E4-D题题解

> 作者：余绍函

题目描述

Gahow 正赶着去参加考试，但是他忘了打印准考证了！在 Gahow 前往考场的路上有 k 家打印店，他必须要在前往考场的途中选择**至少一家**打印店去打印准考证。请你帮他算算至少需要多少时间才能到达考场。

现给出 n 个点，m 条路径，第 i 条路径 (u_i, v_i, t_i) 表示在点 u_i 和点 v_i 之间有一条路径，走过这条路径所需的时间为 t_i。Gahow 从点 1 出发，想要前往考场 n，中途至少经过一家打印店 p_i，请你求出所需的最少时间。

形式化地说，给出 n 个点 m 条边的无向图，请你求出从点 1 到点 n 且经过点集 $p_1, p_2, \cdots, p_k$ 中**至少一个**点所需的最少时间，若无法到达，请输出 -1。

输入

第一行一个正整数 $T(1 \leq T \leq 5)$ 表示数据组数。

接下来 T组数据：

第一行三个正数数 $n, m, k(1 \leq n \leq 10^5, 1 \leq m \leq 5 \times 10^5, 1 \leq k \leq n)$，分别表示点的个数 n，边的个数 m 和打印店的数量 k。

第二行 k 个正整数 $p_i(1 \leq p_i \leq n)$，表示打印店所在的点的编号。

接下来 m 行，每行三个正整数 $u_i, v_i, t_i(1 \leq u_i, v_i \leq n, 1 \leq t_i \leq 109)$表示点 u_i 与 v_i 之间有一条路径，走过这条路径所需时间为 t_i。

输出

对于每组数据，输出所需的最小花费。

输入样例

```
2
5 7 2
2 4
```

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

输出样例

```
5
-1
```

提示

对于第一组测试数据，最小的花费时间为走过 $1 \to 5 \to 4 \to 5$，共需 $1+2+2=5$ 个单位时间。

解题思路

为了简化本题思路，我们可以先考虑从点 1 到点 n 过某一点 k 的最短路。如果从点 1 到点 n 的最短路经过点 k，那么从点 1 到点 n 的最短路就是所求最短路。如果不经过点 k，那么所求最短路即为从点 1 到点 k 的最短路和从点 k 到点 n 的最短路的长度之和。

现在在考虑过点集的最短路，很容易想到：我们只需要对点集中的每一个点求过该点的最短路，然后求最小值就好了。那么我们就需要求点 1 到点集中的点和最短路，和点 n 到点集中的点的最短路。

根据这个需求，我们不难想到 $Dijkstra$ 算法，可以直接求得单元最短路径。但是本题的数据范围较大，时间限制较严，如果使用朴素的$Dijkstra$ 算法很容易被卡时间。这种情况下，只需要用优先队列优化一下就可以完美解决这个问题。

```cpp
#include<bits/stdc++.h>
#define ll long long
#define INF LLONG_MAX
using namespace std;
```

```
int t, n, m, k, u, v, p;
ll w, cost;
vector<int> printers;
vector<ll> weights1, weightsN;
vector<vector<pair<int, ll>>> graph;

struct CompareSecond {
    bool operator()(const pair<int, ll>& left, const pair<int, ll>& right)
const {
        return left.second > right.second;
    }
};

void dijkstra(int st, vector<ll>& dis) {
    priority_queue<pair<int, ll>, vector<pair<int, ll>>, CompareSecond> pq;
    vector<bool> visited(n + 1, false);
    dis[st] = 0;
    pq.emplace(st, 0);
    while(!pq.empty()) {
        u = pq.top().first;
        pq.pop();
        visited[u] = true;
        for(pair<int, ll> p0: graph[u]) {
            if(dis[p0.first] > dis[u] + p0.second && !visited[p0.first]) {
                dis[p0.first] = dis[u] + p0.second;
                pq.emplace(p0.first, dis[p0.first]);
            }
        }
    }
}

int main() {
    scanf("%d", &t);
    while(t--) {
        scanf("%d%d%d", &n, &m, &k);
        printers.clear();
        weights1.assign(n + 1, INF);
        weightsN.assign(n + 1, INF);
        graph.assign(n + 1, vector<pair<int, ll>>());
        for(int _ = 0; _ < k; _++) {
            scanf("%d", &p);
            printers.emplace_back(p);
```

```
        } for(int _ = 0; _ < m; _++) {
            scanf("%d%d%lld", &u, &v, &w);
            graph[u].emplace_back(make_pair(v, w));
            graph[v].emplace_back(make_pair(u, w));
        } dijkstra(1, weights1);
        dijkstra(n, weightsN);
        cost = INF;
        for(int i = 0; i < k; i++) {
            if(weights1[printers[i]] != INF && weightsN[printers[i]] != INF)
                cost = min(cost, weights1[printers[i]] +
weightsN[printers[i]]);
        } printf("%lld\n", cost == INF ? -1 : cost);
    } return 0;
}
```