最小生成树与进阶应用

稀疏图 Kruskal 并查集加边 vs 稠密图 $O(n^2)$ Prim 贪心建树

HGNU-ACM 实验室算法组

配套：切分定理证明 + Kruskal/Prim 选型 + 虚拟源点/瓶颈/次小生成树 + 4 大例题精讲





本章目录（上）：核心概念与两大算法

最小生成树（MST）与切分定理

Kruskal 算法：按边排序 + 并查集判环

Prim 算法与双算法选型对比

第 1～3 节：从修路省钱引入最小生成树，彻底搞懂切分定理为什么让贪心一定对，再逐行吃透 Kruskal 与 Prim 两种实现。

本章目录（下）：进阶变形、例题与收尾

MST 进阶应用与三大经典变形

四大经典例题精讲

常见错误、检查清单与练习安排

第 4～6 节：掌握虚拟源点、瓶颈生成树、严格次小生成树与重构树直觉，精讲 4 道经典真题并完成提交前自查。

承前启后：从“一个人走得最快”到“让所有村都通车”

生活场景：村村通修路工程

假设山里有 4 个村子，村子之间有 5 条可以修路的土路，每条路造价不同。政府要求：任意两个村子之间都能互相走到，但总预算越少越好！

- 第 3 章《最短路》解决的是：从 1 号村出发，到每个村分别怎么走最近（关心的是单源距离之和或路径）。
- 本章《最小生成树》解决的是：只挑一部分路来修，把所有村子连成一张连通网，使总造价最低。
- 既然想省钱，任何多余的环都不该修——只要出现环，删掉环上任意一条路，村子依然全连通，造价却更低！

本页要点

“所有点都连通，且一条多余的环边都没有”——这样的图就是树，总边权最小的那棵就叫最小生成树（**MST**）！

新术语大白话：树、生成树与最小生成树

术语	一句大白话解释
树（**Tree**）	n 个点、恰好 $n-1$ 条边、没有环且全部连通的骨架图
生成树	从原图里挑出全部 n 个点和恰好 $n-1$ 条边，组成一棵连通的树
生成树的权值	挑出来的这 $n-1$ 条边的边权总和
最小生成树（**MST**）	所有可能的生成树里，边权总和最小的那一棵（可能有并列多棵）

- 为什么恰好是 $n-1$ 条边？少于 $n-1$ 条边一定无法把 n 个点连成一块；多于 $n-1$ 条边一定会多出环。

本页要点

记住公式：n 个点的连通生成树，边数不多不少，永远是 $n-1$ 条！

为什么不能暴力枚举所有边？

先算一算暴力做法的操作次数

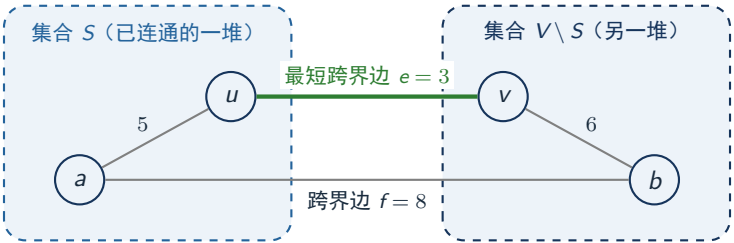
如果有 $n = 1000$ 个点、$m = 5000$ 条边，暴力枚举从 5000 条边里选 999 条，一共有 $\binom{5000}{999}$ 种选法，这是一个几百位数的天文数字，远远超过计算机每秒 10^8 次的极限，必定 **TLE**！

- 直觉猜想：既然总和要最小，我们能不能每次都贪心地挑当前最便宜、且不会形成环的边？
- 必须打个问号：在最短路或背包里，局部贪心有时会掉坑（比如 0-1 背包不能按性价比贪心）。修路挑最便宜的边，会不会把后面“卡死”，导致总造价反而变贵？

本页要点

暴力枚举组合数爆炸必超时；接下来我们要用“切分定理”严格证明：MST 贪心挑最便宜的跨界边，永远不会后悔！

切分定理（Cut Property）：把点分成两半看跨界边



本页要点

切分定理：把点任意分成 S 和 $V \setminus S$ 两堆，所有横跨两堆的边里，权值最小的那条边 e，一定属于某棵最小生成树！

为什么切分定理一定对？（替换论证 / 反证法）

假设存在一棵“没选 e”的最优树 T^*

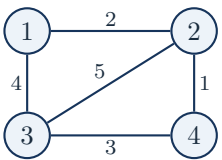
1. 既然 T^* 是一棵生成树，它必须把全图所有点都连通，所以 u 和 v 在 T^* 上一定有路径相连。
2. 这条路径从 S 走到 $V \setminus S$，中间必然用到了至少一条跨界边，设它为 f。因为 e 是跨界边里最便宜的，所以 $w(e) \le w(f)$。
3. 现在我们把 e 加进 T^*，就会和刚才的路径形成一个环；再把环上更贵的 f 剪掉！
4. 新树 $T' = T^* \cup \{e\} \setminus \{f\}$ 依然连通、依然有 $n-1$ 条边，且总造价 $w(T') = w(T^*) + w(e) - w(f) \le w(T^*)$。

本页要点

剪掉更贵的 f、换上更便宜的 e，树照样连通还更省钱——所以贪心选最小跨界边绝对安全！

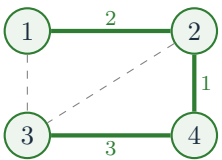
小例子手算：4 个点 5 条边一步步挑出 MST

原图（**4** 点 **5** 边）



挑最小 3 边

最小生成树（总权 $= 1 + 2 + 3 = 6$）



本页要点

按边权 $1, 2, 3$ 依次连上 $(2,4), (1,2), (3,4)$ 后，4 个点全部连通；剩下的边 4 和 5 都会成环，直接舍弃！

如果图本身就不连通怎么办？认识“生成森林”

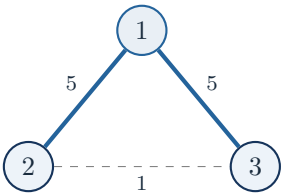
- 并不是所有输入图都是连通的！比如 4 个村子之间只修了 2 条路，中间隔着断崖。
- 在不连通的图上，我们最多只能连出几棵互不相连的小树，合起来叫做最小生成森林。
- 程序怎么判断原图不连通？
  - 统计成功加进生成树的边数 `cnt`。
  - 如果跑完所有边后 `cnt < n - 1`，说明无论怎么修都连不成一个整体，按题目要求输出 `impossible` 或 `-1`。
- 特殊边界 $n = 1$（只有 1 个点、0 条边）：此时需要 $n - 1 = 0$ 条边，`cnt = 0` 恰好等于 $n - 1$，总造价为 0，是连通的！

本页要点

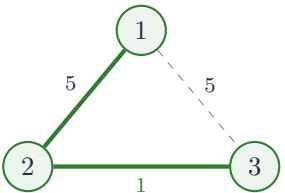
用 `cnt < n - 1` 判不连通，既能拦住断成两半的图，又能完美兼容 $n = 1$ 的极小边界！

