树上问题I

任飞宇

DFS

1. Blood Cousins

有一个家族关系树，描述了 n $(1 \leq n \leq 1e5)$ 人的家庭关系，成员编号为 1 到 n 。

如果 a 是 b 的父亲，那么称 a 为 b 的 1 级祖先；如果 b 有一个 1 级祖先，a 是 b 的 1 级祖先的 $(k-1)$ 级祖先，那么称 a 为 b 的 k 级祖先。

家庭关系保证是一棵树，树中的每个人都只有一个父母，且自己不会是自己的祖先。

如果存在一个人 z ，是两个人 a 和 b 共同的 p 级祖先：那么称 a 和 b 为 p 级表亲。

m $(1 \leq m \leq 1e5)$ 次询问，每次询问给出一对整数 v 和 p ，求编号为 v 的人有多少个 p 级表亲。

问题相当于询问某个点的子树里某个深度有多少点，也就是DFS序上某个区间里某个值出现了几次。

把区间表示成前缀和相减，从左往右扫一遍DFS序，维护cnt数组记录每个值出现几次。

2. Interstellar battle

给定一棵树与每个点消失的概率$1-p_i$，请实现以下两种操作：
•修改一个点消失的概率
•求树上连通块个数期望

首先分析如果表示连通块个数的期望。

用每个连通块的顶点来标记这个连通块，那么每个点对答案有贡献当且仅当该节点的父节点消失了且自己没有消失。得到$E=\sum_{u\neq rt}(1-p_{fa[u]})p_u+p_{rt}$
另一种思路可以考虑森林的连通块数量=点数-边数，也能得到$E=\sum_i p_i-\sum_{(u,v)}p_up_v$。

修改点u时会影响u到父亲的边和u到儿子的边。
u到父亲的边不难处理，答案减去旧值加上新值。
维护每个点儿子的p值之和sum_u，就可以处理u到儿子的边，答案减去$-sum_u*p_u$的旧值，加上新值。
然后维护$sum_{fa[u]}$和p_u的值。

3. 异或

现在有一颗以 1 为根节点的由 n 个节点组成的树，节点从 1 至 n 编号。树上每个节点上都有一个权值 v_i 。现在有 q 次操作，操作如下：

- $1\ x\ z$：查询节点 x 的子树中的节点权值与 z 异或结果的最大值。
- $2\ x\ y\ z$：查询节点 x 到节点 y 的简单路径上的节点的权值与 z 异或结果最大值。

DFS序可以处理操作1，在DFS序上从左往右建持久化Trie，在作差得到的Trie上查询。

对于操作2，从根往下建持久化Trie，在作差得到的Trie上查询。

4. Fusion tree

魔法森林里有一颗大树，下面经常有小孩召开法。

大树可以看做一个有 n 个节点，$n-1$ 条边的无向连通图。大树的每个节点都有若干瓶矿泉水，初始第 i 个节点有 a_i 瓶矿泉水。

麦杰斯住在大树顶端，有一天他想改造一下大树，方便他巨大多喝水之后可以垃圾分类矿泉水瓶。

麦杰斯喜欢二进制运算，所以他会有以下三种操作：

1. 将树上与一个节点 x 距离为 1 的节点上的矿泉水数量 $+1$。这里树上两点间的距离定义为从一点出发到另外一点的最短路径上边的条数。
2. 在一个节点 x 上喝掉 v 瓶水。
3. 询问树上与一个节点 x 距离为 1 的所有节点上的矿泉水数量的异或和。

