱스 i가 존재함

음악프로그램 / 2623

C++

```cpp
const ll MAX = 1010;
const ll INF = 1e12;
ll n, m, ind[MAX], a[MAX];
vector <ll> adj[MAX], result;
queue <ll> q;
```

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n >> m;
    while(m--){
        ll sz; cin >> sz;
        for(int i = 1;i <= sz;i++) cin >> a[i];
        for(int i = 1;i < sz;i++){
            // a[i] -> a[i + 1] 인 간선
            adj[a[i]].push_back(a[i + 1]);
            ind[a[i + 1]]++;
        }
    }

    for(int i = 1;i <= n;i++){
        if(!ind[i]) q.push(i);
    }

    // 위상정렬
    while(!q.empty()){
        ll cur = q.front(); q.pop();
        result.push_back(cur);
        for(auto& nxt : adj[cur]){
            if(!--ind[nxt]) q.push(nxt);
        }
    }

    bool flag = 1;
    for(int i = 1;i <= n;i++){
        // indegree가 0이 아닌 값이 있으면
        // 사이클이 존재 -> 위상 정렬 불가
        if(ind[i]) flag = 0;
    }

    if(flag) for(auto& i : result) cout << i << "\n";
    else cout << 0;

    return 0;
}
```

음악프로그램 / 2623

Python

```python
MAX = 1010
INF = 10**12

n, m = map(int, input().split())
ind = [0] * MAX
a = [0] * MAX
adj = [[] for _ in range(MAX)]
result = []
q = deque()
```

```python
for _ in range(m):
    data = list(map(int, input().split()))
    sz = data[0]
    for i in range(1, sz + 1):
        a[i] = data[i]

    for i in range(1, sz):
        # a[i] -> a[i + 1] 인 간선
        adj[a[i]].append(a[i + 1])
        ind[a[i + 1]] += 1

for i in range(1, n + 1):
    if ind[i] == 0:
        q.append(i)

# 위상정렬
while q:
    cur = q.popleft()
    result.append(cur)
    for nxt in adj[cur]:
        ind[nxt] -= 1
        if ind[nxt] == 0:
            q.append(nxt)

flag = 1
for i in range(1, n + 1):
    # indegree가 0이 아닌 값이 있으면
    # 사이클이 존재 -> 위상 정렬 불가
    if ind[i] != 0:
        flag = 0
        break

