树上数据结构选讲

Part I

wYYSZL

2026.8.8

本节课内容

1. 静态树分治

2. 动态树分治

目录

1. 静态树分治
2. 动态树分治

算法回顾 - 点分治

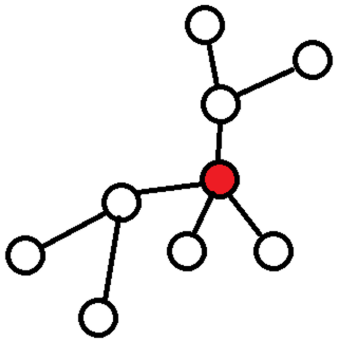

图: 点分治

序列上的分治是每次找一个中点，然后统计经过中点的区间，然后递归下去计算。

树的点分治每次找一个点，然后统计经过重心的路径，然后把这个点删去，树变成了很多连通子图，递归下去计算。这个点我们通常找重心。

算法回顾 - 点分治

重心有三个定义：

- 到每个点的距离之和最小
- 删去这个点之后最大的连通子图大小最小
- 删去这个点之后最大的连通子图大小不过半

在边权为 1 的情况下，三个定义是等价的（证明比较复杂，此处略去），我们考虑第三个定义，于是我们每次递归下去的连通子图大小至少减半，所以递归树的深度 $\mathcal{O}(\log n)$，所以点分治复杂度 $\mathcal{O}(n \log n)$。

算法回顾 - 边分治

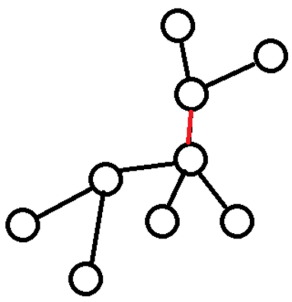

图: 边分治

我们再来看边分治。

树的边分治每次找一条边，然后统计经过这条边的路径，然后把这条边删去，树变成了两个连通子图，递归下去计算。

分治一般是想让分治出的每个子问题大小接近，所以我们尽可能让两边大小接近。

实际上树的边分治更好地对应到了序列的分治。

但存在一些问题。

算法回顾 - 边分治



图: 菊花图

考察菊花图。此时我们发现无论找哪个边来进行分治，都有一个子问题大小是 1。这样，复杂度就是

$$n + (n-1) + (n-2) + ... + 1 = \mathcal{O}(n^2),$$

不能接受。

算法回顾 - 边分治

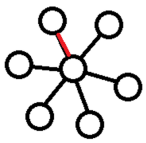

图: 菊花图

考察菊花图。此时我们发现无论找哪个边来进行分治，都有一个子问题大小是 1。这样，复杂度就是 $n + (n-1) + (n-2) + ... + 1 = \mathcal{O}(n^2)$，不能接受。

解决方法

考虑使用“三度化”。

直观来看，刚才的问题在于，菊花图有一个点的度数太大了，导致选取与之相连的边，即使选取得很良好，例如选了最大的子树，其他子树很小，由于其他子树很多，联合起来的点也会非常多。

算法回顾 - 边分治 - 三度化

因而，我们考虑加入虚点虚边，对于每个度数 > 3 的节点，让该节点度数小于等于 3。主要有两种方法，具体见图：

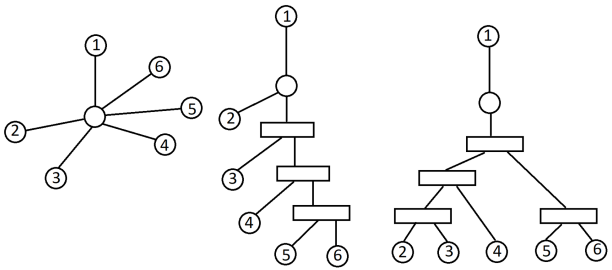

图: 三度化

算法回顾 - 边分治 - 三度化

三度化后再分治的复杂度的正确性依赖这个定理：

引理

N $(N \geq 2)$ 个节点的树，最大度数为 $D \geq 2$ 。一定存在一条边，使得将这条边从树中删除后，所得到的两棵子树 T_1 和 T_2 的节点数量 $|T_1|$ 和 $|T_2|$ 满足以下不等式：

$$\left\lceil \frac{N-1}{D} \right\rceil \leq |T_1|, |T_2| \leq N - \left\lceil \frac{N-1}{D} \right\rceil$$

