

你只要把普通线段树里的p<<1换成lc[p]，p<<1|1换成rc[p]就行了。

灵活运用动态开点线段树可以节省很多内存，而且能做到普通线段树做不到的事情。

比如题目要求在线操作不能离散化，值域又特别大：inf，并且询问q不大

这时候我们就可以用动态开点线段树开qloginf个点过掉这题。（也就是说时间复杂度为qlog值域）

#include<bits/stdc++.h>

#define ll long long

#define inf 1e18

#define mn 100005

using namespace std;

struct w{ll s,tag,ls,rs;}

a[mn\*4];//线段树数组开4倍,虽然动态开点的极限是2倍，但是开4倍比较保险

ll rt,b[mn],tot,n,m;

void up(ll p)

{

a[p].s=a[a[p].ls].s+a[a[p].rs].s;

}

void bui(ll &p,ll l,ll r)

{

if(!p)p=++tot;

if(l==r)

{

a[p].s=b[l];

return;

}

ll mid=(l+r)>>1;

bui(a[p].ls,l,mid);

bui(a[p].rs,mid+1,r);

up(p);

}

void push\_tag(ll &p,ll l,ll r,ll k)

{

if(!p)p=++tot;

a[p].s+=(r-l+1)\*k;

a[p].tag+=k;

}

void down(ll p,ll l,ll r)

{

if(a[p].tag==0)return;

ll mid=(l+r)>>1;

push\_tag(a[p].ls,l,mid,a[p].tag);

push\_tag(a[p].rs,mid+1,r,a[p].tag);

a[p].tag=0;

}

void add(ll &p,ll l,ll r,ll L,ll R,ll k)

{

if(!p)p=++tot;

if(L<=l&&R>=r)

{

push\_tag(p,l,r,k);

return;

}

down(p,l,r);

ll mid=(l+r)>>1;

if(L<=mid)add(a[p].ls,l,mid,L,R,k);

if(R>=mid+1)add(a[p].rs,mid+1,r,L,R,k);

up(p);

}

ll ask(ll p,ll l,ll r,ll L,ll R)

{

if(L<=l&&R>=r)return a[p].s;

down(p,l,r);

ll mid=(l+r)>>1;

ll res=0;

if(L<=mid)res+=ask(a[p].ls,l,mid,L,R);

if(R>=mid+1)res+=ask(a[p].rs,mid+1,r,L,R);

return res;

}

ll op,x,y,k;

int main()

{

scanf("%lld%lld",&n,&m);

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

scanf("%lld",&b[i]);

bui(rt,1,n);//建立的第一个节点即为根

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

{

scanf("%lld",&op);

if(op==1)

{

scanf("%lld%lld%lld",&x,&y,&k);

add(rt,1,n,x,y,k);//从根开始找

//rt为1，可以直接写1，不过我觉得写rt更明白一些

}

else

{

scanf("%lld%lld",&x,&y);

printf("%lld\n",ask(rt,1,n,x,y));

}

}

return 0;

}