洛谷P4299 首都（BZOJ3510）（LCT，树的重心，二分查找）

Update:原来的洛谷U21715已成坑qwq

已经被某位管理员巨佬放进公共题库啦！又可以多一个AC记录啦！

洛谷题目传送门

其实也可以到这里交啦

思路分析

动态维护树的重心

题目中说到国家的首都会选在某个使得其他城市到它距离之和最小的城市，那不就是树的重心了嘛。树的重心性质真的很好，看看wuhulala巨佬的这篇博客。

网上大多数解法都是启发式合并。利用了以重心为根的子树大小不超过原树的一半，每次合并两颗树的时候，小的往大的上面合并，而且是一个点一个点地link上去，每次link完检查一下这个子树如果超过了当前整个树大小的一半，就把重心向当前点移动一下。这样做，合并次数上限$O(N\log N)$，每次更新重心也需要$O(\log N)$的复杂度，所以复杂度极限是$O(N\log^2 N)$，还不够优秀。

因此我就想再优化一下算法。复杂度瓶颈就在于一个一个link，能不能考虑直接连接两颗树对重心的影响呢？

答案是肯定的。又要用到一个性质——连接两颗树后，新的重心一定在原来的两个重心的路径上。这个证明不难，反证一下就好了：对于在路径上的点来说，它的最大子树一定是往原来两个重心方向上的那两个子树的其中一个，它的其它子树肯定比这个最大子树要小，不然重心就不会是那两个点了；而对于不在路径上的点，它的最大子树是往原来两个重心方向上的那个子树，这显然劣于在路径上的点。

那就可以直接link然后把链提出来去找了。我们当然不能把整条链全找遍了，毕竟重心的性质那么多，也要好好利用嘛。

我们可以这样形象地理解：提出来的链就是一个序列，每个点都带了一些若干大小的虚子树。把子树大小和看成序列的权值，而最接近中位数的地方就是重心了。

(下面s和si的定义和蒟蒻的LCT总结中是一样的)

具体找法：类似树上二分，我们需要不断逼近树的重心的位置。记下lsum表示当前链中搜索区间左端点以左的子树大小，rsum表示右端点以右的。x的整个子树就表示了当前搜索区间，在中序遍历中x把搜索区间分成了左右两块（在Splay中对应x的左子树和右子树）。

如果x左子树的s加上lsum和x右子树的s加上rsum（对应原树中x往原来两个重心方向上的两个子树大小）都不超过新树总大小的一半，那么x当然就是重心啦！之前已经提到x其它的子树大小就不用考虑了。当然，如果总大小是奇数，重心只会有一个，那就找到了。否则，因为必须编号最小，所以还要继续找下去。

当我们没有确定答案时，还要继续找下去，那么就要跳儿子了。x把整个链分成了左右两个部分，而重心显然会在大小更大的一部分中，这个也应该好证明。如果左边比右边小，那就跳右儿子继续找。这时候当前搜索区间减小了，搜索区间以外的部分增大了，lsum应该加上si[x]+1。反之亦然。如果跳进了空儿子，那肯定所有情况都考虑完了，直接结束查找。

