最小生成树及其应用

Minimum Spanning Tree (MST)

缘起：CSP-S

背景引入

- 冬令营的面向对象，以 CSP-S 为界， T2 是分水岭，故借此题，用生成树相关的内容开题。
- **生成树问题**：
  - 许多图论问题的核心在于从复杂的网络中提取“骨架”，生成树本身的性质以及在图论的模型转换中的位置都至关重要。

CSP-S T2

感受真题

- 2025 CSP-S T2 《道路修复》给许多同学留下了深刻印象。
- **问题简述**：
  地震摧毁了所有道路，需要修复道路，并连接所有 N 个城市，同时还有 K 个乡镇，可以进行城市化改造并接入网络。如何在保证所有城市互通的前提下，让总预算（修复费用+建设费用）最低？
- 数据范围：$N \leq 10^4, M \leq 10^6, K \leq 10$。

从 CSP-S T2 出发

模型抽取

- 这道题表面看是复杂的“选择+连接”问题，但其核心直指图论中的经典模型——**最小生成树（Minimum Spanning Tree, MST）**。
- 要解决这个级别的难题，单纯背诵模板是不够的，需要建立从“基础算法”到“性质分析”再到“综合优化”的完整知识体系。

引入思考

- **基础问题**：
  - 给定一张无向图，如何以最小的代价（边权和最小）连接所有节点？
  - 这种结构在算法竞赛和现实生活中有何特殊地位？

一、 最小生成树的基本概念

1.1 现实意义

- **设想：要铺设一张城市级的光纤网络**
  1. **连通**：不能让任何一个小区掉线。
  2. **树**：没有多余的回路（环），因为多一条边就是浪费成本。
  3. **最小**：用料总成本最低。

1.2 相关性质与定义

1、树的属性

- 树是一个连通的，无环的无向图，称一个不可能连通的无向图为森林。
- 如果一个图是树，则其边数等于点数减一，两顶点之间路径唯一，添边成环。

2、 生成树的定义

- 对于无向连通图 G 的一个子图，如果它是包含 G 所有顶点的树，它就是 G 的一个生成树（即在图的边集中选择 $n-1$ 条边，将所有顶点连通）。

3、 最小生成树

- 各边权重总和最小的生成树称为最小权重生成树，简称最小生成树。

1.3 核心性质（贪心依据）

MST 算法主要依赖性质：

1、最小边原则

对于一个图 G，权值最小的边（如果唯一的话）一定是在最小生成树上。

2、唯一性定理

对于一个图 G，如果图中的边权值都不相同，则图的最小生成树是唯一的，反之不然。

3、环定理

对于一个图 G，如果加入一条边会形成环，那么这条边的权值一定大于等于环中其他边的权值。换句话说，最小生成树中不会出现权值大于等于环中其他边的权值的边。

例题 01：最小生成树（模板）题面

题目描述

给定结点数为 n ，边数为 m 的带权无向连通图 G ，所有结点编号为 $1, 2. . . . . n$ 。
求 G 的最小生成树的边权和。

输入格式

第一行两个正整数 n, m 。
之后的 m 行，每行三个正整数 $u_i, v_i, w_i (1 \leq u_i, v_i \leq n)$ ，描述一条连接结点 u_i 和 v_i，边权为 w_i的边。

输出格式

一个整数表示 G 的最小生成树的边权和。

例题 01：最小生成树（模板）数据范围

数据范围

对于 30% 的数据，$1 \le n$，$m \le 1000$，$0 \le w_i \le 10^9$。

对于 100% 的数据，$1 \le n \le 2 \times 10^5$，$0 \le m \le 5 \times 10^5$,$0 \le w_i \le 10^9$。

例题 01：最小生成树（模板）思路分析

- **小数据**：
  - 当 N, M 数据较小（如 N，$M \leq 1000$）时，暴力跑搜索即可。
- **大数据**：
  - **挑战**：当 $N \leq 10^5, M \leq 2 \times 10^5$ 时，如何高效实现？
  - **目标**：需要 $O(M \log M)$ 或 $O(M \log N)$ 级别的算法。

二、 探讨两个核心算法

- 1、Prim 算法
- 2、 Kruskal 算法

2.1 Prim 算法

(1) 基本思想

- **加点法**：
  - 从任意一个节点出发，逐步“吞并”最近的节点。
  - 像墨水在纸上扩散一样。
- **集合维护**：
  - 维护一个集合 S（已选点）和 $V-S$（未选点）。
  - 每次选择 **连接** S 与 $V-S$ 的最短边。

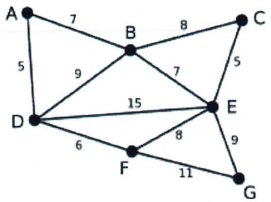
(2) Prim 算法的实施步骤

1. **初始化**：
   - 选定起点（设 **1** 号点）。
   - 距离数组 $dist$ 初始化为 ∞，$dist[1] = 0$。
2. **循环 N 次**：
   - 找到未访问点中 $dist$ 最小的点 x。
   - 标记 x 为已访问（加入集合 S）。
   - 扫描 x 的所有出边 (x, y, w)，更新 $dist[y] = min(dist[y], w)$。

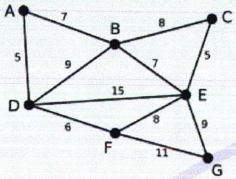
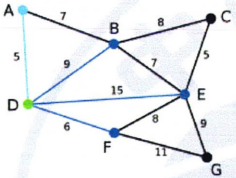
(3) Prim 算法执行实例

初始状态

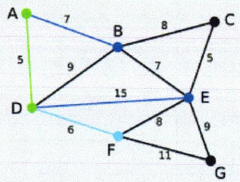
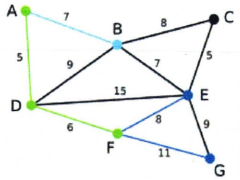
- **说明**：原始加权连通图。
- **状态**：所有点未选。



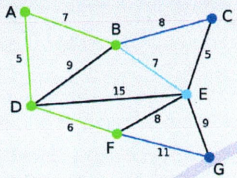
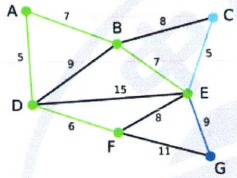
(3) Prim 算法执行实例 (1/4)

