图论杂项

黄嘉盛

2022 年 7 月 14 日

1. Boruvka 算法
2. 斯坦纳树

生成树算法

三种不同的最小生成树（MST）算法：

- Kruskal 算法，从小到大加边
- Prim 算法，从当前构造的树向外扩展
- Boruvka 算法，前两者的结合体

Boruvka 算法

简要的概括 Boruvka 算法，就是“从所有当前的连通块向其他连通块扩展出最小边，直到只剩一个连通块”。

具体来说，就是当连通块个数大于 1 时，首先对每个连通块 i，计算出 mn_i 表示它与其他连通块之间的最小边编号。然后依次扫每个连通块 i，如果 mn_i 仍然连接两个不同的连通块，就把他们合并，并且将 mn_i 的边权加入答案。

Boruvka 算法

由于执行完一轮算法后，每个连通块大小至少为 2（设原来大小均为 1），所以连通块个数至少变成了原来的 $\frac{1}{2}$，设每轮求所有 mn_i 的复杂度为 $O(T)$，则算法总复杂度 $O(T\log n)$

值得注意的是，由于朴素的扫一遍所有边求 mn_i 的方式复杂度常常是 $O(M)$ 的，所以有的题目可能会让 $O(M)$ 变得较大（如给一个完全图，两点之间的边由某种公式计算），这种情况不优化求 mn 方式无法通过

CF 888G XOR MST

有 n 个点构成一个完全图, 每个点的点权为 a_i, 连接 i,j 的边的权值为 $a_i \oplus a_j$ (异或) 。求最小生成树。

$1 \leq n \leq 2 \times 10^5, 0 \leq a_i \leq 2^{30}$

CF 888G XOR MST

如之前所说的，该题的难点也在于求 mn_i，即对一个连通块 S，有 $mn_S = min\{a_i \oplus a_j | i \in S, j \notin S\}$

一般来说，求异或最小值会使用线形基，但是在合并连通块是要将他们之间的边从线形基中删除，这是比较麻烦的，所以使用线形基解决这题并不好

CF 888G XOR MST

除开线形基还可以考虑 Trie 树，于是考虑先建出所有 a_i 的 Trie 树。
当我们询问连通块 S 的 mn_S 时，将 $a_i(i \in S)$ 在字典树中删掉，然后对于每个 $a_i(i \in S)$ 在字典树里贪心地求一个异或最小值，更新 mn_S 。
计算完 mn_S 后，再把 $a_i(i \in S)$ 重新插入字典树。
一次字典树的操作是 $O(\log a_i)$ 的，每次求 mn 时会遍历所有 n 个点一次（因为每个点都属于恰好一个连通块）。因此求一次 mn 是 $O(n \log_2 a_i)$ 的。套上 Boruvka 的复杂度，总体复杂度就是 $O(n \log n \log a_i)$ 。

1. Boruvka 算法
2. 斯坦纳树

斯坦纳树

对于图 $G = (V, E)$, 存在子集 $K \subset V, K$ 中的点称为关键点。斯坦纳树表示总边权最小的树, 仅包含来自 $G(V, E)$ 中的顶点和边, 并包含所有关键点。

如果 $V = K$, 那么问题就变成了熟悉的最小生成树问题, 这时候可以采用 kruskal 或 prim 算法来解决, 时间复杂度为 $O(E \log_2 V)$

如何计算斯坦纳树是一个 NP 问题, 即你无法在多项式时间内求解。但是如果已知 K 非常小, 我们可以在关联于 3^K 的一个时间复杂度内利用动态规划解决这个问题。

斯坦纳树

首先，我们需要意识到斯坦纳树，也是一株树，树必然有根，我们可以枚举根来进行计算。并且由于我们仅关心关键点是否包含在树中，因此我们对关键点进行状态压缩，共 2^K 种状态，我们可以利用二进制来表示。

我们记 $dp(i,s)$ 表示以 i 为根，i 可以是普通点也可以是关键点，满足 s 代表状态的子树。我们记结点 u 的掩码为 $u.mask$，如果 u 是关键点，则 $u.mask$ 表示 u 在二进制中对应的比特位，否则 $u.\mathrm{mask}$ 为 0 。

斯坦纳树

考虑到对于某一株树的根结点，根结点下有三种情况：

- 根结点没有子结点
- 根结点有一个子结点
- 根结点有多个子结点

因此我们也可以根据三种情况建立状态转移公式。

- $dp(i, s) = (s == i.mask)?0 : \inf$
- $dp(i, s \mid i.mask) = \min(dp(i, s \mid i.mask), dp(j, s) + (i, j).w)$
- $dp(i, s) = \min\left(dp(i, s), dp\left(i, s' \mid i.mask\right) + dp\left(i, \left(s - s'\right) \mid i.mask\right)\right)$

其中 $(i, j).w$ 表示从 i 到 j 边的权重。

斯坦纳树复杂度分析

总共有 $V \cdot 2^K$ 个 dp 状态，而每个状态根据状态转移公式，时间复杂度如下：

- $O(1)$
- 可以通过在拥有相同状态的 dp 点上跑 SPFA。时间复杂度摊还为 $O(E)$，如果边权非负，你还可以利用 Dijkstra 来将摊还时间复杂度降低到 $O(V)$ 。
- s 的子集数目为 $2^{B(s)}$，其中 $B(s)$ 表示 s 中 1 的数目。由于 $\sum_{i=0}^{K}\binom{K}{i}2^i = (1+2)^K$，因此摊还时间复杂度为 $O\left(1.5^K\right)$ 。

因此对于任意一个 dp 状态所需的时间复杂度为 $O\left(V + 1.5^K\right)$，总共有 $V \cdot 2^K$ 个 dp 状态，因此总的时间复杂度为 $O\left(V\left(V \cdot 2^K + 3^K\right)\right)$