数据结构：

划分树

#include<cstdio>

#include<iostream>

#include<cstring>

#include<algorithm>

using namespace std;

const int maxn=100010;

int tree[30][maxn];

int sorted[maxn];

int toleft[30][maxn];

void build(int l,int r,int dep)

{

if (l==r) return;

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

int same=mid-l+1;

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

if (tree[dep][i]<sorted[mid]) same--;

int lpos=l;

int rpos=mid+1;

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

{

if (tree[dep][i]<sorted[mid]) tree[dep+1][lpos++]=tree[dep][i];

else if (tree[dep][i]==sorted[mid]&&same>0)

{

tree[dep+1][lpos++]=tree[dep][i];

same--;

}

else tree[dep+1][rpos++]=tree[dep][i];

toleft[dep][i]=toleft[dep][l-1]+lpos-l;

}

build(l,mid,dep+1);

build(mid+1,r,dep+1);

}

int query(int L,int R,int l,int r,int dep,int k)

{

if (l==r) return tree[dep][l];

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

int cnt=toleft[dep][r]-toleft[dep][l-1];

if (cnt>=k)

{

int newl=L+toleft[dep][l-1]-toleft[dep][L-1];

int newr=newl+cnt-1;

return query(L,mid,newl,newr,dep+1,k);

}

else

{

int newr=r+toleft[dep][R]-toleft[dep][r];

int newl=newr-(r-l-cnt);

return query(mid+1,R,newl,newr,dep+1,k-cnt);

}

}

int main()

{

int n,m;

while (~scanf("%d %d",&n,&m))

{

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

{

scanf("%d",&tree[0][i]);

sorted[i]=tree[0][i];

}

sort(sorted+1,sorted+n+1);

build(1,n,0);

while (m--)

{

int a,b,c;

scanf("%d %d %d",&a,&b,&c);

printf("%d\n",query(1,n,a,b,0,c));

}

}

return 0;

}

块状链表

const int m=350;

struct data

{

int s,a[2\*m+5];

data \*next;

};

data \*root;

void insert(int x,int pos)

{

if (root=NULL)

{

root=new(data);

root->s=1;

root->a[1]=x;

return ;

}

data \*k=root;

while (pos>k->s && k->next!=NULL)

{

pos-=k->s;

k=k->next;

}

memmove(k->a+pos+1,k->a+pos,sizeof(int)\*(k->s-pos+1));

k->s++;

k->a[pos]=x;

if (k->s==2\*m)

{

data \*t=new(data);

t->next=k->next;

k->next=t;

memcpy(t->a+1,k->a+m+1,sizeof(int)\*m);

t->s=k->s=m;

}

}

void del(int pos)

{

data \*k=root;

while (pos>k->s && k->next!=NULL)

{

pos-=k->s;

k=k->next;

}

memmove(k->a+pos,k->a+pos+1,sizeof(int)\*(k->s-pos));

k->s--;

}

int find(int pos)

{

data \*k=root;

while (pos>k->s && k->next!=NULL)

{

pos-=k->s;

k=k->next;

}

return (k->a[pos]);

}

void destroy(data \*k)

{

if (k->next!=NULL)

destroy(k->next);

delete(k);

}

树状数组

const int maxn=100000;

int Tree[maxn+10];

inline int lowbit(int x)

{

return (x&-x);

}

void add(int x,int value)

{

for (int i=x;i<=maxn;i+=lowbit(i))

Tree[i]+=value;

}

int get(int x)

{

int sum=0;

for (int i=x;i;i-=lowbit(i))

sum+=Tree[i];

return sum;

}

线段树

struct data{

int l,r;

int min,lazy;

}tr[2000001];

int a[1000050];

int flag=0,pos;

void build(int k,int s,int t)

{

tr[k].l=s;tr[k].r=t;

tr[k].lazy=0;

if(s==t)

{

tr[k].min=a[s];

return;

}

int mid=(s+t)>>1;

build(k<<1,s,mid);

build(k<<1|1,mid+1,t);

tr[k].min=min(tr[k<<1].min,tr[k<<1|1].min);

}

void pushdown(int k)

{

tr[k<<1].min-=tr[k].lazy;

tr[k<<1|1].min-=tr[k].lazy;

tr[k<<1].lazy+=tr[k].lazy;

tr[k<<1|1].lazy+=tr[k].lazy;

tr[k].lazy=0;

}

void change(int k,int a,int b,int c)

{

int l=tr[k].l,r=tr[k].r;

if(a==l&&b==r)

{

tr[k].min-=c;

tr[k].lazy+=c;

if (tr[k].min<0)

{

flag=1;

}

return;

}

if(tr[k].lazy) pushdown(k);

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

if(b<=mid) change(k<<1,a,b,c);

else if(a>mid) change(k<<1|1,a,b,c);

else change(k<<1,a,mid,c),change(k<<1|1,mid+1,b,c);

tr[k].min=min(tr[k<<1].min,tr[k<<1|1].min);

}

左偏树

struct data{

int l,r;

int min,lazy;

}tr[2000001];

int a[1000050];

int flag=0,pos;

void build(int k,int s,int t)

{

tr[k].l=s;tr[k].r=t;

tr[k].lazy=0;

if(s==t)

{

tr[k].min=a[s];

return;

}

int mid=(s+t)>>1;

build(k<<1,s,mid);

build(k<<1|1,mid+1,t);

tr[k].min=min(tr[k<<1].min,tr[k<<1|1].min);

}

void pushdown(int k)

{

tr[k<<1].min-=tr[k].lazy;

tr[k<<1|1].min-=tr[k].lazy;

tr[k<<1].lazy+=tr[k].lazy;

tr[k<<1|1].lazy+=tr[k].lazy;

tr[k].lazy=0;

}

void change(int k,int a,int b,int c)

{

int l=tr[k].l,r=tr[k].r;

if(a==l&&b==r)

{

tr[k].min-=c;

tr[k].lazy+=c;

if (tr[k].min<0)

{

flag=1;

}

return;

}

if(tr[k].lazy) pushdown(k);

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

if(b<=mid) change(k<<1,a,b,c);

else if(a>mid) change(k<<1|1,a,b,c);

else change(k<<1,a,mid,c),change(k<<1|1,mid+1,b,c);

tr[k].min=min(tr[k<<1].min,tr[k<<1|1].min);

}

LCT

const int MAXN = 400010;

int ch[MAXN][2],fa[MAXN],xo[MAXN];

bool rt[MAXN];

int rev[MAXN],a[MAXN];

void update(int x)

{

if (!x) return;

rev[x]^=1;

}

void push\_down(int x)

{

if (rev[x])

{

swap(ch[x][0],ch[x][1]);

update(ch[x][0]);

update(ch[x][1]);

rev[x]=0;

}

}

void push\_up(int x)

{

xo[x]=xo[ch[x][0]]^xo[ch[x][1]]^a[x];

}

void rotate(int x)

{

int y=fa[x],p=ch[y][1]==x;

ch[y][p]=ch[x][!p];

fa[ch[y][p]] = y;

fa[x]=fa[y];

fa[y]=x;

ch[x][!p]=y;

if(rt[y])

rt[y]=false,rt[x]=true;

else

ch[fa[x]][ch[fa[x]][1]==y]=x;

push\_up(y);

push\_up(x);

}

void maintain(int x)

{

if(!rt[x]) maintain(fa[x]);

push\_down(x);

}

void splay(int x)

{

maintain(x);

while(!rt[x])

