#1
#2
#3
#4
#5
#6
#7

树上的题

demerzel

2018 年 9 月 27 日

题目描述

给定一棵 n 个节点的树，你可以进行 $n-1$ 次操作，每次操作步骤如下：

- 选择 u, v 两个度数为 1 的节点。
- 将 u, v 之间的距离加到 ans 上。
- 将 u 从树上删除。

求一个操作序列使得 ans 最大。

题解

- 先把直径拿出来，将直径外的点一个一个的和直径中的某一个端点配对并删掉。最后再将直径删掉。这样就是对的。

题解

- 先把直径拿出来，将直径外的点一个一个的和直径中的某一个端点配对并删掉。最后再将直径删掉。这样就是对的。
- 如果当前直径外已经没有点了，那么显然只能将直径的端点删掉。否则一定不会去删直径的端点。

题解

- 先把直径拿出来，将直径外的点一个一个的和直径中的某一个端点配对并删掉。最后再将直径删掉。这样就是对的。
- 如果当前直径外已经没有点了，那么显然只能将直径的端点删掉。否则一定不会去删直径的端点。
- 因为先删一个直径外的点再删直径端点一定是不劣于先删直径端点再删这个直径外的点的。

题目描述

给定一张 n 个点 m 条边的带权无向联通图，q 次询问，每次询问 u_i 到 v_i 的最短路长度。

$n, q \leq 10^5, m - n \leq 20$

题解

- 首先随便搞一棵生成树，那么会有一些边不在生成树上。

题解

- 首先随便搞一棵生成树，那么会有一些边不在生成树上。
- 把这些边的端点标记为特殊点。~~有一个暴力的想法就是每次跑一遍虚树~~

题解

- 首先随便搞一棵生成树，那么会有一些边不在生成树上。
- 把这些边的端点标记为特殊点。~~有一个暴力的想法就是每次跑一遍虚树~~
- 对于一个询问，如果最短路只经过生成树上的边，就可以直接计算。

题解

- 首先随便搞一棵生成树，那么会有一些边不在生成树上。
- 把这些边的端点标记为特殊点。~~有一个暴力的想法就是每次跑一遍虚树~~
- 对于一个询问，如果最短路只经过生成树上的边，就可以直接计算。
- 否则一定经过了一个特殊点，可以枚举这个特殊点，然后更新答案就行了。

题目描述

给定一棵 n 个节点的树，定义叶子是度数为 1 的节点，点集 S 是好的如果满足任意两个 S 中的点之间的距离 $\leq k$，现在要求将所有叶子分成若干不相交的好的集合，请问最少分成多少个好的集合。

$n, k \leq 10^6$

题解

- 先随便选一个根，方便起见选一个不是叶子的根，然后可以自底向上贪心。

题解

- 先随便选一个根，方便起见选一个不是叶子的根，然后可以自底向上贪心。
- 设 f_p 表示 p 为根的子树内已完成的好的集合的数量，g_p 表示未完成的集合中离 p 最远的叶子的距离。

题解

- 先随便选一个根，方便起见选一个不是叶子的根，然后可以自底向上贪心。
- 设 f_p 表示 p 为根的子树内已完成的好的集合的数量，g_p 表示未完成的集合中离 p 最远的叶子的距离。
- 如何计算 f_p 和 g_p？首先将所有儿子的 f 先求和，然后将所有儿子的 g 排序，检查一下最大的 g 和次大的 g 相加是否大于 k，如果大于，那么就将最大的 g 删掉，并把 f_p 加上 1。否则就令 g_p 等于最大的 g 加 1。

题解

- 先随便选一个根，方便起见选一个不是叶子的根，然后可以自底向上贪心。
- 设 f_p 表示 p 为根的子树内已完成的好的集合的数量，g_p 表示未完成的集合中离 p 最远的叶子的距离。
- 如何计算 f_p 和 g_p？首先将所有儿子的 f 先求和，然后将所有儿子的 g 排序，检查一下最大的 g 和次大的 g 相加是否大于 k，如果大于，那么就将最大的 g 删掉，并把 f_p 加上 1。否则就令 g_p 等于最大的 g 加 1。
- 这样贪心就可以了。

