目录

[Ac自动机 2](#_Toc469162168)

[后缀自动机 3](#_Toc469162169)

[后缀数组 4](#_Toc469162170)

[后缀树 5](#_Toc469162171)

[MANACHER 9](#_Toc469162172)

[回文树 9](#_Toc469162173)

[最小表示法 11](#_Toc469162174)

[DINIC 11](#_Toc469162175)

[费用流 13](#_Toc469162176)

[FFT 14](#_Toc469162177)

[BSGS 16](#_Toc469162178)

[找所有原根 16](#_Toc469162179)

[一个原根 20](#_Toc469162180)

[剩余定理 21](#_Toc469162181)

[高斯消元 21](#_Toc469162182)

[XOR高斯消元 22](#_Toc469162183)

[逆元递推式 22](#_Toc469162184)

[SPLAY 23](#_Toc469162185)

[堆优化DIJKSTRA 24](#_Toc469162186)

[匈牙利DFS 25](#_Toc469162187)

[MCS 26](#_Toc469162188)

[哈密顿回路 27](#_Toc469162189)

[高精度模板 29](#_Toc469162190)

[读写优化 31](#_Toc469162191)

[LINUX对拍 32](#_Toc469162192)

[kdtree 32](#_Toc469162193)

[random 34](#_Toc469162194)

#### Ac自动机

const int maxnode=60;

const int maxs=10;

int next[maxnode][maxs],fail[maxnode],cnt[maxnode];//end[]

int tot,root;

inline int newnode()

{

for(int i=0;i<maxs;i++)next[tot][i]=-1;

cnt[tot]=0;//注意改

return tot++;

}

void init()

{

tot=0;

root=newnode();

}

void insert(char str[],int val)

{

int len=strlen(str);

int now=root;

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

{

int s=str[i]-'0';//注意改

if(next[now][s]==-1)next[now][s]=newnode();

now=next[now][s];

}

cnt[now]|=val;

}

void build()

{

std::queue<int> q;

fail[root]=root;

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

{

if(next[root][i]==-1)next[root][i]=root;

else

{

fail[next[root][i]]=root;

q.push(next[root][i]);

}

}

while(!q.empty())

{

int now=q.front();

q.pop();

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

{

if(next[now][i]==-1)next[now][i]=next[fail[now]][i];

else

{

fail[next[now][i]]=next[fail[now]][i];

q.push(next[now][i]);

}

}

}

}

#### 后缀自动机

const int maxn=2e6+10;

char str[maxn];

int val[maxn],fa[maxn],c[maxn][26],ord[maxn],cc[maxn],cnt[maxn];

int tot,last;

//int len;//多串 每开始加一个串len=0;last=1;

inline int newnode(int step)

{

val[++tot]=step;

fa[tot]=0;

for(int i=0;i<26;i++)c[tot][i]=0;

return tot;

}

inline void extend(int k)

{

// len++;

// if(c[last][k]&&val[c[last][k]]==len)

// {

// last=c[last][k];

// return;

// }

//多串

int p=last;

int np=newnode(val[last]+1);

while(p&&!c[p][k])c[p][k]=np,p=fa[p];

if(!p)fa[np]=1;

else

{

int q=c[p][k];

if(val[q]==val[p]+1)fa[np]=q;

else

{

int nq=newnode(val[p]+1);

for(int i=0;i<26;i++)c[nq][i]=c[q][i];

fa[nq]=fa[q];

fa[q]=fa[np]=nq;

while(p&&c[p][k]==q)c[p][k]=nq,p=fa[p];

}

}

last=np;

}

inline void init()

{

tot=0;

last=newnode(0);

}

inline void getord()

{

rep(i,0,tot)cnt[i]=cc[i]=0;

rep(i,1,tot)cc[val[i]]++;

rep(i,1,tot)cc[i]+=cc[i-1];

rep(i,1,tot)ord[cc[val[i]]--]=i;

//求每个状态代表的子串的出现次数

int cur=1;

for(int i=0;str[i];i++)

{

cur=c[cur][str[i]-'a'];

cnt[cur]++;

}

red(i,tot,1)cnt[fa[ord[i]]]+=cnt[ord[i]];

