백준 9370번 - 미확인 도착지

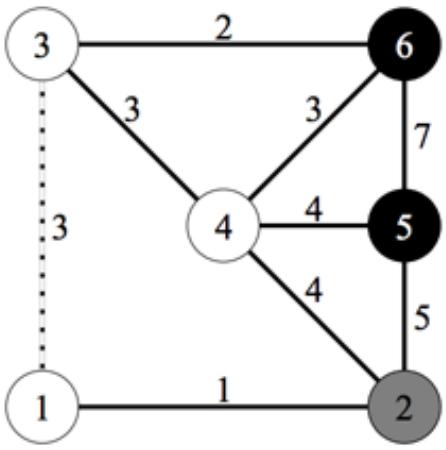
시간제한	메모리 제한	제출	정답	맞은 사람	정답 비율
3초	256MB	13439	3758	2395	25.047%

문제

(취익)B100 요원, 요란한 옷차림을 한 서커스 예술가 한 쌍이 한 도시의 거리들을 이동하고 있다. 너의 임무는 그들이 어디로 가고 있는지 알아내는 것이다. 우리가 알아낸 것은 그들이 s지점에서 출발했다는 것, 그리고 목적지 후보들 중 하나가 그들의 목적지라는 것이다. 그들이 급한 상황이기 때문에 목적지까지 우회하지 않고 최단거리로 갈 것이라 확신한다. 이상이다. (취익)

어휴! (요란한 옷차림을 했을지도 모를) 듀오가 어디에도 보이지 않는다. 다행히도 당신은 후각이 개만큼 뛰어나다. 이 후각으로 그들이 g와 h 교차로 사이에 있는 도로를 지나갔다는 것을 알아냈다.

이 듀오는 대체 어디로 가고 있는 것일까?



예제 입력의 두 번째 케이스를 시각화한 것이다. 이 듀오는 회색 원에서 두 검은 원 중 하나로 가고 있고 점선으로 표시된 도로에서 냄새를 맡았다. 따라서 그들은 6으로 향하고 있다.

입력

첫 번째 줄에는 테스트 케이스의 T(1 ≤ T ≤ 100)가 주어진다. 각 테스트 케이스마다

- 첫 번째 줄에 3개의 정수 n, m, t (2 ≤ n ≤ 2 000, 1 ≤ m ≤ 50 000 and 1 ≤ t ≤ 100)가 주어진다. 각각 교차로, 도로, 목적지 후보의 개수이다.
- 두 번째 줄에 3개의 정수 s, g, h (1 ≤ s, g, h ≤ n)가 주어진다. s는 예술가들의 출발지이고, g, h는 문제 설명에 나와 있다. (g ≠ h)
- 그 다음 m개의 각 줄마다 3개의 정수 a, b, d (1 ≤ a < b ≤ n and 1 ≤ d ≤ 1 000)가 주어진다. a와 b 사이에 길이 d의 양방향 도로가 있다는 뜻이다.
- 그 다음 t개의 각 줄마다 정수 x가 주어지는데, t개의 목적지 후보들을 의미한다. 이 t개의 지점들은 서로 다른 위치이며 모두 s와 같지 않다.

교차로 사이에는 도로가 많아봐야 1개이다. m개의 줄 중에서 g와 h 사이의 도로를 나타낸 것이 존재한다. 또한 이 도로는 목적지 후보들 중 적어도 1개로 향하는 최단 경로의 일부이다.

출력

테스트 케이스마다

- 입력에서 주어진 목적지 후보들 중 불가능한 경우들을 제외한 목적지들을 공백으로 분리시킨 오름차순의 정수들로 출력한다.

예제 입력1

```
2
5 4 2
1 2 3
1 2 6
2 3 2
3 4 4
3 5 3
5
4
6 9 2
2 3 1
1 2 1
1 3 3
2 4 4
2 5 5
3 4 3
3 6 2
4 5 4
4 6 3
5 6 7
5
6
```

예제 출력1

```
4 5
6
```

출처

출처

알고리즘 분류

- 그래프 이론
- 다익스트라

접근 방법

문제를 재해석 해보자면 아래와 같다.

> t개의 후보 중 최단 경로 상에 $g \leftrightarrow h$ 경로를 포함하는 후보를 모두 출력

단, 최단 경로는 여러개가 존재할 수 있으므로 Naive하게 가능한 모든 최단 경로를 구하는 방법을 생각할 수 있겠다.

하지만, 위의 접근 방법으로는 문제를 해결할 수 없을 것이 자명하다. 가능한 모든 경로는 $n!$ 이고, 다익스트라 알고리즘을 통하여 (`d가 양수`) greedy하게 탐색하여 최단 경로를 구한다고 해도 다음과 같이 최단 경로가 여러가지가 나오는 경우는 고려하지 못한다. 다익스트라 알고리즘은 특정 노드로 가는 경로를 고려하는 것이 아닌 특정 노드로 도달하는 최단 시간을 고려하는 알고리즘이기 때문이다.

> - 최단 경로 1 $\neq$ 최단 경로 2
> - 최단 경로1의 cost $=$ 최단 경로2의 cost
>
> 최단 경로1 : $a \rightarrow b \rightarrow c \rightarrow \ldots \rightarrow \ldots \rightarrow z$
>
> 최단 경로2 : $a \rightarrow g \rightarrow h \rightarrow \ldots \rightarrow \ldots \rightarrow z$

위의 경우에서 문제의 해답은 `yes`가 된다.