常见认知误区：最短路树 $\neq$ 最小生成树

从 1 出发的最短路树（总权 10）



最小生成树（总权 6）



本页要点

最短路关心“每个人到起点的距离”，MST 关心“所有修好的路总长”；上图中最短路树总权 $5+5=10$，而 MST 只要 $5+1=6$！

本节小结：3 个概念自测小问

1. 如果图中所有边权都相等（比如全为 1），最小生成树有多少条边？总权值是多少？
   - 只要图连通，任何一棵生成树都是 MST，恰好 $n-1$ 条边，总权值 $n-1$。
2. 如果边权里有负数（比如政府补贴倒贴钱），**MST** 算法还能用吗？
   - 能用！切分定理不要求边权为正，而且选负权边总和更小。注意：即便负权边多，MST 也只选 $n-1$ 条边（不能留环）。
3. 如果题目保证图连通，但有自环（自己连自己）和重边（两点间多条路），需要手动删掉吗？
   - 不需要！自环和较贵的重边都不会让总权更小，算法会自动跳过它们。

本页要点

最小生成树不怕负权、不怕自环和重边；只要抓紧“$n-1$ 条边、无环、跨界最短边安全”即可！

Kruskal 核心想法：先修最便宜的桥，会成环就扔掉

把切分定理变成最直白的操作步骤

1. 把所有 m 条边按边权 w 从小到大排序。
2. 一开始每个点各自是一座孤岛（共 n 个连通块）。
3. 从小到大逐条看每条边 (u, v, w)：
   - 如果 u 和 v 还不在同一个连通块里，说明它是一条跨界最短安全边 $\rightarrow$ 选它！合并两块，累加边权。
   - 如果 u 和 v 已经在同一个连通块里，再连这条边就会出现环 $\rightarrow$ 直接跳过！

本页要点

Kruskal 全程只干两件事：按边权排序 + 判两点是否已连通！

怎么 $O(1)$ 级快速判两点是否已连通？——并查集回顾

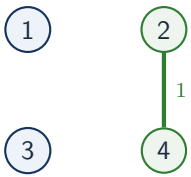
- 如果每次判“u 和 v 能不能互相走到”都跑一遍 DFS，判一次就要 $O(n)$，总复杂度退化成 $O(mn)$，照样 **TLE**！
- 并查集（**Union-Find**）给每个连通块选一个“老大（根节点）”，用数组 `p[x]` 记录每个点指向谁：
  - 初始化：一开始每人自己是自己的老大，`p[i] = i`。
  - 查老大 `find(x)`：顺着 `p[x]` 往上找根，并在回溯时把沿途点的父亲直接改成根（路径压缩），平均几乎 $O(1)$。
  - 判环与合并：若 `find(u) != find(v)`，说明不在同一块，令 `p[rv] = ru` 合并即可！

本页要点

并查集把“判两点连没连通”从 $O(n)$ 压缩到近似 $O(1)$，让 Kruskal 跑得飞快！

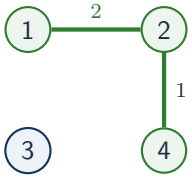
Kruskal 执行过程图解（4 个点 5 条边）

选 $(2,4), w=1$



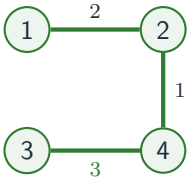
块：{1}{2,4}{3}

选 $(1,2), w=2$



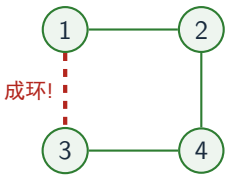
块：{1,2,4}{3}

选 $(3,4), w=3$



块：{1,2,3,4} 已齐

跳过 $(1,3), w=4$



`find(1)==find(3)`

本页要点

每选进一条边，连通块数量就减少 1；当遇到两端老大相同的边，连上去必成环，直接扔掉！

手算跟踪表（5 点 7 边小图，含跳过与终止步）

轮次	当前边 (u, v, w)	find(u)	find(v)	动作	ans	cnt
初态	—	—	—	5 个孤立块 $\{1\}\{2\}\{3\}\{4\}\{5\}$	0	0
1	$(1, 2, 2)$	1	2	合并 $\{1, 2\}$	2	1
2	$(3, 4, 3)$	3	4	合并 $\{3, 4\}$	5	2
3	$(2, 3, 4)$	1	3	合并 $\{1, 2, 3, 4\}$	9	3
4	$(1, 4, 5)$	1	1	同块成环，跳过	9	3
5	$(4, 5, 6)$	1	5	合并 $\{1, 2, 3, 4, 5\}$	15	4
6	$(2, 5, 7)$	1	1	同块成环，跳过	15	4
终止	边已遍历完	—	—	检查 cnt $= 4 ==$ $n - 1$，输出 15	15	4

本页要点

手算要点：哪怕第 5 步已经凑够 $n - 1 = 4$ 条边，只要循环没提前 break，第 6 步也会因为同块被安全跳过！

为什么 Kruskal 一定对？——循环不变量论证

阶段	循环不变量：当前已选边集 A 一定是某棵最小生成树 T^* 的子集
初始化	循环开始前 $A=\varnothing$，空集显然是任何生成树的子集，成立。
保持	若当前边 $e=(u,v)$ 连接两个不同连通块，把所有点按“是否与 u 同块”切分成两半，e 是所有未看过的跨界边里最便宜的。由切分定理，$A\cup\{e\}$ 仍是某棵 MST 的子集！
终止	遍历完所有边后，若 $\mathrm{cnt}=n-1$，A 无环且有 $n-1$ 条边，本身就是一棵完整生成树，故它就是 MST！

本页要点

每一步都只挑“当前合法且最便宜的跨界边”，就像拼图一样一块块安全拼出最终的最小生成树。

Kruskal 完整模板代码（1/2）：读入、排序与并查集

```cpp
#include <bits/stdc++.h>
using namespace std;
struct Edge { int u, v; long long w; };              // 边权可能达 10^9，存成 long long
int main() {
    ios::sync_with_stdio(false); cin.tie(nullptr);
    int n, m; if (!(cin >> n >> m)) return 0;
    vector<Edge> es(m);
    for (auto &[u, v, w] : es) cin >> u >> v >> w;
    sort(es.begin(), es.end(), [](const Edge &a, const Edge &b) {
        return a.w < b.w;                            // 按边权严格升序排序
    });
```

本页要点

用 `struct Edge` 存边表，无向图只存 m 条边即可，不需要连双向边存 $2m$ 条！

Kruskal 完整模板代码（2/2）：并查集判环与判连通

```
    vector<int> p(n + 1); iota(p.begin(), p.end(), 0); // p[i] = i 初始化并查集
    auto find = [&](auto &&self, int x) -> int {
        return p[x] == x ? x : (p[x] = self(self, p[x])); // 路径压缩查根
    };
    long long ans = 0; int cnt = 0;
    for (auto &[u, v, w] : es) {
        int ru = find(find, u), rv = find(find, v);
        if (ru != rv) { p[rv] = ru; ans += w; ++cnt; } // 不同块才合并，累加边权与边
    数
    }
    if (cnt < n - 1) cout << "impossible\n";        // 选不满 n-1 条边说明图不连通
    else cout << ans << '\n';
}
```