if flag:
    sys.stdout.write("\n".join(map(str, result)))
else:
    sys.stdout.write("0")
```

질문?

3주차 - 다익스트라

다익스트라

다익스트라
    최단 경로 알고리즘

최단 경로 알고리즘
    다익스트라, 벨만 포드, 플로이드, SPFA …

위의 알고리즘 중에서 다익스트라가 가장 많이 쓰임

다익스트라

다익스트라

최단 경로 알고리즘

어떤 그래프의 한 정점에 대해서 다른 모든 정점에 대한
최단 경로를 구할 수 있음

단 모든 간선의 비용이 음수이면 안됨
-> 시간 복잡도 보장이 안됨
-> 음수의 사이클이 존재하면 무한 루프가 돌음

다익스트라

다익스트라

0-1. 거리 배열을 큰 값으로 초기화

0-2. 시작 정점을 방문할 수 있는 정점으로 처리함

1. 방문할 수 있는 정점 중 가장 최단 거리인 정점을 방문

2. 현재 정점과 인접한 정점들을 방문함

2-1. 인접한 정점의 거리가 원래 거리보다 최단거리이면
방문할 수 있는 정점으로 처리함

2-2. 원래 거리가 인접한 정점의 거리보다 최단거리이면
이미 최단거리이므로 건너 뜀

다익스트라

다익스트라

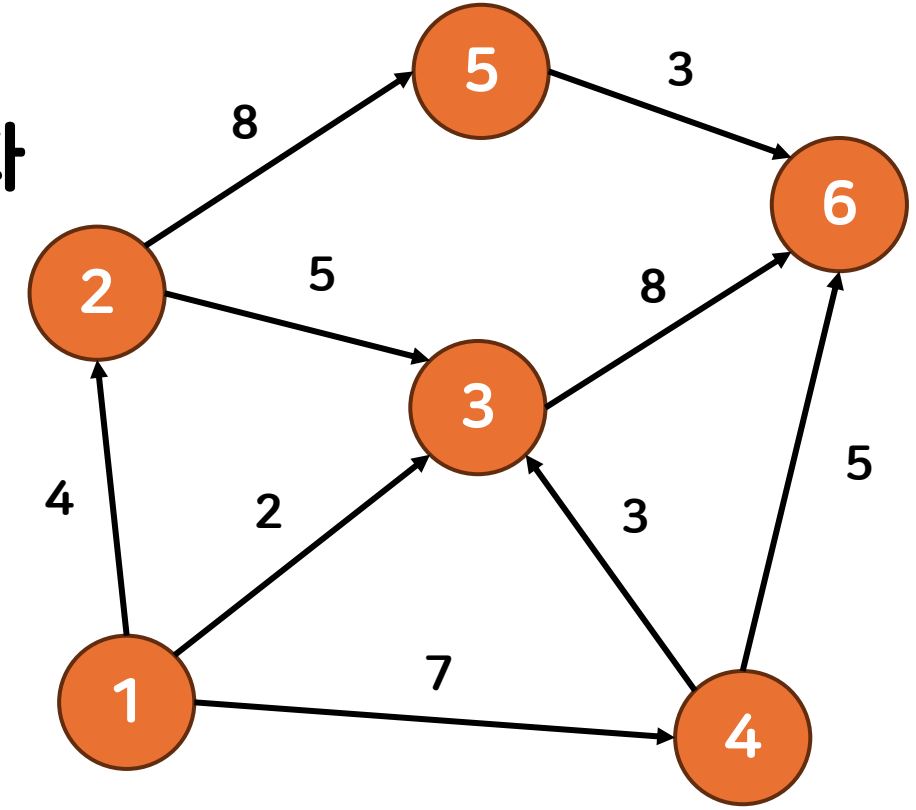
