树上问题选讲

crazy_sea

2025 年 5 月 1 日

一些约定

我们定义 fa_x 为 x 的父亲节点，siz_x 为 x 的大小，d_x 为 x 的深度。

非特殊说明下计数类问题模数为 998244353。

考虑如下问题：

问题

有一棵 n 个点的有根树，第 i 个点有一个重量为 1，权值为 a_i 的物品，定义一组选物品的方案的权值为所选物品的权值乘积，对于所有 $(x,i)(1\le x\le n,1\le i\le m)$，你需要求出 x 子树内所有满足选出物品重量之和为 i 的方案的权值之和。

$m\le n$。

显然大家都知道，直接暴力做复杂度就是 $O(nm)$ 的，我们考虑 $m < n$ 的时候的复杂度分析。

考虑合并的两棵子树分别为 x, y，简单分类讨论一下

- $siz_x, siz_y \leq m$ 根据之前的结论，这部分对总复杂度的贡献显然为 $O(nm)$。
- $siz_x \leq m, siz_y > m$，那么合并复杂度为 $siz_x \times m$，显然所有这样的 x 的 siz 之和 $< n$。
- $siz_x, siz_y > m$，这样的合并显然不会超过 $O(n/m)$ 次，所以对总复杂度的贡献也是 $O(nm)$。

这种分析方式有利于接下来的分析。

考虑如下问题：

问题

有一棵 n 个点的有根树，第 i 个点有一个重量为 b_i，权值为 a_i 的物品，定义一组选物品的方案的权值为所选物品的权值乘积，对于所有 $(x,i)(1\le x\le n,1\le i\le m)$，你需要求出 x 子树内所有满足选出物品重量之和为 i 的方案的权值之和。
不保证 $m\le n$。

一种显然的方法是 NTT，复杂度是 $O(nm\log m)$，尝试从其它的角度进行分析。

实际上，直接暴力做复杂度是 $O(\frac{nm^2}{\log m})$ 的。

具体来说，我们暴力的时候只对有值的位置进行转移，就可以得到上述复杂度的做法。

复杂度分析和之前类似。

实际上，我们还有 $O(nm\log n)$ 的做法。

具体来说，合并两个子树直接合并 dp 数组是有些浪费的，一种想法是把一棵子树里的多项式乘到另一棵子树的 dp 数组当中即可。

于是考虑启发式合并，复杂度就是 $O(nm\log n)$。

事实上刚才所说的“把一棵子树里的多项式乘到另一棵子树的 dp 数组中”这个乘的过程是并没有强调顺序，也就是说这里有很大的操作空间。

一个简答的例子是：比如说我们可以要求选出的物品必须处于同一个连通块。

这种做法还有一个名字，叫做：DFS 序上背包。

都说到这份上了，是不是该来个 $O(nm)$ 的做法了。

讲三种做法结合一下，如果合并的两个子树的 $siz \leq \log m/2$，则用算法一进行处理。

如果两个子树的 $siz > \log m/2$，那么采用 NTT 快速卷积。

如果两个子树有一个 $siz > \log m/2$，一个小于，则用合并的方法进行处理即可。

可以发现复杂度为 $O(nm)$。

例题

问题 (QOJ 7895)

给定一棵 n 个点的树和 k，问有多少种断边的方式使得树的每个连通块的大小都是 k 或 $k+1$。
对 998244353 取模。
$1 \le k < n \le 10^5$。

试试看最优能做到什么复杂度。

首先标算 $O(n^{1.5})$ 是显然的，对 k 根号分治，发现 dp 数组只有 $\min(k, siz/k+1)$ 个位置的值非 0，暴力做即可。

首先考虑一种 $O(n\log n)$ 的做法。

就是你发现如果合并过来的子树 $siz<k$，那么这个合并过程可以 $O(1)$ 处理，我们只需要考虑合并过来的子树 $siz\geq k$ 的情况。

使用 NTT 即可，这样的合并最多 $O(\frac{n}{k})$ 次，复杂度 $O(n\log n)$。

然后发现因为 x 位置 dp 数组中最多只有 $O(\frac{siz_x}{k})$ 个位置有值，所以合并仍然为树形背包。

而背包的上限还会和 k 取 $\min$，所以将 NTT 换成把有值的位置暴力卷积后复杂度为 $O(\frac{n}{k}\times k)=O(n)$。

问题

有一棵 n 个点的有根树，第 i 个点有一个重量为 b_i，权值为 a_i 的物品，定义一组选物品的方案的权值为所选物品的权值和，要求所选物品之间不能有边相连，对于所有 (x, i) $(1 \leq x \leq n, 1 \leq i \leq m)$，你需要求出 x 子树内所有满足选出物品重量之和为 i 的方案的权值和的最大值。
不妨认为 $m > n$。

