

FlashHu

——大步小步走出过往

LCT总结——概念篇+洛谷P3690[模板]Link Cut Tree(动态树) (LCT, Splay)

为了优化体验（其实是强迫症），蒟蒻把总结拆成了两篇，方便不同学习阶段的Dalao们切换。

LCT总结——应用篇戳[这里](#)

概念、性质简述

首先介绍一下**链剖分**的概念（感谢laofu的讲课）

链剖分，是指一类对树的边进行轻重划分的操作，这样做的目的是为了减少某些链上的修改、查询等操作的复杂度。

目前总共有三类：重链剖分，实链剖分和并不常见的长链剖分

重链剖分

实际上我们经常讲的树剖，就是重链剖分的常用称呼。

对于每个点，选择最大的子树，将这条连边划分为重边，而连向其他子树的边划分为轻边。

若干重边连接在一起构成重链，用树状数组或线段树等静态数据结构维护。

至于有怎样优秀的性质等等，不在本总结的讨论范畴了（其实是因为本蒟蒻连树剖都不会）

实链剖分

同样将某一个儿子的连边划分为实边，而连向其他子树的边划分为虚边。

区别在于虚实是可以动态变化的，因此要使用更高级、更灵活的Splay来维护每一条由若干实边连接而成的实链。

基于性质更加优秀的实链剖分，LCT(Link-Cut Tree)应运而生。

LCT维护的对象其实是一个森林。

在实链剖分的基础下，LCT资磁更多的操作

- 查询、修改链上的信息（最值，总和等）
- 随意指定原树的根（即换根）
- 动态连边、删边
- 合并两棵树、分离一棵树（跟上面不是一毛一样吗）
- 动态维护连通性
- 更多意想不到的操作（可以往下滑一滑）

想学Splay的话，推荐[巨佬yyb的博客](#)

LCT的主要性质如下：

1. 每一个Splay维护的是一条从上到下按在原树中深度严格递增的路径，且中序遍历Splay得到的每个点的深度序列严格递增。
是不是有点抽象哈
比如有一棵树，根节点为1（深度1），有两个儿子2,3（深度2），那么Splay有3种构成方式：
 $\{1-2\}, \{3\}$
 $\{1-3\}, \{2\}$
 $\{1\}, \{2\}, \{3\}$ （每个集合表示一个Splay）
而不能把1,2,3同放在一个Splay中（存在深度相同的点）
2. 每个节点包含且仅包含于一个Splay中

公告



欢迎各路大佬和我QQ或邮件交流，也欢迎查错，我会尽量及时回复，感谢！



给我留言

本作品由[胡锦涛](#)采用[知识共享署名-非商业性使用-禁止演绎 4.0 国际许可协议](#)进行许可。

昵称: Flash_Hu

园龄: 1年8个月

粉丝: 138

关注: 78

+加关注

2019年9月						
日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

搜索

积分与排名

积分 - 68037

排名 - 8835

随笔分类 (425)

OI——记事(6)
OI——算法模板(22)
OI——算法总结(25)
OI——题解(86)
OI——小技巧(8)
基础——C++STL——map,multimap(2)
基础——C++STL——priority_queue(1)
基础——C++STL——set,multiset(4)
基础——C++模板(4)
基础——排序——归并排序(2)
基础——排序——基数排序(2)
基础——输入输出(1)
软件——Emacs(1)
软件——gnome-terminal(1)
数据结构——并查集(6)
数据结构——堆——左偏树(1)
数据结构——分块(1)
数据结构——链剖——LCT(20)
数据结构——链剖——树链剖分(5)
数据结构——平衡树——Splay(5)
数据结构——平衡树——珂朵莉树(2)

3. 边分为实边和虚边，实边包含在Splay中，而虚边总是由一棵Splay指向另一个节点（指向该Splay中中序遍历最靠前的点在原树中的父亲）。
- 因为性质2，当某点在原树中有多个儿子时，只能向其中一个儿子拉一条实链（只认一个儿子），而其它儿子是不能在这个Splay中的。
- 那么为了保持树的形状，我们要让到其它儿子的边变为虚边，由对应儿子所属的Splay的根节点的父亲指向该点，而从该点并不能直接访问该儿子（认父不认子）。

各种操作

`access(x)`

LCT核心操作，也是最难理解的操作。其它所有的操作都是在此基础上完成的。

因为性质3，我们不能总是保证两个点之间的路径是直接连通的（在一个Splay上）。

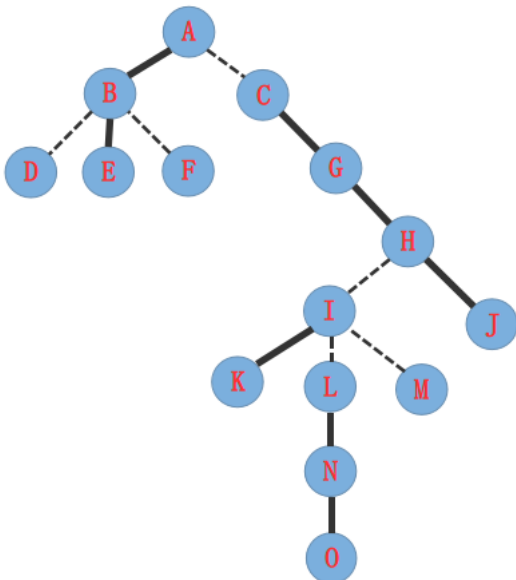
`access`即定义为打通根节点到指定节点的实链，使得一条中序遍历以根开始、以指定点结束的Splay出现。

