# 树链剖分

讲解1：

树链剖分通常用于处理树的形态不变 但点权/边权需要修改查询的题目

在选定一个点作为根后 我们来对这棵树进行操作

第一步

从根开始进行一遍遍历(通常用dfs) 这一遍遍历需要统计以下信息

父亲节点fa 当前节点深度d 子树大小sz 这些显然是在学习前置知识时已经用得比较熟练了的

然后 在这一遍遍历中 我们需要再求当前节点的重儿子son

重儿子的定义是 当前节点的子节点中 子树大小最大的那个 (如果有多个任取一个)

其余的就是轻儿子了

另外所有节点与其重/轻儿子的连边称为重/轻边

把连续的重边定义为重链

我们会发现这样一个性质

从任一节点向根节点走 走过的重链和轻边的条数都是log级别以内的

证明如下:

由于任一轻儿子对应的子树大小要小于父节点所对应子树大小的一半

因此从一个轻儿子沿轻边向上走到父节点后 所对应的子树大小至少变为两倍以上

经过的轻边条数自然是不超过log2N的

然后由于重链都是间断的 (连续的可以合成一条)

所以经过的重链的条数是不超过轻边条数+1的

因此经过重链的条数也是log级别的

综合可知原命题得

从轻边向上走显然每条轻边都可以做到O(1)向上走

而从重链向上走要做到每条重链只用O(1)就必须额外做一些处理

第二步

利用第一遍遍历得到的信息 我们再进行一遍遍历(需用dfs)

对于每个重儿子 要求出沿重链向上走走到顶端的点的位置top

这个top显然是和父节点的一样的

对于每个轻儿子 由于向上走的重链不存在 我们可以令top为自身

现在从重链向上走都只需要O(1)了

不过修改的复杂度仍然不靠谱

对于两点之间的修改和询问操作 轻边我们可以暴力直接修改询问 重链则必须结合一些数据结构进行以保证效率

这个时候 我们或许会回想起学dfs序的时候 我们将树上的点根据dfs序映射到了一维数组上

从而可以利用线段树等数据结构对在dfs序上连续的一段区间进行修改和询问

因此 为了能够用线段树等数据结构进行维护 我们必须将同一条重链上的点映射到一个连续的区间

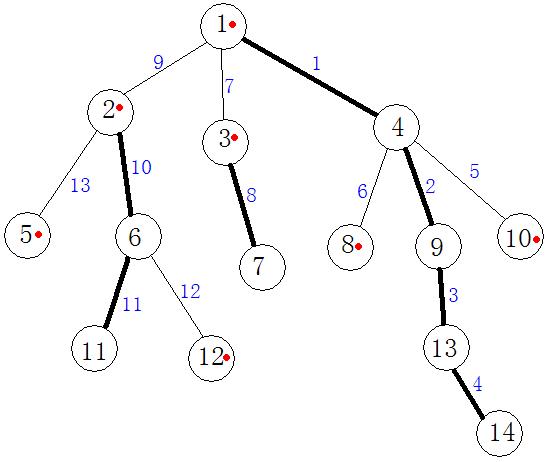
这个操作并不复杂 我们只需在对每个点dfs时先遍历到它的重儿子 最后重链上的点映射到一维数组里便是连续的

做完这两个步骤后 树链剖分的核心部分就结束了

不过注意两个点向上走的时候 不能走得超过这两个点的LCA

这里的具体操作最好独立思考一下 对比倍增LCA的写法会更好理解

讲解2：

树链，就是树上的路径。剖分，就是把路径分类为重链和轻链。  
    记siz[v]表示以v为根的子树的节点数，dep[v]表示v的深度(根深度为1)，top[v]表示v所在的链的顶端节点，fa[v]表示v的父亲，son[v]表示与v在同一重链上的v的儿子节点（姑且称为重儿子），w[v]表示v与其父亲节点的连边（姑且称为v的父边）在线段树中的位置。只要把这些东西求出来，就能用logn的时间完成原问题中的操作。  
  
    重儿子：siz[u]为v的子节点中siz值最大的，那么u就是v的重儿子。  
    轻儿子：v的其它子节点。  
    重边：点v与其重儿子的连边。  
    轻边：点v与其轻儿子的连边。  
    重链：由重边连成的路径。  
    轻链：轻边。  
  
    剖分后的树有如下性质：  
    性质1：如果(v,u)为轻边，则siz[u] \* 2 < siz[v]；  
    性质2：从根到某一点的路径上轻链、重链的个数都不大于logn。  
     