麦杰斯共有 m 次操作，他希望你在每次 3 操作后告诉他答案。

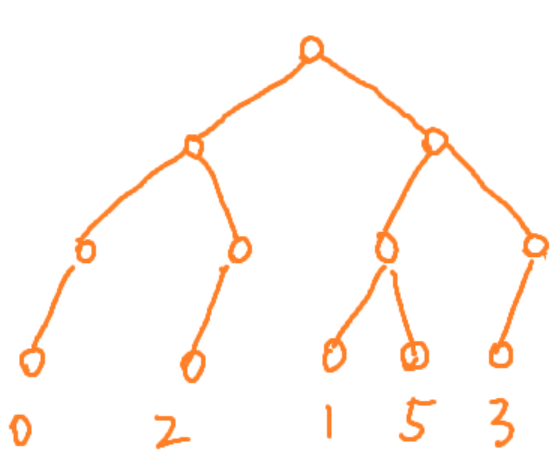
4. Fusion tree

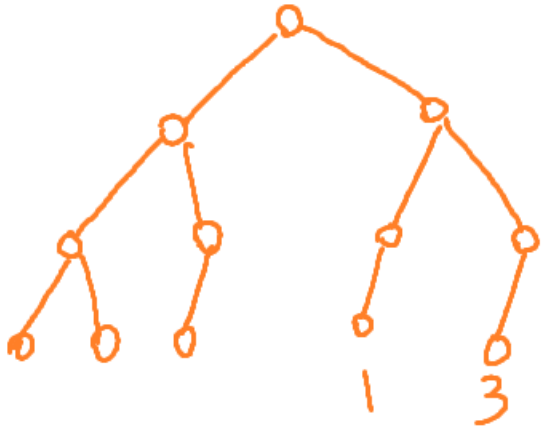
转为有根树，父节点单独处理，维护每个点的所有子节点，需要支持单点减、整体加一、整体求异或和。每个节点维护一个Trie来支持这些操作。

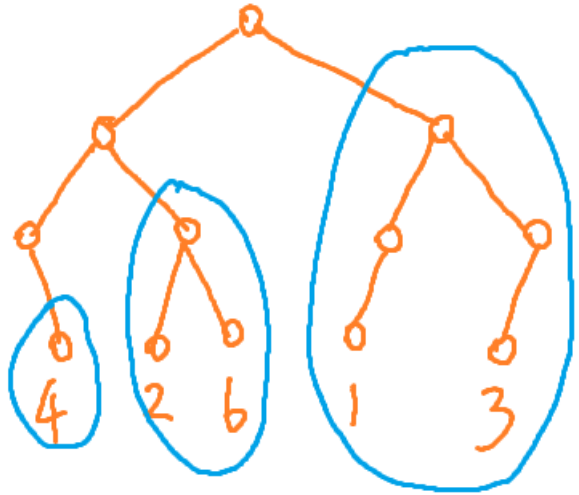
考虑把所有数倒序插入Trie中，那么单点减可以用删除一个数、插入一个数来实现，每个节点维护子树异或和。整体加一操作相当于在Trie树中进行一系列交换。

考虑+1对左子树里所有数的影响，会把最后一位变为1，对右子树里所有数的影响是最后一位变为0然后进位，所以需要交换左右子树。

接下来处理进位，发现这是一个递归的过程。







5. regions

联合国区域发展委员会（The United Nations Regional Development Agency，UNRDA）有一个良好的组织结构。

它任用了 N 名委员，每名委员都属于几个地区中的一个。

委员们按照其资历被编号为 1 到 N，1 号委员是主席，资历最高。委员所属地区被编号为 1 到 R。

除了主席之外所有委员都有一个直接导师。任何直接导师的资历都比他所指导的委员的资历要高。

我们说委员 A 是委员 B 的导师当且仅当 A 是 B 的直接导师或者 A 是 B 的直接导师的导师。显然，主席是所有其他委员的导师，没有任何两名委员互为导师。

现在，联合国区域发展委员会想要建立一个计算机系统：在给定委员之间的直接导师关系的情况下，该系统可以自动地回答下述形式的问题：给定两个地区 r_1 和 r_2，要求系统回答委员会中有多少对委员 e_1 和 e_2，满足 e_1 属于 r_1，而 e_2 属于 r_2，并且 e_1 是 e_2 的导师。

$1 \leq N \leq 2 \times 10^5$，$1 \leq R \leq 2.5 \times 10^4$，$1 \leq Q \leq 2 \times 10^5$