蒟蒻深知没图的痛苦qwq

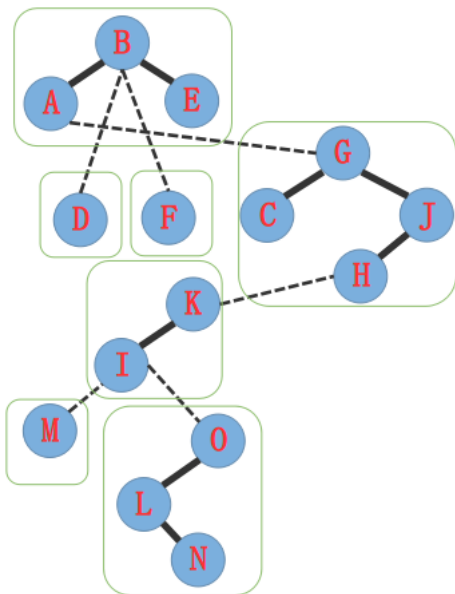
所以还是来几张图吧。

下面的图片参考YangZhe的论文

有一棵树，假设一开始实边和虚边是这样划分的（虚线为虚边）



那么所构成的LCT可能会长这样（绿框中为一个Splay，可能不会长这样，但只要满足中序遍历按深度递增（性质1）就对结果无影响）



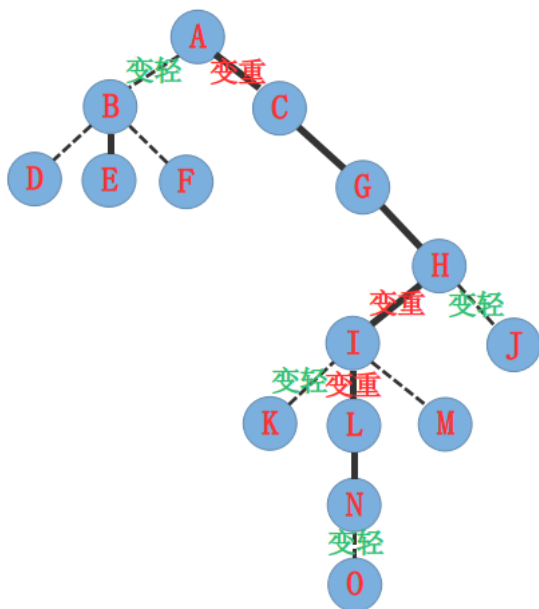
现在我们要`access(N)`，把 $A - N$ 的路径拉起来变成一条Splay。

因为性质2，该路径上其它链都要给这条链让路，也就是把每个点到该路径以外

所以我们希望虚实边重新划分成这样。

148

- 数据结构——树套树(2)
- 数据结构——树状数组(5)
- 数据结构——线段树(9)
- 数据结构——线段树——可持久化线段树(3)
- 数据结构——线段树——李超线段树(1)
- 数据结构——线段树——势能线段树(4)
- 数据结构——线段树——线段树分裂(1)
- 数据结构——线段树——线段树合并(1)
- 数据结构——线段树——主席树(4)
- 数据结构——线性结构——队列——单调队列(6)
- 数据结构——线性结构——队列——双端队列(1)
- 数据结构——线性结构——栈(2)
- 数据结构——线性结构——栈——单调栈(2)
- 数学——博弈论(1)
- 数学——博弈论——SG函数(1)
- 数学——计算几何(4)
- 数学——计算几何——半平面交/线性规划(3)
- 数学——计算几何——凸包(2)
- 数学——计算几何——凸包——闵可夫斯基和(2)
- 数学——计算几何——凸包——旋转卡壳(1)
- 数学——计算几何——自适应辛普森积分(1)
- 数学——计算几何——最小圆覆盖(1)
- 数学——群论——Pólya定理(1)
- 数学——数论——积性函数——杜教筛/Min25筛(1)
- 数学——数论——积性函数——莫比乌斯反演(3)
- 数学——数论——积性函数——欧拉函数(2)
- 数学——数论——积性函数——线性筛(4)
- 数学——数论——同余——BSGS(1)
- 数学——数论——同余——费马小定理/欧拉定理(2)
- 数学——数论——同余——扩展欧几里德(5)
- 数学——数论——同余——卢卡斯定理(2)
- 数学——数论——同余——中国剩余定理(3)
- 数学——线性代数——FWT(1)
- 数学——线性代数——多项式运算(3)
- 数学——线性代数——高斯消元(1)
- 数学——线性代数——生成函数(1)
- 数学——线性代数——线性基(2)
- 数学——组合计数——Catalan数(1)
- 数学——组合计数——Stirling数(2)
- 数学——组合计数——隔板法(2)
- 数学——组合计数——矩阵树定理(1)
- 数学——组合计数——容斥原理(4)
- 算法——倍增——倍增LCA(3)
- 算法——动态规划(1)
- 算法——动态规划——1D1D——决策单调性(6)
- 算法——动态规划——1D1D——斜率优化(4)
- 算法——动态规划——背包(2)
- 算法——动态规划——背包——多重背包(1)
- 算法——动态规划——背包——完全背包(1)
- 算法——动态规划——树形DP(3)
- 算法——动态规划——数位DP(1)
- 算法——动态规划——序列DP——LCS(1)
- 算法——动态规划——序列DP——LIS
- 算法——动态规划——状压DP——插头DP
- 算法——动态规划——状压DP——子集DP(1)
- 算法——二分——WQS二分/带权二分(3)
- 算法——二分——二分查找(3)
- 算法——二分——二分答案(5)
- 算法——二分——分数规划(1)
- 算法——分治(3)
- 算法——分治——CDQ(2)
- 算法——分治——树分治(2)
- 算法——分治——树分治——动态点分治(1)
- 算法——分治——线段树分治(1)
- 算法——莫队(1)
- 算法——搜索——DFS(1)
- 算法——搜索——剪枝(1)
- 算法——搜索——模拟退火(2)
- 算法——贪心(4)
- 算法——贪心——模拟费用流(1)
- 图论——DAG——拓扑排序(4)
- 图论——DFS求环(2)
- 图论——点双/边双——缩点(1)
- 图论——点双/边双——仙人掌(1)



