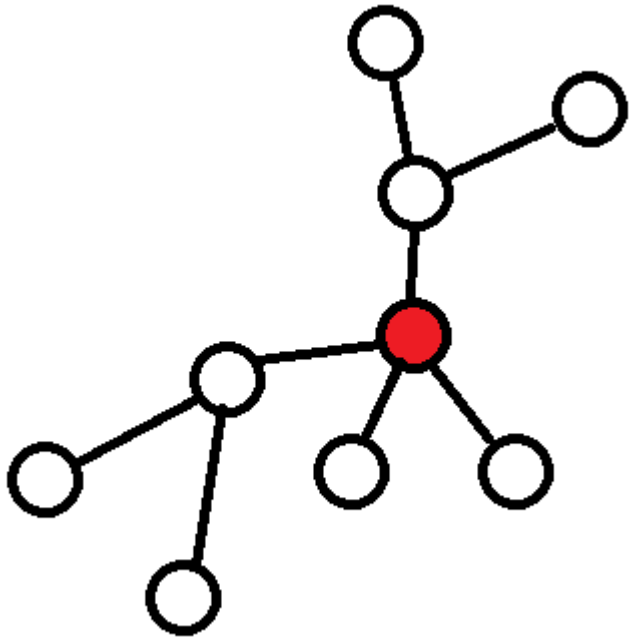
静态树分治

静态点分治

- 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重心，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

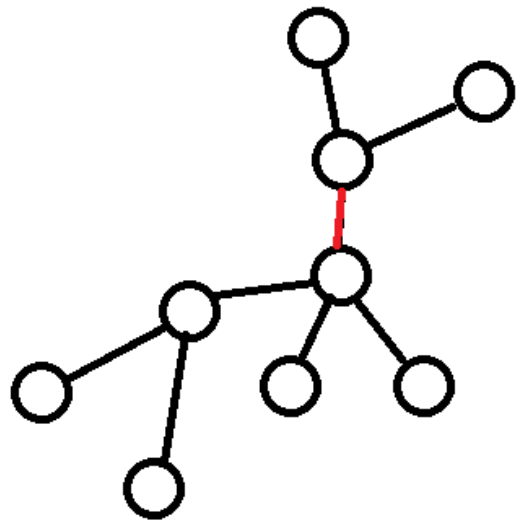


静态点分治

- 重心有三个定义
- 1. 到每个点距离和最小
- 2. 删去这个点之后最大的连通子图大小最小
- 3. 删去这个点之后最大的连通子图大小不过半
- 边权为 1 的时候三个定义好像是等价的，我们这里就当第三个定义来做
- 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 O(logn)，所以点分治复杂度 O(nlogn)

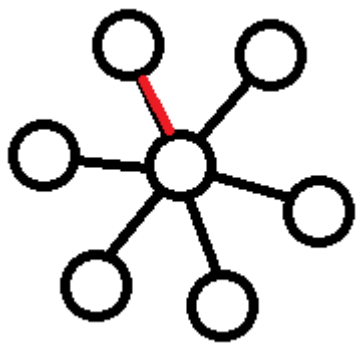
静态边分治

- 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
- 有什么问题呢?



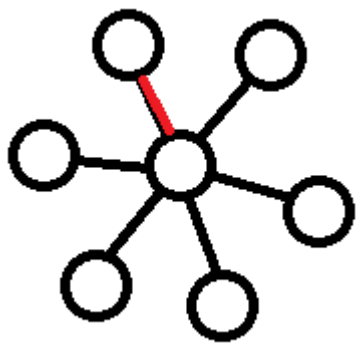
静态边分治

- 当树是菊花图的时候复杂度会有问题
- 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
- 于是 T(n)=T(n-1)+0(n)，复杂度为 0(n^2)



静态边分治

- 如何解决?
- 三度化



三度化

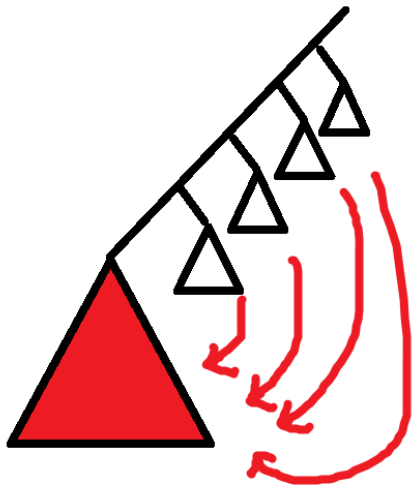
- 对于每个度数 >3 的节点，可以通过加虚点的方法让该节点度数变为 3

三度化

- 注意三度化只是可以解决部分树点度数过大的问题，因为其本质并不是对点度数进行了分治，而仅仅是改变了树的结构

静态链分治

- 基于轻重链剖分的结构，可以看作是每次删去一条重链之后继续分治下去
- 实际上和树上启发式合并是等价的：我们观察启发式合并的时候，我们每次是把 `size` 小的子树插入 `size` 最大的子树，这个可以看做是这个 `size` 大的子树所在的重链被删除了，然后依次合并重链上的轻儿子到这个重儿子上



静态树分治

- 一般维护路径信息的话点分治和链分治比较好写
- 维护子树信息的话链分治会很方便
- 边分治用来分析一些问题会比较方便，因为进行了点度数的分治

静态树分治

- 具体问题需要考虑具体使用哪种树分治会更简单，每种树分治有其优点和缺点

Luogu4149 [IOI2011]Race

- 给一棵树，每条边有权。求一条简单路径，权值和等于 k，且边的数量最小。
- n<=1e5,k<=1e6,边权非负

Solution

- 进行点分治，每次统计经过分治中心点的答案。
- 当前重心是 r 时，计算所有经过 r 的路径。因为 k<=10^6，我们便可以开个桶，g[i] 表示从 r 开始的权值和为 i 的所有路中，边数的最小值。
- 我们用 f[i] 表示当前子树的所有距离和前面子树的桶。
- 每次枚举一个子树，然后枚举子树里面每个点，假设这个点到根权值和为 x，然后 ans=max(ans,f[k-x]+g[x])，更新完答案之后将子树里面的每个元素加入 f 数组中
- 总复杂度 O(nlogn)

CF600E Lomsat gelral

- 查询每个子树的众数，即每个点所对应子树中出现次数最多的数

Solution

- 这种子树的问题用静态链分治，也就是树上启发式合并会比点分治和边分治方便很多
- 如何不使用数据结构的情况下，保证空间复杂度呢

Solution

- 我们开一个全局数组表示当前访问到的点子树中每个颜色出现次数
- 我们 dfs 到一个点的时候，先 dfs 每个轻儿子，算出每个轻儿子的答案
- 然后最后 dfs 重儿子，这样我们重儿子的这个数组可以保留，而不用清空，所以说这里复杂度是轻儿子 size 和
- 而轻重链剖分的性质保证，轻儿子的 size 和是 O(nlogn)，所以说复杂度是 O(nlogn) 的

