树上启发式合并、树分治

宋佳兴

2024 年 7 月 18 日

- 有想法的可以直接和我交流。
- 有问题欢迎上来提问。
- 讲课中途会休息 15~20 分钟。

IOI2011 Race

给定一棵 n 个结点的树，边有边权。找到一条简单路径，使得路径上的边权之和为 k，且边的数量最少。

$n \le 2 \times 10^5$

分别想一想点分治和启发式合并怎么做。

题目链接

点分治

选取整棵树的**重心**作为根，记为 $root$，然后将树上的所有路径分为两类：

- 包含 $root$ 的路径：这类路径具有较好的性质，设 dis_u 表示 u 到 root 的距离，那么路径 $u \to v$ 的长度为 $dis_u + dis_v$。利用这个性质通常可以设计高效算法来统计这一类路径，不妨假设有复杂度为 $T(n)$ 的算法。

点分治

选取整棵树的**重心**作为根，记为 $root$，然后将树上的所有路径分为两类：

- 包含 $root$ 的路径：这类路径具有较好的性质，设 dis_u 表示 u 到 root 的距离，那么路径 $u \to v$ 的长度为 $dis_u + dis_v$。利用这个性质通常可以设计高效算法来统计这一类路径，不妨假设有复杂度为 $T(n)$ 的算法。
- 不包含 $root$ 的路径：将 $root$ 删除，对 $root$ 的每棵子树递归求解。

点分治

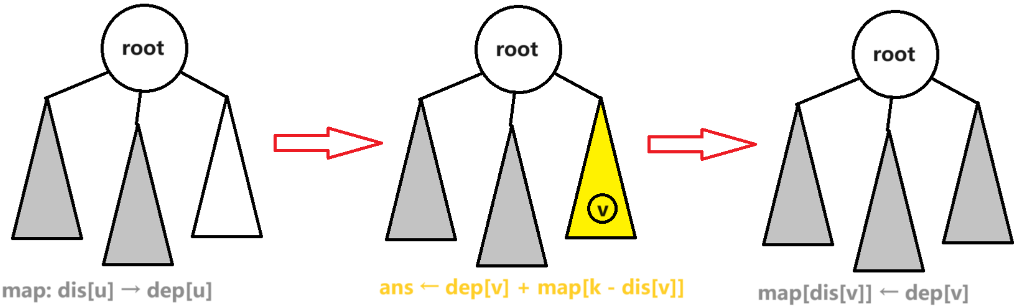
选取整棵树的**重心**作为根，记为 $root$，然后将树上的所有路径分为两类：

- 包含 $root$ 的路径：这类路径具有较好的性质，设 dis_u 表示 u 到 root 的距离，那么路径 $u \to v$ 的长度为 $dis_u + dis_v$。利用这个性质通常可以设计高效算法来统计这一类路径，不妨假设有复杂度为 $T(n)$ 的算法。
- 不包含 $root$ 的路径：将 $root$ 删除，对 $root$ 的每棵子树递归求解。

通过重心的性质可以证明该算法的递归层数不会超过 $\log_2 n$，从而复杂度为 $T(n) \cdot \log n$。

点分治

如何在 $dis_u + dis_v = k$ 的点对 (u, v) 中找到 $dep_u + dep_v$ 的最小值？遍历各个子树，维护一个 map。



启发式合并

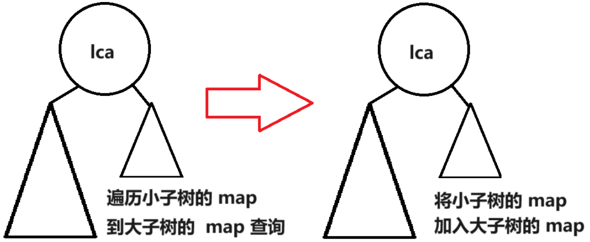
另一种解决方案：固定 1 为树根，路径 (u, v) 的长度为 $dis_u + dis_v - 2dis_{lca}$，边数为 $dep_u + dep_v - 2dep_{lca}$。