然后怎么实现呢？

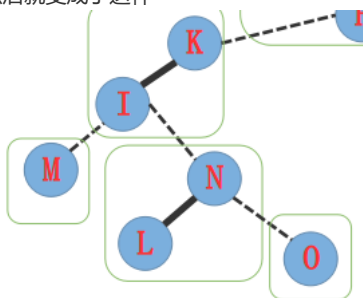
我们要一步步往上拉。

首先把 $splay(N)$ ，使之成为当前Splay中的根。

为了满足性质2，原来 $N - O$ 的重边要变轻。

因为按深度 O 在 N 的下面，在Splay中 O 在 N 的右子树中，所以直接单方面将 N 的右儿子置为 0 （认父不认子）

然后就变成了这样——

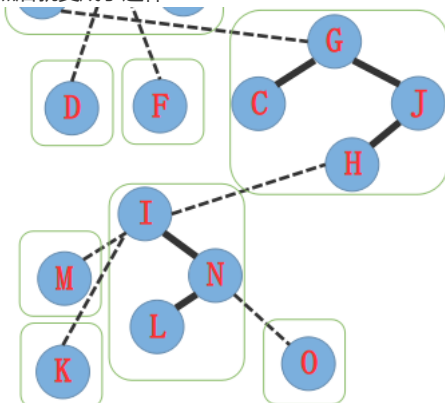


我们接着把 N 所属Splay的虚边指向的 I （在原树上是 L 的父亲）也转到它所属Splay的根， $splay(I)$ 。

原来在 I 下方的重边 $I - K$ 要变轻（同样是将右儿子去掉）。

这时候 $I - L$ 就可以变重了。因为 L 肯定是在 I 下方的（刚才 L 所属Splay指向了 I ），所以 I 的右儿子置为 N ，满足性质1。

然后就变成了这样——



图论——点双/边双——圆方树(1)
图论——二分图——二分图染色(1)
图论——欧拉路径/欧拉回路(1)
图论——平面图/对偶图(1)
图论——树——LCA(2)
图论——树——基环树(1)
图论——树——生成树(3)
图论——树——生成树——Kruskal(4)
图论——树——树的dfn序(5)
图论——树——树的重心(4)
图论——树——树上差分(6)
图论——树——树形图(1)
图论——树——虚树(1)
图论——网络流(3)
图论——网络流——费用流(2)
图论——最短/长路(6)
图论——最短/长路——Floyd(2)
图论——最短/长路——K短路(1)
图论——最短/长路——差分约束(1)
字符串——后缀——后缀平衡树(1)
字符串——后缀——后缀数组(2)
字符串——后缀——后缀自动机(1)
字符串——后缀——后缀自动机——广义SAM(1)
字符串——回文——Manacher(1)
字符串——回文——回文自动机(1)
字符串——匹配——AC自动机(2)
字符串——匹配——KMP(1)
字符串——匹配——子序列匹配(1)
字符串——最小表示法(1)

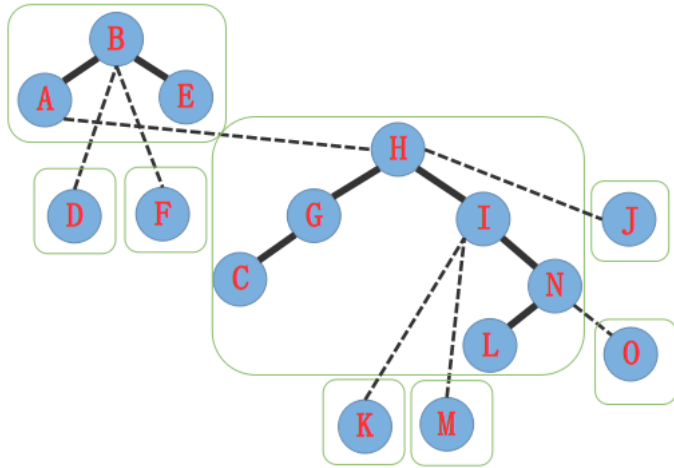
随笔档案 (125)

