树上问题

成都七中　nzhtl1477

树链剖分

- 将树进行轻重链剖分，每次操作时将一条链分为 DFS 序上 log 段

Luogu4216 [SCOI2015] 情报传递

- 给你一棵树，初始每个位置没有点权
- 1 x：让一个点从当前时刻开始，每秒操作点权 ++
- 2 x y c：查询一条链中有多少点的点权大于 c
- 其中每秒操作点权 ++ 就是指我每操作一次，无论是否和那个点有关，那个点权值都会 ++
- 1 操作对于每个点只会开始一次

Solution

- 考虑我们肯定不能每次把 1 操作的点 ++，这样复杂度不对
- 看看 2 操作吧

Solution

- 查询 x -> y 中有多少点权 >c 的点
- 而点权是一秒 +1
- 所以等价于查 x -> y 中有多少当前时间 - 开始时间 >c 的点
- 即查询有多少点满足当前时间 -c> 开始时间
- 相当于查一条链中小于一个数个数的数
- 这样可以树链剖分 + 树套树做到 O(mlog^3n)

Solution2

- 进一步研究
- 如果第 i 次操作是 1 x，则相当于第 i 时刻 x 点开始
- 可以把操作 1 当成第 i 时刻的时候点 x 的权值从 0 变成了 1
- 如果第 i 次操作是 2 x y c，则相当于查询链 x -> y 中开始时间 <i-c 的点的个数
- 等价于查询第 i-c 时刻链 x -> y 上的和
- 通过之前讲的树上差分 + 树状树组直接维护即可
- O(mlogn)

Luogu4211 [LNOI2014]LCA

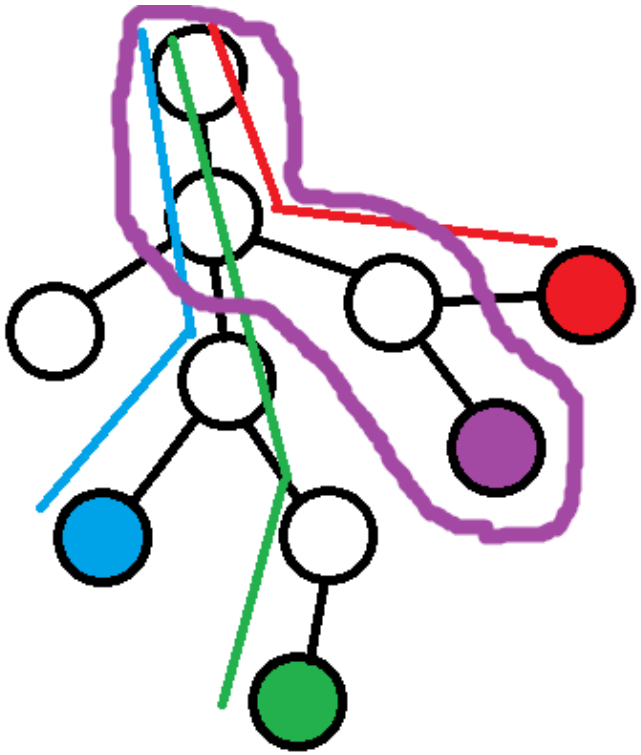
- 给出一个 n 个节点的有根树（编号为 0 到 n-1，根节点为 0）。
  一个点的深度定义为这个节点到根的距离 +1 。
  设 dep[i] 表示点 i 的深度，LCA(i,j) 表示 i 与 j 的最近公共祖先。
  有 q 次询问，每次询问给出 l r z，求 sigma_{l<=i<=r}dep[LCA(i,z)] 。
  （即，求在 [l,r] 区间内的每个节点 i 与 z 的最近公共祖先的深度之和）

Solution

- 首先将查询差分
- (l,r) -> (1,r) - (1,l-1)
- 然后考虑给一个点，怎么求其到一个前缀的点的 LCA 的深度和

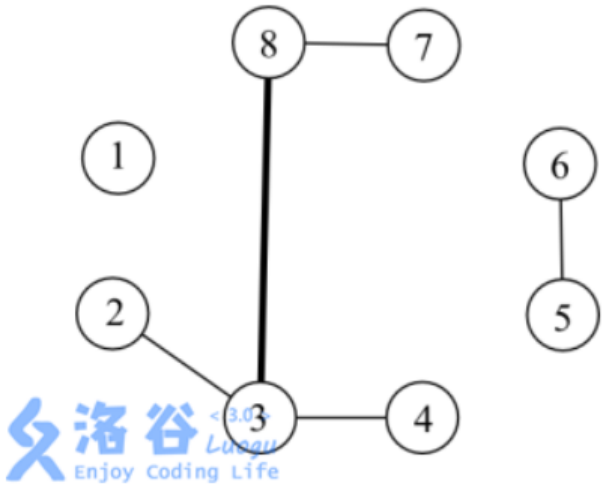
Solution

- 可以把每个点按顺序依次插入，每次插入把这个点到根的路径 ++ ，查询 z 到这些点各自的 lca 的深度和即查询 z 到根路径的和
- 如图，绿色，红色，蓝色的点被插入，
- 查询紫色的点
- O(mlog^2n)



Luogu4219 [BJOI2014] 大融合

小强要在N个孤立的星球上建立起一套通信系统。这套通信系统就是连接N个点的一个树。 这个树的边是一条一条添加上去的。在某个时刻，一条边的负载就是它所在的当前能够 联通的树上路过它的简单路径的数量。