枚举 lca，问题转化为在 $dis_u + dis_v = k + 2dis_{lca}$ 的点对 (u, v) 中找到 $dep_u + dep_v$ 的最小值。

启发式合并

另一种解决方案：固定 1 为树根，路径 (u, v) 的长度为 $dis_u + dis_v - 2dis_{lca}$，边数为 $dep_u + dep_v - 2dep_{lca}$。

枚举 lca，问题转化为在 $dis_u + dis_v = k + 2dis_{lca}$ 的点对 (u, v) 中找到 $dep_u + dep_v$ 的最小值。每棵子树维护一个 map，从 dis_u 映射到 dep_u（dis_u 相同的取最小值）。map 上传时采用启发式合并，复杂度 $O(n\log^2 n)$。



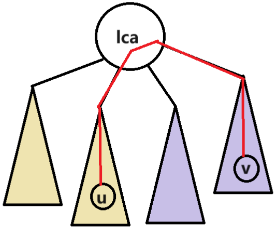
边分治（变种）

假如我们选取一条边作为分治中心，会不会会更方便一些（不需要 map)？但是这样复杂度是无法保证的。

边分治（变种）

假如我们选取一条边作为分治中心，会不会会更方便一些（不需要 map)？但是这样复杂度是无法保证的。

一种近似于边分治的方案：先使用点分治，然后分治处理根的各个儿子的子树。



边分治（变种）

由于各个子树的大小不同，按照 Huffman 树进行分治复杂度最佳，可以证明此时总复杂度和点分治一致。

边分治（变种）

由于各个子树的大小不同，按照 Huffman 树进行分治复杂度最佳，可以证明此时总复杂度和点分治一致。

这个算法比传统边分治常数更小，而且通用性要好一些。另外，实现上通常把 $root$ 视作一个大小为 1 的子树，和其他子树一起建 Huffman 树。

BZOJ3784 树上的路径

给定一棵 n 个结点的树，边有边权（正整数）。将树上 $n(n-1)/2$ 个点对的距离从大到小排序，输出前 m 个距离值。

$n \leq 50{,}000 \quad m \leq \min(300{,}000, \frac{n(n-1)}{2})$

提示：

题目链接

BZOJ3784 树上的路径

给定一棵 n 个结点的树，边有边权（正整数）。将树上 $n(n-1)/2$ 个点对的距离从大到小排序，输出前 m 个距离值。

$n \leq 50{,}000 \quad m \leq \min(300{,}000, \frac{n(n-1)}{2})$

提示：超级钢琴的技巧

题目链接

BZOJ3784 树上的路径

对树进行轻重链剖分，记录重链优先的 DFS 序。考察启发式合并的过程，它本质上是将 $n(n-1)/2$ 个点对分为了 $O(n\log n)$ 类，每一类的端点 u 和 lca 是固定的，而端点 v 的 DFS 序在一个区间中。

BZOJ3784 树上的路径

对树进行轻重链剖分，记录重链优先的 DFS 序。考察启发式合并的过程，它本质上是将 $n(n-1)/2$ 个点对分为了 $O(n\log n)$ 类，每一类的端点 u 和 lca 是固定的，而端点 v 的 DFS 序在一个区间中。

每一类可以用 $(u, lca, left, right)$ 来描述，该类里面的最长路径对应 $[left, right]$ 里深度最大的点。

BZOJ3784 树上的路径

对树进行轻重链剖分，记录重链优先的 DFS 序。考察启发式合并的过程，它本质上是将 $n(n-1)/2$ 个点对分为了 $O(n\log n)$ 类，每一类的端点 u 和 lca 是固定的，而端点 v 的 DFS 序在一个区间中。

