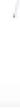


[首页](#)[新随笔](#)[管理](#)

JSOI爆零珂学家yzhang

如果结果不如你所愿，就在尘埃落定前奋力一搏



平衡树实际很简单的

以下讲解都以**Luogu P3369 【模板】普通平衡树**为例

我不会带指针的Splay，所以我就写非指针型的Splay

Splay是基于二叉查找树（bst）实现的

什么是二叉查找树呢？就是一棵树呗，但是这棵树满足性质：
一个节点的左孩子一定比它小，右孩子一定比它大

比如：



这就是一棵最基本二叉查找树

对于每次插入，它的期望复杂度大约是 $\log^2 n$ 级别的，但是存在极端情况，比如9999999 9999998 9999997.....1这种数据，会直接被卡成 n^2 级别

在这种情况下，平衡树出现了！

1. 定义Splay

```
struct node
{
    int v; //权值
    int fa; //父亲节点
    int ch[2]; //0代表左儿子, 1代表右儿子
    int rec; //这个权值的节点出现的次数
    int sum; //子节点的数量
}tree[N]; //N为节点最多有多少
int tot; //tot表示不算重复的有多少节点
```

□ 公告

昵称： JSOI爆零珂学家
yzhang

园龄： 1年

粉丝： 40

关注： 11

[+加关注](#)

□ 最新随笔

1. 2019~2020颓gal计划

2. JXOI2018简要题解

3. TJOI2018简要题解

4. BJOI2018简要题解

5. 【题解】Luogu P4091 [HE...

6. 【题解】Luogu P5301 [G...

7. 【题解】Luogu P5291 [十...

8. 【题解】Luogu P3349 [ZJ...

2.Splay的核心

Rotate

首先考虑一下，我们要把一个点挪到根，那我们首先要知道怎么让一个点挪到它的父节点

情况1：

当x是y的左孩子时



ABC实际可以是子树，但这里假设ABC都是点

这时候如果我们让x成为y的父亲，只会影响到3组点的关系

B与x, x与y, x与R

根据二叉排序树的性质

B会成为y的左儿子

y会成为x的右儿子

x会成为R的儿子，具体是什么儿子，这个要看y是R的啥儿子



情况2：

当x是y的右孩子

本质上是和情况1一样的qaq



旋转后变成



能不能把这两种情况合并呢qaq？结果是肯定的

我们需要一个函数来确定这个节点是他父节点的左孩子还是右孩子

```
inline bool findd(register int x)
{
    return tree[tree[x].fa].ch[0]==x?0:1;
}
```

如果是左孩子的话会返回0，右孩子会返回1

那么我们不难得得到R, Y, x这三个节点的信息

```
int Y=tree[x].fa;
int R=tree[Y].fa;
```

9. 【题解】Luogu P5327 [ZJ...]

10. 【题解】Luogu P5319 [B...

积分与排名

积分 - 24186

排名 - 27081

友链

duyi

Sai_0511

little_sun

M_sea

yurzhang

suwakow

oier_hzy

阅读排行榜

1. 珂朵莉树详解(10019)

2. NOIP2018游记(2528)

3. Splay详解(1593)

4. JSOI2020备考知识点复习...

5. 主席树详解(769)

6. 2019各省省选试题选做及...

7. NOI2019网络同步赛游记(...

8. 莫队详解(535)

9. 质数算法略解(436)

10. 日记 (菜的连比赛都参加...

```
int Yson=findd(x);
int Rson=findd(Y);
```

B的情况我们可以根据x的情况推算出来，根据 \wedge 运算的性质， $0 \wedge 1 = 1, 1 \wedge 1 = 0, 2 \wedge 1 = 3, 3 \wedge 1 = 2$ ，而且B相对于x的位置一定是与x相对于Y的位置是相反的

(否则在旋转的过程中不会对B产生影响)

```
int B=tree[x].ch[Yson^1];
```

然后我们考虑连接的过程

根据上面的图，不难得出 (自己向上翻qaq)

