[报告] E

[source] http://202.114.18.202:8080/judge/contest/view.action?cid=6216#problem/E

[Description]

N 个小孩玩约瑟夫环问题，从第 K 个少年开始，下一轮如果当前少年的值是 m,就顺时针左走走 m 步，如果当前少年值是-m，就逆时针走 m 步。第 s 个少年出列之后，将获得 f(s)个糖果，f(s)为 s 的约数个数，N 个小孩做游戏，获得糖果最多的是谁，有多少个。

[方法]

首先，我们给出反素数的定义：令 f(s)为 s 的约数个数,则对任意的 a,f(a)>f(b)（对所有的 b 小于 a 成立）。那么我们可以知道，本题中获得最多的糖果即为小于等于 N 的最大反素数的约数个数。由于反素数个数有限,我们可以预处理出来所有的符合 N 范围内的反素数。(通过 DFS)。假设小于等于 N 的最大反素数是 S，那么本题只用模拟 S 次即可，每次模拟使用线段树求第 K 大的方法，找到下次应该到的小孩的序号。

[Code]

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
#define N 500000
int antiprime[50]={1,2,4,6,12,24,36,48,60,120,180,240,360,720,840,
```

```
1260,1680,2520,5040,7560,10080,15120,20160,25200,27720,45360,504
00,55440,83160,110880,166320,221760,277200,332640,498960} ;
int number[50]={1,2,3,4,6,8,9,10,12,16,18,20,24,30,32,36,40,48,60,64,
72,80,84,90,96,100,108,120,128,144,160,168,180,192,200};
int val[N<<2];
void build(int left,int right,int idx)
{
    if(left==right){
        val[idx]=1;
        return ;
    }
    int mid=(left+right)>>1;
    build(left,mid,idx<<1);
    build(mid+1,right,idx<<1|1);
    val[idx]=val[idx<<1]+val[idx<<1|1];
}
void update(int left,int right,int id,int idx)
{
    if(left==right){
        val[idx]=0;
        return ;
    }
```

```
    int mid=(left+right)>>1;
    if(mid<id){
        update(mid+1,right,id,idx<<1|1);
    }else{
        update(left,mid,id,idx<<1);
    }
    val[idx]=val[idx<<1]+val[idx<<1|1];
}
int query(int left,int right,int L,int R,int idx)
{
    if(left>=L&&right<=R){
        return val[idx];
    }
    int mid=(left+right)>>1;
    if(mid<L){
        return query(mid+1,right,L,R,idx<<1|1);
    }else if(mid>=R){
        return query(left,mid,L,R,idx<<1);
    }else{
        return
query(left,mid,L,mid,idx<<1)+query(mid+1,right,mid+1,R,idx<<1|1);
    }
```

```
}
int findk(int left,int right,int k,int idx)
{
    if(left==right){
        return left;
    }
    int mid=(left+right)>>1;
    if(val[idx<<1]>=k){
        return findk(left,mid,k,idx<<1);
    }
    else{
        return findk(mid+1,right,k-val[idx<<1],idx<<1|1);
    }
}
struct P
{
    char s[15];
    int op;
}tt[N+10];
int main(void)
{
    int n,m,M,K,Ans;
```

```
while(scanf("%d%d",&n,&m)!=EOF)
{
    for(int i=0;i<35;i++)   {
        if(i==34||(antiprime[i]<=n&&antiprime[i+1]>n)){
            K=antiprime[i];
            Ans=number[i];
            break;
        }
    }
    build(1,n,1);
    for(int i=1;i<=n;i++){
        scanf("%s%d",tt[i].s,&tt[i].op);
    }
    int st=m;
    M=n;
    int mark;
    for(int i=1;i<K;i++){
        M--;
        update(1,n,st,1);
        if(tt[st].op>0){
            int num=tt[st].op;
            num=num%M==0?M:num%M;
```

```
                int sum=query(1,n,st,n,1);
                if(sum>=num) {
                    st=findk(1,n,M-sum+num,1);
                }else{
                    st=findk(1,n,num-sum,1);
                }
            }else{
                int num=-tt[st].op;
                num=num%M==0?M:num%M;
                int sum=query(1,n,1,st,1);
                if(sum>=num) {
                    st=findk(1,n,sum-num+1,1);
                }else{
                    st=findk(1,n,M+sum-num+1,1);
                }
            }
        }
        printf("%s %d\n",tt[st].s,Ans);
    }
    return 0;
}
```