每一类可以用 $(u, lca, left, right)$ 来描述，该类里面的最长路径对应 $[left, right]$ 里深度最大的点。

将所有类放进 priority_queue 中，比较关键字为每一类的最长路径。然后执行以下操作 m 次：

BZOJ3784 树上的路径

对树进行轻重链剖分，记录重链优先的 DFS 序。考察启发式合并的过程，它本质上是将 $n(n-1)/2$ 个点对分为了 $O(n\log n)$ 类，每一类的端点 u 和 lca 是固定的，而端点 v 的 DFS 序在一个区间中。

每一类可以用 $(u, lca, left, right)$ 来描述，该类里面的最长路径对应 $[left, right]$ 里深度最大的点。

将所有类放进 priority_queue 中，比较关键字为每一类的最长路径。然后执行以下操作 m 次：

- 记堆顶为 $(u, lca, left, right)$，mid 为 $[left, right]$ 里深度最大的点，先删除堆顶，再加入 $(u, lca, left, mid-1)$ 和 $(u, lca, mid+1, right)$，这番操作相当于仅从堆中移除了最长路径。

BZOJ3784 树上的路径

总复杂度为 $O((n+m)\log n)$，因为 priority_queue 的构造函数是线性算法。

BZOJ3784 树上的路径

总复杂度为 $O((n+m)\log n)$，因为 priority_queue 的构造函数是线性算法。

思考：点分治或边分治可以单 $\log$ 吗?

CF375D Tree and Queries

给定一棵 n 个结点的树，每个结点都有一个颜色。你需要回答 m 个查询，每个查询包含两个参数 (v, k)，问以 v 为根的子树中颜色出现次数至少为 k 的颜色数量。

$n, m \leq 100{,}000$

题目链接

CF375D Tree and Queries

首先，询问使用离线处理。

一种叫 dsu on tree 的技巧：首先进行轻重链剖分，维护一个初始为空的“桶”，然后从根开始执行以下递归算法：

- 记当前结点为 u，对于 u 的每个轻儿子，先递归下去，递归返回时清空“桶”。
- 递归处理 u 的重儿子，此时“桶”里装了重儿子的子树。
- 将 u 子树内除重子树以外的点一个一个加入“桶”中。
- 此时“桶”里装了 u 的子树，可以对“桶”进行 u 相关的查询操作。

CF375D Tree and Queries

首先，询问使用离线处理。

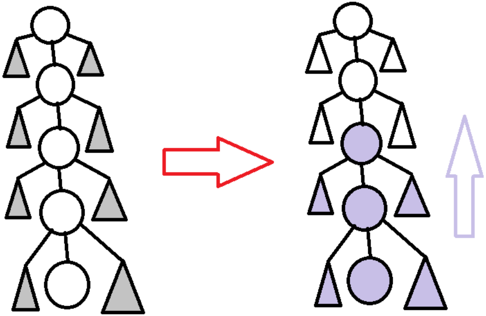
一种叫 dsu on tree 的技巧：首先进行轻重链剖分，维护一个初始为空的“桶”，然后从根开始执行以下递归算法：

- 记当前结点为 u，对于 u 的每个轻儿子，先递归下去，递归返回时清空“桶”。
- 递归处理 u 的重儿子，此时“桶”里装了重儿子的子树。
- 将 u 子树内除重子树以外的点一个一个加入“桶”中。
- 此时“桶”里装了 u 的子树，可以对“桶”进行 u 相关的查询操作。