1.B成为Y的哪个儿子与x是Y的哪个儿子是一样的

2.Y成为x的哪个儿子与x是Y的哪个儿子相反

3.X成为R的哪个儿子与Y是R的哪个儿子相同

```
connect(B,Y,Yson);
connect(Y,x,Yson^1);
connect(x,R,Rson);
```

connect函数也很好写

```
inline void connect(register int x,register int fa,register int son) //把x转为fa的儿子(son是0/1, 表示左孩子或右孩子)
{
    tree[x].fa=fa;
    tree[fa].ch[son]=x;
}
```

Rotate代码总览 (旋转完不要忘了update) :

```
inline void update(register int x)
{
    tree[x].sum=tree[tree[x].ch[0]].sum+tree[tree[x]
    .ch[1]].sum+tree[x].rec;
}
inline bool findd(register int x)
{
    return tree[tree[x].fa].ch[0]==x?0:1;
}
inline void connect(register int x,register int fa,register int son) //把x转为fa的儿子(son是0/1, 表示左孩子或右孩子)
{
    tree[x].fa=fa;
    tree[fa].ch[son]=x;
}
inline void rotate(register int x)
{
    int Y=tree[x].fa;
    int R=tree[Y].fa;
    int Yson=findd(x);
    int Rson=findd(Y);
```



```

int B=tree[x].ch[Yson^1];
connect(B,Y,Yson);
connect(Y,x,Yson^1);
connect(x,R,Rson);
update(Y),update(x);
}

```

Splay

Splay(x,to)是实现把x节点搬到to节点

最简单的办法，对于x这个节点，每次上旋直到to

但是！

毒瘤的出题人可以构造数据把上面的这种方法卡到 n^2

qaq*

下面我们介绍一下双旋的Splay

这里的情况有很多，但是总的来说就三种情况

1.to是x的爸爸，

这样的话吧x旋转上去就好

```

if(tree[tree[x].fa].fa==to)
    rotate(x);

```

2.x和他爸爸和他爸爸的爸爸在一条线上（文字游戏）

其实就是 $\text{findd}(x)=\text{findd}(\text{tree}[x].fa)$

这时候先把y旋转上去，再把x旋转上去就好

```

if(findd(x)==find(tree[x].fa))
    rotate(tree[x].fa),rotate(x);

```

3.x和他爸爸和他爸爸的爸爸不在一条线上（和2相反）

这时候把x旋转两次就好

splay函数的代码：

```

inline void splay(register int x,register int
to)
{
    to=tree[to].fa;
    while(tree[x].fa!=to)
    {
        int y=tree[x].fa;
        if(tree[y].fa==to)
            rotate(x);
        else if(findd(x)==findd(y))
            rotate(y),rotate(x);
        else
            rotate(x),rotate(x);
    }
}

```

Splay的核心代码到此结束



剩下的就是一些其他的东西（虽说有的也挺重要qaq）

3.其他的一些函数

insert

根据前面讲的，我们在插入一个数之后，需要将其旋转到根

首先，当这棵树已经没有节点的时候，我们直接新建一个节点就好

```
inline int newpoint(register int v, register int fa)
{
    tree[++tot].fa=fa;
    tree[tot].v=v;
    tree[tot].sum=tree[tot].rec=1;
    return tot;
}
```

然后，当这棵树有节点的时候，我们根据二叉查找树的性质，不断向下走，直到找到一个可以插入的点，注意在走的时候需要更新一个每个节点的sum值

```
inline void Insert(register int x)
{
    int now=tree[0].ch[1];
    if(tree[0].ch[1]==0)
    {
        newpoint(x,0);
        tree[0].ch[1]=tot;
    }
    else
    {
        while(19260817)
        {
            ++tree[now].sum;
            if(tree[now].v==x)
            {
                ++tree[now].rec;
                splay(now,tree[0].ch[1]);
                return;
            }
            int nxt=x<tree[now].v?0:1;
            if(!tree[now].ch[nxt])
            {
                int p=newpoint(x,now);
                tree[now].ch[nxt]=p;
                splay(p,tree[0].ch[1]);
                return;
            }
            now=tree[now].ch[nxt];
        }
    }
}
```



