树上问题

quarter

2019 年 1 月 14 日

Preface

貌似有专门的数据结构/计数/dp 专题，所以这几个方面在这个课件中并不是重点。

dfs 序

就是 dfs 先序遍历出来的序列。

dfs 序

就是 dfs 先序遍历出来的序列。

可以让一些树上问题下树:

- 子树对应 dfs 序上一个连续区间

dfs 序

就是 dfs 先序遍历出来的序列。

可以让一些树上问题下树：

- 子树对应 dfs 序上一个连续区间
- 判断两结点的从属关系

dfs 序

就是 dfs 先序遍历出来的序列。

可以让一些树上问题下树：

- 子树对应 dfs 序上一个连续区间
- 判断两结点的从属关系
- 按照一定的方法 dfs，可以让一定的链对应连续区间

dfs 序

就是 dfs 先序遍历出来的序列。

可以让一些树上问题下树:

- 子树对应 dfs 序上一个连续区间
- 判断两结点的从属关系
- 按照一定的方法 dfs,可以让一定的链对应连续区间
- 树形依赖背包问题

例题

一棵有根数树上有 N 个节点，1 号点是根，每条边都有一个距离。

Q 个询问：对于某个点 x，以 x 为根的子树上，所有与 x 距离 $\geq k$ 的点与 x 的距离之和。

$N, Q \leq 2 * 10^5$

例题

一棵有根数树上有 N 个节点，1 号点是根，每条边都有一个距离。

Q 个询问：对于某个点 x，以 x 为根的子树上，所有与 x 距离 $\geq k$ 的点与 x 的距离之和。

$N, Q \leq 2 * 10^5$

直接利用 dfs 序，将子树转化为区间。

那么这就是一个很正常的二维数点了。

Description

有一棵 n 个结点的树，每个点 i 有权值 w_i。

树的连通块是树上的一个点集，使得其中任意两个点都可以只经过这个点集中的点而互相到达。

定义联通块的权值为其中所有点的点权和。

求这棵树权值第 K 小的联通块。

$n, k \leq 10^5$，$|w_i| \leq 10^9$

Solution

考虑点分治，问题转化成强制包含根节点的权值 K 小联通块。

Solution

考虑点分治，问题转化成强制包含根节点的权值 K 小联通块。

考虑按 dfs 序转移，定义 pos_x 为 x 在这一颗点分树 dfs 序中的位置。

给每个点 x 建边:

- pos_x 向 $pos_x + 1$，边权 w_x，表示选择 x
- pos_x 向 $pos_x + size_x$，边权 0，表示不选择 x，则 x 的孩子也强制不能被选。

Solution

考虑点分治，问题转化成强制包含根节点的权值 K 小联通块。

考虑按 dfs 序转移，定义 pos_x 为 x 在这一颗点分树 dfs 序中的位置。

给每个点 x 建边:

- pos_x 向 $pos_x + 1$，边权 w_x，表示选择 x
- pos_x 向 $pos_x + size_x$，边权 0，表示不选择 x，则 x 的孩子也强制不能被选。

这样每个树上联通块都可以唯一对应图上的一条路径。

Solution

考虑点分治，问题转化成强制包含根节点的权值 K 小联通块。

考虑按 dfs 序转移，定义 pos_x 为 x 在这一颗点分树 dfs 序中的位置。

给每个点 x 建边:

- pos_x 向 $pos_x + 1$，边权 w_x，表示选择 x
- pos_x 向 $pos_x + size_x$，边权 0，表示不选择 x，则 x 的孩子也强制不能被选。

这样每个树上联通块都可以唯一对应图上的一条路径。

直接套用求 k 短路的经典算法求解即可。

欧拉序

欧拉序貌似有两种:

- dfs 过程中，每个点进栈时把自己加入序列，出栈时把其父亲加入序列（除根外，每个点出现度数次）。

欧拉序

欧拉序貌似有两种:

- dfs 过程中，每个点进栈时把自己加入序列，出栈时把其父亲加入序列（除根外，每个点出现度数次）。

  可以将求 lca 转化为 rmq 问题，使用 st 表解决。

欧拉序

欧拉序貌似有两种：

- dfs 过程中，每个点进栈时把自己加入序列，出栈时把其父亲加入序列（除根外，每个点出现度数次）。

  可以将求 lca 转化为 rmq 问题，使用 st 表解决。

- dfs 过程中，每个点在进、出栈时把自己加入序列（每个点恰好出现两次）。

  也叫括号序。

Description

有一棵点数为 N 的树，以点 1 为根，且有点权。

有 M 个操作，分为三种：

- 把某个节点 x 的点权增加 a
- 把某个节点 x 为根的子树中所有点的点权都增加 a
- 询问某个节点 x 到根的路径中所有点的点权和。

$n, m \leq 10^5$

Solution

显然可以通过重链剖分 dfs 序 + 线段树做到 $O(m \log^2 n)$。

Solution

显然可以通过重链剖分 dfs 序 + 线段树做到 $O(m\log^2 n)$。

然而，如果使用括号序，询问的答案为前缀内左括号权值和减去右括号权值和，复杂度降为 $O(m\log n)$。

树链剖分

把一棵树划分成若干条没有相交的链。

树链剖分

把一棵树划分成若干条没有相交的链。

- 把每个点与其子树大小最大的儿子划分到一起，形成重链。

树链剖分

把一棵树划分成若干条没有相交的链。

- 把每个点与其子树大小最大的儿子划分到一起，形成重链。
  - 每个点到根只会经过 $O(\log n)$ 次轻边。

树链剖分

把一棵树划分成若干条没有相交的链。

- 把每个点与其子树大小最大的儿子划分到一起，形成重链。
  - 每个点到根只会经过 $O(\log n)$ 次轻边。
- 把每个点与其子树高度最大的儿子划分到一起，形成长链。