2019年4月(2)
2019年3月(7)
2019年2月(2)
2019年1月(6)
2018年12月(3)
2018年11月(5)
2018年10月(8)
2018年9月(14)
2018年8月(16)
2018年7月(8)
2018年6月(7)
2018年4月(14)
2018年3月(11)
2018年2月(11)
2018年1月(11)

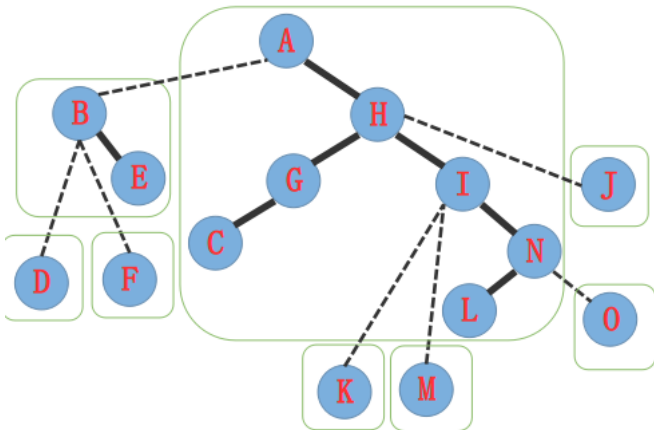
被同级巨佬全方位吊打

%eternal风度 (LJL) %
%mona(hnzzq)%
%LJQ神犇%
%#(天平)#%
%SYCstudio%
%蓝月亮ZSY%orz
GuessWhols%YCBJulao%?
%YL,ppl,帅亮%
%红太阳YYB (装蒟蒻的巨佬系列) %
%ZKJ小胖%
%XZYXZY巨佬%
%FDFDF巨佬%
%DWQ巨佬(中国飞鱼)%
%SYCstudio.com%(SYC的新域名)
%LWH巨佬%
%Brioche(LKJ)%
%stocxorz%
%Cwen!%
%ysner (又一个装蒟蒻的巨佬) %
%Tyher天天吊打我%
%巨强的学霸%

I 指向 H ，接着 $splay(H)$ ， H 的右儿子置为 I 。



H 指向 A ，接着 $splay(A)$ ， A 的右儿子置为 H 。



$A - N$ 的路径已经在—个Splay中了，大功告成！

代码其实很简单。。。。。。循环处理，只有四步——

1. 转到根；
2. 换儿子；
3. 更新信息；
4. 当前操作点切换为轻边所指的父亲，转1

```
inline void access(int x){
    for(int y=0;x;y=x,x=f[x])
        splay(x),c[x][1]=y,pushup(x); // 儿子变了，需要及时上传信息
}
```

makeroot(x)

只是把根到某个节点的路径拉起来并不能满足我们的需要。更多时候，我们要获取指定两个节点之间的路径信息。

然而一定会出现路径不能满足按深度严格递增的要求的情况。根据性质1，这样的路径不能在一个Splay中。

Then what can we do?

makeroot定义为换根，让指定点成为原树的根。

这时候就利用到 $access(x)$ 和Splay的翻转操作。

$access(x)$ 后 x 在Splay中一定是深度最大的点对吧。

$splay(x)$ 后， x 在Splay中将没有右子树（性质1）。于是翻转整个Splay，使得所有点的深度都倒过来了， x 没了左子树，反倒成了深度最小的点（根节点），达到了我们的目的。

代码

```
inline void pushr(int x){ // Splay区间翻转操作
    swap(c[x][0],c[x][1]);
    r[x]^=1; // r为区间翻转懒标记数组
}
```

148

%xzz球爷（又一个装蒟蒻的巨佬）%
 %高神（鸡贼歪）%
 %TJJ(iot)%
 %DSL(杜杜熊)%
 %SMYJL%
 %装newbie的igm神猫贼%
 %CHD(怎么全是装蒟蒻的巨佬啊)%
 %HL(Winlere)%
 %一年AC1700的神仙Itst%
 %神仙般的M_sea%

郑州
32°

一走出校门又被外校巨佬吊打

%zjp_shadow%
 %巨佬Epiphyllum%
 %Judge%
 %AK_Gang%
 %GKxx%
 %Ning_Mew%
 %redbag%
 %奇虎首席工程师Qihoo360%