delete

删除的功能是：删除权值为v的节点

我们不难想到：我们可以先找到他的位置，再把这个节点删掉

找位置用find函数，不要和rotate的find搞混

```
inline int find(register int v)
{
    int now=tree[0].ch[1];
    while(19260817)
    {
        if(tree[now].v==v)
        {
            splay(now,tree[0].ch[1]);
            return now;
        }
        int nxt=v<tree[now].v?0:1;
        if(!tree[now].ch[nxt])
            return 0;
        now=tree[now].ch[nxt];
    }
}
```



下面我们需要删除函数

怎么样才能保证删除节点后整棵树还满足二叉查找树的性质

此时会出现几种情况

1.权值为v的节点已经出现过

这时候直接把他的rec和sum减去1就好

2.本节点没有左右儿子

这样的话就成了一棵空树

3.本节点没有左儿子

直接把他的右儿子设置成根

4.既有左儿子，又有右儿子

在它的左儿子中找到最大的，旋转到根，把它的右儿子当做根(也就是它最大的左儿子)的右儿子

最后把这个节点删掉就好

delete的代码

```
inline void delete(register int x)
{
    int pos=find(x);
    if(!pos)
        return;
    if(tree[pos].rec>1)
    {
        --tree[pos].rec;
        --tree[pos].sum;
```

```

    }
    else
    {
        if(!tree[pos].ch[0]&&!tree[pos].ch[1])
            tree[0].ch[1]=0;
        else if(!tree[pos].ch[0])
        {
            tree[0].ch[1]=tree[pos].ch[1];
            tree[tree[0].ch[1]].fa=0;
        }
        else
        {
            int left=tree[pos].ch[0];
            while(tree[left].ch[1])
                left=tree[left].ch[1];
            splay(left,tree[pos].ch[0]);
            connect(tree[pos].ch[1],left,1);
            connect(left,0,1);
            update(left);
        }
    }
}

```

rank

1.查询x数的排名

十分简短

```

inline int rank(register int v)
{
    int pos=find(v);
    return tree[tree[pos].ch[0]].sum+1;
}

```

2.查询排名为x的数

这个操作就是上面那个操作的逆向操作

```

inline int arank(register int x)
{
    int now=tree[0].ch[1];
    while(19260817)
    {
        int used=tree[now].sum-
tree[tree[now].ch[1]].sum;
        if(x>tree[tree[now].ch[0]].sum&&x<=used)
        {
            splay(now,tree[0].ch[1]);
            return tree[now].v;
        }
        if(x<used)
            now=tree[now].ch[0];
        else
            x-=used,now=tree[now].ch[1];
    }
}

```

求前驱和后继



前驱

这个更容易，我们可以维护一个ans变量，然后对整棵树进行遍历，同时更新ans

```
inline int lower(register int v)
{
    int now=tree[0].ch[1];
    int ans=-inf;
    while(now)
    {
        if(tree[now].v<v&&tree[now].v>ans)
            ans=tree[now].v;
        if(v>tree[now].v)
            now=tree[now].ch[1];
        else
            now=tree[now].ch[0];
    }
    return ans;
}
```



后继

和前驱差不多

```
inline int upper(register int v)
{
    int now=tree[0].ch[1];
    int ans=inf;
    while(now)
    {
        if(tree[now].v>v&&tree[now].v<ans)
            ans=tree[now].v;
        if(v<tree[now].v)
            now=tree[now].ch[0];
        else
            now=tree[now].ch[1];
    }
    return ans;
}
```