这里我们不考虑基于暴力的 $O(\frac{nm^2}{\log m})$。我们只考虑基于启发式合并的做法。

但值得注意的是，我们考虑一个节点是否要选的时候其实也考虑了这个点的父亲是否选，也就是说，做 DFS 序背包的时候需要记录一下祖先的状态，于是就很容易倒闭。

考虑分析复杂度，设第 x 个点跳到根跳了 f_x 次轻链，则复杂度为 $\sum 2^{f_x}$。

从另一个角度分析，设当前子树大小为 n，儿子子树大小分别为 $a_{1\cdots m}$，且 a_1 最大，那么复杂度 $T(n) = T(a_1) + 2\sum_{i=2}^{m} T(a_i)$。可以通过一些分析得到当树为二叉树是复杂度取到最值，此时总复杂度为 $n^{\log_2 3}m$，可以近似看作是 $O(n^{1.5}m)$。

上述算法在 m 很大，n 较小的时候比较有优势，然而当 n, m 差不多大的时候，上述算法还有优化空间。

具体来说，我们考虑设阈值 B，并同时删除所有 $siz \leq B$ 的点，这样树的叶子数量就是 $O(n/B)$ 级别了，也就是说合并次数就是 $O(n/B)$ 了，这部分复杂度就是 $O(\frac{nm^2}{B})$。

考虑对于那些被删除的子树，考虑用之前的方法进行计算，具体来说就是先把这个点没删的那些子树贡献算上，然后用启发式合并的方法把被删掉的子树的贡献算上，复杂度为 $O(B^{1.5} \times \frac{n}{B} \times m) = O(nB^{0.5} \times m)$。

平衡一下，B 取到 $m^{2/3}$ 时复杂度为 $O(nm^{4/3})$。

问题

有一棵 n 个点的有根树，第 i 个点有一个重量为 1，权值为 a_i 的物品，定义一组选物品的方案的权值为所选物品的权值乘积，要求选的物品对应的点构成一个包含根节点的连通块。求出所有连通块大小为 $i(1 \leq i \leq n)$ 的方案的权值之和。

$n \leq 10^5$。

还是常规方法，设 $f_{i,j}$ 表示 i 子树内连通块点数为 j 的所有方案的权值和，不过状态数直接就爆了。

考虑对树重链剖分，我们只在每条重链的链顶记这个状态 $f_{i,j}$，这样状态数就是 $O(n\log n)$ 了，接下来问题变成了如何转移。

考虑对于一条重链，我们用分治 + 矩阵乘法进行转移。

具体来说，我们对这条链进行分治，设当前分治区间为 $[l,r]$，我们记录状态 $g_{l,r,i,0/1,0/1}$，其中 i 的范围为这条链上 $[l,r]$ 区间的点的轻子树大小和，表示当前区间总共选出 i 个点，左端点/右端点有没有被选的情况下所有方案的权值和。

简单分析一下复杂度，对于一条重链链顶节点 x，设其子树大小为 m，复杂度显然不超过 S 个一次多项式卷积的结果，也就是 $O(m\log^2 m)$，所有的 m 之和为 $n\log n$ 级别，也就是说总复杂度 $O(n\log^3 n)$。

注意到分治的时候可以带权分治，像全局平衡二叉树那样处理，复杂度就是 $O(n\log^2 n)$。

例题

问题 (GYM 102331J)

给定一棵 n 个点的树，每条边多有边权。
对于 $k = 1 \sim n - 1$，求出匹配数为 k 的最大权匹配，或报告不存在。
$n \leq 2 \times 10^5$。