1. 방문할 수 있는 정점 중 가장 최단 거리인 정점을 방문

가장 최단 거리인 정점을 알아야 함
-> 우선순위 큐 사용

다익스트라

다익스트라

0-1. 모든 경로를 큰 값으로 초기화



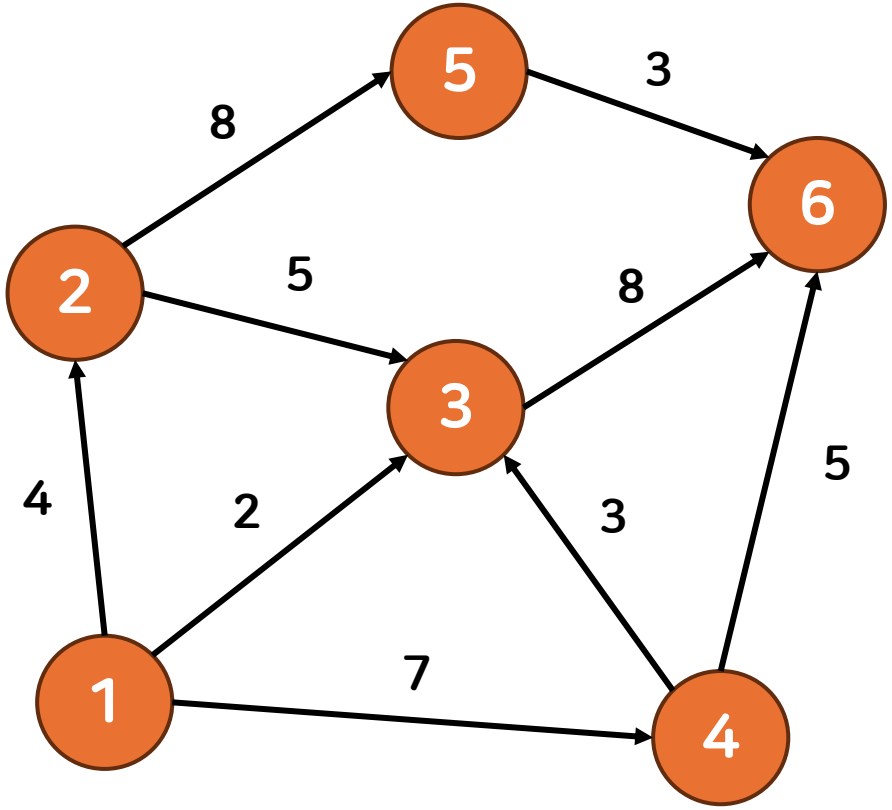
1	2	3	4	5	6
INF	INF	INF	INF	INF	INF

다익스트라

다익스트라

0-2. 시작 정점을 방문 할 수 있는 정점으로 처리함

시작 정점은 거리가 0



(0, 1)

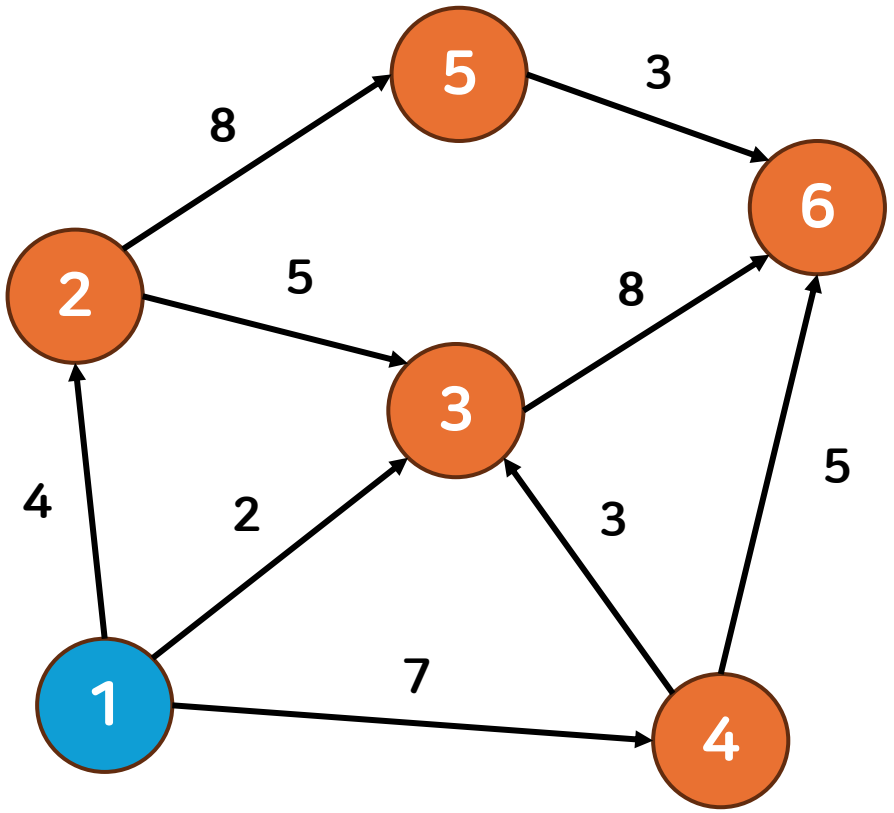
1	2	3	4	5	6
INF	INF	INF	INF	INF	INF

다익스트라

다익스트라

1. 방문할 수 있는 정점 중 가장 최단 거리인 정점을 방문

거리 배열의 값이 우선순위 큐의 값보다 작으므로 거리 갱신



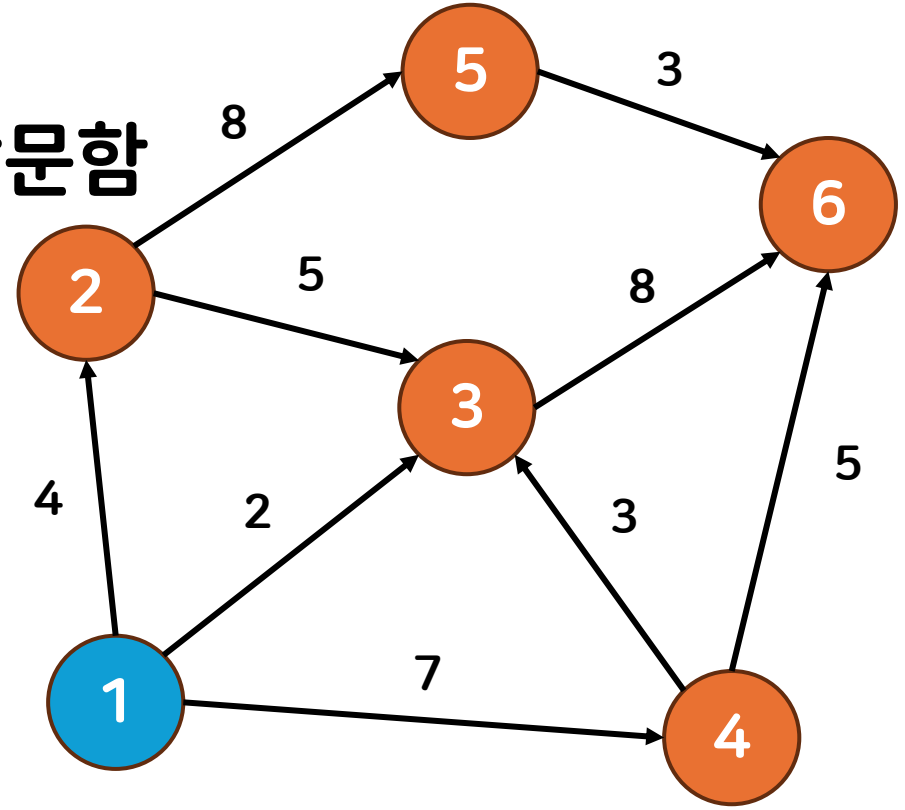
1	2	3	4	5	6
0	INF	INF	INF	INF	INF

다익스트라

다익스트라

2. 현재 정점과 인접한 정점들을 방문함

2-1. 인접한 정점의 거리가 원래 거리보다 최단거리이면 방문할 수 있는 정점으로 처리함



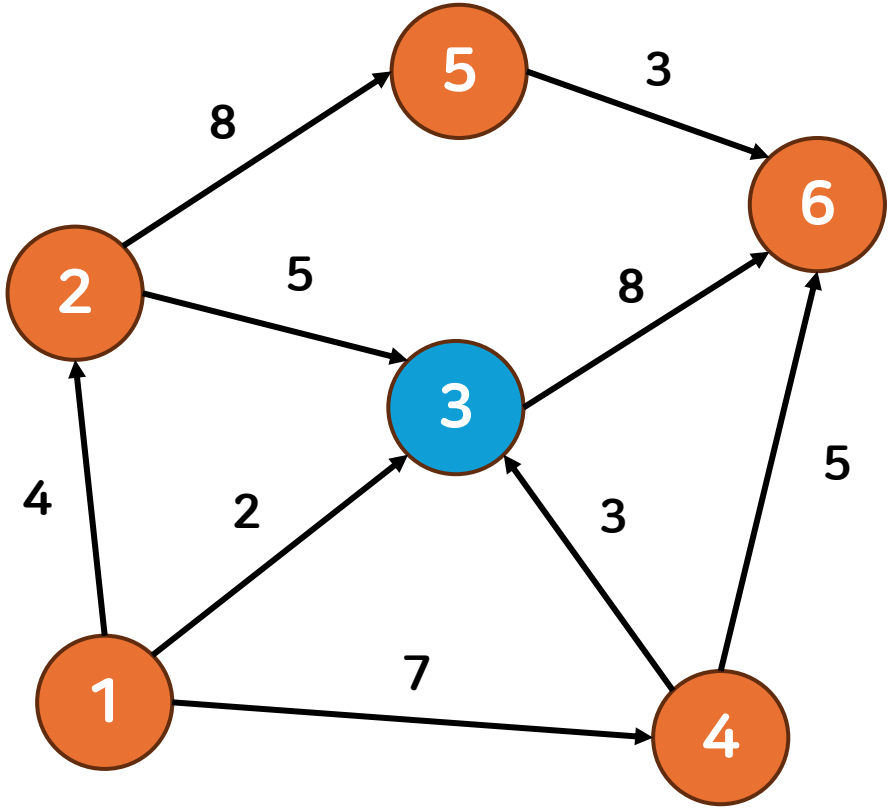
(2, 3) (4, 2) (7, 4)