[Ynoi2008] stcm

给定一棵树，你可以维护一个集合，支持三种操作：

1. 当前集合中插入一个节点 x
2. 撤回上一次插入操作
3. 将当前点集标为第 i 个点的子树补信息

一个点 x 的子树补信息定义为，树的点集除去 x 的子树（包括 x）内的点得到的集合；需要保证每个点的子树补信息都是正确的。

- n<=1e5，允许的插入操作次数为 4.5e6

Solution

- 对原树进行轻重链剖分；
- 假设当前在处理以 w 为根的大小为 n 的子树，且 w 是重链头，当前莫队状态为 w 的子树补。
- 可以在 O(n) 时间扫出 w 所在重链上，每个结点的子树补（这部分总代价为 O(nlogn)）；
- 然后将 w 所在重链的每个点的轻儿子找出来，之后要递归处理；

Solution

- 将这些轻儿子按子树大小加权，以及重链上每个点按 1 加权，建哈夫曼树，在哈夫曼树上 dfs，维护哈夫曼树上当前点的子树补，到达叶子时若叶子对应轻儿子则递归处理轻子树。

Solution

- 将整个递归过程中，产生的哈夫曼树拼接起来，得到所有叶子权为 1 的哈夫曼树，由于哈夫曼树上，
- 每向下两层子树权值和至少减小一个常数比例，由此得到遍历哈夫曼树过程的莫队的复杂度是 O(nlogn) 的。
- 因此总时间 O(nlogn) 。

Solution

- 使用轻重链剖分
- 1. 对根所在重链，可以在 n 次插入得到重链上每个点的答案，
- 2. 轻子树需要建立哈夫曼树，叶子的权重为子树规模，重链也作为一个叶子，权重为重链长度；
- 在哈夫曼树上 dfs，维护当前子树外对应的集合，移动到叶子时就递归到了轻子树上的子问题，所 需插入次数为所有非叶结点的子树规模之和（子树规模为子树中叶结点的权重之和）；
- 计算得到上界是 2.5nlogn+O(n)

[Ynoi2008] rdCcot & THUWC2020 某科学的动态仙人掌

给一棵边权为 1 的树和一个常数 C，节点用 1 到 n 的整数表示。

定义 $dist(a,b)$ 为节点 a,b 在树上的距离，即 a 到 b 的简单路径上的边权和，特别地，$dist(a,a)=0$。

每次查询的时候给出一个区间 $[l,r]$，查询有多少个 C-块，定义如下：

对任意两个节点 a,b，定义 a,b 是 C-连通的，当且仅当存在一个长为 t 的节点序列 $\{v_i\}$，满足：

1. $v_1=a$
2. $v_t=b$
3. 对任意 $1\le i\le t-1$，$dist(v_i,v_{i+1})\le C$
4. 对任意 $1\le i\le t$，$l\le v_i\le r$

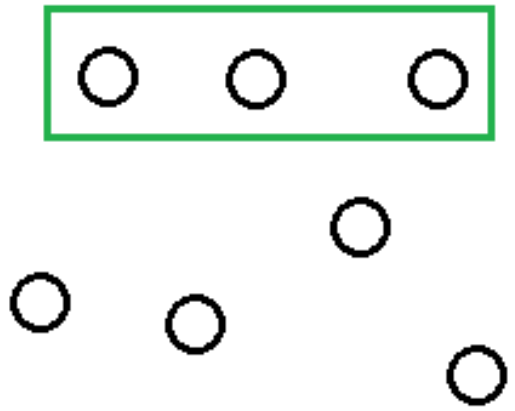
定义“C-块”为一个点集 S，满足：

1. 对任意 a 属于 S，b 属于 S 的补集，a,b 不 C-连通
2. 对任意 a,b 属于 S，a 和 b C-连通
3. 对任意 a 属于 S，有 $l\le a\le r$

- n<=3e5,m<=6e5

Solution

- 对每个连通块，考虑维护出一个代表元
- 维护每个连通块中深度最浅的点，来代表这个连通块，这样只需要对连通块最浅的点做计数即可
- 每个连通块可能有多个最浅的点，其深度相同



Solution

- 每个连通块最浅的点 x 需要满足条件:
- 1. 比其浅的点距离其距离 >C
- 2. 与其同深度且距离 <=C 的点需去重
- 定义树邻域 N(a,b) 表示距离 a<=b 的所有点构成的点集
- 第一个条件可以转换为
- 若 C 为奇数，则 N(fa(x,(C+1)/2),(C-1)/2) 中没有点在区间 [l,r] 内
- 若 C 为偶数，则 N(fa(x,C/2),C/2-1) U N(fa(x,C/2+1),C/2-1) 中没有点在区间 [l,r] 内

Solution

- 我们发现对于深度相等的一层节点，如果两个点 x,y 的距离 <=C，则 x,y 的 floor(C/2) 祖先一定是同一个点，这个条件是充分必要的
- 对 C 的奇偶性分类讨论
- C 为偶数时:
- -> : dist(x,y)<=C，深度相同，故对 k<=C， dist(fa(x,k),fa(y,k))<=C-2k，带入 k=C/2 得 dist(fa(x,k),fa(y,k))==0
- <- : 平凡
- C 为奇数时类似讨论
- 这个结论可以推论出深度相同的节点之间 C- 连通的传递性

