问题概述

差分一般是用来解决修改多而查询少的区间问题，而树上差分顾名思义就是把差分的思想应用到树上从而使我们算法的复杂度可以降低

普通差分

我们首先介绍最简单的差分情况来描述差分的思想。

假设我们有一个数列，我们希望实现区间加的操作，不过现在情况比较特殊，我们只需要在所有操作完成以后进行一次查询。显然我们可以用线段树□ST表等一系列数据结构进行维护，不过它们的复杂度大部分都是$O(n \log n)$的，当然这也是非常优秀的时间复杂度，但是当查询次数较少时，我们可以有更优的思路。

首先假设我们的数列的第i个数是a_i□我们定义$d_i = a_i - a_{i-1}(i \ge 2)$且$d_1 = a_1$□显然我们发现如果要求$a_i$□我们只需要求出$d_i$的前缀和即可。我们考虑把区间加这个操作在$d_i$上进行，我们发现，这样的本质就是，假设我们要把区间$[l,r]$增加x□那么区间内的d_i不会改变，而$a_l - a_{l-1}$也就是d_l会增加x□而$a_{r+1}-a_r$也就是d_{r+1}会减少x□于是我们发现，在差分意义下，要实现区间加我们只需要修改两个数。当所有的修改完成后，我们对d_i求一次前缀和就可以得到修改后的a_i□时间复杂度为$O(q+n)$□其中q是修改次数。

树上差分

有了普通的差分思想，我们就可以把它应用到树上，假设我们现在有一棵树，我们希望实现把某两个点之间的路径上的所有点点权增加x□仍然只需要进行一次查询，用树链剖分的方法，我们需要用到$O(n \log^2 n)$的复杂度，那么用树上差分的方法呢。

首先，我们可以先进行树链剖分，然后在剖出的链上进行差分，这样我们的复杂度就会少一个log了。不过我们还有一种办法，我们在路径的两个端点把点权加上x□然后在它们的LCA和LCA的父亲处把点权减少x□在所有操作完成后，我们对这棵树进行遍历，一个点的子树里的点权之和就是这个点的权值。这样的复杂度是$O(q \log n + n)$□正确性简单说明：对于一个点，只有这个点在路径上时最终点权才会增加，如果一个点是LCA的父亲或者LCA的其它祖先，那么由于它的子树里进行了$+2x$和$-2x$□所以最终点权没有增加，对于LCA□它的子树里进行了$+2x$和$-x$□最终点权增加x□对于路径上的其他点，它们的子树里一定只有路径两端的其中一个，所以最终点权增加了x□

如果一棵树要进行边权的差分，那么我们把所有的边权下放到深度更大的点，然后每次操作的时候不改LCA的父亲而是把LCA的点权减少$2x$即可。