{

int f=fa[x], ff=fa[f];

if(rt[f])

rotate(x);

else if( (ch[ff][1]==f)==(ch[f][1]==x) )

rotate(f),rotate(x);

else

rotate(x),rotate(x);

}

}

void access(int x)

{

int y=0,z=x;

for(;x;x=fa[y=x])

{

splay(x);

rt[ch[x][1]]=true,rt[ch[x][1]=y]=false;

push\_up(x);

}

splay(z);

}

int findroot(int x)

{

access(x);

splay(x);

while (ch[x][0]!=0) x=ch[x][0];

return x;

}

bool judge(int x,int y)

{

return findroot(x)==findroot(y);

}

void mroot(int x)

{

access(x);

rev[x]^=1;

}

void cut(int x,int y)

{

mroot(x);

splay(y);

fa[ch[y][0]]=fa[y];

fa[y]=0;

rt[ch[y][0]]=true;

ch[y][0]=0;

push\_up(y);

}

void link(int x,int y)

{

mroot(x);

fa[x]=y;

}

int ask(int x,int y)

{

mroot(x);

access(y);

return xo[ch[y][0]]^a[y];

}

int judgeedge(int x,int y)

{

mroot(x);

access(y);

int p=ch[y][0];

while (ch[p][1])

p=ch[p][1];

return p==x;

}

ST表

const int MAX=100000;

int stTable[MAX][32];

int preLog2[MAX];

void st\_prepare(int n,int \*array)

{

preLog2[1]=0;

for (int i=2;i<=n;i++)

{

preLog2[i]=preLog2[i-1];

if ((1<<preLog2[i]+1)==i)

{

preLog2[i]++;

}

}

for (int i=n-1;i>=0;i--)

{

stTable[i][0]=array[i];

for (int j=1;(i+(1<<j)-1)<n;j++)

{

stTable[i][j]=min(stTable[i][j-1],stTable[i+(1<<j-1)][j-1]);

}

}

}

int query\_min(int l,int r)

{

int len=r-l+1,k=preLog2[len];

return min(stTable[l][k],stTable[r-(1<<k)+1][k]);

}

Subsplay

#include<cstdio>

#include<iostream>

#include<algorithm>

#include<cmath>

#include<cstring>

#include<cstdlib>

#include<set>

#include<map>

#include<stack>

#include<queue>

#include<vector>

using namespace std;

const int maxn=110000;

struct node

{

int data,num,size,c[2],fa,rev;

}tree[maxn];

int root=0,sum=0,res,n,m,num=0;

void downdate(int x)

{

if (!tree[x].rev) return;

tree[x].rev=0;

tree[tree[x].c[0]].rev^=1;

tree[tree[x].c[1]].rev^=1;

swap(tree[x].c[0],tree[x].c[1]);

}

void update(int x)

{

tree[x].size=tree[tree[x].c[0]].size+tree[tree[x].c[1]].size+tree[x].num;

}

void rotate(int x,int p)

{

int y=tree[x].fa;

if (tree[x].c[p]) tree[tree[x].c[p]].fa=y;

tree[y].c[p^1]=tree[x].c[p];

tree[x].c[p]=y;

tree[x].fa=tree[y].fa;

tree[y].fa=x;

if (tree[x].fa) if (y==tree[tree[x].fa].c[0]) tree[tree[x].fa].c[0]=x;

else tree[tree[x].fa].c[1]=x;

update(y);

update(x);

}

void splay(int x,int p)

{

while (tree[x].fa!=0)

{

int y=tree[x].fa;

int z=tree[y].fa;

downdate(z);

downdate(y);

downdate(x);

if (z==0||(tree[z].fa==0&&p))

{

if (p&&z==0) break;

if (tree[y].c[0]==x) rotate(x,1);

else rotate(x,0);

break;

}

int a=y==tree[z].c[0]?1:0;

int b=x==tree[y].c[0]?1:0;

if (a^b) rotate(x,b),rotate(x,a);

else rotate(y,a),rotate(x,b);

}

if (!p) root=x;

}

void maintain(int x)

{

while (x!=0)

update(x),x=tree[x].fa;

}

void init(int x,int sum)

{

tree[sum].data=x;

tree[sum].fa=0;

tree[sum].c[0]=0;

tree[sum].c[1]=0;

tree[sum].num=1;

tree[sum].size=1;

tree[sum].rev=0;

}

void insert(int x,int sum)

{

int y=root;

int pos=0;

init(x,sum);

if (y==0)

{

root=sum;

return;

}

while (true)

{

tree[y].size++;

int p=1;

if (x<tree[y].data) p=0;

if (tree[y].c[p]==0)

{

tree[y].c[p]=sum;

tree[sum].fa=y;

break;

}

else y=tree[y].c[p];

}

if (pos==0) pos=sum;

splay(pos,0);

}

int findpos(int x)

{

int y=root;

downdate(y);

while (x<=tree[tree[y].c[0]].size||x>tree[tree[y].c[0]].size+tree[y].num)

{

if (x<=tree[tree[y].c[0]].size) y=tree[y].c[0];

else

{

x-=tree[tree[y].c[0]].size+tree[y].num;

y=tree[y].c[1];

}

downdate(y);

}

return y;

}

int find(int x,int p)

{

x=findpos(x);

splay(x,0);

int y=root;

downdate(y);

y=tree[y].c[p];

while (tree[y].c[p^1])

y=tree[y].c[p^1],downdate(y);

return y;

}

void print(int x)

{

downdate(x);

if (tree[x].c[0]) print(tree[x].c[0]);

if (x!=1&&x!=n+2)

{

num++;

printf("%d",tree[x].data);

if (num!=n) printf(" ");

}

if (tree[x].c[1]) print(tree[x].c[1]);

}

int main()

{

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

for (int i=1;i<=n+2;i++)

insert(i-1,++sum);

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

{

int x,y;

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

int fx=find(x+1,0),fy=find(y+1,1);

splay(fx,0);

splay(fy,1);

tree[tree[fy].c[0]].rev^=1;

}

print(root);

return 0;

}

Treap

#include<cstdio>

#include<cstring>

#include<ctime>

#include<iostream>

#include<algorithm>

#include<cstdlib>

#include<cmath>

#include<ctime>

#define MAXN 1000005

using namespace std;

const int MAXINF=1e9;

int tcnt;

struct Treap

{

int root,key[MAXN],pri[MAXN],c[MAXN][2],cnt[MAXN],size[MAXN];

void init()

{

root=0;

pri[0]=MAXINF;

size[0]=0;

}

void update(int x)

{

size[x]=size[c[x][0]]+cnt[x]+size[c[x][1]];

}

void rotate(int &x,int p)

{

int y=c[x][p];

c[x][p]=c[y][1-p];

c[y][1-p]=x;

update(x);

update(y);

x=y;

}

void insert(int &x,int k)

{

if (x)

{

if (key[x]==k) cnt[x]++;

else

{

int p=key[x]<k;

insert(c[x][p],k);

if (pri[c[x][p]]<pri[x]) rotate(x,p);

}

}

else

{

x=++tcnt;

key[x]=k;

cnt[x]=1;

pri[x]=rand();

c[x][0]=c[x][1]=0;

}

update(x);

}

void erase(int &x,int k)

{

if (key[x]==k)

{

if (cnt[x]>1) cnt[x]--;

else

{

if (c[x][0]==0&&c[x][1]==0)

{

x=0;

return;

}

int p=pri[c[x][0]]>pri[c[x][1]];

rotate(x,p);

erase(x,k);

}

}

else erase(c[x][key[x]<k],k);

update(x);

}

int getKth(int &x,int k)