题目描述

给定 n 个节点，每个节点上有一个数字，要求用这些节点构建一棵二叉搜索树，满足有边相邻的两个节点上的数字互质。

$n \leq 600$

题解

- 先排序，设 $f[i][j][k]$ 表示第 i 个点到第 j 个点，以 k 为根是否可行。

题解

- 先排序，设 $f[i][j][k]$ 表示第 i 个点到第 j 个点，以 k 为根是否可行。
- 转移的时候查看是否存在 $u \in [i, k-1]$ 满足 $f[i][k-1][u] = 1$ 且 u 上的数字和 k 上的数字互质，右边也一样。这样就是 $O(n^4)$ 的。

题解

- 先排序，设 $f[i][j][k]$ 表示第 i 个点到第 j 个点，以 k 为根是否可行。
- 转移的时候查看是否存在 $u \in [i, k-1]$ 满足 $f[i][k-1][u] = 1$ 且 u 上的数字和 k 上的数字互质，右边也一样。这样就是 $O(n^4)$ 的。
- 注意到上面的做法有一些重复运算。于是设 $L[i][j]$ 表示是否存在 $k \in [i, j]$ 满足 $f[i][j][k] = 1$ 且 k 上的数字和 $j+1$ 上的数字互质，类似地定义 $R[i][j]$。

题解

- 先排序，设 $f[i][j][k]$ 表示第 i 个点到第 j 个点，以 k 为根是否可行。
- 转移的时候查看是否存在 $u \in [i, k-1]$ 满足 $f[i][k-1][u] = 1$ 且 u 上的数字和 k 上的数字互质，右边也一样。这样就是 $O(n^4)$ 的。
- 注意到上面的做法有一些重复运算。于是设 $L[i][j]$ 表示是否存在 $k \in [i, j]$ 满足 $f[i][j][k] = 1$ 且 k 上的数字和 $j+1$ 上的数字互质，类似地定义 $R[i][j]$。
- 于是复杂度就被降为 $O(n^3)$。

题目描述

给定一棵以 1 号点为根的树。若满足以下条件，则认为节点 p 处有一个 k 叉高度为 m 的堆：

- 若 $m = 1$，则 p 本身就是一个 k 叉高度为 1 的堆。
- 若 $m > 1$，则 p 需要有至少 k 个儿子满足在儿子处有一个 k 叉高度为 $m-1$ 的堆。

令 $dp[k][p]$ 表示在 p 处 k 叉堆的最大高度，求 $\sum_{i=1}^{n}\sum_{j=1}^{n} dp[i][j]$。
$n \leq 3 * 10^5$

题解

- 如果固定 k，求所有的 $dp[p][k]$。

题解

- 如果固定 k，求所有的 $dp[p][k]$。
- dfs 整棵树，对于某个点 p，若其子节点数 $< k$，则 $dp[p][k] = 1$，否则 $=$ 子节点的 dp 值中第 k 大的值 $+1$。

题解

- 如果固定 k，求所有的 $dp[p][k]$。
- dfs 整棵树，对于某个点 p，若其子节点数 $< k$，则 $dp[p][k] = 1$，否则 $=$ 子节点的 dp 值中第 k 大的值 $+1$。
- 运用 *nth_element* 可以做到 $O(n)$。这样总复杂度就是 $O(n^2)$。

题解

- 观察到若 $k > \sqrt{n}$，则 $dp[p][k] \leq 2$。

题解

- 观察到若 $k > \sqrt{n}$，则 $dp[p][k] \leq 2$。
- 于是可以将 k 分开讨论，这样是 $O(n^{\frac{3}{2}})$。但还是太慢。

题解

- 观察到若 $k > \sqrt{n}$，则 $dp[p][k] \leq 2$。
- 于是可以将 k 分开讨论，这样是 $O(n^{\frac{3}{2}})$。但还是太慢。
- 观察到若 $k > n^{\frac{1}{3}}$，则 $dp[p][k] \leq 3$。于是又可以讨论一下。

题解

- 观察到若 $k > \sqrt{n}$，则 $dp[p][k] \leq 2$。
- 于是可以将 k 分开讨论，这样是 $O(n^{\frac{3}{2}})$。但还是太慢。
- 观察到若 $k > n^{\frac{1}{3}}$，则 $dp[p][k] \leq 3$。于是又可以讨论一下。
- 然而并不需要一直这样讨论。