最新评论

1. Re:洛谷P2900 [USACO08MAR]土地征用Land Acquisition（动态规划，斜率优化，决策单调性，线性规划，单调队列）
那小学生怎么办（滑稽）
--longlongzhu123
2. Re:LCT总结——概念篇+洛谷P3690[模板]Link Cut Tree(动态树) (LCT, Splay)
我会说cdqz的课件里面access有20+行吗：（顺便求大佬友链啊QwQlcyfrog :...
--lcyfrog
3. Re:LCT总结——概念篇+洛谷P3690[模板]Link Cut Tree(动态树) (LCT, Splay)
大佬太巨了，学会了。
--Sword_Art_Online
4. Re:二进制高精度模板（高精度）
可以用operator bool
--autoint
5. Re:DP的各种优化（动态规划，决策单调性，斜率优化，带权二分，单调栈，单调队列）
好！对我很有帮助！
--jklover
6. Re:洛谷P2542 [AHOI2005]航线规划（LCT，双连通分量，并查集）
太巨了！！！！排序二分找边太巨了！
--红色OI再临
7. Re:DP的各种优化（动态规划，决策单调性，斜率优化，带权二分，单调栈，单调队列）
@百叶_LLouver博主退役了qaq，不太会改了只能麻烦大佬看看其它大佬的吧...
--Flash_Hu
8. Re:DP的各种优化（动态规划，决策单调性，斜率优化，带权二分，单调栈，单调队列）
所以能否请您看一眼您的洛谷P3515的题解QAQ
代码是不是锅了
--百叶_LLouver
9. Re:DP的各种优化（动态规划，决策单调性，斜率优化，带权二分，单调栈，单调队列）
@百叶_LLouver不会吧，这样所有的pi不是等于1了吗...
--Flash_Hu
10. Re:模拟退火总结（模拟退火）
博主威武
--任新宇
11. Re:DP的各种优化（动态规划，决策单调性，斜率优化，带权二分，单调栈，单调队列）
@Flash_Hu 决策单调性成立的条件应该是 $\pi \leq \pi+1$ 或 $\pi \geq \pi+1$ 吧
--百叶_LLouver
12. Re:Outsider (HNOI2019)
@小蒟蒻yyb您是全机房的红太阳！...
--Flash_Hu

```
inline void makeroot(int x){
    access(x);splay(x);
    pushr(x);
}
```

关于pushdown和makeroot的一个相关的小问题详见下方update（关于pushdown的说明）

findroot(x)

找 x 所在原树的树根，主要用来判断两点之间的连通性（ $\text{findroot}(x)=\text{findroot}(y)$ 表明 x,y 在同一棵树中）

代码：

```
inline int findroot(R x){
    access(x); splay(x);
    while(c[x][0])pushdown(x),x=c[x][0];
    //如要获得正确的原树树根，一定pushdown！详见下方update（关于findroot中pushdown的
    splay(x); //保证复杂度
    return x;
}
```

同样利用性质1，不停找左儿子，因为其深度一定比当前点深度小。

split(x,y)

神奇的makeroot已经出现，我们终于可以访问指定的一条在原树中的链啦！
 $\text{split}(x,y)$ 定义为拉出 $x-y$ 的路径成为一个Splay（本蒟蒻以 y 作为该Splay的根）

```
inline void split(int x,int y){
    makeroot(x);
    access(y);splay(y);
}
```

x 成为了根，那么 x 到 y 的路径就可以用 $\text{access}(y)$ 直接拉出来了，将 y 转到Splay根后，我们就可以直接通过访问 y 来获取该路径的有关信息

link(x,y)

连一条 $x-y$ 的边（本蒟蒻使 x 的父亲指向 y ，连一条轻边）

```
inline bool link(int x,int y){
    makeroot(x);
    if(findroot(y)==x)return 0; //两点已经在同一子树中，再连边不合法
    f[x]=y;
    return 1;
}
```

如果题目保证连边合法，代码就可以更简单

```
inline void link(int x,int y){
    makeroot(x);
    f[x]=y;
}
```

cut(x,y)

将 $x-y$ 的边断开。
如果题目保证断边合法，倒是很方便。
使 x 为根后， y 的父亲一定指向 x ，深度相差一定是1。当 $\text{access}(y),\text{splay}(y)$ 以后， x 一定是 y 的左儿子，直接双向断开连接

```
inline void cut(int x,int y){
    split(x,y);
    f[x]=c[y][0]=0;
}
```

