

Rabbit_hair

If you want something you have never had,you must be willing to do something you have never done.

博客园 首页 新随笔 联系 订阅 管理 随笔- 54 文章- 0 评论- 3

粉丝: 11
关注: 3
+加关注

<

2013年5月

>

日	一	二	三	四	五	六
28	29	30	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	31	1
2	3	4	5	6	7	8

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

随笔档案

2014年2月 (2)

2013年12月 (1)

2013年11月 (4)

2013年10月 (3)

2013年9月 (1)

2013年8月 (7)

2013年7月 (7)

2013年6月 (4)

2013年5月 (14)

2013年4月 (5)

2013年1月 (2)

2012年9月 (1)

2012年8月 (2)

2012年5月 (1)

阅读排行榜

1. 主席树：动态 Kth(1902)

2. 数位DP(1661)

3. 莫比乌斯反演(853)

4. 主席树 静态区间第k大(678)

5. 后缀数组题目(541)

评论排行榜

1. 2013多校第三场(3)

推荐排行榜

1. 概率DP(1)

2. 主席树 静态区间第k大(1)

1 /*zoj2112 http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=2112

2 动态 kth

3 每一棵线段树是维护每一个序列前缀的值在任意区间的个数，

4 如果还是按照静态的来做的话，那么每一次修改都要遍历O(n)棵树，

5 时间就是O(2*M*nlogn)->TLE

6 考虑到前缀和，我们通过树状数组来优化，即树状数组套主席树，

7 每个节点都对应一棵主席树，那么修改操作就只要修改logn棵树，

8 o(nlognlogn+Mlognlogn)时间是可以的，

9 但是直接建树要nlogn*logn(10^7)会MLE

10 我们发现对于静态的建树我们只要nlogn个节点就可以了，

11 而且对于修改操作，只是修改M次，每次改变俩个值（减去原先的，加上现在的）

12 也就是说如果把所有初值都插入到树状数组里是不值得的，

13 所以我们分两部分来做，所有初值按照静态来建，内存O(nlogn)，

14 而修改部分保存在树状数组中，每次修改logn棵树，每次插入增加logn个节点

15 O(M*logn*logn+nlogn)

16

17

18 */

19 #include<cstdio>

20 #include<cstring>

21 #include<cstdlib>

22 #include<iostream>

23 #include<algorithm>

24 #include<vector>

25 #include<cmath>

26 #define ls(i) T[i].ls

27 #define rs(i) T[i].rs

28 #define w(i) T[i].w

29 #define Find(i) (lower_bound(LX.begin(),LX.begin()+n1,i)-LX.begin())+1

30

31 using namespace std;

32 const int N=60000+10;

33 struct node{

34 int ls,rs,w;

35 node() {ls=rs=w=0;}

36 }T[2000000];

37 struct ope{

38 int i,l,r,k;

39 }op[11000];

40 vector<int> LX,Q1,Q2;

41 int n,n1,m,cnt;

42 int a[61000],root[61000*2];

43 inline int lowbit(int x){

44 return x&-x;

45 }

46 void build(int &i,int l,int r,int x){

47 T[++cnt]=T[i]; i=cnt;

48 w(i)++;

49 if (l==r) return;

50 int m=(l+r)>>1;

51 if (x<=m) build(ls(i),l,m,x);

52 else build(rs(i),m+1,r,x);

53 }

54 void ins(int &i,int l,int r,int x,int v){

55 if (i==0){ T[++cnt]=T[i]; i=cnt; }

56 w(i)+=v;

57 if (l==r) return;

58 int m=(l+r)>>1;

59 if (x<=m) ins(ls(i),l,m,x,v);

60 else ins(rs(i),m+1,r,x,v);

61 }

62 void my_ins(int pos,int x,int v){

63 int t=Find(x);

64 for (int i=pos;i<=n;i+=lowbit(i)){

65 ins(root[i],1,n1,t,v);

66 }

67 }

68 int Qy(vector<int> Q1,vector<int> Q2,int l,int r,int k){

69 if (l==r) return l;