{

if (k<=size[c[x][0]]) return getKth(c[x][0],k);

k-=size[c[x][0]]+cnt[x];

if (k<=0) return key[x];

return getKth(c[x][1],k);

}

}T;

int main()

{

int n,m;

while(scanf("%d %d",&n,&m)==2)

{

tcnt=0;

T.init();

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

{

char x;

int y;

scanf("%c",&x);

while (x!='Q'&&x!='I')

scanf("%c",&x);

if (x=='I') scanf("%d",&y),T.insert(T.root,y);

else printf("%d\n",T.getKth(T.root,T.size[T.root]-m+1));

}

}

return 0;

}

Valuesplay

#include<cstdio>

#include<iostream>

#include<algorithm>

#include<cmath>

#include<cstring>

#include<cstdlib>

#include<set>

#include<map>

#include<stack>

#include<queue>

#include<vector>

using namespace std;

const int maxn=110000;

struct node

{

int data,num,size,c[2],fa;

}tree[maxn];

int root=0,sum=0,res;

void update(int x)

{

tree[x].size=tree[tree[x].c[0]].size+tree[tree[x].c[1]].size+tree[x].num;

}

void rotate(int x,int p)

{

int y=tree[x].fa;

if (tree[x].c[p]) tree[tree[x].c[p]].fa=y;

tree[y].c[p^1]=tree[x].c[p];

tree[x].c[p]=y;

tree[x].fa=tree[y].fa;

tree[y].fa=x;

if (tree[x].fa) if (y==tree[tree[x].fa].c[0]) tree[tree[x].fa].c[0]=x;

else tree[tree[x].fa].c[1]=x;

update(y);

update(x);

}

void splay(int x)

{

while (tree[x].fa!=0)

{

int y=tree[x].fa;

int z=tree[y].fa;

if (z==0)

{

if (tree[y].c[0]==x) rotate(x,1);

else rotate(x,0);

break;

}

int a=y==tree[z].c[0]?1:0;

int b=x==tree[y].c[0]?1:0;

if (a^b) rotate(x,b),rotate(x,a);

else rotate(y,a),rotate(x,b);

}

root=x;

}

void maintain(int x)

{

while (x!=0)

update(x),x=tree[x].fa;

}

void init(int x,int sum)

{

tree[sum].data=x;

tree[sum].fa=0;

tree[sum].c[0]=0;

tree[sum].c[1]=0;

tree[sum].num=1;

tree[sum].size=1;

}

void insert(int x,int sum)

{

int y=root;

int pos=0;

init(x,sum);

if (y==0)

{

root=sum;

return;

}

while (true)

{

tree[y].size++;

if (x==tree[y].data)

{

pos=y;

tree[y].num++;

maintain(y);

break;

}

int p=1;

if (x<tree[y].data) p=0;

if (tree[y].c[p]==0)

{

tree[y].c[p]=sum;

tree[sum].fa=y;

break;

}

else y=tree[y].c[p];

}

if (pos==0) pos=sum;

splay(pos);

}

int findpos(int x,int z)

{

int p,y=root,d=0;

while (x!=tree[y].data)

{

p=0;

if (x>tree[y].data) p=1;

if (p&&z) d+=tree[tree[y].c[0]].size+tree[y].num;

y=tree[y].c[p];

}

if (z) return d+tree[tree[y].c[0]].size+1;

else return y;

}

int find(int x,int p)

{

x=findpos(x,0);

splay(x);

int y=root;

y=tree[y].c[p];

while (tree[y].c[p^1])

y=tree[y].c[p^1];

return y;

}

void del(int x)

{

int pos=findpos(x,0);

splay(pos);

if (tree[pos].num>1)

{

tree[pos].num--;

tree[pos].size--;

}

else

{

if (tree[pos].c[0]!=0)

{

tree[tree[pos].c[0]].fa=0;

int f=find(tree[pos].data,0);

tree[tree[pos].c[1]].fa=f;

tree[f].c[1]=tree[pos].c[1];

maintain(tree[pos].c[1]);

root=tree[pos].c[0];

tree[pos].c[0]=0;

tree[pos].c[1]=0;

}

else

{

tree[tree[pos].c[1]].fa=0;

root=tree[pos].c[1];

}

}

}

int finddata(int x)

{

int y=root;

while (x<=tree[tree[y].c[0]].size||x>tree[tree[y].c[0]].size+tree[y].num)

{

if (x<=tree[tree[y].c[0]].size) y=tree[y].c[0];

else

{

x-=tree[tree[y].c[0]].size+tree[y].num;

y=tree[y].c[1];

}

}

return tree[y].data;

}

int main()

{

int n,op,y;

scanf("%d",&n);

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

{

scanf("%d%d",&op,&y);

switch(op)

{

case 1:insert(y,++sum);break;

case 2:del(y);break;

case 3:printf("%d\n",findpos(y,1));break;

case 4:printf("%d\n",finddata(y));break;

case 5:insert(y,++sum);printf("%d\n",tree[find(y,0)].data);del(y);break;

case 6:insert(y,++sum);printf("%d\n",tree[find(y,1)].data);del(y);break;

}

}

return 0;

}

并查集

#include<cstdio>

#include<iostream>

#include<algorithm>

#include<cmath>

#include<cstring>

#include<cstdlib>

#include<set>

#include<map>

#include<stack>

#include<queue>

#include<vector>

using namespace std;

const int maxn=110000;

struct node

{

int data,num,size,c[2],fa;

}tree[maxn];

int root=0,sum=0,res;

void update(int x)

{

tree[x].size=tree[tree[x].c[0]].size+tree[tree[x].c[1]].size+tree[x].num;

}

void rotate(int x,int p)

{

int y=tree[x].fa;

if (tree[x].c[p]) tree[tree[x].c[p]].fa=y;

tree[y].c[p^1]=tree[x].c[p];

tree[x].c[p]=y;

tree[x].fa=tree[y].fa;

tree[y].fa=x;

if (tree[x].fa) if (y==tree[tree[x].fa].c[0]) tree[tree[x].fa].c[0]=x;

else tree[tree[x].fa].c[1]=x;

update(y);

update(x);

}

void splay(int x)

{

while (tree[x].fa!=0)

{

int y=tree[x].fa;

int z=tree[y].fa;

if (z==0)

{

if (tree[y].c[0]==x) rotate(x,1);

else rotate(x,0);

break;

}

int a=y==tree[z].c[0]?1:0;

int b=x==tree[y].c[0]?1:0;

if (a^b) rotate(x,b),rotate(x,a);

else rotate(y,a),rotate(x,b);

}

root=x;

}

void maintain(int x)

{

while (x!=0)

update(x),x=tree[x].fa;

}

void init(int x,int sum)

{

tree[sum].data=x;

tree[sum].fa=0;

tree[sum].c[0]=0;

tree[sum].c[1]=0;

tree[sum].num=1;

tree[sum].size=1;

}

void insert(int x,int sum)

{

int y=root;

int pos=0;

init(x,sum);

if (y==0)

{

root=sum;

return;

}

while (true)

{

tree[y].size++;

if (x==tree[y].data)

{

pos=y;

tree[y].num++;

maintain(y);

break;

}

int p=1;

if (x<tree[y].data) p=0;

if (tree[y].c[p]==0)

{

tree[y].c[p]=sum;

tree[sum].fa=y;

break;

}

else y=tree[y].c[p];

}

if (pos==0) pos=sum;

splay(pos);

}

int findpos(int x,int z)

{

int p,y=root,d=0;

while (x!=tree[y].data)

