1

树的经典问题和方法

宁波市镇海蛟川书院 杨明天

目录

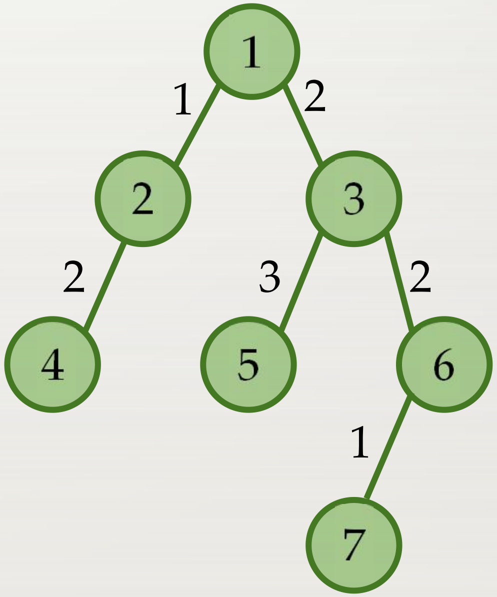
1.基本定义与方法

1.1.树的序列表示

- 一些竞赛题目中，往往可以通过把树用序列表示出来，将树上问题转化为序列问题。
- 常见的序列表示包括：DFS序、欧拉序列、括号序列、树链剖分序列和Prüfer序列。

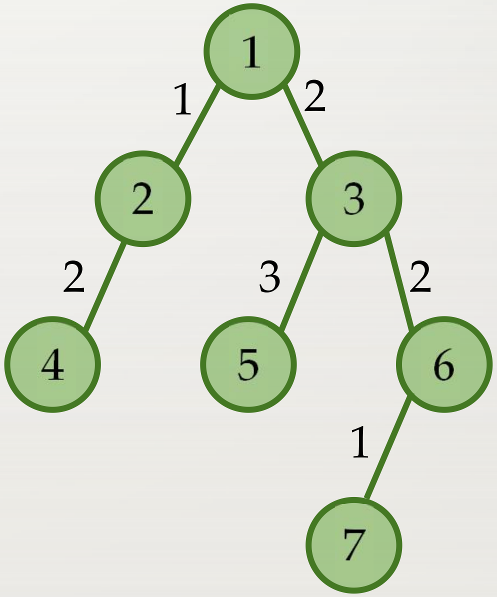
1.1.1.DFS序

- 从根结点开始DFS，访问到一个结点就将该结点加入序列中，得到的长度为n的序列就是树的DFS序。
- 如图，该树的一个DFS序为：
  - 1-2-4-3-5-6-7



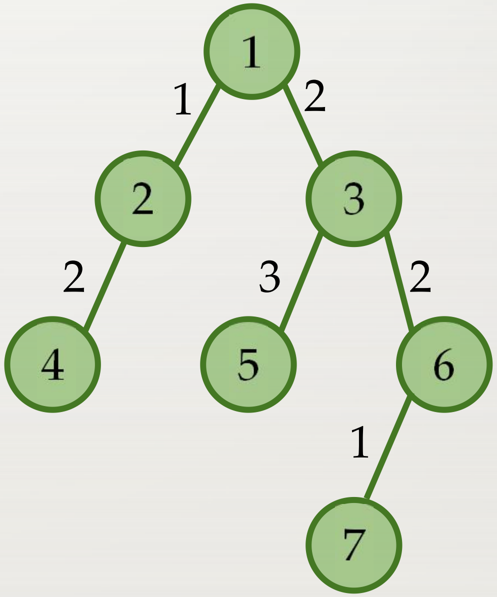
1.1.2.Euler序列定义

- 从根结点开始DFS，无论是递归还是回溯，每次到达一个结点时都将编号记录下来，求出的带一个长度为2n-1的序列，即为Euler序列。
- 如图，该树的一个Euler序列为：
    - 1-2-4-2-1-3-5-3-6-7-6-3-1