13. Re:Outsider (HNOI2019) ? ? ?

14. Re:Outsider (HNOI2019) @ Flash_Huqwq...

15. Re:Outsider (HNOI2019) 加油的说 --JSOI爆零珂学家yzhang

16. Re:Outsider (HNOI2019) @ smyr被您嘲讽了qqq...

17. Re:Outsider (HNOI2019) @ 自为风月马前卒您也加油啊！话说今年几省联考啊qqq...

18. Re:Outsider (HNOI2019) 省选加油呀！ --自为风月马前卒

19. Re:贪心相关/模拟网络流、费用流细节梳理/模板（贪心，模拟费用流，栈） Flash_Hu 吼强啊！！！

20. Re:Outsider (HNOI2019) Orz

阅读排行榜

1. LCT总结——概念篇+洛谷P3690[模板]Link Cut Tree(动态树) (LCT, Splay) (19551)
2. 模拟退火总结（模拟退火） (14871)
3. LCT总结——应用篇（附题单） (LCT) (5321)
4. 可持久化线段树总结（可持久化线段树，线段树） (4861)
5. DP的各种优化（动态规划，决策单调性，斜率优化，带权二分，单调栈，单调队列） (4465)
6. CDQ分治总结（CDQ，树状数组，归并排序） (3008)
7. 主席树总结（经典区间第k小问题）（主席树，线段树） (1891)
8. C++实用整数快速输入输出模板（C++） (1711)
9. Outsider (HNOI2019) (1574)
10. Codeforces访问提速攻略（小技巧） (1423)
11. 博弈论总结（只会打表，永不证明）（博弈论） (1386)
12. 洛谷P1337 【JSOI2004】平衡点 / 吊打XXX（模拟退火） (1376)
13. NOIP2018退役记（记事） (1376)
14. 由素数筛法到欧拉函数（欧拉函数，线性筛） (1024)
15. 有趣的线段树模板合集（线段树，最短/长路，单调栈，线段树合并，线段树分裂，树上差分，Tarjan-LCA，势能线段树，李超线段树） (994)
16. 无标号树的计数原理（组合计数，背包问题，隔板法，极点的重心） (864)
17. 数列分块总结——题目总版（hzwer分块九题及其他题目）（分块） (862)
18. Emacs配置（考场必备）（Emacs） (847)
19. 区间子集最大/最小异或和问题（线性基，树上差分） (799)
20. Fake or True (HNOI2018) (748)
21. 洛谷P4338 [ZJOI2018]历史（LCT，树形DP，树链剖分） (571)
22. 字符串数据结构模板/题单（后缀数组，后缀自动机，LCP，后缀平衡树，回文自动机） (560)
23. 组合数学知识要点(499)
24. 作为一个蒟蒻谈一点考试经验（总结） (491)
25. 洛谷P4180 [BJWC2010]次小生成树（最小生成树,LCT,主席树,倍增LCA,倍增,树链剖分） (488)

评论排行榜

1. LCT总结——概念篇+洛谷P3690[模板]Link Cut Tree(动态树) (LCT, Splay) (82)
2. DP的各种优化（动态规划，决策单调性，斜率优化，带权二分，单调栈，单调队列） (34)
3. 模拟退火总结（模拟退火） (17)
4. NOIP2018退役记（记事） (16)


```

pushup(y); //少了个儿子，也要上传一下
}

```

那如果不一定存在该边呢？

充分利用好Splay和LCT的各种基本性质吧！

正确姿势——先判一下连通性（注意 $findroot(y)$ 以后 x 成了根），再看看 x, y 是否有父子关系，还要看 y 是否有左儿子。

因为 $access(y)$ 以后，假如 y 与 x 在同一Splay中而没有直接连边，那么这条路径上就一定会有其它点，在中序遍历序列中的位置会介于 x 与 y 之间。

那么可能 y 的父亲就不是 x 了。

也可能 y 的父亲还是 x ，那么其它的点就在 y 的左子树中

只有三个条件都满足，才可以断掉。

```

inline bool cut(int x, int y){
    makeroot(x);
    if(findroot(y) != x || f[y] != x || c[y][0]) return 0;
    f[y] = c[x][1] = 0; //x在findroot(y)后被转到了根
    pushup(x);
    return 1;
}

```

如果维护了 $size$ ，还可以换一种判断

```

inline bool cut(int x, int y){
    makeroot(x);
    if(findroot(y) != x || sz[x] > 2) return 0;
    f[y] = c[x][1] = 0;
    pushup(x);
    return 1;
}

```

解释一下，如果他们直接连边的话， $access(y)$ 以后，为了满足性质1，该Splay只会剩下 x, y 两个点了。

反过来说，如果有其它的点， $size$ 不就大于2了么？

其实，还有一些LCT中的Splay的操作，跟我们以往学习的纯Splay的某些操作细节不甚相同。包括 $splay(x), rotate(x), nroot(x)$ （看到许多版本LCT写的是 $isroot(x)$ ，但我觉得反过来会方便些）

这些区别之处详见下面的模板题注释。

update (关于findroot中pushdown的说明)

蒟蒻真的一时没注意这个问题。。。。。。Splay根本没学好

找根的时候，当然不能保证Splay中到根的路径上的翻转标记全放掉。

所以最好把pushdown写上。

Candy巨佬的总结对pushdown问题有详细的分析

只不过蒟蒻后来经常习惯这样判连通性（我也不知道怎么养成的）

```

makeroot(x);
if(findroot(y) == x) //后续省略

```

这样好像没出过问题，那应该可以证明是没问题的（makeroot保证了 x 在LCT的顶端， $access(y)+splay(y)$ 以后，假如 x, y 在一个Splay里，那 x 到 y 的路径一定全部放完了标记）导致很久没有发现错误。。。。。。

另外提一下，假如LCT题目在维护连通性的情况中只可能出现合并而不会出现分离的话，其实可以用并查集哦！（实践证明findroot很慢）

这样的例子有不少，比如下面“维护链上的边权信息”部分的两道题都是的。

甚至听到Julao们说有少量题目还专门卡这个常数。。。。。。XZY巨佬的博客就提到了

update (关于pushdown的说明)

我pushdown和makeroot有时候会这样写，常数小一点

```

void pushdown(int x){
    if(r[x]){
        r[x] = 0;
    }
}

```

148

5. Fake or True (HNOI2018) (15)

6. 数列分块总结——题目总版 (hzwer分块目) (分块) (14)

郑州
32°

7. 有趣的线段树模板合集（线段树，最短/长路，单调栈，线段树合并，线段树分裂，树上差分，Tarjan-LCA，势能线段树，李超线段树）(13)

8. BSGS及扩展BSGS总结 (BSGS, map) (11)

9. 洛谷P1337 【JSOI2004】平衡点 / 吊打XXX (模拟退火) (11)