设 $f_{i,0/1,j}$ 表示处理完以 i 为根的子树，i 这个点选/没选，匹配数为 j 最大权匹配。

注意到这个 dp 数组显然是关于 j 凸的，所以 $c_i = \max_{x+y=i}(a_x + b_y)$ 卷积可以做到线性。

然后继续采用上题的做法即可。

复杂度 $O(n \log^2 n)$ 或 $O(n \log n)$。

接下来讲讲长链剖分。
废话不多说，直接上论文。

长链剖分相关概念

定义 (高度)

定义一个节点的**高度**为该节点子树内距离该节点最远的点到该节点的距离 $+1$，叶子的高度是 1。

定义 (长儿子)

定义一个节点的**长儿子**为所有儿子中高度最大的儿子，如果有多个可以任选一个，如果是叶子则无长儿子。

定义 (短儿子)

定义一个节点的短儿子为不是长儿子的那些儿子构成的集合。

长链剖分相关概念

定义 (长链)

对于所有非叶节点只保留到长儿子的边之后构成的那些链（可以为单点）称之为是长链。定义长链的长度为其包含的节点数。

定理

设 $f(x)$ 为点 x 的短儿子的高度和。
$\sum_{i=1}^{n} f(i) = n -$ 根节点的高度。

证明.

对 $\sum_{i=1}^{n} f(i)$ 进行贡献拆解之后可以发现，只有根节点所在的长链没有被算进去，其它的长链都恰好被算了一次。 □

邻域相关概念

部分名字是笔者自己取的。

定义 (邻域)

定于二元组 (x, d) 表示的邻域为树上距离 x 小于等于 d 的点构成的点集。

定义 (内邻域)

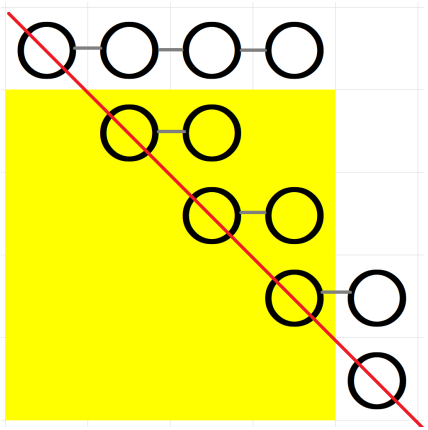
定义二元组 (x, d) 表示的内邻域为 x 子树内距离 x 小于等于 d 的点构成的点集。有些人也会把这个称之为是有向邻域。

定义 (外邻域)

定义二元组 (x, d) 表示的外邻域为 x 子树外距离 x 小于等于 d 的点构成的点集。

本章节将讨论离线情况下树上邻域查询的处理方法。

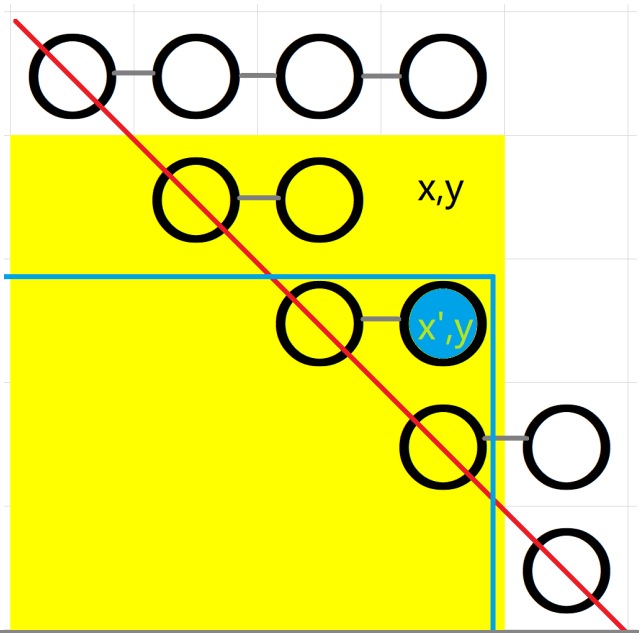
首先我们要解决的是数据结构方面的问题，我们先考虑以下形式的查询（允许离线）：



其中黄色的区域表示查询的区域，每个黑点上都有一个交换半群信息。下文中，在非特说说明情况下半群均默认为是交换半群。更具体来说，有序列 $a_{1\cdots n}$，在二维平面上 (i,j) 是黑点当且仅当 $i \le j < i + a_j$，每个黑点上有一个半群信息，然后进行若干次查询，每次查询一个位置（不一定是黑点）左下方的黑点的半群信息并。