}

#### 后缀数组

const int maxn=200010;

int sa[maxn],rnk[maxn],het[maxn];

int tp1[maxn],tp2[maxn],tp3[maxn];

inline bool cmp(int \*str,int a,int b,int k){return str[a]==str[b] && str[a+k]==str[b+k];}

void getsa(int \*str,int n,int m)

{

n++;

int \*x=tp1,\*y=tp2,\*c=tp3,i,j,p=0;

for(i=0;i<m;i++)c[i]=0;

for(i=0;i<n;i++)c[x[i]=str[i]]++;

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

for(i=n-1;i>=0;i--)sa[--c[x[i]]]=i;

for(j=1;j<=n && p<n;j<<=1,m=p)

{

p=0;

for(i=n-j;i<n;i++)y[p++]=i;

for(i=0;i<n;i++)if(sa[i]>=j)y[p++]=sa[i]-j;

for(i=0;i<m;i++)c[i]=0;

for(i=0;i<n;i++)c[x[y[i]]]++;

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

for(i=n-1;i>=0;i--)sa[--c[x[y[i]]]]=y[i];

swap(x,y);p=1;x[sa[0]]=0;

for(i=1;i<n;i++)x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;

}

for(i=1;i<n;i++)rnk[sa[i]]=i;

n--;

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

{

if(k)k--;

int j=sa[rnk[i]-1];

while(str[i+k]==str[j+k])k++;

het[rnk[i]]=k;

}

}

#### 后缀树

struct suftree

{

struct node

{

int l,r,par,link=-1,chcnt=0,ch[26];

node(int l=0,int r=0,int par=-1):l(l),r(r),par(par){clr(ch,-1);}

int Length(){return r-l;}

int& get(char c){return ch[c-'a'];}

void SetEdge(char c,int node\_id)

{

if(ch[c-'a']!=-1&&node\_id==-1)--chcnt;

else if(ch[c-'a']==-1&&node\_id!=-1)++chcnt;

ch[c-'a']=node\_id;

}

};

struct State

{

int v,pos;

State(int v,int pos):v(v),pos(pos){}

};

string text;

queue<int> qleaves;

vector<node>tree;

State ptr=State(0,0);

LL strcnt=0;

suftree(char \*str)

{

text=string(str);

tree.clear();

tree.reserve(2\*text.size()+1);

tree.pb(node(0));

}

State go(State st,int l,int r)

{

while(l<r)

{

if(st.pos==tree[st.v].Length())

{

st=State(tree[st.v].get(text[l]),0);

if(st.v==-1)return st;

}

else

{

if(text[tree[st.v].l+st.pos]!=text[l])return State(-1,-1);

if(r-l<tree[st.v].Length()-st.pos)return State(st.v,st.pos+r-l);

l+=tree[st.v].Length()-st.pos;

st.pos=tree[st.v].Length();

}

}

return st;

}

int split(const State& st)

{

if(st.pos==tree[st.v].Length())return st.v;

if(st.pos==0)return tree[st.v].par;

const node& v=tree[st.v];

int id=tree.size();

tree.pb(node(v.l,v.l+st.pos,v.par));

tree[v.par].SetEdge(text[v.l],id);

tree[id].SetEdge(text[v.l+st.pos],st.v);

tree[st.v].par=id;

tree[st.v].l+=st.pos;

return id;

}

int getlink(int v)

{

if(tree[v].link!=-1)return tree[v].link;

if(tree[v].par==-1)return 0;

int to=getlink(tree[v].par);

return tree[v].link=split(go(State(to,tree[to].Length()),tree[v].l+(tree[v].par==0),tree[v].r));

}

void extend(int pos)//pos为扩展字符下标

{

int mid;

strcnt+=qleaves.size();

do

{

State nptr=go(ptr,pos,pos+1);

if(nptr.v!=-1)

{

ptr=nptr;

return;

}

mid=split(ptr);

int leaf=tree.size();

strcnt++;

qleaves.push(leaf);

tree.pb(node(pos,text.size(),mid));

tree[mid].SetEdge(text[pos],leaf);

ptr.v=getlink(mid);

ptr.pos=tree[ptr.v].Length();

}while(mid!=0);

}

void delpre(int pos)//pos为当前扩展到的最后一个字符的下标+1

{

int leaf=qleaves.front();

qleaves.pop();

int par=tree[leaf].par;

while(tree[leaf].chcnt==0)