Solution

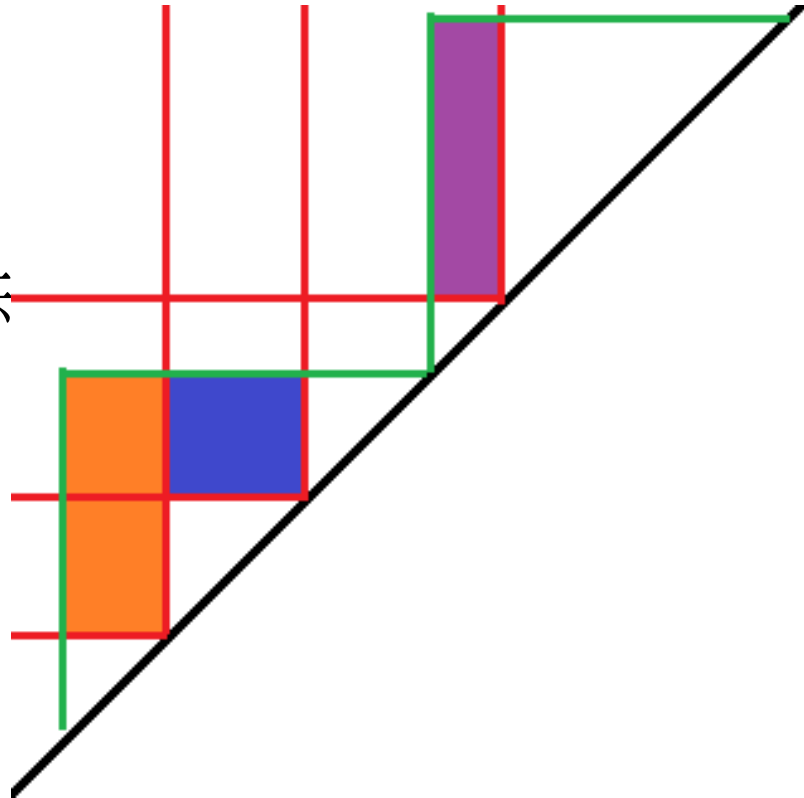
- 考虑对同深度，互相 C- 连通的一个极大点集，统计其 floor(C/2) 祖先，这样就可以对连通块唯一标号了
- 考虑每个深度的点的 floor(C/2) 祖先在哪些询问中被统计进去
- 对每个询问将其两个端点视为二维平面的两个维，询问变为二维平面上的点
- 每个祖先被统计进答案的范围是一些矩形的并
- 注意这里需要对每个等价类分别统计之后加起来，而不是每层深度的点的每个等价类的矩形的并

Solution

- 考虑对每个同深度的点集，按照 floor(C/2) 祖先划分等价类
- 对每个等价类，对每个元素 x ，找出其对应祖先的对应邻域中的前驱 a ，后继 b
- 即若询问区间包含 x ，则不能包含 a ， b
- 即 x 的贡献范围是 (a,x] * [x,b) 的询问
- 为了避免一个询问查询到多个该等价类中的点，对这些矩形求并进行去重

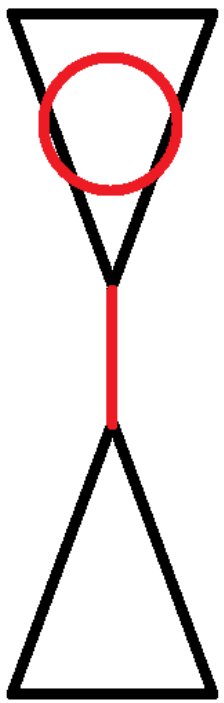
Solution

- 这个特殊的矩形形式，x 个矩形的并可以用 O(x) 个矩形表示
- 如图，红色的范围的并表示所有包含这个深度的点的询问区间的二维平面对应点
- 绿色的范围表示对每个点的前驱后继的限制
- 可能有一些点共用了前驱后继
- 这里 y 个点共用前驱后继则会划分出 O(y) 个矩
- 即所有在绿色下方，红色上方的点构成的矩形
- 总矩形数与该深度点数正比

