最小生成树 Minimum Spanning Tree

庄博伟

ZUCC ACM Group

January 25th 2022

目录

最小生成树问题

给一张 N 点 M 边的无向有边权连通图，要求保留 N-1 条边，删去其余边，使得该图为树。问怎么选取能使得这 N-1 条边的边权和最小。

最小生成树

这样的 N-1 条边和 N 个点构成的树叫这张图的最小生成树，一张图的最小生成树可能有复数个

目录

算法过程

- 维护一个答案图，初始时 N 点 0 边，即一条边都不选，所有点均不连通

算法过程

- 维护一个答案图，初始时 N 点 0 边，即一条边都不选，所有点均不连通
- 将所有边按照边权从小到大排序，按照这个顺序枚举每条边尝试加入答案图中

算法过程

- 维护一个答案图，初始时 N 点 0 边，即一条边都不选，所有点均不连通
- 将所有边按照边权从小到大排序，按照这个顺序枚举每条边尝试加入答案图中
- 枚举一条边 (u, v)，如果 u 和 v 已经连通，说明这条边多余，扔掉，否则将这条边加入答案图

算法过程

- 维护一个答案图，初始时 N 点 0 边，即一条边都不选，所有点均不连通
- 将所有边按照边权从小到大排序，按照这个顺序枚举每条边尝试加入答案图中
- 枚举一条边 (u, v)，如果 u 和 v 已经连通，说明这条边多余，扔掉，否则将这条边加入答案图
- 最终答案图上的 N-1 条边就是边权和最小的方案

算法过程

- 维护一个答案图，初始时 N 点 0 边，即一条边都不选，所有点均不连通
- 将所有边按照边权从小到大排序，按照这个顺序枚举每条边尝试加入答案图中
- 枚举一条边 (u, v)，如果 u 和 v 已经连通，说明这条边多余，扔掉，否则将这条边加入答案图
- 最终答案图上的 N-1 条边就是边权和最小的方案

维护点之间是否连通需要使用并查集

代码示例

```cpp
int fa[N];
int find(int x) {
  return x == fa[x] ? x : fa[x] = find(fa[x]);
}
bool merge(int x, int y) {  // 有效合并返回true，无效返回false
  int fx = find(x);
  int fy = find(y);
  if (fx != fy) {
    fa[fy] = fx;
    return true;
  }
  return false;
}
struct Edge {
  int u, v, w;
  bool operator < (const Edge &tmp) const {
    return w < tmp.w;
  }
} e[M];
int MST(int n, int m) {
  int ans = 0;
  sort(e, e + m);
  for (int i = 0; i < n; i++) fa[i] = i; // 并查集初始化
  for (int i = 0; i < m; i++) {
    int u = e[i].u;
    int v = e[i].v;
    if (merge(u, v)) ans += e[i].w;
  }
  return ans;
}
```

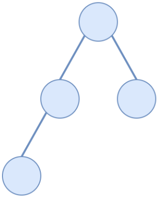
目录

算法证明

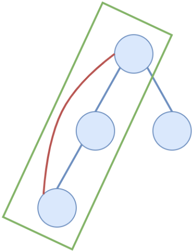
- 对于图中任意一个点 x，所有从 x 连出去的边，边权最小的那一条一定存在于 MST 上

算法证明

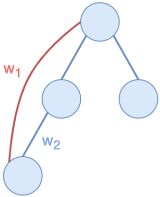
- 对于图中任意一个点 x，所有从 x 连出去的边，边权最小的那一条一定存在于 MST 上
- 反证：如果该条边不存在于 MST 上，那么在答案 MST 上加上这条边，一定会形成一个环，删除环上任意一条边，则又能将答案图变回一颗树
- 环上必定有两条和 x 相连的边，那么保留 x 连出去边权最小的那一条一定更优



原本的MST



新加一条边，形成一个环
删除环上任意一条边可以让其变回树



$w_1 \leq w_2$，因此删w_2一定更好

算法证明

前置结论

对于图中任意一个点 x，所有从 x 连出去的边，边权最小的那一条一定存在于 MST 上

算法证明

前置结论

对于图中任意一个点 x，所有从 x 连出去的边，边权最小的那一条一定存在于 MST 上

- 刚开始一条边都不选的时候，选一条全局最短的边，它肯定是某个点相连的边权最小的边，直接选

算法证明

前置结论

对于图中任意一个点 x，所有从 x 连出去的边，边权最小的那一条一定存在于 MST 上

- 刚开始一条边都不选的时候，选一条全局最短的边，它肯定是某个点相连的边权最小的边，直接选
- 在当前已经选了一些边的情况下，将每个连通分量看成一个点（缩点思想）
- 此时，选择边权最小的边，只要它不是多余的边，则直接选

算法证明

前置结论

对于图中任意一个点 x，所有从 x 连出去的边，边权最小的那一条一定存在于 MST 上

- 刚开始一条边都不选的时候，选一条全局最短的边，它肯定是某个点相连的边权最小的边，直接选
- 在当前已经选了一些边的情况下，将每个连通分量看成一个点（缩点思想）
- 此时，选择边权最小的边，只要它不是多余的边，则直接选

从这个角度出发，Kruskal 其实就是每次挑选边权最小的、非多余的边，挑出 N-1 条来就是答案

目录

MST 之间的转换

- 定义一次操作为：在树上添加一条边，然后在产生的环上删除一条边
- 若树 A 和树 B 是无向图的 2 个 MST，则 A 和 B 之间可以通过若干次操作互相转换

MST 之间的转换

- 定义一次操作为：在树上添加一条边，然后在产生的环上删除一条边
- 若树 A 和树 B 是无向图的 2 个 MST，则 A 和 B 之间可以通过若干次操作互相转换

多个 MST

有时候一次操作替换的两条边的边权是相同的，这也是一张图有多个 MST 的原因

MST 的边权分布

- 对于同一张图，它的所有 MST 的边权分布固定。即所有 MST 的 N-1 条边按照边权升序排好序，得到的边权序列相同
- 通俗点说就是不存在 $1+3=2+2$ 的多个 MST

MST 的边权分布

- 对于同一张图，它的所有 MST 的边权分布固定。即所有 MST 的 N-1 条边按照边权升序排好序，得到的边权序列相同
- 通俗点说就是不存在 $1+3=2+2$ 的多个 MST

推论

这个性质说明：如果无向图的所有边权均不相同，则其 MST 是唯一的

不过这个结论的逆命题并不成立，MST 唯一的无向图是可能有相同边权的

END