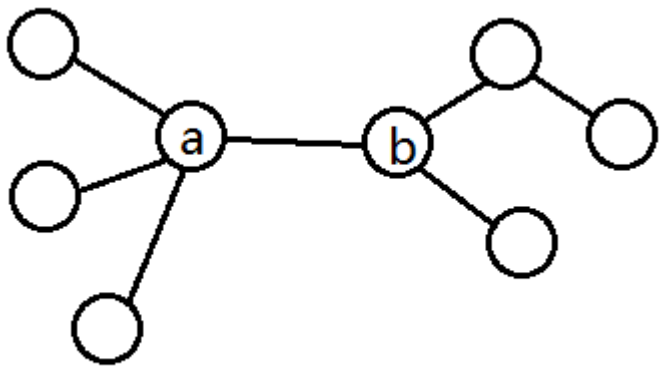


例如，在上图中，现在一共有了5条边。其中，$(3,8)$这条边的负载是6，因 为有六条简单路径$2-3-8$，$2-3-8-7$,$3-8$,$3-8-7$,$4-3-8$,$4-3-8-7$路过了$(3,8)$。

现在，你的任务就是随着边的添加，动态的回答小强对于某些边的负载的 询问。

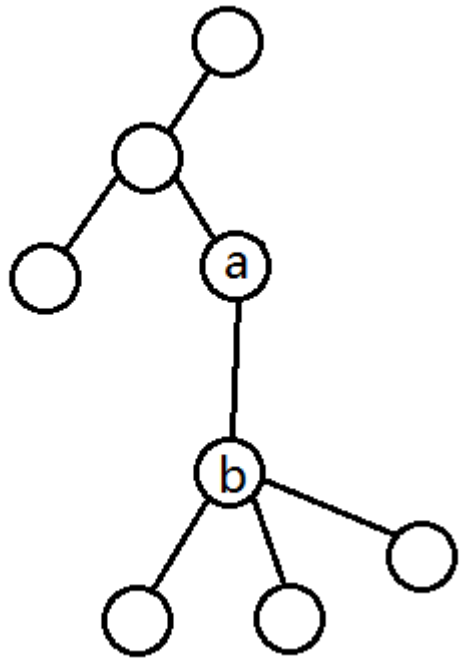
Solution

- 可以发现每次查询点 a， b 两端构成的简单路径个数
- 即等价于 a 不经过 b 的子树大小和 b 不经过 a 的子树大小的乘积
- 因为任意左边一个点和右边一个点都有唯一的而且不同的一条路径



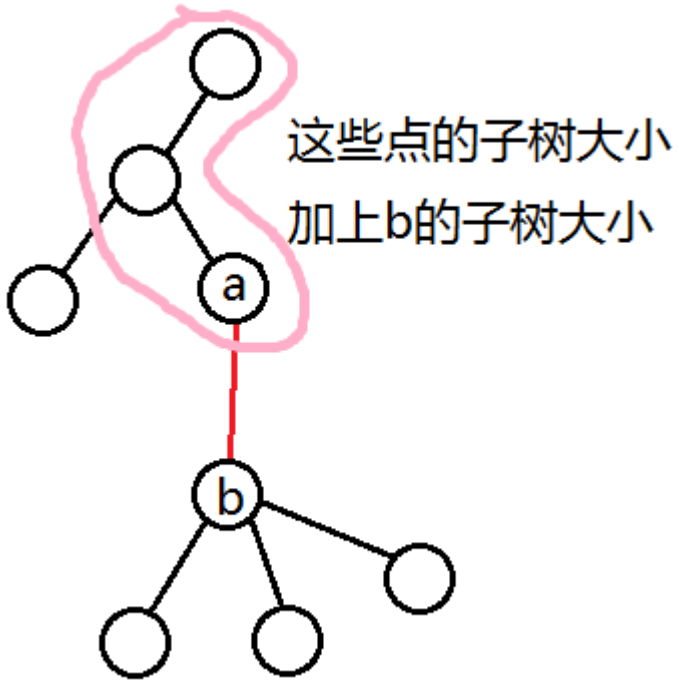
Solution

- 于是考虑离线，先把这棵树建出来，每个点维护当前连通状态下子树大小
- 假设 a 是 b 的父亲，则 a 和 b 之间路径的答案为 (a 所在联通块大小 -b 子树大小)* [illegible]



Solution

- 每个点维护子树大小，用并查集维护每个联通块的大小
- 则每次连接一条边的时候，等价于把一条链上的子树大小都加上一个值
- 比如连接 a 和 b 就是把：
- a 所在联通块里面深度最低的那个点到 a
- 这一条链加一个数
- 深度最低的那个点用并查集维护即可



Solution

- 问题转换为链加，单点求值
- 用树状数组轻松维护即可
- O(qlogn)

Luogu5354 [Ynoi2017] 由乃的 OJ

- 给你一个有 n 个点的树，每个点的包括一个位运算 opt 和一个权值 x，位运算有 &,|,^ 三种，分别用 1,2,3 表示。
- 每次询问包含三个数 x,y,z, 初始选定一个数 v 。然后 v 依次经过从 x 到 y 的所有节点，每经过一个点 i， v 就变成 v opti x i，所以他想问你，最后到 y 时，希望得到的值尽可能大，求最大值？给定的初始值 v 必须是在 [0,z] 之间。
- 每次修改包含三个数 x,y,z，意思是把 x 点的操作修改为 y，数值改为 z

Solution

- 如果我们知道每个位经过链上的变化之后变成了什么，那就可以贪心处理了
- 直接 HLD+ 拆位维护的话是 O(nlog^2nlogv)
- 用静态 LCT+ 拆位维护的话是 O(nlognlogv)
- 实际上不用拆位，可以压位来处理，每次处理所有的 0 位变成什么， 1 位变成什么即可
- 这样复杂度 O(nlogn)

bzoj 4771

- 一棵树，询问一个点的子树中与其距离不超过 d 的点有多少种不同的点权。
- n,m<=5e5,d 每次给出 ,5s

Solution

- 每个节点维护一个以深度为关键字的数据结构，如线段树
- 然后从下往上进行线段树合并
- 因为链分治的复杂度是 O(nlogn)，线段树合并的复杂度是插入一个元素 O(logn) 的，每次会把一段深度区间进行区间 +1