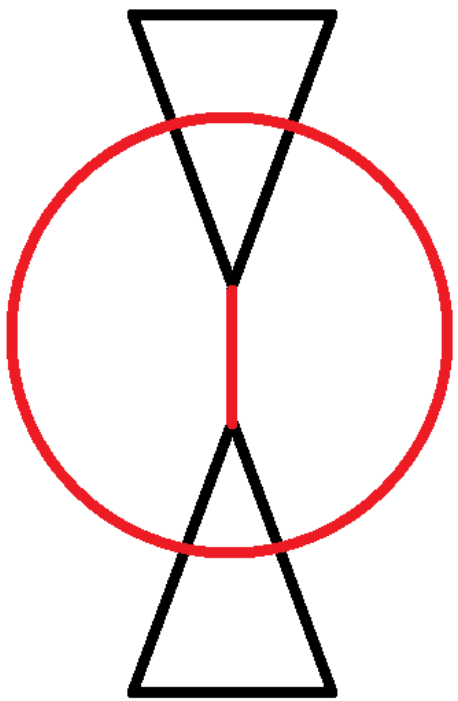


Solution

- 数据结构部分的问题是查询一个邻域中的前驱
- 邻域可以用边分治来划分出来
- 如果邻域在分治中心的一边

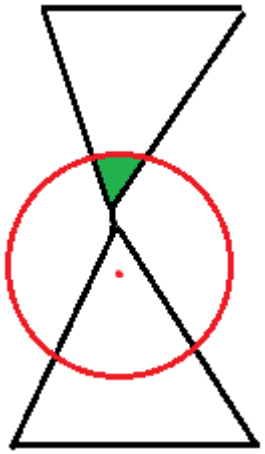


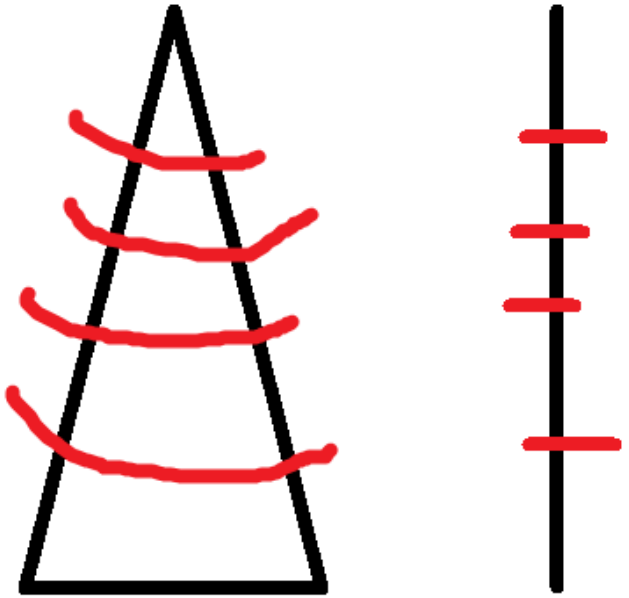
- 如果邻域被分治中心分成两半



Solution

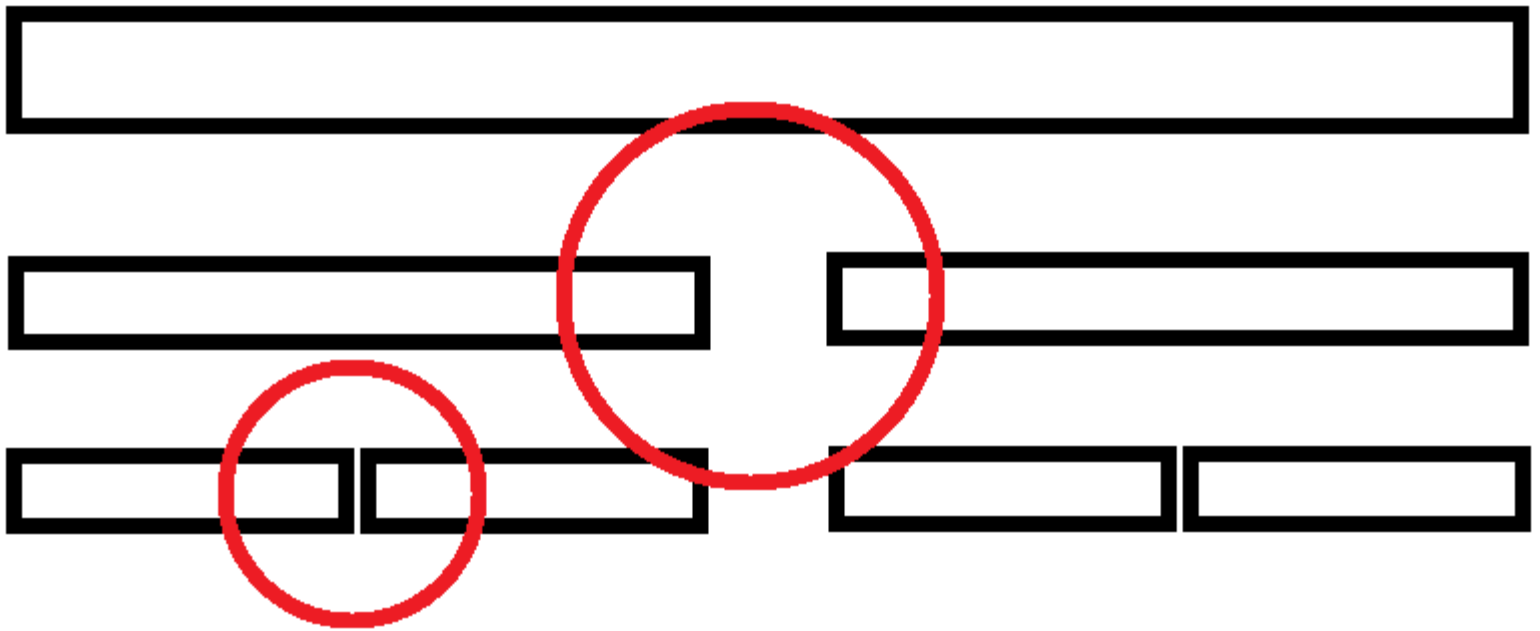
- 和序列分治类似，我们在第二种情况中，这个询问就是查询一个分治子图的前缀，邻域中心在的另一个分治子图递归下去继续分治
- 这里分治子图的前缀实际上和树的结构无关了，我们可以将其序列化





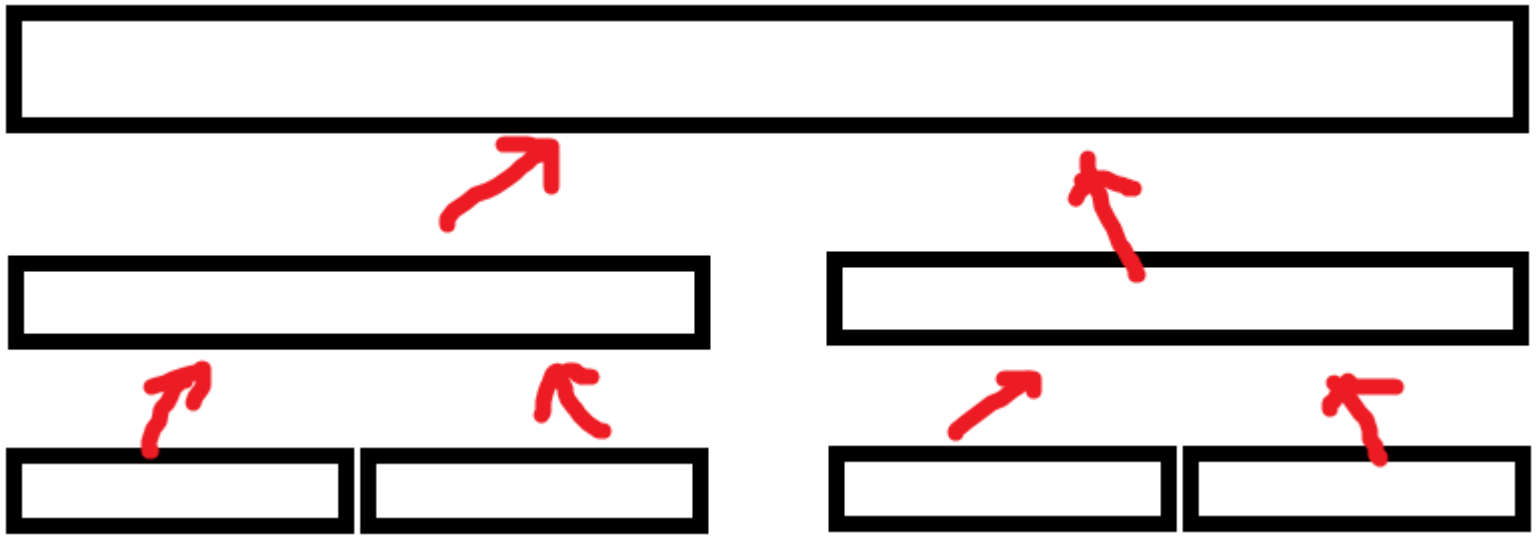
Solution

- 于是可以看做是一棵线段树，每层需要查询一些区间前后缀的前驱：



Solution

- 如何维护出每个线段树节点的每个前缀和后缀中 x 的前驱呢?
- 可以归并排序上去



Solution

- 在每层中，我们做的问题是：每次查询前缀 [1,x] 中小于 y 的最大的数
- 这个如果我们在线正着做，插入一个数，查询前驱，复杂度 o(loglogn)
- 离线?

