# 史上最详尽的LCT讲解

#### 

#### **『LCT能干嘛』**

（1）维护一个序列，支持下列操作：   
区间求和   
区间求最值   
区间修改   
求连续子段和   
这个****线段树****就可以解决 具体做法不加累述了   
（2）维护一个序列，支持下列操作：   
区间求和   
区间求最值   
区间修改   
求连续子段和   
****添加一段区间****   
****删除一段区间****   
****翻转一段区间****   
****Splay****的基本操作   
（3）维护一棵树，支持下列操作：   
链上求和   
链上求最值   
链上修改   
子树修改   
子树求和   
****树链剖分！！！****   
（4）维护一棵树，支持下列操作：   
链上求和   
链上求最值   
链上修改   
****断开树上的一条边****   
****连接两个点，保证连接后仍然是一棵树****   
由于树是动态的，我们不能每次操作都重标号一遍 树链剖分搞不了了   
然而我们可以沿用树链剖分的轻重链剖分的概念   
既然是动态那么我们肯定要把静态的线段树换成动态的Splay   
于是就有LCT≈树链剖分+Splay

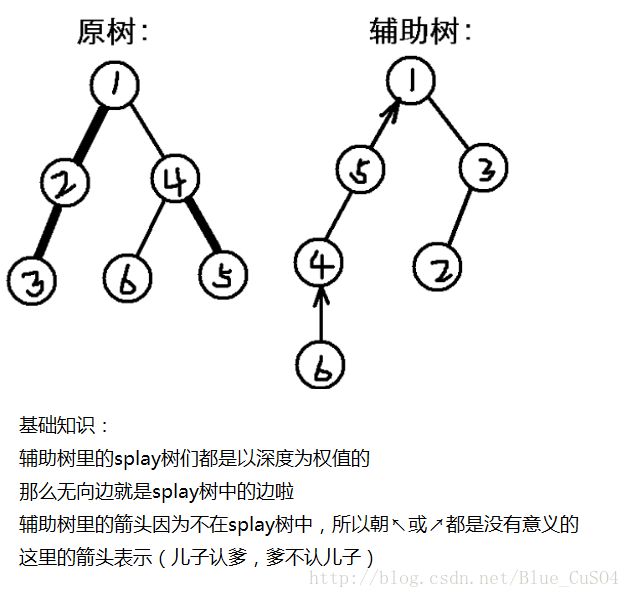
#### **『引入一些概念』**

Preferred Child：重儿子，重儿子与父亲节点同在一棵Splay中，一个节点最多只能有一个重儿子   
Preferred Edge：重边，连接父亲节点和重儿子的边   
Preferred Path：重链，由重边及重边连接的节点构成的链

#### **『Auxiliary Tree(辅助树)』**

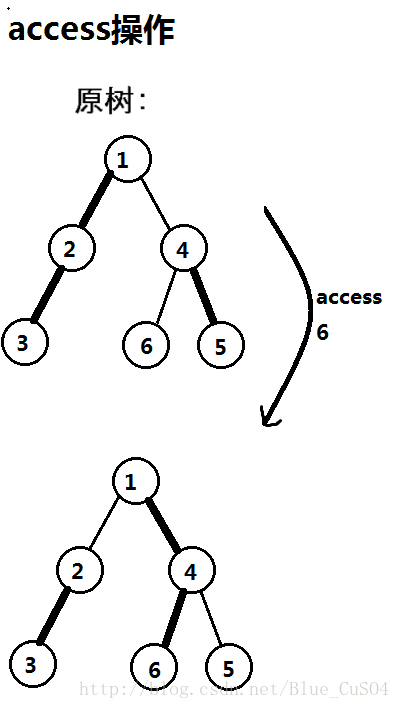
****由一条重链上的所有节点所构成的Splay称作这条链的辅助树****   
****每个点的键值为这个点的深度****   
辅助树的根节点的父亲指针指向链顶的父亲节点，然而链顶的父亲节点的儿子并不指向辅助树的根节点（儿子认爹&&爹不认儿子）   
原树中的****重链**** -> 辅助树中两个节点位于****同一棵Splay****中   
原树中的****轻链**** -> 辅助树中子节点所在Splay的根节点的father指向其父节点   
注意原树与辅助树的结构并不相同