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

다익스트라

다익스트라

1. 방문할 수 있는 정점 중 가장 최단 거리인 정점을 방문



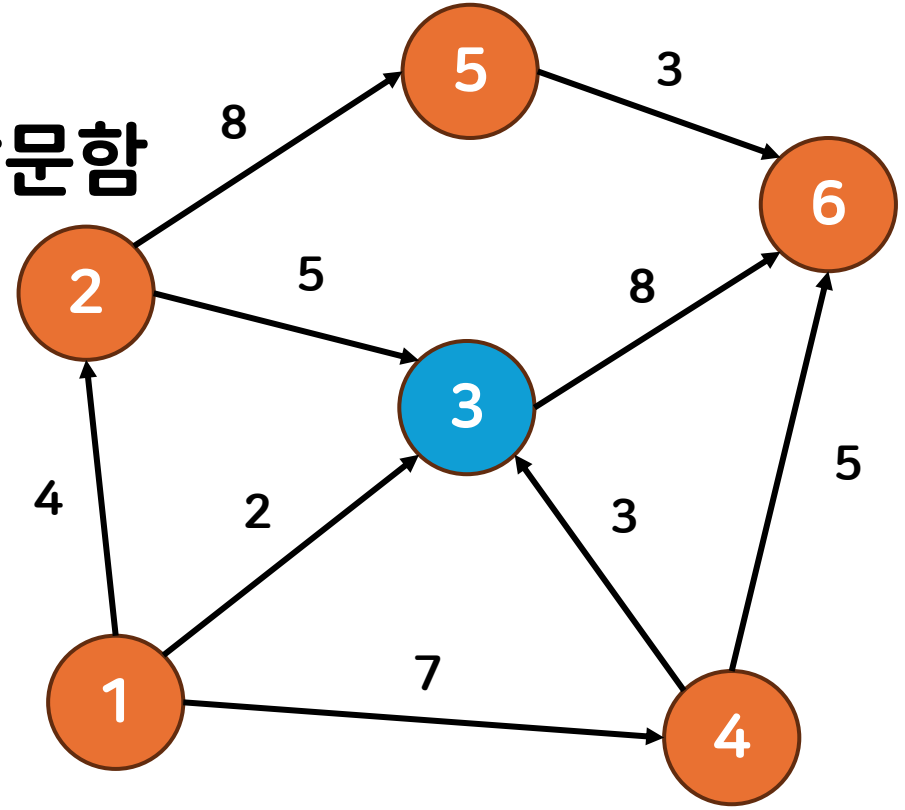
(4, 2) (7, 4)

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

다익스트라

다익스트라

2. 현재 정점과 인접한 정점들을 방문함



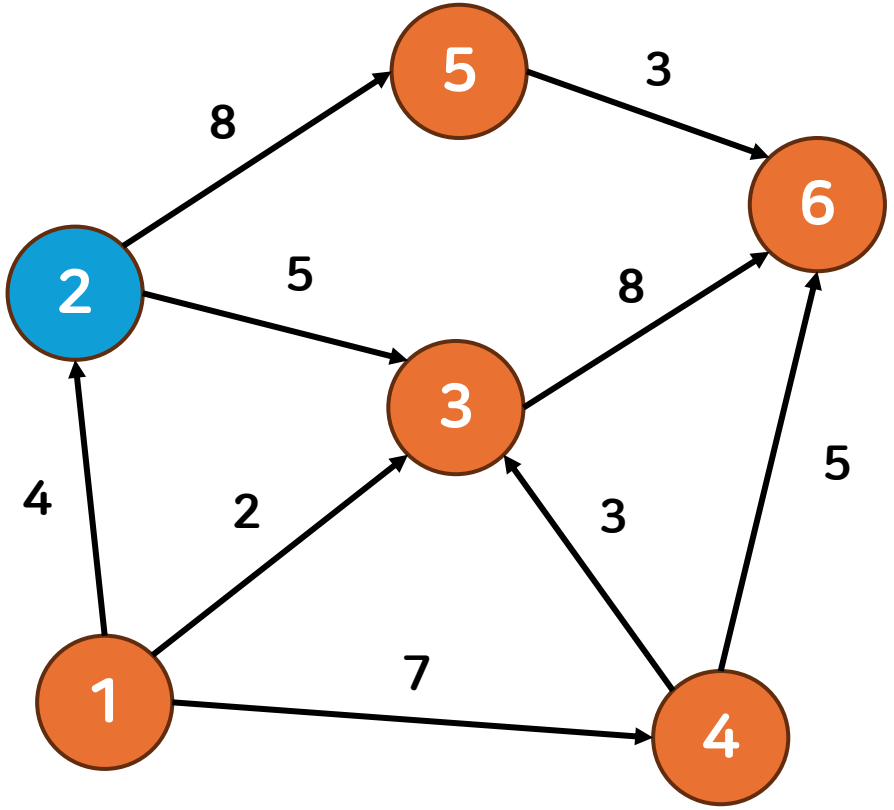
(4, 2)	(7, 4)	(10, 6)

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

다익스트라

다익스트라

1. 방문할 수 있는 정점 중 가장 최단 거리인 정점을 방문



(7, 4) (10, 6)

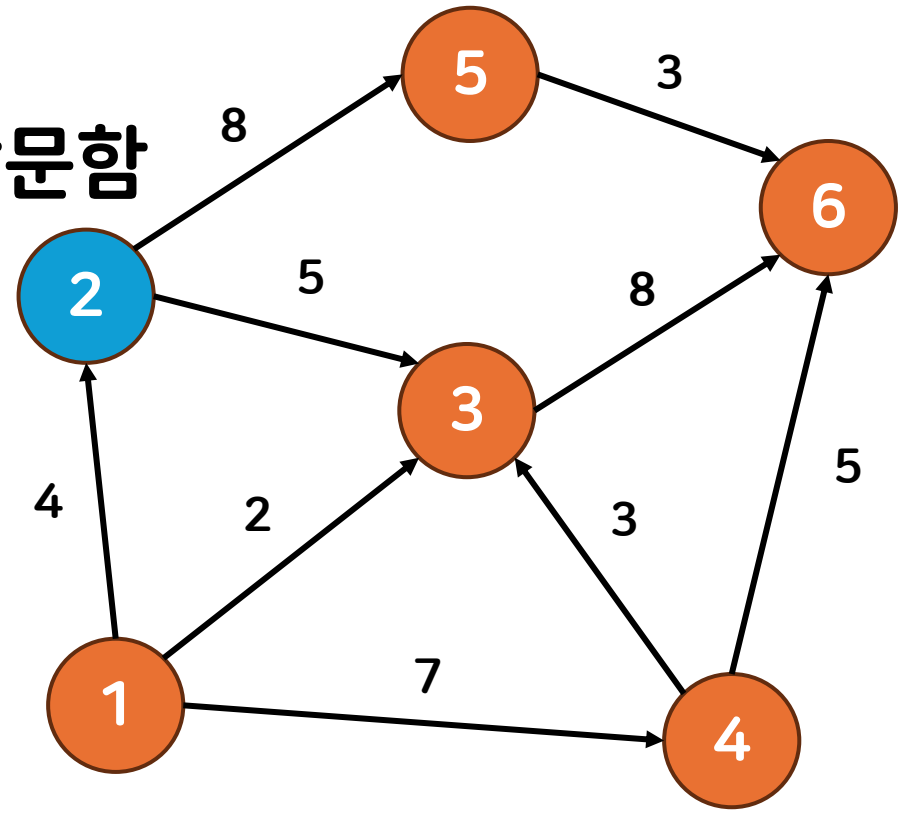
1	2	3	4	5	6
0	4	2	INF	INF	INF

다익스트라

다익스트라

2. 현재 정점과 인접한 정점들을 방문함

정점 3은 이미 최단거리 이므로
갱신 X

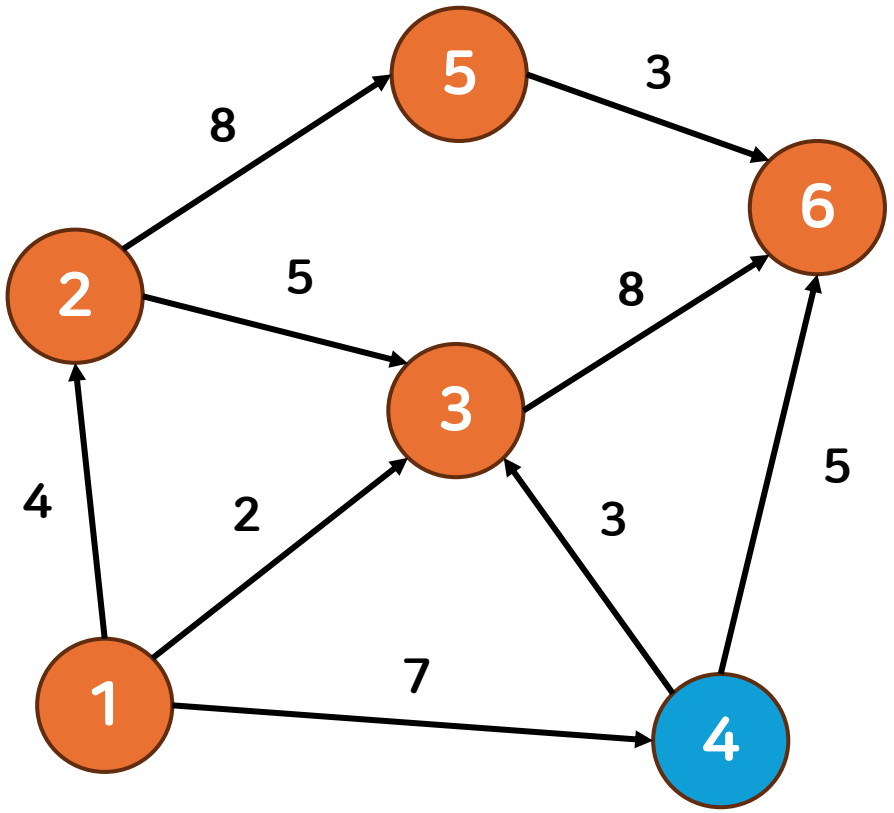


