

各种树模板(splay,线段树,可持久化线段树...)

这是裸的排序Splay

AC tyvj1728 普通平衡树

```
1 #include <cstdio>
2 #include <iostream>
3 #include <fstream>
4
5 #include <cstdlib>
6 #include <cstring>
7 #include <cmath>
8 #include <algorithm>
9
10 typedef long long int ll;
11 typedef double db;
12
13 using namespace std;
14
15 struct SplayTree
16 {
17     struct node
18     {
19         int v;
20         int tot;
21         node*s[2];
22         node*f;
23
24         void update()
25         {
26             tot=s[0]->tot + s[1]->tot +1;
27         }
28     };
29     node*pool;
30     node*nt;
31     node*nil;
32     node*newnode (node*f, int v)
33     {
34         nt->v=v;
35         nt->tot=1;
36         nt->s[0]=nt->s[1]=nil;
37         nt->f=f;
38         return nt++;
39     }
40
41     node*root;
42
43     SplayTree(int size)
44     {
45         pool=new node[size+1];
46         nt=pool;
47         nil=newnode (NULL,-1);
48         nil->tot=0;
49         nil->f=nil->s[0]=nil->s[1]=nil;
50         root=nil;
51     }
52
53     //=====
54
55     void update (node*x)
56     {
57         x->tot= x->s[0]->tot + x->s[1]->tot +1;
58     }
59
60     void rot (node*x)
61     {
62         if (x==nil) return ;
63
64     }
```

昵称: DragoonKiller

园龄: 1个月

粉丝: 2

关注: 1

+加关注

2015年4月						
日	一	二	三	四	五	六
29	30	31	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

搜索

找找看

谷歌搜索

- 常用链接
- 我的随笔

我的评论

我的参与

最新评论

我的标签

更多链接

- 随笔分类
- OI(11)

模板(14)

算法(13)

- 随笔档案
- 2015年4月 (3)

2015年3月 (5)

2015年2月 (7)

2015年1月 (3)

2014年12月 (4)

2014年11月 (2)

- 最新评论
1. Re:生命游戏和随机数之间某种不可言说的秘密

如果要n个不重复的随机数的话（重复当然可以= =），可以找一个>n的素数然后找一个原根a然后随便找一个b然后从a^b mod n得到随机数，然后b+=1。然后就能生的n-1个随机数了.....

--iwtwii
2. Re:并查集,以及可拆分并查集

@iwtwii我不知道怎样的数据能卡啊囧...

--DragoonKiller

```

64     node*y=x->f;
65     int k=(x==y->s[0]);
66
67     y->s[k^1]=x->s[k];
68     if(x->s[k]!=nil) x->s[k]->f=y;
69
70     x->f=y->f;
71     if(y==y->f->s[0]) y->f->s[0]=x;
72     else if(y==y->f->s[1]) y->f->s[1]=x;
73
74     y->f=x;
75     x->s[k]=y;
76
77     y->update();
78 }
79
80 void splay(node*x) { splay(x,nil); }
81 void splay(node*x,node*t)
82 {
83     if(x==nil) return ;
84     while(x->f!=t)
85     {
86         node*y=x->f;
87         if(y->f!=t)
88             if((x==y->s[0])^(y==y->f->s[0]))
89                 rot(x); else rot(y);
90         rot(x);
91     }
92     x->update();
93
94     if(t==nil) root=x;
95 }
96
97 //=====
98
99 void Insert(int v)
100 {
101     if(root==nil)
102     {
103         root=newnode(nil,v);
104         return ;
105     }
106
107     node*x=root;
108     node*y=x;
109     while(x!=nil)
110     {
111         y=x;
112         if(v<x->v) x=x->s[0];
113         else x=x->s[1];
114     }
115
116     int k=(v<y->v);
117     y->s[k]=newnode(y,v);
118     splay(y->s[k]);
119 }
120
121
122 node*Find(int v)
123 {
124     node*x=root;
125     node*y=x;
126     node*r=nil;
127     while(x!=nil)
128     {
129         y=x;
130         if(x->v==v) r=x;
131         if(v<=x->v) x=x->s[0];
132         else x=x->s[1];
133     }
134     splay(y);
135     return r;
136 }
137
138 node* FindRank(int k)
139 {
140     node*x=root;
141     node*y=x;
142     while(x!=nil)
143     {

```

### 3. Re:并查集,以及可拆分并查集

@DragoonKiller反正我没遇到过....随便出一个nlogn的数据不就能卡了吗...