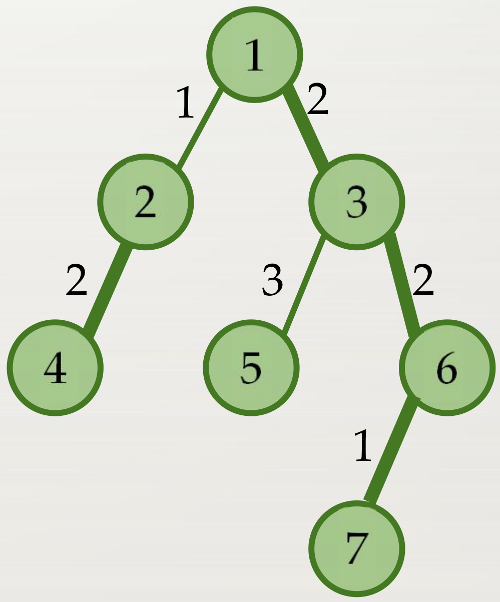
1.1.3.括号序列
定义

- 从根结点开始DFS。进入结点时添加一个左括号，退出结点时添加一个有括号。所得到的长度为2n的序列就是树的括号序列。
- 如图，该树的一个括号序列为：
  - ((())(()(())))
- 将其用结点编号表示，就是：
  - 1-2-4-4-3-3-5-5-6-7-7-6-3-1



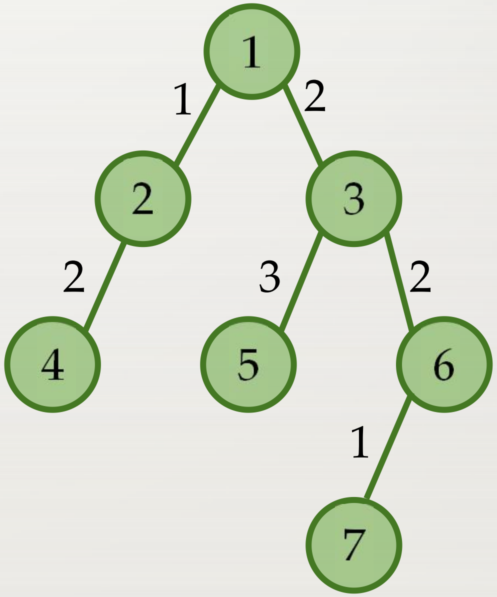
1.1.4.树链剖分序列

- 进行树链剖分时，通过先访问重子树，再访问其它子树，得到的长度为n的序列就是树链剖分序列。
- 树链剖分序列也是DFS序的一种。
- 如图，该树的一种树链剖分序列为：
  - 1-3-6-7-5-2-4
- 树链剖分序列可以保证同一条重链上的结点一定在一起，往往使用线段树维护信息。



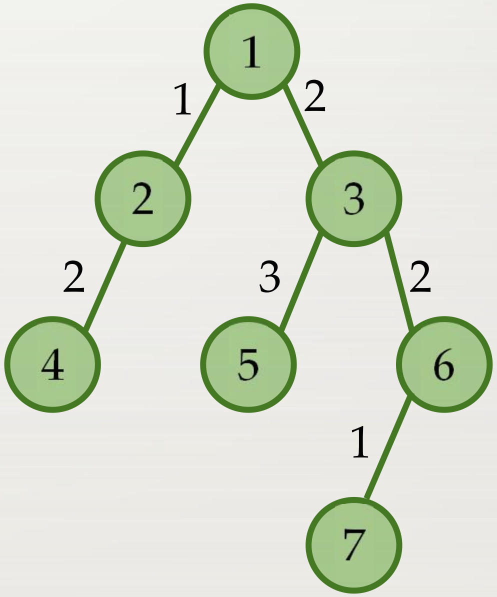
1.1.5.Prüfer序列

- 每次找到编号最小的叶子结点，将其从树上删去，并将与其相邻的结点加入序列，不断重复，直到只剩下2个结点。最后得到一个长度为n-2的序列就是Prüfer序列。
- 如图，该树的Prüfer序列为：
  - 2-1-3-3-6
- 不难发现，构造Prüfer序列的方法适用于无根树，且一棵树的Prüfer序列是唯一确定的。