Solution

- 每次删除一个数，查询前驱?
- 可以链表维护?
- 然而我们这个问题不太一样，我们有两个集合 A 和 B ，每次删除 A 集合中一个数，或者查询 B 集合中一个数在 A 集合中的前驱
- 如果大量 B 集合中的数都挤在一起，这样是不能用链表维护的

Technology

- 我们用并查集来维护，每次删除一个点之后将其和前面的点合并起来
- 然后我们预处理的时候把 B 集合每个元素挂在其最近的 A 集合元素上，这个把 AB 集合的元素一起排序就行了，可以在之前说的结构上归并上来
- O(α(n))

Technology

- 线性并查集：已知合并顺序的并查集可以做到 O(1)
- 考虑将并查集分为 θ(logn) 大小的块，这样并查集只用开 O(n/logn) 大小
- 每一个块里面每个位置是否被删除可以用二进制表示
- 当一个块所有位置都被删除后，将这个块和前面合并
- 我们这样只能定位到是在哪一个块里面，具体定位值需要位运算优化，这个可以做到 O(1)

Solution

- 分治并且归并的总复杂度 O(nlogn)
- 对于每个询问会拆成 O(logn) 个前缀前驱询问，每个询问是 O(1) 的

- 总时间复杂度 O((n+m)logn)

Solution

- 于是我们找出了每个深度的点贡献的矩形范围
- 问题变为给定一些矩形 +1，查询单点的值
- 可以扫描线 + 树状数组维护

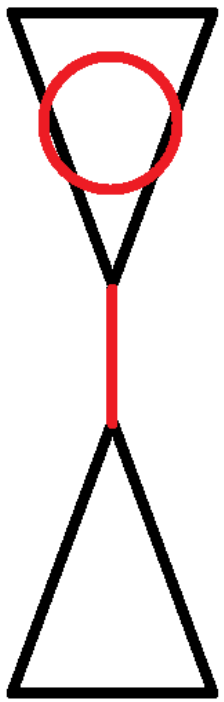
- 总时间复杂度 O((n+m)logn)
- 总空间复杂度 O(n+m)

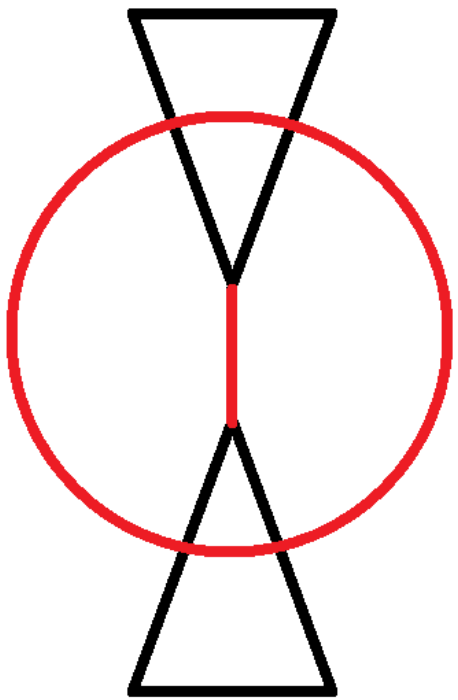
一个有趣的问题

- 给一棵边权为 1，有点权的树，每次查询一个距离 a 小于等于 b 的所有点中点权小于 x 的最大的数
- （树邻域前驱）

Solution

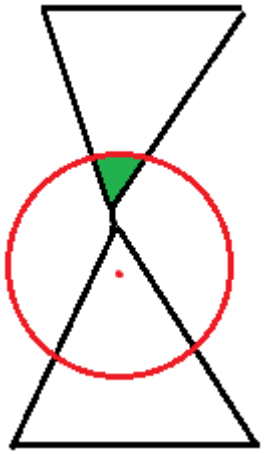
- 我们把距离 a<=b 的点集叫做树的 N(a,b) 邻域
- 邻域可以用边分治来划分出来
- 如果邻域在分治中心的一边 ||| 如果邻域被分治中心分成两半

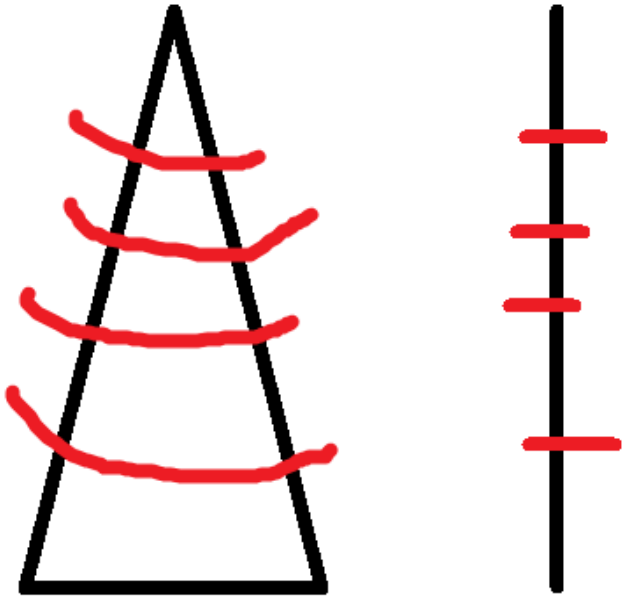




Solution

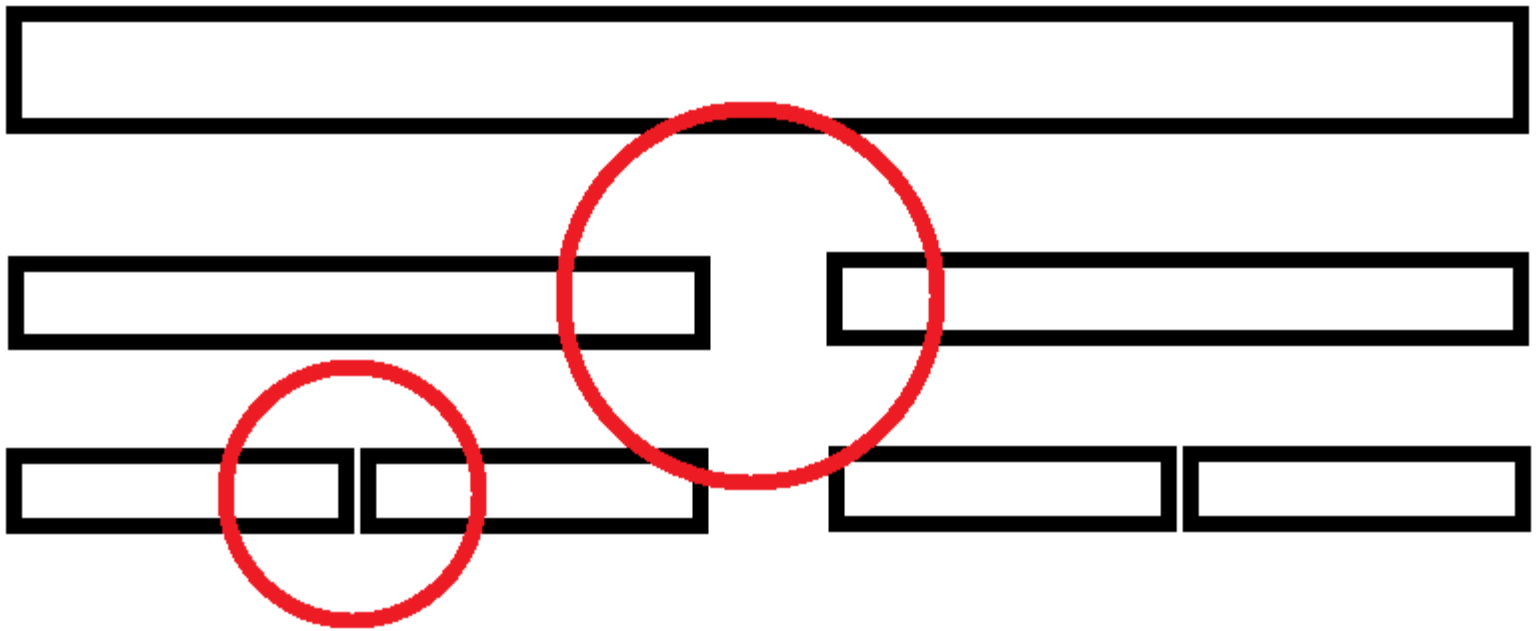
- 和序列分治类似，我们在第二种情况中，这个询问就是查询一个分治子图的前缀，邻域中心在的另一个分治子图递归下去继续分治
- 这里分治子图的前缀实际上和树的结构无关了，我们可以将其序列化