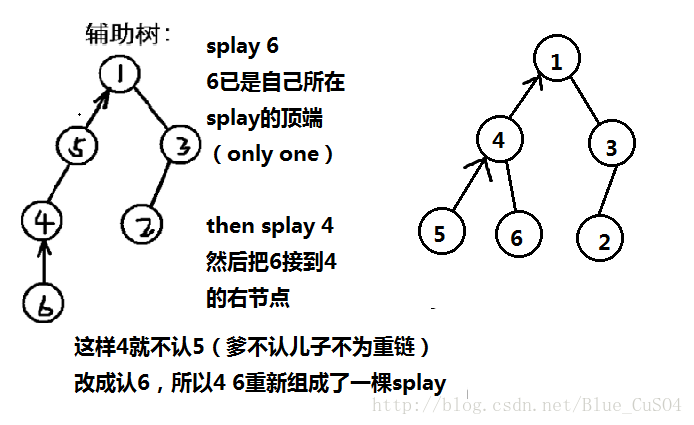
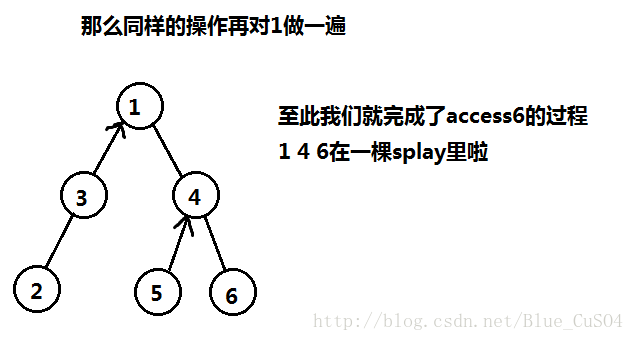
### **那么切入正题咯**



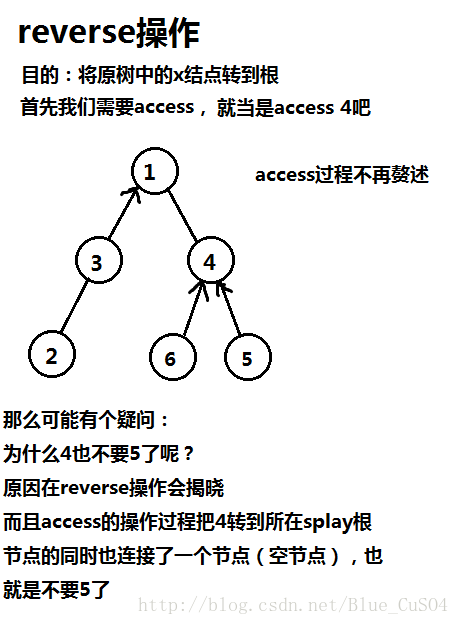
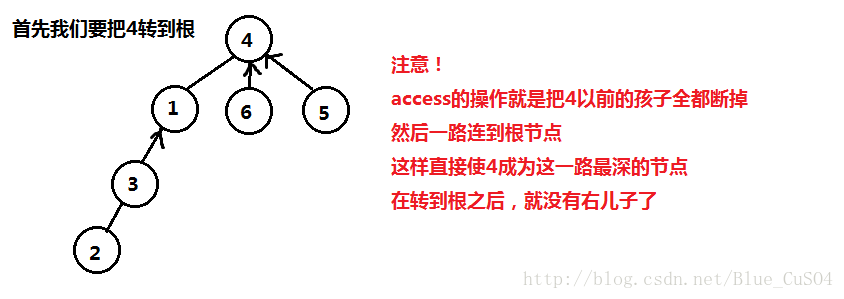
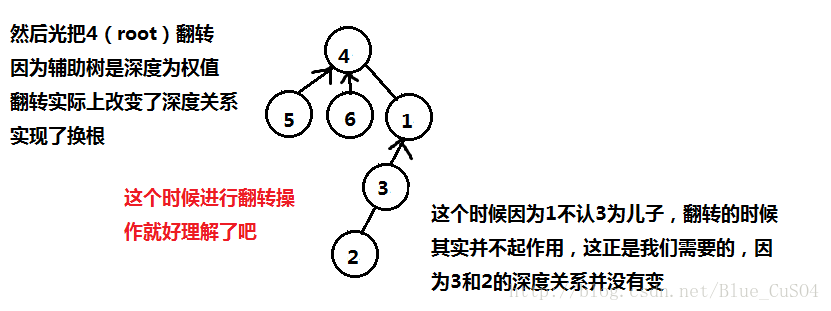
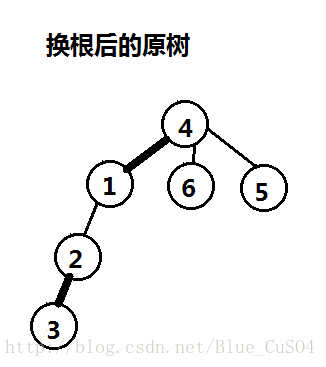
那么LCT中最重要的操作可以说是….