{

p=0;

if (x>tree[y].data) p=1;

if (p&&z) d+=tree[tree[y].c[0]].size+tree[y].num;

y=tree[y].c[p];

}

if (z) return d+tree[tree[y].c[0]].size+1;

else return y;

}

int find(int x,int p)

{

x=findpos(x,0);

splay(x);

int y=root;

y=tree[y].c[p];

while (tree[y].c[p^1])

y=tree[y].c[p^1];

return y;

}

void del(int x)

{

int pos=findpos(x,0);

splay(pos);

if (tree[pos].num>1)

{

tree[pos].num--;

tree[pos].size--;

}

else

{

if (tree[pos].c[0]!=0)

{

tree[tree[pos].c[0]].fa=0;

int f=find(tree[pos].data,0);

tree[tree[pos].c[1]].fa=f;

tree[f].c[1]=tree[pos].c[1];

maintain(tree[pos].c[1]);

root=tree[pos].c[0];

tree[pos].c[0]=0;

tree[pos].c[1]=0;

}

else

{

tree[tree[pos].c[1]].fa=0;

root=tree[pos].c[1];

}

}

}

int finddata(int x)

{

int y=root;

while (x<=tree[tree[y].c[0]].size||x>tree[tree[y].c[0]].size+tree[y].num)

{

if (x<=tree[tree[y].c[0]].size) y=tree[y].c[0];

else

{

x-=tree[tree[y].c[0]].size+tree[y].num;

y=tree[y].c[1];

}

}

return tree[y].data;

}

int main()

{

int n,op,y;

scanf("%d",&n);

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

{

scanf("%d%d",&op,&y);

switch(op)

{

case 1:insert(y,++sum);break;

case 2:del(y);break;

case 3:printf("%d\n",findpos(y,1));break;

case 4:printf("%d\n",finddata(y));break;

case 5:insert(y,++sum);printf("%d\n",tree[find(y,0)].data);del(y);break;

case 6:insert(y,++sum);printf("%d\n",tree[find(y,1)].data);del(y);break;

}

}

return 0;

}

数学

FFT

const double PI = acos(-1.0);

struct complex

{

double r,i;

complex(double \_r = 0.0,double \_i = 0.0){r = \_r; i = \_i;}

complex operator +(const complex &b){return complex(r+b.r,i+b.i);}

complex operator -(const complex &b){return complex(r-b.r,i-b.i);}

complex operator \*(const complex &b){return complex(r\*b.r-i\*b.i,r\*b.i+i\*b.r);}

};

void change(complex y[],int len)

{

int i,j,k;

for(i=1,j=len/2;i<len-1;i++)

{

if(i<j)swap(y[i],y[j]);

k=len/2;

while(j>=k)

{

j-=k;

k/=2;

}

if(j<k) j+=k;

}

}

void fft(complex y[],int len,int on)

{

change(y,len);

for(int h = 2; h <= len; h <<= 1)

{

complex wn(cos(on\*2\*PI/h),sin(on\*2\*PI/h));

for(int j = 0;j < len;j+=h)

{

complex w(1,0);

for(int k = j;k < j+h/2;k++)

{

complex u = y[k], t = w\*y[k+h/2];

y[k] = u+t;

y[k+h/2] = u-t;

w = w\*wn;

}

}

}

if(on == -1)

for(int i = 0;i < len;i++)

y[i].r /= len;

}

int main()

{

for(int i=0;i<na;i++)

x1[i]=complex(s1[na-1-i]-'0',0);

for(int i=na;i<N;i++)

x1[i]=complex(0,0);

for(int i=0;i<nb;i++)

x2[i]=complex(s2[nb-1-i]-'0',0);

for(int i=nb;i<N;i++)

x2[i]=complex(0,0);

fft(x1,N,1);

fft(x2,N,1);

for(int i=0;i<N;i++)

x1[i]=x1[i]\*x2[i];

fft(x1,N,-1);

}

扩展GCD

LL extended\_gcd(LL a,LL b,LL &x,LL &y)

{

if (b==0)

{

x=1;

y=0;

return a;

}

else

{

LL r=extended\_gcd(b,a%b,y,x);

y-=x\*(a/b);

return r;

}

}

扩展LUCAS

LL mod,a[110000],b[110000];

int n,m,v[110000],p[110000],s[100],tot,num;

struct divi

{

int p,c,pc;

}w[110000];

void prepare()

{

int cnt=0;

for (int i=2;i<=100000;i++)

if (!v[i])

{

p[++cnt]=i;

for (int j=i;j<=100000;j+=i)

v[j]=1;

}

num=0;

LL x=mod;

for (int i=1;i<=cnt;i++)

{

if (x<=1) break;

if (x%p[i]) continue;

w[++num].p=p[i];

w[num].pc=1;

while (x%p[i]==0)

w[num].c++,x/=p[i],w[num].pc\*=p[i];

b[num]=w[num].pc;

}

}

LL pow\_mod(LL a,LL i,LL n)

{

if (i==0) return 1%n;

LL temp=pow\_mod(a,i>>1,n);

temp=temp\*temp%n;

if (i&1) temp=(LL) temp\*a%n;

return temp;

}

LL extended\_gcd(LL a,LL b,LL &x,LL &y)

{

if (b==0)

{

x=1;

y=0;

return a;

}

else

{

LL r=extended\_gcd(b,a%b,y,x);

y-=x\*(a/b);

return r;

}

}

LL inv(LL a,LL n)

{

LL xx,yy,d=extended\_gcd(a,n,xx,yy);

return (xx%n+n)%n;

}

pa fac(int k,LL n)

{

if (n==0) return mp(0,1);

int x=n/w[k].p,y=n/w[k].pc;

LL ans=1;

if (y)

{

for (int i=2;i<w[k].pc;i++)

if (i%w[k].p!=0)

ans=ans\*1LL\*i%w[k].pc;

ans=pow\_mod(ans,y,w[k].pc);

}

for (int i=y\*w[k].pc+1;i<=n;i++)

if (i%w[k].p!=0)

ans=ans\*1LL\*i%mod;

pa p=fac(k,x);

return mp(x+p.fi,ans\*p.se%mod);

}

LL get(int k,LL n,LL m)

{

if (n<m) return 0;

pa a=fac(k,n),b=fac(k,m),c=fac(k,n-m);

LL mod=w[k].pc;

return pow\_mod(w[k].p,a.fi-b.fi-c.fi,mod)\*

a.se%mod\*inv(b.se,mod)%mod\*

inv(c.se,mod)%mod;

}

LL crt(LL a[],LL b[],int num,LL mod)

{

LL M=mod,d,y,x=0;

for (int i=1;i<=num;i++)

{

LL r=M/b[i];

d=extended\_gcd(w[i].pc,r,d,y);

x=(x+r\*y\*a[i])%M;

}

return (x+M)%M;

}

LL C(int n,int m)

{

for (int i=1;i<=num;i++)

a[i]=get(i,n,m);

return (crt(a,b,num,mod))%mod;

}

扩展CRT

int extended\_crt(int n,LL m[],LL r[])

{

LL m1,r1,m2,r2,d,x,y,c,t;

int flag=0;

m1=m[1],r1=r[1];

flag=0;

for(int i=2;i<=n;i++)

{

m2=m[i],r2=r[i];

d=extended\_gcd(m1,m2,x,y);

c=r2-r1;

if(c%d)

{

flag=1;

break;

}

t=m2/d;

x=(c/d\*x%t+t)%t;

r1=m1\*x+r1;

m1=m1\*m2/d;

}

if(flag)return -1;

else return r1;