(7, 4)	(10, 6)	(12, 5)

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

다익스트라

다익스트라

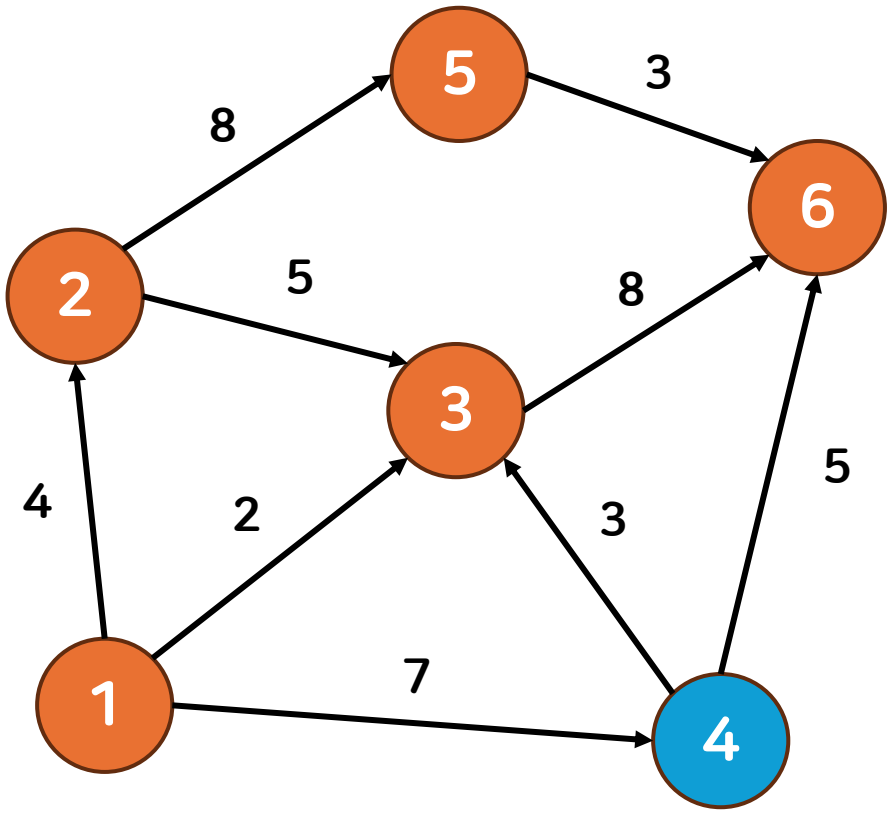


(10, 6) (12, 5)

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

다익스트라

다익스트라

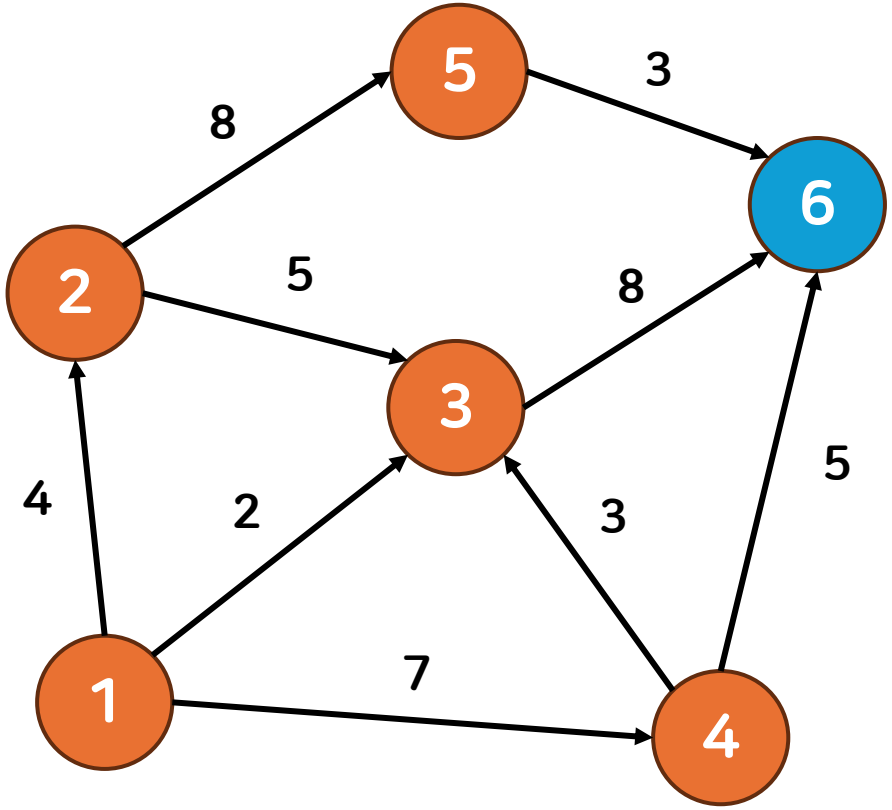


(10, 6)	(12, 5)	(12, 6)

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

다익스트라

다익스트라

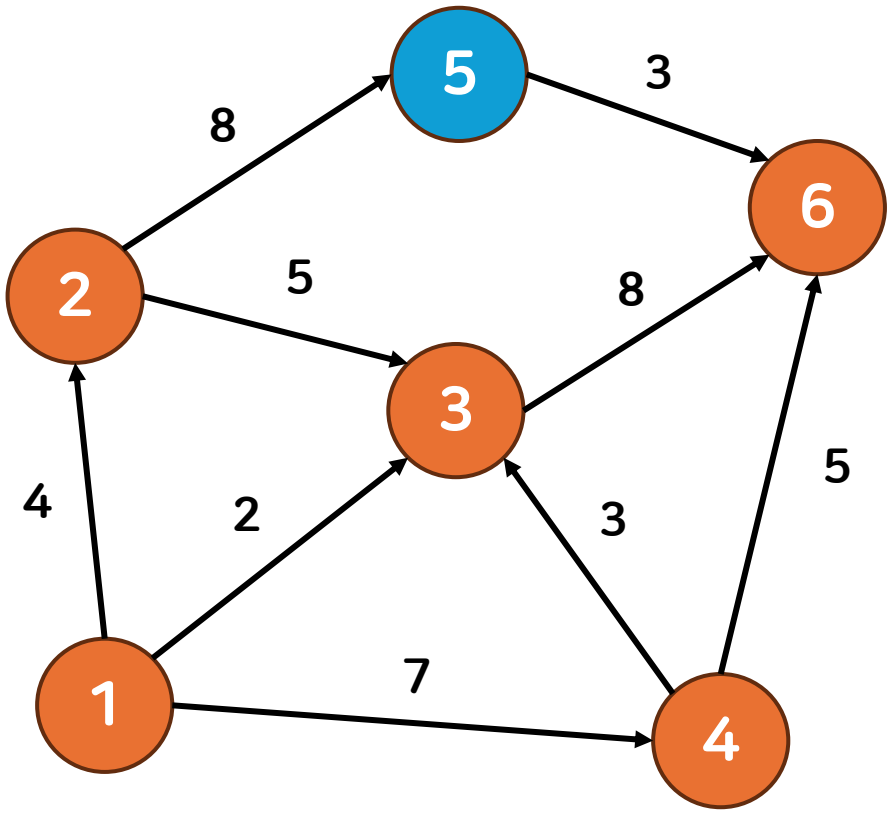


(12, 5) (12, 6)

1	2	3	4	5	6
0	4	2	7	INF	10

다익스트라

다익스트라

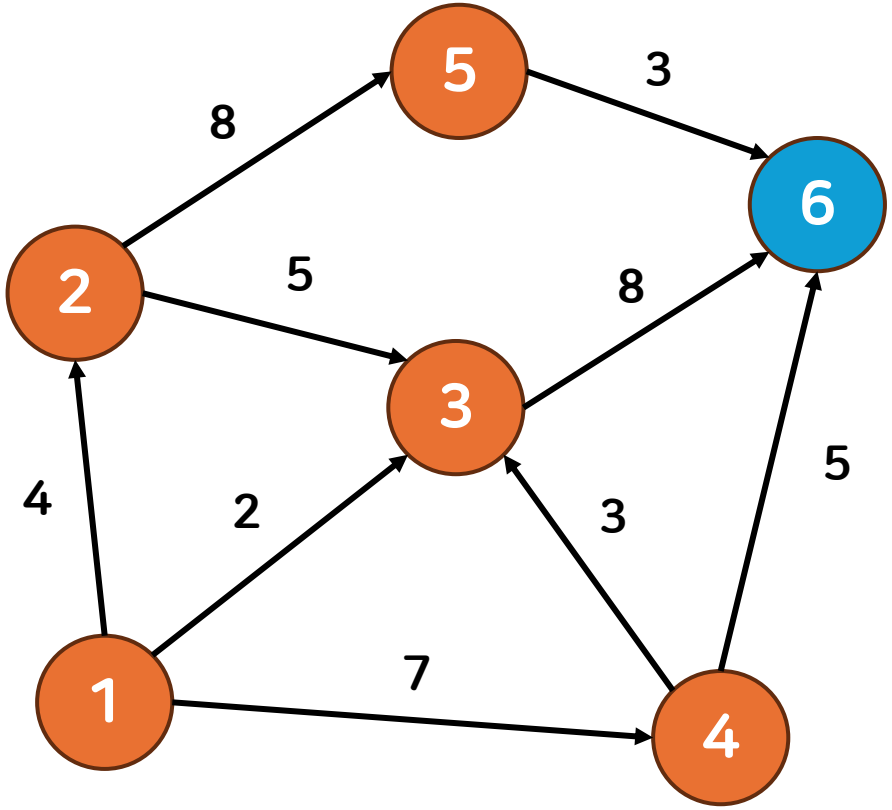


(12, 6)

1	2	3	4	5	6
0	4	2	7	12	10

다익스트라

다익스트라



1	2	3	4	5	6
0	4	2	7	12	10

질문?

최소비용 구하기 / 1916

백준 1916 / https://www.acmicpc.net/problem/1916