当然，重心找到了就还是要伸展一下，保证复杂度。（蒟蒻造数据的时候懒得去卡了qwq）

这一部分套用Splay的复杂度，是均摊$O(\log N)$的，总复杂度也就降到了$O(N\log N)$。

findroot实在很慢，于是可以写个并查集来维护每个点所在树的重心。

```
#include<cstdio>
#include<cstdlib>
#define R register int
#define I inline void
const int N=100009,INF=2147483647;
int f[N],c[N][2],si[N],s[N],h[N];
```

```
bool r[N];
#define lc c[x][0]
#define rc c[x][1]
inline bool nroot(R x){return c[f[x]][0]==x||c[f[x]][1]==x;}
I pushup(R x){
    s[x]=s[lc]+s[rc]+si[x]+1;
}
I pushdown(R x){
    if(r[x]){
        R t=lc;lc=rc;rc=t;
        r[lc]^=1;r[rc]^=1;r[x]=0;
    }
}
I pushall(R x){
    if(nroot(x))pushall(f[x]);
    pushdown(x);
}
I rotate(R x){
    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;
    f[f[f[c[c[x][!k]=y][k]=w]=y]=x]=z;pushup(y);//为三行rotate打call
}
I splay(R x){
    pushall(x);
    R y;
    while(nroot(x)){
        if(nroot(y=f[x]))rotate((c[f[y]][0]==y)^(c[y][0]==x)?x:y);
        rotate(x);
    }
    pushup(x);
}
I access(R x){
    for(R y=0;x;x=f[y=x]){
        splay(x);
        si[x]+=s[rc];
        si[x]-=s[rc=y];
        pushup(x);
    }
}
I makeroot(R x){
    access(x);splay(x);
    r[x]^=1;
}
I split(R x,R y){
    makeroot(x);
    access(y);splay(y);
}
I link(R x,R y){
    split(x,y);
    si[f[x]=y]+=s[x];
    pushup(y);
}
int geth(R x){
    if(h[x]==x)return x;
    return h[x]=geth(h[x]);
}
inline int update(R x){
    R l,r,ji=s[x]&1,sum=s[x]>>1,lsum=0,rsum=0,newp=INF,nowl,nowr;
    while(x){
        pushdown(x);//注意pushdown
        nowl=s[l=lc]+lsum;nowr=s[r=rc]+rsum;
        if(nowl<=sum&&nowr<=sum){
            if(ji){newp=x;break;}//剪枝，确定已经直接找到
            else if(newp>x)newp=x;//选编号最小的
        }
```

```
        if(nowl<nowr)lsum+=s[l]+si[x]+1,x=r;
        else         rsum+=s[r]+si[x]+1,x=l;//缩小搜索区间
    }
    splay(newp);//保证复杂度
    return newp;
}
#define G ch=getchar()
#define gc G;while(ch<'-')G
#define in(z) gc;z=ch&15;G;while(ch>'-')z*=10,z+=ch&15,G;
int main(){
    register char ch;
    R n,m,x,y,z,Xor=0;
    in(n);in(m);
    for(R i=1;i<=n;++i)s[i]=1,h[i]=i,Xor^=i;
    while(m--){
        gc;
        switch(ch){
            case 'A':in(x);in(y);link(x,y);
                split(x=geth(x),y=geth(y));//提出原重心路径
                z=update(y);
                Xor=Xor^x^y^z;
                h[x]=h[y]=h[z]=z;//并查集维护好
                break;
            case 'Q':in(x);printf("%d\n",geth(x));break;
            case 'X':gc;gc;printf("%d\n",Xor);
        }
    }
    return 0;
}
```

分类: 数据结构——链剖——LCT , 图论——树——树的重心 , 算法——二分——二分查找 , OI——题解

标签: LCT , 树的重心 , 二分查找

好文要顶 关注我 收藏该文

Flash_Hu
关注 - 78
粉丝 - 152
+加关注

« 上一篇: 洛谷P4172 [WC2006]水管局长 （LCT，最小生成树）
» 下一篇: 洛谷SP16549 QTREE6 - Query on a tree VI （LCT）

posted @ 2018-03-22 19:42 Flash_Hu 阅读(459) 评论(3) 编辑 收藏

评论列表

#1楼 2019-03-01 16:09 **_____夏丶沐瑾**

请问判定重心的方法其正确性如何证明啊？ T-T

#2楼 [楼主] 2019-03-01 17:15 **Flash_Hu**

@ _____夏丶沐瑾
加了一点蒟蒻的描述qaq

#3楼 2019-03-01 17:16 **_____夏丶沐瑾**

Orz

刷新评论 刷新页面 返回顶部

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。