树链剖分

把一棵树划分成若干条没有相交的链。

- 把每个点与其子树大小最大的儿子划分到一起，形成重链。
  - 每个点到根只会经过 $O(\log n)$ 次轻边。
- 把每个点与其子树高度最大的儿子划分到一起，形成长链。
  - 线性统计每个子树中以深度为下标的可合并信息。

树链剖分

把一棵树划分成若干条没有相交的链。

- 把每个点与其子树大小最大的儿子划分到一起，形成重链。
  - 每个点到根只会经过 $O(\log n)$ 次轻边。
- 把每个点与其子树高度最大的儿子划分到一起，形成长链。
  - 线性统计每个子树中以深度为下标的可合并信息。
  - $O(n \log n)$ 预处理，$O(1)$ 询问任意点 x 的任意深度 d 的祖先：每条长链预处理出顶点向上链长个祖先；先用倍增数组跳到 $\max_{2^k \leq d}\{2^k\}$ 次祖先 y，然后利用 y 所在长链的信息，向上或向下寻找 d 深度祖先。

Description

给出一棵 n 个结点的树，无边权。

求无序三元组 (x, y, z) 的个数，满足 $dis(x, y) = dis(x, z) = dis(y, z)$。

$n \leq 10^5$

Solution

记：

- $f(i,j) = \sum_{x \in subtree(i)} [dis(i,x) = j]$
- $g(i,j) = \sum_{lca(x,y) \in subtree(i)} [dis(x,lca) = dis(y,lca) = dis(i,lca) + j]$

Solution

记:

- $f(i, j) = \sum_{x \in subtree(i)} [dis(i, x) = j]$
- $g(i, j) = \sum_{lca(x,y) \in subtree(i)} [dis(x, lca) = dis(y, lca) = dis(i, lca) + j]$

每次将一个儿子 y 转移到 x:

- $ans+ = \sum f(y, i) * g(x, i+1) + g(y, i) * f(x, i-1)$
- $f(x, i+1)+ = f(y, i)$
- $g(x, i-1)+ = g(y, i)$
- $g(x, i+1)+ = f(x, i+1) * f(y, i)$

Solution

记:

- $f(i,j) = \sum_{x \in subtree(i)} [dis(i,x) = j]$
- $g(i,j) = \sum_{lca(x,y) \in subtree(i)} [dis(x,lca) = dis(y,lca) = dis(i,lca) + j]$

每次将一个儿子 y 转移到 x:

- $ans+ = \sum f(y,i) * g(x,i+1) + g(y,i) * f(x,i-1)$
- $f(x,i+1)+ = f(y,i)$
- $g(x,i-1)+ = g(y,i)$
- $g(x,i+1)+ = f(x,i+1) * f(y,i)$

长链剖分，链最长的儿子可以用指针操作一下优化掉。

$O(n)$

Description

一棵 n 个点的树，每个点有点权 A_i 和 B_i，找一条长度为 m 的路径，设路径点集为 S，使得 $\frac{\sum_{i \in S} A_i}{\sum_{i \in S} B_i}$ 最小。

$n \leq 2 * 10^5$

Solution

很明显的 01 分数规划，二分答案，每个点的权值转为 $A_i - B_i * ans$，找到一条权值 < 0、长度为 m 的路径。

Solution

很明显的 01 分数规划，二分答案，每个点的权值转为 $A_i - B_i * ans$，找到一条权值 < 0、长度为 m 的路径。

一个直接的想法是点分治，但实际上用长链剖分可以做到线性判断答案。

Description

一棵 n 个结点的有根树。

定义 $dep(x) = x$ 到根结点的简单路径上的点数。

定义 $f(x, y) = \sum_{p \in subtree(y), p != y} [dep(p) \leq dep(x)]$。

定义 $g(x) = \sum_{x \in subtree(p), p != x} f(x, p)$。

对于所有 $i \in [1, n]$，求 $g(i)$。

$n \leq 5 * 10^5$

Solution

首先，考虑每个 p 对 $g(i)$ 个贡献，不难推导出，
$g(i) = \sum_{dep(p) \leq dep(i), p \neq i} dep(lca(p, i))$。

Solution

首先，考虑每个 p 对 $g(i)$ 个贡献，不难推导出，
$g(i) = \sum_{dep(p) \leq dep(i), p \neq i} dep(lca(p, i))$。

然后，$g(i) = g(fa(i)) + dep(fa(i)) + \sum_{dep(p) = dep(i), p \neq i} dep(lca(p, i))$

Solution

首先，考虑每个 p 对 $g(i)$ 个贡献，不难推导出，
$g(i) = \sum_{dep(p) \le dep(i), p \ne i} dep(lca(p,i))$。

然后，$g(i) = g(fa(i)) + dep(fa(i)) + \sum_{dep(p)=dep(i), p \ne i} dep(lca(p,i))$

所以，问题转化为求出 $h(i) = \sum_{dep(p)=dep(i), p \ne i} dep(lca(p,i))$

Solution

首先，考虑每个 p 对 $g(i)$ 个贡献，不难推导出，
$g(i) = \sum_{dep(p) \leq dep(i), p \neq i} dep(lca(p, i))$。

然后，$g(i) = g(fa(i)) + dep(fa(i)) + \sum_{dep(p)=dep(i), p \neq i} dep(lca(p, i))$