- 所以总复杂度是 O(nlog^2n) 的，使用长链剖分可能可以做到 O(nlogn)

经典问题（好像是 51nod 上面的）

- 给出一棵树，边权为 1，每次询问给两个 点编号的区间，求从两个区间中各选出一个点能得到的树上最远距离。
- 就是从 [l1,r1] 中选一个 a，[l2,r2] 中选一个 b，求 max dist(a,b)

Solution

- 直径的性质： A 集合中 a 到 b 最远， B 集合中 c 到 d 最远，这里有很多个直径的话只用选其中一个
- 则 A 和 B 的并集中直径是从这四个点里面选两个构成的
- 线段树维护区间直径端点即可
- 总合并次数 O(mlogn)，如果使用 O(nlogn) – O(1) 的 rmq
- 可以做到 O((n+m)logn)，这里不带修改，所以理论上可以做到 O((n+m)α(n))

Codechef DGCD

- 给出一棵 n 个点点权树，有 m 次操作：
  1 、询问一条路径上的 GCD ，即最大公约数
  2 、将一段路径上的点权加上 d

Solution

- 在序列上如何维护呢?
- gcd(a , b) = gcd(a - b , b)
- 将每个位置差分:
- b[i] = a[i-1]-a[i]
- 则 a 的区间加对应了 b 的单点修改
- a 的区间 gcd 和 b 的区间 gcd （特判端点）相同

Solution

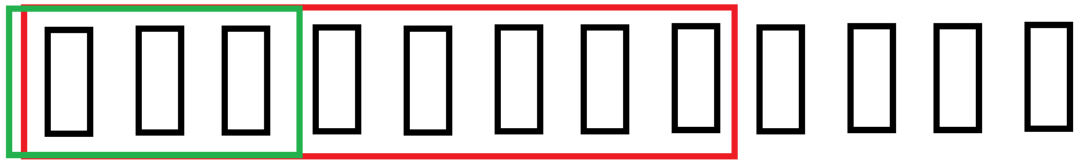
- 所以我们可以维护差分后的树，这里可以把每个位置和其父亲差分，也可以在树链剖分的 DFS 序上差分，区间加变成了单点修改，就可以维护了，注意需要特判端点
- 区间 gcd 是 O(logn + logv) 的

Luogu T11738

- 给一棵 n 个点的树，有 m 次查询
- 每次查询的时候给定 a 条链和 b 个子树还有一个值 t
- 把每条链上每个点 ++ ，每个子树中每个点 ++
- 求树上有多少点值 >=t
- 每次询问独立
- n=100000
- m=400000
- a 的和 =1500000 ， b 的和 =1000000

Solution

- 树链剖分，这样链变成 O(logn) 个区间，子树变成 O(1) 个区间
- 想想怎么不用数据结构维护这个东西
- 可以先想想差分



一次区间加差分为红色的前缀加和绿色的前缀减

Solution

- 我们肯定不能每次操作都暴力扫这个差分数组，会 TLE 的
- 所以我们考虑离散化
- 可以发现本来所有数都是一样的
- 进行了一次区间加之后，被加的那个区间和没被加的数值不一样了
- 但是不会对其他的数造成影响
- 也就是说区间加只会影响两个端点上的数，而区间中，或者区间外的点中，原本与相邻的点值一样的点还是与其相邻的点值一样

Solution

- 于是每次把所有差分后的前缀修改拿来排序离散化就可以了
- 我们可以离线，对每次操作的数一起排序
- 然后不用快速排序，而使用计数排序或者基数排序
- 这样时间复杂度变为 $O(a\log n + b)$
- 可以通过 100 分的数据
- 这里实际上可以用虚树做到 $O(a + b)$，不过要同时维护链和子树的虚树，然后对每个部分的每段合并起来，比较麻烦

Wannafly 挑战赛 13F

- 维护一棵 N 个结点的无根树。 支持两种操作:
- 1. 在链 (u,v) 的每个点上放一个可爱值为 k 的 fafa。
- 2. 问所有可爱值在 [l,r] 内的 fafa 到点 p 的距离之和。
- 一个点上可以放多个 fafa。
- N,Q≤ 10^5 。

Solution

- 链插入点？点插入链！
- 维护一个点集，支持加链，询问一个点到点集内所有点的距离和
- 距离和可以用之前讲的方法转化为 lca 深度和
- 所以就是链加等差数列，链和

Solution

- 每次查询是查询可爱值在 [l,r] 内的 fafa 到点 p 的距离之和，也就是点到区间的点集的距离和
- 所以这里加上一层线段树的分治，用线段树套树链剖分 + 线段树，或者树链剖分 + 树套树实现，单次 O(log^3n)
- 可以用高级动态树做到 O(log^2n)

Codeforces 757G

- 给出一棵 n 个点的树及一个 1~n 的排列 pi，边有边权，有 q 次操作：
- 1 l r x 求 ∑i=dis(p_i,x) , l <= i <= r
- 2 x swap(p_x,p_{x+1}) n,q<=2 * 10^5，强制在线
- 时限给大点给个 6s， 1024MB 吧

Solution

- 点到集合的 dist 和之前讲过，可以用点到根加点到根和的方法处理，这个有 1log 做法，不过用 2log 的树链剖分 + 线段树也可以
- 不带修改的话就可持久化一下线段树，每次询问差分 [l,r]=[1,r]-[1,l-1]，然后在两个可持久化线段树的位置上查询
- 这个 swap 相邻的操作如何处理?
- 直接换成修改?
- 但是这样会多一个 log