문제

N개의 도시가 있다. 그리고 한 도시에서 출발하여 다른 도시에 도착하는 M개의 버스가 있다. 우리는 A번째 도시에서 B번째 도시까지 가는데 드는 버스 비용을 최소화 시키려고 한다. A번째 도시에서 B번째 도시까지 가는데 드는 최소비용을 출력하여라. 도시의 번호는 1부터 N까지이다.

입력

첫째 줄에 도시의 개수 N(1 ≤ N ≤ 1,000)이 주어지고 둘째 줄에는 버스의 개수 M(1 ≤ M ≤ 100,000)이 주어진다. 그리고 셋째 줄부터 M+2줄까지 다음과 같은 버스의 정보가 주어진다. 먼저 처음에는 그 버스의 출발 도시의 번호가 주어진다. 그리고 그 다음에는 도착지의 도시 번호가 주어지고 또 그 버스 비용이 주어진다. 버스 비용은 0보다 크거나 같고, 100,000보다 작은 정수이다.

그리고 M+3째 줄에는 우리가 구하고자 하는 구간 출발점의 도시번호와 도착점의 도시번호가 주어진다. 출발점에서 도착점을 갈 수 있는 경우만 입력으로 주어진다.

출력

첫째 줄에 출발 도시에서 도착 도시까지 가는데 드는 최소 비용을 출력한다.

최소비용 구하기 / 1916

N개의 정점과 M개의 간선이 주어짐
마지막 줄에 시작점 S와 도착점 E가 주어졌을 때
S -> E의 최단 경로를 구하는 문제

예제 입력 1 복사