题解

- 观察到若 $k > \sqrt{n}$，则 $dp[p][k] \leq 2$。
- 于是可以将 k 分开讨论，这样是 $O(n^{\frac{3}{2}})$。但还是太慢。
- 观察到若 $k > n^{\frac{1}{3}}$，则 $dp[p][k] \leq 3$。于是又可以讨论一下。
- 然而并不需要一直这样讨论。
- 显然对于 $k > 1$，有 $dp[p][k] \leq log(n)$，所以可以考虑对 dp 值分段计算。

题解

- 观察到若 $k > \sqrt{n}$，则 $dp[p][k] \leq 2$。
- 于是可以将 k 分开讨论，这样是 $O(n^{\frac{3}{2}})$。但还是太慢。
- 观察到若 $k > n^{\frac{1}{3}}$，则 $dp[p][k] \leq 3$。于是又可以讨论一下。
- 然而并不需要一直这样讨论。
- 显然对于 $k > 1$，有 $dp[p][k] \leq log(n)$，所以可以考虑对 dp 值分段计算。
- 设 $f[p][x]$ 表示满足 $dp[p][k] \leq x$ 的最大的 k。

题解

- 观察到若 $k > \sqrt{n}$，则 $dp[p][k] \leq 2$。
- 于是可以将 k 分开讨论，这样是 $O(n^{\frac{3}{2}})$。但还是太慢。
- 观察到若 $k > n^{\frac{1}{3}}$，则 $dp[p][k] \leq 3$。于是又可以讨论一下。
- 然而并不需要一直这样讨论。
- 显然对于 $k > 1$，有 $dp[p][k] \leq log(n)$，所以可以考虑对 dp 值分段计算。
- 设 $f[p][x]$ 表示满足 $dp[p][k] \leq x$ 的最大的 k。
- f 只有 $nlog(n)$ 个状态，计算 f 的时间复杂度是 $O(nlog^2(n))$，计算出 f 后就可以轻松得到答案了。

题目描述

一棵 n 个点的树，每个点有权值。你想砍树。

你可以砍任意次，每次你选择一些边断开，需要满足砍完后每个连通块的权值和是相等的。求有多少种砍树方案。

$n \leq 10^6$

题解

- 先假设只砍一次。令所有点权和为 S，那么假设要砍成 k 个连通块，则每个连通块的权值和均为 $\frac{S}{k}$。

题解

- 先假设只砍一次。令所有点权和为 S，那么假设要砍成 k 个连通块，则每个连通块的权值和均为 $\frac{S}{k}$。
- 考虑如何得到砍的方案，以 1 号点为根 dfs，若当前点的子树之和 $= \frac{S}{k}$，则将当前子树砍下来。若最后砍了 k 次（包括 1 号点），则得到了一个合法的砍树方案。

题解

- 先假设只砍一次。令所有点权和为 S，那么假设要砍成 k 个连通块，则每个连通块的权值和均为 $\frac{S}{k}$。
- 考虑如何得到砍的方案，以 1 号点为根 dfs，若当前点的子树之和 $= \frac{S}{k}$，则将当前子树砍下来。若最后砍了 k 次（包括 1 号点），则得到了一个合法的砍树方案。
- 可以看出实际上就是要满足 $\sum_{i=1}^{n}[\frac{S}{k}|sum_i] = k$。

题解

- 先假设只砍一次。令所有点权和为 S，那么假设要砍成 k 个连通块，则每个连通块的权值和均为 $\frac{S}{k}$。
- 考虑如何得到砍的方案，以 1 号点为根 dfs，若当前点的子树之和 $= \frac{S}{k}$，则将当前子树砍下来。若最后砍了 k 次（包括 1 号点），则得到了一个合法的砍树方案。
- 可以看出实际上就是要满足 $\sum_{i=1}^{n}[\frac{S}{k}|sum_i] = k$。
- 上式变形得到 $\frac{S}{gcd(S,sum_i)}|k$，于是可以在 $O(n\,ln(n))$ 的时间内计算。

