题解 P3380 【【模板】二逼平衡树（树套树）】

2016-12-03 07:59:35

thumb\_up *56*

说说此题我知道的几种方法

可修改区间k大的骗分做法:分块。

分成根号n个块，块内排序。

每个数在块内便可二分查找前驱，从而得到多少个数比他小。

块外部分暴力检验。

每次二分至一个值i,使得<i-1的数<k,<i的数<=k。

一次时间O(log范围\*(根号n\*log(根号n)))

暴力做法:跟上一种类似，线段树套平衡树

正解：

1 整体二分

就是二分答案跑cdq，

对于查询k大，我们得到左边的个数num，如果k<=num,去左边，否则k-=num,去右边

否则k<=mid分左，否则分右边。

对于查询rank，我们只在k>mid的时候，用num更新答案。

对于查询前驱，我们只在k>mid的时候，用左边的对应区间max更新答案(线段树维护即可)。

(我通过将所有数取反来将后继转化为前驱处理。)

此法较好实现。

设U为数的范围，时间(n+m)\*logUlogn。离散后应该是(n+m)\*lognlogn，但我懒得离散。。

目前拿了这题洛谷的rank2。

核心部分:

void solve(int L,int R,int l,int r)

{

if (l>r) return;

if (L==R)

{

for (i=l;i<=r;++i)

{

x=q[i];

if (a[x].type==2) ans[a[x].id]=L;

}

return;

}

int mid=(L+R)>>1;

t1=t2=t3=0;

for (i=l;i<=r;++i)

{

now=q[i];

if (a[now].type==2)

{

num=T.qiu\_s(a[now].l,a[now].r);

if (num<a[now].x) { a[now].x-=num;q2[++t2]=now; }

else q1[++t1]=now;

} else

{

if (a[now].x>mid) q2[++t2]=now;

else q1[++t1]=now;

if (a[now].type==3)

{

if (a[now].x<=mid)

{

q3[++t3]=now;

T.add(a[now].l,a[now].id,a[now].x);

}

} else

if (a[now].x>mid)

if (a[now].type==4)

{

chmax(ans[a[now].id],T.qiu\_m(a[now].l,a[now].r));

} else

ans[a[now].id]+=T.qiu\_s(a[now].l,a[now].r);

}

}

while (t3) { now=q3[t3--];T.clear(a[now].l); }

int k=l-1;

for (i=1;i<=t1;++i) q[++k]=q1[i];

for (i=1;i<=t2;++i) q[k+i]=q2[i];

solve(L,mid,l,k);solve(mid+1,R,k+1,r);

}

2树状数组套动态加点线段树

不知道为什么很多人说是可持久化线段树。其实我们是对树状数组每个节点套了个线段树，彼此之间并没有关联的。

想法就是，假如我们得到了[l,r]的权值线段树，那么显然一切操作都能logn做了。

由于我们线段树维护的是权值区间内的点的个数，这是可以加和的。

所以如果我们给每个点建个线段树，那么就是询问[l,r]的线段树的和。

那么实际上这就是以线段树为对象的单点修改，区间求和!

用树状数组应该是最好的，减少了修改时涉及的线段树的个数，保护时空。

数据也是要先离散的，保护时空。

其实主要是保护空间啦，毕竟动态加点的上限只有8e6(再大也开不了)，这是远远小于时间上限的。

时间(n+m)\*lognlogn。空间也一样。

目前拿了这题bzoj的rank1(当然也是洛谷的)。

#include<bits/stdc++.h>

using namespace std;

#define ch\_top 10000000

char ch[ch\_top],\*now\_r=ch;

void read(int &x)

{

while (\*now\_r<48) ++now\_r;

for (x=\*now\_r-48;\*++now\_r>=48;)

x=(x<<1)+(x<<3)+\*now\_r-48;

}

#define N 50010

#define M 20

#define MN 8000000

#define oo 2147483647

struct tree

{

int c[2],s;

}T[MN];int TOT;

struct query

{

int o,l,r,k;

}a[N];

int c[N],last[N];

int \*p[N<<1],dy[N<<1],pt;

#define L(x) T[x].c[0]

#define R(x) T[x].c[1]

#define S(x) T[x].s

#define LX T[x].c[0]

#define RX T[x].c[1]

#define mid ((l+r)>>1)

int l,r,w;

void modify(int x,int i)

{

l=1;r=pt;

for (;;)