图例	说明	不可选	可选	已选 (Vnew)
	此为原始的加权连通图。每条边一侧的数字代表其权值。	-	-	-
	顶点 D 被任意选为起始点。顶点 A、B、E 和 F 通过单条边与 D 相连。A 是距离 D 最近的顶点，因此将 A 及对应边 AD 以高亮表示。	C, G	A, B, E, F	D

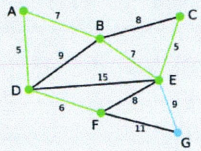
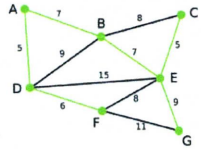
(3) Prim 算法执行实例 (2/4)

图例	说明	不可选	可选	已选 (Vnew)
	下一个顶点为距离 D 或 A 最近的顶点。B 距 D 为 9，距 A 为 7，E 为 15，F 为 6。因此，F 距 D 或 A 最近，因此将顶点 F 与相应边 DF 以高亮表示。	C, G	B, E, F	A, D
	算法继续重复上面的步骤。距离 A 为 7 的顶点 B 被高亮表示。	C	B, E, G	A, D, F

(3) Prim 算法执行实例 (3/4)

图例	说明	不可选	可选	已选 (Vnew)
	在当前情况下，可以在 C、E 与 G 间进行选择。C 距 B 为 8，E 距 B 为 7，G 距 F 为 11。点 E 最近，因此将顶点 E 与相应边 BE 高亮表示。	无	C, E, G	A, D, F,
	这里，可供选择的顶点只有 C 和 G。C 距 E 为 5，G 距 E 为9，故选取 C，并与边 EC一同高亮表示。	无	C, G	A, D, F, B, E

(3) Prim 算法执行实例 (4/4)

图例	说明	不可选	可选	已选 (Vnew)
	顶点 G 是唯一剩下的顶点，它距 F 为 11，距 E 为 9，E 最近，故高亮表示 G 及相应边 EG。	无	G	A, D, F, B, E, C
	所有顶点均已被选取，图中绿色部分即为连通图的最小生成树。在此例中，最小生成树的权值之和为 39。	无	无	A, D, F, B, E, C, G

```
    for(int i = 0; i < n; i++) {
        int t = -1;    // 寻找距离集合最近的点 t
        for(int j = 1; j <= n; j++)
            if(!st[j] && (t == -1 || dist[j] < dist[t])) t = j;

        // 如果图不连通 return INF;
        if(i && dist[t] == 0x3f3f3f3f) return INF;

        st[t] = true;
        // 用 t 更新其他点
        for(int j = 1; j <= n; j++)
            dist[j] = min(dist[j], g[t][j]);
    }
}
```


(5) Prim 算法的优化

- **瓶颈：**

- **瓶颈**：
  - 朴素版寻找最小 dist 的点需要 $O(N)$，总复杂度 $O(N^2)$。
- **优化**：
  - 使用 **优先队列 (Priority Queue)** 维护 dist。
  - 类似 Dijkstra 的堆优化。
- **复杂度**：
  - $O(M \log N)$。
- **适用性**：
  - 稠密图表现优异。
  - 稀疏图通常不如 Kruskal 易写。

(6) Prim 算法代码框架（优化版本）

```
int prim() {
    memset(dist, 0x3f, sizeof dist);
    dist[1] = 0;
    priority_queue<PII, vector<PII>, greater<PII>> q;
    q.push({0, 1}); // {距离, 点编号}

    while(q.size()) {
        auto t = q.top(); q.pop();
        int ver = t.second, distance = t.first;

        if(st[ver]) continue;
        st[ver] = true;
        res += distance;

        for(int i = h[ver]; i != -1; i = ne[i]) {
            int j = e[i];
            if(dist[j] > w[i]) {
                dist[j] = w[i];
                q.push({dist[j], j});
            }
        }
    }
    return res; // 需注意判断连通性
}
```

2.2 Kruskal 算法

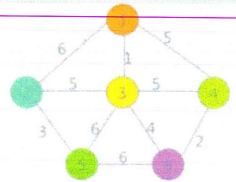
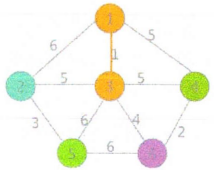
(1) 基本思想

- **加边法**：
  - 关注边而非点。
  - 贪心地选取权值最小的边。
- **判断依据**：
  - 如果这条边连接的两个点当前**不连通**，则加入这条边。
  - 如果已连通，则加入会形成环，必须忽略。

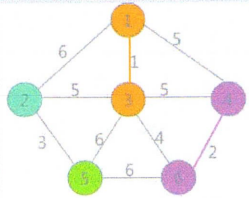
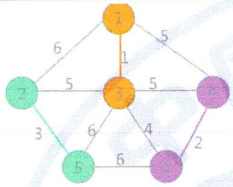
(2) Kruskal 算法的实施步骤

1. **排序**：
   - 将所有边按权值从小到大排序。
2. **初始化**：
   - 初始化并查集（DSU），每个点自成一个集合。
3. **遍历**：
   - 依次遍历每条边 (x, y, w)。
   - 查询 $find(u)$ 是否等于 $find(v)$。
   - 若不等 $\rightarrow union(x, y)$，累加权值。
   - 若相等 $\rightarrow$ 跳过。

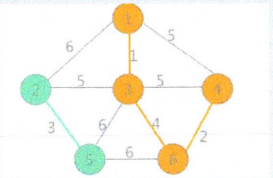
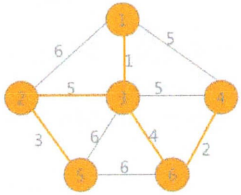
(3) Kruskal 算法的执行实例 (1/4)

图例	说明
	在初始状态下，对各顶点赋予不同的标记（用颜色区别）
	对所有边按照权值的大小进行排序，按照从小到大的顺序进行判断，首先是（1，3），由于顶点 1 和顶点 3 标记不同，所以可以构成生成树的一部分，遍历所有顶点，将与顶点 3 标记相同的全部更改为顶点 1 的标记

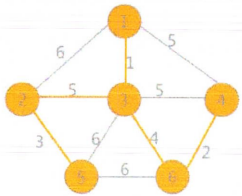
(3) Kruskal 算法的执行实例 (2/4)

图例	说明
	其次是（4，6）边，两顶点标记不同，所以可以构成生成树的一部分，更新所有顶点的标记
	其次是（2，5）边，两顶点标记不同，可以构成生成树的一部分，更新所有顶点的标记