本页要点

完整源码见 code/kruskal_mst.cpp；注意 ans 开 long long，合并后记得 ++cnt！

为什么这样写？（3 个高频疑问一次讲透）

- 为什么无向图用 **Kruskal** 只要存 m 条边，不用存 $2m$ 条？
  - 因为 Kruskal 不从点出发找邻居，而是直接拿一条边 (u, v) 的两个端点去并查集里合并，(u, v) 和 (v, u) 合并效果完全一样。
- 为什么 `ans` 要开 `long long`，即使单条边权在 `int` 范围内？
  - 比如 $n = 10^5$ 个点、每条边权 10^9，总和高达 10^{14}，远超 `int` 上限 2.14×10^9，不开 `long long` 必溢出成负数！
- 为什么不提前 `if (cnt == n - 1) break;`？
  - 可以加上提前退出稍微加速，但就算不加，后面所有边都会因为 `ru == rv` 被跳过，时间复杂度仍然是排序的 $O(m \log m)$。

本页要点

Kruskal 代码短、坑点集中在数据类型（`long long`）与判连通（`cnt < n - 1`）。

上机验证：3 组测试输入输出与反事实追问

测试场景	输入数据	预期输出与说明
1. 普通连通图	`4 5 / 1 2 2, 1 3 4, 2 4 1, 3 4 3, 2 3 5`	6（选边权 $1+2+3=6$）
2. 最小边界 $n=1$	`1 0`	0（$cnt=0==n-1$）
3. 不连通图	`4 2 / 1 2 5, 3 4 7`	impossible（$cnt=2<3$）

反事实追问

如果在模板里不小心删掉了 `if (ru != rv)` 判环，直接无脑合并并累加边权，第 1 组输入会输出什么？

$\rightarrow$ 会把全部 5 条边权相加输出 15（甚至多出环），完全失去生成树意义！

本页要点

写完模板一定要亲手测一遍 $n=1$ 和断成两块的图，确保都能正确输出！

Kruskal 易错对照（WA / TLE / RE 定位）

错误表现	常见错误写法	正确做法与原因
WA（溢出成负）	`int ans = 0;`	边权总和用 `long long ans = 0;`
WA（漏判不连通）	直接输出 `ans` 不看 `cnt`	必须检查 `if (cnt < n - 1)`
WA/死循环	忘记写 `iota(p.begin(), p.end(), 0);`	`p[i]` 默认全是 0，会导致 `find` 误判全部已连通
RE（数组越界）	点编号从 $1 \sim n$ 但只开 `p(n)`	并查集大小至少开 `n + 1`

本页要点

Kruskal 时间复杂度为 $O(m \log m)$，只要 $m \le 5 \times 10^5$ 都能稳过；若遇到 $n \le 2000$ 的稠密矩阵图，下一节的 Prim 更好用！

换个视角建树：从 1 个起点“一圈圈向外拉最近的点”

生活比喻：修地铁网

Kruskal 是“全国最便宜的路段先修，东一截西一截最后拼起来”；
而 **Prim** 是从中心站（比如 0 号点）出发，每次在已经通车的站与还没通车的站之间，挑一条最便宜的铁轨把新站拉进来！

- 每一轮都把全图切分成“已入树集合 S”和“未入树集合 $V \setminus S$”。
- 由切分定理，连接 S 和 $V \setminus S$ 的最短边一定属于最小生成树！
- 所以我们只需维护一个数组 `low[v]`：记录点 v 连向当前整棵树 S 的最小单边权。

本页要点

Kruskal“以边为主角”，适合边少的稀疏图；Prim“以点为主角”，每次拉一个离树最近的新点入伙！

对比第 3 章 Dijkstra：dist[v] vs low[v]

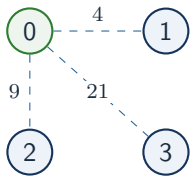
对比项	Dijkstra（单源最短路）	Prim（最小生成树）
数组含义	`dist[v]`：起点 s 到 v 的总路程	`low[v]`：v 到已建成树 S 的单边长度
松弛公式	`dist[v] = min(..., dist[u] + w)`	`low[v] = min(..., w[u][v])`
是否累加	答案是各点的 `dist[v]`	答案是每次拉入点时的 `low[u]` 之和
时间复杂度	朴素 $O(n^2)$ / 堆优化 $O(m \log n)$	朴素 $O(n^2)$ / 堆优化 $O(m \log n)$

本页要点

Prim 和 Dijkstra 长得几乎一模一样，唯一核心区别是松弛时不加 `low[u]`，只看新边 `w[u][v]` 本身！

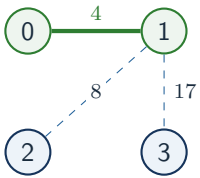
Prim 执行流程独占图解（4 点完全图）

起步：拉入 0 号点



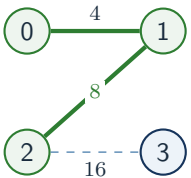
low=[0,4,9,21]

选最小 low[1]=4



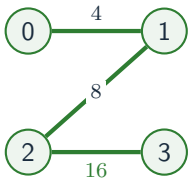
low=[✓,✓,8,17]

选最小 low[2]=8



low=[✓,✓,✓,16]

选 low[3]=16 完成



总和 $4+8+16=28$

本页要点

每拉进一个新点 u，就拿 u 的出边去尝试把树外邻居 v 的 low[v] 变得更小！

Prim 手算跟踪表（对应上图 4×4 邻接矩阵）

轮次 step	选出点 u	累加 low[u]	ans	low[1]	low[2]	low[3]
初态	—	—	0	∞	∞	∞
0	0	0	0	4	9	21
1	1	4	4	已入树	$\min(9,8)=8$	$\min(21,17)=17$
2	2	8	12	已入树	已入树	$\min(17,16)=16$
3	3	16	28	已入树	已入树	已入树
终止	$step=4$ 结束	—	28	—	—	—

本页要点

一共跑 n 轮，每轮恰好拉入 1 个新点；第 0 轮拉入起点 0 时 low[0]=0，后面 $n-1$ 轮共加进 $n-1$ 条树边！

朴素 $O(n^2)$ Prim 完整模板代码（邻接矩阵版）

```
vector<long long> low(n, LLONG_MAX / 4);
vector<char> vis(n, 0);
low[0] = 0; long long ans = 0;                          // 从 0 号点起步，入树代价为 0
for (int step = 0; step < n; ++step) {
    int u = -1;
    for (int v = 0; v < n; ++v)
        if (!vis[v] && (u == -1 || low[v] < low[u])) u = v; // 找树外 low 最小的点
    if (low[u] >= LLONG_MAX / 8) { ans = -1; break; }       // 不连通特判
    vis[u] = 1; ans += low[u];
    for (int v = 0; v < n; ++v)
        if (!vis[v]) low[v] = min(low[v], w[u][v]); // 用 u 的出边更新树外点到树的距
    离
}
```

本页要点

朴素 Prim 只有 12 行核心代码，完全不用排序，也不用写并查集！

为什么这样写？（Prim 模板的 3 个细节）

- 为什么初值设成 `LLONG_MAX / 4` 而不是 `INT_MAX`？
  - 一来防边权过大超过 `int`，二来留足安全余量，配合 `low[u] >= LLONG_MAX / 8` 能稳稳判出不连通图。
- 为什么内层两处都要写 `if (!vis[v])`？
  - 第一处：不能再把已经在树里的点选出来重复加边。
    第二处：不能用 `w[u][v]` 去改树内老点的 `low`，不过其实不写也不影响后续选点，但写上更清晰安全。
- 邻接矩阵输入有自环怎么办？
  - 第 0 轮先标记 `vis[u]=1` 再更新，所以 u 不会更新自己，自环完全不影响结果。