--iwtwiioi

### 4. Re:各种树模板(splay,线段树,可持久化线段树,...)

@iwtwiioi噢我还以为数据范围是一样的...

--DragoonKiller

### 5. Re:并查集,以及可拆分并查集

@iwtwiioi有卡掉nlogn并查的题么.....

--DragoonKiller

### 6. Re:并查集,以及可拆分并查集

那个普通并查集的复杂度不是 $O(\alpha(n))$ 的...  
你没有加启发式合并...要加了启发式合并+路径压缩的并查集才是 $O(\alpha(n))$ 的..

--iwtwiioi

### 7. Re:各种树模板(splay,线段树,可持久化线段树,...)

妈呀....你交我的代码不改数据范围tle了很正常啊= =

--iwtwiioi

## 阅读排行榜

1. 【DK的柳高OI课(zhuang)程(bi)目录】(47)

2. 各种树模板(splay,线段树,可持久化线段树...)(45)

3. 并查集,以及可拆分并查集(31)

4. 后缀数据结构(21)

5. AC自动机模板(20)

## 评论排行榜

1. 并查集,以及可拆分并查集(4)

2. 各种树模板(splay,线段树,可持久化线段树...)(2)

3. 生命游戏和随机数之间某种不可言说的秘密(1)

```

144         y=x;
145         if(k==x->s[0]->tot+1) break;
146
147         if(k<x->s[0]->tot+1)
148             x=x->s[0];
149         else
150             {
151                 k=x->s[0]->tot+1;
152                 x=x->s[1];
153             }
154     }
155     splay(y);
156     return x;
157 }
158
159 int GetRank(node*x)
160 {
161     splay(x);
162     return x->s[0]->tot+1;
163 }
164
165 int GetRevRank(node*x)
166 {
167     splay(x);
168     return x->s[1]->tot+1;
169 }
170
171 node*Delete(node*x)
172 {
173     int k=GetRank(x);
174     node*L=FindRank(k-1);
175     node*R=FindRank(k+1);
176
177     splay(L);
178     splay(R,L);
179
180     if(L==nil && R==nil) root=nil;
181     else if(R==nil) L->s[1]=nil;
182     else R->s[0]=nil;
183
184     if(R!=nil) update(R);
185     if(L!=nil) update(L);
186
187     return x;
188 }
189
190 node*prefix(int v)
191 {
192     node*x=root;
193     node*y=x;
194     node*r=nil;
195     while(x!=nil)
196     {
197         y=x;
198         if(x->v<v) r=x;
199         if(v<=x->v) x=x->s[0];
200         else x=x->s[1];
201     }
202     splay(y);
203     return r;
204 }
205
206 node*suffix(int v)
207 {
208     node*x=root;
209     node*y=x;
210     node*r=nil;
211     while(x!=nil)
212     {
213         y=x;
214         if(x->v>v) r=x;
215         if(v<=x->v) x=x->s[0];
216         else x=x->s[1];
217     }
218     splay(y);
219     return r;
220 }
221
222
223

```

```

224
225 //=====
226 void output() { output(root); printf("%s\n", root==nil ? "empty tree!" : ""); }
227 void output(node*x)
228 {
229     if(x==nil) return ;
230     output(x->s[0]);
231     printf("%d ", x->v);
232     output(x->s[1]);
233 }
234
235 void test() { test(root); printf("%s\n", root==nil ? "empty tree!" : ""); }
236 void test(node*x)
237 {
238     if(x==nil) return ;
239     test(x->s[0]);
240     printf("%p [ v:%d f:%p L:%p R:%p tot:%d ] \n", x, x->v, x->f, x->s[0], x->s[1], x->tot);
241     test(x->s[1]);
242 }
243
244 };
245
246
247 int n;
248
249 int main()
250 {
251     scanf("%d", &n);
252     SplayTree st(n);
253
254     for(int i=0; i<n; i++)
255     {
256         int c;
257         scanf("%d", &c);
258         switch(c)
259         {
260             case 1: //Insert
261                 scanf("%d", &c);
262                 st.Insert(c);
263                 break;
264             case 2: //Delete
265                 scanf("%d", &c);
266                 st.Delete(st.Find(c));
267                 break;
268             case 3: //Rank
269                 scanf("%d", &c);
270                 printf("%d\n", st.GetRank(st.Find(c)));
271                 break;
272             case 4: //FindRank
273                 scanf("%d", &c);
274                 printf("%d\n", st.FindRank(c)->v);
275                 break;
276             case 5: //prefix
277                 scanf("%d", &c);
278                 printf("%d\n", st.prefix(c)->v);
279                 break;
280             case 6: //suffix
281                 scanf("%d", &c);
282                 printf("%d\n", st.suffix(c)->v);
283                 break;
284             case 7: //test
285                 st.test();
286                 break;
287             default: break;
288         }
289     }
290
291     return 0;
292 }

```

2015年2月19日更新:

我现在才发现我写的双旋一直都是错的!!!!