带入 $D = 3$，发现每次切分最大的一个连通块大小不大于 $2n/3 + 1$（加一处理上取整），所以层数是 $\mathcal{O}(\log n)$ 的。

算法回顾 - 边分治 -三度化

上面的引理，采用反证法证明：

令 $k = \lceil \frac{N-1}{D} \rceil$。我们只需证明存在边 e 使得删去 e 后的两部分节点数均 $\geq k$ 即可。

假设对任意边，都没有两部分均 $\geq k$ 的情形。由于两部分之和为 N，且易证不可能两部分均 $< k$，故对每条边，必有一部分 $< k$，另一部分 $> N - k$。

为每条边定向：从节点数 $< k$ 的部分指向 $> N - k$ 的部分。因树无环，该定向必有至少一个汇点 v，其所有关联边均指向 v。这意味着对 v 的每个邻居 u，删去边 uv 后不含 v 的分支大小均 $< k$（即 $\leq k-1$）。设 v 的度数为 d（$d \leq D$），则

$$N \leq 1 + d(k-1) \leq 1 + D(k-1).$$

但由 $k = \lceil \frac{N-1}{D} \rceil$ 知 $k - 1 < \frac{N-1}{D}$，即 $D(k-1) < N-1$，从而 $N > 1 + D(k-1)$，矛盾。故假设不成立，必存在满足条件的边。□

例题 A

A - 【模板】点分治

给定一棵有 n 个点的树，边有边权，询问 m 次树上距离为 k 的点对是否存在。$n \leq 10^4, m \leq 100, k \leq 10^7$。

例题 A 题解

对树进行点分治，每次统计经过分治中心的路径，即两端点在中心的不同子树内的路径。

维护一个数组，记录当前已合并的子树中各点到分治中心的距离。

处理一棵子树时，对其中每个点 x（到中心距离为 d），先查询数组中有没有 $k-d$，若有则存在一条长为 k 的路径；处理完整棵子树后，再把这些距离全部插入数组。

先处理后插入，保证配对的点来自不同子树，且每条路径恰被统计一次。

总时间复杂度 $\mathcal{O}(nm\log n)$。

例题 B

B - 消息传递

给出一棵边权为 1 的树，多次查询距离 x 恰为 k 的点数。

$n, m \leq 10^5$

例题 B 题解

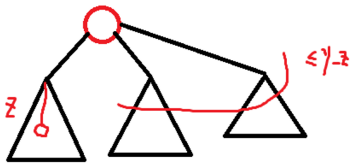

图: 例题 B 示意图

考虑查询的 x，在当前分治子树 a 中，x 到分治中心距离是 z。点分治只需考虑其他分治子树中的点，其他分治子树中的点中距离 x 为 k 的点，满足距离分治中心为 $k-z$。

考虑对每个点挂询问，然后点分治到这个点的时候处理一下这个点上所有询问。每层预处理前缀和可以 $\mathcal{O}(1)$ 查询，每个询问都是挂在点上的所以不会超过 $\mathcal{O}(\log n)$ 次，时间复杂度 $O((n+m)\log n)$。