5. regions

考虑询问(r_1,r_2)，如果所属地区为r_2的点不多，是否有较好的做法？

大小分治，如果所属地区为r_2的点不超过$\sqrt{n}$个，那么我们可以将询问(r_1,r_2)分解到这些点上，每个点上挂有一个询问形为：该点上方有多少所属地区为r_1的点？

离线做，维护dfs路径上的信息。

如果所属地区为r_2的点超过$\sqrt{n}$个，那么这样的r_2不会超过$\sqrt{n}$种。

我们可以将询问(r_1,r_2)分解到所属地区为r_1的点上，分解为不超过$n\sqrt{n}$个询问形如：
该点下方有多少所属地区为r_2的点？

转为询问DFS序上的区间，转为前缀和相减，离线从左往右扫，维护每个地区的点数。

也可以按照所属地区为r_1的点数来大小分治，做法同理。

6. Bessie's Snow Cow

农场下雪啦！Bessie 和往年开冬一样在堆雪牛。她之前是个写实派，总是想把她的雪牛堆得和个真牛一样。但今年不一样，受到来自东方的神秘力量的影响，她想来点抽象艺术，因此她想堆成一棵树的样子。这棵树由 N 个雪球，$N-1$ 根树枝构成，每根树枝连接两个雪球，并且每两个雪球之间路径唯一。

Bessie 要给她的雪牛来点细节。因此她给其中一个雪球加了个鼻子，来表示这是他那抽象的牛的头，并且把它称作雪球 1。为了让雪牛更好看，她还要给某些雪球来点不同的颜色。于是，她用旧牛奶桶装满了颜料泼到雪牛上。这些颜料分别被编号为 $1, 2, \ldots 10^5$，且每种颜色都无限量供应。

当 Bessie 把一桶颜料泼到一个雪球上时，这个雪球子树上的所有雪球也会被染色（我们称雪球 y 在雪球 x 的子树里当且仅当雪球 x 处在雪球 y 到雪球 1 的路径上）。Bessie 有着精确的泼颜料技术，因此在泼完一种颜料后，一个雪球上之前被染过的所有颜色依然清晰可见。例如，一个雪球之前显现出来颜色 $[1, 2, 3]$，然后 Bessie 把装有 4 号颜色的牛奶桶泼上去，那么这个雪球将显现出来颜色 $[1, 2, 3, 4]$。 在泼了几桶颜料以后，Bessie 可能想要了解她的雪牛有多五彩斑斓。令雪球 x 的『颜色丰富度』为这个雪球被染上的不同颜色总数，当 Bessie 想了解雪球 x 的相关信息时，你应该回答她雪球 x 的子树中所有的雪球的颜色丰富度之和。