所以，问题转化为求出 $h(i) = \sum_{dep(p)=dep(i), p \neq i} dep(lca(p, i))$

长链剖分，两个集合 A、B 在 lca 合并时，$\forall a \in A$，$h(a)$ 加上 $size(B) * dep(lca)$，B 集合类似。

Solution

首先，考虑每个 p 对 $g(i)$ 个贡献，不难推导出，
$g(i) = \sum_{dep(p) \le dep(i), p \ne i} dep(lca(p, i))$。

然后，$g(i) = g(fa(i)) + dep(fa(i)) + \sum_{dep(p) = dep(i), p \ne i} dep(lca(p, i))$

所以，问题转化为求出 $h(i) = \sum_{dep(p) = dep(i), p \ne i} dep(lca(p, i))$

长链剖分，两个集合 A、B 在 lca 合并时，$\forall a \in A$，$h(a)$ 加上 $size(B) * dep(lca)$，B 集合类似。

考虑重构树，每个集合用一棵树存储，集合合并时，B 的根 b 向 A 的根 a 连一条权值为 $w_{b,a} = h(b) - h(a)$ 的边。集合合并后，$h(b) - h(a) = w_{b,a}$ 不再变化，最终的 $h(b) = h(a) + w_{b,a}$。

Solution

首先，考虑每个 p 对 $g(i)$ 个贡献，不难推导出，
$g(i) = \sum_{dep(p)\le dep(i), p\ne i} dep(lca(p,i))$。

然后，$g(i) = g(fa(i)) + dep(fa(i)) + \sum_{dep(p)=dep(i), p\ne i} dep(lca(p,i))$

所以，问题转化为求出 $h(i) = \sum_{dep(p)=dep(i), p\ne i} dep(lca(p,i))$

长链剖分，两个集合 A、B 在 lca 合并时，$\forall a \in A$，$h(a)$ 加上 $size(B) * dep(lca)$，B 集合类似。

考虑重构树，每个集合用一棵树存储，集合合并时，B 的根 b 向 A 的根 a 连一条权值为 $w_{b,a} = h(b) - h(a)$ 的边。集合合并后，$h(b) - h(a) = w_{b,a}$ 不再变化，最终的 $h(b) = h(a) + w_{b,a}$。

$O(n)$

树的直径

树上最长的简单路径。

树的直径

树上最长的简单路径。

- 找到任意点 u 的最远点 v，再找到 v 的最远点 w，那么 (v, w) 是一条直径。

树的直径

树上最长的简单路径。

- 找到任意点 u 的最远点 v，再找到 v 的最远点 w，那么 (v, w) 是一条直径。
- 在不保证权值非负的情况下，可以用 dp 找直径。

树的直径

树上最长的简单路径。

- 找到任意点 u 的最远点 v，再找到 v 的最远点 w，那么 (v, w) 是一条直径。
- 在不保证权值非负的情况下，可以用 dp 找直径。
- 线段树维护直径：同一棵树上，两个点集 A, B 的最远点对为 $(u_A, v_A), (u_B, v_B)$，那么 $A \cup B$ 的最远点对 $\in \{u_A, v_A, u_B, v_B\}$。那么，对于线段树上每一个 dfs 序区间，维护区间对应点集的最远点对，可以 $O(\log n)$ 地查询子树（或者其他可以用若干个 dfs 序区间表示出的点集）的直径。

树的直径

树上最长的简单路径。

- 找到任意点 u 的最远点 v，再找到 v 的最远点 w，那么 (v, w) 是一条直径。
- 在不保证权值非负的情况下，可以用 dp 找直径。
- 线段树维护直径：同一棵树上，两个点集 A, B 的最远点对为 $(u_A, v_A), (u_B, v_B)$，那么 $A \cup B$ 的最远点对 $\in \{u_A, v_A, u_B, v_B\}$。那么，对于线段树上每一个 dfs 序区间，维护区间对应点集的最远点对，可以 $O(\log n)$ 地查询子树（或者其他可以用若干个 dfs 序区间表示出的点集）的直径。

51NOD 1766: 树上的最远点对

n 个点的一棵树，q 个询问。

每个询问给出 $[a,b],[c,d]$ 两个区间，求 $\max\{dis(i,j)|a \le i \le b, c \le j \le d\}$。

$n,m \le 10^5$

51NOD 1766: 树上的最远点对

n 个点的一棵树，q 个询问。

每个询问给出 $[a, b], [c, d]$ 两个区间，求 $\max\{dis(i, j) | a \le i \le b, c \le j \le d\}$。

$n, m \le 10^5$

直接线段树维护直径。

Description

一棵 n 个黑白结点的树，有边权。这棵黑白树中有 m 个黑点，其它都是白点。

对于一个黑点我们定义他的好朋友为离他最远的黑点。如果有多个黑点离它最远那么都是它的好朋友。两点间的距离定义为两点之间的最短路的长度。

现在你要摧毁一个白点。

摧毁后有一些黑点会不高兴。一个黑点不高兴当且仅当他不能到达任何一个在摧毁那个白点前的好朋友。

最大化不高兴的黑点数。

$m < n \leq 10^5$。

Solution

一棵树的直径不一定唯一，但是所有直径的中点是确定的，即树的中心。

Solution

一棵树的直径不一定唯一，但是所有直径的中点是确定的，即树的中心。

一个点到其最远点必定经过中心。

Solution

一棵树的直径不一定唯一，但是所有直径的中点是确定的，即树的中心。

一个点到其最远点必定经过中心。

我们将黑点构成的虚树的中心（如果中心在一条边上，就任选中心所在边的一个端点）作为根结点，那么一个黑点到最远黑点必然经过根。

Solution

一棵树的直径不一定唯一，但是所有直径的中点是确定的，即树的中心。

一个点到其最远点必定经过中心。

我们将黑点构成的虚树的中心（如果中心在一条边上，就任选中心所在边的一个端点）作为根结点，那么一个黑点到最远黑点必然经过根。

枚举摧毁的白点 w，考虑每个黑点 b:

- b 在 w 子树内，w 一定会截断 b 到最远黑点的路径。

Solution

一棵树的直径不一定唯一，但是所有直径的中点是确定的，即树的中心。

一个点到其最远点必定经过中心。

我们将黑点构成的虚树的中心（如果中心在一条边上，就任选中心所在边的一个端点）作为根结点，那么一个黑点到最远黑点必然经过根。

枚举摧毁的白点 w，考虑每个黑点 b:

- b 在 w 子树内，w 一定会截断 b 到最远黑点的路径。
- b 不在 w 子树内，但为根的同一儿子的后代，w 必然不在路径上。

Solution

一棵树的直径不一定唯一，但是所有直径的中点是确定的，即树的中心。

一个点到其最远点必定经过中心。

我们将黑点构成的虚树的中心（如果中心在一条边上，就任选中心所在边的一个端点）作为根结点，那么一个黑点到最远黑点必然经过根。

枚举摧毁的白点 w，考虑每个黑点 b:

- b 在 w 子树内，w 一定会截断 b 到最远黑点的路径。
- b 不在 w 子树内，但为根的同一儿子的后代，w 必然不在路径上。
- w、b 为根的不同儿子的后代，判断 w 的子树是否包括了所有离 b 最远的黑点即可，记录每个子树内最深深度、最深点个数。

Solution

一棵树的直径不一定唯一，但是所有直径的中点是确定的，即树的中心。

一个点到其最远点必定经过中心。

我们将黑点构成的虚树的中心（如果中心在一条边上，就任选中心所在边的一个端点）作为根结点，那么一个黑点到最远黑点必然经过根。

枚举摧毁的白点 w，考虑每个黑点 b:

- b 在 w 子树内，w 一定会截断 b 到最远黑点的路径。
- b 不在 w 子树内，但为根的同一儿子的后代，w 必然不在路径上。
- w、b 为根的不同儿子的后代，判断 w 的子树是否包括了所有离 b 最远的黑点即可，记录每个子树内最深深度、最深点个数。

$O(n)$

树的重心

一棵树删去其重心，剩下最大的连通块的大小最小。

树的重心

一棵树删去其重心，剩下最大的连通块的大小最小。

- 一棵大小为 n 的树删去重心，剩下的连通块大小 $\leq \frac{1}{2}n$。

树的重心

一棵树删去其重心，剩下最大的连通块的大小最小。

- 一棵大小为 n 的树删去重心，剩下的连通块大小 $\leq \frac{1}{2}n$。
- 一棵树有 $1 \sim 2$ 个重心，如果有两个重心，两个重心相邻。

树的重心

一棵树删去其重心，剩下最大的连通块的大小最小。

- 一棵大小为 n 的树删去重心，剩下的连通块大小 $\leq \frac{1}{2}n$。
- 一棵树有 $1 \sim 2$ 个重心，如果有两个重心，两个重心相邻。
- 重心是树中到所有点距离和最小的点。

树的重心

一棵树删去其重心，剩下最大的连通块的大小最小。

- 一棵大小为 n 的树删去重心，剩下的连通块大小 $\leq \frac{1}{2}n$。
- 一棵树有 1 ∼ 2 个重心，如果有两个重心，两个重心相邻。
- 重心是树中到所有点距离和最小的点。
- 把两棵树用一条边连接，新树的重心在原来两树重心的路径上。

Description

给出一棵 n 个结点的有根树。

求每个子树的重心。

$n \leq 3 * 10^5$

Solution

每个结点 u 记录一个 set，保存子树内所有结点，按照子树大小 $size$ 排序，$size_v \geq \lceil \frac{size_u}{2} \rceil$ 的、$size_v$ 最小的结点 v 即为重心（之一）。

启发式合并，$O(n \log^2 n)$。

Solution

每个结点 u 记录一个 set，保存子树内所有结点，按照子树大小 $size$ 排序，$size_v \geq \lceil \frac{size_u}{2} \rceil$ 的、$size_v$ 最小的结点 v 即为重心（之一）。

启发式合并，$O(n \log^2 n)$。

一个点 u 的某个子树的大小若大于 $\frac{1}{2} size_u$，那么重心必定在这个子树内，否则重心为 u 本身。

Solution

每个结点 u 记录一个 set，保存子树内所有结点，按照子树大小 $size$ 排序，$size_v \geq \lceil \frac{size_u}{2} \rceil$ 的、$size_v$ 最小的结点 v 即为重心（之一）。

启发式合并，$O(n \log^2 n)$。

一个点 u 的某个子树的大小若大于 $\frac{1}{2} size_u$，那么重心必定在这个子树内，否则重心为 u 本身。

从这个子树的重心开始向父亲枚举，直到找到重心。

Solution

每个结点 u 记录一个 set，保存子树内所有结点，按照子树大小 $size$ 排序，$size_v \geq \lceil \frac{size_u}{2} \rceil$ 的、$size_v$ 最小的结点 v 即为重心（之一）。