Solution

- 注意到我们只 swap 相邻的两个位置
- 比如 swap 了 x 和 x+1 的位置，那对 1,2,…x-1 的可持久化数据结构没有影响，由于有交换律，所以对 x+2,x+3,…n 的可持久化数据结构也没有影响
- 可以把 x-1 的可持久化数据结构插入 p[x+1]，作为 x 的可持久化数据结构

- 树链剖分 O((n+m)log^2n)，可以做到 O((n+m)logn)

Luogu5314 [Ynoi2011]ODT（弱化版）

- 给一棵树，边权为 1，支持：
- 1. 把一条链上所有点加上 k
- 2. 查询距离一个点 <=1 的所有点的点权 kth
- n<=2e5， 3s
- 部分分： n<=1e5， k=1

Solution1

- 暴力

- 总复杂度 O(nm)
- 期望得分: 20

Solution2

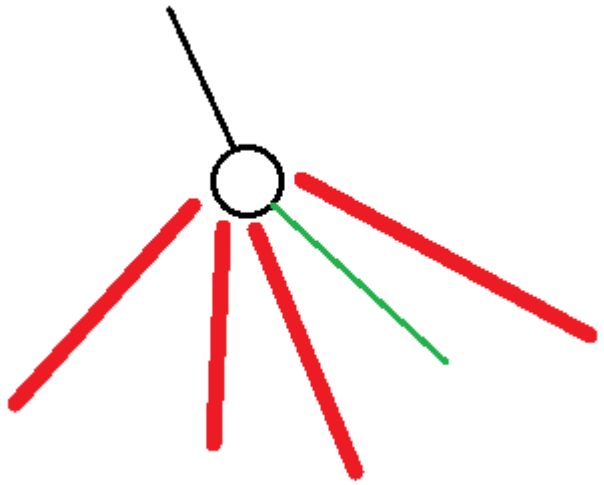
- 对点的度数进行根号分治
- 度数 >sqrt(n) 的点有 sqrt(n) 个，其他点度数都 <sqrt(n)
- 维护所有大点的数据结构
- 每次修改 O(sqrtn * 数据结构复杂度)

Solution2

- 查询的时候如果是大点就直接查，如果是小点就是暴力查询一圈
- 这个是 O(sqrt(n) * 数据结构复杂度)
- 对两个数据结构都用一个 O(sqrt(n)) - O(1) 平衡的分块
- 于是做到了 O(msqrt(n)) 的复杂度

Solution3

- 考虑链分治
- 一个点距离 <=1 的点里面，只有 3 个可能不是重链头
- 自己，父亲，轻儿子



Solution3

- 然后每次查询的时候暴力插入这 3 个（ O(1) 个）点
- 每次修改的时候把一条链上所有重链头的父亲都修改这个重链头的值
- 于是做到了查询 O(logn)
- 修改 O(log^2n)

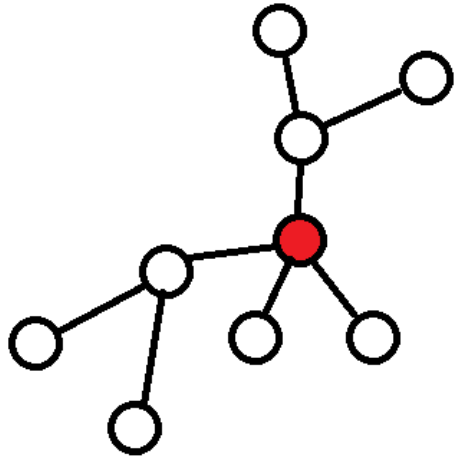
静态树分治

静态点分治

- 如何查询树上所有路径中满足某种条件的路径条数?
- 可以类比序列
- 序列上我们可以分治
- 树上呢?

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
- 点分治：每次找到一个点，统计经过这个点的路径条数，然后删掉这个点，递归到每个连通子图中统计
- 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

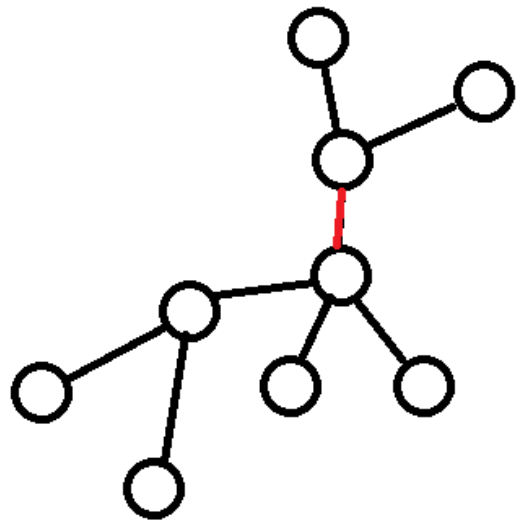


静态点分治

- 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)

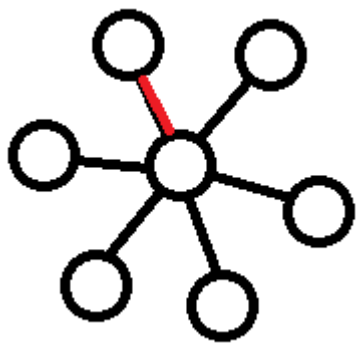
静态边分治

- 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?