题解

- 先假设只砍一次。令所有点权和为 S，那么假设要砍成 k 个连通块，则每个连通块的权值和均为 $\frac{S}{k}$。
- 考虑如何得到砍的方案，以 1 号点为根 dfs，若当前点的子树之和 $= \frac{S}{k}$，则将当前子树砍下来。若最后砍了 k 次（包括 1 号点），则得到了一个合法的砍树方案。
- 可以看出实际上就是要满足 $\sum_{i=1}^{n}[\frac{S}{k}|sum_i] = k$。
- 上式变形得到 $\frac{S}{gcd(S,sum_i)}|k$，于是可以在 $O(n\,ln(n))$ 的时间内计算。
- 由此也可以看出砍成 k 个连通块的方案如果存在，那么也是唯一的。

题解

- 先假设只砍一次。令所有点权和为 S，那么假设要砍成 k 个连通块，则每个连通块的权值和均为 $\frac{S}{k}$。
- 考虑如何得到砍的方案，以 1 号点为根 dfs，若当前点的子树之和 $= \frac{S}{k}$，则将当前子树砍下来。若最后砍了 k 次（包括 1 号点），则得到了一个合法的砍树方案。
- 可以看出实际上就是要满足 $\sum_{i=1}^{n}[\frac{S}{k}|sum_i] = k$。
- 上式变形得到 $\frac{S}{gcd(S,sum_i)}|k$，于是可以在 $O(n\,ln(n))$ 的时间内计算。
- 由此也可以看出砍成 k 个连通块的方案如果存在，那么也是唯一的。
- 那么如果砍多次呢？可以看出如果第一次砍成了 x 块，那么第二次砍成的块数 y 必须满足 $x|y$。这部分也可以 $O(n\,ln(n))$ 算。

题目描述

给定一棵 n 个点的带边权树，m 次询问，每次询问给出两个值 p, k，求以下值：

$$\sum_{q|dis(p,q)\leq k} dis(q,p)$$

题解

- 为什么突然出现了一道裸题呢 O_O?

题解

- 为什么突然出现了一道裸题呢 O_O?
- 首先点分治，把分治结构建好。

题解

- 为什么突然出现了一道裸题呢 O_O?
- 首先点分治，把分治结构建好。
- 每个分治中心要记录当前分治区域所有点到分治中心的距离，以及分别记录每个分治子区域中的所有点到分治中心的距离，以及当前分治中心到其分治祖先的距离。

题解

- 为什么突然出现了一道裸题呢 O_O?
- 首先点分治，把分治结构建好。
- 每个分治中心要记录当前分治区域所有点到分治中心的距离，以及分别记录每个分治子区域中的所有点到分治中心的距离，以及当前分治中心到其分治祖先的距离。
- 每次查询时在每个分治中心根据以上信息查询就可以了。

题目描述

给定一棵 n 个点的带点权树，以 1 为根，m 次询问，每次询问给出两个值 p, k，求以下值：

- p 的子树中距离 $p \leq k$ 的所有点权最小值。

$n \leq 10^5, m \leq 5 * 10^6$

题解

- 最暴力的想法是：以深度为 x 轴坐标，dfs 序为 y 轴坐标，建立一棵二维线段树。

题解

- 最暴力的想法是：以深度为 x 轴坐标，dfs 序为 y 轴坐标，建立一棵二维线段树。
- 然后询问都变成了矩形内最小值查询。复杂度 $O(nlog^2n + mlog^2n)$。

题解

- 最暴力的想法是：以深度为 x 轴坐标，dfs 序为 y 轴坐标，建立一棵二维线段树。
- 然后询问都变成了矩形内最小值查询。复杂度 $O(nlog^2n + mlog^2n)$。
- 不过这题还有一个高妙做法。

#1
#2
#3
#4
#5
#6
#7

题解

- 最暴力的想法是：以深度为 x 轴坐标，dfs 序为 y 轴坐标，建立一棵二维线段树。
- 然后询问都变成了矩形内最小值查询。复杂度 $O(nlog^2n + mlog^2n)$。
- 不过这题还有一个高妙做法。
- 首先将整棵树画出来，同一深度的点放在同一水平线上从左到右排好。