(3) Kruskal 算法的执行实例 (3/4)

图例	说明
	然后最小的是（3，6）边，两者标记不同，可以连接，遍历所有顶点，将与顶点 6 标记相同的所有顶点的标记更改为顶点 1 的标记
	继续选择权值最小的边，此时会发现，权值为 5 的边有 3 个，其中（1，4）和（3，4）各自两顶点的标记一样，如果连接会产生回路，所以舍去，而（2，3）标记不一样，可以选择，将所有与顶点 2 标记相同的顶点的标记全部改为同顶点 3 相同的标记

(3) Kruskal 算法的执行实例 (4/4)

图例	说明
	当选取的边的数量相比与顶点的数量小 1 时，说明最小生成树已经生成。

(4) Kruskal 算法代码框架

```
struct Edge { int u, v, w; } edges[M];
bool cmp(Edge a, Edge b) { return a.w < b.w; }

int kruskal() {
    sort(edges, edges + m, cmp);
    for (int i = 0; i < m; i++) {
        int a = find(edges[i].u), b = find(edges[i].v);
        if (a != b) {
            p[a] = b; // 并查集合并
            res += edges[i].w;
            cnt++;
        }
    }
    return cnt < n - 1 ? -1 : res;
}
```

(5) 复杂度分析

- **时间复杂度**：
  - **排序**：$O(M \log M)$。这是 Kruskal 的主要瓶颈。
  - **并查集**：$O(M\alpha(N))$。其中 $\alpha(N)$ 是反阿克曼函数，增长极慢（< 5），通常视为常数。
  - **总计**：$O(M \log M)$。
- **空间复杂度**：
  - $O(M)$ 存边，$O(N)$ 存并查集。

三、 建图基础和模型转化的思想

3.1 基础建图

将各种关系转化为图的模型

- **策略**：
  - 识别实体为“节点”。
  - 识别关系代价为“边权”。

例题 02：通讯网络 题面（1/2）

题目描述

Alien 们由于通信的需要，需要在 N 个城市之间建立通讯网络。 为了方便问题的描述，将每个城市抽象为一个平面上的点 (X_i, Y_i) 。旧社会留下了一些仍可以使用的双向通讯线路，共 M 条。
每一条线路的费用是这两个城市之间的欧几里德距离。
现在 Alien 们委托你来做一个方案，在这些城市之间建立起双向的通讯线路，使得任意两个城市的通讯畅通，并且费用总和最小。

例题 02：通讯网络 题面（2/2）

输入格式

第一行两个整数 N, M，意义如上。
接下来 N 行，每行两个数 X_i, Y_i，表示一个点的坐标。
接下来 M 行，每行两个数 A_i, B_i，表示已经存在的通讯线路。

输出格式

一行一个数 Ret，表示答案，精确到小数点后两位。

数据范围

对于 100% 的数据：$N, M \leq 1000$，$X_i, Y_i \leq 1000$。

例题 02：通讯网络 解法分析

解法分析

本题需要先按照题意，建立所有点边关系，边权为两点间的欧几里得距离，而已存在的通讯线路的点与点的边权即为 0。再在建好的图上，求最小生成树即可。

3.2 超级源点

- **问题场景**：
  - 既有**边权**（连接两点的代价）。
  - 又有**点权**（在某点建立发电站/打井的代价）。
- **转化技巧**：
  - 建立一个虚拟节点（超级源点 S）。
  - 将每个点 i 与 S 连边，边权 = 点权（建立代价）。
  - 原图中的边保持不变。

例题 03：新的开始 题面（1/2）

题目描述

发展采矿业当然首先得有矿井，小 FF 花了上次探险获得的千分之一的财富请人在岛上挖了 n 口矿井，但他似乎忘记考虑的矿井供电问题……

为了保证电力的供应，小 FF 想到了两种办法：

1. 在这一口矿井上建立一个发电站，费用为 v（发电站的输出功率可以供给任意多个矿井）。
2. 将这口矿井与另外的已经有电力供应的矿井之间建立电网，费用为 p。

小 FF 希望身为「NewBeOne」计划首席工程师的你帮他想出一个保证所有矿井电力供应的最小花费。

例题 03：新的开始 题面（2/2）

输入格式

第一行一个整数 n，表示矿井总数。
第 $2 \sim n+1$ 行，每行一个整数，第 i 个数 $v[i]$ 表示在第 i 口矿井上建立发电站的费用。

接下来为一个 $n*n$ 的矩阵 P，其中 $p[i,j]$ 表示在第 i 口矿井和第 j 口矿井之间建立电网的费用（数据保证有 $p[i,j]=p[j,i]$, 且 $p[i,i]=0$）。

输出格式

仅一个整数，表示让所有矿井获得充足电能的最小花费。

数据范围

对于 30% 的数据，$1 \le n \le 50$;
对于 100% 的数据，$1 \le n \le 300$，$1 \le v_i, P_{i,j} \le 10^5$。

例题 03：新的开始 解法分析

- **解法**：
  - 引入超级源点 S。
  - S 到点 i 的边权为在 i 处建立矿井的费用。
  - 点 i 到点 j 的边权为铺设管道的费用。
  - 对新图求 MST 即可。
  - 这样就把“点权”转化为了“边权”，统一处理。

3.3 最大生成树

- **场景**：
  - 某些求解要求取边权的极大值（如收益最大化）。
- **逆向操作**：
  - **方法一**：更改排序规则，边权从大到小排序（使用 Kruskal）。
  - **方法二**：将所有边权取相反数，求最小生成树，结果再取反。

例题 04：征募 题面（1/2）

题目描述

Windy 有一个国家，他想建立一支军队来保护国家。
他挑选了 N 个女孩和 M 个男孩，想把他们收集起来当他的士兵。要收缴一名没有任何特权的士兵，他必须支付 1 万元人民币。女孩和男孩之间有一些关系，Windy 可以利用这些关系来降低成本：

- 如果女孩 x 和男孩 y 的关系是 d，并且其中一个已经被收取，Windy 可以用 $10000 - d$ 人民币收取另一个。

现在考虑到女孩和男孩之间的所有关系，你的任务是找到 Windy 必须支付的最低金额。

请注意，收集一名士兵时只能使用一个关系。

例题 04：征募 题面（2/2）

输入格式

输入的第一行是测试用例的数量 t，每个测试用例的第一行包含三个整数，N、M 和 R。
然后是 R 行，每行包含三个整数 x_i、y_i 和 d_i。
注意：每个测试用例前都有一个空行。