启发式合并，$O(n \log^2 n)$。

一个点 u 的某个子树的大小若大于 $\frac{1}{2} size_u$，那么重心必定在这个子树内，否则重心为 u 本身。

从这个子树的重心开始向父亲枚举，直到找到重心。

$O(n)$

树分治

- 点分治：

  找到一棵树的重心，处理掉穿过重心的路径/连通块，然后递归处理各个子树。

  每个点/边只会出现 $O(\log n)$ 次。

树分治

- 点分治：

  找到一棵树的重心，处理掉穿过重心的路径/连通块，然后递归处理各个子树。

  每个点/边只会出现 $O(\log n)$ 次。

- 边分治：

  每次找到一条边，使得两侧的点数尽可能均匀，然后处理掉经过这条边的路径，并递归。

树分治

- 点分治：

  找到一棵树的重心，处理掉穿过重心的路径/连通块，然后递归处理各个子树。

  每个点/边只会出现 $O(\log n)$ 次。

- 边分治：

  每次找到一条边，使得两侧的点数尽可能均匀，然后处理掉经过这条边的路径，并递归。

  如果一个点的度数太大，可以通过添加新点来转为二叉树，保证复杂度。

Description

有一棵 n 个结点的树。

树的每条边有两个权值 a_i、b_i，经过第 i 条边花费时间 $a_i * t + b_i$。

给定 m，求 $t = 0, 1, 2, ..., m-1$ 时，最长的路径长度。

$n \leq 10^5$，$m \leq 10^6$。

Solution

可以通过增加 $a_i = b_i = 0$ 的边来转为二叉树，然后边分治。

Solution

可以通过增加 $a_i = b_i = 0$ 的边来转为二叉树，然后边分治。

对于每次分治，找出两侧所有 $a * t + b$，并求出凸壳，然后求它们的闵科夫斯基和。

$O(n \log^2 n)$

Description

一个 n 个结点的树，有边权。

给出 m 个二元组 (a_i, b_i)。

求 $\max_{i,j\in[1,m]}\{dis(a_i, a_j) + dis(b_i, b_j)\}$。

$n, m \leq 10^5$

Solution

先把点分树搞出来。

Solution

先把点分树搞出来。

对于每个分治重心 x，考虑 (a_i, a_j) 为分治块内跨过 x 的路径的二元组对。

Solution

先把点分树搞出来。

对于每个分治重心 x，考虑 (a_i, a_j) 为分治块内跨过 x 的路径的二元组对。

一个二元组 (a_i, b_i)，枚举 (b_i, b_j) 可能的跨过的分治重心 y（也就是 b_i 在点分树上的祖先），将 $dis(a_i, x) + dis(b_i, y)$ 的信息挂在 y 上，并利用 y 上已有信息更新答案。

Solution

先把点分树搞出来。

对于每个分治重心 x，考虑 (a_i, a_j) 为分治块内跨过 x 的路径的二元组对。

一个二元组 (a_i, b_i)，枚举 (b_i, b_j) 可能的跨过的分治重心 y（也就是 b_i 在点分树上的祖先），将 $dis(a_i, x) + dis(b_i, y)$ 的信息挂在 y 上，并利用 y 上已有信息更新答案。

为了避免 (a_i, x) 和 (a_j, x) 来自同一子树，我们对于每个子树内所有 a_i 先全部完成询问，再插入信息。

Solution

先把点分树搞出来。

对于每个分治重心 x，考虑 (a_i, a_j) 为分治块内跨过 x 的路径的二元组对。

一个二元组 (a_i, b_i)，枚举 (b_i, b_j) 可能的跨过的分治重心 y（也就是 b_i 在点分树上的祖先），将 $dis(a_i, x) + dis(b_i, y)$ 的信息挂在 y 上，并利用 y 上已有信息更新答案。

为了避免 (a_i, x) 和 (a_j, x) 来自同一子树，我们对于每个子树内所有 a_i 先全部完成询问，再插入信息。

而防止 (b_i, y) 和 (b_j, y) 来自同一子树，可以在每个分治重心记录最大信息、最大信息的来源子树、以及不来自该子树的次大信息。

Solution

先把点分树搞出来。

对于每个分治重心 x，考虑 (a_i, a_j) 为分治块内跨过 x 的路径的二元组对。

一个二元组 (a_i, b_i)，枚举 (b_i, b_j) 可能的跨过的分治重心 y（也就是 b_i 在点分树上的祖先），将 $dis(a_i, x) + dis(b_i, y)$ 的信息挂在 y 上，并利用 y 上已有信息更新答案。

为了避免 (a_i, x) 和 (a_j, x) 来自同一子树，我们对于每个子树内所有 a_i 先全部完成询问，再插入信息。

而防止 (b_i, y) 和 (b_j, y) 来自同一子树，可以在每个分治重心记录最大信息、最大信息的来源子树、以及不来自该子树的次大信息。

$O(m * \log^2 n + n * \log n)$

Prufer 序列

Prufer 序列是有标号无根树的一种编码，Prufer 序列和有标号无根树一一对应。

Prufer 序列

Prufer 序列是有标号无根树的一种编码，Prufer 序列和有标号无根树一一对应。

- 一个 n 个结点的无根树，对应一个长度为 $n-2$、所有元素均为 $[1,n]$ 内整数的序列，这个序列叫 Prufer 序列。

Prufer 序列

Prufer 序列是有标号无根树的一种编码，Prufer 序列和有标号无根树一一对应。

- 一个 n 个结点的无根树，对应一个长度为 $n-2$、所有元素均为 $[1,n]$ 内整数的序列，这个序列叫 Prufer 序列。
- 无根树 $\Rightarrow$Prufer 序列：删除编号最小的叶子，将其邻点编号加入数列，持续这个过程直到图中只剩 2 个点。