6. Bessie's Snow Cow

每种颜色分开看，不妨先只看一种颜色。

任意时刻，树上染颜色c的点一定是若干颗子树。
这些子树里的点丰富度要+1。

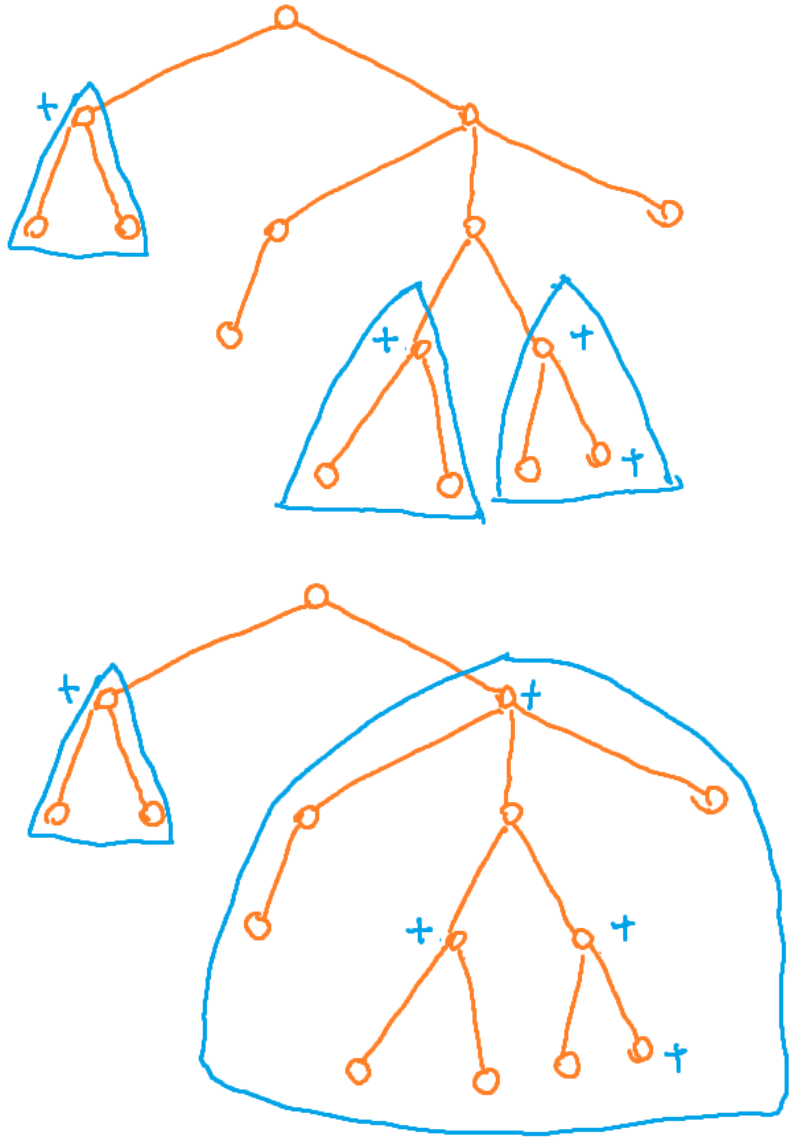
我们维护这些子树的根。

向点x泼颜料时，先判断这个点上面是否已经被泼过。
在DFS序上找上一个泼颜色c的点，如果是x的祖先，那么这次操作就可以跳过。

如果x上面没被泼过，那么x会成为新子树的根，x下面的子树需要删掉。

在DFS序上找x下面的子树，用set的upper_bound实现。

每颗子树里的点丰富度要+1，在DFS序上维护区间加减区间求和即可。



树上差分

有时求树上一条链的数字之和，我们可以维护每个点到根的数字之和，然后作差；

如果需要给树上一条链全部加上某个东西，可以在链的底部打一个加标记，顶部打一个减标记，每个点的值就是该点的子树和。

7. 运输计划

公元 2044 年，人类进入了宇宙纪元。

L 国有 n 个星球，还有 $n-1$ 条双向航道，每条航道建立在两个星球之间，这 $n-1$ 条航道连通了 L 国的所有星球。

小 P 掌管一家物流公司， 该公司有很多个运输计划，每个运输计划形如：有一艘物流飞船需要从 u_i 号星球沿最快的宇航路径飞行到 v_i 号星球去。显然，飞船驶过一条航道是需要时间的，对于航道 j，任意飞船驶过它所花费的时间为 t_j，并且任意两艘飞船之间不会产生任何干扰。

为了鼓励科技创新， L 国国王同意小 P 的物流公司参与 L 国的航道建设，即允许小 P 把某一条航道改造成虫洞，飞船驶过虫洞不消耗时间。

在虫洞的建设完成前小 P 的物流公司就预接了 m 个运输计划。在虫洞建设完成后，这 m 个运输计划会同时开始，所有飞船一起出发。当这 m 个运输计划都完成时，小 P 的物流公司的阶段性工作就完成了。

如果小 P 可以自由选择将哪一条航道改造成虫洞， 试求出小 P 的物流公司完成阶段性工作所需要的最短时间是多少?

