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

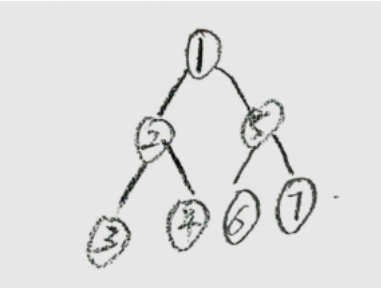
可持久化线段树

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

```
int build_tree(int l,int r){
  int pos=++tot;
  tree[pos].v=0;
  if(l==r){
    return pos;
  }
  int mid=l+(r-l)/2;
  tree[pos].l=build_tree(l,mid);
  tree[pos].r=build_tree(mid+1,r);
  return pos;
}
```

建好的树类似于这样：

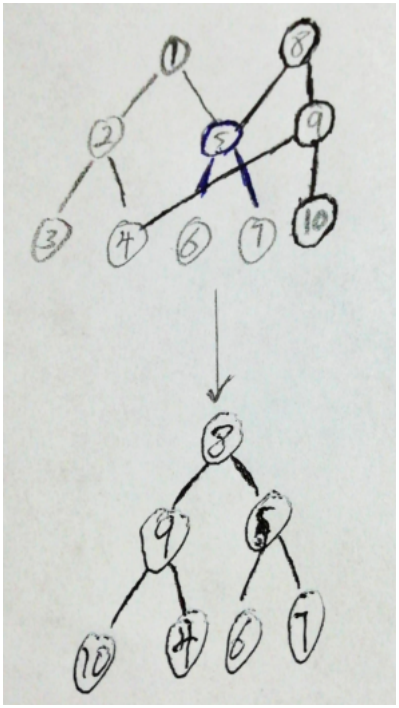


树建好之后要是修改一个数值并且要保留之前那个数怎么办?
当然不能重建一棵树了，这样代价太高了。
可以发现，与那个数相关的数就是它的父亲，父亲的父亲，一直到根
那么我们就只要在增加那一条就好了，让新增的那一条与本来相连但没有影响的相连

code

```
int update(int now,int tar,int l,int r){
  int pos=++tot;
  if(l==r){
    tree[pos].v=tree[now].v+1;
    return pos;
  }
  int mid=l+(r-l)/2;
  tree[pos].l=tree[now].l;
  tree[pos].r=tree[now].r;
  if(tar<=mid) tree[pos].l=update(tree[now].l,tar,l,mid);
  else tree[pos].r=update(tree[now].r,tar,mid+1,r);
  tree[pos].v=tree[tree[pos].l].v+tree[tree[pos].r].v;
  return pos;
}
```

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：

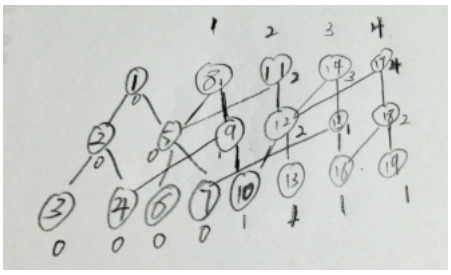
1. `sort(b+1,b+1+n);` 排序。
2. `int res=unique(b+1,b+n+1)-(b+1);` 去重，返回最后那个完成去重的点往后一个位置，所以减一个b+1。
3. `a[i]=lower_bound(b+1,b+1+res,a[i])-b;` 返回$b[1]$到$b[res]$中$a[i]$的地址。

code

```
for(int i=1;i<=n;i++){
    scanf("%d", &a[i]);
    b[i]=a[i];
  }
  sort(b+1,b+1+n);
  int res=unique(b+1,b+n+1)-(b+1);
  for(int i=1;i<=n;i++){
    a[i]=lower_bound(b+1,b+1+res,a[i])-b;
  }
```

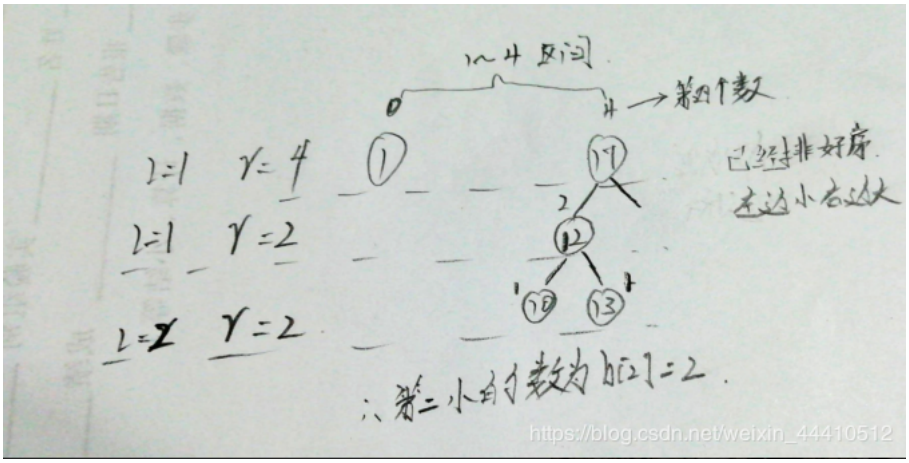
静态主席树

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

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

例题

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

AC code

```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define maxn 100010
int n,m,tot;
int a[maxn],b[maxn],root[maxn];
struct node{
  int l,r,v;
}tree[maxn<<5];
int build_tree(int l,int r){
  int pos=++tot;
  tree[pos].v=0;
  if(l==r){
    return pos;
  }
  int mid=l+(r-l)/2;
  tree[pos].l=build_tree(l,mid);
  tree[pos].r=build_tree(mid+1,r);
  return pos;
}
int update(int now,int tar,int l,int r){
  int pos=++tot;
```

```
  if(l==r){
    tree[pos].v=tree[now].v+1;
    return pos;
  }
  int mid=l+(r-l)/2;
  tree[pos].l=tree[now].l;
  tree[pos].r=tree[now].r;
  if(tar<=mid) tree[pos].l=update(tree[now].l,tar,l,mid);
  else tree[pos].r=update(tree[now].r,tar,mid+1,r);
  tree[pos].v=tree[tree[pos].l].v+tree[tree[pos].r].v;
  return pos;
}
int query(int now,int last,int k,int l,int r){
  if(l==r){
    return l;
  }
  int mid=l+(r-l)/2;
  int cnt=tree[tree[now].l].v-tree[tree[last].l].v;
  if(k<=cnt) return query(tree[now].l,tree[last].l,k,l,mid);
  else return query(tree[now].r,tree[last].r,k-cnt,mid+1,r);
}
int main(){
  int n,m;
  scanf("%d %d", &n, &m);
  for(int i=1;i<=n;i++){
    scanf("%d", &a[i]);
    b[i]=a[i];
  }
  sort(b+1,b+1+n);
  int res=unique(b+1,b+n+1)-(b+1);
  for(int i=1;i<=n;i++){
    a[i]=lower_bound(b+1,b+1+res,a[i])-b;
  }
  root[0]=build_tree(1,res);
  for(int i=1;i<=n;i++){
    root[i]=update(root[i-1],a[i],1,res);
  }
  int l,r,k;
  while(m--){
    scanf("%d %d %d", &l, &r, &k);
    printf("%d\n", b[query(root[r],root[l-1],k,1,res)]);
  }
  return 0;
}
```

有 0 个人打赏

文章最后发布于: 2019-07-10 20:49:45