}

扩展bsgs

LL extended\_bsgs(LL a,LL b,LL n)

{

LL t,c=0,v=1;

while ((t=gcd(a,n))!=1)

{

if (b%t)

return -1;

n/=t;

b/=t;

v=v\*a/t%n;

c++;

if (b==v) return c;

}

LL x,y;

extended\_gcd(v,n,x,y);

b=b\*x%n;

LL ret=bsgs(a,b,n);

return ~ret ? ret+c:ret;

}

欧拉函数欧拉定理

int minDiv[maxn],phi[maxn];

void getPhi()

{

for (int i=1;i<maxn;i++)

minDiv[i]=i;

for (int i=2;i\*i<maxn;i++)

{

if (minDiv[i]==i)

{

for (int j=i\*i;j<max;j+=i)

minDiv[j]=i;

}

}

phi[1]=1;

for (int i=2;i<max;i++)

{

phi[i]=phi[i/minDiv[i]];

if ((i/minDiv[i])%minDiv[i]==0) phi[i]\*=minDiv[i];

else phi[i]\*=minDiv[i]-1;

}

}

LL phi(LL n)

{

LL i;

LL result = n;

LL t = (LL)sqrt(n\*1.0);

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

{

if(n%i==0)

{

result = result/i\*(i-1);

while(n%i==0)

n = n/i;

}

}

if(n>1)

result = result/n\*(n-1);

return result;

}

LL phi(LL n)

{

LL res=n;

for(int i=1;prim[i]\*prim[i]<=n&&i<=primm;i++)

{

if(n%prim[i]==0)

{

res=res-res/prim[i];

while(n%prim[i]==0) n/=prim[i];

}

}

if(n>1)

res=res-res/n;

return res%p;

}

CRT

LL crt(LL a[],LL b[],int num,LL mod)

{

LL M=mod,d,y,x=0;

for (int i=1;i<=num;i++)

{

LL r=M/b[i];

d=extended\_gcd(w[i].pc,r,d,y);

x=(x+r\*y\*a[i])%M;

}

return (x+M)%M;

}

BSGS

long long bsgs(int x,int n,int m)

{

map<long long,int> rec;

int s=(int)(sqrt((double)m));

for(;(long long)s\*s <= m;)

s++;

long long cur=x%m;

rec[1]=0;

for (int i=1;i<s;i++)

{

if (rec[cur]==0&&cur!=1) rec[cur]=i;

cur=(cur\*x)%m;

}

long long mul=cur;

cur=1;

for (int i=0;i<s;i++)

{

long long more=(long long)n\*pow\_mod(cur,m-2,m)%m;

if (rec.count(more)) return i\*s+rec[more];

cur=(cur\*mul)%m;

}

return -1;

}

组合

{

C[0][0]=1;

for(int i=1;i<=31;i++){

C[i][0]=1;

for (int j=1;j<=i;j++)

C[i][j]=C[i-1][j]+C[i-1][j-1];

}

}

LL C(int n,int m)

{

LL res=(f[n]\*inv[m])%mod;

res=(res\*inv[n-m])%mod;

return res;

}

{

f[0]=1;

for (int i=1;i<=1000010;i++)

f[i]=(f[i-1]\*i)%mod;

ff[1]=ff[0]=inv[1]=inv[0]=1;

for (int i=2;i<=1000010;i++)

{

inv[i]=(LL)(mod-mod/i)\*inv[mod%i]%mod;

ff[i]=inv[i];

}

for (int i=2;i<=1000010;i++)

inv[i]=(inv[i-1]\*inv[i])%mod;

}

自适应幸普森

template<class T>

double simpson(const T &f,double a,double b,int n)

{

const double h=(b-a)/n;

double ans=f(a)+f(b);

for (int i=1;i<n;i+=2) ans+=4\*f(a+i\*h);

for (int i=2;i<n;i+=2) ans+=2\*f(a+i\*h);

return ans\*h/3;

}

线性筛逆元

inv[1]=1;

for (int i=2;i<=1000000;i++)

inv[i]=(LL)(mod-mod/i)\*inv[mod%i]%mod;

素数筛

#define maxn 1000000

bool valid[maxn];

void getPrime(int n,int &tot,int ans[maxn]) //n为筛素数的范围,tot为素数总数,ans[]为prim数组

{

memset(valid,true,sizeof(valid));

for (int i=2;i<=n;i++)

{

if (valid[i])

{

tot++;

ans[tot]=i;

for (int j=2;j<=n/i;j++)

valid[i\*j]=0;

}

}

}

求原根

LL findprimitiveroot(LL x)

{

for (LL i=2;i<=x;i++)

{

LL flag=0,now=1;

for (int j=1;j<=x-1;j++)

{

now=(now\*LL(i))%x;

if (now==1)

{

flag=1;

break;

}

}

if (flag==0) return i;

}

}

奇素数原根

#include<cstdio>

#include<cstring>

#include<iostream>

#include<algorithm>

using namespace std;

#define LL long long

const int maxn=1000100;

bool valid[maxn];

int prim[maxn],pr[maxn];

int primm=0,c; //c为n质因子的个数

void getPrime(int n,int &tot,int ans[maxn]) //n为筛素数的范围,tot为素数总数,ans[]为prim数组

{

memset(valid,true,sizeof(valid));

for (int i=2;i<=n;i++)

{

if (valid[i])

{

tot++;

ans[tot]=i;

for (int j=2;j<=n/i;j++)

valid[i\*j]=0;

}

}

}

void cal(LL n)

{

LL t=n,a;

c=0;

for (int i=0;prim[i]\*prim[i]<=n;i++)

{

if (n%prim[i]==0)

{

pr[c]=prim[i];

while (n%prim[i]==0) n/=prim[i];

c++;

}

}

if (n>1)

{

pr[c]=n;

c++;

}

}

LL quick\_mod(LL a,LL b,LL m)

{

LL ans=1;

a%=m;

while (b)

{

if (b&1)

{

ans=ans\*a%m;

b--;

}

b>>=1;

a=a\*a%m;

}

return ans;

}

int main()

{

LL P,t,g,root;

getPrime(maxn-1,primm,prim);

while (cin>>P)

{

cal(P-1);

for (g=2;g<P;g++)

{

bool flag=true;

for (int i=0;i<c;i++)

{

t=(P-1)/pr[i];

if (quick\_mod(g,t,P)==1)

{

flag=false;

break;

}

}

if (flag)

{

root=g;

cout<<root<<endl;

}

}

}

return 0;

}

quadratic remain

int modsqr(int a,int n)

{

int b,k,i,x;

if (n==2) return a%n;

if (pow\_mod(a,(n-1)/2,n)==1)

{

if (n%4==3) x=pow\_mod(a,(n+1)/4,n);

else

{

for (b=1;pow\_mod(b,(n-1)/2,n)==1;b++);

i=(n-1)/2;

k=0;

do

{

i/=2;

k/=2;

if ((pow\_mod(a,i,n)\*(long long)pow\_mod(b,k,n)+1)%n==0) k+=(n-1)/2;

}

while (i%2==0)

x=(pow\_mod(a,(i+1)/2,n)\*(long long)pow\_mod(b,k/2,n))%n;

}

if (x\*2>n) x=n-x;

return x;

}

return -1;

}

Pow\_mod

LL pow\_mod(LL a,LL b)

{

LL ret=1;

while(b)

{

if(b&1) ret=multi(ret,a)%p;

a=multi(a,a)%p;

b>>=1;

}

return ret;