输出格式

对于每个测试用例，在一行中输出答案。

数据范围

对于100%的数据，保证
$1 \le N, M \le 10000, 0 \le R \le 50,000, 0 \le xi < N, 0 \le yi < M, 0 < di < 10000$。

例题 04：征募 解法分析

解法分析

本题为了使费用最小，就要使关系亲密度最大。由于收集一个士兵只能使用一种关系，那么就是要求关系的最大生成树，$ans = (n + m) * 10000$ - 最大生成树（森林）

3.4 类最小生成树

- **场景**：
  - 某些题目不直接求 MST 权值和，但需要使用生成树的性质解答。

例题 05：公路网络 题面（1/2）

题目描述

喵国的公路网络由 N 座城市和 M 条公路组成，其中第 i 条公路连接城市 u_i, v_i ，其维护成本为 c_i 。最优公路网络指的是选择这 M 条公路中的若干条，使得 N 座城市能互达且公路的维护成本之和最小。

现在喵国想进行公路网络的优化，他们计划进行 Q 次考察，每次考察的内容为，若在城市 t_i, g_i 之间新建一条维护成本为 w_i 的公路，这条公路是否也在最优公路网中。

请你设计程序，对每次考察进行判断。

例题 05：公路网络 题面（2/2）

输入格式

共 $Q+M+1$ 行：
第一行三个整数 N, M, Q，依次表示城市数量，公路数量和考察次数；
接下来的 M 行每行三个整数 u_i, v_i, c_i，依次表示第 i 条公路连接的两个城市编号，以及对应的维护成本；
接下来的 Q 行每行三个整数 t_i, g_i, w_i，依次表示第 i 次考察对应的新建公路连接的两个城市编号，以及维护成本；

输出格式

Q 行，每行一个字符串，若第 i 次考察新建的公路在最小公路网中，输出 Yes，否则输出 No。
注意，每个考察都是独立的。

例题 05：公路网络 解法分析

- **分析**：
  - 可能涉及查询两点间路径上的最大边权。
  - 或者判断某条边是否在 MST 中。
- **思路**：
  - 模拟最小生成树kruskal算法的过程中，若加入的边的边权小于当前边的边权，则当前边被取代。
  - 离线储存每个询问 t, g, w, id ，在每次并查集合并前，按 w 的大小从小到大把待询问道的 w 与当前边权对比，若 w 比当前边权小，显然应用 w 。

回顾与尝试

CSP-S T2 道路修复

- 最小生成树的相关知识已经系统进行了梳理和复习
- 小试牛刀：尝试挑战 CSP-S T2 道路修复

例题00：道路修复 题面（1/4）

Time Limit: 1 sec / Memory Limit: 512 MiB

问题描述

C 国的交通系统由 n 座城市与 m 条连接两座城市的双向道路构成，第 i $(1 \le i \le m)$ 条道路连接城市 u_i 和 v_i。**任意两座城市都能通过若干条道路相互到达。**

然而，近期由于一场大地震，所有 m 条道路都被破坏了，修复第 i $(1 \le i \le m)$ 条道路的费用为 w_i。与此同时，C 国还有 k 个准备进行城市化改造的乡镇。对于第 j $(1 \le j \le k)$ 个乡镇，C 国对其进行城市化改造的费用为 c_j。在城市化改造完第 j $(1 \le j \le k)$ 个乡镇后，可以在这个乡镇与原来的 n 座城市间建造若干条道路，其中在它与第 i $(1 \le i \le n)$ 座城市间建造一条道路的费用为 $a_{j,i}$。C 国可以在这 k 个乡镇中选择**任意多个**进行城市化改造，也可以不选择任何乡镇进行城市化改造。

为尽快恢复城市间的交通，C 国政$ $府希望以最低的费用将**原有**的 n 座城市两两连通，也即任意两座原有的城市都能通过若干条修复或新建造的道路相互到达。你需要帮助他们求出，将原有的 n 座城市两两连通的最小费用。

50

例题00：道路修复 题面（2/4）

输入格式

输入的第一行包含三个非负整数 n, m, k，分别表示原有的城市数量、道路数量和准备进行城市化改造的乡镇数量。

输入的第 $i+1$ $(1 \le i \le m)$ 行包含三个非负整数 u_i, v_i, w_i，表示第 i 条道路连接的两座城市与修复该道路的费用。

输入的第 $j+m+1$ $(1 \le j \le k)$ 行包含 $n+1$ 个非负整数 $c_j, a_{j,1}, a_{j,2}, \ldots, a_{j,n}$，分别表示将第 j 个乡镇进行城市化改造的费用与在该乡镇与原有的城市间建造道路的费用。

输出格式

输出一行一个非负整数，表示将原有的 n 座城市两两连通的最小费用。

例题00：道路修复 题面（3/4）

【数据范围】

对于所有测试数据，保证：

- $1 \le n \le 10^4$，$1 \le m \le 10^6$，$0 \le k \le 10$；
- 对于所有 $1 \le i \le m$，均有 $1 \le u_i, v_i \le n$，$u_i \ne v_i$ 且 $0 \le w_i \le 10^9$；
- 对于所有 $1 \le j \le k$，均有 $0 \le c_j \le 10^9$；
- 对于所有 $1 \le j \le k$，$1 \le i \le n$，均有 $0 \le a_{j,i} \le 10^9$；
- 任意两座原有的城市都能通过若干条原有的道路相互到达。
- 特殊性质 A：对于所有 $1 \le j \le k$，均有 $c_j = 0$ 且均存在 $1 \le i \le n$ 满足 $a_{j,i} = 0$

例题00：道路修复 题面（4/4）

1767664807501

例题00：道路修复 题意分析

- 本题的核心是求解一个图的 **最小生成树 (MST)**，但引入了 K 个可选的“辅助点”（乡镇）。
- 由于 K 非常小 ($K \leq 10$)，从数据上提示了，可以在 K 上做文章，通常涉及状态压缩或子集枚举。