10. Heaven of Imaginary (PKUSC2018) (10)

11. 二进制高精度模板（高精度）(9)

12. 洛谷P4332 [SHOI2014] 二叉神经网络 (LCT, 树剖, 二分查找, 拓扑排序) (9)

13. CDQ分治总结 (CDQ, 树状数组, 归并排序) (8)

14. Outsider (HNOI2019) (8)

15. 珂朵莉树模板 (珂朵莉树) (7)

16. LCT总结——应用篇 (附题单) (LCT) (7)

17. 字符串数据结构模板/题单 (后缀数组, 后缀自动机, LCP, 后缀平衡树, 回文自动机) (7)

18. 洛谷P4581 [BJOI2014] 想法 (玄学算法, 拓扑排序) (7)

19. 基数排序模板 (基数排序, C++模板) (6)

20. Codeforces访问提速攻略 (小技巧) (6)

21. (伪) 再扩展中国剩余定理 (洛谷P4774 [NOI2018] 屠龙勇士) (中国剩余定理, 扩展欧几里德, multiset) (6)

22. 主席树总结 (经典区间第k小问题) (主席树, 线段树) (6)

23. Emacs配置 (考场必备) (Emacs) (5)

24. 博弈论总结 (只会打表, 永不证明) (博弈论) (5)

25. 贪心相关/模拟网络流、费用流细节梳理/模板 (贪心, 模拟费用流, 栈) (5)

推荐排行榜

1. LCT总结——概念篇+洛谷P3690[模板]Link Cut Tree(动态树) (LCT, Splay) (148)

2. 模拟退火总结 (模拟退火) (20)

3. DP的各种优化 (动态规划, 决策单调性, 斜率优化, 带权二分, 单调栈, 单调队列) (20)

4. 可持久化线段树总结 (可持久化线段树, 线段树) (15)

5. CDQ分治总结 (CDQ, 树状数组, 归并排序) (11)

6. LCT总结——应用篇 (附题单) (LCT) (11)

7. 洛谷P1337 【JSOI2004】平衡点 / 吊打XXX (模拟退火) (8)

8. 有趣的线段树模板合集 (线段树, 最短/长路, 单调栈, 线段树合并, 线段树分裂, 树上差分, Tarjan-LCA, 势能线段树, 李超线段树) (7)

9. 计算几何细节梳理&模板(4)

10. 洛谷U19464 山村游历(Wander) (LCT) (4)

11. FFT/NTT总结+洛谷P3803 【模板】多项式乘法 (FFT) (FFT/NTT) (3)

12. 主席树总结 (经典区间第k小问题) (主席树, 线段树) (3)

13. 洛谷P4338 [ZJOI2018] 历史 (LCT, 树形DP, 树链剖分) (3)

14. 博弈论总结 (只会打表, 永不证明) (博弈论) (3)

15. 数论细节梳理&模板(3)

16. 组合数学知识要点(2)

17. 洛谷P2900 [USACO08MAR] 土地征用 Land Acquisition (动态规划, 斜率优化, 决策单调性, 线性规划, 单调队列) (2)

18. 洛谷CF868F Yet Another Minimization Problem (动态规划, 决策单调性, 分治) (2)

19. 洛谷P4332 [SHOI2014] 二叉神经网络 (LCT, 树剖, 二分查找, 拓扑排序) (2)

20. 洛谷P3348 [ZJOI2016] 大森林 (LCT, 虚点, 树上差分) (2)

21. 洛谷P2542 [AHOI2005] 航线规划 (LCT, 双连通分量, 并查集) (2)

22. 洛谷P1516 青蛙的约会 (扩展欧几里德) (2)

23. 数列分块总结——题目总版 (hzwer分块九题及其他题目) (分块) (2)

24. 洛谷P4219 [BJOI2014] 大融合 (LCT) (2)

25. 洛谷P3203 [HNOI2010] 弹飞绵羊 (LCT, Splay) (2)

```

    int t=c[x][0];
    r[c[x][0]=c[x][1]]^=1;
    r[c[x][1]=t]^=1;
}
}
void makeroot(int x){
    access(x);splay(x);
    r[x]^=1;
}
}

```

这种写法等于说当x有懒标记时，x的左右儿子还是反的

那么如果findroot里实在要写pushdown，那么这种pushdown就会出现问題（参考评论区@zjp_shadow巨佬的指正）

再次update，蒟蒻发现这种问题还是可以避免的，若用这种pushdown，findroot这样写就好啦

```

inline int findroot(int x){
    access(x);splay(x);
    pushdown(x);
    while(lc)pushdown(x=lc);
    splay(x);
    return x;
}

```

当题目中维护的信息与左右儿子顺序有关的时候，pushdown如果用这种不严谨写法会是错的（比如[NOI2005]维护数列（这是Splay题）和洛谷P3613-睡觉困难综合征）

再次update，夏、沐瑾巨佬指出这种问题也是可以避免的，把pushup这样写就好啦

```

inline void pushup(int x){
    pushdown(lc);pushdown(rc);//加上两个
    //.....
}