记住如果是折线就两次rot(x), 直线才是先y后x!  $\text{if}((x==y \rightarrow s[0]) \wedge (y==y \rightarrow f \rightarrow s[0])) \text{rot}(x); \text{else rot}(y);$

要点1. 别忘了有事没事splay一下保证复杂度.....

要点2.各种if的顺序别搞混了!有些if是不能合并的.

要点3.记住splay前的特判.如果要单独使用rotate就给rotate也加特判.

要点4.不要有事没事就更某些子树的位置!比如在delete的时候,提x作根,然后找到右子树最左边的节点后,合并左右两颗子树,这是会超时的!

代码超长...神犇勿拍....

还有优化的余地.....

另外我发现好多神奔居然除了insert和delete,都不splay....然后我写就狂T.....ORZ..

还是不怎么会计二叉查找树的常规....比如FindRank之类....多练就能熟练吧..

这个Splay在TYVJ的T1728上跑了大约400ms. 数据范围是操作数=10W. 鉴于以前写单旋splay能差不多这个时间过,数据应该是随机的....

也就是说...这个Splay的常数大概有20+....好恐怖...我线段树好歹也只有5到6.....

也就是说不能用这个模板做树套树之类了TAT

还是找个时间改进吧....

```
//=====
===
```

可持久化线段树

AC Vijos 1081 野生动物园

一道非常裸的区间k大

```

1  const int INF=(1<<30)-1;
2
3  struct node
4  {
5      int t;
6      node*l,*r;
7      node(){ t=0; l=r=NULL; }
8
9      void update()
10     { t=l->t+r->t; }
11 }pool[400000];
12 int nt;
13 node*newnode()
14 { return &pool[nt++]; }
15
16 node*nil;
17
18 node*root[100050];
19
20 void SegmentTreeInit(int size)
21 {
22     nil=newnode();
23     nil->l=nil->r=nil;
24     nil->t=0;
25     for(int i=0;i<=size;i++)
26         root[i]=nil;
27 }
28
29 int cp;
30 node*Change(node*x,node*y,int l,int r)
31 {
32     if(cp<l || r<cp) return y;
33     x=newnode();
34     if(l==r) { x->t=l+y->t; return x; }
35     int mid=(l+r)>>1;
36     x->l=Change(x->l,y->l,l,mid);
37     x->r=Change(x->r,y->r,mid+1,r);
38     x->update();
39     return x;
40 }
41 void Change(int i,int j,int pos)
42 { cp=pos; root[i]=Change(nil,root[j],0,INF); }
43
44 int Query(int ql,int qr,int k)
45 {
```

```

46     node*x=root[ql],*y=root[qr];
47     int l=0,r=INF;
48     while(l!=r)
49     {
50         int mid=(l+r)>>1;
51         if( k<= x->l->t - y->l->t )
52             r=mid,x=x->l,y=y->l;
53         else
54         {
55             k-=x->l->t-y->l->t;
56             l=mid+1,x=x->r,y=y->r;
57         }
58     }
59     return l;
60 }
61
62
63
64 int n;
65
66
67
68 int main()
69 {
70
71     int q;
72     scanf("%d",&n);
73     scanf("%d",&q);
74
75     SegmentTreeInit(n);
76
77
78     for(int i=0;i<n;i++)
79     {
80         int c;
81         scanf("%d",&c);
82         cp=c;
83         root[i+1]=Change(root[i+1],root[i],0,INF);
84     }
85
86
87     for(int i=0;i<q;i++)
88     {
89         int a,b,k;
90         scanf("%d%d%d",&a,&b,&k);
91         printf("%d\n",Query(b,a-1,k));
92     }
93
94     return 0;
95 }

```

要点1.使用nil节点可以省一点代码

要点2.千万注意空间开销.一般为nlogv级别,数组经常开上百万(懒得写离散化系列)

要点3.注意前缀和的写法.  $tree[R]-tree[L-1]$ . 这就要求 $root[0]=nil$ .

