题解 P3380 【【模板】二逼平衡树（树套树）】

2018-08-02 00:16:18

thumb\_up *47*

今天给大家介绍一种新的做法

这种做法是**目前所有本题在线做法的题解中，理论复杂度最优的做法**

这种做法是**在线的，并且时间复杂度为O(n(logn)^2),空间复杂度为o（nlogn）**

常规一些的思路是，外层树记录位置，内层一颗平衡树记录数值，但是这样做，对于查询排名为k的数的值不好操作，需要先二分答案，再判定，时间复杂度高达三个log

还有一种思路是，树状数组套主席树，这种方法虽然能做到两个log查询排名为k的数的值，但是空间复杂度高达两个log,及其容易MLE

常规思路的瓶颈就在于查询这个排名为k的值，我们转换下思路，改为 **外层树记录数值，内层树记录位置**，用一个动态开点的权值线段树套平衡树来解决

对于操作1，我们查询值在[0,k-1]中，有多少位置在给定区间内，再加一就是排名

对于操作2，我们直接在线段树上二分，并查询该节点有多少数在给定位置区间内判定即可

对于操作3，我们先把原来的那些节点全部删除，然后再插入新的节点

对于操作4，我们先查询k的排名，如果是1说明没有前驱，否则输出排名为k的排名-1的数

对于操作5，我们先查询k+1的排名，如果k+1的排名大于r-l+1，说明没有后继，否则输出排名为k+1的排名的数

以下是代码，

#include<stdio.h>

#define fa 4800000

#define fa2 1600000

#define LL long long

typedef struct Splay

{

typedef struct Ts

{

int size,x;

Ts\*f,\*l,\*r;

}Ts;

static Ts mem[fa],\*stk[fa],\*\*top;

Ts\*root;

inline Ts\*New(int x,Ts\*f)

{

Ts\*p=\*--top;

p->size=1;

p->x=x;

p->f=f;

p->l=p->r=0;

return p;

}

inline void Del(Ts\*&t)

{

\*top++=t;

t=0;

}

void init()

{

root=New(999999,0);

root->l=New(-1,root);

root->size=2;

}

void del(Ts\*t)

{

if(t)

{

del(t->l);

del(t->r);

Del(t);

}

}

inline void updata(Ts\*t)

{

t->size=1;

if(t->l)t->size+=t->l->size;

if(t->r)t->size+=t->r->size;

}

inline void zuoxuan(Ts\*t)

{

Ts\*p=t->f;

p->r=t->l;

if(t->l)t->l->f=p;

t->l=p;

if(p->f)

{

if(p->f->l==p)p->f->l=t;

else p->f->r=t;

}

t->f=p->f;

p->f=t;

updata(p);

updata(t);

}

inline void youxuan(Ts\*t)

{

Ts\*p=t->f;

p->l=t->r;

if(t->r)t->r->f=p;

t->r=p;

if(p->f)

{

if(p->f->l==p)p->f->l=t;

else p->f->r=t;

}

t->f=p->f;

p->f=t;

updata(p);

updata(t);

}

void danxuan(Ts\*t)

{

if(t->f->l==t)youxuan(t);

else zuoxuan(t);

}

void splay(Ts\*t,Ts\*f=0)

{

Ts\*tt;

while((tt=t->f)!=f)

{

if(tt->f!=f)

{

if((tt->f->l==tt)==(t->f->l==t))danxuan(tt);

else danxuan(t);

}

danxuan(t);

}

if(!f)root=t;

}

void insert(int x)

{

Ts\*t=root;

while(1)

{

t->size++;

if(x<t->x)

{

if(t->l)t=t->l;

else

{

t->l=New(x,t);

t=t->l;

break;

}

}

else

{

if(t->r)t=t->r;

else

{

t->r=New(x,t);

t=t->r;

break;

}

}

}

splay(t);

}

void erase(int x)

{

Ts\*t=root;

while(t->x!=x)

{

if(x<t->x)t=t->l;

else t=t->r;

}

splay(t);

Ts\*l=t->l,\*r=t->r;

while(l->r)l=l->r;

while(r->l)r=r->l;

splay(l);splay(r,l);

Del(r->l);

l->size--;r->size--;

}

Ts\*qianqu(int x)