}

NNT

typedef long long LL;

const int maxn=200000;

const LL P=1004535809; //P=C\*2^k+1，P是质数

const LL g=3; //P的原根

int N,na,nb;

int a[maxn\*2],b[maxn\*2],W[2][maxn\*2],rev[maxn\*2];

LL Pow(LL a,int b)

{

LL c=1;

for (;b; b>>=1,a=a\*a%P) if (b&1) c=c\*a%P;

return c;

}

void NTT(int\*a,int f)

{

for (int i=0;i<N;i++) if (i<rev[i]) swap(a[i],a[rev[i]]);

for (int i=1;i<N;i<<=1)

for (int j=0,t=N/(i<<1);j<N;j+=i<<1)

for (int k=0,l=0,x,y;k<i;k++,l+=t) x=(LL)W[f][l]\*a[j+k+i]%P,y=a[j+k],a[j+k]=(y+x)%P,a[j+k+i]=(y-x+P)%P;

if (f) for (int i=0,x=Pow(N,P-2);i<N;i++) a[i]=(LL)a[i]\*x%P;

}

void change()

{

for (int i=0;i<N;i++)

{

int x=i,y=0;

for (int k=1; k<N; x>>=1,k<<=1) (y<<=1)|=x&1;

rev[i]=y;

}

}

int main()

{

for (N=1; N<na||N<nb; N<<=1); N<<=1;

for (int i=na;i<N;i++)

a[i]=b[i]=0;

W[0][0]=W[1][0]=1;

for (int i=1,x=Pow(g,(P-1)/N),y=Pow(x,P-2); i<N; i++)

W[0][i]=(LL)x\*W[0][i-1]%P,W[1][i]=(LL)y\*W[1][i-1]%P;

change();

NTT(a,0),NTT(b,0);

for (int i=0;i<N;i++) a[i]=(LL)a[i]\*b[i]%P;

NTT(a,1);

}

快速乘

LL multi(LL a, LL b)

{

LL ret=0;

while(b)

{

if(b&1) ret=(ret+a)%p;

a=(a+a)%p;

b>>=1;

}

return ret;

}

莫比乌斯反演

void mobius()

{

memset(vis,0,sizeof(vis);

mu[1]=1;

cnt=0;

for (int i=2;i<N;i++)

{

if (!vis[i])

{

prime[++cnt]=i;

mu[i]=-1;

}

for (int j=0;j<cnt&&i\*prime[j]<N;j++)

{

vis[i\*prime[j]]=1;

if (i%prime[j]) mu[i\*prime[j]]=-mu[i];

else

{

mu[i\*prime[j]]=0;

break;

}

}

}

}

Miller-Robin

bool test(int n,int a,int d)

{

if (n==2) return true;

if (n==a) return true;

if ((n&1)==0) return false;

while (!(d&1)) d=d>>1;

int t=pow\_mod(a,d,n);

while ((d!=n-1)&&(t!=1)&&(t!=n-1))

{

t=(long long) t\*t%n;

d=d<<1;

}

return(t==n-1||(d&1)==1);

}

bool isPrime(int n)

{

if (n<2) return false;

int a[3]={2,3,61};

for (int i=0;i<=2;i++)

if (!test(n,a[i],n-1)) return false;

return true;

}

void rho(LL n)

{

int flag=0;

while (true)

{

LL c=rand()%n+1;

LL x=rand()%n+1;

LL y=x;

if (isPrime(n))

{

num++;

a[num]=n;

return;

}

LL k=2;

LL i=0;

while (true)

{

i++;

LL d=gcd(n+y-x,n);

if (d>1&&d<n)

{

flag=1;

if (isPrime(d))

{

num++;

a[num]=d;

}

else rho(d);

d=n/d;

if (isPrime(d))

{

num++;

a[num]=d;

}

else rho(d);

}

if (i==k)

{

y=x;

k\*=2;

}

x=(pow\_mod(x,x,n)+n-c)%n;

if (y==x) break;

}

}

}

矩阵类

const int mod=9973;

const int MAXN=20;

const int MAXM=20;

struct Matrix

{

int n,m;

int a[MAXN][MAXM];

void clear()

{

n=m=0;

memset(a,0,sizeof(a));

}

Matrix operator +(const Matrix &b) const

{

Matrix tmp;

tmp.n=n; tmp.m=m;

for (int i=0;i<n;i++)

for (int j=0;j<m;j++)

tmp.a[i][j]=a[i][j]+b.a[i][j];

return tmp;

}

Matrix operator -(const Matrix &b) const

{

Matrix tmp;

tmp.n=n; tmp.m=m;

for (int i=0;i<n;i++)

for (int j=0;j<m;j++)

tmp.a[i][j]=a[i][j]-b.a[i][j];

return tmp;

}

Matrix operator \*(const Matrix &b) const

{

Matrix tmp;

tmp.clear();

tmp.n=n;

tmp.m=b.m;

for (int i=0;i<n;i++)

for (int j=0;j<b.m;j++)

for (int k=0;k<m;k++)

tmp.a[i][j]+=a[i][k]\*b.a[k][j];

return tmp;

}

Matrix operator %(const int b) const

{

Matrix tmp;

tmp.clear();

tmp.n=n;

tmp.m=m;

for (int i=0;i<n;i++)

for (int j=0;j<m;j++)

tmp.a[i][j]=((abs(a[i][j])/b+1)\*b+a[i][j])%b;

return tmp;

}

};

Matrix pow\_mod(Matrix a,LL i,LL n)

{

if (i==0)

{

Matrix b;

b.n=b.m=a.n;

for (int j=0;j<b.n;j++)

for (int k=0;k<b.n;k++)

if (j!=k) b.a[j][k]=0;

else b.a[j][k]=1;

return b;

}

Matrix temp=pow\_mod(a,i>>1,n);

temp=temp\*temp%n;

if (i&1) temp=temp\*a%n;

return temp;

}

Inverse matrix

inline vector<double> operator \* (vector<double> a, double b)

{

int N=a.size();

vector<double> res(N,0);

for (int i=0;i<N;i++)

res[i]=a[i]\*b[i];

return res;

}

inline vector<double> operator - (vector<double> a,vector<double> b)

{

int N=a.size();

vector<double> res(N,0);

for (int i=0;i<N;i++)

res[i]=a[i]-b[i];

return res;

}

inline void inverse(vector<double> A[], vector<double> C[],int N)

{

for (int i=0;i<N;i++)

C[i]=vector<double>(N,0);

for (int i=0;i<N;i++)

C[i][i]=1;

for (int i=0;i<N;i++)

{

for (int j=i;j<N;j++)

if (fabs(A[j][i]>0)

{

swap(A[i],A[j]);

swap(C[i],C[j]);

break;

}

C[i]=C[i]\*(1/A[i][i]);

A[i]=A[i]\*(1/A[i][i]);

for (int j=0;j<N;j++)

if (j!=i&&fabs(A[j][i]>0))

{

C[j]=C[j]-C[i]\*A[j][i];

A[j]=A[j]-A[i]\*A[j][i];

}

}

}

INV

LL inv(LL a,LL n)

{

LL xx,yy,d=extended\_gcd(a,n,xx,yy);

return (xx%n+n)%n;

}

高斯消元

const double EPS=1E-12;

const double inf=1E+12;

inline int sign(double x)

{

return x<-EPS?-1:x>EPS;

}

inline double get(const vector<double>&coef,double x)

{

double e=1,s=0;

for (int i=0;i<coef.size();i++)

s+=coef[i]\*e,e\*=x;

return s;

}

double find(const vector<double>&coef,int n,double lo,double hi)