7. 运输计划

首先二分一个时间，判断能否把一条边长设为0，使得所有链长小于等于二分的值。

那些已经满足的链显然不用管。要选的边一定属于还没满足的链的交集。

于是我们要找的边就是还没满足条件的链的交集中，边长最大的边。允许用$O(n)$复杂度。

给链上所有边+1表示被这条链覆盖了，那么被覆盖次数=不满足条件的链数，就表示边在交集里

找到边后验证一下即可。

8. 天天爱跑步

小c 同学认为跑步非常有趣，于是决定制作一款叫做《天天爱跑步》的游戏。《天天爱跑步》是一个养成类游戏，需要玩家每天按时上线,完成打卡任务。

这个游戏的地图可以看作一一棵包含 n 个结点和 $n-1$ 条边的树，每条边连接两个结点,且任意两个结点存在一条路径互相可达。树上结点编号为从 1 到 n 的连续正整数。

现在有 m 个玩家，第 i 个玩家的起点为 s_i，终点为 t_i。每天打卡任务开始时，所有玩家在第 0 秒同时从自己的起点出发，以每秒跑一条边的速度，不间断地沿着最短路径向着自己的终点跑去，跑到终点后该玩家就算完成了打卡任务。(由于地图是一棵树，所以每个人的路径是唯一的)

小c 想知道游戏的活跃度，所以在每个结点上都放置了一个观察员。在结点 j 的观察员会选择在第 w_j 秒观察玩家，一个玩家能被这个观察员观察到当且仅当该玩家在第 w_j 秒也正好到达了结点 j 。 小c 想知道每个观察员会观察到多少人?

注意：我们认为一个玩家到达自己的终点后该玩家就会结束游戏，他不能等待一 段时间后再被观察员观察到。 即对于把结点 j 作为终点的玩家：若他在第 w_j 秒前到达终点,则在结点 j 的观察员不能观察到该玩家；若他正好在第 w_j 秒到达终点,则在结点 j 的观察员可以观察到这个玩家。

8. 天天爱跑步

转为有根树考虑，每个玩家可以分为上行和下行两部分。

如果一个起点为s_i的上行玩家对点j有贡献，则$d[s_i]-w[j]=d[j]$。

一个上行玩家对链上满足$w[j]+d[j]=d[s_i]$的所有点j有贡献。

使用树上差分，把玩家挂在链的两端，对每个点求子树内满足等式的玩家个数

转dfs序，即求区间内满足等式的玩家个数。

因为可以离线，考虑把询问的区间表示为前缀和相减，拆成两个询问，从左往右扫一遍

重链剖分

我们给出一些定义：

定义 **重子节点** 表示其子节点中子树最大的子结点。如果有多个子树最大的子结点，取其一。如果没有子节点，就无重子节点。

定义 **轻子节点** 表示剩余的所有子结点。

从这个结点到重子节点的边为 **重边**。

到其他轻子节点的边为 **轻边**。

若干条首尾衔接的重边构成 **重链**。

把落单的结点也当作重链，那么整棵树就被剖分成若干条重链。

重链剖分

重链剖分的性质

树上每个节点都属于且仅属于一条重链。

重链开头的结点不一定是重子节点（因为重边是对于每一个结点都有定义的）。

所有的重链将整棵树 **完全剖分**。

在剖分时 **重边优先遍历**，最后树的 DFN 序上，重链内的 DFN 序是连续的。按 DFN 排序后的序列即为剖分后的链。

一颗子树内的 DFN 序是连续的。

可以发现，当我们向下经过一条 **轻边** 时，所在子树的大小至少会除以二。

因此，对于树上的任意一条路径，把它拆分成从 lca 分别向两边往下走，分别最多走 $O(\log n)$ 次，因此，树上的每条路径都可以被拆分成不超过 $O(\log n)$ 条重链。

9. 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 m 次询问，每次询问给出 l r z，求 $\sum_{i=l}^{r} dep[LCA(i,z)]$ 。