要点4.智商捉急表示普通查找操作总是写错...splay也一样.....思考...思考.....写之前一定要想好....

//=====

第一次写树套树

AC VJOS 1665

树状数组 套 动态开点的权值线段树

题目就是裸的带修改区间K大

写了一个多小时...调了大概....一个半小时?

树状数组怎么写都快忘记了.....

由于懒得离散化.....所以.....开了一个巨大的数组.....

VJ的内存限制不错....先把数组从400W改到800W...还是RE..怒而改到1.3kW...AC了.....

看了看空间消耗.....160M.....

这告诉我们千万不要忽视离散化的力量! 千万不要忽视常数空间!

但是我还是很懒=w=能不写离散化就不写离散化=w=

好吧.....

下面是代码.....

附带一大版的文件头以及调试信息2333

```
1 #include <cstdio>
2 #include <fstream>
3 #include <iostream>
4
5 #include <cstdlib>
6 #include <cstring>
7 #include <algorithm>
8 #include <cmath>
9
10 #include <queue>
11 #include <vector>
12 #include <map>
13 #include <set>
14 #include <stack>
15 #include <list>
16
17 typedef unsigned int uint;
18 typedef long long int ll;
19 typedef unsigned long long int ull;
20 typedef double db;
21
22 using namespace std;
23
24 inline int getint()
25 {
26     int res=0;
27     char c=getchar();
28     bool mi=false;
29     while(c<'0' || c>'9') mi=(c=='-'),c=getchar();
30     while('0'<=c && c<='9') res=res*10+c-'0',c=getchar();
31     return mi ? -res : res;
32 }
33 inline ll getll()
34 {
35     ll res=0;
36     char c=getchar();
37     bool mi=false;
38     while(c<'0' || c>'9') mi=(c=='-'),c=getchar();
39     while('0'<=c && c<='9') res=res*10+c-'0',c=getchar();
40     return mi ? -res : res;
41 }
42
43 db eps=1e-18;
44 inline bool feq(db a,db b)
45 { return fabs(a-b)<eps; }
46
47 inline int avg(const int a,const int b)
48 { return a+((b-a)>>1); }
49
50 //=====
51 //=====
52 //=====
53 //=====
54
55 const int INF=(1<<30)-1;
56
57 int n;
58 struct node*nil;
59 struct node
60 {
61     int v; //total
62     node*l,*r;
63     void update()
64     { v=l->v+r->v; }
65 }pool[13000000];
66 node*nt=pool;
67 node*newnode()
68 {
69     nt->l=nt->r=nil;
70     nt->v=0;
```

```

71     return nt++;
72 }
73
74 int cp,cv;
75
76 //Sub segment trees
77 struct SegmentTree
78 {
79     node*root;
80
81     node*Change(node*x,int l=0,int r=INF)
82     {
83         if(cp<l || r<cp) return x;
84         if(x==nil) x=newnode();
85         if(l==r) { x->v+=cv; return x; }
86         int mid=avg(l,r);
87         x->l=Change(x->l,l,mid);
88         x->r=Change(x->r,mid+1,r);
89         x->update();
90         return x;
91     }
92
93     void Set(int p,int v)
94     {
95         if(root<pool) root=nil;
96         cp=p;
97         cv=v;
98         root=Change(root);
99     }
100 };
101
102 //original segment tree
103 //performed as tree array
104
105 #define bt(x) (x&-x)
106
107 int a[1000000]; //this array must be stay here....
108 SegmentTree t[1000000];
109 void Change(int p,int s,int v) //location of point, value of point, delete or add in.
110 { for(int i=p;i<=n;i+=bt(i)) t[i].Set(s,v); }
111
112 node*c[1000];
113 int adt,ct;
114
115 int Query(int l,int r,int k) //find the element which is rank k.
116 {
117     adt=0;
118
119     for(int i=r;i>0;i-=bt(i))
120         c[adt++]=t[i].root;
121
122     ct=adt;
123     for(int i=l-1;i>0;i-=bt(i))
124         c[ct++]=t[i].root;
125
126     //we perform add when i<adt, and than dec when adt<=i<ct
127
128
129     l=0,r=INF;
130     while(l!=r)
131     {
132         //for(int i=0;i<ct;i++)
133         //cout<<c[i]<<' '; cout<<endl; cout<<l<<' '<<r<<endl; cout<<endl;
134
135         int mid=avg(l,r);
136         int clv=0; //current node's left node count.
137
138         for(int i=0;i<adt;i++)
139             clv+=c[i]->l->v;
140         for(int i=adt;i<ct;i++)
141             clv-=c[i]->l->v;
142
143         if(k<=clv) //the element we want find is on the left block
144         {
145             for(int i=0;i<ct;i++)
146                 c[i]=c[i]->l;
147             r=mid;
148         }
149         else
150         {