1.2.树的重心
定义

- 对于一棵n个结点的树，找到一个点，使得把树变成以该点为根的有根树时，最大子树的节点数最小。我们将满足这一条件的点称作树的重心。
- 如图，树的重心是3。

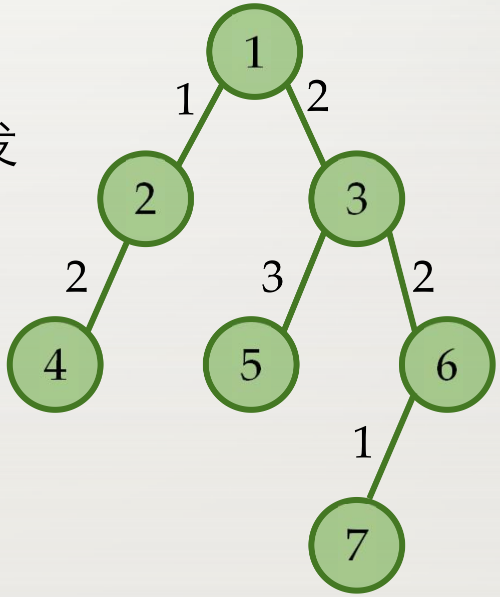


1.2.树的重心 方法

- 首先从随便一个点开始DFS，设用d(i)表示以i为根的结点个数，j为i的子结点，不难发现$d(i)=1+\Sigma d(j)$。
- 对于在i点以上的那个子树，它的结点个数即为n-d(i)。
- 这样，对于每个i我们就可以求出去掉i点后，每个子树的大小。
- 而子树大小最大值最小的那个点i，就是树的重心。
- 显然树的重心要么只有1个，要么为2个邻接的点。

1.3.树的直径、中心和半径定义

- 对于一棵n个结点的无根树，找到一条尽可能长的路径。我们将这样的路径称作树的直径。
- 如图，4-5和4-7都是树的直径。
- 树的中心是树上一点，满足从该点出发的最长路径最短。这样的路径为半径。
- 如图，1和3都是树的中心，树的半径是5。

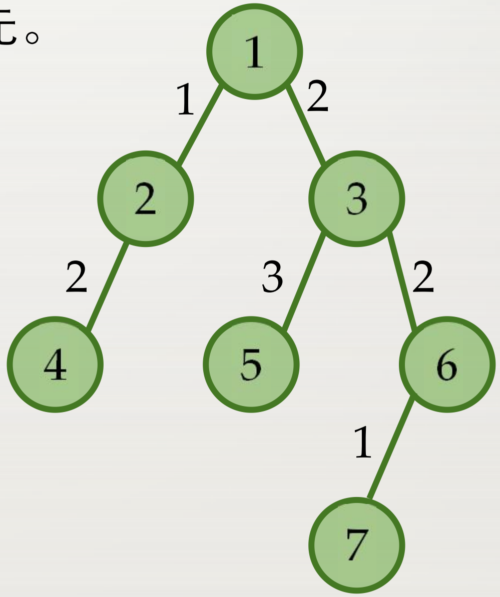


1.3.树的直径、中心和半径
方法

- 首先从随便一个点开始DFS，到达它能到达的最远的点u。
- 然后从点u开始DFS，到达它能到达的最远的点v。
- 路径u-v即为树的直径。
- 在直径u-v上枚举点w，使得max(dis(u,w),dis(v,w))尽可能小，点w即为树的中心。此时max(dis(u,w),dis(v,w))就是半径长度。
- 显然树的直径和中心、和半径也不一定唯一，但直径和半径的长度是唯一的。中心要么只有1个，要么为2个邻接的点。

1.4.最近公共祖先
定义

- 对于一棵n个结点的树，对于两个结点u和v，若存在一个点w，使得w既是u的祖先，也是v的祖先，且w到u和v的距离尽可能近。我们将w称作u和v的最近公共祖先。
- 如图，5和7的最近公共祖先是3。



1.4.最近公共祖先
算法1：倍增

- 对于树上所有结点构造稀疏表，用anc[x][k]表示x的2^k次祖先。
- 显然anc[x][k]=anc[anc[x][k-1]][k-1]。
- 可以用O(n log n)的时间预处理出稀疏表，对于每次的询问，u和v中深度较大的结点沿着自己的祖先向上跳，直到跳到LCA。
- 每次询问的时间复杂度是O(log n)的。

1.4.最近公共祖先
算法2：RMQ

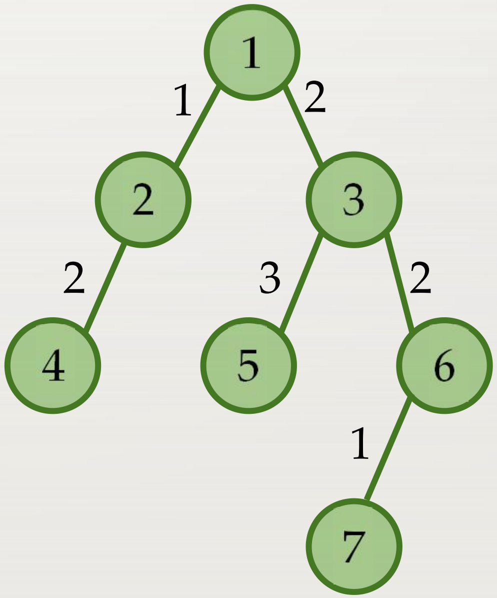
- 首先，按照结点的DFS序得到序列vis[i]和对应的深度dep[i]。
- 对于每个顶点v，记其在vis中首次出现的下标为id[v]。
- 这些只需要O(n)的时间即可求得。而LCA(u,v)就是访问u之后到访问v之前所经过顶点中离根最近的那个，假设id[u]≤id[v]，那么有：
  - LCA(u,v)=vis[id[u]≤i≤id[v]中令depth(i)最小的i]
- 而这可以利用RMQ在O(n)的时间内预处理，在O(1)的时间内求得。