70 int c=0;

71 int m=(l+r)>>1;

www.cnblogs.com/R1emon/archive/2013/05/24/3096264.html

1/3

```

72 for (int i=0;i<Q1.size();i++) c-=w(ls(Q1[i]));
73 for (int i=0;i<Q2.size();i++) c+=w(ls(Q2[i]));
74 for (int i=0;i<Q1.size();i++) Q1[i]=(c>=k?ls(Q1[i]):rs(Q1[i]));
75 for (int i=0;i<Q2.size();i++) Q2[i]=(c>=k?ls(Q2[i]):rs(Q2[i]));
76
77 if (c>=k) return Qy(Q1,Q2,l,m,k);
78 else return Qy(Q1,Q2,m+1,r,k-c);
79 }
80 void query(int l,int r,int k){
81     Q1.clear();Q2.clear();
82     Q1.push_back(root[l!=1?l-1+n:0]);
83     Q2.push_back(root[r+n]);
84     for (int i=l-1;i>0;i-=lowbit(i)) Q1.push_back(root[i]);
85     for (int i=r;i>0;i-=lowbit(i)) Q2.push_back(root[i]);
86
87     int t=Qy(Q1,Q2,l,nl,k);
88     printf("%d\n",LX[t-1]);
89 }
90 void work(){
91     cnt=0;
92     //for (int i=0;i<nl;i++) cout<<list[i]<<" ";cout<<endl;
93     memset(root,0,sizeof(root));
94     for (int i=1;i<=n;i++){
95         root[i+n]=root[i-1+n];
96         int t=Find(a[i]);
97         build(root[i+n],1,nl,t);
98     }
99     for (int i=0;i<m;i++){
100         if (op[i].i==0){
101             query(op[i].l,op[i].r,op[i].k);
102             // cout<<"*** "<<i<<endl;
103         }else{
104             my_ins(op[i].l,a[op[i].l],-1);
105             my_ins(op[i].l,op[i].r,1);
106             a[op[i].l]=op[i].r;
107         }
108     }
109
110 }
111 int main(){
112     int Cas;scanf("%d",&Cas);
113     while (Cas--){
114         scanf("%d%d",&n,&m);
115         LX.clear();
116         for (int i=1;i<=n;i++){
117             scanf("%d",&a[i]);LX.push_back(a[i]);
118         }
119         char s[10];
120         for (int i=0;i<m;i++){
121             scanf("%s",s);
122             if (s[0]=='Q'){
123                 op[i].i=0;
124                 scanf("%d%d%d",&op[i].l,&op[i].r,&op[i].k);
125             }else{
126                 op[i].i=1;
127                 scanf("%d%d",&op[i].l,&op[i].r);
128                 LX.push_back(op[i].r);
129             }
130         }
131         sort(LX.begin(),LX.end());
132         nl=unique(LX.begin(),LX.end())-LX.begin();
133         work();
134     }
135
136     return 0;
137 }
138 }

```



绿色通道:

好文要顶

关注我

收藏该文

与我联系



Rabbit hair

关注 - 3

粉丝 - 11

+加关注

0

推荐

0

反对

(请您对文章做出评价)

« 上一篇: [主席树 静态区间第k大](#)

» 下一篇: [zoi 3540](#)

posted @ 2013-05-24 10:00 [Rabbit_hair](#) 阅读(1903) 评论(0) [编辑](#) [收藏](#)

[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
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：
- [Windows 10新泄漏：MSN应用、纸牌预览版、其他界面改动](#)
 - [开发者自白：我是如何不花一分钱收获230万应用下载量的！](#)
 - [Windows 10 Build 10056可通过注册表开启黑色主题](#)
 - [第一人称射击游戏之父John Carmack](#)
 - [Apache web server: 20年发展历程](#)
- » [更多新闻...](#)

- 最新知识库文章：
- [什么是对象，为什么要面向对象，怎么才能面向对象？](#)
 - [细数移动IM开发中的那些坑](#)
 - [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](#)
- » [更多知识库文章...](#)