4.Splay整体代码

```
#pragma GCC optimize("O3")
#include <bits/stdc++.h>
#define N 100005
#define inf 1000000005
using namespace std;
inline int read()
{
    register int x=0,f=1;register char
    ch=getchar();
    while(ch<'0'||ch>'9') {if(ch=='-'
    ') f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9') x=(x<<3)+(x<<1)+ch-
    '0',ch=getchar();
    return x*f;
}
inline void write(register int x)
```

```

{
    if(!x) putchar('0'); if(x<0) x=-x,putchar('-');
    static int sta[36];int tot=0;
    while(x) sta[tot++]=x%10,x/=10;
    while(tot) putchar(sta[--tot]+48);
}

struct node
{
    int v;
    int fa;
    int ch[2];
    int rec;
    int sum;
}tree[N];
int tot;
inline void update(register int x)
{

    tree[x].sum=tree[tree[x].ch[0]].sum+tree[tree[x]
        .ch[1]].sum+tree[x].rec;
}
inline bool findd(register int x)
{
    return tree[tree[x].fa].ch[0]==x?0:1;
}
inline void connect(register int x,register int
fa,register int son) //把x转为fa的son(son是0/1, 表
示左孩子或右孩子)
{
    tree[x].fa=fa;
    tree[fa].ch[son]=x;
}
inline void rotate(register int x)
{
    int Y=tree[x].fa;
    int R=tree[Y].fa;
    int Yson=findd(x);
    int Rson=findd(Y);
    int B=tree[x].ch[Yson^1];
    connect(B,Y,Yson);
    connect(Y,x,Yson^1);
    connect(x,R,Rson);
    update(Y),update(x);
}
inline void splay(register int x,register int
to)
{
    to=tree[to].fa;
    while(tree[x].fa!=to)
    {
        int y=tree[x].fa;
        if(tree[y].fa==to)
            rotate(x);
        else if(findd(x)==findd(y))
            rotate(y),rotate(x);
        else
            rotate(x),rotate(x);
    }
}
}

```



```

inline int newpoint(register int v,register int fa)
{
    tree[++tot].fa=fa;
    tree[tot].v=v;
    tree[tot].sum=tree[tot].rec=1;
    return tot;
}
inline void Insert(register int x)
{
    int now=tree[0].ch[1];
    if(tree[0].ch[1]==0)
    {
        newpoint(x,0);
        tree[0].ch[1]=tot;
    }
    else
    {
        while(19260817)
        {
            ++tree[now].sum;
            if(tree[now].v==x)
            {
                ++tree[now].rec;
                splay(now,tree[0].ch[1]);
                return;
            }
            int nxt=x<tree[now].v?0:1;
            if(!tree[now].ch[nxt])
            {
                int p=newpoint(x,now);
                tree[now].ch[nxt]=p;
                splay(p,tree[0].ch[1]);
                return;
            }
            now=tree[now].ch[nxt];
        }
    }
}
inline int find(register int v)
{
    int now=tree[0].ch[1];
    while(19260817)
    {
        if(tree[now].v==v)
        {
            splay(now,tree[0].ch[1]);
            return now;
        }
        int nxt=v<tree[now].v?0:1;
        if(!tree[now].ch[nxt])
            return 0;
        now=tree[now].ch[nxt];
    }
}
inline void delete(register int x)
{
    int pos=find(x);
    if(!pos)

```



```

        return;
    if(tree[pos].rec>1)
    {
        --tree[pos].rec;
        --tree[pos].sum;
    }
    else
    {
        if(!tree[pos].ch[0]&&!tree[pos].ch[1])
            tree[0].ch[1]=0;
        else if(!tree[pos].ch[0])
        {
            tree[0].ch[1]=tree[pos].ch[1];
            tree[tree[0].ch[1]].fa=0;
        }
        else
        {
            int left=tree[pos].ch[0];
            while(tree[left].ch[1])
                left=tree[left].ch[1];
            splay(left,tree[pos].ch[0]);
            connect(tree[pos].ch[1],left,1);
            connect(left,0,1);
            update(left);
        }
    }
}
inline int rank(register int v)
{
    int pos=find(v);
    return tree[tree[pos].ch[0]].sum+1;
}
inline int arank(register int x)
{
    int now=tree[0].ch[1];
    while(19260817)
    {
        int used=tree[now].sum-
tree[tree[now].ch[1]].sum;
        if(x>tree[tree[now].ch[0]].sum&&x<=used)
        {
            splay(now,tree[0].ch[1]);
            return tree[now].v;
        }
        if(x<used)
            now=tree[now].ch[0];
        else
            x-=used,now=tree[now].ch[1];
    }
}
inline int lower(register int v)
{
    int now=tree[0].ch[1];
    int ans=-inf;
    while(now)
    {
        if(tree[now].v<v&&tree[now].v>ans)
            ans=tree[now].v;
        if(v>tree[now].v)