本页要点

“选点判 `!vis[v]`、加点先 `vis[u]=1`、更新只看 `w[u][v]`”，记住这三句就写不错！

为什么 Prim 一定对？——循环不变量论证

阶段	循环不变量：已选点集 S 与对应的边集 A 构成某棵 MST 的一棵连通子树
初始化	$S=\{0\}$，$A=\varnothing$，单点显然是任意生成树的子树。
保持	每轮选出树外 `low[u]` 最小的点 u，这条权为 `low[u]` 的边正是跨越 S 与 $V\setminus S$ 的所有边里最便宜的！由切分定理，把 u 和这条边加入 (S,A) 依然安全。
终止	循环执行 n 轮后 S 包含全部 n 个点，共有 $n-1$ 条边且无环，必为一棵最小生成树。

本页要点

Kruskal 和 Prim 背后站的是同一位大佬——切分定理，只是切分的方式一个按边排、一个按点扩！

上机验证：3 组测试与反事实追问

测试场景	输入矩阵	预期输出
1. 4 点完全图	样例 4×4 矩阵（见上页）	28
2. 最小边界 $n = 1$	`1 / 0`	`0`（只走 1 轮 `low[0]=0`）
3. 不连通图	$n = 3$，第 3 点与前两点边权全为 ∞	`-1`（触发不连通特判）

反事实追问

如果把更新语句 `low[v] = min(low[v], w[u][v])` 误写成 Dijkstra 的 `low[v] = min(low[v], low[u] + w[u][v])`，样例会输出什么？

$\rightarrow$ 第 1 轮后 `low[2]` 会被更新成 $4 + 8 = 12$ 而不是 8，整个算法就变成了求单源最短路之和，而不是最小生成树！

本页要点

写 Prim 时脑子里默念：“我要的是新点离整棵树的距离，不是离起点的距离！”

Prim 易错对照与终极双算法选型表

维度	**Kruskal**	**Prim**（朴素 $O(n^2)$）
时间复杂度	$O(m \log m)$	$O(n^2)$
适用图类型	稀疏图（边数 $m \le 2 \times 10^5$）	稠密图 / 完全图（$n \le 2000$ 且给矩阵）
空间与存图	只需边表结构体，省空间	直接读 $n \times n$ 矩阵，不用建边表
常见 **WA** 坑	忘开 `long long`、漏判 `cnt < n-1`	漏写 `!vis[v]`、混淆 `low[u]+w`

- 实战口诀：给边表选 **Kruskal**（90% 比赛题都用它）；给 $n \times n$ 矩阵或点数少边极多选 朴素 **Prim**（省空间还不用 log）！

本页要点

稀疏图用 Kruskal，稠密矩阵图用 $O(n^2)$ Prim；堆优化 Prim ($O(m \log n)$) 既不如 Kruskal 好写，实际跑得也往往更慢。

遇到“点权 + 边权”怎么建 MST？——虚拟超级源点

常见场景：村里打水井 / 矿井建发电站

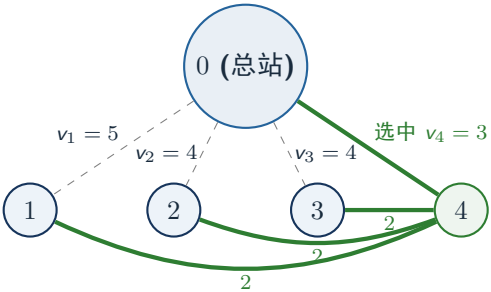
每个点 i 自己建发电站要花 v_i 元，或者不建站而是从别的已有电的矿井拉电线花 $p_{i,j}$ 元。要求所有点都有电，问最少花多少钱？

- 难点：标准 MST 只有“边权”，没有“点权”！而且允许建不止一座发电站（森林而非单棵树）。
- 神来之笔：凭空造一个 0 号超级发电总站！
  - 在点 i 建发电站 $\iff$ 在 0 号点与 i 号点之间连一条权为 v_i 的边！
- 把所有 $n+1$ 个点（$0 \sim n$）连成一棵最小生成树：选中边 $(0, i)$ 就代表“在 i 号点建发电站”！

本页要点

遇到“自己单干有代价 v_i，拉手合作有代价 $p_{i,j}$”，立刻引入 0 号虚拟源点连边 $(0, i, v_i)$，秒变标准 MST！

虚拟超级源点建模独占图解



本页要点

不管原图选几个点自建发电站，在 $n+1$ 个点的 MST 里，它们都会自动连向 0 号点，而且至少会选一条 $(0, i)$ 边！

进阶 2：瓶颈生成树（只关心最大边尽量小）

新问题：只看“最贵的那一条边”

有时候总造价不重要，题目只要求：选出 $n-1$ 条边连成生成树，使得树上权值最大的那条边尽量小。这棵树叫做最小瓶颈生成树。

- 黄金结论：无向图的最小生成树，一定是最小瓶颈生成树！
- 为什么？假设某棵树的最大边比 MST 的最大边 $W_{\max}$ 还小，那在 Kruskal 算法扫描所有 $\le W_{\max}$ 的边之前就已经把全图连通了——这与 Kruskal 必须用到 $W_{\max}$ 才连通矛盾！
- 所以遇到“极小化路径/树上的最大边权”，直接跑 Kruskal 即可！

本页要点

记住结论：MST 天生就是“最大边最小”的瓶颈生成树，不用另写二分答案！

变形：如果允许留下 k 个连通块（卫星设备模型）



- 如果有 k 个点可以“免费空降连线”（如配备卫星），相当于允许全图分成 k 个连通块！
- Kruskal 一开始有 n 个块，每合并 1 次减少 1 个块；只要连通块数降到 k（即合并了 $n-k$ 条边）就停手，最后加入的那条边权就是最小瓶颈值！

本页要点

留 k 个连通块 $\Longleftrightarrow$ 砍掉 MST 中最大的 $k-1$ 条边 $\Longleftrightarrow$ Kruskal 选到第 $n-k$ 条树边直接停止！

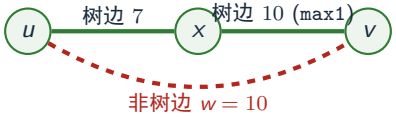
进阶 3：次小生成树（枚举非树边替换路径最大边）

- 核心思路：先求出一棵 MST（总权 M_0）。可以证明，次小生成树一定可以通过往 **MST** 里加一条非树边 (u, v, w)，再删掉树上 $u \to v$ 路径里的某条边得到！
- 要想新树尽量小，删掉的那条树边应该越大越好，也就是 $u \to v$ 路径上的最大边权 `max1`！
- 新生成树权值 $= M_0 - \texttt{max1} + w$。
- 枚举所有非树边，取这个新权值的最小值即可！