```

所以此总结以及下面模板里的pushdown，常数大了一点点，却是更稳妥、严谨的写法

```

//pushr同上方makeroot部分
void pushdown(int x){
    if(r[x]){
        if(c[x][0])pushr(c[x][0]);//copy自模板，然后发现if可以不写
        if(c[x][1])pushr(c[x][1]);
        r[x]=0;
    }
}
void makeroot(int x){
    access(x);splay(x);
    pushr(x);//可以看到两种写法造成makeroot都是不一样的
}

```

这种写法等于说当x有懒标记时，x的左右儿子已经放到正确的位置了，只是儿子的儿子还是反的
那么这样就不会出问题啦

两种写法差别还确实有点大呢

模板

洛谷P3690 【模板】Link Cut Tree（动态树）（点击进入题目）

最基本的LCT操作都在这里，也没有更多额外的复杂操作了，确实很模板。

```

#include<bits/stdc++.h>
#define R register int
#define I inline void
#define G if(++ip==ie)if(fread(ip=buf,1,SZ,stdin))
#define lc c[x][0]
#define rc c[x][1]
using namespace std;
const int SZ=1<<19,N=3e5+9;

```

```

char buf[SZ],*ie=buf+SZ,*ip=ie-1;
inline int in(){
    G;while(*ip<'-' )G;
    R x=*ip&15;G;
    while(*ip>'-' ){x*=10;x+=*ip&15;G;}
    return x;
}
int f[N],c[N][2],v[N],s[N],st[N];
bool r[N];
inline bool nroot(R x){//判断节点是否为一个Splay的根 (与普通Splay的区别1)
    return c[f[x]][0]==x||c[f[x]][1]==x;
}
//原理很简单,如果连的是轻边,他的父亲的儿子没有它
I pushup(R x){//上传信息
    s[x]=s[lc]^s[rc]^v[x];
}
I pushr(R x){R t=lc;lc=rc;rc=t;r[x]^=1;}//翻转操作
I pushdown(R x){//判断并释放懒标记
    if(r[x]){
        if(lc)pushr(lc);
        if(rc)pushr(rc);
        r[x]=0;
    }
}
I rotate(R x){//一次旋转
    R y=f[x],z=f[y],k=c[y][1]==x,w=c[x][!k];
    if(nroot(y))c[z][c[z][1]==y]=x;c[x][!k]=y;c[y][k]=w;//额外注意if(nroot(y)
    if(w)f[w]=y;f[y]=x;f[x]=z;
    pushup(y);
}
I splay(R x){//只传了一个参数,因为所有操作的目标都是该Splay的根 (与普通Splay的区
    R y=x,z=0;
    st[++z]=y;//st为栈,暂存当前点到根的整条路径, pushdown时一定要从上往下放标记
    while(nroot(y))st[++z]=y=f[y];
    while(z)pushdown(st[z--]);
    while(nroot(x)){
        y=f[x];z=f[y];
        if(nroot(y))
            rotate((c[y][0]==x)^(c[z][0]==y)?x:y);
        rotate(x);
    }
    pushup(x);
}
/*当然了,其实利用函数堆栈也很方便,代替上面的手工栈,就像这样
I pushall(R x){
    if(nroot(x))pushall(f[x]);
    pushdown(x);
}*/
I access(R x){//访问
    for(R y=0;x=f[y=x])
        splay(x),rc=y,pushup(x);
}
I makeroot(R x){//换根
    access(x);splay(x);
    pushr(x);
}
int findroot(R x){//找根 (在真实的树中的)
    access(x);splay(x);
    while(lc)pushdown(x),x=lc;
    splay(x);
    return x;
}
I split(R x,R y){//提取路径
    makeroot(x);
    access(y);splay(y);
}

```



```
I link(R x,R y){//连边
    makeroot(x);
    if(findroot(y)!=x)f[x]=y;
}
I cut(R x,R y){//断边
    makeroot(x);
    if(findroot(y)==x&&f[y]==x&&!c[y][0]){
        f[y]=c[x][1]=0;
        pushup(x);
    }
}
int main()
{
    R n=in(),m=in();
    for(R i=1;i<=n;++i)v[i]=in();
    while(m--){
        R type=in(),x=in(),y=in();
        switch(type){
            case 0:split(x,y);printf("%d\n",s[y]);break;
            case 1:link(x,y);break;
            case 2:cut(x,y);break;
            case 3:splay(x);v[x]=y;//先把x转上去再改，不然会影响Splay信息的正确性
        }
    }
    return 0;
}
```

分类: 数据结构——链剖——LCT , 数据结构——平衡树——Splay , OI——算法总结 , OI——题解

标签: LCT , Splay

好文要顶

关注我

收藏该文

Flash_Hu

关注 - 78

粉丝 - 138

+加关注

« 上一篇: 洛谷P2633 Count on a tree (主席树，倍增LCA，树上差分)
» 下一篇: 洛谷P1501 [国家集训队]Tree II (LCT,Splay)
posted @ 2018-01-21 16:16 Flash_Hu 阅读(19551) 评论(82) 编辑 收藏

评论列表

#51楼 2018-12-25 08:31 smy

@ M_sea

M_sea!!!!!!!!!!!!!!

捕捉

#52楼 2018-12-25 08:32 M_sea

@ Flash_Hu

因为smy在fAke qwq

#53楼 2018-12-25 08:32 M_sea

@ smyj

您fAke死了