```



```

        now=tree[now].ch[1];
    else
        now=tree[now].ch[0];
    }
    return ans;
}
inline int upper(register int v)
{
    int now=tree[0].ch[1];
    int ans=inf;
    while(now)
    {
        if(tree[now].v>v&&tree[now].v<ans)
            ans=tree[now].v;
        if(v<tree[now].v)
            now=tree[now].ch[0];
        else
            now=tree[now].ch[1];
    }
    return ans;
}
int main()
{
    int m=read();
    while(m--)
    {
        int opt=read(),x=read();
        if(opt==1)
            Insert(x);
        else if(opt==2)
            delet(x);
        else if(opt==3)
        {
            write(rank(x));
            printf("\n");
        }
        else if(opt==4)
        {
            write(arank(x));
            printf("\n");
        }
        else if(opt==5)
        {
            write(lower(x));
            printf("\n");
        }
        else
        {
            write(upper(x));
            printf("\n");
        }
    }
    return 0;
}

```



5.相关题目

1.Luogu P2234 [HNOI2002]营业额统计

平衡树板题

2.Luogu P1503 鬼子进村

平衡树板题

3.Luogu P3871 [TJOI2010]中位数

平衡树板题

4.Luogu P1533 可怜的狗狗

莫队+平衡树苟过

5.Luogu P2073 送花

平衡树板题

以上是splay的基本应用qaq

还有一种操作没讲，就是如何进行区间操作

实现起来很简单

假设我们要在[l,r]之间上搞事情，我们首先把l的前驱旋转到根节点，再把r的后继转到根节点的右儿子

那么此时根节点右儿子的左儿子代表的就是区间[l,r]

这应该很好理解qaq

然后就可以像线段树的lazy标记一样，给区间l,r打上标记，延迟更新，比如区间反转的时候更新的时候直接交换左右儿子

我们下面以P3391 【模板】文艺平衡树（Splay）为例

这里有一个奇技淫巧：如果一个区间被打了两次，那么就相当于不打

所以我们用一个bool变量来储存该节点是否需要被旋转

pushdown函数可以这么写（rev就是翻转标记）

```
inline void pushdown(register int x)
{
    if(tree[x].rev)
    {
        swap(tree[x].ch[0],tree[x].ch[1]);
        tree[tree[x].ch[0]].rev^=1;
        tree[tree[x].ch[1]].rev^=1;
        tree[x].rev=0;
    }
}
```