本页要点

非严格次小生成树（允许权值与 MST 相等）：只需用非树边 w 替换掉树上 (u, v) 路径的最大边 `max1`。

陷阱！严格次小生成树为什么还要记“严格次大边 max2”？



- 如果非树边权 w 恰好等于路径最大边 max1（如上图 $w = 10, \texttt{max1} = 10$），换完之后总权值 $M_0 - 10 + 10 = M_0$，没有严格变大！
- 因此，当 $w == \texttt{max1}$ 时，必须退而求其次，删掉路径上严格小于 **max1** 的最大边 **max2**（上图中是 7），得到 $M_0 - 7 + 10 = M_0 + 3$！

本页要点

求严格次小生成树，树上路径必须同时维护最大边 **max1** 和严格次大边 **max2**！

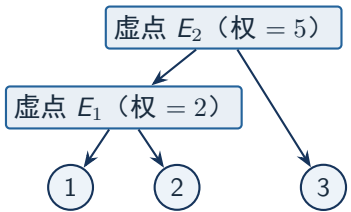
树上路径最大边怎么求？——从 $O(n)$ DFS 到 $O(\log n)$ 倍增

- 当 $n \le 2000$ 时（如一本通 **1490**）：直接从每个起点 $s = 1 \sim n$ 跑一次 DFS，把 `max1[s][v]` 和 `max2[s][v]` 预处理进二维数组，总时间 $O(n^2)$，代码不到 10 行！
- 当 $n \le 10^5$ 时：$O(n^2)$ 存不下，使用树上倍增（**LCA**）：
  - `f[u][k]` 记录 u 向上跳 2^k 步的祖先；
  - `g1[u][k]`，`g2[u][k]` 记录这段路径上的最大边和严格次大边；
  - 查询 (u, v) 只需跳到 LCA，把两段的最大/次大合并，单次查询 $O(\log n)$！

本页要点

数据小（$n \le 2000$）用 $O(n^2)$ 全源 DFS 最稳最快；数据大（$n \le 10^5$）再上倍增 LCA 维护 g1，g2。

进阶 4：Kruskal 重构树直觉（两点瓶颈 = LCA 点权）



- Kruskal 每次合并两个连通块时，不直接连边，而是新建一个点权等于当前边权 w 的虚点，把两个块的老大挂在它下面作为左右儿子。
- 性质：这是一棵大根堆二叉树；原图任意两点 u, v 之间所有路径上“最大边权的最小值”，恰好就是重构树上 $\mathrm{LCA}(u, v)$ 的点权！

本页要点

遇到“多次询问两点之间只走 $\leq x$ 的边能到哪些点”，就用 Kruskal 重构树 + 倍增往上跳！

MST 进阶变形速查表

问题特征	对应模型	一句话解法
点有自建费 v_i，边有连线费	虚拟超级源点	加 0 号点向每个 i 连权 v_i 的边，跑标准 MST
最大边最小 / 留 k 个块	瓶颈生成树	跑 Kruskal 到只剩 k 个块（$n-k$ 条边）即停
要求严格次小生成树	枚举非树边替换	维护树上路径 `max1`，`max2`，逐条非树边尝试替换
多次查 $u \to v$ 瓶颈或可达性	Kruskal 重构树	合并时建虚点权为 w，查 $\mathrm{LCA}(u, v)$ 点权

本页要点

万变不离其宗：所有 MST 进阶题都建立在“切分定理 + Kruskal/Prim 骨架”之上！

例 1 | [基础] 最短网络（一本通 1350）：先读懂要求

大白话题意

给 N（$3 \le N \le 100$）个农场，以及一张完整的 $N \times N$ 矩阵表示任意两个农场之间铺设光纤的费用 $w_{i,j} \le 10^5$（对角线为 0）。求把全部农场连成一张网的最小总费用。

- 输入：第 1 行整数 N；接下来是 $N \times N$ 个整数（注意原题每行可能折行，用 `cin` 读满 N^2 个数即可）。
- 输出：一个整数，表示最小生成树的总边权。
- 样例：$N = 4$，矩阵为 `[0,4,9,21; 4,0,8,17; 9,8,0,16; 21,17,16,0]` $\Longrightarrow$ 输出 `28`。

本页要点

难度标签：**[基础]**。给了完整 $N \times N$ 费用矩阵（完全图），直接求最小生成树总边权！

例 1 | 思路与建模：矩阵图用 $O(n^2)$ Prim 最省事



- 这是一张边数 $M = N(N-1)/2$ 的完全图（稠密图），矩阵已经现成给了出来。
- 如果用 Kruskal，还得把 N^2 个数拆成结构体边表再去排序；用朴素 Prim 直接在二维数组上扫两遍，不用排序、代码更短、运行更快！

本页要点

记住：看见“给 $N \times N$ 矩阵求全连通最小代价”，优先选 $O(n^2)$ 朴素 Prim！

例 1 | 样例手推跟踪表（4×4 矩阵推导 28）

step	选出 u	low[u]	ans	low[1]	low[2]	low[3]
初态	—	—	0	∞	∞	∞
0	0	0	0	4	9	21
1	1	4	4	已入树	$\min(9,8)=8$	$\min(21,17)=17$
2	2	8	12	已入树	已入树	$\min(17,16)=16$
3	3	16	28	已入树	已入树	已入树
终止	$step=4$	—	28	—	—	—

本页要点

手推核对：选边分别是 $(0,1)=4$、$(1,2)=8$、$(2,3)=16$，总和 $4+8+16=28$，与样例输出一致！

例 1 | 核心步骤表与模板改动

按顺序计算	纸上得到	核对问题
1. 读入 $n \times n$ 矩阵 `w`	完整的费用表（0-indexed）	下标从 0 还是 1 开始？
2. 设 `low[0]=0` 其余 ∞	准备从 0 号农场开始扩树	有没有全设成足够大的 ∞？
3. 选树外 `low[u]` 最小点	当前离树最近的农场 u	判断 `!vis[v]` 了没有？
4. `vis[u]=1, ans+=low[u]`	农场 u 正式通网，费用累加	`ans` 开 `long long` 了吗？
5. 用 `w[u][v]` 更新树外 `low`	刷新各未通网农场的最小接法	有没有误写成 `low[u]+w[u][v]`？

本页要点

相对 **Prim** 模板改了哪里？题目保证图一定连通，所以省略了不连通特判，直接跑完 n 轮输出 `ans` 即可。

例 1 | 核心代码与 易错自测（源码 code/prim_agrinet.cpp）

```
vector<long long> low(n, LLONG_MAX / 4); vector<char> vis(n, 0);
low[0] = 0; long long ans = 0;
for (int step = 0; step < n; ++step) {
    int u = -1;
    for (int v = 0; v < n; ++v)
        if (!vis[v] && (u == -1 || low[v] < low[u])) u = v;
    vis[u] = 1; ans += low[u];                    // 挑出离已建成树最近的点拉进来
    for (int v = 0; v < n; ++v)
        if (!vis[v]) low[v] = min(low[v], w[u][v]);
}
cout << ans << '\n';
```

- 易错与自测： 漏写 !vis[v] 会重复选自己（WA）； 自测 3 组：样例 $N = 4 \to 28$；最小边权全 0 矩阵 $\to 0$；最大 $N = 100$ 且边权 $10^5 \to 9.9 \times 10^6$。