{

double sign\_lo,sign\_hi;

if ((sign\_lo=sign(get(coef,lo)))==0) return lo;

if ((sign\_hi=sign(get(coef,hi)))==0) return hi;

if (sign\_lo\*sign\_hi>0) return inf;

for (int step=0;step<100 && hi-lo>EPS;step++)

{

double m=(lo+hi)\*.5;

int sign\_mid=sign(get(coef,m));

if (sign\_mid==0) return m;

if (sign\_lo\*sign\_mid<0) hi=m;

else lo=m;

}

return (lo+hi)\*.5;

}

vector <double>equation(vector<double> coef,int n)

{

vector<double> ret;

if (n==1)

{

if (sign(coef[1])) ret.push\_back(-coef[0]/coef[1]);

return ret;

}

vector<double> dcoef(n);

for (int i=0;i<n;i++)

dcoef[i]=coef[i+1]\*(i+1);

vector<double> droot=solve(dcoef,n-1);

droot.insert(droot.begin(),-inf);

droot.push\_back(inf);

for (int i=0;i+1<droot.size();i++)

{

double tmp=find(coef,n,droot[i],droot[i+1]);

if (tmp<inf) ret.push\_back(tmp);

}

return ret;

}

取素数

#define maxn 1000000

bool valid[maxn];

int primm;

LL prim[maxn];

void getPrime(int n)

{

memset(valid,true,sizeof(valid));

for (int i=2;i<=n;i++)

{

if (valid[i])

{

primm++;

prim[primm]=i;

for (int j=2;j<=n/i;j++)

valid[i\*j]=0;

}

}

}

Gauss

const double EPS=1e-12;

inline int solve(double a[][MAXN],bool l[],double ans[],const int &n)

{

int res=0,r=0;

for (int i=0;i<n;i++)

l[i]=false;

for (int i=0;i<n;i++)

{

for (int j=r;j<n;j++)

if (fabs(a[i][j])>EPS)

{

for (int k=i;i<=n;k++)

swap(a[j][k],a[r][k]);

break;

}

if (fabs(a[r][i])<EPS)

{

res++;

continue;

}

for (int j=0;j<n;j++)

if (j!=r&&fabs(a[i][j])>EPS)

{

double tmp=a[j][i]/a[r][i];

for (int k=i;k<=n;k++)

a[j][k]-=tmp\*a[r][k];

}

l[i]=true;

r++;

}

for (int i=0;i<n;i++)

if (l[i])

for (int j=0;j<n;j++)

if (fabs(a[j][i])>0)

ans[i]=a[j][n]/a[j][i];

return res;

}

组合游戏

//威佐夫博弈

int main()

{

double x=(1+sqrt(5.0))/2.0;

int k,temp,n,m;

while(scanf("%d%d",&a,&b)!=EOF&&(a+b)){

if(a>b){

temp = a;

a=b;

b=temp;

}

k=b-a;

if((int)(k\*x)==a)

printf("%d\n",0);

else{

printf("%d\n",1);

for(int i=1;i<=a;i++){

n=a-i;

m=b-i;

k=m-n;

if((int)(k\*x)==n)

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

}

for(int i=b-1;i>=0;i--){

n=a;

m=i;

if(n>m) swap(n,m);

k=m-n;

if((int)(k\*x)==n)

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

}

}

}

return 0;

}

//k倍动态博弈

int a[2000000],b[2000000];

int main()

{

int n,k;

int T=0,cas=0;

cin>>T;

while (T--){

scanf("%d %d",&n,&k);

a[0]=b[0]=1;

int i=0,j=0;

while (n>a[i]){

i++;

a[i]=b[i-1]+1;

while (a[j+1]\*k<a[i]) j++;

if (k\*a[j]<a[i]) b[i]=b[j]+a[i];

else b[i]=a[i];

}

cas++;

printf("Case %d: ",cas);

if (n==a[i]) printf("lose\n");

else{

int ans;

while (n){

if (n>=a[i]){

n-=a[i];

ans=a[i];

}

i--;

}

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

}

}

return 0;

}

//every-SG

int main()

{

int n;

while(scanf("%d",&n)!=EOF){

int ans=0;

while(n--){

int p,q,len=2,f[100];

scanf("%d%d",&p,&q);

if(p<q) swap(p,q);

f[1]=p;f[2]=q;

while(f[len]){

f[len+1]=f[len-1]%f[len];

len++;

}

int k=len-2,j=-1;

for(int i=1;i<len-1;i++){

if(f[i]/f[i+1]>1){

if(j>0&&i%2!=j%2)

k++;

j=i;

}

}

ans=max(ans,k);

}

if(ans&1) printf("MM\n");

else printf("GG\n");

}

return 0;

}

//NIM积

int f(int x,int y);

int g(int x,int y)

{

if (gg[x][y]!=-1) return gg[x][y];

if (!x) return gg[x][y]=1<<y;

if (!y) return gg[x][y]=1<<x;

int k=1,ans=1,t,x1=x,y1=y,x2=x,y2=y;

while (x||y){

t=1<<k;

if ((x&1||y&1)&&!((x&1)&&(y&1))) ans\*=t;

k<<=1;

x>>=1,y>>=1;

}

k=1;

while (x1||y1){

t=1<<k;

if ((x1&1)&&(y1&1)) ans=f(ans,t/2\*3);

k<<=1;

x1>>=1,y1>>=1;

}

return (gg[x2][y2]=ans);

}

int f(int x,int y)

{

if (!x||!y) return 0;

if (x==1) return y;

if (y==1) return x;

int ans=0;

for (int i=x,k1=0;i;i>>=1,k1++)

if (i&1) for (int j=y,k2=0;j;j>>=1,k2++)

if (j&1) ans^=g(k1,k2);

return ans;

}

//Anti-Nim

int n;

int main()

{

int T;

cin>>T;

while (T--){

scanf("%d",&n);

int flag=0,s=0,j=0;

for (int i=0;i<n;i++){

int x;

scanf("%d",&x);

s^=x;

if (x>1) flag=1;

if (x==0) j++;

}

if (flag==0) if ((n-j)%2) printf("Brother\n");

else printf("John\n");

else if (s==0) printf("Brother\n");

else printf("John\n");

}

return 0;

}

分数类

struct Fraction

{

LL num,den;

Fraction(LL num=0,LL den=1)

{

if (den<0)

{

num=-num;

den=-den;

}

LL g=gcd(abs(num),den);

this->num=num/g;

this->den=den/g;

}

Fraction operator +(const Fraction &o) const

{

return Fraction(num\*o.den+den\*o.num,den\*o.den);

}

Fraction operator -(const Fraction &o) const

{

return Fraction(num\*o.den-den\*o.num,den\*o.den);

}

Fraction operator \*(const Fraction &o) const

{

return Fraction(num\*o.num,den\*o.den);

}

Fraction operator /(const Fraction &o) const

{

return Fraction(num\*o.den,den\*o.num);

}

bool operator <(const Fraction &o) const

{

return num\*o.den<den\*o.num;

}

bool operator >(const Fraction &o) const

{

return num\*o.den>den\*o.num;

}

bool operator ==(const Fraction &o) const

{

return num\*o.den==den\*o.num;

}

};

图论

最大流

int n,m;

int en;

int st,ed;

int dis[maxn] ;

int que[9999999];

struct edge

{

int to,c,next;

};

edge e[9999999];

int head[maxn];

void add(int a,int b,int c)

{

e[en].to=b;

e[en].c=c;

e[en].next=head[a];

head[a]=en++;

e[en].to=a;

e[en].c=0;

e[en].next=head[b];

head[b]=en++;