9. LCA

$dep[LCA(x,y)]$可以表示为在x到根的路径上打上1标记后，y到根路径上的和。

对于一个询问$\sum_{i=l}^{r} dep[LCA(i,z)]$，如果把z到根标上1，我们需要快速得到很多链的和，不好处理。

尝试把i到根标上1，求z到根的和。

离线问题可以把区间转为前缀和的差。于是问题转化为链加减、求链和，使用树链剖分解决。

10. 保卫王国

Z 国有 n 座城市，$(n-1)$ 条双向道路，每条双向道路连接两座城市，且任意两座城市都能通过若干条道路相互到达。

Z 国的国防部长小 Z 要在城市中驻扎军队。驻扎军队需要满足如下几个条件：

- 一座城市可以驻扎一支军队，也可以不驻扎军队。
- 由道路直接连接的两座城市中至少要有一座城市驻扎军队。
- 在城市里驻扎军队会产生花费，在编号为 i 的城市中驻扎军队的花费是 p_i。

小 Z 很快就规划出了一种驻扎军队的方案，使总花费最小。但是国王又给小 Z 提出了 m 个要求，每个要求规定了其中两座城市是否驻扎军队。小 Z 需要针对每个要求逐一给出回答。具体而言，如果国王提出的第 j 个要求能够满足上述驻扎条件（不需要考虑第 j 个要求之外的其它要求），则需要给出在此要求前提下驻扎军队的最小开销。如果国王提出的第 j 个要求无法满足，则需要输出 -1。现在请你来帮助小 Z。

10. 保卫王国

首先，最小点权覆盖=全集−最大点权独立集。

如果规定某点一定要选，给该点权值$+\infty$，如果规定某点一定不选，该点权值设为$-\infty$。

树上的最大点权独立集可以dp求解。$f_{x,0/1}$表示x选/不选，x为根的子树的最大点权独立集。

这里我们稍微修改下dp的顺序。树链剖分后，求解当前重链节点的f值前，先求下面的所有重链。于是当前重链节点的所有轻儿子的f值都有了。

对当前重链里的节点u，我们可以先处理$g_{u,0}=\sum_v \max(f_{v,0},f_{v,1}),g_{u,1}=\sum_v f_{v,0}$，其中$v$是$u$的轻儿子。