{

if(ptr.v!=leaf)

{

tree[par].SetEdge(text[tree[leaf].l],-1);

strcnt-=min(tree[leaf].r,pos)-tree[leaf].l;

leaf=par;

par=tree[leaf].par;

}

else

{

if(ptr.pos==min(tree[leaf].r,pos)-tree[leaf].l)break;

int mid=split(ptr);

ptr.v=mid;

strcnt-=min(tree[leaf].r,pos)-tree[leaf].l;

tree[mid].SetEdge(text[tree[leaf].l],-1);

tree[leaf]=tree[mid];

tree[tree[mid].par].SetEdge(text[tree[mid].l],leaf);

tree.pop\_back();

break;

}

}

if(leaf!=0&&tree[leaf].chcnt==0)

{

qleaves.push(leaf);

int to=(tree[leaf].par==0)?0:tree[tree[leaf].par].link;

ptr=go(State(to,tree[to].Length()),tree[leaf].l+(tree[leaf].par==0),tree[leaf].r);

tree[leaf].l=pos-tree[leaf].Length();

tree[leaf].r=text.size();

}

}

};

#### MANACHER

char s[MAXN], t[MAXN<<1];

int p, l, ans, po,len[MAXN<<1];

int main()

{

while (~scanf("%s",s))

{

l = strlen(s);

t[0] = '@';

for (int i = 1; i <= 2 \* l;i+=2)

{

t[i] = '#';

t[i + 1] = s[i / 2];

}

l = 2 \* l + 1;

t[l] = '#';

t[l + 1] = '%';

t[l + 2] = 0;

memset(len, 0, sizeof(len));

p = 1; po = 1; ans = 0;

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

{

if (po > i)len[i] = min(po - i+1, len[2 \* p - i]);

else len[i] = 1;

while (t[i - len[i]] == t[i + len[i]])len[i]++;

if (i + len[i]-1 > po)

{

p = i;

po = i + len[i]-1 ;

}

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

}

cout << ans - 1 << endl;

}

return 0;

}

#### 回文树

const int maxn=100010;

const int N=26;

struct Ptree

{

int nxt[maxn][N];

int fail[maxn],cnt[maxn],num[maxn],len[maxn],S[maxn];

int last,n,p;

int newnode(int ln)

{

for(int i=0;i<N;i++)nxt[p][i]=0;

num[p]=cnt[p]=0;len[p]=ln;

return p++;

}

void init()

{

p=0;newnode(0);newnode(-1);

n=last=0;

S[n]=-1;

fail[0]=1;

}

int gfail(int x)

{

while(S[n-len[x]-1]!=S[n])x=fail[x];

return x;

}

void add(int c)

{

c-='a';S[++n]=c;

int cur=gfail(last);

if(!nxt[cur][c])

{

int now=newnode(len[cur]+2);

fail[now]=nxt[gfail(fail[cur])][c];

nxt[cur][c]=now;

num[now]=num[fail[now]]+1;

}

last=nxt[cur][c];

cnt[last]++;

}

void count()//计算每种回文串出现次数

{

for(int i=p-1;i>=0;i--)cnt[fail[i]]+=cnt[i];

}

}ptree;

#### 最小表示法

scanf("%s",s);

int len=strlen(s),i=0,j=1,k=0;

while(i<len&&j<len&&k<len)

{

if(s[(i+k)%len]==s[(j+k)%len])k++;

else

{

if(s[(i+k)%len]>s[(j+k)%len])i=i+k+1;

else j=j+k+1;

if(i==j)j++;

k=0;

}

}

ans=min(i,j);

#### DINIC

const int inf=0x7fffffff;

struct edge

{

int v,w,next;

}e[maxm];

int h[maxn],ecnt;

int hh[maxn],tot;

void init()

{

clr(h,-1);

ecnt=0;

tot=//总点数

}

int lv[maxn];

bool bfs(int s,int t)

{

queue<int> q;

clr(lv,-1);

lv[s]=0;

q.push(s);

while(!q.empty())

{

int u=q.front();q.pop();

if(u==t)return 1;

for(int i=h[u];~i;i=e[i].next)