17

1.4.最近公共祖先
算法2：RMQ

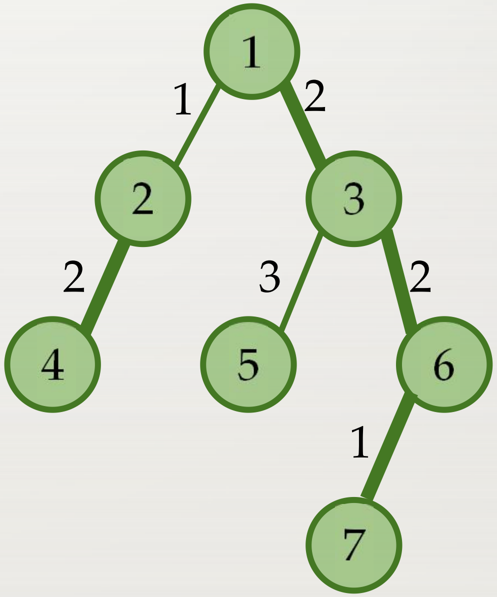
i	1	2	3	4	5	6	7	8	9	10	11	12	13
vis	1	2	4	2	1	3	5	3	6	7	6	3	1
dep	1	2	3	2	1	2	3	2	3	4	3	2	1

i	1	2	3	4	5	6	7
id	1	2	6	3	7	9	10



1.4.最近公共祖先
算法3：树链剖分

- 对直接对原树进行树链剖分，类似于倍增的做法，用链顶端位置较深的点往上跳。
- 如图，加粗的是重链，
  若查询u=4和v=7的LCA，
  则跳转过程如下：
  - 1.u从4跳到1；
  - 2.发现1和7在同一条链上，
    则LCA(4,7)=1。



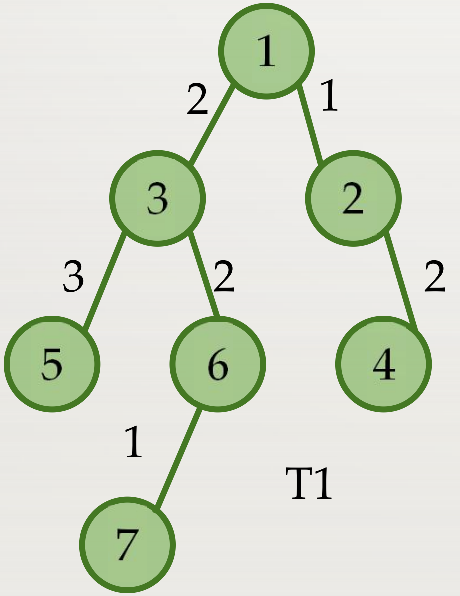
19

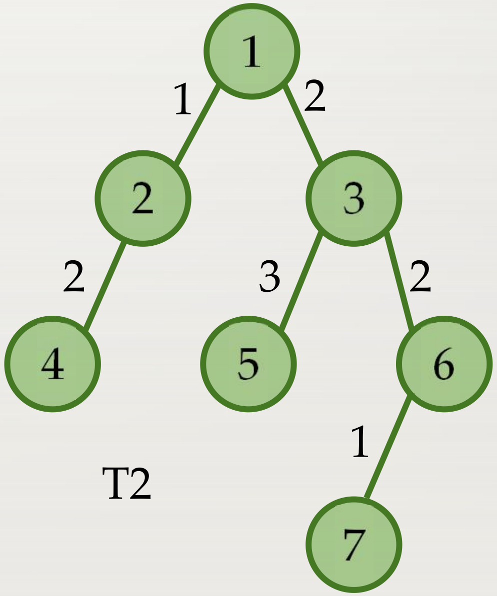
1.4.最近公共祖先
算法4：Tarjan

- 用于离线求解LCA问题。
- 从根结点往下DFS，每次用并查集将当前结点u合并到父结点par所在的集合中。
- 对于当前结点u，判断一下与其相关的询问中，另一个结点y是否已经被访问过。如果是，那么这时v在并查集中的祖先即为LCA。