例题00：道路修复 解法分析一

朴素枚举 + Kruskal （期待得分：30-40分）

适用范围： N, M 较小的数据点（如 $N \leq 100, M \leq 1000$）。

思路：
最直观的想法是枚举 K 个乡镇的所有选择方案。

1. 枚举 2^K 种乡镇选取状态（使用二进制状态压缩）。
2. 对于每一种状态：
    - 构建一个新图：包含原有的 N 个城市和被选中的乡镇。
    - 边集包含：原有的 M 条道路 + 被选中乡镇与城市的连接边。
    - 对这个新图跑 Kruskal 算法求 MST。
    - 如果连通，计算总费用（MST边权和 + 选中乡镇的建设费），并更新全局最小值。

例题00：道路修复 解法分析一

复杂度分析：

- 每次 Kruskal 需要排序。最坏情况下边数为 $M + K \times N$。
- 总复杂度：$O(2^K \cdot (M + KN)\log(M + KN))$。
  当 $M = 10^6$ 时，单次排序开销巨大，乘上 $2^{10} = 1024$，总运算量超 10^{10}，必然超时。

例题00：道路修复 解法分析二

Prim 算法（稠密图优化）（期待得分：50-60分）

适用范围： N 较小 ($N \leq 1000$) 但 M 很大 ($M \approx N^2$) 的情况。

思路：
当 M 很大时，Kruskal 的排序 ($M \log M$) 会很慢。如果选定了乡镇，新图的节点数 $V' = N + K_{sel}$。
Prim 算法的时间复杂度为 $O(V'^2)$，与边数无关。
如果 $N \leq 1000$，则 $V'^2 \approx 10^6$，配合 $2^K \approx 1000$，总计算量约 10^9，勉强可过或稍微超时，优于解法一。

例题00：道路修复 解法分析三

从MST的性质分析 （期待得分：100分）

在原图的 M 条道路中，**只有那些属于原图 MST 的边（最多 $N-1$ 条）才是有可能被选用的。**

58

例题00：道路修复 解法分析三

从MST的性质分析 （期待得分：100分）

证明：
如果一条边 (u, v) 不在原图的 MST 中，说明在原图中存在一条连接 u, v 的路径，且路径上的每一条边权都小于等于 $w(u, v)$。
当引入新的乡镇节点后，这些新节点提供了连接 u, v 的**新路径**。

1. 如果新路径比原路径更优，则应该选新路径。
2. 如果新路径不如原路径，则还是选原 MST 路径。

- 无论哪种情况，那条“因为太重而被原 MST 抛弃”的边，依然没有机会翻身。它不仅比原 MST 路径重，大概率也比经过乡镇的新路径重（否则它就是连接两点的最优解了）。
- 因此，**可以先对原图 N 个点跑一遍 Kruskal，筛选出 $N-1$ 条边。**

例题00：道路修复 解法分析三

从MST的性质分析 （期待得分：100分）

算法流程：

1. **预处理**：对原图 M 条边进行 Kruskal，保留 MST 上的边，记为 E_{base}。边数从 10^6 降至 10^4。
2. **枚举**：枚举 K 个乡镇的选取状态 $mask$。
3. **合并**：将 E_{base} 与当前选中的乡镇边（$K \times N$ 条）合并。
4. **计算**：对合并后的边集跑 Kruskal。此时边数规模仅为 $O(N)$。

复杂度：

$O(M \log M)$ (预处理) + $O(2^K \cdot NK \cdot \alpha(N))$ (枚举求解，**需利用归并优化排序**)。
计算量约为 $10^7 + 1000 \times 10^5 \approx 10^8$，可以通过。
若不优化排序，复杂度为 $O(2^K \cdot NK \log(NK))$，计算量约 1.7×10^9，会**超时**。 60

回顾与思考

- 懂算法的基础原理，尝试应用，是第一步
- 理解算法的本质，才能应对更多变的题目与环境

四、 性质深挖与问题延展

从以上的内容来看，MST更多是一种模型抽取的过程算法

- 从图中抽取出来的树、其连通性与边的各种性质值得继续探讨
- 生成树的过程，也应该被重视与思考

4.1 瓶颈生成树

- **定义**：
  - 一棵生成树，其**最大边权**在所有生成树中是最小的。
- **重要性质**：
  - **最小生成树一定是瓶颈生成树。**
  - (注意：瓶颈生成树不一定是最小生成树，可能有多种选择，但 MST 的最大边权一定是所有瓶颈树中最小的)。

例题 06：繁忙都市 题面（1/2）

题目描述

城市 C 是一个非常繁忙的大都市，城市中的道路十分的拥挤，于是市长决定对其中的道路进行改造。城市 C 的道路是这样分布的：城市中有 n 个交叉路口，有些交叉路口之间有道路相连，两个交叉路口之间最多有一条道路相连接。这些道路是双向的，且把所有的交叉路口直接或间接的连接起来了。每条道路都有一个分值，分值越小表示这个道路越繁忙，越需要进行改造。但是市政府的资金有限，市长希望进行改造的道路越少越好，于是他提出下面的要求：

1、改造的那些道路能够把所有的交叉路口直接或间接的连通起来。
2、在满足要求 (1) 的情况下，改造的道路尽量少。
3、在满足要求 (1)、(2) 的情况下，改造的那些道路中分值最大的道路分值尽量小。

任务：作为市规划局的你，应当作出最佳的决策，选择那些道路应当被修建。

例题 06：繁忙都市 题面（2/2）

输入格式

第一行有两个整数 n, m 表示城市有 n 个交叉路口，m 条道路。
接下来 m 行是对每条道路的描述，u, v, c 表示交叉路口 u 和 v 之间有道路相连，分值为 c。

输出格式

两个整数 s, max，表示你选出了几条道路，分值最大的那条道路的分值是多少。

数据范围

对于 100% 的数据：$1 \le n \le 300$，$1 \le c \le 10000$。

例题 06：繁忙都市 解法分析

- **目标**：
  - 选择 $N-1$ 条边连接所有点，使得其中最贵的边最便宜。
- **解法**：
  - 这正是瓶颈生成树的定义。
  - 直接运行 Kruskal 算法求 MST。
  - MST 中最后加入的那条边（权值最大）即为答案。

例题 07：北极通讯网络 题面（1/2）

题目描述

国防部计划用无线网络连接若干个边防哨所。2 种不同的通讯技术用来搭建无线网络；每个边防哨所都要配备无线电收发器；有一些哨所还可以增配卫星电话。

任意两个配备了一条卫星电话线路的哨所（两边都拥有卫星电话）均可以通话，无论他们相距多远。而只通过无线电收发器通话的哨所之间的距离不能超过 D，这是受收发器的功率限制。收发器的功率越高，通话距离 D 会更远，但同时价格也会更贵。