* 例题 C

*C - 通道

给定包含 n 个节点的 3 棵树 T_1, T_2, T_3，边有边权，请你求出

$$\max_{(x,y)}\{dis_1(x,y) + dis_2(x,y) + dis_3(x,y)\}$$

$n \leq 10^5, w \leq 10^{12}$

* 例题 C 题解

重工业。

类似高维偏序，我们一层一层固定一棵树的限制。先考虑对 T_1 进行边分治，这样我们只需要考虑跨过边的答案。设这条边为 (a, b)，删去后形成的两个连通块内的与 a, b 在同一个连通块的点分别为“黑点”和“白点”，（更高层分出去的点无关，称为“灰点”）问题转化为最大化：

$$dis_1(x, a) + dis_1(b, y) + dis_2(x, y) + dis_3(x, y)$$

且 x 为黑点，y 为白点。

* 例题 C 题解

之后，考虑枚举 x, y 在 T_2 的 LCA。我们显然不能暴力枚举 x, y 来找到 LCA，但是可以考虑建立包含所有“白点”和“黑点”的虚树。设白点黑点的点数和为 $|s|$，不难发现 LCA 的个数也是 $\mathcal{O}(|s|)$ 级别的。

设枚举的 LCA 为 p，这样，问题就又转为最大化：

$$dis_1(x,a) + dis_1(b,y) + deep_2(x) + deep_2(y) - 2deep_2(p) + dis_3(x,y)$$

再设 $w(x) = dis_1(x, a/b) + deep_2(x) + deep_3(x)$，于是问题继续变为最大化：

$$w(x) + w(y) - 2deep_3(LCA_3(x,y))$$

且 x 为黑点，y 为白点，且 x, y 在 T_2 中在 p 的不同子树中。

* 例题 C 题解

注意到，这个式子可以看作一个广义的距离，又由于一个性质：

性质

对于一棵边权全部是正的树，设这棵树的互不相交的两个点集 A, B，点集 A 中最长路的端点为 u, v，点集 B 的最长路端点分别是 a, b，那么跨越点集 A 和 B 的最长路端点只可能是 $(u, a), (v, b), (u, b), (v, a)$ 四种情况之一。

* 例题 C 题解

我们考虑这样做：对 T_2 进行树上 DP，设 $f_{u,1/0}$ 为 u 子树内黑/白点内部的最长距离对应的点。转移显然，考虑如何统计答案。转移时，设 x 为父亲，y 为儿子，求解目前 f_x 中黑点和 f_y 中白点的最长广义距离（四个点对之一），然后反过来求（f_x 中白点和 f_y 中黑点），并且此时 LCA 为 x。

这样，我们就在 $\mathcal{O}(n\log^2 n)$ 的时间内解决了这道题。听起来就非常重工业。

点分治 - 边分治

从刚才那道题我们就能看出来边分治的优势。

点分治会将树分为多个集合。而一般合并的时候是两两合并，这就导致一次点分治合并的时候需要多次两两合并。如果能够保证一次合并的复杂度是依赖于合并进来的块（的大小/信息）的，就能保证总的复杂度不退化；但如果是像上一道题，复杂度依赖于合并后的大块（的大小/信息），多次合并将导致大块被反复遍历，使得复杂度退化。

而边分治就不会存在问题，边分治总是将树分治为两个部分，只需要一次两两合并。

当然，边分治会比较难写，尤其是如何保证虚拟点/边不参与答案计算。

目录

1. 静态树分治
2. 动态树分治

算法回顾 - 点分树

点分树是动态的点分治，用于处理信息改变（但树的形态不会改变）的问题。
例如这样的问题：

D - 【模板】点分树

给定一棵 n 个节点的树，边权均为 1，每个点 u 有权值 w_u。
需要支持 m 次操作，修改点权，或查询与 x 的距离 $\leq k$ 的点权和。
$n, m \leq 10^5$

解决方法是将我们每次取重心的过程建立成一棵树，那么这棵树的树高不会超过 $\mathcal{O}(\log n)$。换句话说，这棵树是一种重构树，它的树高比较优秀。