#### **『Access操作』**

目的：将x的重边切断，并将x到根的路径上所有的边都搞成重边。   
具体实现：根据辅助树按照深度为关键字的性质。不断地将一个结点的父亲转到根，然后把这个结点接到它父亲的右儿子，此时要切断x下面的所有重边   


#### **『Reverse操作』**

目的：将原树中的x结点转到根。   
具体实现：因为原树是虚树，所以在原树中进行变换实际上是在辅助树中进行变换。首先Access一个点，再将这个点在辅助树中转到根。又是根据辅助树按照深度为关键字的性质，将这个点所在的splay树反转，实际上改变了深度的关系，也就是实现的原树的换根。   
   
  
  


**LCT中有关splay的操作：**

void up(int k){ //向上更新数据(本题是求xor值)

xyz[k]=xyz[son[k][0]]^xyz[son[k][1]]^val[k];

}

void revers(int k){ //区间翻转

swap(son[k][0],son[k][1]);

rev[k]^=1;`

}

void down(int k){ //下推标记

if(rev[k]){

revers(son[k][0]);revers(son[k][1]);

rev[k]=0;`

}

}

void rotate(int k){

if(isroot(k)) return;

int f=fa[k],gran=fa[f],d=(son[f][1]==k);

if(!isroot(f)) son[gran][son[gran][1]==f]=k; //额外注意if(!isroot(f))语句，此处不判断会引起致命错误

son[f][d]=son[k][d^1];

if(son[k][d^1]) fa[son[k][d^1]]=f;

son[k][d^1]=f;fa[f]=k;fa[k]=gran;

up(f);up(k);

}

void splay(int k){

int z=0,y=k,gran;

sta[++z]=y;

while(!isroot(y)) sta[++z]=y=fa[y];

while(z) down(sta[z--]);

while(!isroot(k)){

y=fa[k];gran=fa[y];

if(!isroot(y)) rotate((son[y][1]==k)^(son[gran][1]==y)?k:y);

rotate(k);

}

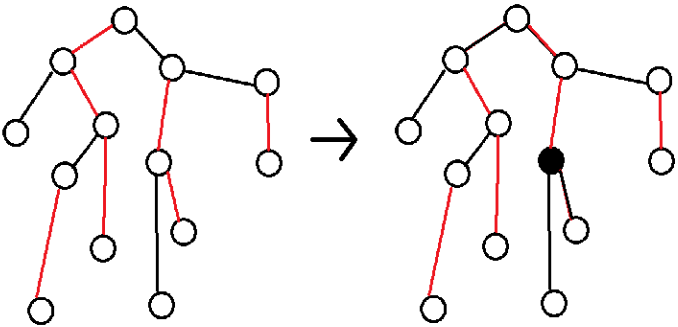
up(k);