题解

- 最暴力的想法是：以深度为 x 轴坐标，dfs 序为 y 轴坐标，建立一棵二维线段树。
- 然后询问都变成了矩形内最小值查询。复杂度 $O(nlog^2n + mlog^2n)$。
- 不过这题还有一个高妙做法。
- 首先将整棵树画出来，同一深度的点放在同一水平线上从左到右排好。
- 然后设 L_p 为深度为 $dep_p + 1$ 的点中，在 p 的子树中及右边的点中最左的点。设 R_p 为深度 $dep_p + 1$ 的点中，在 p 的子树中及左边的点中最右的点。

题解

- 最暴力的想法是：以深度为 x 轴坐标，dfs 序为 y 轴坐标，建立一棵二维线段树。
- 然后询问都变成了矩形内最小值查询。复杂度 $O(nlog^2n + mlog^2n)$。
- 不过这题还有一个高妙做法。
- 首先将整棵树画出来，同一深度的点放在同一水平线上从左到右排好。
- 然后设 L_p 为深度为 $dep_p + 1$ 的点中，在 p 的子树中及右边的点中最左的点。设 R_p 为深度 $dep_p + 1$ 的点中，在 p 的子树中及左边的点中最右的点。
- 先约定 $p + k$ 代表 p 右边的第 k 个点，$p - k$ 代表 p 左边的第 k 个点。

题解

- 最暴力的想法是：以深度为 x 轴坐标，dfs 序为 y 轴坐标，建立一棵二维线段树。
- 然后询问都变成了矩形内最小值查询。复杂度 $O(nlog^2n + mlog^2n)$。
- 不过这题还有一个高妙做法。
- 首先将整棵树画出来，同一深度的点放在同一水平线上从左到右排好。
- 然后设 L_p 为深度为 $dep_p + 1$ 的点中，在 p 的子树中及右边的点中最左的点。设 R_p 为深度 $dep_p + 1$ 的点中，在 p 的子树中及左边的点中最右的点。
- 先约定 $p + k$ 代表 p 右边的第 k 个点，$p - k$ 代表 p 左边的第 k 个点。
- 接着设 $f[p][i][j]$ 表示从 p 到 $p + 2^i - 1$ 这些点的子树中深度差 $< 2^j$ 的所有点的点权最小值。

题解

- L_p 和 R_p 是很好求的，那么关键是如何求 $f[p][i][j]$。

题解

- L_p 和 R_p 是很好求的，那么关键是如何求 $f[p][i][j]$。
- 首先如果 $i = 0$ 或 $j = 0$ 也很好求。

题解

- L_p 和 R_p 是很好求的，那么关键是如何求 $f[p][i][j]$。
- 首先如果 $i=0$ 或 $j=0$ 也很好求。
- 否则令 $q=p+2^i-1$，u 是 p 沿着 L 数组跳 2^{j-1} 后的点，v 是 q 沿着 R 数组跳 2^{j-1} 后的点。

题解

- L_p 和 R_p 是很好求的，那么关键是如何求 $f[p][i][j]$。
- 首先如果 $i=0$ 或 $j=0$ 也很好求。
- 否则令 $q=p+2^i-1$，u 是 p 沿着 L 数组跳 2^{j-1} 后的点，v 是 q 沿着 R 数组跳 2^{j-1} 后的点。
- 那么 $f[p][i][j]=min(f[p][i][j-1],f[u][k][j-1],f[v-2^k+1][k][j-1])$，其中 k 是满足 $u+2^k-1$ 在 v 左边的最大的 k（类似 ST 表）。

#1
#2
#3
#4
#5
#6
#7

题解

- L_p 和 R_p 是很好求的，那么关键是如何求 $f[p][i][j]$。
- 首先如果 $i=0$ 或 $j=0$ 也很好求。
- 否则令 $q=p+2^i-1$，u 是 p 沿着 L 数组跳 2^{j-1} 后的点，v 是 q 沿着 R 数组跳 2^{j-1} 后的点。
- 那么 $f[p][i][j]=min(f[p][i][j-1],f[u][k][j-1],f[v-2^k+1][k][j-1])$，其中 k 是满足 $u+2^k-1$ 在 v 左边的最大的 k（类似 ST 表）。
- 询问跟上面差不多处理就可以了。