Prufer 序列

Prufer 序列是有标号无根树的一种编码，Prufer 序列和有标号无根树一一对应。

- 一个 n 个结点的无根树，对应一个长度为 $n-2$、所有元素均为 $[1,n]$ 内整数的序列，这个序列叫 Prufer 序列。
- 无根树 $\Rightarrow$Prufer 序列：删除编号最小的叶子，将其邻点编号加入数列，持续这个过程直到图中只剩 2 个点。
- Prufer 序列 $\Rightarrow$ 无根树：建立一个集合 $\{1,2,...,n\}$，找出集合中最小的、未出现在 Prufer 序列中的元素，将其与序列首项连边，并删去这个元素和序列首项，持续这个过程直到序列为空，然后把集合中最后两个数连边。

Prufer 序列

Prufer 序列是有标号无根树的一种编码，Prufer 序列和有标号无根树一一对应。

- 一个 n 个结点的无根树，对应一个长度为 $n-2$、所有元素均为 $[1,n]$ 内整数的序列，这个序列叫 Prufer 序列。
- 无根树 $\Rightarrow$Prufer 序列：删除编号最小的叶子，将其邻点编号加入数列，持续这个过程直到图中只剩 2 个点。
- Prufer 序列 $\Rightarrow$ 无根树：建立一个集合 $\{1,2,...,n\}$，找出集合中最小的、未出现在 Prufer 序列中的元素，将其与序列首项连边，并删去这个元素和序列首项，持续这个过程直到序列为空，然后把集合中最后两个数连边。
- 点数为 n 的无根树个数 $=$ 长度为 $n-2$ 的 Prufer 序列个数 $= n^{n-2}$

Prufer 序列

Prufer 序列是有标号无根树的一种编码，Prufer 序列和有标号无根树一一对应。

- 一个 n 个结点的无根树，对应一个长度为 $n-2$、所有元素均为 $[1,n]$ 内整数的序列，这个序列叫 Prufer 序列。
- 无根树 $\Rightarrow$Prufer 序列：删除编号最小的叶子，将其邻点编号加入数列，持续这个过程直到图中只剩 2 个点。
- Prufer 序列 $\Rightarrow$ 无根树：建立一个集合 $\{1,2,...,n\}$，找出集合中最小的、未出现在 Prufer 序列中的元素，将其与序列首项连边，并删去这个元素和序列首项，持续这个过程直到序列为空，然后把集合中最后两个数连边。
- 点数为 n 的无根树个数 $=$ 长度为 $n-2$ 的 Prufer 序列个数 $= n^{n-2}$
- 无根树中一个点的度数 $=$ 点的编号在 Prufer 序列中出现次数 $+1$。

Description

一棵 n 个点的有根树，要在每个点 i 处维护该点的石子数 c_i，并执行 m 个操作，支持：

- 修改某个点处的石子数。
- 给出一个点 x 与一个距离 L，求 x 子树内每个小于 L 的深度的点权和的异或和。

$n, m \le 10^5$，时限 $4s$

Solution

注意到和的异或很难合并，只能单次修改。考虑离线，每 $\sqrt{m}$ 次操作后重建。

Solution

注意到和的异或很难合并，只能单次修改。考虑离线，每 $\sqrt{m}$ 次操作后重建。

长链剖分，启发式合并。

Solution

注意到和的异或很难合并，只能单次修改。考虑离线，每 $\sqrt{m}$ 次操作后重建。

长链剖分，启发式合并。

现需要处理 $O(\sqrt{m})$ 次区间询问异或和，$O(m)$ 次单点修改，可以用分块维护。

$O(m\sqrt{m})$。

Description

交互库有一棵有根树，有 n 个叶子，叶子从 1 到 n 编号，非叶子结点都有恰好 2 个儿子。

每次向交互库询问 (a, b, c)：交互库返回 $lca(a, b), lca(a, c), lca(b, c)$ 中哪一个深度最大。

询问不超过 $10 * n$ 次，构建出一棵同构的树。

$n \leq 10^3$

Solution

每个非叶子结点 u 可以用左右子树内各一个叶子结点 x, y 表示，询问 (x, y, z) 即可知道 z 在 u 的左子树中/右子树中/子树外。

Solution

每个非叶子结点 u 可以用左右子树内各一个叶子结点 x, y 表示，询问 (x, y, z) 即可知道 z 在 u 的左子树中/右子树中/子树外。

考虑将所有叶子依次加入树中，每次需要在 $O(\log n)$ 次询问内完成加入。

Solution

每个非叶子结点 u 可以用左右子树内各一个叶子结点 x, y 表示，询问 (x, y, z) 即可知道 z 在 u 的左子树中/右子树中/子树外。

考虑将所有叶子依次加入树中，每次需要在 $O(\log n)$ 次询问内完成加入。

将重心改为删去其后叶子数最大的连通块叶子数最小的点，点分治即可。

Description

有一棵 n 个结点的树，边权均为 1。

计算有多少个 $\{1, 2, ..., n\}$ 的排列 $\{p_1, p_2, ..., p_n\}$，使得 $\sum_{i=1}^{n} dist(i, p_i)$ 最大化。