收发器需要统一购买和安装，所以全部哨所只能选择安装一种型号的收发器。换句话说，每一对哨所之间的通话距离都是同一个 D。
你的任务是确定收发器必须的最小通话距离 D，使得每一对哨所之间至少有一条通话路径（直接的或者间接的）。

例题 07：北极通讯网络 题面（2/2）

输入格式

第一行两个整数 S 和 P，S 表示可安装的卫星电话的哨所数，P 表示边防哨所的数量。
接下来 P 行，每行描述一个哨所的平面坐标 (x, y)，以 km 为单位，整数。

输出格式

一行一个实数 D，表示无线电收发器的最小传输距离。精确到小数点后两位。

数据范围

对于 20% 的数据：$P = 2$，$S = 1$；
对于另外 20% 的数据：$P = 4$，$S = 2$；
对于 100% 的数据：$1 \le S \le 100$，$S < P \le 500$，$0 \le x, y \le 10000$。

例题 07：北极通讯网络 解法分析

- **核心转化**：
  - 拥有卫星电话的哨所可以视为全部连通（或者说在同一个“超级连通块”中）。
  - 问题等价于：将 P 个点通过边权 $\leq D$ 的边连接成 **不超过 S 个连通块**。
  - 希望 D 越小越好。
- **解法**：
  - 这依然是 **Kruskal** 算法的应用。
  - 初始有 P 个连通块。
  - 每次选择最小的边合并两个连通块，连通块数量减 1。
  - 当连通块数量降至 S 时停止。
  - 此时刚刚加入的那条边的权值，即为最小的 D。

4.2 最小瓶颈路

- **定义**：
  - 在一张无向图中，找出从 u 到 v 的一条简单路径，使得路径上行所有边中 **最大值最小**。
- **性质**：
  - MST 上 u 到 v 的路径，就是 u 到 v 的最小瓶颈路。
  - 等价：两点间在 MST 上的路径，其最大边权 $\leq$ 原图中任意其他路径的最大边权。

最小瓶颈路问题分为 单次查询 和 多次查询 两大类

4.2.1 最小瓶颈路的单次查询

- 利用 Kruskal 的过程，将所有边升序排列，枚举每条边，尝试将边两端点所在集合进行合并。
- 由于边的枚举是由小到大，对于 $u \to v$ 的单次查询，当 u 和 v 所在的集合第一次连通时，那条刚加入的边的权值就是答案。

例题 08：营救 题面（1/2）

题目描述

“咚咚咚……”“查水表！”原来是查水表来了，现在哪里找这么热心上门的查表员啊！小明感动得热泪盈眶，开起了门……

妈妈下班回家，街坊邻居说小明被一群陌生人强行押上了警车！妈妈丰富的经验告诉她小明被带到了 t 区，而自己在 s 区。

该市有 m 条大道连接 n 个区，一条大道将两个区相连接，每个大道有一个拥挤度。小明的妈妈虽然很着急，但是不愿意拥挤的人潮冲乱了她优雅的步伐。所以请你帮她规划一条从 s 至 t 的路线，使得经过道路的拥挤度最大值最小。

例题 08：营救 题面（2/2）

输入格式

第一行有四个用空格隔开的 n，m，s，t。

接下来 m 行，每行三个整数 u, v, w，表示有一条大道连接区 u 和区 v，且拥挤度为 w。

输出格式

输出一行一个整数，代表最大的拥挤度。

两个区之间可能存在多条大道。

数据范围

对于 100% 的数据：保证 $1 \le n \le 10^4$，$1 \le m \le 2 \times 10^4$，$w \le 10^4$，$1 \le s, t \le n$。
且从 s 出发一定能到达 t 区。

例题 08：营救 解法分析

- **问题**：从起点到终点，使得经过的拥挤度（边权）最大值最小。
- **单次查询解法**：
  - 直接在 Kruskal 过程中，当 S 和 T 连通时停止，当前边权即答案。
  - 类似 Prim 或 Dijkstra 的变种（“宽搜+二分”或“最短路变种”）。

4.2.2 最小瓶颈路的多次查询

- **场景**：如果有 Q 次询问 (u, v) 的最小瓶颈路。
- **解法**：
  1. 构建 MST。
  2. 问题转化为：查询 MST 上 u, v 路径上的最大边权。
  3. 使用 **倍增 LCA (Lowest Common Ancestor)**。
  4. 预处理 max_edge[u][i] 表示 u 向上跳 2^i 步路径上的最大边权。
  5. 复杂度：预处理 $O(N \log N)$，单次查询 $O(\log N)$。

LCA的倍增数组可以预处理 max_edge

4.3 次小生成树

- **定义**：
  - 边权和仅次于 MST 的生成树。
  - 通常区分**非严格**和**严格**两种。

次小生成树算法图解 (1/3)

步骤 1：构建 MST

- 先求出最小生成树（黑色实线），权值和为 $Cost_{MST}$。
- 此时图中还有若干条**非树边**（红色虚线）。
-

次小生成树算法图解 (2/3)

步骤 2：枚举非树边与环路

- 选定一条非树边 (u, v)，权值为 w_{new}。
- 在 MST 上找到 u 到 v 的唯一路径（蓝色高亮）。
- 此时 (u, v) 与树上路径形成了一个**环**。
-

次小生成树算法图解 (3/3)

步骤 3：替换最大边

- 在树上路径中找到权值**最大**的边 (x, y)，权值为 w_{max}。
- 断开 (x, y)，连接 (u, v)。
- 新生成树权值 $= Cost_{MST} - w_{max} + w_{new}$。
- 遍历所有非树边，取最小的新权值即为次小生成树。
-

4.3.1 非严格次小生成树

- **定义**：权值和 $\geq$ MST 权值和。
- **算法思想**：
  - 枚举每条不在 MST 中的边 (u, v, w)。
  - 在 MST 中，u 到 v 的路径上必然有一条最大边，设权值为 val_{max}。
  - 尝试用 (u, v) 替换这条最大边。
  - 新的树权值 $= MST - val_{max} + w$。
  - 取所有替换方案中的最小值。