再考虑节点u的重儿子h，$f_{u,0}=g_{u,0}+\max(f_{h,0},f_{h,1}),f_{u,1}=g_{u,1}+f_{h,0}+a_u$。

从f_h到f_u的转移可以表示成关于g_u的矩阵。用线段树维护重链上矩阵的乘积。

考虑修改，从修改的点向上，每次修改重链中某个位置的矩阵，用线段树快速得到重链上所有矩阵的乘积，也就得到了重链顶点的f，修改顶点父亲的g，然后继续改上面的重链。

树上启发式合并

11. Lomsat gelral

给出一个树，求出每个节点的子树中出现次数最多的颜色，如果有多个颜色出现次数最多，给出它们的编号和。

$1 \leq n \leq 10^5, 1 \leq c_i \leq n.$

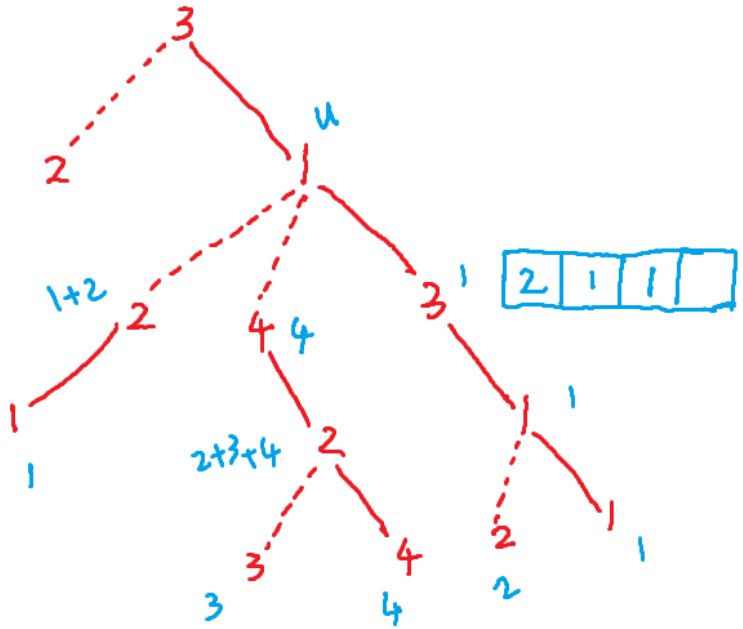
11. Lomsat gelral

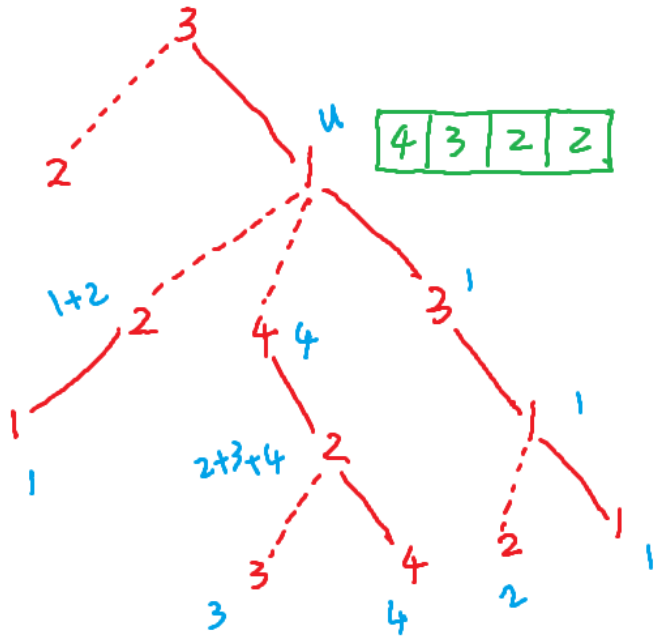
从下往上为每个节点的子树求一个数组，表示每种颜色的出现次数。点u的数组可以用它儿子的数组合并得到。
但是如果迁移所有儿子的数组到当前点的数组，复杂度就是n^2了，而且我们开不了这么多的数组。
所以我们只保留重儿子的数组，遍历轻子树里的每个点，加到数组里，得到当前点的数组。