可以看出，dsu on tree 和启发式合并具有相同的复杂度。

CF375D Tree and Queries

宏观上，dsu on tree 过程如下图所示：



CF375D Tree and Queries

现在，要将“桶”替换为一个具体的数据结构，支持加入结点，和查询出现次数至少为 k 的颜色数量。

CF375D Tree and Queries

现在，要将“桶”替换为一个具体的数据结构，支持加入结点，和查询出现次数至少为 k 的颜色数量。

容易想到用数组 occ_i 表示颜色 i 的出现次数，并用 cnt_i 表示 $occ = i$ 的颜色数量。问题转化为单点修改和查询后缀和，使用树状数组维护。

CF375D Tree and Queries

现在，要将“桶”替换为一个具体的数据结构，支持加入结点，和查询出现次数至少为 k 的颜色数量。

容易想到用数组 occ_i 表示颜色 i 的出现次数，并用 cnt_i 表示 $occ = i$ 的颜色数量。问题转化为单点修改和查询后缀和，使用树状数组维护。

然而，还有一个更聪明的做法。

CF375D Tree and Queries

现在，要将“桶”替换为一个具体的数据结构，支持加入结点，和查询出现次数至少为 k 的颜色数量。

容易想到用数组 occ_i 表示颜色 i 的出现次数，并用 cnt_i 表示 $occ = i$ 的颜色数量。问题转化为单点修改和查询后缀和，使用树状数组维护。

然而，还有一个更聪明的做法。

注意，当 occ_i 增大 1 时，cnt_{occ_i} 和 cnt_{occ_i+1} 一个 -1 一个 $+1$，只有 1 个位置的后缀和会发生变化。因此我们可以直接维护 cnt 的后缀和。

复杂度 $O(n\log n + m)$，只使用启发式合并是做不到这个复杂度的。

CF375D Tree and Queries

现在，要将“桶”替换为一个具体的数据结构，支持加入结点，和查询出现次数至少为 k 的颜色数量。

容易想到用数组 occ_i 表示颜色 i 的出现次数，并用 cnt_i 表示 $occ = i$ 的颜色数量。问题转化为单点修改和查询后缀和，使用树状数组维护。

然而，还有一个更聪明的做法。

注意，当 occ_i 增大 1 时，cnt_{occ_i} 和 cnt_{occ_i+1} 一个 -1 一个 $+1$，只有 1 个位置的后缀和会发生变化。因此我们可以直接维护 cnt 的后缀和。

复杂度 $O(n \log n + m)$，只使用启发式合并是做不到这个复杂度的。

基本上启发式合并能做的事情 dsu on tree 都能做，而且有时 dsu on tree 还能少一个 $\log$。

CF1824C LuoTianyi and XOR-Tree

给定一棵 n 个结点的树，树的根为结点 1，点有点权。每次操作可以将一个结点的权值改为任意非负整数。

需要找到最少的操作次数，使得从根到每个叶子结点的路径上的所有点权的异或和为 0。

$n \leq 10^5$

提示：

题目链接

CF1824C LuoTianyi and XOR-Tree

给定一棵 n 个结点的树，树的根为结点 1，点有点权。每次操作可以将一个结点的权值改为任意非负整数。

需要找到最少的操作次数，使得从根到每个叶子结点的路径上的所有点权的异或和为 0。

$n \leq 10^5$

提示：把每个点的点权改为“根到该点的路径异或和”，然后设计一个复杂度较高的 DP。

题目链接

CF1824C LuoTianyi and XOR-Tree

首先，将问题进行转化：把每个点的点权改为“根到该点的路径异或和”，每次操作变成了对子树进行异或操作，目标是使所有叶子的点权都为 0。

CF1824C LuoTianyi and XOR-Tree

首先，将问题进行转化：把每个点的点权改为“根到该点的路径异或和”，每次操作变成了对子树进行异或操作，目标是使所有叶子的点权都为 0。

不难想到如下的 DP，设 $dp_{u,i}$ 表示使得 u 子树内所有叶子点权都为 i 的最少操作次数。转移如下：

- $dp_{u,i} = \sum_v dp_{v,i}$
- 记 $x_u = \min dp_{u,i}$，然后令 $dp_{u,i} = \min(dp_{u,i}, x_u + 1)$。

CF1824C LuoTianyi and XOR-Tree

首先，将问题进行转化：把每个点的点权改为“根到该点的路径异或和”，每次操作变成了对子树进行异或操作，目标是使所有叶子的点权都为 0。

不难想到如下的 DP，设 $dp_{u,i}$ 表示使得 u 子树内所有叶子点权都为 i 的最少操作次数。转移如下：

- $dp_{u,i} = \sum_v dp_{v,i}$
- 记 $x_u = \min dp_{u,i}$，然后令 $dp_{u,i} = \min(dp_{u,i}, x_u + 1)$。