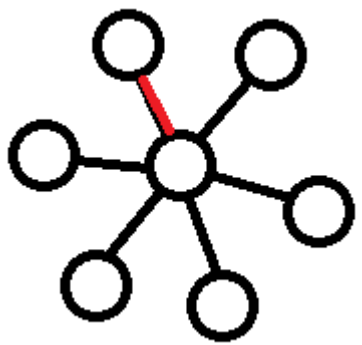
静态边分治

- 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

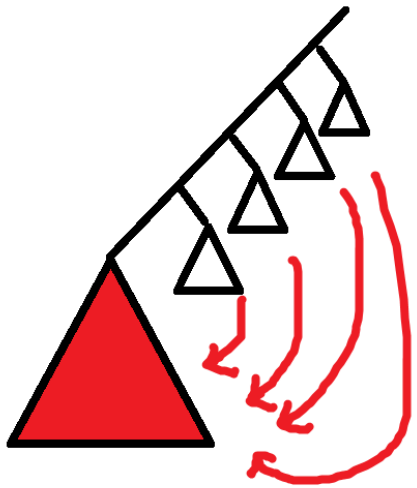
- 对于每个度数 >3 的节点，可以通过加虚点的方法让该节点度数变为 3

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

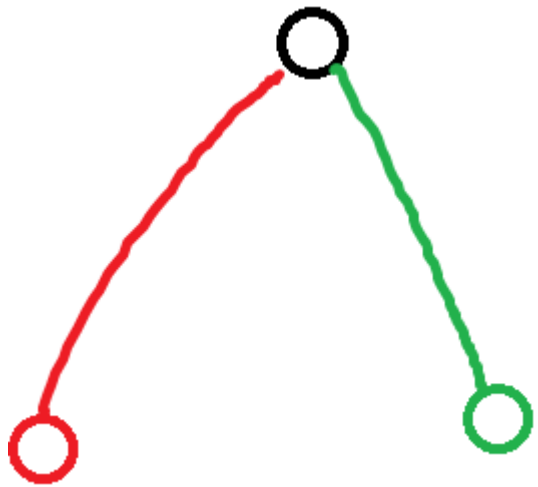
- 具体问题需要考虑具体使用哪种树分治会更简单，每种树分治有其优点和缺点

Luogu3806 【模板】点分治 1

- 给定一棵有 n 个点的树，询问 m 次树上距离为 k 的点对是否存在
- n<=1e4,m<=100,k<=1e7

Solution

- 对树进行点分治
- 每次考虑合并跨过分治中心的链
- 维护一个当前合并进来的链的集合，每次插入一条链的时候顺便查询这条链和集合中的链是否长度和加起来为 k



Solution

- 可以开一个 set 或者哈希表，每次插入一条链的时候插入其到当前分治中心的长度
- 如果插入 x 时， set 中有 k-x ，则有一条链长为 k ，否则没有

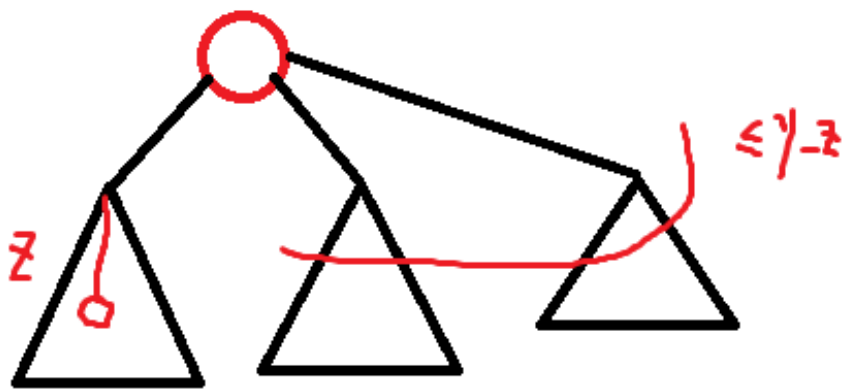
- O(nmlogn) 或 O(nmlog^2n) ，看实现
- 由于常数比较小所以能跑过

2020 年多省联考 B 卷 T2

- 给出一棵边权为 1 的树，多次查询距离 x<=y 的点数
- n,m<=1e5

Solution

- 讲一下点分治如何维护这个
- 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
- 然后其他分治子树中距离 x<=y 的点，满足距离分治中心 <=y-z
- 对每个点挂询问，然后点分治到每个点的时候处理一下这个点上所有询问，并把该递归下去的东西递归下去挂询问
- 每层预处理前缀和可以 O(1) 查询，时间复杂度 O((n+m)logn



随便 YY 的题 1

- 给一棵树，边有边权，求所有链中边权 xor 和最大的一条链

Solution

- 这里是边权
- 可以发现 x 到 y 的边权 xor 和等价于 x 到根的边权 xor 和，与 y 到根的边权 xor 和，的 xor
- 为什么

Solution

- 因为 xor 的性质，一条边被 xor 两次会自动抵消
- 可以发现 lca 到根的路径上每条边都被 xor 了两次，所以都抵消了
- 于是使用 trie 树，维护最大 xor 和即可

随便 YY 的题 2

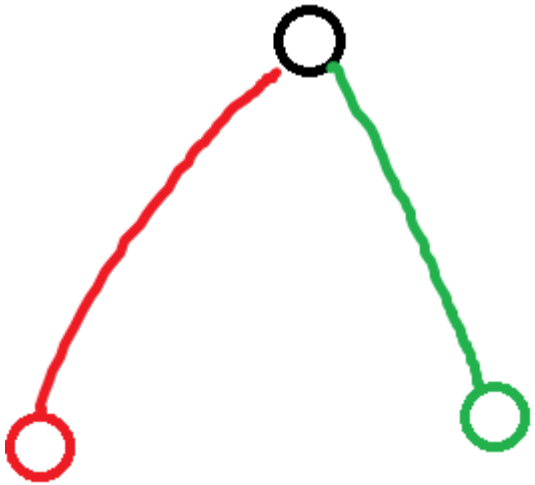
- 给一棵树，点有点权，求所有链中点权 xor 和最大的一条链

Solution

- 这个就需要树分治了

点分治

- 考虑使用数据结构优化计算过分治中心的路径
- 我们要选两个点到分治中心的链使得 xor 起来最大



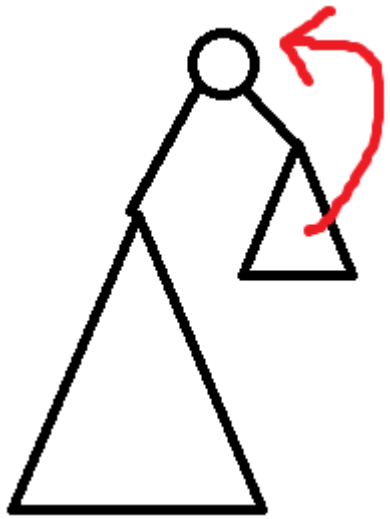
- 开一棵 trie 树，边插入点到分治中心构成的链，边查询最大 xor 和
- O(nlog^2n)