$n \leq 5 * 10^3$

Solution

考虑每条边 e 分别的贡献，若 e 两侧点数为 x_e、y_e，那么贡献的上界为 $2\min\{x_e, y_e\}$。

Solution

考虑每条边 e 分别的贡献，若 e 两侧点数为 x_e、y_e，那么贡献的上界为 $2\min\{x_e, y_e\}$。

不难发现 $\sum_e 2\min\{x_e, y_e\}$ 总是可以达到的。

Solution

考虑每条边 e 分别的贡献，若 e 两侧点数为 x_e、y_e，那么贡献的上界为 $2\min\{x_e, y_e\}$。

不难发现 $\sum_e 2\min\{x_e, y_e\}$ 总是可以达到的。

讨论两种情况：

- 如果这个树有两个重心，只需保证所有 (i, p_i) 都穿过两重心之间的边即可，方案为 $\frac{n}{2}! * \frac{n}{2}!$。

Solution

考虑每条边 e 分别的贡献，若 e 两侧点数为 x_e、y_e，那么贡献的上界为 $2\min\{x_e, y_e\}$。

不难发现 $\sum_e 2\min\{x_e, y_e\}$ 总是可以达到的。

讨论两种情况：

- 如果这个树有两个重心，只需保证所有 (i, p_i) 都穿过两重心之间的边即可，方案为 $\frac{n}{2}! * \frac{n}{2}!$。
- 如果只有一个重心，那么需要使得所有路径均经过重心。f_S 表示 $x \in S$ 的 (x, p_x) 未经过重心的排列数，那么答案为 $\sum(-1)^{|S|} f_S$，而 f_S 的计算只需一个 $O(n^2)$ 的 dp。

Description

有一棵 n 个结点的树，边权均为 1。

选择 k 个结点 $\{x_1, x_2, ..., x_k\}$。

每个结点 p 可以表示为一个 k 维向量 $\{dist(p, x_1), dist(p, x_2), ..., dist(p, x_k)\}$。

最小化 k，使得每个 k 维向量互不相同。

$2 \leq n \leq 10^5$

Solution

如果两个点 u, v 满足 $dist(u, v)$ 为奇数，那么它们的向量在任意一维上均不同。

Solution

如果两个点 u, v 满足 $dist(u, v)$ 为奇数，那么它们的向量在任意一维上均不同。

否则，找到 $path(u, v)$ 的中点 mid，u, v 能区分开，当且仅当：以 mid 为根，存在至少一个 x_i，与 u 或 v 属于同一个 mid 孩子的后代。

Solution

如果两个点 u, v 满足 $dist(u, v)$ 为奇数，那么它们的向量在任意一维上均不同。

否则，找到 $path(u, v)$ 的中点 mid，u, v 能区分开，当且仅当：以 mid 为根，存在至少一个 x_i，与 u 或 v 属于同一个 mid 孩子的后代。相当于：对于任意一个点，删去它，那么至多有 1 个连通块内不存在 x_i。

Solution

如果两个点 u, v 满足 $dist(u, v)$ 为奇数，那么它们的向量在任意一维上均不同。

否则，找到 $path(u, v)$ 的中点 mid，u, v 能区分开，当且仅当：以 mid 为根，存在至少一个 x_i，与 u 或 v 属于同一个 mid 孩子的后代。相当于：对于任意一个点，删去它，那么至多有 1 个连通块内不存在 x_i。

如果树为一条链，那么答案为 1（选择一个叶子）。

Solution

如果两个点 u, v 满足 $dist(u, v)$ 为奇数，那么它们的向量在任意一维上均不同。

否则，找到 $path(u, v)$ 的中点 mid，u, v 能区分开，当且仅当：以 mid 为根，存在至少一个 x_i，与 u 或 v 属于同一个 mid 孩子的后代。相当于：对于任意一个点，删去它，那么至多有 1 个连通块内不存在 x_i。

如果树为一条链，那么答案为 1（选择一个叶子）。否则找到一个度数 ≥ 3 的点，将其作为根，那么条件转化为：对于任意一个点，如果它有 a 个儿子，那么至少有 $a-1$ 个子树内有存在 x_i。

Solution

如果两个点 u, v 满足 $dist(u, v)$ 为奇数，那么它们的向量在任意一维上均不同。

否则，找到 $path(u, v)$ 的中点 mid，u, v 能区分开，当且仅当：以 mid 为根，存在至少一个 x_i，与 u 或 v 属于同一个 mid 孩子的后代。相当于：对于任意一个点，删去它，那么至多有 1 个连通块内不存在 x_i。

如果树为一条链，那么答案为 1（选择一个叶子）。否则找到一个度数 ≥ 3 的点，将其作为根，那么条件转化为：对于任意一个点，如果它有 a 个儿子，那么至少有 $a-1$ 个子树内有存在 x_i。

然后随便贪心下，$O(n)$。

Description

给出一棵 n 个结点、有标号的树，边全为蓝色。

每次可以选择一条全为蓝色的路径，删去其中一条边，并在路径两端连上一条红色边。

再给出一个边全为红色的树，判断蓝色树是否可以操作成红色树。

$n \le 10^5$

Solution

不妨反过来考虑最后一条加入的红边：添加它之前，树中只有一条蓝色边了，那么这两条边连接的点必定相同。

Solution

不妨反过来考虑最后一条加入的红边：添加它之前，树中只有一条蓝色边了，那么这两条边连接的点必定相同。

所以，两个树中如果不存在这样相同的边，那么无解。

Solution

不妨反过来考虑最后一条加入的红边：添加它之前，树中只有一条蓝色边了，那么这两条边连接的点必定相同。

所以，两个树中如果不存在这样相同的边，那么无解。否则，我们显然可以将这条边连接的点缩起来，并重复这个过程直到只剩下一个点。

Solution

不妨反过来考虑最后一条加入的红边：添加它之前，树中只有一条蓝色边了，那么这两条边连接的点必定相同。

所以，两个树中如果不存在这样相同的边，那么无解。否则，我们显然可以将这条边连接的点缩起来，并重复这个过程直到只剩下一个点。

至于实现方法。可以用 set 记录每个点连接的边，当两点合并时，将较小 set 启发式合并进较大 set；而寻找相同的边直接用个 map，当某条边出现次数达到了 2，将其压入一个队列。