从第二条转移可以看出一个性质：$dp_{u,i}$ 只有 x_u 和 $x_u + 1$ 两种取值，因此我们可以只记录 x_u 和 $dp_{u,i} = x_u$ 的那些 i 构成的集合 s_u。

CF1824C LuoTianyi and XOR-Tree

然后考虑如何转移：

- $x_u = \sum_v x_v + \min_i \sum_v [i \notin s_v]$
- 将 $\sum_v [i \notin s_v]$ 取到最小值，即 $\sum_v [i \in s_v]$ 取到最大值的 i 加入 s_u。

CF1824C LuoTianyi and XOR-Tree

然后考虑如何转移：

- $x_u = \sum_v x_v + \min_i \sum_v [i \notin s_v]$
- 将 $\sum_v [i \notin s_v]$ 取到最小值，即 $\sum_v [i \in s_v]$ 取到最大值的 i 加入 s_u。

根据 $\max \sum_v [i \in s_v]$ 的取值，分两种情况讨论：

- $\max \sum_v [i \in s_v] = 1$，意思是所有 s_v 两两不交，此时 s_u 为所有 s_v 的并，使用 set 启发式合并。
- 其他情况，最终 s_u 里的每个元素必须在两个儿子的 s_v 里出现过，也就是必须在轻儿子出现一次，枚举轻儿子的 s_v 依次判定即可。

复杂度 $O(n \log^2 n)$。

来源未知

给定一棵 n 个结点、以 1 为根的树，点有点权。定义一棵树的权值为其所有**简单路径**的 LIS 的最大值。试求给定树的每棵子树的权值。

$n \le 2 \times 10^5$

来源未知

给定一棵 n 个结点、以 1 为根的树，点有点权。定义一棵树的权值为其所有**简单路径**的 **LIS** 的最大值。试求给定树的每棵子树的权值。

$n \leq 2 \times 10^5$

提示：需要用到二分求 LIS 的算法。

来源未知

二分法求 LIS 的方法是定义一个辅助数组：g_i 表示长度为 i 的上升子序列的末尾元素的最小值。

来源未知

二分法求 LIS 的方法是定义一个辅助数组：g_i 表示长度为 i 的上升子序列的末尾元素的最小值。

推广到树上，定义 $dp1_{u,i}$ 表示 u 子树内，长度为 i 且**深度递减**的上升子序列的末尾元素的最小值，$dp2_{u,i}$ 对应下降子序列。

来源未知

二分法求 LIS 的方法是定义一个辅助数组：g_i 表示长度为 i 的上升子序列的末尾元素的最小值。

推广到树上，定义 $dp1_{u,i}$ 表示 u 子树内，长度为 i 且**深度递减**的上升子序列的末尾元素的最小值，$dp2_{u,i}$ 对应下降子序列。

由于 dp 数组的第二维不会超过 u 子树的高度，可以联想到使用长链剖分优化 dp，转移过程相对简单：

来源未知

二分法求 LIS 的方法是定义一个辅助数组：g_i 表示长度为 i 的上升子序列的末尾元素的最小值。

推广到树上，定义 $dp1_{u,i}$ 表示 u 子树内，长度为 i 且**深度递减**的上升子序列的末尾元素的最小值，$dp2_{u,i}$ 对应下降子序列。

由于 dp 数组的第二维不会超过 u 子树的高度，可以联想到使用长链剖分优化 dp，转移过程相对简单：

- 从长儿子继承 dp 数组，然后依次将其余儿子的 dp 数组合并进去，同时统计所有以 u 为 lca 的路径的 LIS 的最大值。

来源未知

二分法求 LIS 的方法是定义一个辅助数组：g_i 表示长度为 i 的上升子序列的末尾元素的最小值。

推广到树上，定义 $dp1_{u,i}$ 表示 u 子树内，长度为 i 且**深度递减**的上升子序列的末尾元素的最小值，$dp2_{u,i}$ 对应下降子序列。