边分治

- 与点分治类似，除了需要三度化以外
- 优势是每次只用合并两棵子树，虽然在这道题中看不出优越性

链分治

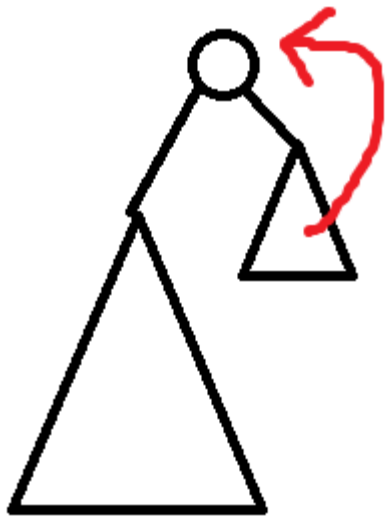
- 考虑启发式合并上来的过程



- 重儿子维护一棵 `trie` 树，启发式合并的过程中将轻儿子到当前节点的链插入 `trie` 树中，同时查询最大 `xor` 和

链分治

- 重儿子这里会每次在上面加一个点，所以需要把 trie 上所有元素都 xor 上一个值
- 实际上可以懒惰处理，记一下全局 xor 了 x ，之后把插入和查询的值都 xor 上 x 即可
- O(nlog^2n)



启发式合并

- 实际上是链分治

Luogu3224 [HNOI2012] 永无乡

- 给定一个图，初始有一些边，每个点有点权
- 1. 加边
- 2. 查询一个点所在连通块中的第 k 小点权

Solution

- 这题就是一个启发式合并的过程
- 每个点所在连通块开个并查集，然后开个平衡树来维护权值
- 合并的时候把小的连通块的权值插入到大的里面
- 用一些写法是 O(nlogn) 的，一些写法是 O(nlog^2n) 的，不过差不多

Luogu4197 Peaks

- 给定一个图，有点权和边权
- 每次查询给定 x,y,k
- 查询从 x 开始只能经过边权 <=y 的边，到达的所有点里面点权 k th

Solution

- 考虑将边权从小到大排序后加入
- 然后当加入的边权为 y 时，处理 x 节点关于边权 y 的询问
- 这个时候问题就和上一题的启发式合并类似了
- 复杂度相同

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<=2e5,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

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

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

Solution

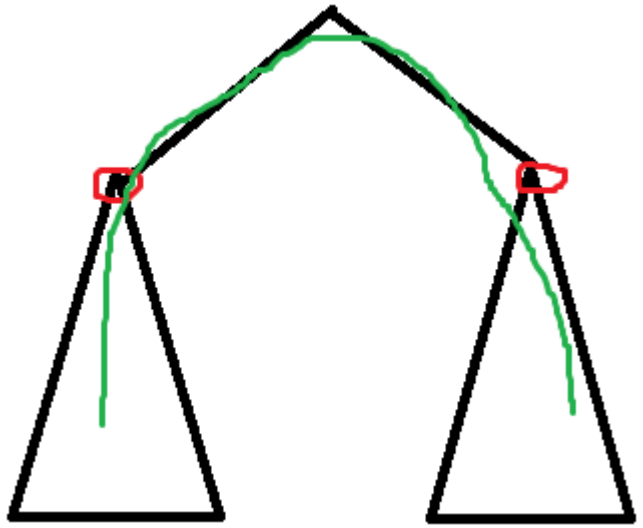
- 一条链可以重排为回文串当且仅当只有 0 或者 1 个字符出现奇数次
- 这个可以用 xor 的性质帮助维护
- x -> 1 << (x – ‘a’)
- 即查询树上有多少链 xor 和为 0,1,2,4,…1<<22
- 树分治即可，方法和 race 那题类似，也是每个值开个桶，然后复杂度是 O(cnlogn)， c 是字符集大小