$O(n * \log^2 n)$。

Description

一棵 n 个结点的树。

结点 r 作为扩展起点，每次从未被扩展、且与至少一个已被扩展的结点相邻的结点中，选择编号最小的扩展。

对于每个 $r \in (1, n]$，求出结点 1 是第几个被扩展的。

$n \leq 2 * 10^5$。

Solution

考虑每个 x，在何种情况下会对 y 产生贡献。

Solution

考虑每个 x，在何种情况下会对 y 产生贡献。

首先 y 先扩展到 $z = lca(x, y)$。

如果 $\max_{i \in path(1, fa_z)}\{i\} > \max_{i \in path(x,z), i \neq z}\{i\}$，那么 y 会扩展到 x，否则会先扩展到 1。

Solution

考虑每个 x，在何种情况下会对 y 产生贡献。

首先 y 先扩展到 $z = lca(x, y)$。

如果 $\max_{i \in path(1, fa_z)}\{i\} > \max_{i \in path(x,z), i \neq z}\{i\}$，那么 y 会扩展到 x，否则会先扩展到 1。

不难发现，对于每个 x，我们只需找出满足上述条件、深度最低的 z，那么 $subtree_z$ 即为所有被贡献的 y 的集合。

用一个栈稍微维护下即可，$O(n)$。

Description

有一棵 n 个点的有根树，每个点有权值 w_i，初始每个结点上都没有石子。

Snuke 准备了一些石子，并把它们拿在手中。可以进行以下两种操作任意多次：

- 从手中取 w_i 个石子放在结点 i 上，进行该操作要求结点 i 的所有孩子 j 上都有 w_j 个石子。
- 将结点 i 上的所有石子收回手中。

对于每个 i，为了在结点 i 上放 w_i 个石子，Snuke 至少需要准备多少石子。

$n \le 2 * 10^5$，$1 \le w_i \le 10^9$

Solution

考虑倒着做：每次在 i 上放 w_i 个石子，或在 i 所有孩子都有石子时取走 i 的石子；初始时树中一个点上有石子，目标是清空树上所有石子，最小化历史最大总石子数。

Solution

考虑倒着做：每次在 i 上放 w_i 个石子，或在 i 所有孩子都有石子时取走 i 的石子；初始时树中一个点上有石子，目标是清空树上所有石子，最小化历史最大总石子数。

首先，一个点 i 的所有孩子 j 应该同时放上石子，接着立刻取走 i 上石子。

Solution

考虑倒着做：每次在 i 上放 w_i 个石子，或在 i 所有孩子都有石子时取走 i 的石子；初始时树中一个点上有石子，目标是清空树上所有石子，最小化历史最大总石子数。

首先，一个点 i 的所有孩子 j 应该同时放上石子，接着立刻取走 i 上石子。

将这个过程描述为二元组 $(-w_i + \sum_j w_j, \sum_j w_j)$，表示这个过程结束后石子的增量，过程中石子的历史最大值与过程开始时石子数的差。

Solution

考虑倒着做：每次在 i 上放 w_i 个石子，或在 i 所有孩子都有石子时取走 i 的石子；初始时树中一个点上有石子，目标是清空树上所有石子，最小化历史最大总石子数。

首先，一个点 i 的所有孩子 j 应该同时放上石子，接着立刻取走 i 上石子。

将这个过程描述为二元组 $(-w_i + \sum_j w_j, \sum_j w_j)$，表示这个过程结束后石子的增量，过程中石子的历史最大值与过程开始时石子数的差。

定义 (x, y) 的优先级：

- $x < 0$ 优先于 $x \geq 0$。
- 同时满足 $x < 0$ 时，y 小的优先。
- 同时满足 $x \geq 0$ 时，$y - x$ 大的优先。

Solution

我们找出最优的一个过程。如果他的父亲过程已完成，则立刻执行这个过程。否则在其父亲过程完成后立刻执行它，即：将这两个二元组合并，
$(a, b) + (c, d) = (a + c, \max(b, a + d))$。

Solution

我们找出最优的一个过程。如果他的父亲过程已完成，则立刻执行这个过程。否则在其父亲过程完成后立刻执行它，即：将这两个二元组合并，$(a, b) + (c, d) = (a + c, \max(b, a + d))$。

寻找最优二元组用优先队列维护即可。

Solution

我们找出最优的一个过程。如果他的父亲过程已完成，则立刻执行这个过程。否则在其父亲过程完成后立刻执行它，即：将这两个二元组合并，$(a, b) + (c, d) = (a + c, \max(b, a + d))$。

寻找最优二元组用优先队列维护即可。

不难发现，每个子树的选择顺序是全局的子序列，那么只需做一遍全局贪心，用数据结构维护答案。

Solution

我们找出最优的一个过程。如果他的父亲过程已完成，则立刻执行这个过程。否则在其父亲过程完成后立刻执行它，即：将这两个二元组合并，$(a, b) + (c, d) = (a + c, \max(b, a + d))$。

寻找最优二元组用优先队列维护即可。

不难发现，每个子树的选择顺序是全局的子序列，那么只需做一遍全局贪心，用数据结构维护答案。

$O(n \log^2 n)$

讲完了。

~~diaoye 建模预警~~