```



```

151         for(int i=0;i<ct;i++)
152             c[i]=c[i]->r;
153         k-=clv;
154         l=mid+1;
155     }
156 }
157
158 return l;
159 }
160
161 int q;
162
163 int main()
164 {
165     nil=newnode();
166     nil->l=nil->r=nil;
167     nil->v=0;
168
169     n=getint();
170     q=getint();
171     for(int i=0;i<n;i++)
172         Change(i+1,a[i+1]=getint(),1);
173
174     for(int i=0;i<q;i++)
175     {
176         char c[5];
177         scanf("%s",c);
178         if(c[0]=='C')
179         {
180             int i=getint();
181             int v=getint();
182             Change(i,a[i],-1);
183             Change(i,a[i]=v,1);
184         }
185         else
186         {
187             int i=getint();
188             int j=getint();
189             int k=getint();
190             printf("%d\n",Query(i,j,k));
191         }
192     }
193
194     return 0;
195 }
196
197

```

需要注意的地方:

1.树状数组什么的一级结构别写错了啊啊啊啊啊啊

2.既然是动态开点(即便离散化了也给我动态!绝对不要写静态的树套在另外的树里面!)....

那么,我们只需要记录每棵树的根节点就好了.其它节点在访问的时候会碰到.

3.嗯...(结构相同的)线段树是可加的.....所以不要再去写二分,直接在加起来的那棵树上隐式二分即可.详见代码.可以降低一个log的复杂度.

4.二分的界,以及二分后的操作(k=...)一定要考虑清楚.

2015年3月4日更新

<5.不知道为什么,用@iwtwio在某些地方的AC代码交到VJ,TLE了...就AC了我第一次提交没有RE的那两个范围较小的点.....  
<http://www.cnblogs.com/iwtwio/p/3929957.html>>


嘛,以上TLE的原因是在BZOJ上原题的数据范围是 $n \leq 1W$ ,VJOS上是 $N \leq 5W$ .....噢.....不知道当时看走眼了还是什么的囧

```
//=====
=====
```

不带其余任何标记的,带换根的LCT.

AC BZOJ2049 一道非常裸的LCT.

```

☐

1 #include <stdio>
2 #include <fstream>
3 #include <iostream>

```

```

4
5 #include <cstdlib>
6 #include <cstring>
7 #include <algorithm>
8 #include <cmath>
9
10 #include <queue>
11 #include <vector>
12 #include <map>
13 #include <set>
14 #include <stack>
15 #include <list>
16
17 typedef unsigned int uint;
18 typedef long long int ll;
19 typedef unsigned long long int ull;
20 typedef double db;
21
22 using namespace std;
23
24 inline int getint()
25 {
26     int res=0;
27     char c=getchar();
28     bool mi=false;
29     while(c<'0' || c>'9') mi=(c=='-'),c=getchar();
30     while('0'<=c && c<='9') res=res*10+c-'0',c=getchar();
31     return mi ? -res : res;
32 }
33 inline ll getll()
34 {
35     ll res=0;
36     char c=getchar();
37     bool mi=false;
38     while(c<'0' || c>'9') mi=(c=='-'),c=getchar();
39     while('0'<=c && c<='9') res=res*10+c-'0',c=getchar();
40     return mi ? -res : res;
41 }
42
43 db eps=1e-18;
44 inline bool feq(db a,db b)
45 { return fabs(a-b)<eps; }
46
47 template<typename Type>
48 inline Type avg(const Type a,const Type b)
49 { return a+((b-a)/2); }
50
51 //=====
52 //=====
53 //=====
54 //=====
55
56
57
58
59
60 struct node* nil;
61
62 struct node
63 {
64     bool rev;
65     node*s[2],*f;
66
67     bool root()
68     { return this!=f->s[0] && this!=f->s[1]; }
69
70     void pushtag()
71     {
72         if(rev)
73         {
74             s[0]->rev^=1;
75             s[1]->rev^=1;
76             swap(s[0],s[1]);
77             rev=false;
78         }
79     }
80
81 };
82
83