本页要点

代码简洁到只有 11 行，完全匹配 code/prim_agrinet.cpp！

例 2 | [基础] 新的开始： 先读懂要求

大白话题意

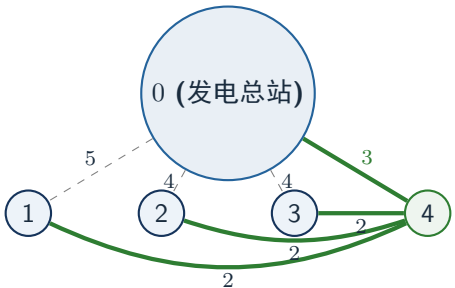
有 n（$1 \le n \le 300$）口矿井。让每口矿井都有电有两种办法： 在第 i 口矿井建发电站，花 v_i 元； 从已有电的矿井 j 拉电网到 i，花 $p_{i,j}$ 元。求让所有矿井都有电的最小总花费。

- 输入：第 1 行 n；第 $2 \sim n+1$ 行每行一个 v_i；接下来 $n \times n$ 矩阵 $p_{i,j}$（$0 \le v_i, p_{i,j} \le 10^5$）。
- 输出：最小总花费。
- 样例：$n = 4$，$v = [5, 4, 4, 3]$，电网矩阵 p 对角线外全为 $2, 3, 4 \implies$ 输出 9（在 4 号建站花 3，再连 $1, 2, 3$ 各花 2）。

本页要点

难度标签：**[基础]**。既有“点权”（建站）又有“边权”（拉线），不能直接套单棵生成树！

例 2 | 思路与建模：0 号虚拟源点把点权变成边权



- 引入 0 号虚拟节点代表“发电总站”，令 $w[0][i] = w[i][0] = v_i$。
- 建发电站 $\iff$ 选中边 $(0, i)$！只要把 $0 \sim n$ 共 $n+1$ 个点连成最小生成树，就自动同时选出了“建几个站”和“怎么拉网”。

本页要点

超级源点一加，直接变成“$n+1$ 个点的邻接矩阵求 MST”，继续用 $O(n^2)$ Prim 解决！

例 2 | 样例手推跟踪表（0～4 号点共 5 轮 Prim）

step	选出 u	low[u]	ans	low[1]	low[2]	low[3]	low[4]
0	0	0	0	5	4	4	3
1	4	3	3	$\min(5,2)=2$	$\min(4,3)=3$	$\min(4,4)=4$	已入树
2	1	2	5	已入树	$\min(3,2)=2$	$\min(4,2)=2$	已入树
3	2	2	7	已入树	已入树	$\min(2,3)=2$	已入树
4	3	2	9	已入树	已入树	已入树	已入树

本页要点

手推验证：选出 $(0,4)=3,(4,1)=2,(1,2)=2,(1,3)=2$，总花费 $3+2+2+2=9$！

例 2 | 核心步骤表与模板改动

按顺序计算	纸上得到	核对问题
1. 开 $(n+1)\times(n+1)$ 矩阵	下标 $0\sim n$ 的费用矩阵	别忘了第 0 行和第 0 列都要赋值！
2. 填 $w[0][i]=w[i][0]=v_i$	0 号点与每口矿井相连	矩阵 p 是对称填入 $1\sim n$ 吗？
3. 从 0 号点跑 $n+1$ 轮 Prim	依次拉入剩余 n 口矿井	循环条件是 `step <= n` 还是 `< n`？

本页要点

相对 **Prim** 模板改了哪里？点数从 n 变成 $n+1$（编号 $0\sim n$），循环边界全部改为 `<= n`。

例 2 | 核心代码（源码 code/prim_new_start.cpp）

```
vector<vector<long long>> w(n + 1, vector<long long>(n + 1, 0));
for (int i = 1; i <= n; ++i) { long long v; cin >> v; w[0][i] = w[i][0] = v; }
for (int i = 1; i <= n; ++i) for (int j = 1; j <= n; ++j) cin >> w[i][j];
vector<long long> low(n + 1, LLONG_MAX / 4); vector<char> vis(n + 1, 0);
low[0] = 0; long long ans = 0;
for (int step = 0; step <= n; ++step) {      // 共 n+1 个点（含 0 号虚拟源点）
    int u = -1;
    for (int v = 0; v <= n; ++v) if (!vis[v] && (u == -1 || low[v] < low[u])) u
 = v;
    vis[u] = 1; ans += low[u];
    for (int v = 0; v <= n; ++v) if (!vis[v]) low[v] = min(low[v], w[u][v]);
}
cout << ans << '\n';
```

本页要点

建站费 v_i 很便宜、拉线很贵时，Prim 会自动选多条 $(0, i)$ 边，相当于建多个发电站！

例 2 | 易错与 3 组自测边界

- 易错 **1**（循环少跑一轮）：因为加了 0 号虚拟点，共有 $n+1$ 个点，循环写成 `step < n` 会漏掉最后一口矿井（WA）。
- 易错 **2**（矩阵少赋对称边）：只写了 `w[0][i] = v` 忘了 `w[i][0] = v`，若以后从非 0 点起步就会出错。
- **3** 组自测边界：
  1. 样例 $n = 4 \implies$ 输出 9；
  2. 最小边界 $n = 1, v_1 = 7, p = [[0]] \implies$ 输出 7；
  3. 拉线极贵、建站极便宜（$v_i = 1, p_{i,j} = 100$） $\implies$ 各自建站，输出 n。

本页要点

记住：引入 0 号虚拟源点后，全图总点数变成 $n+1$，所有循环上界同步改成 `<= n`！

例 3 | [提高] 北极通讯网络： 先读懂要求

大白话题意

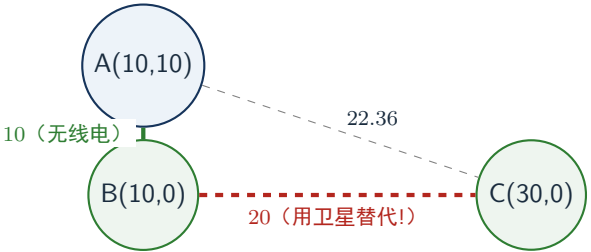
北极有 n（$1 \le n \le 500$）座村庄，已知坐标 (x_i, y_i)。现有 k（$0 \le k \le 100$）台卫星设备，拥有卫星的两村无论多远都能直接通信；其余村庄靠无线电通信（距离 $\le d$ 才能连）。求让所有村庄都连通所需的最小无线电距离 d（保留 2 位小数）。

- 输入：第 1 行 n, k；接下来 n 行每行两个整数 x_i, y_i。
- 输出：最小的实数 d，保留两位小数。
- 样例：$n = 3, k = 2$，坐标 $(10,10), (10,0), (30,0) \implies$ 输出 `10.00`。

本页要点

难度标签：**[提高]**。这是一道经典的“瓶颈生成树 + 留 k 个连通块”建模题！

例 3 | 思路与建模：k 台卫星 $\Longleftrightarrow$ 允许剩 k 个连通块



- k 台卫星可以把任意 k 个连通块“在天上连成一块”，所以地面上只需用无线电连成 $\max(1,k)$ 个连通块即可！
- 由瓶颈生成树结论，Kruskal 按边长从小到大合并，连通块数降到 $\max(1,k)$ 时停手，最后连的那条边就是最小 d！

