T4 大战杀马特(smart)

50 pts

每次询问做一遍 【NOI2010 超级钢琴】

复杂度 $O(mn\log n+(\sum k)\log(\sum k))$

另 15 pts: k=1

考虑线段树解决

对于每个点我们维护：区间最大值 max、区间最小值 min、区间的最大答案 ans

这样对于一个节点 p， $ans_p=\min(ans_{p->ls},ans_{p->rs},max_{p->ls}-min_{p->rs})$

区间 query 方法类似

对于区间加，tag 对 max,min 的影响都是直接加，而对 ans 没有影响

（注意到这个和动态最大子段和求解的方法非常像，实际上是因为把 a 进行差分之后我们所求的基本就是最大子段和……）

复杂度 $O(m\log n)$

100pts

我们假设你已经会了 【NOI2010 超级钢琴】

注意到这道题的正解当中采用了“候补答案集合”的思想：把左端点 $=x$ ，右端点 $\in[l,r]$ 的所有答案视作一个 node 放进优先队列，每次取答案最小的一个进行累加，然后按最优解的位置把这个“候补答案集合”分裂成两个

这显然不够带劲。我们能不能直接用一个 node 表示一个矩形（左端点属于一个区间，右端点也属于一个区间）的答案?

一般的矩形是不好求解最优答案的。但是有两种可以：左右端点属于的区间完全重合（也就是 $k=1$ 的做法）和左右端点属于的区间完全相离（在左边取最大值，再在右边取最小值，二者相减）

我们的目标是求解这两种矩形，然后在分裂的时候也保证得到的“候补答案集合”也属于这两种矩形，事实上这是可以做到的

先看第一种：左端点、右端点都 $\in[l,r]$，假设最优解位于 (x,y)

我们把它分裂成如下矩形：

左端点$\in [l,x-1]$，右端点$\in [l,x-1]$ $(x>l)$

左端点$\in [l,x-1]$，右端点$\in [x,r]$ $(x>l)$

左端点$\in [x,x]$，右端点$\in [x,x]$ $(x\neq y)$

左端点$\in [x,x]$，右端点$\in [x+1,y-1]$ $(x<y-1)$

左端点$\in [x,x]$，右端点$\in [y+1,r]$ $(y<r)$

左端点$\in [x+1,r]$，右端点$\in [x+1,r]$ $(x<r)$

再看第二种：左端点 $\in [l_0,r_0]$ ，右端点 $\in [l_1,r_1]$，$l_1>r_0$，假设最优解位于 (x,y)

我们把它分裂成如下矩形：

左端点$\in [l_0,x-1]$，右端点$\in [l_1,r_1]$ $(x>l_0)$

左端点$\in [x,x]$，右端点$\in [l_1,y-1]$ $(l_1<y)$

左端点$\in [x,x]$，右端点$\in [y+1,r_1]$ $(y<r_1)$

左端点$\in [x+1,r_0]$，右端点$\in [l_1,r_1]$ $(x<r_0)$

就好了（

复杂度 $O(n\log n+(\sum k)(\log n+\log(\sum k)))$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=100005;
int n,m;
ll a[N];
struct node{
    ll mx,mn,ans;
    int mxp,mnp,pl,pr;
    inline node operator +(const node &b)const{
        node ret;
        if(mx>b.mx) ret.mx=mx,ret.mxp=mxp;
        else ret.mx=b.mx,ret.mxp=b.mxp;
        if(mn<b.mn) ret.mn=mn,ret.mnp=mnp;
        else ret.mn=b.mn,ret.mnp=b.mnp;
        if(mx-b.mn>=max(ans,b.ans)) ret.ans=mx-b.mn,ret.pl=mxp,ret.pr=b.mnp;
        else if(ans>=max(mx-b.mn,b.ans)) ret.ans=ans,ret.pl=pl,ret.pr=pr;
        else ret.ans=b.ans,ret.pl=b.pl,ret.pr=b.pr;
        return ret;
    }
};
struct sgt{
    int l,r,ls,rs;
    node w;
    int tag;
}s[3*N];
int rt,P;
inline void upd(int p){
    s[p].w=s[s[p].ls].w+s[s[p].rs].w;
}
int build(int l,int r){
    int p=++P;
    s[p].l=l;s[p].r=r;
```