{

int v=e[i].v;

if(~lv[v] || !e[i].w)continue;

lv[v]=lv[u]+1;

q.push(v);

}

}

return 0;

}

int dfs(int u,int flow,int t)

{

if(u==t)return flow;

int ret=0,f;

for(int &i=hh[u];~i;i=e[i].next)

{

int v=e[i].v;

if(lv[v]!=lv[u]+1 || !e[i].w)continue;

f=dfs(v,min(flow-ret,e[i].w),t);

ret+=f;

e[i].w-=f;

e[i^1].w+=f;

if(ret==flow)return ret;

}

return ret;

}

int dinic(int s,int t)

{

int ret=0;

while(bfs(s,t))

{

memcpy(hh,h,sizeof(h[0])\*tot);

ret+=dfs(s,inf,t);

}

return ret;

}

#### 费用流

const int maxn=1010;

const int maxm=50010;

const int INF=0x3f3f3f3f;

struct edge

{

int v,w,c,next;

}e[maxm];

int h[maxn],ecnt;

inline void addedge(int u,int v,int w,int c)

{

e[ecnt]=(edge){v,w,c,h[u]};

h[u]=ecnt++;

}

inline void Addedge(int u,int v,int w,int c)

{

addedge(u,v,w,c);

addedge(v,u,0,-c);

addedge(v,u,w,c);//undirected

addedge(u,v,0,-c);

}

int dis[maxn],pre[maxn],peg[maxn];

bool vis[maxn];

int flow,cost;

bool spfa(int s,int t)

{

clr(vis,0);clr(pre,-1);

clr(dis,0x3f);

queue<int> q;

q.push(s);

vis[s]=1;

dis[s]=0;

while(!q.empty())

{

int u=q.front();q.pop();vis[u]=0;

for(int i=h[u];~i;i=e[i].next)

{

int v=e[i].v;

if(e[i].w && dis[u]+e[i].c<dis[v])

{

dis[v]=dis[u]+e[i].c;

pre[v]=u;

peg[v]=i;

if(!vis[v])q.push(v);

vis[v]=1;

}

}

}

return dis[t]!=INF;

}

void mcf(int s,int t)

{

flow=cost=0;

while(spfa(s,t))

{

int mf=INF;

for(int u=t;~pre[u];u=pre[u])mf=min(mf,e[peg[u]].w);

for(int u=t;~pre[u];u=pre[u])

{

e[peg[u]].w-=mf;

e[peg[u]^1].w+=mf;

}

flow+=mf;

cost+=mf\*dis[t];

}

}

#### FFT

const double pi=acos(-1);

struct cp

{

double r,i;

cp(double \_r=0,double \_i=0):r(\_r),i(\_i){}

cp operator+(const cp &x){return cp(r+x.r,i+x.i);}

cp operator-(const cp &x){return cp(r-x.r,i-x.i);}

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

}A[maxn];

int N,L,rev[maxn];

void fftinit(int mx)

{

for(N=1,L=0;N<mx;N<<=1,L++);N<<=1;L++;

rev[0]=0;

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

{

rev[i]=rev[i-1];

int t=L-1;

while((1<<t)&rev[i])rev[i]^=1<<t,t--;

rev[i]^=1<<t;

}

}

void fft(cp a[],int flag)

{

cp x,y;

for(int i=0;i<N;i++)A[i]=a[rev[i]];

for(int i=0;i<N;i++)a[i]=A[i];

for(int i=2;i<=N;i<<=1)

{

cp wn(cos(2\*pi/i),flag\*sin(2\*pi/i));

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

{

cp w(1,0);

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

{

x=a[j];

y=a[j+i/2]\*w;

a[j]=x+y;

a[j+i/2]=x-y;

w=w\*wn;

}

}

}

if(flag==-1)for(int i=0;i<N;i++)a[i].r/=N;

}

//注意清零，特别是0和超过长度的位置

//1004535809, 998244353, 104857601 g=3

#### BSGS

int bsgs(int a,int b,int p)//a^x %p =b

{

a%=p,b%=p;

if(b==1)return 0;

int cnt=0;

LL t=1;

for(int g=gcd(a,p);g!=1;g=gcd(a,p))

{

if(b%g)return -1;

p/=g,b/=g,t=t\*a/g%p;

cnt++;

if(b==t)return cnt;