例题 09：帝国大道 题面（1/3）

题目描述

猪猪帝国的道路状况非常糟糕，多年来一直被忽视和缺乏维护。每条道路连接两个不同的城市 A 和 B，可以双向行驶。每对城市之间最多只有一条道路，利用现有的道路可以在任意一对城市之间旅行。猪猪帝国的新皇帝刚刚提高了税收，并承诺修复至少一些道路，保证只需修复一些道路便能在任意两个城市之间旅行。

给定猪猪帝国道路的描述及各自的修复成本，请编写一个程序来回答一组查询：
对于每个查询，将会得到一个特定的应该修复的道路，必须确定修复一组道路的最低成本（包括给定的特定道路），以便只需修复一些道路便能在任意两个城市之间旅行。

例题 09：帝国大道 题面（2/3）

输入格式

第一行包含两个整数 N 和 R，分别表示猪猪帝国中城市的数量和道路的数量。城市由从 1 到 N 的不同整数标识。

接下来的 R 行中，每行描述一条道路，包含三个整数 A、B 和 C，表示城市 A 和 B 之间有一条道路，修复成本为 C（每对城市之间最多只有一条道路，利用现有的道路可以在任意一对城市之间旅行）。

接下来一行包含一个整数 Q，表示查询的数量。接下来的 Q 行中，每行描述一个查询，包含两个整数 U 和 V，表示应该修复的特定道路。

数据保证：没有重复的查询。

例题 09：帝国大道 题面（3/3）

输出格式

输出 Q 行，每一行都有一个整数，表示输入的相应查询的答案，即修复一组道路（包括查询中的特定道路）的最小成本，以便克格罗尼亚人能够仅使用修复的道路在任何一对城市之间旅行。

数据范围

对于 100% 的数据：$2 \le N \le 10^5$，$N-1 \le R \le 2 \times 10^5$，$1 \le A \le B \le N$，$1 \le C \le 10^4$，$1 \le Q \le 10^5$，$1 \le U < V \le N$。

例题 09：帝国大道 解法分析

- **问题转化**：
  - 题目要求**强制选择**某条边 (U, V) 后的最小生成树。
  - 这是一个典型的 **“特定边生成树”** 问题，本质上与次小生成树的思想一致。

例题 09：帝国大道 解法分析

- **分类讨论**：
  1. **若 (U, V) 就在原 MST 中**：
     - 答案即为 MST 的总权值。
  2. **若 (U, V) 不在原 MST 中**：
     - 强制加入 (U, V) 会在 MST 上形成一个**环**。
     - 为了恢复树结构，必须断开环上的一条边。
     - 为了成本最低，应断开环上（除了新加入的边外）**权值最大**的边。
     - 即：MST 上 $U \to V$ 路径上的最大边。

例题 09：帝国大道 解法分析

- **算法流程**：
  1. 求 MST，记录总权值 Sum。
  2. 倍增法预处理 MST（LCA + 路径最大值）。
  3. 对于每个查询 (U, V)，计算 $Sum - max_edge(U, V) + w(U, V)$。

例题 10：始皇修路 题面（1/3）

题目描述

中国有 n 个城市，秦始皇希望它们都能通过 $n-1$ 条道路相连，以便他可以从都城咸阳到每个城市。

始皇希望所有道路的总长度尽可能最小，这样道路系统就不会夺走太多人的生命。一个名叫许复的道士告诉秦始皇，他可以用魔法修建一条道路，那条魔法道路既不需要花费金钱，也不需要劳动。但许复只能为秦始皇修建一条魔法道路。因此秦始皇必须决定在哪里修建这条魔法道路。
秦始皇希望所有非魔法道路的总长度尽可能小，而许复希望魔法道路能造福尽可能多的人——所以秦始皇决定 A/B 的值必须是最大的，其中 A 是由魔法道路连接的两个城市的总人口，B 是所有非魔法道路的总长度。

一个城市可以被视为一个点，一条道路可以被视为连接两个点的线段。

例题 10：始皇修路 题面（2/3）

输入格式

第一行包含一个整数 t，表示有 t 个测试用例。

对于每个测试用例：
第一行是一个整数 n ，表示有 n 个城市。
然后跟随 n 行：每行包含三个整数 X、Y 和 P 。(X,Y) 是一个城市的坐标，P 是该城市的人口。

保证每个城市的位置都是不同的。

输出格式

对于每个测试用例，输出一行表示上述提到的最大比率 A/B 。结果应四舍五入保留小数点后 2 位。

例题 10：始皇修路 题面（3/3）

数据范围

对于 100% 的数据：$1 \le t \le 10$，$2 < n \le 1000$，$0 \le X, Y \le 1000, 0 < P < 10^5$。

例题 10：始皇修路 解法分析

- **核心转换**：
  - 题目要求“非魔法道路总长最小”。这意味着，如果选定 (u, v) 作为魔法路（费用为 0），剩下的边构成的树，其权值和必须最小。
  - 其实最终的树结构（如果不看权值）本质上还是**最小生成树**的形态（或者通过边替换得到）。
  - **结论**：对于任意点对 (u, v)，如果连魔法路，此时剩下的“非魔法道路总长” $B = W_{MST} - w(max_edge(u, v))$。
  - 其中 $w(max_edge(u, v))$ 是原 MST 上 $u \to v$ 路径上的最大边权。

例题 10：始皇修路 解法分析

- **算法流程**：
  1. **最小生成树预处理**：对全图所有边按权值排序，执行 Kruskal 算法构建 MST。记录 MST 的总权值 Sum。
  2. **树上倍增预处理**：在 MST 上进行 DFS，利用倍增法（Binary Lifting）预处理 $fa[u][i]$ 和 $Max[u][i]$。
     - $Max[u][i]$ 表示从 u 向上跳 2^i 步路径上的最大边权。
  3. **枚举求解**：遍历每一条边 (u, v)（包括树边和非树边）：
     - **若为树边**：该边即为 MST 的一部分，将其作为魔法道路（权值为 0），则总代价为 $Sum - w(u, v)$。
     - **若为非树边**：连接 (u, v) 会形成环。为了恢复树结构，必须移除环上权值最大的边（即 MST 上 $u \to v$ 路径上的最大边，利用 LCA 查询得到）。总代价为 $Sum - query_max(u, v)$。
  4. **计算比率**：对于每条边，计算 $\frac{P_u+P_v}{New_Sum}$，更新全局最大值。