1.5.树的同构

- 给定两个数T1和T2，如果不考虑子结点的顺序，T1与T2相同，那么我们称T1和T2为同构树。
- 如图，T1和T2是同构的。





21

1.5.树的同构
判定方法1：最小表示法

- 括号序列对应一棵唯一的树，但树的括号序列是不唯一的。因此，我们可以取字典序最小的括号序列。我们把这样的括号序列称为树的最小表示。
- 如果两棵树的最小表示相同，那么这两棵树一定是同构的，我们把用树的最小表示判断树的同构的方法称为最小表示法。

1.5.树的同构判定方法2：Hash

- 每次递归对子树进行Hash，再用来计算母树的Hash值。
- 计算母树时首先对于子树的Hash值排序，因为子结点顺序的不同并不会导致树的不同。而计算顺序的不同却会导致最后Hash值不同。

23

1.5.树的同构
无根树的同构判定

- 上面介绍的两种方法都只适用于有根树，如何通过一些转化，将相同的方法是用于无根树呢？
- 考虑通过将无根树转化成有根树来判定同构。
- 我们需要先选出一个点作为根，使得这两个点是同构树上的对应点。
- 显然我们没法找出这样的点。
- 再退一步想，是否可以找出为数不多的几个点，使得这些点中肯定会有一对点对应呢？

1.5.树的同构
无根树的同构判定

- 考虑重心/中心的性质。
- 在一棵树中，重心/中心至多2个，我们可以先求出两棵树的重心/中心，然后暴力枚举哪两个是对应点即可，最多枚举4次。
- 事实上，如果树是同构的，那么这些点一定是两两对应的，所以我们只需要试2次就能够知道两棵树是否同构。

2.习题选讲

2.1.[TJOI2017]城市

- 题目大意：
  - 给你一棵n个结点的带边权的树，请你改变其中一条边的位置，使得剩下的图仍是一棵树，并使树的直径尽可能小。求改造后树的直径。
- 数据范围：
  - $1 \le n \le 5{,}000$。

2.1.[TJOI2017]城市 Solution

- 枚举每条边e，并计算替换这条边以后的树的最小直径。
- 显然，每次去掉边e时，会把整棵树分成两棵不相交的新树。
- 对于每棵新树，可以分别求出它们的直径d1,d2，半径r1,r2。
- 那么修改这条边后，树的最小直径是max(d1,d2,r1+e+r2)。
- 优化：要修改的边一定在树的直径上，因此我们可以先求出树的直径，再枚举直径上的边。
- 时间复杂度：$O(n^2)$。

28

2.2.[SDOI2011]消防

- 题目大意：
  - 给你一棵n个结点的带边权的树，让你找出一条长度不超过s的链，使得结点到链的最大距离最小，求这个距离的最小值。
- 数据范围：
  - n≤300,000。

29

2.2.[SDOI2011]消防 Solution

- 不难发现链在直径上的情况一定是最优的。
- 首先找出这个直径，然后二分答案m。
- 二分的下界为结点到直径距离的最大值，上界为直径的长度。
- 对于每一个m，把直径往里缩，使得两边缩的长度均≤m。
- 判断一下缩完以后的直径是不是≤s。

2.3.[SDOI2013]直径

- 题目大意：
  - 给你一棵n个结点的带边权的树，求该树直径必经边的个数。
- 数据范围：
  - n≤200,000。

2.3.[SDOI2013]直径 Solution

- 显然直径必经的所有边的肯定是一个直径上的连续一段。
- 首先求出原树的任一直径。
- 预处理出该直径上从结点i出发，不经过直径上其它结点的最长链长度far[i]。
- 从直径的两端往里缩，如果当前缩到的点i的far[i]等于i到被缩的那一端的距离，就说明直径的这一段至少有两种不重合的情况，肯定不是必经部分。
- 最后看一下中间没有被缩的部分经过了几个边。

2.4.[IOI2013]Dreaming

- 题目大意：
  - 有n个点，由m条边连接，第i条边的边权是wi。这些点和边构成了一个森林。你必须要新建若干条条权值为W的边，使得原图恰好变成一棵树，并且让任意两个点间最长距离最短。求该距离。
- 数据范围：
  - $1 \le n \le 500,000$。