本页要点

注意边界：当 $k=0$ 或 $k=1$ 时，卫星帮不上忙（1 台卫星不能连两村），地面必须连成 1 个连通块！

例 3 | 样例手推跟踪表（3 座村庄坐标推导 10.00）

步骤	枚举边 / 动作	边权（欧氏距离）	块数	ans
1. 建边	$(A,B),(B,C),(A,C)$	$10.00, 20.00, 22.36$	3	0.00
2. 排序	升序排列	$10.00 < 20.00 < 22.36$	3	0.00
3. 选边 1	(A,B) 不同块，合并	10.00	$3 \to 2$	10.00
4. 终止	`blocks` $== \max(1,k) = 2$	**break!**	2	10.00

本页要点

刚合并第一条边 (A,B) 时连通块数就从 3 降到了 $k=2$，于是给 B,C 配卫星，答案直接定格在 `10.00`！

例 3 | 核心步骤表与模板改动

按顺序计算	纸上得到	核对问题
1. 特判 $k \geq n$	直接输出 `0.00`	每村一台卫星，还要用无线电吗？
2. 枚举点对建 $n(n-1)/2$ 边	存成边权为 `double` 的边表	距离公式 `hypot(dx, dy)` 对吗？
3. 排序并设 `target=max(1,k)`	目标剩 `target` 个块	$k = 0$ 时 `target` 是 1 吗？
4. Kruskal 合并到 `blocks==target`	记录最后一条边权 `ans`	输出保留 2 位小数了吗？

本页要点

相对 **Kruskal** 模板改了哪里？ 边权改为 `double`； 维护 `blocks` 从 n 减到 `max(1, k)` 即 `break`。

例 3 | 核心代码（1/2）：特判与全图建边排序

```
int n, k; if (!(cin >> n >> k)) return 0;
vector<double> x(n), y(n);
for (int i = 0; i < n; ++i) cin >> x[i] >> y[i];
if (k >= n) { cout << "0.00\n"; return 0; }  // 每座村都能分到卫星，无需无线电

vector<Edge> es; es.reserve(n * (n - 1) / 2);
for (int i = 0; i < n; ++i)
    for (int j = i + 1; j < n; ++j)
        es.push_back({i, j, hypot(x[i] - x[j], y[i] - y[j])});
sort(es.begin(), es.end(), [](const Edge &a, const Edge &b) { return a.w < b.w;
 });
```

本页要点

用 `es.reserve(n*(n-1)/2)` 提前预留空间，避免 12.5 万条边多次扩容！

例 3 | 核心代码（2/2）（code/kruskal_arctic.cpp）

```
vector<int> p(n); iota(p.begin(), p.end(), 0);
auto find = [&](auto &&self, int a) -> int { return p[a] == a ? a : (p[a] =
self(self, p[a])); };
int blocks = n, target = max(1, k); double ans = 0.0;
for (auto &[u, v, w] : es) {
    int ru = find(find, u), rv = find(find, v);
    if (ru != rv) {
        p[rv] = ru; --blocks; ans = w;
        if (blocks == target) break;        // 连通块降到 target 个即停
    }
}
cout << fixed << setprecision(2) << ans << '\n';
```

本页要点

注意 $n = 1, k = 0$ 时 $k < n$ 且 `blocks=1 == target`，循环不执行直接输出 `0.00`，完全正确！

例 3 | 易错与 3 组自测边界

- 易错 **1**（忘处理 $k = 0$）：若直接写 `if (blocks == k) break;`，当 $k = 0$ 时 `blocks` 永远降不到 0，会错输出最大边（WA）。
- 易错 **2**（精度或格式）：输出必须用 `fixed << setprecision(2)` 严格保留两位小数。
- **3** 组自测边界：
  1. 样例 $n = 3, k = 2 \implies$ 输出 `10.00`；
  2. $k \geq n$（如 $n = 3, k = 3$）$\implies$ 直接输出 `0.00`；
  3. $k = 0, n = 3$（样例坐标）$\implies$ 必须连成 1 块，输出 `20.00`。

本页要点

记住公式：地面上要留下的连通块数量 `target = max(1, k)`！

例 4 | [提高] 秘密的牛奶运输： 先读懂要求

大白话题意

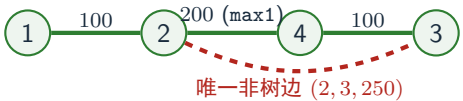
给一张 N（$1 \le N \le 2000$）个点、M（$1 \le M \le 20000$）条无向边的图（可能含重边），边权 $z \le 10^9$。要求找出总权值严格第二小的生成树（即严格大于 MST 权值的最小生成树）。

- 输入：第 1 行 N, M；接下来 M 行 x, y, z。
- 输出：仅一行，严格次小生成树的总边权。
- 样例：$N = 4, M = 4$，边 $(1, 2, 100), (2, 4, 200), (2, 3, 250), (3, 4, 100) \implies$ MST 为 400，输出 450。

本页要点

难度标签：**[提高]**。核心考点是“非树边替换树上路径最大边（或严格次大边）”！

例 4 | 思路与建模：建树 + 预处理路径 max1/max2



- 第 1 步：跑 Kruskal 求出 MST（权 M_0），并把选中的 $N-1$ 条边存成邻接表 `tree`，同时给每条边打上 `used` 标记。
- 第 2 步：本题 $N \le 2000$，直接从每个点 $s = 1 \sim N$ 跑一遍 DFS，在 $O(N^2)$ 时间内算出任意两点 (s, v) 在树上路径的最大边 `max1[s][v]` 和严格次大边 `max2[s][v]`。
- 第 3 步：枚举每条未选边 (u, v, w)，若 $w >$ `max1[u][v]` 就替 `max1`，若 $w ==$ `max1[u][v]` 就替 `max2`！

本页要点

利用 $N \le 2000$ 选 $O(N^2)$ 全源 DFS，比写 $O(M \log N)$ 树上倍增简单得多、还不容易写错！

例 4 | 样例手推跟踪表

阶段	处理对象	关键计算结果	备注
Kruskal	排 $100, 100, 200, 250$	选 $(1,2),(3,4),(2,4)$, $M_0 = 400$	剩非树边 $(2,3,250)$
路径查询	树上 $2 \to 4 \to 3$	权 $\{200, 100\} \Longrightarrow$ `max1` $= 200$, `max2` $= 100$	$250 > 200$ 严格更大
替换更新	非树边 $w = 250$	$400 - 200 + 250 = 450$	最终答案 `450`

本页要点

如果非树边恰好也是 200，因 $w ==$ `max1`，代码会改用 `max2` $= 100$ 替换，算出 $400 - 100 + 200 = 500$！

例 4 | 核心步骤表与模板改动

按顺序计算	纸上得到	核对问题
1. Kruskal 建树标 used	mst 与树边邻接表 tree	重边是否会误标成全 used？不会，按边下标标！
2. 枚举起点 s 跑 DFS	max1[s][u]，max2[s][u]	更新次大边时判断 w < nm1 了吗？
3. 枚举 !e.used 尝试替换	取最小的严格次小总权 ans	记得检查 m1 != -1 和 m2 != -1！