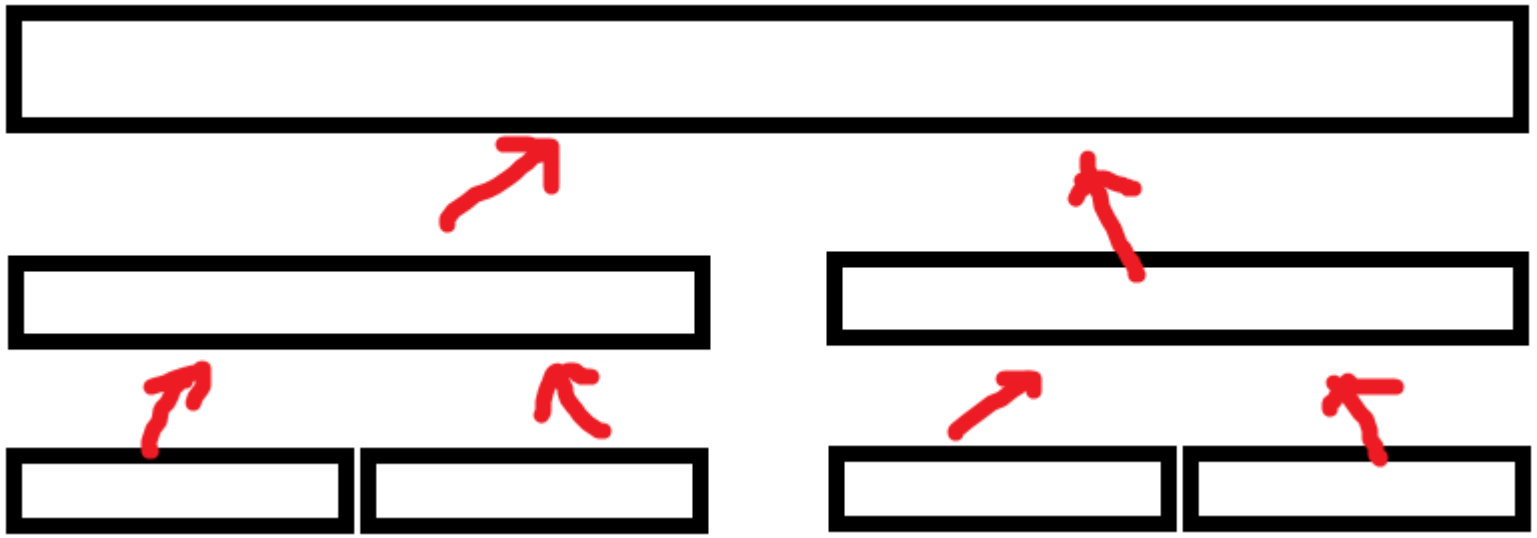
Solution

- 于是可以看做是一棵线段树，每层需要查询一些区间前后缀的前驱：



Solution

- 如何维护出每个线段树节点的每个前缀和后缀中 x 的前驱呢?
- 可以归并排序上去



Solution

- 在每层中，我们做的问题是：每次查询前缀 [1,x] 中小于 y 的最大的数
- 这个如果我们在线正着做，插入一个数，查询前驱，复杂度 o(loglogn)
- 离线?

Solution

- 每次删除一个数，查询前驱?
- 可以链表维护?
- 然而我们这个问题不太一样，我们有两个集合 A 和 B ，每次删除 A 集合中一个数，或者查询 B 集合中一个数在 A 集合中的前驱
- 如果大量 B 集合中的数都挤在一起，这样是不能用链表维护的

Technology

- 我们用并查集来维护，每次删除一个点之后将其和前面的点合并起来
- 然后我们预处理的时候把 B 集合每个元素挂在其最近的 A 集合元素上，这个把 AB 集合的元素一起排序就行了，可以在之前说的结构上归并上来
- O(α(n))

Technology

- 线性并查集：已知合并顺序的并查集可以做到 O(1)
- 考虑将并查集分为 θ(logn) 大小的块，这样并查集只用开 O(n/logn) 大小
- 每一个块里面每个位置是否被删除可以用二进制表示
- 当一个块所有位置都被删除后，将这个块和前面合并
- 我们这样只能定位到是在哪一个块里面，具体定位值需要位运算优化，这个可以做到 O(1)

Solution

- 分治并且归并的总复杂度 O(nlogn)
- 对于每个询问会拆成 O(logn) 个前缀前驱询问，每个询问是 O(1) 的

- 总时间复杂度 O((n+m)logn)

CF741D Arpa's letter-marked tree and Mehrdad's Dokhtar-kosh paths

- 树，边权，字符集 22，好像是'a' to 'v'，问每个子树中有多少链可以重排为回文串
- n<=5e5，时限 3s

Solution

- 一条链可以重排为回文串当且仅当只有 0 或者 1 个字符出现奇数次
- 这个可以用 xor 的性质帮助维护
- x -> 1 << (x – ‘a’)
- 即查询树上有多少链 xor 和为 0,1,2,4,…1<<22
- 树分治即可，方法和 race 那题类似，也是每个值开个桶，然后复杂度是 O(cnlogn)， c 是字符集大小

Codechef TSUM2

给定一棵 N 个节点的树（编号为 $1 \sim N$）。每个节点有点权，记第 x 个点的点权为 W_x。

定义一条简单路径 $v_1, v_2, \ldots, v_k$ 的权值为 $\sum_{i=1}^{k} i \cdot W_{v_i}$。树上的简单路径是满足下面条件的节点序列 $v_1, v_2, \ldots, v_k$：