```

```

84 struct LinkCutTree
85 {
86     node*nt;
87     LinkCutTree(int size)
88     {
89         nt=new node[size];
90         for(int i=0;i<size;i++)
91         {
92             nt[i].s[0]=nt[i].s[1]=nt[i].f=nil;
93             nt[i].rev=false;
94         }
95     }
96
97     void cleartag(node*x)
98     { if(!x->root()) cleartag(x->f); x->pushtag(); }
99
100     node*operator[](int k)
101     { return nt+k; }
102
103     void rot(node*x)
104     {
105         if(x->root()) return ;
106         node*y=x->f;
107         bool k=(x==y->s[0]);
108
109         y->s[!k]=x->s[k];
110         if(x->s[k]!=nil) x->s[k]->f=y;
111
112         x->f=y->f;
113         if(y==y->f->s[0]) y->f->s[0]=x;
114         else if(y==y->f->s[1]) y->f->s[1]=x;
115
116         y->f=x;
117         x->s[k]=y;
118     }
119
120     void splay(node*x)
121     {
122         cleartag(x);
123         while(!x->root())
124         {
125             node*y=x->f;
126             if(!y->root())
127             {
128                 if((x==y->s[0])^(y==y->f->s[0]))
129                     rot(y); else rot(x);
130             }
131             rot(x);
132         }
133     }
134
135     node*access(node*x)
136     {
137         node*y=nil;
138         node*t=x;
139         while(t!=nil)
140         {
141             splay(t);
142             t->s[0]=y;
143             y=t;
144             t=t->f;
145         }
146         splay(x);
147         return x;
148     }
149
150     node*FindRoot(node*x)
151     {
152         access(x);
153         while(x->s[1]!=nil) x=x->s[1];
154         return x;
155     }
156
157     node*SetRoot(node*x)
158     {
159         access(x)->rev^=1;
160         return x;
161     }
162
163     void Link(node*x,node*y)

```

```

164     {
165         SetRoot(x)->f=y;
166     }
167
168     void Cut(node*x,node*y)
169     {
170         SetRoot(x);
171         access(y);
172         y->s[1]->f=nil;
173         y->s[1]=nil;
174     }
175
176     void output(int i)
177     { cout<<i<<' '<<&nt[i]<<' '<<nt[i].s[0]<<' '<<nt[i].s[1]<<' '<<nt[i].f<<endl; }
178 };
179
180 int n,m;
181
182 int main()
183 {
184     nil=new node;
185     nil->s[0]=nil->s[1]=nil->f=nil;
186
187     n=getint();
188     m=getint();
189
190     LinkCutTree t(n);
191
192     for(int i=0;i<m;i++)
193     {
194         char c[20];
195         scanf("%s",c);
196         if(c[0]=='C') //Link
197         {
198             t.Link(t[getint()-1],t[getint()-1]);
199         }
200         else if(c[0]=='D') //Cut
201         {
202             t.Cut(t[getint()-1],t[getint()-1]);
203         }
204         else if(c[0]=='Q') //Query
205         {
206             if(t.FindRoot(t[getint()-1])==t.FindRoot(t[getint()-1])) printf("Yes\n");
207             else printf("No\n");
208         }
209     }
210
211     return 0;
212 }

```

很奇怪,Link,Cut,FindRoot和SetRoot这些函数换一种写法就各种TLE/RE,还有cleartag()也是一样,不知道为什么.....

还是LCT.

AC VJOS1190 LCT-MST

[View Code](#)

调半天发现update写错了.....TAT

要点:

- 1.所有的操作函数都不要写错啊!虽然很短.....
- 2.access操作返回的是上一次access时的路径与本次access节点在有根树的LCA.  
 所以用expend函数来作为返回x的access操作.
- 3.找到根后别忘记splay根上去,毕竟访问了根节点就要保证其复杂度.
- 4.Cut操作一定要换根,expend(y)之后,把y与y的右儿子断掉.尤其是x,y不一定相邻的时候.
- 5.update()与pushtag()别写错!
- 6.在splay前把要splay的节点的父节点的tag清空.因为splay操作是以结构作为基础的.
- 7.不要在写LCT的时候纠结struct node到底该写哪怎么写的问题!想到怎么写就怎么写,并且LCT函数全部用node\*形式.

NOI2014 Day1 T2 魔法森林 用LCT维护MST.上一题就是为这个做准备的.