这道题完整代码

```
#include <bits/stdc++.h>
#define N 100005
#define inf 0x7fffffff
using namespace std;
inline int read()
```



```

{
    register int x=0,f=1;register char
ch=getchar();
    while(ch<'0'||ch>'9') {if(ch=='-'
') f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9') x=(x<<3)+(x<<1)+ch-
'0',ch=getchar();
    return x*f;
}
inline void write(register int x)
{
    if(!x) putchar('0');if(x<0) x=-x,putchar('-');
    static int sta[36];int cnt=0;
    while(x) sta[cnt++]=x%10,x/=10;
    while(cnt) putchar(sta[--cnt]+48);
}
int n,m;
struct node{
    int fa,ch[2],tot;
    bool rev;
}tree[N];
int root,PosL,PosR;
inline bool findd(register int x)
{
    return x==tree[tree[x].fa].ch[1];
}
inline void connect(register int x,register int
fa,register int son)
{
    tree[x].fa=fa;
    tree[fa].ch[son]=x;
}
inline void update(register int x)
{
    tree[x].tot=tree[tree[x].ch[0]].tot+tree[tree[x]
.ch[1]].tot+1;
}
inline void rotate(register int x)
{
    int Y=tree[x].fa;
    if(Y==root)
        root=x;
    int R=tree[Y].fa;
    int Yson=findd(x);
    int Rson=findd(Y);
    int B=tree[x].ch[Yson^1];
    connect(B,Y,Yson);
    connect(Y,x,Yson^1);
    connect(x,R,Rson);
    update(Y);
    update(x);
}
inline void splay(register int x,register int
to)
{
    while(tree[x].fa!=to)
    {
        int y=tree[x].fa;
        if(y==root)
            rotate(x);
        else
            if(y==tree[y].ch[0])
                if(x==tree[y].ch[1])
                    double_rotate(y,x);
                else
                    single_rotate(y,x);
            else
                if(x==tree[y].ch[0])
                    single_rotate(y,x);
                else
                    double_rotate(y,x);
    }
}

```



```

        if(tree[y].fa==to)
            rotate(x);
        else if(findd(x)==findd(y))
            rotate(y),rotate(x);
        else
            rotate(x),rotate(x);
    }
    update(x);
}

inline int buildsplay(register int l,register
int r)
{
    if(l>r)
        return 0;
    int mid=l+r>>1;
    connect(buildsplay(l,mid-1),mid,0);
    connect(buildsplay(mid+1,r),mid,1);
    tree[mid].rev=0;
    update(mid);
    return mid;
}
inline void pushdown(register int x)
{
    if(tree[x].rev)
    {
        swap(tree[x].ch[0],tree[x].ch[1]);
        tree[tree[x].ch[0]].rev^=1;
        tree[tree[x].ch[1]].rev^=1;
        tree[x].rev=0;
    }
}
inline int find(register int x)
{
    int now=root;
    --x;
    pushdown(now);
    while(x!=tree[tree[now].ch[0]].tot)
    {
        if(tree[tree[now].ch[0]].tot<x)
            x--;
        =tree[tree[now].ch[0]].tot+1,now=tree[now].ch[1]
    ;
        else
            now=tree[now].ch[0];
        pushdown(now);
    }
    return now;
}
inline void print(register int now)
{
    if(!now)
        return;
    pushdown(now);
    print(tree[now].ch[0]);
    if(now!=1&&now!=n+2)
        write(now-1,putchar(' '));
    print(tree[now].ch[1]);
}
int main()

```



```

{
    n=read(),m=read();
    root=buildsplay(1,n+2);
    while(m--)
    {
        int l=read(),r=read();
        PosL=find(l);
        splay(PosL,0);
        PosR=find(r+2);
        splay(PosR,root);
        tree[tree[PosR].ch[0]].rev^=1;
    }
    print(root);
    return 0;
}

```

[平衡树](#)[好文要顶](#)[关注我](#)[收藏该文](#)

[上一篇：JSOI2020备考知识点复习](#)

[下一篇：](#)

【题解】P2234 [HNOI2002]营业额统计

posted @ 2018-11-14 14:45 JSOI爆零珂学家yzhang 阅读

(1592) 评论(3) 编辑 收藏



#1楼 2019-08-19 20:53 燥VOLCANO

博主您好，文章很棒，只是...图片挂了(((φ(°口°;)φ)))

[支持\(0\)](#)

[反对\(0\)](#)

#2楼 2019-08-20 13:53 dsjkafdsaf

%%%%%

[支持\(0\)](#)

[反对\(0\)](#)



2019-08-20 13:53 dsjkafdsaf

支持(0) 反对(0)



注册用户登录后才能发表评论，请[登录](#)或[注册](#)，
[访问](#)网站首页。

Copyright © 2019 JSOI爆零珂学家y়zhang
Powered by .NET Core 3.0 Preview 8 on Linux