Loj 6276

- 树，点有颜色，有多少链满足上面的颜色互不相同
- 每种颜色出现次数 <=20， n<=1e5， 4s

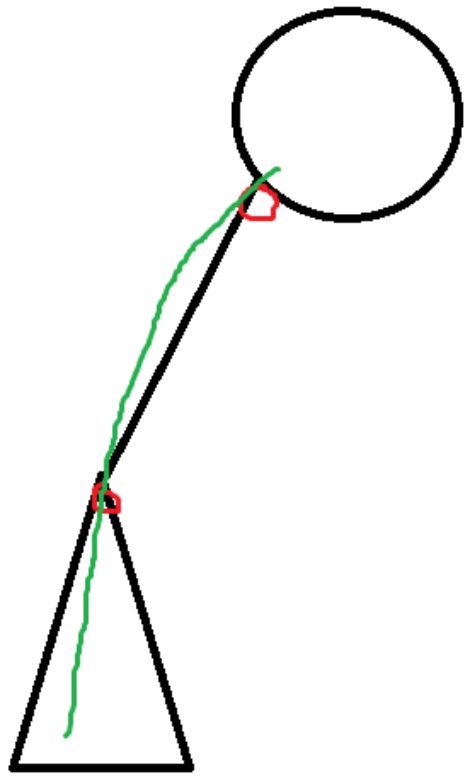
Solution

- 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）

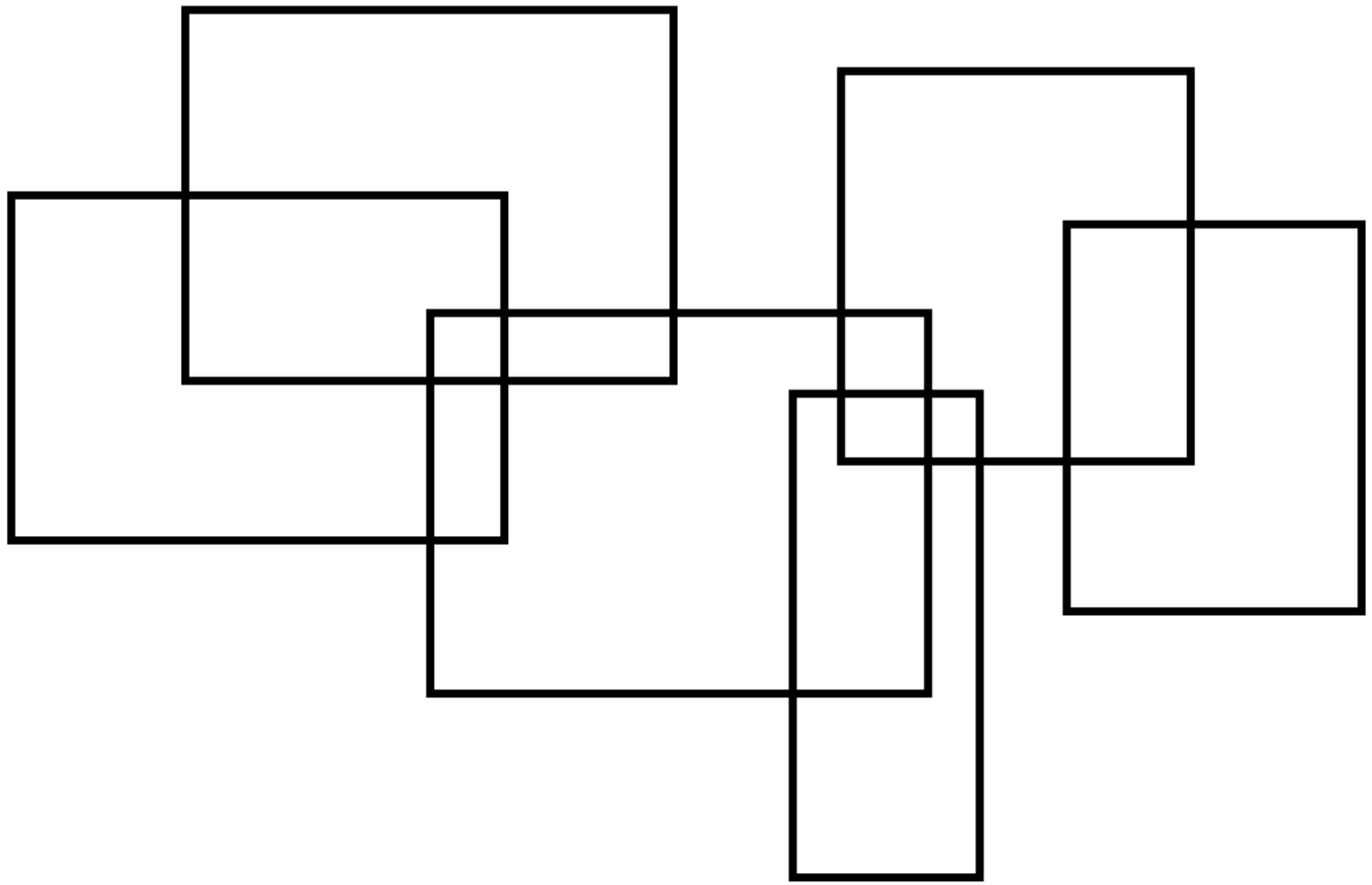


Solution

- 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)

矩形面积并

- 给定二维平面上的一堆矩形，求矩形面积的并



Solution

- 先离散化，因为端点个数只有 O(n) 个
- 然后扫描线，使用线段树维护：
- 进入一个矩形的时候区间 +1
- 走出一个矩形的时候区间 -1
- 这一部分中被覆盖的位置就是全局 >0 的位置
- 这里每个位置加了个权，就是其对应 y 轴区间长度
- 如何维护呢?

Solution

- 可以发现每个位置都 >=0
- 所以可以用那个维护 min 和 min 出现次数的方法维护 0 位置的个数

- 总复杂度 O((n+m)logn)

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，点权非负

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 6145

- 给出一棵树，每次询问一个点 x 到编号在 [l,r] 中的点的距离的最小值。

Solution

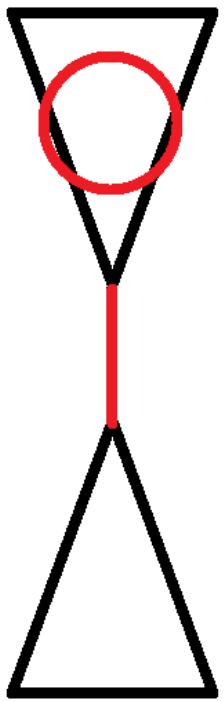
- 先把点分树搞出来，然后对每个分治中心按点编号顺序开一棵线段树来记录每个点到分治中心的距离最小值。
  查询的话，就在该点在点分树上到根的路径中所有的线段树上查询即可。
  为什么这样是对的呢？首先因为所有点都会被算到，其次，我们虽然可能算重，把一个点多算几次，或者把一个距离算的更长，但是不会少算，而且因为 min 是具有幂等律的信息，即合并多次和合并一次等价，所以这里不会构成影响
- 由于不带修改，所以可以使用静态的 rmq 结构
- O((n+m)logn)