☐



```
1 #include <cstdio>
2 #include <fstream>
3 #include <iostream>
4
5 #include <cstdlib>
6 #include <cstring>
7 #include <algorithm>
8 #include <cmath>
9
10 #include <queue>
11 #include <vector>
12 #include <map>
13 #include <set>
14 #include <stack>
15 #include <list>
16
17 typedef unsigned int uint;
18 typedef long long int ll;
19 typedef unsigned long long int ull;
20 typedef double db;
21
22 using namespace std;
23
24 inline int getint()
25 {
26     int res=0;
27     char c=getchar();
28     bool mi=false;
29     while(c<'0' || c>'9') mi=(c=='-'),c=getchar();
30     while('0'<=c && c<='9') res=res*10+c-'0',c=getchar();
31     return mi ? -res : res;
32 }
33 inline ll getll()
34 {
35     ll res=0;
36     char c=getchar();
37     bool mi=false;
38     while(c<'0' || c>'9') mi=(c=='-'),c=getchar();
39     while('0'<=c && c<='9') res=res*10+c-'0',c=getchar();
40     return mi ? -res : res;
41 }
42
43 db eps=1e-18;
44 inline bool feq(db a,db b)
45 { return fabs(a-b)<eps; }
46
47 template<typename Type>
48 inline Type avg(const Type a,const Type b)
49 { return a+((b-a)/2); }
50
51 //=====
52 //=====
53 //=====
54 //=====
55
56
57 const int INF=(1<<30)-1;
58
59
60 struct node* nil;
61
62 struct node
63 {
64     node*mxp;
65     int code;
66     int v;
67     bool rev;
68     node*s[2],*f;
69
70     bool root()
71     { return this!=f->s[0] && this!=f->s[1]; }
72
73     void pushtag()
74     {
75         if(rev)
76         {
77             s[0]->rev^=1;
78             s[1]->rev^=1;
79             swap(s[0],s[1]);
```

```

80         rev=false;
81     }
82 }
83
84 void update()
85 {
86     mxp=this;
87     int lv=s[0]->mxp->v;
88     int rv=s[1]->mxp->v;
89     if(lv>mxp->v) mxp=s[0]->mxp;
90     if(rv>mxp->v) mxp=s[1]->mxp;
91 }
92 };
93
94
95 struct LinkCutTree
96 {
97     node*nt;
98     LinkCutTree(int size)
99     {
100         nt=new node[size];
101         for(int i=0;i<size;i++)
102         {
103             nt[i].s[0]=nt[i].s[1]=nt[i].f=nil;
104             nt[i].rev=false;
105             nt[i].mxp=&nt[i];
106             nt[i].v=-INF;
107             nt[i].code=i;
108         }
109     }
110
111     void cleartag(node*x)
112     { if(!x->root()) cleartag(x->f); x->pushtag(); }
113
114     node*operator[](int k)
115     { return nt+k; }
116
117     void rot(node*x)
118     {
119         if(x->root()) return ;
120         node*y=x->f;
121         bool k=(x==y->s[0]);
122
123         y->s[!k]=x->s[k];
124         if(x->s[k]!=nil) x->s[k]->f=y;
125
126         x->f=y->f;
127         if(y==y->f->s[0]) y->f->s[0]=x;
128         else if(y==y->f->s[1]) y->f->s[1]=x;
129
130         y->f=x;
131         x->s[k]=y;
132
133         y->update();
134     }
135
136     void splay(node*x)
137     {
138         cleartag(x);
139         while(!x->root())
140         {
141             node*y=x->f;
142             if(!y->root())
143             {
144                 if((x==y->s[0])^(y==y->f->s[0]))
145                     rot(y); else rot(x);
146             }
147             rot(x);
148         }
149         x->update();
150     }
151
152     node*access(node*x)
153     {
154         node*y=nil;
155         while(x!=nil)
156         {
157             splay(x);
158             x->s[0]=y;
159             y=x;