}

map<int,int> hs;

int m=ceil(sqrt(p+0.1));

LL base=b;

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

{

hs[base]=i;

base=base\*a%p;

}

base=quipow(a,m,p);

LL now=t;

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

{

now=now\*base%p;

if(hs.count(now))

return i\*m-hs[now]+cnt;

}

return -1;

}

#### 找所有原根

int phi[maxn+10],prm[maxn+10],ptot;

bool vis[maxn+10];

void init()

{

phi[1]=1;

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

{

if(!vis[i])

{

phi[i]=i-1;

prm[++ptot]=i;

}

for(int j=1;j<=ptot&&(LL)i\*prm[j]<=maxn;j++)

{

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

if(i%prm[j]==0)

{

phi[i\*prm[j]]=phi[i]\*prm[j];

break;

}

else phi[i\*prm[j]]=phi[i]\*(prm[j]-1);

}

}

}

inline LL quipow(LL x,LL k,int mod)

{

LL ret=1;

while(k)

{

if(k&1)ret=ret\*x%mod;

x=x\*x%mod;

k>>=1;

}

return ret;

}

inline bool check(int n)

{

if(n%2==0)n/=2;

if(!vis[n])return 1;

for(int i=3;i\*i<=n;i+=2)

{

if(n%i==0)

{

while(n%i==0)n/=i;

return n==1;

}

}

return 0;

}

vector<int> p;

inline void getp(int x)

{

p.clear();

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

{

if(x%i==0)

{

p.pb(i);

while(x%i==0)x/=i;

}

}

if(x>1)p.pb(x);

}

inline bool gao(int g,int m)

{

if(quipow(g,phi[m],m)!=1)return 0;

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

{

if(quipow(g,phi[m]/p[i],m)==1)return 0;

}

return 1;

}

inline int gcd(int a,int b)

{

int t;

while(b)

{

t=a;a=b;b=t%b;

}

return a;

}

vector<int>ans;

inline void solve(int n)

{

getp(phi[n]);

int g;

rep(i,2,n)

{

if(gao(i,n))

{

g=i;

break;

}

}

ans.clear();

ans.pb(g);

int tmp=g,tt=phi[n]-1;

rep(i,2,tt)

{

tmp=(LL)tmp\*g%n;

if(gcd(i,phi[n])==1)ans.pb(tmp);

}

sort(ans.begin(),ans.end());

for(int i=1;i-1<ans.size();i++)

{

printf("%d%c",ans[i-1],i==ans.size()?'\n':' ');

}

}

int main()

{

init();

int n;

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

{

if(n==2)

{

puts("1");

}

else if(n==4)

{

puts("3");

}

else if(check(n))

{

solve(n);

}

else puts("-1");

}

return 0;

}

#### 一个原根

inline LL quipow(LL x,LL k,int mod)

{

LL ret=1;

while(k)

{

if(k&1)ret=ret\*x%mod;

x=x\*x%mod;

k>>=1;

}

return ret;

}

vector<int> p;

inline void getp(int x)

{

p.clear();

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

{

if(x%i==0)

{

p.pb(i);

while(x%i==0)x/=i;

}

}

if(x>1)p.pb(x);

}

inline bool gao(int g,int m)

{

if(quipow(g,m-1,m)!=1)return 0;

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

{

if(quipow(g,(m-1)/p[i],m)==1)return 0;

}

return 1;

}

inline void solve(int n)

{

getp(n-1);

int g;

rep(i,2,n)

{

if(gao(i,n))

{

g=i;

break;

}

}

cout<<g<<endl;

}

int main()

{

int n;

solve(n);

return 0;

}

#### 剩余定理

int CRT(int cnt)

{

LL ret=0,x,y;

rep(i,1,cnt)

{

exgcd(m/p[i],p[i],x,y);

ret=(ret+m/p[i]\*x\*a[i])%m;//爆LL手写mul

}

return (ret+m)%m;

}

#### 高斯消元

const int maxn=1000;

const double eps=1e-8;

double a[maxn][maxn],x[maxn];

void gauss(int n)

{

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

{

int mxr=i;

for(int j=i+1;j<=n;j++)if(fabs(a[j][i])>fabs(a[mxr][i])+eps)mxr=j;

if(mxr!=i)