本页要点

相对 **Kruskal** 模板改了哪里？给 Edge 增加 bool used，选中时同步加进树邻接表 tree，再多两段 DFS 与枚举。

例 4 | 核心代码（1/2）：Kruskal 建树与路径预处理

```
vector<vector<pair<int, long long>>> tree(n + 1);
long long mst = 0;
for (auto &e : es) {
    int ru = find(find, e.u), rv = find(find, e.v);
    if (ru != rv) {
        p[rv] = ru; mst += e.w; e.used = true;
        tree[e.u].push_back({e.v, e.w}); tree[e.v].push_back({e.u, e.w});
    }
}
vector<vector<long long>> max1(n + 1, vector<long long>(n + 1, -1));
vector<vector<long long>> max2(n + 1, vector<long long>(n + 1, -1));
```

本页要点

用 e.used 标记每条具体边，即使输入有重边，没被选上的那条重边依然能正常参与后面的替换！

例 4 | 核心代码（2/3）：全源 DFS 预处理 max1，max2

```
for (int s = 1; s <= n; ++s) {
    auto dfs = [&](auto &&self, int u, int fa, long long m1, long long m2) {
        max1[s][u] = m1; max2[s][u] = m2;
        for (auto [v, w] : tree[u]) if (v != fa) {
            long long nm1 = m1, nm2 = m2;
            if (w > nm1) { nm2 = nm1; nm1 = w; }
            else if (w < nm1 && w > nm2) nm2 = w;
            self(self, v, u, nm1, nm2);
        }
    };
    dfs(dfs, s, 0, -1, -1);
}
```

本页要点

严格次大条件 `w < nm1 && w > nm2`：次大边必须严格小于最大边！

例 4 | 核心代码（3/3）：枚举非树边替换（code/second_mst.cpp）

```
long long ans = LLONG_MAX;
for (auto &e : es) if (!e.used) {
    long long m1 = max1[e.u][e.v], m2 = max2[e.u][e.v];
    if (e.w > m1 && m1 != -1) ans = min(ans, mst - m1 + e.w);
    else if (e.w == m1 && m2 != -1) ans = min(ans, mst - m2 + e.w);
}
cout << ans << '\n';
```

本页要点

当 e.w > m1 时替换 m1；当 e.w == m1 时只能退而替换严格次大边 m2！

例 4 | 易错与 3 组自测边界

- 易错 **1**（漏写 `w < nm1`）：如果让 `max2 == max1`，遇到与最大边等权的非树边就会算出等于 MST 的假答案（WA）。
- 易错 **2**（`ans` 开成 `int`）：$N \le 2000, z \le 10^9$，总权值可达 2×10^{12}，必须全程 `long long`。
- **3** 组自测边界：
  1. 样例 $N = 4, M = 4 \implies$ 输出 `450`；
  2. 有等权非树边（树边 $10, 20$，非树边 20）$\implies$ 跳过 `max1=20` 替 `max2=10`；
  3. 含重边（两点间有 $10, 10, 20$）$\implies$ 按边下标标 `used`，重边正常替换。

本页要点

注意 `max1` 和 `max2` 在 $N = 2000$ 时共占约 64 MB 内存，完全在题目限制之内！

最小生成树高频错误对照表

题目 / 场景	常见错误写法	正确做法与原因
未判不连通直接输出	`cnt < n - 1` 时仍印 `ans`	原图分成几块连不成树，必须检查选边数是否等于 `n - 1`
Prim 选点漏判 `!vis`	设 `low[0]=0` 后循环少写 `!vis[v]`	会重复挑选已经在树里的老点反复加自己导致 WA
总代价和开成 `int`	$m = 10^5$ 边权 10^9 用 `int ans`	必溢成负数，边权总和必须使用 `long long`
北极通讯网络边界	忽略 $k = 0$ 或 $k \geq n$	$k = 0$ 等价 $k = 1$ 留 1 块；$k \geq n$ 直接输出 `0.00`
严格次小生成树	没判 `w < nm1` 更新次大边	`max2` 变成和 `max1` 相等，替换后没严格变大

本页要点

“边表排序不漏边，合并成功计次数，总和必用 `long long`，严格次小记 `max2`”！

评测结果快速定位指南（WA / TLE / RE）

评测结果	优先排查方向
WA	1. ans 或单条边权是否忘了开 `long long`； 2. 图可能不连通时，是否漏了 `cnt < n - 1` 或 `low[u] >= INF` 判定； 3. Prim 内层更新是否误写成了 `low[u] + w[u][v]`。
TLE	1. 稠密完全图（$m \approx 10^6$）是否误用了 $O(m \log m)$ Kruskal 而不是 $O(n^2)$ Prim； 2. 并查集 `find` 是否忘了写路径压缩 `p[x] = self(self, p[x])`。
RE	1. 并查集或邻接矩阵是否开小了（1-indexed 需开 `n + 1`）； 2. 虚拟源点题循环有没有越界。

本页要点

遇到 WA 先查 `long long` 和 `cnt < n-1`；遇到 TLE 先看图密度选型和路径压缩！

提交前检查清单（六必查）

1. 选用 Kruskal 时，边结构体数组是否按边权 `w` 严格升序排列，并查集是否初始化了 `p[i]=i`？
2. 合并前有没有调 `find(u)` 和 `find(v)` 检查是否已同组（仅当不同组才合并、累加并 `++cnt`）？
3. 当可能存在不连通情况时，最后有没有判断 `cnt < n - 1`（或当 $n = 1$ 时 $cnt = 0 == n - 1$）？
4. 最小生成树的总边权和（以及所有边权比较/替换）是否开了 `long long`？
5. 稠密矩阵图跑朴素 Prim 时，是否给 `low[0]` 置 0，并在寻找最小距离点和更新邻居时都加上 `!vis[v]`？
6. 遇到点权建站题，是否引入了 0 号虚拟源点；遇到严格次小，是否同时维护了 `max1` 和 `max2`？

本页要点

交题前自测两组极限：一组 $n = 1$（0 条边），一组明显断成两块的图，验证输出后再提交！

本章全部练习与完成标准

1. 基础模板：Kruskal 模板（`code/kruskal_mst.cpp`）、最短网络 agrinet（一本通 1350，`code/prim_agrinet.cpp`）、最优布线问题（1349）。
2. 进阶模型：新的开始（虚拟源点，`code/prim_new_start.cpp`）、北极通讯网络（1487，`code/kruskal_arctic.cpp`）、秘密的牛奶运输（1490，`code/second_mst.cpp`）、黑暗城堡（1486）。

完成标准

能闭眼默写 15 行 Kruskal 和 $O(n^2)$ Prim 两种实现；能说出为什么有虚拟源点就能解点权选址题、为什么严格次小要记 `max2`；独立 AC 上述精选习题。

课后安排与下一章预告

- 复习：合上课件，独立用口述讲清“切分定理为什么让贪心挑最小边不会错”。
- 练习：用 Kruskal 和 Prim 两种方式同时 AC《最短网络》，体会各自优势，再挑战《北极通讯网络》与《秘密的牛奶运输》。
- 下一章预告：第 5 章《强连通分量与割点桥》——有向图里互相都能走到的一群点怎么缩成一个“超级大点”变成无环图？无向图里剪断哪座桥就会立刻断成两半？看 Tarjan 一遍 DFS 全部找出！

本章一句话

稀疏图按边从小到大用并查集接树，稠密图贴着距离数组拉点扩树，遇到点权加虚点、瓶颈停在 k 块、次小记好 `max2`。

谢谢！