  
    算法实现：  
    我们可以用两个dfs来求出fa、dep、siz、son、top、w。  
    dfs\_1：把fa、dep、siz、son求出来，比较简单，略过。  
    dfs\_2：⒈对于v，当son[v]存在（即v不是叶子节点）时，显然有top[son[v]] = top[v]。线段树中，v的重边应当在v的父边的后面，记w[son[v]] = totw+1，totw表示最后加入的一条边在线段树中的位置。此时，为了使一条重链各边在线段树中连续分布，应当进行dfs\_2(son[v])；  
           ⒉对于v的各个轻儿子u，显然有top[u] = u，并且w[u] = totw+1，进行dfs\_2过程。  
           这就求出了top和w。将树中各边的权值在线段树中更新，建链和建线段树的过程就完成了。  
    修改操作：例如将u到v的路径上每条边的权值都加上某值x。  
    一般人需要先求LCA，然后慢慢修改u、v到公共祖先的边。而高手就不需要了。  
    记f1 = top[u]，f2 = top[v]。  
    当f1 <> f2时：不妨设dep[f1] >= dep[f2]，那么就更新u到f1的父边的权值(logn)，并使u = fa[f1]。  
    当f1 = f2时：u与v在同一条重链上，若u与v不是同一点，就更新u到v路径上的边的权值(logn)，否则修改完成；  
    重复上述过程，直到修改完成。  
   求和、求极值操作：类似修改操作，但是不更新边权，而是对其求和、求极值。  
    就这样，原问题就解决了。鉴于鄙人语言表达能力有限，咱画图来看看：[[](http://photo.blog.sina.com.cn/showpic.html#blogid=7a1746820100wp67&url=http://s16.sinaimg.cn/orignal/6974c8b2gb4c1e1110f6f)](http://photo.blog.sina.com.cn/showpic.html#blogid=7a1746820100wp67&url=http://s16.sinaimg.cn/orignal/6974c8b2gb4c1e1110f6f)  
  
    如右图所示，较粗的为重边，较细的为轻边。节点编号旁边有个红色点的表明该节点是其所在链的顶端节点。边旁的蓝色数字表示该边在线段树中的位置。图中1-4-9-13-14为一条重链。  
  
    当要修改11到10的路径时。  
    第一次迭代：u = 11，v = 10，f1 = 2，f2 = 10。此时dep[f1] < dep[f2]，因此修改线段树中的5号点，v = 4, f2 = 1；  
    第二次迭代：dep[f1] > dep[f2]，修改线段树中10--11号点。u = 2，f1 = 2；  
    第三次迭代：dep[f1] > dep[f2]，修改线段树中9号点。u = 1，f1 = 1；  
    第四次迭代：f1 = f2且u = v，修改结束。

## Poj 3237

题意很简单，给出n个节点的一棵树，有三种操作：

1、C修改第i条边的值为v

2、N改变节点a到b内边的权值的符号（取反）

3、Q询问节点a到b内权值的最大值

首先树链剖分，注意一般的树链剖分是点权的，所以要将边整合到线段树上，应该把边权赋给深度较大的结点，所以没个结点代表的连向父亲的那一条边，根节点无意义，同样我们也求出dfs序列，然后建立一颗线段树，因为存在取反操作，所以最大值可能是由最小值取反得到，所以记录最大和最小值，cl[i][0]记录第i段的最大值，cl[i][1]记录最小值，tag做标记，标记该段是否取反。对于两点之间的修改和询问操作 轻边我们可以暴力直接修改询问 重链则必然在线段树中是连续一段。所以保证时间复杂度为nlogn^2。

## Hdu6031

女生赛的抢杯题

题目大意是给一颗树，n个点，1为根，m次询问，每次询问给出两个集合A，B。求集合A和集合B中的点的深度最大的LCA,保证所有集合A,B加起来不超过10w。 n,m=10w.

一看这数据范围就像是nlog^2的树链剖分了。

我们将数剖分后，以每个点的深度作为每个点的权值，然后求出每个点的最大值val，然后每次进来一个集合A，对线段树进行区间修改，修改线段树1到pos[num]的所有结点tag，表示这个结点的是否被A访问过，如果是那么就更新每个点的真实最大值ans(因为有的点可能不会被A访问)然后对每一个B,我们只要在线段树中查询1到pos[num]的最大值即可。

PS：某位有女生想出的可怕算法，处理出每个点的父节点，倍增法处理出fa[i][j]数组。   
每次询问时，二分目标深度，找出集合A中每个点在目标深度的祖先，并插入集合S，再找出集合B中每个点在目标深度的祖先，如果祖先在集合S中，说明这个深度A，B存在公共祖先。时间复杂度为nlogn^2,而且常数小好写，不知道高明到哪里去了