{

S(x)+=w;

if (l==r) return;

if (i<=mid)

{

if (!LX) LX=++TOT;

x=LX;r=mid;

}

else

{

if (!RX) RX=++TOT;

x=RX;l=mid+1;

}

}

}

int n;

void mo(int i,int x)

{

for (;i<=n;i+=i&-i)

modify(c[i],x);

}

int ql[M][M],qr[M][M];

void init(int \*q,int i)

{

for (q[0]=0;i;i-=i&-i)

q[++q[0]]=c[i];

}

int i;

void \_move(int \*qx,int \*qy,bool d)

{

for (i=qy[0]=qx[0];i;--i)

qy[i]=T[qx[i]].c[d];

}

void move(int x,int y,bool d)

{

\_move(ql[x],ql[y],d);

\_move(qr[x],qr[y],d);

}

int Get;

int \_get(int \*q,bool d)

{

for (i=1,Get=0;i<=q[0];++i) Get+=S(T[q[i]].c[d]);

return Get;

}

int get(int x,bool d)

{

return \_get(qr[x],d)-\_get(ql[x],d);

}

int ans;

void order(int x)

{

l=1;r=pt;

for (ans=1;l!=r;)

if (x>mid) {ans+=get(0,0);move(0,0,1);l=mid+1;} else

{move(0,0,0);r=mid;}

}

void find\_by\_order(int k)

{

l=1;r=pt;

while (l!=r)

{

ans=get(0,0);

if (k>ans) {k-=ans;move(0,0,1);l=mid+1;} else

{move(0,0,0);r=mid;}

}

ans=dy[l];

}

struct ever

{

int l,r;

}e[M];

int top;

void pre(int x)

{

l=1;r=pt;top=0;

while (l!=r)

if (x>mid)

{

e[top]={l,mid};

move(top,top+1,1);++top;

l=mid+1;

}

else

{ move(top,top,0);r=mid; }

while (top--)

if ( get(top,0) )

{

move(top,0,0);

l=e[top].l;r=e[top].r;

while (l!=r)

if ( get(0,1) ) { move(0,0,1);l=mid+1; }

else { move(0,0,0);r=mid; }

ans=dy[l];

return ;

}

ans=-oo;

}

void next(int x)

{

l=1;r=pt;top=0;

while (l!=r)

if (x>mid)

{

move(top,top,1);

l=mid+1;

}

else

{

e[top]={mid+1,r};

move(top,top+1,0);++top;

r=mid;

}

while (top--)

if ( get(top,1) )

{

move(top,0,1);

l=e[top].l;r=e[top].r;

while (l!=r)

if ( get(0,0) ) { move(0,0,0);r=mid; }

else { move(0,0,1);l=mid+1; }

ans=dy[l];

return ;

}

ans=oo;

}

bool xiao(const int \*x,const int \*y)

{

return \*x<\*y;

}

int main()

{ //freopen("1.in","r",stdin);freopen("1.out","w",stdout);

ch[fread(ch,1,ch\_top,stdin)]=0;

int m,i,o;

read(n);read(m);

for (i=1;i<=n;++i)

{

read(last[i]);

p[i]=last+i;

}

int tot=n;

for (i=1;i<=m;++i)

{

read(a[i].o);read(a[i].l);read(a[i].r);

if (a[i].o==3) p[++tot]=&a[i].r;

else

{

read(a[i].k);

if (a[i].o!=2) p[++tot]=&a[i].k;

}

}

//先离散掉，节约开线段树的时空

sort(p+1,p+tot+1,xiao);

dy[1]=\*p[1];\*p[1]=pt=1;

for (i=2;i<=tot;++i)

{

if (\*p[i]!=dy[pt]) dy[++pt]=\*p[i];

\*p[i]=pt;

}

w=1;

for (i=1;i<=n;++i) c[i]=i;TOT=n;

for (i=1;i<=n;++i) mo(i,last[i]);

for (o=1;o<=m;++o)

{

if (a[o].o==3)

{

w=-1;i=a[o].l;

mo(i,last[i]);

w=1;

mo(i,last[i]=a[o].r);

} else

{

init(ql[0],a[o].l-1);init(qr[0],a[o].r);

if (a[o].o==1) order(a[o].k); else

if (a[o].o==2) find\_by\_order(a[o].k); else

if (a[o].o==4) pre(a[o].k);

else next(a[o].k);

printf("%d\n",ans);

}

}

}