如果信息可减，我们直接对行进行扫描线，从下向上扫，并且维护每个后缀和即可。

如果信息不可减，对于查询的 (x, y)（表示第 x 行第 y 列左下方所有位置的半群信息并），我们先快速求出其正下方第一个和他所在的列相同并且有值的位置 (x', y)，并预处理所有有值的位置的左下方的半群信息并，离线情况下找到 x' 是容易的，然后再求出第 $[x, x')$ 行的半群信息，合并起来就是答案。



也就是说，这类问题，可减情况下直接离线扫描线即可，对于不可减的情况，该问题可以直接转化成区间半群信息查询。

内邻域问题

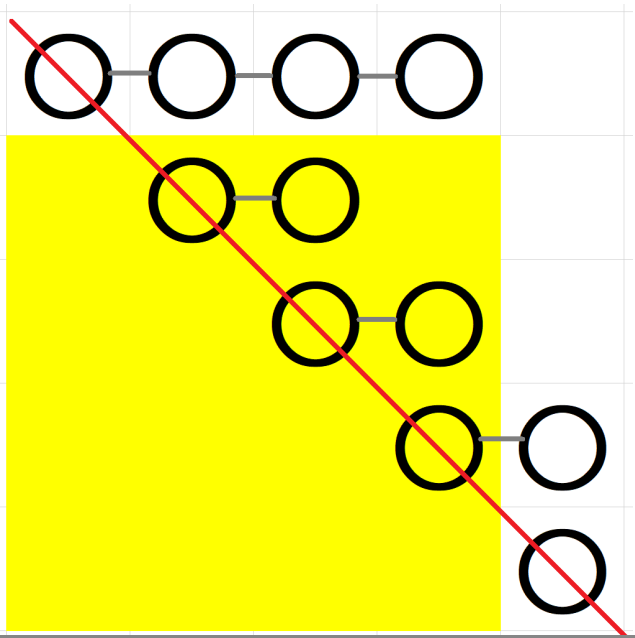
接下来我们将探讨内邻域的处理方法。

我们对于一条长度为 m、链顶节点为 x 的长链，我们先预处理出 x 子树内距离 x 等于 i 的那些点的半群信息。显然这部分直接暴力处理时间复杂度为线性。

然后现在我们对于每条长链进行考虑，我们对于一个点 x，我们预处理出一个长度为其所有短儿子高度的最大值的数组，其中第 i 项表示所有短儿子的子树中距离 x 等于 i 的点的半群信息并。

这样我们就可以对树的结构进行简化。我们可以把树看作是若干条长链，然后每个点除了所在长链外还可能挂了一条额外的链表示短儿子里面的信息，并且点数之和是 $O(n)$ 的，于是我们可以对于每条长链分别进行处理了。

对于内邻域的查询，本质上如图。直接使用刚才的方法处理即可。(其中红线上的点是长链上的点，其它点是长链上的点挂的额外的链上的点)：



邻域问题

接下来我们将探讨邻域问题，我们先进行一些简要的分析。我们记 x 点的高度记作 h_x。

考虑邻域查询，对于一个邻域 (x, d)，如果将其变成 $(fa_x, d-1)$，那么 x 子树外的点是否在这个邻域内的情况是没有改变的，如果 $d \geq h_x + 1$，那么邻域 $(x, d), (fa_x, d-1)$ 都包含了 x 子树内的所有点，那么我们转化成求 $(fa_x, d-1)$ 是不会改变答案的。

考虑不断令 (x, d) 变成 $(fa_x, d-1)$ 直到不能变为止，这样做有个好处，就是保证了询问的 (x, d) 有 $d \leq x$ 所在长链长度。

直接跳可以采取倍增，不过复杂度是 $O(\log n)$ 的，不优美，然而该问题做起来比较困难，再加上该操作本质上是用来优化复杂度的，所以我们实际上可以适当放宽限制。

定义 (跳链操作)

对于邻域 (x, d)，找到 x 最近的一个祖先 y 满足 y 所在的长链长度 $\times 2 \geq d$，然后跳到 y 位置，并将 d 减去 x, y 的深度差。显然，得到的邻域 (y, d') 满足 $d' \leq 2 \times$ y 所在长链长度。

接下来我们将证明跳链操作换成一步一步向上跳，即 (x, d) 每次变成 $(fa_x, d-1)$ 直到 $x = y$ 之前时都有 $d \geq h_x + 1$。

证明.