由于 dp 数组的第二维不会超过 u 子树的高度，可以联想到使用长链剖分优化 dp，转移过程相对简单：

- 从长儿子继承 dp 数组，然后依次将其余儿子的 dp 数组合并进去，同时统计所有以 u 为 lca 的路径的 LIS 的最大值。
- 把 u 通过二分法插入 dp 数组。

来源未知

二分法求 LIS 的方法是定义一个辅助数组：g_i 表示长度为 i 的上升子序列的末尾元素的最小值。

推广到树上，定义 $dp1_{u,i}$ 表示 u 子树内，长度为 i 且**深度递减**的上升子序列的末尾元素的最小值，$dp2_{u,i}$ 对应下降子序列。

由于 dp 数组的第二维不会超过 u 子树的高度，可以联想到使用长链剖分优化 dp，转移过程相对简单：

- 从长儿子继承 dp 数组，然后依次将其余儿子的 dp 数组合并进去，同时统计所有以 u 为 lca 的路径的 LIS 的最大值。
- 把 u 通过二分法插入 dp 数组。

复杂度为 $O(n \log n)$，主要瓶颈在于二分操作。

来源未知

给定一棵 n 个结点的树，点有点权。定义一条有向路径的权值为其点权序列的 LIS 的长度。对于每个结点 u，求出以 u 为起点的最大路径权值。

$n \leq 2 \times 10^5$

提示：

来源未知

给定一棵 n 个结点的树，点有点权。定义一条有向路径的权值为其点权序列的 LIS 的长度。对于每个结点 u，求出以 u 为起点的最大路径权值。

$n \leq 2 \times 10^5$

提示：尝试使用点分治。

来源未知

先使用点分治，现在只需考虑所有跨过 $root$ 的路径，以及它们对答案的贡献。

来源未知

先使用点分治，现在只需考虑所有跨过 $root$ 的路径，以及它们对答案的贡献。

类比上一道题的定义，设计一个 dp 状态：$dp_{u,i}$ 表示 u 子树内，所有长度为 i 且**深度递减**的下降子序列中，末尾元素的最大值。

来源未知

先使用点分治，现在只需考虑所有跨过 $root$ 的路径，以及它们对答案的贡献。

类比上一道题的定义，设计一个 dp 状态：$dp_{u,i}$ 表示 u 子树内，所有长度为 i 且**深度递减**的下降子序列中，末尾元素的最大值。

由于 dp 数组的第二维不会超过 u 子树的高度，只要转移采用启发式合并，复杂度就和长链剖分相同。

来源未知

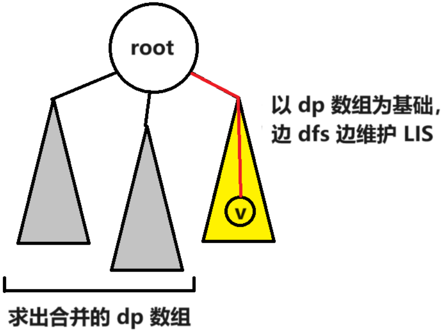
先使用用点分治，现在只需考虑所有跨过 $root$ 的路径，以及它们对答案的贡献。