4.3.2 严格次小生成树

- **定义**：权值和 $>$ MST 权值和。
- **挑战**：
  - 如果 $w == val_{max}$，替换后权值不变，不是严格次小。
- **解法**：
  - 维护路径上的**最大值** val_{max} 和**次大值** val_{second}。
  - 如果 $w > val_{max}$，替换最大值。
  - 如果 $w == val_{max}$，替换次大值（从而保证总权值增加）。

例题 11：小C的生成树 题面（1/2）

题目描述

小 C 最近学了很多最小生成树的算法，$Prim$ 算法、$Kurskal$ 算法、消圈算法等等。正当小 C 洋洋得意之时，小 P 又来泼小 C 冷水了。小 P 说，让小 C 求出一个无向图的次小生成树，而且这个次小生成树还得是严格次小的，也就是说：如果最小生成树选择的边集是EM，严格次小生成树选择的边集是 ES，那么需要满足：（$value(e)$ 表示边 e 的权值）

$\sum_{e\in E_M} value(e) < \sum_{e\in E_S} value(e)$

这下小 C 蒙了，他找到了你，希望你帮他解决这个问题。

例题 11：小C的生成树 题面（2/2）

输入格式

第一行包含两个整数 N 和 M，表示无向图的点数与边数。 接下来 M 行，每行 3 个数 $x\ y\ z$ 表示，点 x 和点 y 之间有一条边，边的权值为 z。

输出格式

包含一行，仅一个数，表示严格次小生成树的边权和。(数据保证必定存在严格次小生成树)

数据范围

数据中无向图**不保证无自环**

对于 100%的数据： $N \le 10^5$，$M \le 3 \times 10^5$，边权 $\in [0, 10^9]$。

94

例题 11：小C的生成树 解法分析

- **核心思路**：
  - 与非严格次小生成树类似，枚举非树边 (u, v, w) 尝试替换。
  - **关键区别**：如果 $w == max1$（路径最大值），替换后权值不变，不满足“严格”要求。
  - **对策**：此时必须替换路径上的 **严格次大边** $(max2)$。

例题 11：小C的生成树 解法分析

- **算法流程**：
  1. 求 MST，统计总权值 Sum。
  2. **倍增预处理**：维护 u 向上 2^i 步路径上的 **最大值** $Max1[u][i]$ 和 **严格次大值** $Max2[u][i]$。
  3. 枚举非树边 (u, v, w)，查询路径上的 $m1, m2$：
     - 若 $w > m1$，更新 $Ans = \min(Ans, Sum - m1 + w)$。
     - 若 $w == m1$，更新 $Ans = \min(Ans, Sum - m2 + w)$。
- **复杂度**：$O(M \log M + M \log N)$。

4.4 Kruskal 重构树

- **定义与构造**：
  - 在 Kruskal 算法执行过程中，每当合并两个集合 u, v（边权为 w）时：
    - 新建一个虚拟节点 ext，点权设为 w。
    - 将 $find(u)$ 和 $find(v)$（作为两个子树的根）连向 ext。
    - 将 $find(u)$ 和 $find(v)$ 的父指针指向 ext。
- **结果**：
  - 最终形成一棵二叉树（叶子是原图节点，非叶子是边权化点）。
  - 根节点代表最后合并的边。

Kruskal 重构树的性质

1. **堆性质**：
   - 从叶子到根，点权单调递增（因为 Kruskal 是按边权从小到大合并的）。
   - 它是大顶堆。
2. **LCA 性质**：
   - 原图中 u, v 两点路径上的瓶颈边权（最大边权） = 重构树上 $LCA(u, v)$ 的点权。

例题 12：星际导航 题面(1/3)

题目描述

外星喵做好了回到喵喵星球的硬件准备，但是飞船上的导航系统还没有完全设计好。为了方便起见，可以认为宇宙是一张有 N 个顶点和 M 条边的带权无向图，顶点表示各个星系，两个星系之间有边就表示两个星系之间可以直航，而边权则是航行的危险程度。

外星喵想把回家的危险程度降到最小，即有对于若干个 (A, B) 的询问，外星喵想知道从顶点 A 航行到顶点 B 所经过的最危险的边的 **危险程度值** 最小可能是多少。
作为外星喵的同学，请帮助外星喵返回家园，兼享受安全美妙的宇宙航行。

例题 12：星际导航 题面(2/3)

输入格式

第一行包含两个正整数 N 和 M，表示点数和边数。
之后 M 行，每行三个整数 A，B 和 L，表示顶点 A 和 B 之间有一条边长为 L 的边。顶点从 1 开始标号。

下面一行包含一个正整数 Q，表示询问的数目。

之后 Q 行，每行两个整数 A 和 B，表示询问 A 和 B 之间最危险的边危险程度的可能最小值。

输出格式

对于每个询问， 在单独的一行内输出结果。如果两个顶点之间不可达， 输出 impossible。

100

对于 40% 的数据：满足 $N \le 1000, M \le 3000, Q \le 1000$。
对于 80% 的数据：满足 $N \le 10000, M \le 10^5, Q \le 1000$。
对于 100% 的数据：满足 $N \le 10^5, M \le 3 \times 10^5, Q \le 10^5, L \le 10^9$。数据不保证没有重边和自环。

例题 12：星际导航 解法分析

- **题目解析**：
  - 典型的 **最小瓶颈路** 问题。
  - 需要频繁查询两点间“最大边权最小”的路径。

例题 12：星际导航 解法分析

- **解法一：MST + 倍增**
  - 构建 MST。
  - 倍增维护路径最大值，查询 $O(Q \log N)$。

例题 12：星际导航 解法分析

- **解法二：Kruskal 重构树 (推荐)**
  - 构建 Kruskal 重构树。
  - **性质**：原图 u, v 瓶颈路的最大边权 = 重构树上 $LCA(u, v)$ 的点权。
  - **流程**：
    1. 构建重构树，记录每个非叶节点的权值。
    2. 对每个询问 (A, B)，若不连通输出 [illegible]。
    3. 否则求 $LCA(A, B)$，其点权即为答案。
  - **复杂度**：$O(M \log M + Q \log N)$，代码更简洁。
  - 处理多组询问的连通性或瓶颈路限制问题。
  - 相比倍增 LCA 直接在 MST 上维护边权，重构树将“边权”化为“点权”，处理更直观。
  - 只需在重构树上找 LCA 即可回答瓶颈路问题。

104

请批评指正

Q & A

请多加练习，巩固图论基础！