注意到 x 子树中任何一个点到 x 的距离均 $< x$ 的高度，而对于 x 到 y 路径上除 y 之外的点 z，都满足其所在的长链长度 $\times 2 \geq d$，也就是说从 x 跳到 z 时，d 的减小少量小于 z 所在长链长度，所以满足条件。 □

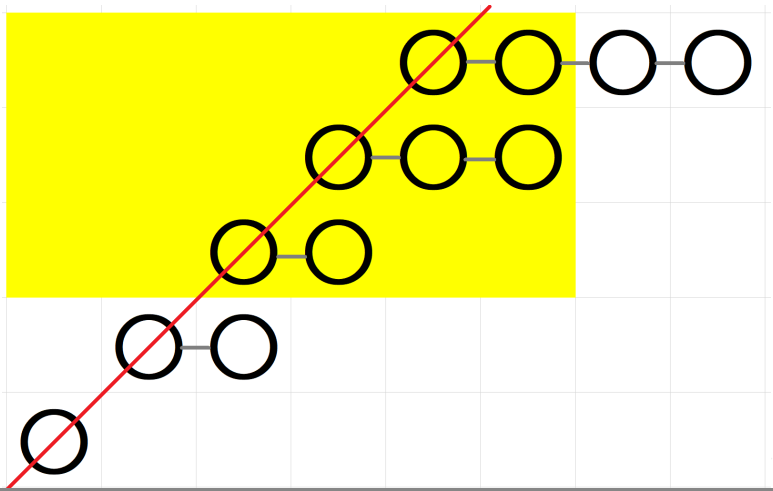
离线情况下的线性求法

该问题本质上来说就是给定 x, k，找到最近的祖先 y 满足 y 所在长链长度 $\geq k$。考虑允许离线的情况下怎么做，我们 DFS 并开 n 个栈，第 i 个栈表示当前 $d = i$ 的时候 x 应该跳到哪一条长链上。DFS 过程中我们优先 DFS 长儿子，当 DFS 到一个短儿子的时候，设这个短儿子所在的长链编号为 id，长度为 m，我们在第 $1 \sim m$ 个栈中压入 id，在完成这个短儿子的 DFS 之后再弹出第 $1 \sim m$ 个栈的栈顶。DFS 到 x 时对于询问 k，我们只需要访问第 k 个栈的栈顶就可以快速知道最后这个点会跳到哪条长链上。

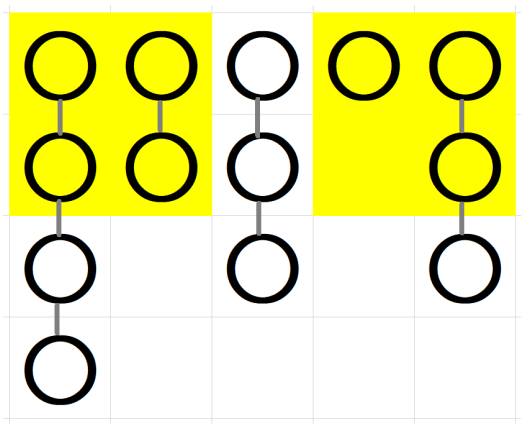
我们在 DFS 过程中再维护一点简单的信息就可以确定 (x, d) 邻域经过跳链操作之后得到的结果。

进行完跳链操作之后，我们再把邻域拆成内邻域和外邻域分开处理，对于每条长链的顶端节点 x，如果这条长链长度为 m，则我们再预处理出外邻域 $(x,1),(x,2),\cdots,(x,2m)$ 的答案。

如果我们能够预处理出来，那么可以发现，处于该长链的所有满足 $d \le 2 \times$ 该长链长度 的外邻域询问都可以像内邻域那样用同样的方式进行处理。具体来说，我们把贡献拆成链顶子树内的和链顶子树外的，而链顶子树外的（也就是链顶的一个外邻域）已经被预处理了，而链顶子树内的情况我们可以像内邻域那样求出该链链顶子树内的答案，如图：