类比上一道题的定义，设计一个 dp 状态：$dp_{u,i}$ 表示 u 子树内，所有长度为 i 且**深度递减**的下降子序列中，末尾元素的最大值。

由于 dp 数组的第二维不会超过 u 子树的高度，只要转移采用启发式合并，复杂度就和长链剖分相同。

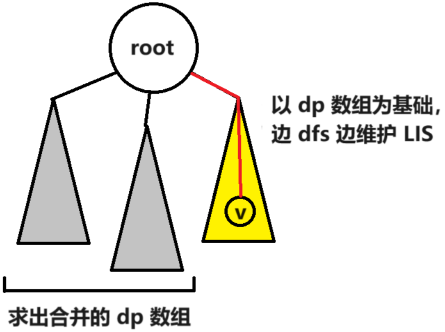
来源未知

接下来考虑如何计算答案，二分求 LIS 支持在末尾动态插入元素，也可以撤销。利用这个特性，通过一遍 dfs 就可以求出 $root$ 一个儿子子树内的答案。



来源未知

接下来考虑如何计算答案，二分求 LIS 支持在末尾动态插入元素，也可以撤销。利用这个特性，通过一遍 dfs 就可以求出 $root$ 一个儿子子树内的答案。



复杂度 $O(n \log^2 n)$。

USACO2018 Cow at Large

贝茜被农民们逼进了一个偏僻的农场。农场可视为一棵有 n 个结点的树。每个叶子结点都是出入口。

开始时，每个出入口都可以放一个农民（也可以不放）。每个时刻，贝茜和农民都可以移动到相邻的一个结点。如果某一时刻农民与贝茜相遇了（在边上或点上均算），则贝茜将被抓住。抓捕过程中，农民们与贝茜均知道对方在哪个结点。

请问：对于每个结点 i，如果开始时贝茜在该结点，最少有多少农民，她才会被抓住。

$n \le 7 \times 10^4$

题目链接

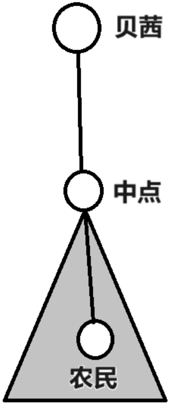
USACO2018 Cow at Large

假如贝茜位于根结点 $root$，怎样以最少的农民抓住她?

简单分析可以知道，假如农民位于叶子结点 u，那么贝茜无论如何也无法到达 u 和 $root$ 的中点，所以中点的子树都是不可达的。

问题转化为：选择尽可能少的子树，使得所有叶子被覆盖。

进一步转化：所有能被覆盖的结点构成若干棵不相交的子树，而这些子树的数量就是答案。结点 u 能被覆盖的条件为 $dis(root, u) \geq \min(u)$，其中 $\min(u)$ 表示 u 到最近叶子的距离。



USACO2018 Cow at Large

计算这些子树的数量有几种方法，最适合这道题的方法是：对于每个能被覆盖的结点 u，令它产生 $1 - child(u)$ 的贡献。这里 $child(u)$ 表示 u 的儿子数量，只要 u 不是根，$child(u) = deg(u) - 1$。

USACO2018 Cow at Large

计算这些子树的数量有几种方法，最适合这道题的方法是：对于每个能被覆盖的结点 u，令它产生 $1 - child(u)$ 的贡献。这里 $child(u)$ 表示 u 的儿子数量，只要 u 不是根，$child(u) = deg(u) - 1$。

整理上述思路，当**非叶结点** $root$ 为根时，答案为

$$\sum_u [dis(root, u) \geq \min(u)](2 - deg(u))$$

可以直接用点分治求解。

WC2018 通道（选讲）

给定三棵 n 个结点的树，边有边权（非负）。

用 $d_1(u,v), d_2(u,v), d_3(u,v)$ 分别表示三棵树上 u, v 的距离。求

$$\max_{u,v} d_1(u,v) + d_2(u,v) + d_3(u,v)$$

$n \le 10^5$

提示：

题目链接

WC2018 通道（选讲）

给定三棵 n 个结点的树，边有边权（非负）。

用 $d_1(u,v), d_2(u,v), d_3(u,v)$ 分别表示三棵树上 u,v 的距离。求

$$\max_{u,v} d_1(u,v) + d_2(u,v) + d_3(u,v)$$

$n \le 10^5$

提示：先思考一棵树和两棵树怎么做。

题目链接

WC2018 通道（选讲）

一棵树的情况是简单的。接下来思考两棵树的情况。

WC2018 通道（选讲）

一棵树的情况是简单的。接下来思考两棵树的情况。

对于树上的一个结点集合 S，定义其直径为 S 中距离最远的一对点。

有结论：$S \cap T$ 的直径端点一定是 S 的直径端点或 T 的直径端点，因此知道 S 和 T 的直径端点就可以通过 ST 表 $O(1)$ 求 $S \cap T$ 的直径端点。