}

splay的操作类似于普通的splay，但是又有所不同，比如在进行rotate时我们需要判断点是否为当前splay的根。而在进行splay操作时，必须时一定要从上往下放标记。

Access操作：

access操作是LCT的精髓。access(x)的操作是将当前LCT的根到x的路径上的所有边变为实边，比如下图。



我们将从根到黑点的所有边变为实边，那么就必须将一些本来的实边变为虚边。

那我们对于每个splay都将x旋转到根，则x的连向右儿子的边变为虚边。并将x的右儿子变成上次access的点，然后上提x（将x变为其父亲），继续进行access操作即可。由于splay的性质，右儿子的深度大于点x，而由于我们已经确定了实边，剩下的边都是虚边，所以将右儿子变为虚边即可。（删去x的右儿子，但是fa[右儿子]=x）,我们这样一直往上提，就可以完成access操作。

void access(int k){ //打通LCT根到k的路径

int y=0;

do{

splay(k);son[k][1]=y;up(k);y=k;k=fa[k];

}while(k);

}

**Makeroot操作：**

makeroot(x)操作指的是将点x变为当前LCT的根，那我们先access(x)，打通x到根的路径，然后再splay一下，则x就变成了当前splay的根，但是目前的splay是不满足性质的，因为（当前）深度最小的x在树根处，而所有的点都是x的左子树，而实际上所有的点都应该在x的右子树里（因为深度大于x），所以我们直接将整个区间reverse一下，翻转一遍就可以满足splay的性质了。

void makeroot(int k){ //将k变为LCT根

access(k);splay(k);

revers(k);

}

**Findroot操作：**

findroot(x)表示x在的splay的树根。我们先打通x到根的链，然后把x旋转到根，由于我们要找的根深度最小（之前），所以直接一直寻找当前点的左儿子即可。注意在进行findroot时一定要下推标记，不然可能会错。

int findroot(int k){ //找k所在LCT的根

access(k);splay(k);

while(son[k][0]) down(k),k=son[k][0];

return k;

}

**Split操作：**

split(x,y)可以处理出x到y的路径。我们makeroot(x)，然后再access(y)，这样就形成了一个从x到y路径的splay，然后我们将y旋转到根，则y点上的值就是我们要求的路径的值。

void split(int x,int y){ //查询x到y的路径

makeroot(x);access(y);

splay(y);

}

**Link操作：**

link(x,y)表示连边(x,y)，我们连x和y之前要先判断一下，如果find(x)==find(y)，即x和y在同一个splay里，在同一个联通块里，那么就直接return，不需要连边。否则我们先makeroot(x),然后将x的父亲变成y即可。

void link(int x,int y){ //连边x到y

makeroot(x);

if(findroot(y)==x) return;

fa[x]=y;

}

**Cut操作：**

cut(x,y)表示删边(x,y)。我们考虑哪些情况不需要cut，首先如果find(x)!=find(y)，肯定不需要cut，因为它们根本不在同一个联通块里。我们makeroot(x)，那么如果fa[x]!=y那也可以不删，因为不存在x到y的边。那如果x与y有边就一定可以cut吗？不，如果x有右儿子就不能删，因为x有右儿子就会意味着x和y之间有别的边。

void cut(int x,int y){ //删边x到y

makeroot(x);

if(findroot(y)!=x||fa[x]!=y||son[x][1]) return; //顺序不能打反

fa[x]=son[y][0]=0;

up(y);