여러개의 최단 경로 중 임의의 최단 경로에 $g \leftrightarrow h$ 를 포함할 경우 해당 후보는 문제의 정답 중 하나이기 때문이다.

‼️ 하지만, 다익스트라 알고리즘을 통하여 Naive하게 탐색할 경우, 탐색 순서에 따라 정답을 못 구할 수도 있다는 치명적인 결함이 존재한다.(ex, 최단 경로1로 탐색을 진행하여, `No`로 판단하는 경우)

즉, 경로 $u_1 \rightarrow u_2 \rightarrow \ldots \rightarrow u_k$ 까지의 거리와

p_1: $u_1 \rightarrow \ldots \rightarrow g \rightarrow h \rightarrow \ldots \rightarrow u_k$

p_2 : $u_1 \rightarrow \ldots \rightarrow h \rightarrow g \rightarrow \ldots \rightarrow u_k$

p_1, p_2 로 간 최단 시간이 같다면, 최단 경로 상에 $g \leftrightarrow h$ 를 포함하고 있다고 판단할 수 있다.

그렇다면 위 방법을 가장 Naive하게 구현하기 위해서는 아래와 같이 접근할 수 있다.

1. $s \leftrightarrow v$ (v는 후보) 의 최단 경로 구하기
2. $p_1 : s \rightarrow g \rightarrow h \rightarrow v$ 의 최단 경로 구하기
3. $p_2 : s \rightarrow h \rightarrow g \rightarrow v$ 의 최단 경로 구하기
4. $s \leftrightarrow v$의 거리와 p_1 또는 p_2의 최단 경로가 같다면 v는 정답

위의 방법은 3번의 다익스트라 알고리즘을 통하여 구할 수 있다.

좀 더 최적화하여 다익스트라 알고리즘 1회로 정답을 구할 수 있다.

다음 방법은 아래와 같은 **의문점**에서 시작한다.

> 특정 간선이 포함된다면, 그것을 어떻게 판별할 수 있을까?

다음과 같은 수의 특성을 생각해볼 수 있다.

> 1. 짝수는 $k(0 \leq k)$번 곱해져도 짝수이다.
> 2. 짝수에 홀수를 더하면 홀수이다.

$g \leftrightarrow h$ 의 간선을 홀수 cost로 둔다면, 최단 경로가 1개의 $g \leftrightarrow h$ 간선을 포함한다면 최단 시간은 홀수가 된다.

즉, 모든 간선의 대소관계를 유지하면서 $g \leftrightarrow h$ 간선을 제외한 모든 간선을 짝수로 변환하는 방법은 단순하게 2를 곱하는 것으로 해결할 수 있다.

$g \leftrightarrow h$ 의 간선은 홀수로 변환하는 방법은 대표적으로 2가지가 있겠다.

```
1. $2d + 1$
1. $2d - 1$
```

결론부터 말하자면 1번 방식은 불가능하다.

> $p_1 : \ldots \rightarrow g \rightarrow h \rightarrow \ldots\ v$
>
> $p_2 : \ldots \rightarrow g \rightarrow \alpha \rightarrow \ldots\ v$
>
> 로 가는 최단경로 p_1, p_2가 존재한다고 가정. $(h \neq \alpha)$

이 경우 $2d + 1$로 $g \rightarrow h$의 cost를 설정해준다면 $g \rightarrow \alpha$ 의 cost보다 더 높아질 것이고, 다익스트라 알고리즘의 내부동작에 의하여 p_2로 탐색을 진행할 것이다.

즉, $g \rightarrow h$의 cost와 동일한 간선이 여러 개 존재할 경우 $g \rightarrow h$의 간선의 우선순위가 더 높아야한다는 뜻이된다.

이렇게 설정하더라도 d의 범위가 $1 \leq d \leq 1,000$ 이기 때문에 서로 다른 cost의 간선과 대소관계를 유지할 수 있다.

위 내용들을 구현한 소스코드는 아래와 같다.

소스코드

```
#define FASTIO cin.tie(0)->sync_with_stdio(false), cout.tie(0)
////////////////////////////////////////////////////////////////
#include <bits/stdc++.h>
using namespace std;
```

```cpp
int main(void){
    FASTIO;
////////////////////////////////////////////////////////////////
    int T;
    cin >> T;
    while(T--){
        int n, m, t;
        cin >> n >> m >> t;

        int s, g, h;
        cin >> s >> g >> h;

        vector<pair<int,int>> adj[2005];
        for(int i=0; i<m; i++){
            int a, b, d;
            cin >> a >> b >> d;
            if((a == g && b == h) || (a == h && b == g)){
                // g <-> h 사이의 간선만 홀수 cost
                d = 2 * d - 1;
            }else{
                d *= 2;
            }
            adj[a].push_back({d, b});
            adj[b].push_back({d, a});
        }

        vector<int> cand(t);
        for(auto &item : cand)
            cin >> item;

        // dijkstra
        typedef tuple<long long, int> tp;
        priority_queue<tp, vector<tp>, greater<tp>> PQ;
        PQ.push({0, s});

        vector<int> dist(n+1, 1e9);
        while(!PQ.empty()){
            auto [acCost, cur] = PQ.top(); PQ.pop();
            for(auto [cost, next] : adj[cur]){
                if(acCost + cost >= dist[next])
                    continue;
                dist[next] = acCost + cost;
                PQ.push({dist[next], next});
            }
        }
        sort(cand.begin(), cand.end());
        for(auto c : cand){
            if(dist[c] & 1)
                cout << c << ' ';
```

```
54          }
55          cout << '\n';
56      }
57      return 0;
58  }
```