

原创

静态主席树（可持久化线段树+离散化）

2019-07-10 20:49:45   \_Y-\_-Y-    阅读数 58

编辑

版权声明：本文为博主原创文章，遵循 [CC 4.0 BY-SA](#) 版权协议，转载请附上原文出处链接和本声明。  
本文链接：[https://blog.csdn.net/weixin\\_44410512/article/details/95368744](https://blog.csdn.net/weixin_44410512/article/details/95368744)

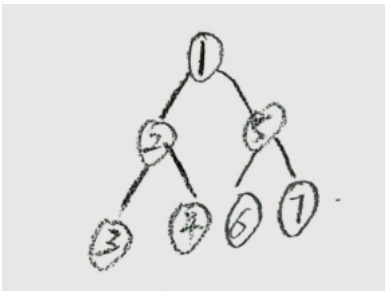
### 可持久化线段树

首先建一颗树，因为要可持久化，所以下标之间就没有关系了，所以要用结构体存左右儿子。

code

```
1 | int build_tree(int l,int r){
2 |     int pos=++tot;
3 |     tree[pos].v=0;
4 |     if(l==r){
5 |         return pos;
6 |     }
7 |     int mid=l+(r-1)/2;
8 |     tree[pos].l=build_tree(l,mid);
9 |     tree[pos].r=build_tree(mid+1,r);
10 |    return pos;
11 | }
```

建好的树类似于这样：

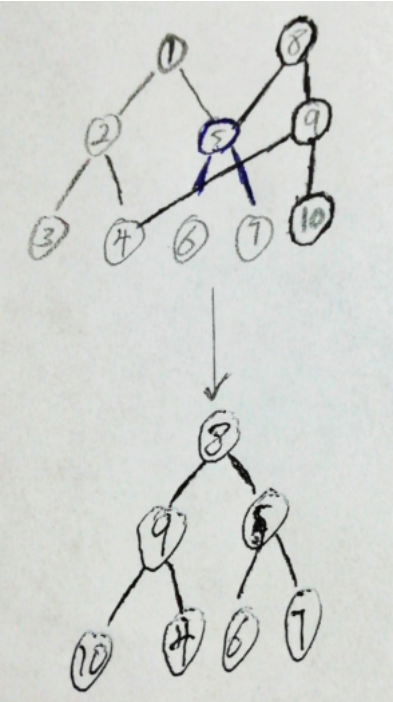


树建好之后要是修改一个数值并且要保留之前那个数怎么办？  
当然不能重建一棵树了，这样代价太高了。  
可以发现，与那个数相关的数就是它的父亲，父亲的父亲，一直到根  
那么我们就只要在增加那一条就好了，让新增的那一条与本来相连但没有影响的相连

code

```
1 | int update(int now,int tar,int l,int r){
2 |     int pos=++tot;
3 |     if(l==r){
4 |         tree[pos].v=tree[now].v+1;
5 |         return pos;
6 |     }
7 |     int mid=l+(r-1)/2;
8 |     tree[pos].l=tree[now].l;
9 |     tree[pos].r=tree[now].r;
10 |    if(tar<=mid) tree[pos].l=update(tree[now].l,tar,l,mid);
11 |    else tree[pos].r=update(tree[now].r,tar,mid+1,r);
12 |    tree[pos].v=tree[tree[pos].l].v+tree[tree[pos].r].v;
13 |    return pos;
14 | }
```

如图：



这样，就可以不覆盖原来的点，且有线段树的功能

离散化

离散化就是把无限空间中有限的个体映射到有限的空间中去，以此提高算法的时空效率。

比如数据过大时，建立线段树无法开辟那么多单元，此时就要用到离散化了。

具体步骤如下：

- 1. `sort(b+1,b+1+n)`；排序。
- 2. `int res=unique(b+1,b+n+1)-(b+1)`；去重，返回最后那个完成去重的点往后一个位置，所以减一个b+1。
- 3. `a[i]=lower_bound(b+1,b+1+res,a[i])-b`；返回b[1]到b[res]中a[i]的地址。

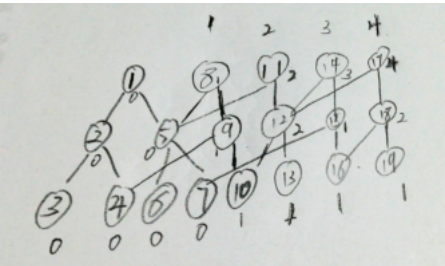
code

```
1  for(int i=1;i<=n;i++){
2      scanf("%d",&a[i]);
3      b[i]=a[i];
4  }
5  sort(b+1,b+1+n);
6  int res=unique(b+1,b+n+1)-(b+1);
7  for(int i=1;i<=n;i++){
8      a[i]=lower_bound(b+1,b+1+res,a[i])-b;
9  }
```