{

Ts\*q,\*t=root;

int min=9999999;

while(t)

{

if(t->x>=x)t=t->l;

else

{

if(x-t->x<min)

{

q=t;

min=x-t->x;

}

t=t->r;

}

}

return q;

}

Ts\*houji(int x)

{

Ts\*q,\*t=root;

int min=9999999;

while(t)

{

if(t->x<=x)t=t->r;

else

{

if(t->x-x<min)

{

q=t;

min=t->x-x;

}

t=t->l;

}

}

return q;

}

int num(int l\_,int r\_)

{

Ts\*l=qianqu(l\_),\*r=houji(r\_);

splay(l);splay(r,l);

return r->l?r->l->size:0;

}

}Splay;

Splay::Ts Splay::mem[fa],\*Splay::stk[fa],\*\*Splay::top=Splay::stk+fa;

typedef struct Ts

{

Splay t;

Ts\*l,\*r;

}Ts;

Ts\*root,mem[fa2],\*stk[fa2],\*\*top=stk+fa2;

inline Ts\*New()

{

Ts\*p=\*--top;

p->l=p->r=0;

p->t.init();

return p;

}

inline void Del(Ts\*&t)

{

\*top++=t;

t->t.del(t->t.root);

t=0;

}

void insert(Ts\*&t,int x,int y,int L=0,int R=100000000)

{

if(!t)t=New();

t->t.insert(y);

if(L^R)

{

int mid=L+R>>1;

if(x<=mid)insert(t->l,x,y,L,mid);

else insert(t->r,x,y,mid+1u,R);

}

}

void erase(Ts\*&t,int x,int y,int L=0,int R=100000000)

{

t->t.erase(y);

if(L^R)

{

int mid=L+R>>1;

if(x<=mid)erase(t->l,x,y,L,mid);

else erase(t->r,x,y,mid+1u,R);

}

if(t->t.root->size==2)Del(t);

}

int kth(Ts\*t,int l,int r,int l\_,int r\_,int L=0,int R=100000000)

{

if(!t)return 0;

if(l==L&&r==R)return t->t.num(l\_,r\_);

else

{

int mid=L+R>>1;

if(r<=mid)return kth(t->l,l,r,l\_,r\_,L,mid);

else if(l>mid)return kth(t->r,l,r,l\_,r\_,mid+1,R);

else return kth(t->l,l,mid,l\_,r\_,L,mid)+kth(t->r,mid+1,r,l\_,r\_,mid+1,R);

}

}

int num(Ts\*t,int k,int l\_,int r\_,int L=0,int R=100000000)

{

if(L==R)return L;

int ls=t->l?t->l->t.num(l\_,r\_):0;

int mid=L+R>>1;

if(ls<k)return num(t->r,k-ls,l\_,r\_,mid+1,R);

else return num(t->l,k,l\_,r\_,L,mid);

}

int a[50000];

int main()

{

for(int i=0;i<fa;i++)Splay::stk[i]=Splay::mem+i;

for(int i=0;i<fa2;i++)stk[i]=mem+i;

int N,M,ls,l,r,k;

scanf("%d%d",&N,&M);

for(int i=0;i<N;i++)

{

scanf("%d",a+i);

insert(root,a[i],i);

}

while(M--)

{

scanf("%d",&ls);

if(ls==1)

{

scanf("%d%d%d",&l,&r,&k);

if(k==0)puts("1");

else printf("%d\n",kth(root,0,k-1,l-1,r-1)+1);

}

else if(ls==2)

{

scanf("%d%d%d",&l,&r,&k);

printf("%d\n",num(root,k,l-1,r-1));

}

else if(ls==3)

{

scanf("%d%d",&l,&k);--l;

erase(root,a[l],l);

insert(root,a[l]=k,l);

}

else if(ls==4)

{

scanf("%d%d%d",&l,&r,&k);

int no=kth(root,0,k-1,l-1,r-1);

if(no==0)puts("-2147483647");

else printf("%d\n",num(root,no,l-1,r-1));

}

else

{

scanf("%d%d%d",&l,&r,&k);

int no=kth(root,0,k,l-1,r-1);

if(no>r-l)puts("2147483647");

else printf("%d\n",num(root,no+1,l-1,r-1));

}

}

}