定义dfs(u)过程为求u的子树里每个点的答案：

1. 递归u的儿子dfs(v)，
   并保留重儿子的数组
2. 扫一遍轻子树的点，
   加到数组里

整个过程的复杂度是$nlogn$的
考虑每个点在step2里被扫的次数，
等于该点到根的轻边数量





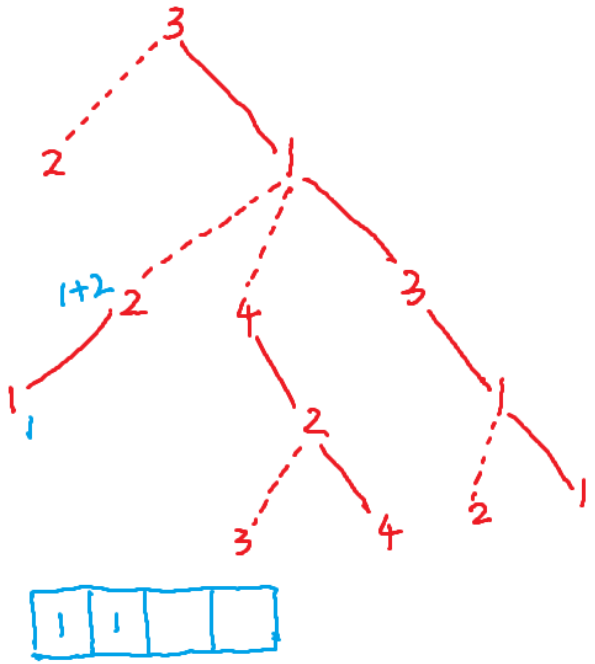
11. Lomsat gelral

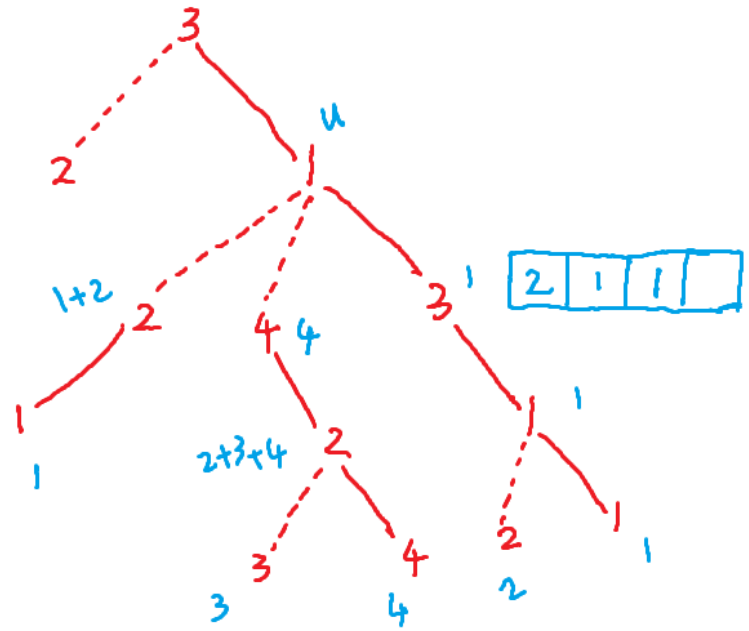
想实现整个过程，我们全程只需要一个数组。

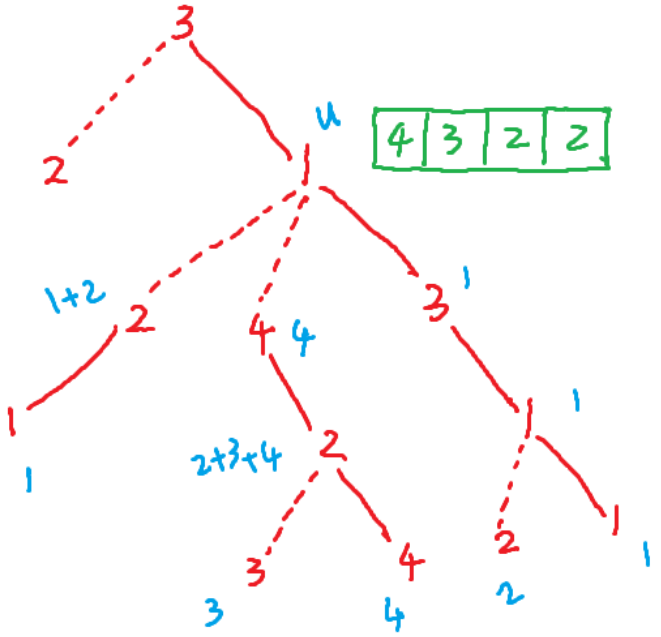
在$dfs(u)$里，优先递归$dfs(u$的轻儿子$v_i)$，在$dfs(v_i)$结束时，数组代表了v_i的子树，
轻子树的信息不保留，所以我们清空数组。注意这里清空数组不能扫整个数组，而是扫数组对应的轻子树

然后递归$dfs(u$的重儿子$h)$，在$dfs(h)$结束时，数组代表了h的子树，保留这个数组。

然后遍历轻子树里的点以及点u加入数组中，此时数组代表了u的子树，得到点u的答案，$dfs(u)$结束，返回







11. Lomsat gelral

```
void add(int x, int fa, int val) {
    cnt[col[x]] += val;
    if(cnt[col[x]] > Mx) Mx = cnt[col[x]], sum = col[x];
    else if(cnt[col[x]] == Mx) sum += (LL)col[x];
    for(int i = 0; i < v[x].size(); i++) {
        int to = v[x][i];
        if(to == fa || to == Son) continue;
        add(to, x, val);
    }
}
void dfs2(int x, int fa, int opt) {
    for(int i = 0; i < v[x].size(); i++) {
        int to = v[x][i];
        if(to == fa) continue;
        if(to != son[x]) dfs2(to, x, 0);
    }
    if(son[x]) dfs2(son[x], x, 1), Son = son[x];

    add(x, fa, 1); Son = 0;
    ans[x] = sum;
    if(!opt) add(x, fa, -1), sum = 0, Mx = 0;
}
```