```
5
8
1 2 2
1 3 3
1 4 1
1 5 10
2 4 2
3 4 1
3 5 1
4 5 3
1 5
```

예제 출력 1 복사

```
4
```

최소비용 구하기 / 1916

C++

```cpp
const ll MAX = 1010;
const ll INF = 1e12;
ll n, m, d[MAX];
vector <pair<ll, ll>> adj[MAX];

using pll = pair<ll, ll>;
priority_queue <pll, vector<pll>, greater<pll>> pq;

int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n >> m;
    while(m--){
        ll s, e, c; cin >> s >> e >> c;
        adj[s].push_back({e, c});
    }
    ll s, e; cin >> s >> e;
```

```cpp
    // 거리 배열 큰 값으로 초기화
    for(int i = 0;i < MAX;i++) d[i] = INF;
    // 시작 정점을 방문 가능한 정점으로 처리
    pq.push({0, s});

    while(!pq.empty()){
        // 방문 가능한 정점 중 최단 거리인 정점 방문
        auto[cd, cur] = pq.top(); pq.pop();
        // 이미 최단거리이면 건너 뜀
        if(d[cur] <= cd) continue;
        // 거리 갱신
        d[cur] = cd;

        for(auto& [nxt, co] : adj[cur]){
            // 다음 정점이 이미 최단거리이면 건너 뜀
            if(d[nxt] <= cd + co) continue;
            // 방문 가능한 정점에 포함
            pq.push({cd + co, nxt});
        }
    }

    cout << d[e];

    return 0;
}
```

최소비용 구하기 / 1916

Python

```python
import sys
import heapq
input = sys.stdin.readline

INF = 10**12

n = int(input().rstrip())
m = int(input().rstrip())
adj = [[] for _ in range(n + 1)]
for _ in range(m):
    s, e, c = map(int, input().split())
    adj[s].append((e, c))

s, e = map(int, input().split())

# 거리 배열 큰 값으로 초기화
dist = [INF] * (n + 1)
```

```python
pq = []
# 시작 정점을 방문 가능한 정점으로 처리
heapq.heappush(pq, (0, s))

while pq:
    # 방문 가능한 정점 중 최단 거리인 정점 방문
    cd, cur = heapq.heappop(pq)
    # 이미 최단거리이면 건너 뜀
    if dist[cur] <= cd:
        continue

    # 거리 갱신
    dist[cur] = cd
    for nxt, co in adj[cur]:
        nd = cd + co
        # 현재 거리가 최단 거리이면
        if dist[nxt] > nd:
            # 방문 가능한 정점에 포함
            heapq.heappush(pq, (nd, nxt))