- $k \geq 1$；
- 对于任意 $1 \leq i \leq k-1$，在节点 v_i 和 v_{i+1} 之间有边相连；
- 对于任意 $i \neq j$ 都有 $v_i \neq v_j$。

请求出树上所有简单路径中权值最大的一条。

- n<=5e4

Solution

- 点分治之后，因为路径是有向的，所以对于每一条路径都有向上和向下的两种。
- 如果一条向上的路径，点数为 s1，单独考虑这条路径的权值和为 v1，和一条向下的路径，点权和为 s2，单独考虑这条路径的权值和为 v2，这两条路径进行拼接（分治中心算在向上路径中，这样 s1>0）

Solution

- 那么拼接起来的路径的权值和就是 s1*s2+v1+v2 。
- 如果我们枚举到了一条向上的路径，对于每一条向下路径能够和这条向上路径拼接产生的贡献是一个一次函数的形式，同时横坐标范围在 1 到 50000 之间，所以可以使用李超线段树维护最值。
- 具体来说，我们在线段树上插入所有的二元组 (s1,v1)，然后对每个 (s2,v2)，我们需要找到一个最大的 s1*s2+v1+v2

Solution

- 考虑对每个二元组 (s2,v2)， v2 是常数了，所以实际上是最优化 s2*s1+v1，这里可以看做一个 ka+b 的形式，于是就是全局插入直线，单点插值查询最大

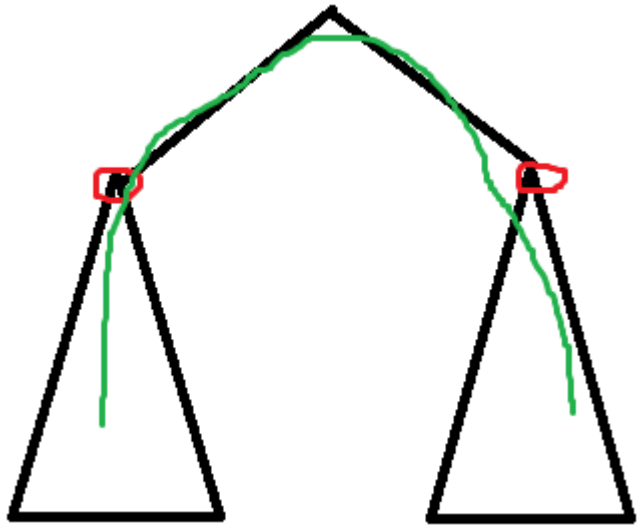
- O(nlog^3n)？复杂度感觉不好说

Loj 6276

- 树，有多少链满足上面的颜色互不相同
- 颜色数 20

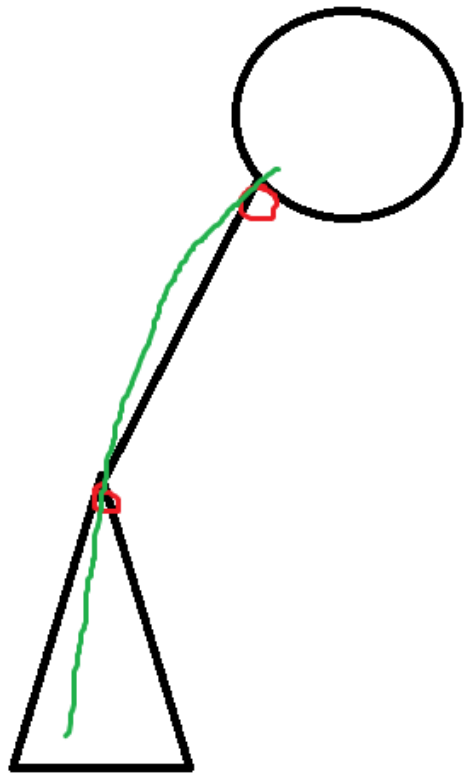
Solution

- 我们考虑提取出每种颜色
- 假设这个颜色出现在 x 和 y 的位置，如果 x 和 y 不构成祖先关系，则 DFS 序在 [lx,rx] x [ly,ry] 这个矩形中的所有链都是不可行的



Solution

- 如果二者构成祖先关系，则这个相当于是一个区间补的形式（就是删除一个子树的 DFS 序，也可以用 O(1) 个矩形表示）



Solution

- 所以我们可以预处理出每种颜色所导致的限制条件，然后问题转换为，给定 O(cn) 个矩形，求面积并
- 扫描线维护矩形面积并，总时间复杂度 O(cnlogn)

Luogu 5439【XR-2】永恒

有一颗 n 个点的永恒的树，树中每个点 $x(1 \leq x \leq n)$ 上都有一个永恒的字符串 $S(x)$。

但这世界没有永恒，所有的一切都会随着时光消失。我们只能给每个所谓永恒的东西定义一个永恒值 f。这个值本身没有意义，只是一个象征罢了。

- 一个字符串 S 的永恒值 $f(S)$ 定义为它的长度 $\mathrm{Len}(S)$，即：

$$f(S) = \mathrm{Len}(S)$$

- 树上的一条无向路径 $K = [u, v](u < v)$ 指的是 u, v 之间的简单路径（包括 u, v），其永恒值 $f(K)$ 定义为路径上所有不同的无序点对 $(x, y)(x \in K, y \in K, x < y)$ 上的字符串 $S(x), S(y)$ 的最长公共前缀 $\mathrm{LCP}(S(x), S(y))$ 的永恒值 $f(\mathrm{LCP}(S(x), S(y)))$ 之和，即：