考虑如何预处理那些外邻域，对于外邻域 (x,i)，我们可以转化成外邻域 $(fa_x,i-1)$ 拼上 $f(x,i-1)$，其中 $f(x,i)$ 表示 fa_x 去掉 x 之后大小为 i 的内邻域。对于 $(fa_x,i-1)$，按照上一页的求法求即可。对于 $f(x,i)$（显然保证了 x 是 fa_x 的短儿子），我们可以在 fa_x 处先算出长儿子的部分对答案的贡献，然后把所有短儿子提取出来，进行如下形式的计算：



直接暴力即可，复杂度线性。可以发现，我们将邻域问题在线性时间内转化为了区间半群信息查询。对于不可减情况下外邻域的做法，本质上相同但更为复杂，为了节省时间，这里略去。

在线情况

本部分需要使用 $O(n) - O(1)$ 在线树上 k 级祖先。

接下来我们将讨论强制在线情况下的处理方法。在此之前我们先探讨一个简单的问题。

问题 (序列前驱问题)

给定正整数序列 $a_{1\cdots n}$，有若干次询问，每次询问给定 i, x，你需要找到最大的 j 满足 $j \le i, a_j \ge x$ 或报告不存在，强制在线。设 $S = \sum_{i=1}^{n} a_i$，要求 $O(n+S)$ 预处理 $O(1)$ 查询。

离线情况是容易的，对于在线的情况，我们考虑建一个 S 个点的图，其中对于 $i = 1 \sim n$，建立 a_i 个点，分别为 $(i,1),(i,2),\cdots,(i,a_i)$。

然后对于每个存在的点 (x,y)，找到最大的 z 满足 $z \le x$ 并且点 $(z,y+1)$ 存在，然后 (x,y) 向 $(z,y+1)$ 连边，如果不存在则不连边，这构成了一棵内向树森林。

不难发现，对于每次询问，我们只需要找到 $(i,1)$ 的 $x-1$ 级祖先即可，这可以用 $O(n)-O(1)$ 树上 k 级祖先来实现。

之前的问题都需要做一个操作：对于 x，找到最近的祖先 y 满足 y 所在的长链长度大于等于一个指定的数 k。该问题离线下的做法已经讲述过了，接下来尝试在线做。

具体来说，我们对于一个已经长链剖分好的树，我们将每个长链看作一个点，长链 a 向长链 b 连边当且仅当 a 链顶的父亲节点为于长链 b 上。

事实上，做到这里，该问题已经可以用类似序列前驱问题的做法解决，对于点 x，设其代表的长链长度为 a_x，则建立节点 $(x,1),(x,2),\cdots,(x,a_x)$，然后对于 (x,y)，找到 x 最近的祖先（可以是 x）z 满足 $(z,y+1)$ 存在，然后 (x,y) 向 $(z,y+1)$ 连边，然后查询的时候找到 x 所在长链 i，求 $(i,1)$ 的 $k-1$ 级祖先即可。

注意我们现在求出的内容是 x 会一直跳直到跳到某条指定的长链的链上节点时会停止，为了求出这个点具体是哪个点，我们可以求出该链链底节点与 x 的 LCA。

回到内邻域问题的求法，我们的目标相当于是对于查询 (x, y)，如何找到其下方第一个和他所在的列相同并且有值的位置。

该问题做法也和序列前驱问题相同。具体来说，对于每个有值的点 (x, y)，找到最小的 i 满足 $(i, y+1)$ 有值且 $i \geq x$ 并向 $(i, y+1)$ 连边（如果不存在这样的 i 则不连边）。这样的话，对于查询，我们可以先找到 (x, x)（显然一定有值），然后查询其 $y-x$ 级祖先即可，使用在线 $O(n)-O(1)$ 树上 k 级祖先可以解决此问题。剩余部分都可以通过序列上的区间半群信息查询实现。

对于其它种类的邻域查询，做法类似但更加麻烦，介于篇幅问题这里不再展开。

属于长链剖分的应用，可能与邻域数点无关。

注意一道题长链剖分只是第一步，接下来可能还需要配合各种线性的处理方法，包括但不限于：前缀和、单调队列等。

问题 (CF150E)

