比赛链接：AtCoder Beginner Contest 128

E - Roadwork

题意

在一条直线上有 N 个障碍，位于 x_i且每个障碍仅在 $[S_i,T_i)$ 生效。有 Q 个人从 $ x = 0$ 等待 D_i 秒后出发，以每秒一单位长度的速度向正方向行走，遇到生效的障碍则立刻停下。输出每个人走的路程，如果无穷则输出 -1。

数据范围

$1 \le N,Q \le 2\times 10^5$

$0 \le S_i,T_i \le 10^9$

$1 \le X_i \le 10^9$

$0 \le D_1 \lt D_2 \lt \cdots \lt D_Q \le 10^9$

题解

第 j 个人在第 i 个障碍停下当且仅当 i 是满足 $S_i \le x_i + D_j \lt T_i$ 且 x_i 最小的。也即 $S_i - x_i \le D_j \lt T_i - x_i$

所以考虑对每个障碍按 x_i 排序，然后从大到小枚举。每次二分得到满组 $S_i - x_i \le D_j \lt T_i - x_i$ 的一个区间，然后用线段树染色。

```
#include<bits/stdc++.h>
#define ll long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

int s[maxn],t[maxn],x[maxn],d[maxn],id[maxn];
int tr[maxn<<2],lazy[maxn<<2];
```

```
void push_down(int id)
{
    if(lazy[id]!=-1){
        tr[id<<1] = tr[id<<1|1] = lazy[id<<1] = lazy[id<<1|1] = lazy[id];
        lazy[id] = -1;
    }
}
void build(int id,int l,int r)
{
    tr[id] = -1,lazy[id] = -1;
    if(l==r) return;
    int mid = (l+r)>>1;
    build(id<<1,l,mid);build(id<<1|1,mid+1,r);
}
void update(int id,int stl,int str,int l,int r,int k)
{
    if(stl==l && str==r){
        tr[id] = k;
        lazy[id] = k;
        return ;
    }
    push_down(id);
    int mid = (stl+str)>>1;
    if(r<=mid) update(id<<1,stl,mid,l,r,k);
    else if(l>mid) update(id<<1|1,mid+1,str,l,r,k);
    else update(id<<1,stl,mid,l,mid,k),update(id<<1|1,mid+1,str,mid+1,r,k);
}
int query(int id,int stl,int str,int pos)
{
    if(stl == str) return tr[id];
    push_down(id);
    int mid = (stl+str)>>1;
    if(pos<=mid) return query(id<<1,stl,mid,pos);
    else return query(id<<1|1,mid+1,str,pos);
}
int main()
{
    fastio();
    int n,q;
    cin>>n>>q;
    rep(i,1,n) cin>>s[i]>>t[i]>>x[i];
    rep(i,1,q) cin>>d[i];
    build(1,1,q);
    rep(i,1,n) id[i] = i;
    sort(id+1,id+n+1,[](int a,int b){return x[a]<x[b];});
    per(i,n,1){
        int L = lower_bound(d+1,d+q+1,s[id[i]]-x[id[i]]) - d;
        int R = lower_bound(d+1,d+q+1,t[id[i]]-x[id[i]]) - d - 1;
        if(R<L || R==0) continue;
        update(1,1,q,L,R,x[id[i]]);
```

```
    }
    rep(i,1,q){
        cout<<query(1,1,q,i)<<endl;
    }
    return 0;
}
```

F - Frog Jump

题意

在一条数轴上有 N 个点 $0,1,2 \dots n-1$，每个点权值为 s_i，在开始前选两个正整数 A,B，青蛙从 0 开始跳，跳的序列为$0,A,A-B,2A-B\cdots k(A-B),k(A-B)+A = n-1$。

每次必须跳在 $0\codts n-1$ 上，且每个点最多跳一次，求跳过得所有点的最大权值和。

数据范围

$3 \le N \lt 10^5$

$-10^9 \le s_i \le 10^9$

题解

将序列拆分成两个序列，$k*(A-B)$ 和 $k*(A-B) + A$。

记 $d = A-B$,可以看成是两个等差数列 $k*d$ 和 $n-1 - k*d$

所以预处理每个 d 的前后缀，再枚举 k 和 d 取最大值。

```
#include<bits/stdc++.h>
#define ll long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
```

```
const int maxn = 1e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

int s[maxn];
vector<ll> pre[maxn],suf[maxn];
int main()
{
    fastio();
    int n; cin>>n;
    rep(i,0,n-1) cin>>s[i];
    rep(i,1,n-1){
        for(int j=0;j<n;j+=i){
            if(pre[i].size() == 0) pre[i].pb(s[j]);
            else pre[i].pb(pre[i].back() + s[j]);
        }
        for(int j=n-1;j>=0;j-=i){
            if(suf[i].size() == 0) suf[i].pb(s[j]);
            else suf[i].pb(suf[i].back() + s[j]);
        }
    }
    ll ans = 0;
    rep(k,1,n-1){
        for(int d=1;d*k<n-1;d++){
            int A = n-1-k*d;
            int B = A-d;
            if(B <= 0) break;
            if(A%d==0 && A/d<=k) continue;
            ans = max(ans,pre[d][k] + suf[d][k]);
        }
    }
    cout<<ans<<endl;
    return 0;
}
```

题解