WC2018 通道（选讲）

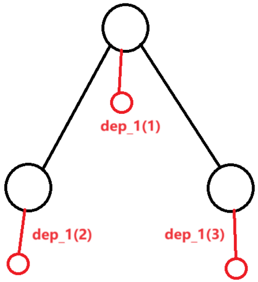
一棵树的情况是简单的。接下来思考两棵树的情况。

对于树上的一个结点集合 S，定义其直径为 S 中距离最远的一对点。

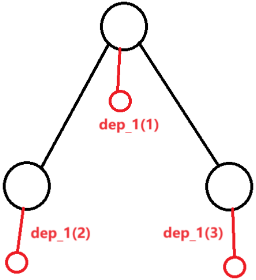
有结论：$S \cap T$ 的直径端点一定是 S 的直径端点或 T 的直径端点，因此知道 S 和 T 的直径端点就可以通过 ST 表 $O(1)$ 求 $S \cap T$ 的直径端点。

固定 u, v 在第一棵树的 lca，那么只需要最大化 $dep_1(u) + dep_1(v) + d_2(u, v)$，对第二棵树按如图方式添加叶子，就可以消除 dep 项。

WC2018 通道（选讲）

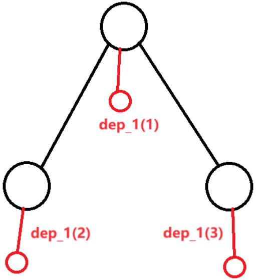


WC2018 通道（选讲）



设 dp_u 表示 u 的子树（第一棵树）内的结点，对应到第二棵树后的直径，利用直径合并的结论来实现转移，转移时顺便就求出了答案。

WC2018 通道（选讲）



设 dp_u 表示 u 的子树（第一棵树）内的结点，对应到第二棵树后的直径，利用直径合并的结论来实现转移，转移时顺便就求出了答案。

除了 ST 表预处理以外，复杂度是 $O(n)$ 的。

WC2018 通道（选讲）

接下来思考三棵树的情况。

WC2018 通道（选讲）

接下来思考三棵树的情况。

对第一棵树进行 **边分治（变种）**，把分治中心两边的点分别称为红点和蓝点，那么红点 u 和蓝点 v 在第一棵树上的距离为 $dis_1(u) + dis_1(v)$。

WC2018 通道（选讲）

接下来思考三棵树的情况。

对第一棵树进行 **边分治（变种）**，把分治中心两边的点分别称为红点和蓝点，那么红点 u 和蓝点 v 在第一棵树上的距离为 $dis_1(u)+dis_1(v)$。

把所有红点和蓝点在第二棵树上建立虚树，在虚树上固定 u,v 的 lca，目标是最大化 $dis_1(u)+dis_1(v)+dep_2(u)+dep_2(v)+d_3(u,v)$，$dis$ 和 dep 项都可以用添加叶子的方法消除，最后做一遍 dp 即可。

WC2018 通道（选讲）

接下来思考三棵树的情况。

对第一棵树进行 **边分治（变种）**，把分治中心两边的点分别称为红点和蓝点，那么红点 u 和蓝点 v 在第一棵树上的距离为 $dis_1(u) + dis_1(v)$。

把所有红点和蓝点在第二棵树上建立虚树，在虚树上固定 u, v 的 lca，目标是最大化 $dis_1(u) + dis_1(v) + dep_2(u) + dep_2(v) + d_3(u, v)$，$dis$ 和 dep 项都可以用添加叶子的方法消除，最后做一遍 dp 即可。

建虚树时需按 DFS 序排序，导致总复杂度为 $O(n\log^2 n)$，如果在边分治的同时进行归并排序，就可以把复杂度降为 $O(n\log n)$。

谢谢大家！