给定一棵 n 个点的有根树，每个点有点权 a_i。
你需要找到一条经过的边数为 $[L, R]$ 的简单路径，使得经过的点的点权中位数尽量大，输出这个中位数即可。
定义一个长度为 m 的序列的中位数为该序列第 $\lceil \frac{m+1}{2} \rceil$ 大的数的值。

尝试做到 $O(n \log n)$。

首先二分，将问题转化成点权为 ± 1，问是否存在长度为 $[L, R]$ 且经过的点的权值和非负的路径。

首先长剖。

然后复杂度瓶颈变成了如何合并两条长链。当 $L = R$ 的时候是好做的。

当 $L < R$ 的时候，我们其实可以对于长链上的 i 位置，再维护一下从 i 开始向后 $R - L + 1$ 个位置的最大值。显然，这个的修改量就是合并长链的长度。

其实到这里这题只做完了一半，具体来说，还有一种查询，是查询长剖数组当前的一个前缀，因为长度 $< R - L + 1$，所以上一页讲述的处理方法并不能解决此问题。

实际上，这个可以套用邻域数点的方法，转化成序列区间最大值问题，虽然也是线性，但是不太优美。

如果只考虑 $L = 1$ 的情况（显然，这种情况下复杂度是最满的），相当于是合并一个长度为 m 的长链，就需要做 $R - 1, R - 2, \cdots, R - m$ 的前缀查询，并且我们希望复杂度为 $O(m)$。

实际上也不难，用单调队列维护即可。

这部分以数据结构为主。

ZJOI2019 语言

有一棵 n 个点的树，每个点都有一个集合，初始均为空，现在在上面做 m 次操作。

第 i 次操作给定 x_i, y_i，并将 x_i, y_i 路径上的所有点的集合中都加入 i。

然后询问有多少对点满足它们的集合交非空。

$1 \leq n, m \leq 10^5$。

存在一些简单粗暴的方法，比如树剖扫描线，扫描线过程用全局平衡二叉树维护即可做到 $O(n\log^2 n)$，不过好像听说 $3\log$ 也能过。

考虑对于每个点，维护和它有交的那些点构成的虚树大小。

注意到对于 m 个点 $a_{1\cdots m}$，按照 DFN 序进行排序之后 $\frac{d(a_1,a_m)+\sum_{i=1}^{m-1} d(a_i,a_{i+1})}{2}$ 就是虚树大小。

所以直接用 DFN 序做下标线段树合并维护即可。

当然理论上也可以直接用线段树合并对这棵虚树进行维护，只不过就复杂很多了。

Ynoi2008 rdCcot

给定一棵 n 个点的树以及参数 d，考虑一个 n 个点的新图 G，G 中 x, y 有连边当且仅当 x, y 在原树上的距离 $\leq d$。

现在有 Q 次询问，每次询问给定 l, r，求只保留 G 中编号为 $[l, r]$ 的点时形成的连通块数量。

$n \leq 3 \times 10^5, Q \leq 6 \times 10^5$。

数连通块通常有两种处理方法，点减边容斥和钦定代表元。

前者显然不靠谱，考虑后者。

考虑构造这个一个排列 p，使得对于 x, y, z 满足 $p_x < p_y < p_z$，如果 $dis(x, y), dis(y, z) \leq d$，那么 $dis(x, z) \leq d$，如果存在这样的排列，那么我们直接钦定连通块中 p 最小的那个点即可。

经过一些观察可以发现，p 为树的 BFS 序时条件满足。

然后相当于对于任意一个点，求出一对 L_i, R_i，表示只有满足 $L_i < l \leq x \leq r < R_i$ 时 x 才会成为代表元，从而转化成二维数点问题。

求 L_i, R_i 可以使用点分治做到 $O(n \log^2 n)$。

Ynoi2011 成都七中

给定一棵 n 个点的树，每个点有一个颜色，有 Q 次查询。

查询操作给定参数 l, r, x，求只保留编号为 $[l, r]$ 的点之后 x 所在的连通块的不同颜色数。

$n, Q \leq 10^5$。

考虑点分治，找到点分树上最浅的结点 u 满足 u, x 在保留 $[l, r]$ 的点后在同一个连通块中。

这样总共要处理的点的数量就不多了，然后对于一个分治中心，对于每个点求出要走到这个点 l, r 的范围。

然后就变成 2-side 矩形数颜色问题了，直接做即可，复杂度 $O(n \log^2 n + Q \log n)$。

Ynoi2008 rplexq

给定一棵 n 个点的树，有 Q 次询问，每次给定 l, r, x，求有多少对 (i, j) 满足 $l \leq i < j \leq r, \mathsf{lca}(i, j) = x$。