```
    if(l==r){
        s[p].w.mx=s[p].w.mn=a[l];s[p].w.ans=0;
        s[p].w.mxp=s[p].w.mnp=s[p].w.pl=s[p].w.pr=l;
        return p;
    }
    int mid=(l+r)>>1;
    s[p].ls=build(l,mid);
    s[p].rs=build(mid+1,r);
    upd(p);
    return p;
}
inline void pushd(int p){
    if(!s[p].tag) return;
    if(s[p].ls) s[s[p].ls].tag+=s[p].tag;
    if(s[p].rs) s[s[p].rs].tag+=s[p].tag;
    s[p].w.mx+=s[p].tag;s[p].w.mn+=s[p].tag;
    s[p].tag=0;
}
void chg(int p,int l,int r,ll x){
    pushd(p);
    if(s[p].l==l&&s[p].r==r){
        s[p].tag+=x;
        pushd(p);
        return;
    }
    int mid=(s[p].l+s[p].r)>>1;
    if(l>mid){
        chg(s[p].rs,l,r,x);
        pushd(s[p].ls);
    }
    else if(r<=mid){
        chg(s[p].ls,l,r,x);
        pushd(s[p].rs);
    }
    else{
        chg(s[p].ls,l,mid,x);
        chg(s[p].rs,mid+1,r,x);
    }
    upd(p);
}
node qry(int p,int l,int r){
    pushd(p);
    if(s[p].l==l&&s[p].r==r)
        return s[p].w;
    int mid=(s[p].l+s[p].r)>>1;
    if(l>mid) return qry(s[p].rs,l,r);
    else if(r<=mid) return qry(s[p].ls,l,r);
    else return qry(s[p].ls,l,mid)+qry(s[p].rs,mid+1,r);
}
struct answ{
    int l0,r0,l1,r1;
    int pl,pr;
    ll ans;
    inline answ(int l0,int r0,int l1,int r1):l0(l0),r0(r0),l1(l1),r1(r1){}
    inline bool operator <(const answ &b)const{
        return ans<b.ans;
    }
};
```

```
priority_queue<answ> q;
inline void put(int l0,int r0,int l1,int r1){
    answ a(l0,r0,l1,r1);
    if(a.l0==a.l1&&a.r0==a.r1){
        node qq=qry(rt,a.l0,a.r0);
        a.pl=qq.pl;a.pr=qq.pr;a.ans=qq.ans;
    }
    else{
        node ql=qry(rt,a.l0,a.r0),qr=qry(rt,a.l1,a.r1);
        a.pl=ql.mxp;a.pr=qr.mnp;a.ans=ql.mx-qr.mn;
    }
    q.push(a);
}
int main(){
    int i,j;
    scanf("%d%d",&n,&m);
    for(i=1;i<=n;i++) scanf("%lld",&a[i]);
    rt=build(1,n);
    int k=0;
    while(m--){
        int op,x,y;ll z;scanf("%d%d%d%lld",&op,&x,&y,&z);
        if(op==1)
            chg(rt,x,y,z);
        else{
            while(!q.empty()) q.pop();
            put(x,y,x,y);
            ll ans=0;
            k+=z;
            while(z--){
                answ cur=q.top();q.pop();
                ans+=cur.ans;
                int l=cur.pl,r=cur.pr,l0=cur.l0,r0=cur.r0,l1=cur.l1,r1=cur.r1;
                if(l0==l1&&r0==r1){
                    if(l0<l){
                        put(l0,l-1,l0,l-1);
                        put(l0,l-1,l,r0);
                    }
                    if(l!=r) put(l,l,l,l);
                    if(l+1<=r-1) put(l,l,l+1,r-1);
                    if(r+1<=r0) put(l,l,r+1,r0);
                    if(l+1<=r0) put(l+1,r0,l+1,r0);
                }
                else{
                    if(l0<l) put(l0,l-1,l1,r1);
                    if(l1<r) put(l,l,l1,r-1);
                    if(r+1<=r1) put(l,l,r+1,r1);
                    if(l+1<=r0) put(l+1,r0,l1,r1);
                }
            }
            printf("%lld\n",ans);
        }
    }
    return 0;
}
```