$$f(K) = \sum_{x \in K, y \in K, x < y} f(\mathrm{LCP}(S(x), S(y)))$$

- 一颗树 T 的永恒值 $f(T)$ 定义为树上所有的无向路径 $[u, v](u \in T, v \in T, u < v)$ 的永恒值之和，即：

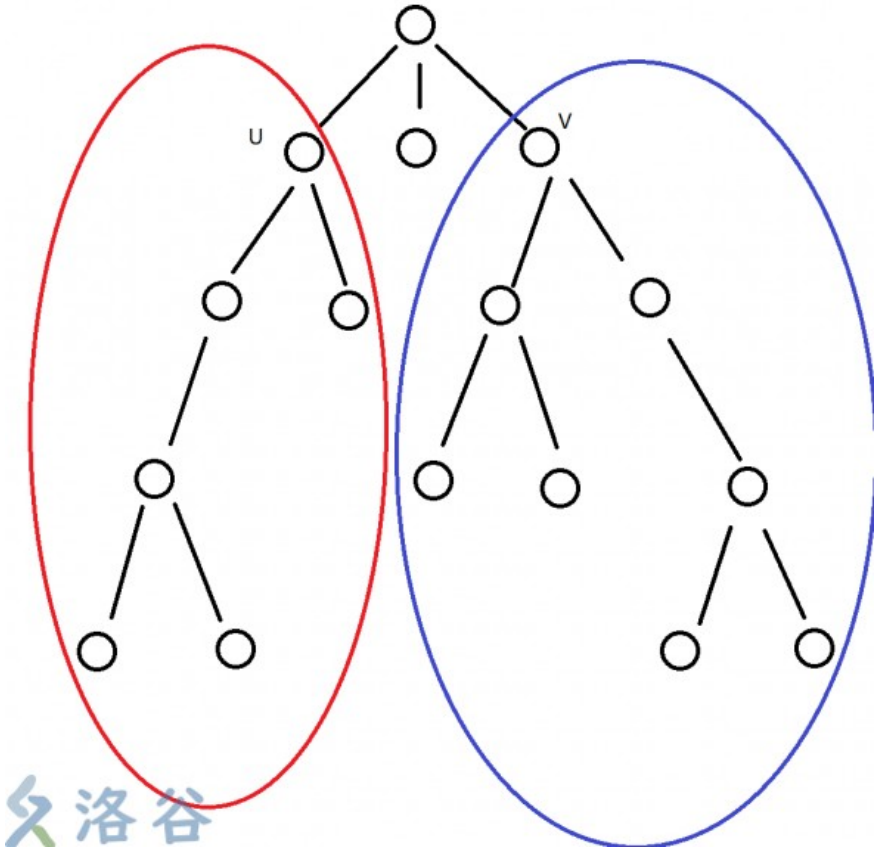
$$f(T) = \sum_{u \in T, v \in T, u < v} f([u, v])$$

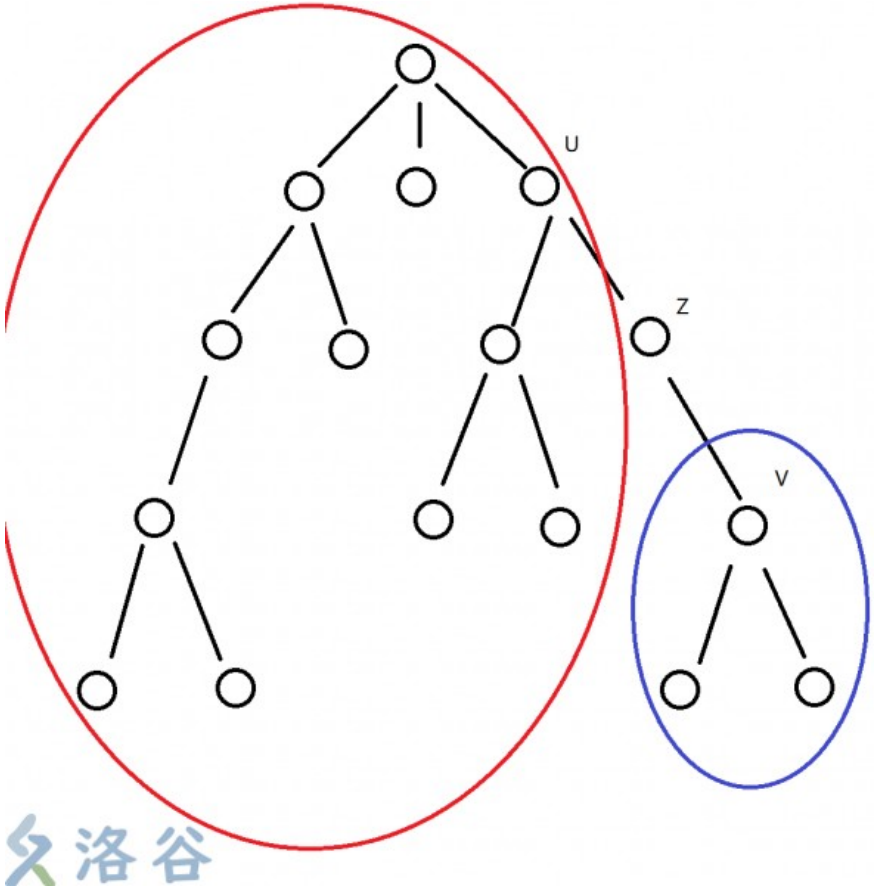
特别的是，树中每个点上的字符串都来自一颗永恒的以点 1 为根的 Trie 树，即每个树中的点都对应着一个 Trie 树中的点，点上的字符串就是 Trie 树中从根节点到其对应的点形成的字符串。

你需要求出这棵树的永恒值，答案对 998244353 取模。

Solution

- 首先链的子链可以同个给每个链 x -> y 加个 x 左边的点个数 * y 右边的点个数这样的权值，变成普通的链统计





Solution

- 然后树分治一下，由于每个字符串都是 trie 上的一条根开始的链，发现这个就是对分出来的点集，求两两 lca 深度和
- 由于树分治，所以点集大小和为 O(nlogn)
- 这个显然可以枚举每个点，点到根加点到根和来做掉
- O(nlog^2n)

Solution

- 其实我们只用统计这个信息的话可以虚树维护
- 虚树然后离线计数排序的话可以 O(n) 统计
- 于是这道题可以做到 O(nlogn) 时间

Luogu3292 [SCOI2016] 幸运数字

- 一棵 n 个点带点权的树， m 次询问，求树上两点间路径异或值最大子集的异或值
- n<=2e4,m<=2e5,6s
- 由于不带修改，所以也算静态问题

Solution

- 链线性基
- 由这道题我们分析一下各种树上分治算法的复杂度吧
- 线性基是很特殊的分治信息，特性：
- 设字节为 w
- 空间 O(w)
- 合并 O(w^2)
- 插入元素 O(w)

Solution1

- HLD：每次合并 O(log^2n) 个信息，每合并两个信息的代价是 O(w^2)，总复杂度 O((n+mlog^2n)w^2)

Solution2

- LCT：每次合并 O(logn) 个信息，每合并两个信息的复杂度是 O(w^2)，总复杂度 O((n+mlogn)w^2)

Solution3

- 倍增：预处理合并 O(nlogn) 个信息，查询每次合并 O(log n) 个信息，每合并两个信息的复杂度是 O(w^2)，总复杂度 O((n+m)lognw^2)

Solution4

- 树分治：分治过程中总共插入 O(nlogn) 个数，因为树分治是把每条链拆成两条链合并，所以每次查询只需要合并两个信息，总复杂度 O(nlognw + mw^2)，比其他做法优越
- 能不能更低呢?
- 好像有人说可以 1log，但我不懂线性基的额外性质所以不知道怎么做