2.4.[IOI2013]Dreaming Solution

- 首先找出每棵树的直径和中心及其对应半径。
- 加边的过程一定是在这些中心之间加边。
- 考虑这些中心的连接方式。
- 选择一个中心，让其它中心都直接连上这个点肯定是最优的。
- 则答案要么是原树中的直径，要么是加边以后的新的直径。
- 新的直径分两种情况，一种是最大半径和次大半径直径直接用一条新边相连，另一种是次大边和第三大边用两条新边相连。
- 时间复杂度O(n)。

2.5.[CF911F]Tree Destruction

- 题目大意：
  - 给你一棵n个结点的树，每次选取两个叶子结点，往答案中加上它们的距离，并删去其中一个结点。你可以自由进行上述操作，使得最后答案最大。问答案最大是多少，并输出任意一种方案。
- 数据范围：
  - $n \le 200,000$。

37

2.5.[CF911F]Tree Destruction Solution

- 贪心。
- 首先找出树的直径，然后枚举直径外的叶子结点。
- 看一下该结点到直径两端距离哪个长，加上这个距离，并删去这个点。
- 最后删直径上的点。
- 每次存一下方案，最后直接输出即可。
- 时间复杂度O(n)。

2.6.[HDU5732]Subway

- 题目大意：
  - 给你两棵n个结点的同构的无根树，求两树结点间可能的一种对应关系。
- 数据范围：
  - $n \le 100,000$。

2.6.[HDU5732]Subway Solution

- 由题意得，这是一组同构无根树。
- 考虑无根树的同构判定方法。
- 先枚举重心/中心求出一组对应点。然后求出每一个子树的Hash值，找到结点的对应关系。

2.7.[CEOI2017]One-Way Streets

- 题目大意：
  - 给你一个n个点、m条边无向图，现在告诉你q个点对(u,v)，要你在保证从u到v的所有路径都不消失的情况下，尽可能把所有的边变成单向边。问你可以唯一确定哪些边的方向，以及方向是从u到v还是从v到u。
- 数据范围：
  - n,m,q≤100,000。

41

2.7.[CEOI2017]One-Way Streets Solution 1

- 不难发现环上的边都不能确定方向，所以可以先Tarjan缩环。
- 然后就变成一棵树，考虑对树进行一些操作来确定边的方向。
- 从u到v的路径可以拆成两段：u->lca,lca->v。
- 树链剖分，用线段树维护边是从上到下还是从下到上。
- 如果发现当前维护的边与已知方向相反，那么就是双向边。
- 最后统计答案时，只需判一下边(x,y)是x在上还是y在上即可。

2.8.[HAOI2015]树上操作

- 题目大意：
  - 给你一棵n个点的带点权的树，按顺序进行如下操作共m次：
    - 1.将某个点的权值增加a；
    - 2.将以某个点为根的子树中的所有权值增加a；
    - 3.询问某个点到根的所有点权和。
- 数据范围：
  - $n,m \le 100000$。

2.8.[HAOI2015]树上操作 Solution 1

- 比模板还要水的树剖裸题。
- 操作1：单点修改。
- 操作2：一个子树中的结点在树链剖分序列中的位置肯定是连续的，因此相当于一个区间修改。
- 操作3：往根结点上跳，同时统计答案即可。

2.8.[HAOI2015]树上操作 Solution 2

- 题目求的是点到根路径上的权值和，那么修改单点对整棵子树所有结点的询问都有贡献。
- 给整棵子树增加权值显然也会对整棵子树的询问产生贡献。
- 若v是u的子树中的一个点，对结点u为根的子树的每一个结点权值增加a，相当于给针对v的询问答案增加了(dis(u,v)+1)a。
- 考虑将树“压扁”，求出树的DFS序。
- 对于结点u的子树，DFS序范围是dfn[u]~dfn[u]+size[u]-1。
- 用线段树维护两个值x和y，分别代表需要乘dis的权值和不需要乘dis的权值，询问时返回dep[u]*x[u]+y[u]即可。

2.8.[HAOI2015]树上操作 Solution 3

- 同样还是一种将树“压扁”的方法。
- 考虑我们之前讲过的括号序列。
- 左括号进，右括号出，那么在左括号上维护一个正的权值，右括号上维护一个负的权值，操作3中的询问就变成了前缀和。
- 按照括号序列建线段树，操作1就变成了单点修改，操作2就变成了区间+a/-a。