$n, Q \leq 2 \times 10^5$。

根号分治，对于儿子数 $\leq \sqrt{n}$ 的，直接用 $O(\sqrt{n}) - O(1)$ 分块计算即可。

否则的话，对于儿子子树大小 $\geq \sqrt{n}$ 的，同样暴力即可，对于子树大小 $\leq \sqrt{n}$ 的，将贡献拆成 siz^2 对，可以发现，总共的 siz^2 之和不超过 $n\sqrt{n}$。

用 $O(1) - O(\sqrt{n})$ 分块即可。

总复杂度 $O((n+Q)\sqrt{n})$。

QOJ 5015 树

这是一道交互题。

交互库一棵大小 n 的树，你需要通过询问确定这棵树的形态。

你每次可以给定一个点 x 和点集 S（要求 S 中的元素两两不同），交互库会返回该点到集合内的点的树上距离之和。

要求询问次数不超过 A，所有询问的 S 的大小之和不超过 B。

$n \leq 10^3, A = 8.5 \times 10^3, B = 3 \times 10^5$。

时间限制：$0.5s$，空间限制：$1024MB$。

首先随机一个根节点，然后确定每个点的深度。

然后考虑确定同一层之间的连边。

不妨假设现在是要确定 A, B 之间的连连边，其中 A 以及之前的部分都已经确定，现在要确定 B 的父亲。

考虑随机取出 A 的一个子集 S，那么 B 中 i 是 A 中 j 的父亲那么就有 $ask(i, S) = ask(j, S) + |S|$，于是可以按照 $ask(j, S)$ 和 $ask(i, S)$ 继续去递归。

我们可以多次随机 S 并找出其中最优的 S，这样做实测程序的运行次数不超过 7500。

QOJ 8283 Game of votes

给定一颗大小为 n 的有根树，每个点有权值 a_i, b_i，每个点的父亲都小于当前点编号。

定义一个点的 c 值为其所有没被标记的儿子的 b 值加上该点的 a 值。

初始整棵树的点都没被标记，每次计算出所有点的 c 值，然后选择没被标记的点中 c 值最大的那个并标记，如果有多个则取编号最大的。

现在有 Q 次操作，每次询问形式为如下两种之一：

1. 给定 x, A, B，更改 x 的 a, b 值为 A, B。
2. 给定 x, y，询问如果进行题目中所描述的标记过程，问 x 是否比 y 先被标记。

$1 \le n, Q \le 2 \times 10^5$, $1 \le a_i, b_i \le 10^8$。

时间限制：$2s$，空间限制：$1024MB$。

考虑设 g_i 表示第 i 个人被 ban 的时候记录一下此时他的 a 的值，这样比较两个人谁先被 ban 直接比较他们被 ban 时 a 值即可。

考虑如何计算一个点的 g 值，可以发现之和他所有儿子的 b, g 值有关。

具体来说，如果儿子的 b, g 值会发生变化，如何维护当前点的 g 值。不难发现一棵权值线段树即可解决所有问题。

于是不难想出一种单次修改 $O(dep \times \log V)$ 的做法。不过显然不能接受。

这显然是不能接受的，考虑继续优化。其实接下来的操作就比较（?）简单了。

重链剖分，然后用线段树来维护。具体来说，线段树节点 $[l, r]$（这里不妨假设线段树维护链是从上往下维护）记录三个信息 $f[0/1], p$。

其中 p 表示 r 点的重儿子一定在 r 之后才被标记的情况下该点的 g 值，$f[0/1]$ 表示 r 点的重儿子先/后比 r 标记时 l 点的 g 值。合并是显然的。

还有一个小问题，注意到所有点的一次更新会跳 $O(\log n)$ 条重链，所以空间也是 $O(n \log n \log V)$ 的，无法接受。

这个其实并不是问题，给动态开点线段树加一个空间回收即可。