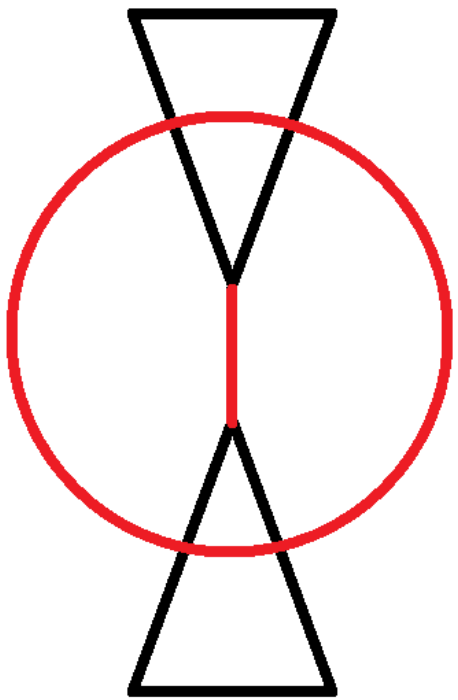
一个题的子集

- 给一棵边权为 1，有点权的树，每次查询给出 a b x，输出距离 a 小于等于 b 的所有点中点权小于 x 的最大的数

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 IIII 如果邻域被分治中心分成两半





Solution

- 第一种情况中，我们这一层不产生新的询问
- 第二种情况中，假设 a 在端点为 c 的部分，另一部分的端点是 d，我们这一层需要查询距离 d<=b-dist(d,a) 的所有点，然后递归进 c 的子树中
- 所以每层只会多一个询问，总共 O(logn) 个询问

Solution

- 我们会进行 O(logn) 次查询距离一个分治中心 <=x 的元素里面， y 的前驱
- 这里使用一个数据结构维护即可 O(logn)
- 总时间复杂度 O((n+m)log^2n)，可以做到 O((n+m)log n)

Loj 121

- 维护一个 n 个节点的无向图
- 1. 加边
- 2. 删边
- 3. 查询 x 和 y 是否连通
- 允许离线

线段树时间分治

- 考虑对时间建一棵线段树，离线算出每条边在哪些时间中存在，然后将其插入线段树的一个区间中
- 这样每条边只被插入 O(logn) 个节点中
- 然后考虑 DFS 这棵线段树，每到一个节点的时候，将所有这个节点的边所对应的 x 和 y 合并

线段树时间分治

- 可以发现这样每条边只被插入一次
- 但是我们这个结构必须支持撤回，也就是一个子树空了之后必须能删除这个子树
- 所以需要使用可撤销的并查集，复杂度为 O(logn)
- 每次记录下哪些内存位置被改成了什么，这样维护一个栈撤回，和之前讲过的不删除莫队一样
- 总复杂度 O((n+m)log^2n + qlogn)

树分块

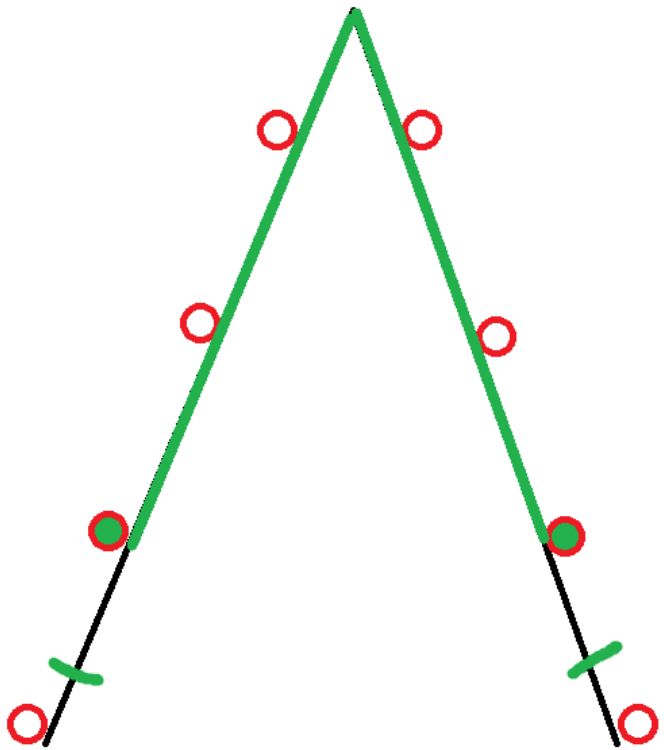
- 一般只需要查链信息
- 一般都是静态分块问题
- 在序列上很容易找关键点——每隔 $O(\sqrt{n})$ 个位置放一个即可
- 在树上如何找关键点?
- 随机撒 $O(\sqrt{n})$ 个点，期望复杂度是正确的

COT

- 树，点权
- 强制在线，查询链颜色数

Solution

- 进行树分块，预处理任意两个关键点之间有多少种颜色
- 每次查询的时候一直往上跑，找到最近的关键点，得到整块的答案



Solution

- 对于零散部分的贡献，即需要查询这个颜色是否在一条链上出现过
- 可以考虑使用 O(sqrtn) 修改， O(1) 查询的可持久化树上前缀和，其实就是一个可持久化数组
- 于是可以计算零散部分贡献，每次查询的时候零散部分共 O(sqrtn) 个点

- 这样时间复杂度 O(msqrtn)，空间复杂度 O(nsqrtn)，（空间复杂度其实是 O(n^(1+eps))）

Thanks for listening