백준 1800번 - 인터넷 설치

시간제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2초	128MB	1758	577	425	32.295%

문제

오늘 팀전을 다들 열심히 하시는 것을 보고 원장선생님은 세미나 실에 인터넷을 설치해 주기로 마음을 먹으셨다. 하지만 비 협조적인 코레스코 콘도는 원장님께서 학생들에게 인터넷 선을 연결하는 것에 대해서 일부에 대해 돈을 요구하였다.

각각의 학생들의 번호가 1부터 N까지 붙여져 있다고 하면 아무나 서로 인터넷 선이 연결되어 있지 않다. P(P<=10,000)개의 쌍만이 서로 이어 질수 있으며 서로 선을 연결하는데 가격이 다르다.

1번은 다행히 인터넷 서버와 바로 연결되어 있어 인터넷이 가능하다. 우리의 목표는 N번 컴퓨터가 인터넷에 연결하는 것이다. 나머지 컴퓨터는 연결 되어 있거나 연결 안되어 있어도 무방하다.

하지만 코레스코에서는 K개의 인터넷 선에 대해서는 공짜로 연결해주기로 하였다. 그리고 나머지 인터넷 선에 대해서는 남은 것 중 제일 가격이 비싼 것에 대해서만 가격을 받기로 하였다. 이때 원장선생님이 내게 되는 최소의 값을 구하여라.

입력

첫 번째 줄에 N(1 ≤ N ≤ 1,000), 케이블선의 개수 P(1 ≤ P ≤ 10,000), 공짜로 제공하는 케이블선의 개수 K(0 ≤ K < N)이 주어진다. 다음 P개의 줄에는 케이블이 연결하는 두 컴퓨터 번호와 그 가격이 차례로 들어온다. 가격은 1 이상 1,000,000 이하다.

출력

첫째 줄에 원장선생님이 내게 되는 최소의 돈을 출력한다. 만약 1번과 N번 컴퓨터를 잇는 것이 불가능 하다면 -1을 출력한다.

예제 입력1

```
5 7 1
1 2 5
3 1 4
2 4 8
3 2 3
5 2 9
3 4 7
4 5 6
```

예제 출력1

```
4
```

출처

출처

알고리즘 분류

- 그래프 이론
- 이분 탐색
- 다익스트라

접근 방법

가장 Naive하게 접근해본다면 1 $\rightarrow$ N 으로 가는 모든 경로들을 확인하여 그 중, K개를 제거한 것 중 최댓값의 최솟값을 찾는 방법을 생각할 수 있겠다. 해당 방법은 $O(N!)$의 시간복잡도를 가지므로 해당 방법으로 문제를 해결할 수 없다.

관찰을 통하여 우리는 다음과 같은 사실을 알 수 있다.

- 1 $\rightarrow$ ... $\rightarrow$ N 으로 가는 여러개의 경로가 있다고 가정해보자.

`K개를 제거한 것 중 최댓값`의 최솟값이 항상 최단 경로로 N에 도착할 때에 존재하는가?

쉽게 아니라는 것을 알 수 있다.

d_k = {1, 2, 3, ..., N}

d_m = {1, 2, 4, ..., N}

라는 [1 $\rightarrow$ N] 으로 가는 임의의 경로 2개에 대해서 생각해보자. ($|d_k| < |d_m|$)

그리고 d_k 와 d_m 을 구성하는 간선들의 가중치가 아래와 같다고 생각하였을 때 쉽게 반례를 찾아낼 수 있다.

e_k = {5, 4, 3, 2, 1}

e_m = {100, 1, 1, 1, 1}

즉, 최단 경로에서 K개를 제거한 것은 최적해가 항상 될 수 없다는 결론이다.

1에서 N까지의 모든 경로에 대한 탐색이 필수 불가결해 보인다.

관점을 바꾸어 결정론적인 방법으로 생각해보자.

1에서 N까지의 임의의 경로 e_k 에 대해서 두 정점의 거리를 $c_1, c_2, \ldots$ 로 가정하자.

(일반성을 잃지 않고 $c_1 \leq c_2 \leq \ldots$ 로 가정)

우리가 K개를 제거한 후 `최댓값` 은 c_{k+1}가 되는 것을 확인할 수 있다.

c_i 는 $1 \leq c_i \leq 10^6$ 이므로 우리가 찾고자하는 정답은 이 범위 내에 존재한다는 것이 보장된다.

임의의 c_i 에 대해서 고려해보았을 때, c_i 보다 큰 가중치를 가진 간선의 개수를 Counting 했을 때

이것이 K개라면, 우리가 찾는 최적해인 것이다.

K+1 개 이상이라면 우리가 찾고자하는 최적해보다 c_i 는 더 큰 값이므로 c_i 의 값을 줄여나가며 탐색을 진행한다.

K개 미만이라면 우리가 찾고자하는 최적해보다 c_i는 더 작은 값이므로 c_i 을 증가시키며 탐색을 진행한다.

`F(m) := 1 ~ N까지의 경로 중 가중치 m+1 이상인 간선의 개수가 K개 이하 인가?` 로 함수를 정의했을 때, 이 함수는 최적해를 기준으로 단조함수의 형태를 띤다. 최적해는 항상 정수범위에 존재하고 최적해가 존재하는 구간의 범위는 탐색을 거치며 $O(logN)$ 의 시간으로 줄어든다. 즉, 최적해의 수렴성이 보장된다.

총 시간복잡도는 $O(PlogNlog(10^6))$ 이다.

소스코드

```
#define FASTIO cin.tie(0)->sync_with_stdio(false), cout.tie(0)
////////////////////////////////////////////////////////////////
#include <bits/stdc++.h>
using namespace std;
int N, P, K;
vector<pair<int, int>> adj[1005];
typedef long long ll;
ll ans = 1e9;
bool F(ll m){
```

```
    // cost 가 > m 인 간선들의 가중치를 1로 두자
    int dist[1005];
    fill(dist, dist+1005, 1e9);
    dist[1] = 0;

    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> PQ;
    PQ.push({0, 1});
    while(!PQ.empty()) {
        auto [acDist, current] = PQ.top(); PQ.pop();
        for(auto [cost, next] : adj[current]) {
            int realCost = (cost > m);
            if(dist[next] <= dist[current] + realCost)
                continue;
            dist[next] = dist[current] + realCost;
            PQ.push({dist[next], next});
        }
    }
    return dist[N] <= K;
}
int main(void){
    FASTIO;
////////////////////////////////////////////////////////////////
    cin >> N >> P >> K;
    // K개 지원, 남은 (E - K) 중 max만 지불
    for (int i = 0; i < P; i++) {
        int a, b, c;
        cin >> a >> b >> c;
        adj[a].push_back({c, b});
        adj[b].push_back({c, a});
    }

    // 1 -> u -> v -> .. -> N
    // dist[u][cost_i] 로 도착하는 모든 경우의 수에 대해서 고려가 필요
    // memoization 불가능 (10^6 * 10^3)

    // F(k) := 1 -> N 까지 가능한 모든 경로 상에 존재하는 edge중 K+1번째 cost의 edge가 k이하인가?
    // Existence 를 보이자

    ll l = 0, r = 1000000;
    while(l <= r){
        ll m = (l + r) / 2;
        if(F(m)){
            ans = m;
            r = m - 1;
        }else
            l = m + 1;
    }
```

```
57
58      if(ans == 1e9)
59          cout << -1;
60      else
61          cout << ans;
62      return 0;
63  }
```