}

这些操作做完之后就完成了最基础的LCT操作，下面是LUOGU上LCT模板题的CODE

#include<bits/stdc++.h>

#define MAXN 300005

using namespace std;

int read(){

char c;int x=0,y=1;while(c=getchar(),(c<'0'||c>'9')&&c!='-');

if(c=='-') y=-1;else x=c-'0';while(c=getchar(),c>='0'&&c<='9')

x=x\*10+c-'0';return x\*y;

}

int son[MAXN][2],fa[MAXN],val[MAXN],xyz[MAXN],rev[MAXN],sta[MAXN];

void up(int k){

xyz[k]=xyz[son[k][0]]^xyz[son[k][1]]^val[k];

}

void revers(int k){ //区间翻转

swap(son[k][0],son[k][1]);

rev[k]^=1;

}

void down(int k){

if(rev[k]){

revers(son[k][0]);revers(son[k][1]);

rev[k]=0;

}

}

int isroot(int k){ //判断k是否为其splay的根

return (son[fa[k]][0]!=k)&&(son[fa[k]][1]!=k);

}

void rotate(int k){

if(isroot(k)) return;

int f=fa[k],gran=fa[f],d=(son[f][1]==k);

if(!isroot(f)) son[gran][son[gran][1]==f]=k;

son[f][d]=son[k][d^1];

if(son[k][d^1]) fa[son[k][d^1]]=f;

son[k][d^1]=f;fa[f]=k;fa[k]=gran;

up(f);up(k);

}

void splay(int k){

int z=0,y=k,gran;

sta[++z]=y;

while(!isroot(y)) sta[++z]=y=fa[y];

while(z) down(sta[z--]);

while(!isroot(k)){

y=fa[k];gran=fa[y];

if(!isroot(y)) rotate((son[y][1]==k)^(son[gran][1]==y)?k:y);

rotate(k);

}

up(k);

}

void access(int k){ //打通LCT根到k的路径

int y=0;

do{

splay(k);son[k][1]=y;up(k);y=k;k=fa[k];

}while(k);

}

void makeroot(int k){ //将k变为LCT根

access(k);splay(k);

revers(k);

}

int findroot(int k){ //找k所在LCT的根

access(k);splay(k);

while(son[k][0]) down(k),k=son[k][0];

return k;

}

void split(int x,int y){ //查询x到y的路径

makeroot(x);access(y);

splay(y);

}

void link(int x,int y){ //连边x到y

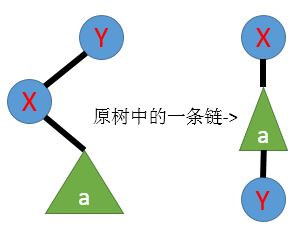
makeroot(x);

if(findroot(y)==x) return;

fa[x]=y;

}

void cut(int x,int y){//删边x到y。判断：连通性，x,y是否有父子关系，x是否有右儿子

 makeroot(x);

if(findroot(y)!=x||fa[x]!=y||son[x][1]) return;

fa[x]=son[y][0]=0;

up(y);

}

int n,m;

int main()

{

n=read();m=read();

for(int i=1;i<=n;i++) val[i]=read();

for(int i=1;i<=m;i++){

int type=read(),x=read(),y=read();

if(type==0){

split(x,y);printf("%d\n",xyz[y]);

}

if(type==1) link(x,y);

if(type==2) cut(x,y);

if(type==3){

splay(x);val[x]=y;

}

}

return 0;

}

# **推荐的题目**

[BZOJ 2049 [Sdoi2008]Cave 洞穴勘测](https://www.lydsy.com/JudgeOnline/problem.php?id=2049" \t "https://blog.csdn.net/wang3312362136/article/details/_blank)

这题只有link和cut两个操作，比较水

[BZOJ 3282 Tree](https://www.lydsy.com/JudgeOnline/problem.php?id=3282" \t "https://blog.csdn.net/wang3312362136/article/details/_blank)

LCT真板子题

Qtree系列

据说都可以用LCT写？