例题 D 题解

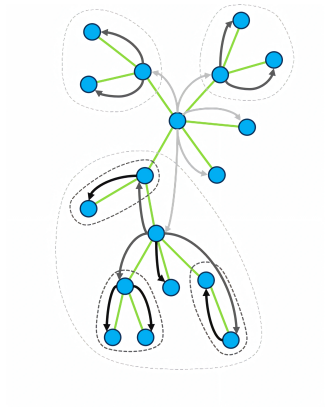

图: 点分树

具体地，我们将每次找到的重心与上一层的重心之间建立父子关系。每一个点都是重构树的一个节点，它在点分治过程中作为分治的重心时，该连通块包含的点就是它在重构树的子树的所有节点。因此，当修改点权/统计答案的时候，我们要找到这个点在重构树中对应点，然后不断跳父亲节点修改标记/统计答案。比如这个题就是修改距离重心距离为 i 的点权和的前缀和数组。

同理

自然也有边分树，只是实在太少见也就不必讨论。

例题 E

E - 捉迷藏

给定 n 个点的树，点分黑白两类。

m 次询问，修改一个点的颜色，或者求所有黑点之间的最大距离。

$n \leq 10^5, m \leq 5 \times 10^5$。

例题 E 题解

建出点分树，每个节点用 multiset 存与子树内黑点的距离，并将两个黑点的距离相加求出穿过该节点的最长黑点距离，全局用一个 multiset 维护这个信息。
对于一个修改，除了改变每个节点的 multiset 还改变将改变全局的 multiset，从 x 向上跳点分树，类似于初始化时的修改即可。
时间复杂度 $O(n \log^2 n)$。

Hint

- 本题有复杂度更优秀的做法，具体可以看这道题的题解。
- 但显然点分树的做法更加易懂。但是 multiset 的常数很大，可以使用优先队列。考虑到有删除，可以使用各种可删堆，或者简单地使用垃圾堆的技巧，这些我们之后会提到。

例题 F

F - 随便口胡的题目

给定一棵树，给定常数 $w > 0$。任选一个节点作为中心，再任选若干个节点（可为空），定义收益为：w 乘以所选节点数的平方根，再减去所选节点到中心的距离之和。求所有选择中收益的最大值。

$n \leq 2 \times 10^5$

例题 F 题解

枚举中心点，然后考虑找到收益的最大值。

贪心地找，尽可能找距离近的点。设选取的集合为 S。当收益 $<$ 代价，即

$$w(\sqrt{|S|+1}-\sqrt{|S|})<dist$$

时，停止，这时就是收益最大值。

但是 $dist$ 不是定值，我们不太容易找到最优秀的 S 的大小。但是注意到 $\sqrt{|S|}$ 上凸，也就是说每新选一个点收益递减，距离（代价）递增，所以收益是单峰的，于是可以三分。

例题 F 题解

三分点数不易计算，可以三分选取的最后一个点的距离，然后计算强制最后一个点的距离是这个的最大收益。这时候 $dist$ 是定值，容易计算。不过我们需要知道 $< dist$ 和 $= dist$ 的点有多少个，点分树上求解。也可以三分之后查看第一个选这个距离和最后一个选这个距离的收益的正负情况，然后判断，但是这样完了也得计算收益，不如直接算。

这样，我们在 $\mathcal{O}(n \log^2 n)$ 的时间内解决问题。

如何计算最优的集合大小？

实际上就是解方程 $w(\sqrt{x+1} - \sqrt{x}) < dist$。直接解比较麻烦，有两种思路，第一种是对于每一个 $dist$，计算满足条件的最小 x。x 随 $dist$ 增加而减少，可以双指针。第二种思路是数学的思路，考虑展开一项，$(1+x)^{0.5} = (1+\frac{1}{x})^{0.5} * \sqrt{x} \approx \sqrt{x} + \frac{1}{2\sqrt{x}}$，于是原方程化为 $\frac{w}{2\sqrt{x}} < dist$，就比较容易。