静态主席树

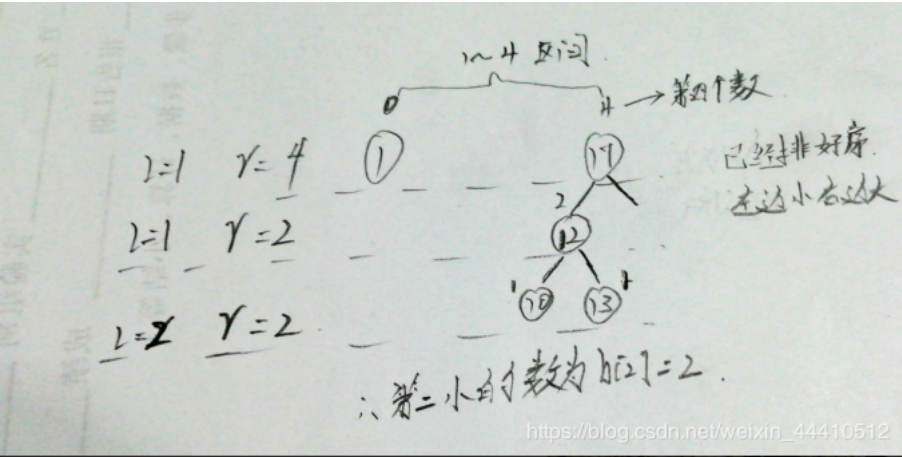
建立一个 $a[4] = \{1, 2, 3, 4\}$ 的主席树

如图：



先离散化，

通过可持久化线段树可以按照a的顺序从主树上建立新树  
以tree[8] 为根的树是第1个数字建的树  
以tree[11]为根的树是第2个数字建的树  
.  
.  
.  
tree[i].v存的是这个数出现的次数  
比如我要寻找1~4区间内第二小的数  
如图:



code

```
1 int query(int now,int last,int k,int l,int r){
2     if(l==r){
3         return l;
4     }
5     int mid=l+(r-1)/2;
6     int cnt=tree[tree[now].l].v-tree[tree[last].l].v;
7     if(k<=cnt) return query(tree[now].l,tree[last].l,k,l,mid);
8     else return query(tree[now].r,tree[last].r,k-cnt,mid+1,r);
9 }
```

例题

<https://cn.vjudge.net/problem/POJ-2104>

AC code

```
1 #include <cstdio>
2 #include <cstring>
3 #include <algorithm>
4 using namespace std;
5 #define maxn 100010
6 int n,m,tot;
7 int a[maxn],b[maxn],root[maxn];
8 struct node{
9     int l,r,v;
10 }tree[maxn<<5];
11 int build_tree(int l,int r){
12     int pos=++tot;
13     tree[pos].v=0;
14     if(l==r){
15         return pos;
16     }
17     int mid=l+(r-1)/2;
18     tree[pos].l=build_tree(l,mid);
19     tree[pos].r=build_tree(mid+1,r);
20     return pos;
21 }
22 int update(int now,int tar,int l,int r){
23     int pos=++tot;
```

```
24     if(l==r){
25         tree[pos].v=tree[now].v+1;
26         return pos;
27     }
28     int mid=l+(r-1)/2;
29     tree[pos].l=tree[now].l;
30     tree[pos].r=tree[now].r;
31     if(tar<=mid) tree[pos].l=update(tree[now].l,tar,l,mid);
32     else tree[pos].r=update(tree[now].r,tar,mid+1,r);
33     tree[pos].v=tree[tree[pos].l].v+tree[tree[pos].r].v;
34     return pos;
35 }
36 int query(int now,int last,int k,int l,int r){
37     if(l==r){
38         return l;
39     }
40     int mid=l+(r-1)/2;
41     int cnt=tree[tree[now].l].v-tree[tree[last].l].v;
42     if(k<=cnt) return query(tree[now].l,tree[last].l,k,l,mid);
43     else return query(tree[now].r,tree[last].r,k-cnt,mid+1,r);
44 }
45 int main(){
46     int n,m;
47     scanf("%d %d", &n, &m);
48     for(int i=1;i<=n;i++){
49         scanf("%d", &a[i]);
50         b[i]=a[i];
51     }
52     sort(b+1,b+1+n);
53     int res=unique(b+1,b+1+n)-(b+1);
54     for(int i=1;i<=n;i++){
55         a[i]=lower_bound(b+1,b+1+res,a[i])-b;
56     }
57     root[0]=build_tree(1,res);
58     for(int i=1;i<=n;i++){
59         root[i]=update(root[i-1],a[i],1,res);
60     }
61     int l,r,k;
62     while(m--){
63         scanf("%d %d %d", &l, &r, &k);
64         printf("%d\n", b[query(root[r],root[l-1],k,1,res)]);
65     }
66     return 0;
67 }
```

有 0 个人打赏

文章最后发布于: 2019-07-10 20:49:45

©2019 CSDN 皮肤主题: 大白 设计师: CSDN官方博客