print(dist[e])
```

다익스트라

시간 복잡도?

간선의 개수 -> E

항상 우선순위 큐에서 제일 거리가 작은 값을 빼서 사용함
우선순위 큐에 원소가 아무리 많아도 E를 넘을 수 없음

우선순위 큐 연산 -> O(log E)

다익스트라

시간 복잡도?

간선의 개수 -> E
정점의 개수 -> V
우선순위 큐 연산 -> O(log E)

우선순위 큐에 값은 최대 E번 들어가게 됨
-> O(E log E)

다익스트라의 또 다른 구현으로 O(E log V)가 가능함
보통 E > V이기 때문에 다른 구현이 약간 빠름

다익스트라

시간 복잡도?

간선의 개수 -> E, 정점의 개수 -> V
우선순위 큐 연산 -> O(log V)

어떤 정점을 방문할 때는 항상 최단거리인 상태에서 방문하게 됨
-> 각 정점은 최대 한번 방문함
-> 각 간선도 최대 한번 방문함
우선순위 큐에 값은 최대 V번 들어가게 됨

질문?

파티 / 1238

백준 1238 / https://www.acmicpc.net/problem/1238

문제

N개의 숫자로 구분된 각각의 마을에 한 명의 학생이 살고 있다.

어느 날 이 N명의 학생이 X (1 ≤ X ≤ N)번 마을에 모여서 파티를 벌이기로 했다. 이 마을 사이에는 총 M개의 단방향 도로들이 있고 i번째 길을 지나는데 T_i(1 ≤ T_i ≤ 100)의 시간을 소비한다.

각각의 학생들은 파티에 참석하기 위해 걸어가서 다시 그들의 마을로 돌아와야 한다. 하지만 이 학생들은 워낙 게을러서 최단 시간에 오고 가기를 원한다.

이 도로들은 단방향이기 때문에 아마 그들이 오고 가는 길이 다를지도 모른다. N명의 학생들 중 오고 가는데 가장 많은 시간을 소비하는 학생은 누구일지 구하여라.

입력

첫째 줄에 N(1 ≤ N ≤ 1,000), M(1 ≤ M ≤ 10,000), X가 공백으로 구분되어 입력된다. 두 번째 줄부터 M+1번째 줄까지 i번째 도로의 시작점, 끝점, 그리고 이 도로를 지나는데 필요한 소요시간 T_i가 들어온다. 시작점과 끝점이 같은 도로는 없으며, 시작점과 한 도시 A에서 다른 도시 B로 가는 도로의 개수는 최대 1개이다.

모든 학생들은 집에서 X에 갈수 있고, X에서 집으로 돌아올 수 있는 데이터만 입력으로 주어진다.

출력

첫 번째 줄에 N명의 학생들 중 오고 가는데 가장 오래 걸리는 학생의 소요시간을 출력한다.

파티 / 1238

정점의 개수 N, 간선의 개수 M, 정점 X가 주어짐
Dist[i][X] + Dist[X][i]의 최댓값을 구하는 문제

예제 입력 1 복사

```
4 8 2
1 2 4
1 3 2
1 4 7
2 1 1
2 3 5
3 1 2
3 4 4
4 2 3
```

예제 출력 1 복사

```
10
```

파티 / 1238

Dist[i][X] + Dist[X][i]의 최댓값을 구하는 문제

다익스트라는 한 정점에 대한 최단 거리만 구할 수 있음
Dist[X][i]는 X을 기준으로 다익스트라를 구하면 됨

Dist[i][X]를 구하는 방법?

파티 / 1238

Dist[i][X]를 구하는 방법?

간단하게 생각하면 모든 정점에 대해서 다익스트라를 돌리면 됨
그러면 모든 정점 쌍의 최단거리를 구할 수 있음

시간 복잡도?

다익스트라 O(M log N)
정점이 N개이므로 O(NM log N)
= 1000 * 1000 * 10 = 1억

파티 / 1238

시간 복잡도?

다익스트라 O(M log N)

정점이 N개이므로 O(NM log N)

= 1000 * 1000 * 10 = 1억

제한 시간이 1초이므로 애매하긴 하지만 돌아가긴 함

-> 이 풀이가 정해는 아님

파티 / 1238

Dist[i][X]를 구하는 방법?

Dist[i][X]는 i -> X로 가는 최단 거리

거꾸로 생각해보자
-> 모든 간선들을 뒤집어보자

파티 / 1238

Dist[i][X]는 i -> X로 가는 최단 거리

모든 간선들의 방향을 뒤집었을 때
Dist[i][X] = X -> i로 가는 최단 거리

모든 역방향 간선들에 대해서 X를 기준으로 둔 다익스트라

Dist[i][X]와 Dist[X][i]를 구했으므로
둘의 합의 최댓값을 찾으면 됨

파티 / 1238

C++

```cpp
const ll MAX = 1010;
const ll INF = 1e12;
ll n, m, x, d[MAX][2];
vector <pair<ll, ll>> adj[MAX][2];

using pll = pair<ll, ll>;
priority_queue <pll, vector<pll>, greater<pll>> pq;

int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n >> m >> x;
    while(m--){
        ll s, e, c; cin >> s >> e >> c;
        // 정방향 간선
        adj[s][0].push_back({e, c});
        // 역방향 간선
        adj[e][1].push_back({s, c});
    }

    for(int i = 0;i < MAX;i++){
        for(int j = 0;j < 2;j++) d[i][j] = INF;
    }
```

```cpp
    for(int i = 0;i <= 1;i++){
        while(!pq.empty()) pq.pop();
        pq.push({0, x});

        while(!pq.empty()){
            auto[cd, cur] = pq.top(); pq.pop();
            if(d[cur][i] <= cd) continue;
            d[cur][i] = cd;

            for(auto& [nxt, co] : adj[cur][i]){
                if(d[nxt][i] <= cd + co) continue;
                pq.push({cd + co, nxt});
            }
        }
    }

    ll result = 0;
    for(int i = 1;i <= n;i++){
        // 역방향 간선 + 정방향 간선의 최댓값
        result = max(result, d[i][0] + d[i][1]);
    }
    cout << result;

    return 0;
}
```

파티 / 1238

Python

```python
import sys
import heapq
input = sys.stdin.readline

INF = 10**12
n, m, x = map(int, input().split())

adj = [[[] for _ in range(2)] for _ in range(n+1)]
for _ in range(m):
    s, e, c = map(int, input().split())
    # 정방향 간선
    adj[s][0].append((e, c))
    # 역방향 간선
    adj[e][1].append((s, c))
```

```python
d = [[INF, INF] for _ in range(n+1)]
for i in range(2):
    pq = []
    heapq.heappush(pq, (0, x))
    while pq:
        cd, cur = heapq.heappop(pq)
        if d[cur][i] <= cd:
            continue
        d[cur][i] = cd
        for nxt, co in adj[cur][i]:
            nd = cd + co
            if d[nxt][i] > nd:
                heapq.heappush(pq, (nd, nxt))

result = 0
for i in range(1, n+1):
    # 역방향 간선 + 정방향 간선의 최댓값
    result = max(result, d[i][0] + d[i][1])

print(result)
```

질문?

기본 과제

최소비용 구하기 – https://www.acmicpc.net/problem/1916

파티 - https://www.acmicpc.net/problem/1238

해킹 – https://www.acmicpc.net/problem/10282

소가 길을 건너간 이유 7 -
https://www.acmicpc.net/problem/14461

수열과 개구리 - https://www.acmicpc.net/problem/32294

고생하셨습니다