题解

- L_p 和 R_p 是很好求的，那么关键是如何求 $f[p][i][j]$。
- 首先如果 $i=0$ 或 $j=0$ 也很好求。
- 否则令 $q=p+2^i-1$，u 是 p 沿着 L 数组跳 2^{j-1} 后的点，v 是 q 沿着 R 数组跳 2^{j-1} 后的点。
- 那么 $f[p][i][j]=min(f[p][i][j-1],f[u][k][j-1],f[v-2^k+1][k][j-1])$，其中 k 是满足 $u+2^k-1$ 在 v 左边的最大的 k（类似 ST 表）。
- 询问跟上面差不多处理就可以了。
- 现在还有一个问题，就是如何求沿 L 或 R 跳 k 步后的点。注意到 L 数组形成了一棵树，那么可以用树上倍增。这样复杂度就是 $O(nlog^2n+mlogn)$ 的。

题解

- L_p 和 R_p 是很好求的，那么关键是如何求 $f[p][i][j]$。
- 首先如果 $i=0$ 或 $j=0$ 也很好求。
- 否则令 $q=p+2^i-1$，u 是 p 沿着 L 数组跳 2^{j-1} 后的点，v 是 q 沿着 R 数组跳 2^{j-1} 后的点。
- 那么 $f[p][i][j]=min(f[p][i][j-1],f[u][k][j-1],f[v-2^k+1][k][j-1])$，其中 k 是满足 $u+2^k-1$ 在 v 左边的最大的 k（类似 ST 表）。
- 询问跟上面差不多处理就可以了。
- 现在还有一个问题，就是如何求沿 L 或 R 跳 k 步后的点。注意到 L 数组形成了一棵树，那么可以用树上倍增。这样复杂度就是 $O(nlog^2n+mlogn)$ 的。
- 有一种长链剖分黑科技可以 $O(1)$ 求上面这个问题，那么复杂度就变成了 $O(nlog^2n+m)$。

题解

- 对一棵树长链剖分后，预处理几个数组：

#1
#2
#3
#4
#5
#6
#7

题解

- 对一棵树长链剖分后，预处理几个数组：
- 每个点处理其第 2^i 个祖先是哪个点。对于每条链，假设其长度为 len，那么给这条链挂上一个长度为 $len*2$ 的数组，存储这条链上的所有点以及从链顶往上的 len 个点。

#1
#2
#3
#4
#5
#6
#7

题解

- 对一棵树长链剖分后，预处理几个数组：
- 每个点处理其第 2^i 个祖先是哪个点。对于每条链，假设其长度为 len，那么给这条链挂上一个长度为 $len * 2$ 的数组，存储这条链上的所有点以及从链顶往上的 len 个点。
- 假设现在询问 p 的第 k 个祖先，首先找到最大的 i 满足 $2^i \leq k$，并且跳 2^i 步。

#1
#2
#3
#4
#5
#6
#7

题解

- 对一棵树长链剖分后，预处理几个数组：
- 每个点处理其第 2^i 个祖先是哪个点。对于每条链，假设其长度为 len，那么给这条链挂上一个长度为 $len * 2$ 的数组，存储这条链上的所有点以及从链顶往上的 len 个点。
- 假设现在询问 p 的第 k 个祖先，首先找到最大的 i 满足 $2^i \leq k$，并且跳 2^i 步。
- 如果跳完后到了 q，那么可以确定 q 所在的链长 $\geq 2^i$（长链剖分的性质）。

#1
#2
#3
#4
#5
#6
#7

题解

- 对一棵树长链剖分后，预处理几个数组：
- 每个点处理其第 2^i 个祖先是哪个点。对于每条链，假设其长度为 len，那么给这条链挂上一个长度为 $len * 2$ 的数组，存储这条链上的所有点以及从链顶往上的 len 个点。
- 假设现在询问 p 的第 k 个祖先，首先找到最大的 i 满足 $2^i \leq k$，并且跳 2^i 步。
- 如果跳完后到了 q，那么可以确定 q 所在的链长 $\geq 2^i$（长链剖分的性质）。
- 由于 $k - 2^i \leq 2^i$，所以剩下的部分直接在挂在链上的数组上查就行了。