```

```

160         x=x->f;
161     }
162     return x;
163 }
164
165 node*expend(node*x)
166 {
167     access(x);
168     splay(x);
169     return x;
170 }
171
172 node*FindRoot (node*x)
173 {
174     expend(x);
175     while(x->s[1]!=nil) x=x->s[1];
176     splay(x);
177     return x;
178 }
179
180 node*SetRoot (node*x)
181 {
182     expend(x)->rev^=1;
183     return x;
184 }
185
186 void Link(node*x,node*y)
187 {
188     SetRoot(x)->f=y;
189 }
190
191 void Cut (node*x,node*y)
192 {
193     SetRoot(x);
194     expend(y);
195     y->s[1]->f=nil;
196     y->s[1]=nil;
197 }
198
199 node* GetMax(node*x,node*y)
200 {
201     SetRoot(x);
202     return expend(y)->mxp;
203 }
204
205
206 void output(int i)
207 { printf("%p[ id:%d v:%d L:%p R:%p f:%p rev:%d max:%d
] \n", &nt[i], nt[i].code, nt[i].v, nt[i].s[0], nt[i].s[1], nt[i].f, nt[i].rev, nt[i].mxp->v); }
208 };
209
210 int n,m;
211 int EL[100050];
212 int ER[100050];
213 int Ea[100050];
214 int Eb[100050];
215 int p[100050];
216 bool cmp(int a,int b)
217 { return Ea[a]<Ea[b]; }
218
219 int res=INF;
220
221 int main()
222 {
223     nil=new node;
224     nil->mxp=nil->s[0]=nil->s[1]=nil->f=nil;
225     nil->v=-INF;
226     nil->code=-1;
227
228     n=getint();
229     m=getint();
230
231     LinkCutTree T(n+m+2);
232
233     int st=0,ed=n-1;
234
235     for(int i=0;i<m;i++)
236     {
237         EL[i]=getint()-1;
238         ER[i]=getint()-1;

```

```

239     Ea[i]=getint();
240     Eb[i]=getint();
241 }
242
243 for(int i=0;i<m;i++) p[i]=i;
244 stable_sort(p,p+m,cmp);
245
246 int curbase=0;
247
248 for(int i=0;i<m;i++)
249 {
250     int e=p[i]; //the id of current edge.
251     int cur=n+i; //the id of current edge node.
252     node*L=T[EL[e]];
253     node*R=T[ER[e]];
254     curbase=Ea[e]; //current value count.
255     T[cur]->v=Eb[e]; //assign value of current edge node.
256
257     if(L==R) continue;
258
259     if(T.FindRoot(L)==T.FindRoot(R))
260     {
261         node*X=T.GetMax(L,R);
262         int v=X->v;
263
264         if(v>Eb[e]) //replace point mxp
265         {
266             T.Cut(T[EL[X->code-n]],X);
267             T.Cut(T[ER[X->code-n]],X);
268
269             T.Link(L,T[cur]);
270             T.Link(R,T[cur]);
271         }
272     }
273     else
274     {
275         T.Link(L,T[cur]);
276         T.Link(R,T[cur]);
277     }
278
279     if(T.FindRoot(T[n-1])==T.FindRoot(T[0]))
280     {
281         T.SetRoot(T[n-1]);
282         T.expend(T[0]);
283         res=min(res,curbase+T[0]->mxp->v);
284     }
285 }
286
287 if(res==INF) printf("-1\n");
288 else printf("%d\n",res);
289
290 return 0;
291 }

```

分类: [OI](#) [模板](#)

绿色通道: [好文要顶](#) [关注我](#) [收藏该文](#) [与我联系](#) 

 [DragoonKiller](#)  
[关注 - 1](#)  
[粉丝 - 2](#)

[+加关注](#)

0

0

(请您对文章做出评价)

« 上一篇: [大整数模板](#)

» 下一篇: [上下界网络流初探](#)

posted @ 2014-12-27 20:07 DragoonKiller 阅读(44) 评论(2) 编辑 收藏

#1楼 2015-03-04 12:35 iwtwiiioi

“妈呀....你交我的代码不改数据范围tle了很正常啊= =

支持(0) 反对(0)



“@iwtwioi  
噢我还以为数据范围是一样的

支持(0) 反对(0)

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。

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【活动】ACE一键建站服务免费公测中

融云，免费为你的App加入IM功能——让你的App“聊”起来！！

最新IT新闻：

- 吃掉每一盘在你眼前的东西
  - AnyPresence为移动与物联网开发者发布了一个新的API平台
  - 为什么说招到合适的人比融到钱更加重要？
  - 在线教育先驱Lynda.com创始人：二十年磨一剑
  - 过来人告诉你众筹产品后期融资最佳时机
- » 更多新闻...

最新知识库文章：

- 驱动方法不能改变任何事情
  - 推行TDD的思考
  - 首席工程师揭秘：LinkedIn大数据后台是如何运作的
  - 码农的思维训练：超越专家
  - 如何摆脱令人头疼的架构依赖？
- » 更多知识库文章...