}

int bfs()

{

memset(dis,-1,sizeof(dis));

dis[st]=0;

int front=0,rear=0;

que[rear++]=st;

while(front<rear)

{

int j=que[front++];

for(int k=head[j];k!=-1;k=e[k].next)

{

int i=e[k].to;

if(dis[i]==-1&&e[k].c)

{

dis[i] = dis[j]+ 1 ;

que[rear++]=i;

if(i==ed) return true;

}

}

}

return false;

}

int dfs(int x,int mx)

{

int i,a;

if(x==ed) return mx ;

int ret=0;

for(int k=head[x];k!=-1&&ret<mx;k=e[k].next)

{

if(e[k].c&&dis[e[k].to]==dis[x]+1)

{

int dd=dfs(e[k].to,min(e[k].c,mx-ret));

e[k].c-=dd;

e[k^1].c+=dd;

ret+=dd;

}

}

if(!ret) dis[x]=-1;

return ret;

}

int dinic()

{

int tmp=0;

int maxflow=0;

while(bfs())

{

while(tmp=dfs(st,INF)) maxflow+=tmp;

}

return maxflow;

}

字符串

KMP

void getnext(int t[],int m)

{

int i,j;

i=0,j=1;

kmpnext[1]=0;

while (j<=m)

if (i==0||t[i]==t[j])

{ ++i,++j;

kmpnext[j]=i;

}

else i=kmpnext[i];

}

int kmp(int s[],int t[],int n,int m)

{

int i,j;

j=0;

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

{

while (j>0&&t[j+1]!=s[i])) j=kmpnext[j];

if (t[j+1]==s[i]) j++;

if (j==m)

{

return i-m+1;

}

}

return -1;

}

MANACHAR

int manacher()

{

int i,p,ans=0;

r[1]=0,p=1;

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

{ if (i<=p+r[p])

r[i]=min(r[2\*p-i],p+r[p]-i);

else r[i]=1;

while (st[i-r[i]]==st[i+r[i]])

++r[i];

--r[i];

if (i+r[i]>p+r[p])

p=i;

ans=max(ans,r[i]);

}

return ans;

}

TIRE

const int CHARSET=26,BASE='a',MAX\_NODE=100000;

struct Trie

{

int tot,root,child[MAX\_NODE][CHARSET];

bool flag[MAX\_NODE];

Trie()

{

memset(child[1],0,sizeof(child[1]));

flag[1]=false;

root=tot=1;

}

void insert(const char \*str)

{

int \*cur=&root;

for (const char \*p=str;\*p;++p)

{

cur=&child[\*cur][\*p-BASE];

if (\*cur==0)

{

\*cur=++tot;

memset(child[tot],0,sizeof(child[tot]));

flag[tot]=false;

}

}

flag[\*cur]=true;

}

bool query(const char \*str)

{

int \*cur=&root;

for (const char \*p=str;\*p && \*cur;++p)

cur=&child[\*cur][\*p-BASE];

return (\*cur&&flag[\*cur]);

}

};

后缀数组

void radix(int \*str,int \*a,int \*b,int n,int m)

{

static int count[200000];

memset(count,0,sizeof(count));

for (int i=0;i<n;i++) ++count[str[a[i]]];

for (int i=1;i<=m;i++) count[i]+=count[i-1];

for (int i=n01;i>=0;i--) b[--count[str[a[i]]]]=a[i];

}

void suffix\_array(int \*str,int \*sa,int n,int m)

{

static int rank[200000],a[200000],b[200000];

for (int i=0;i<n;i++) rank[i]=i;

radix(str,rank,sa,n,m);

rank[sa[0]]=0;

for (int i=1;i<n;i++) rank[sa[i]]=rank[sa[i-1]]+(str[sa[i]]!=str[sa[i-1]]);

for (int i=0;i<<1 <n;i++)

{

for (int j=0;j<n;j++)

{

a[j]=rank[j]+1;

b[j]=j+(1<<i) >=n ? 0:rank[j+(1<<i)]+1;

sa[j]=j;

}

radix(b,sa,rank,n,n);

radix(a,rank,sa,n,n);

rank[sa[0]]=0;

for (int j=1;j<n;j++)

{

rank[sa[j]]=rank[sa[j-1]]+(a[sa[j-1]]!=a[sa[j]] || b[sa[j-1]]!=b[sa[j]]);

}

}

}

void calc\_height(int \*str,int \*sa,int \*h,int n)

{

static int rank[200000];

int k=0;

h[0]=0;

for (int i=0;i<n;i++) rank[sa[i]]=i;

for (int i=0;i<n;i++)

{

k= k==0? 0: k-1;

if (rank[i]!=0)

while (str[i+k] == str[sa[rank[i]-1]+k]) k++;

h[rank[i]]=k;

}

}

AC自动机

struct tree

{

char ch;

int son,next,father,danger,suffix;

};

tree a[110000];

void insert(char \*s,int l,int t,int x)

{

int i;

if (a[x].danger) return ;

if (a[x].son==0)

{

m++;

a[x].son=m;

a[m].father=x;

a[m].ch=s[t];

if (t+1==l)

a[m].danger=1;

else

insert(s,l,t+1,m);

}

else

{

i=a[x].son;

while (1)

{

if (a[i].next==0||a[i].ch==s[t])

break;

i=a[i].next;

}

if (a[i].ch==s[t] && t+1==l) a[i].danger=1;

else if (a[i].ch==s[t]) insert(s,l,t+1,i);

else

{

m++;

a[i].next=m;

a[m].father=x;

a[m].ch=s[t];

if (t+1==l) a[m].danger=1;

else insert(s,l,t+1,m);

}

}

}

void build()

{

int child(int ,char);

int i,l,r;

l=r=1;

q[1]=1;

a[1].suffix=1;

if (a[1].son==0) return;

while (l<=r)

{

if (!a[q[l]].danger)

{

i=a[q[l]].son;

while (1)

{

r++;

q[r]=i;

i=a[i].next;

if (i==0) break;

}

}

i++;

}

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

{

if (a[q[i]].father==1)

{

a[q[i]].suffix=1;

continue;

}

a[q[i]].suffix=child(a[a[q[i]].father].suffix,a[q[i]].ch);

if (a[a[q[i]].suffix].danger)

a[q[i]].danger=1;

}

}

int child(int x,char ch)

{

int i

i=a[x].son;

while (i!=0)

{

if (a[i].ch==ch) break;

i=a[i].next;

}

if (i!=0) return i;

else if (x==1) return 1;

else return child(a[x].suffix,ch);

}

扩展KMP

void getnext()

{ int i,j,k=2,p,l;

next[1]=n;

next[2]=n-1;

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

if (a[i]!=a[i+1])

{ next[2]=i-1;

break ;

}

for (i=3;i<=n;++i)

{ p=k+next[k]-1;

if (i+next[i-k+1]<=p)

next[i]=next[i-k+1];

else

{ j=max(p-i+1,0);

while (i+j<=n&&a[i+j]==a[j+1])

++j;

next[i]=j;

k=i;

}

}

}

void ex\_kmp()

{ int l,i,j,k=1,p;

l=min(n,m);

ex[1]=l;

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

if (a[i]!=b[i])

{ ex[1]=i-1;

break ;

}

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

{ p=k+ex[k]-1;

if (i+next[i-k+1]<=p)

ex[i]=next[i-k+1];

else

{ j=max(p-i+1,0);

while (i+j<=m&&j<=n&&b[i+j]==a[j+1])

++j;

ex[i]=j,k=i;

}

}

}