{

for(int j=i;j<=n+1;j++)swap(a[i][j],a[mxr][j]);

}

for(int j=i+1;j<=n;j++)if(fabs(a[j][i])>eps)

{

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

for(int k=i;k<=n+1;k++)a[j][k]-=tmp\*a[i][k];

}

}

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

{

for(int j=i+1;j<=n;j++)a[i][n+1]-=a[i][j]\*x[j];

x[i]=a[i][n+1]/a[i][i];

}

}

#### XOR高斯消元

void gauss()

{

LL i,j,k=0;

for(j=1ll<<59;j;j>>=1)

{

for(i=k+1;i<=tot;i++)

if(a[i]&j)break;

if(i==tot+1)continue;

swap(a[++k],a[i]);

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

if(i!=k&&(a[i]&j))a[i]^=a[k];

}

tot=k;

}

#### 逆元递推式

inv[i]=(p-p/i)\*inv[p%i] %p

#### SPLAY

int ch[maxn][2],fa[maxn],sz[maxn],val[maxn];

inline void pushup(int x){};

inline void rotate(int x,int &k)

{

int y=fa[x],z=fa[y],l=(ch[y][1]==x),r=l^1;

if(k==y)k=x;else ch[z][ch[z][1]==y]=x;

fa[x]=z;fa[y]=x;fa[ch[x][r]]=y;

ch[y][l]=ch[x][r];ch[x][r]=y;

pushup(y);pushup(x);

}

inline void splay(int x,int &k)

{

int y,z;

while(x!=k)

{

y=fa[x],z=fa[y];

if(y!=k)

{

if(ch[y][0]==x ^ ch[z][0]==y)rotate(x,k);

else rotate(y,k);

}

rotate(x,k);

}

}

inline void newnode(int &x,int f,char c)

{

x=++tot;

fa[x]=f;val[x]=c;sz[x]=1;

ch[x][0]=ch[x][1]=0;

}

inline void init()

{

root=tot=0;

}

void splayk(int k,int &x)

{

int p=root;

while(sz[ch[p][0]]!=k-1)

{

if(sz[ch[p][0]]>k-1)p=ch[p][0];

else

{

k-=sz[ch[p][0]]+1;

p=ch[p][1];

}

}

splay(p,x);

}

#### 堆优化DIJKSTRA

typedef pair<LL,int> mp;

void dijkstra(int s)

{

priority\_queue<mp,vector<mp>,greater<mp> > q;

clr(vis,0);

clr(dis,0x3f);

dis[s]=0;

q.push(mp(0,s));

while(!q.empty())

{

mp p=q.top();q.pop();

int u=p.second;

if(vis[u])continue;

for(int i=h[u];~i;i=e[i].next)

{

int v=e[i].v;

if(dis[u]+e[i].w<dis[v])

{

dis[v]=dis[u]+e[i].w;

q.push(mp(dis[v],v));

}

}

vis[u]=1;

}

}

#### 匈牙利DFS

const int MAXN=300;

bool bmap[MAXN][MAXN];

bool bmask[MAXN];

int nx,ny;

int cx[MAXN],cy[MAXN];

int findpath(int u)

{

rep(i,1,ny)

{

if(bmap[u][i]&&!bmask[i])

{

bmask[i]=1;

if(cy[i]==-1||findpath(cy[i]))

{

cy[i]=u;

cx[u]=i;

return 1;

}

}

}

return 0;

}

int maxmatch()

{

int ret=0;

rep(i,1,nx)cx[i]=-1;

rep(i,1,ny)cy[i]=-1;

rep(i,1,nx)

{

if(cx[i]==-1)

{

rep(j,1,ny)bmask[j]=0;

ret+=findpath(i);

}

}

return ret;

}

#### MCS

const int maxn=1010;

const int maxm=3500010;

struct List

{

int v,next;

}list[maxm];

int eh[maxn],tot,mh[maxn];

inline void add(int h[],int u,int v)

{

list[tot].v=v;

list[tot].next=h[u];

h[u]=tot++;

}

int n,m,best;

int q[maxn],f[maxn],g[maxn];

bool vis[maxn];

void MCS()

{

clr(mh,-1);clr(vis,0);clr(f,0);

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

best=0;

for(int j=n;j;j--)

{

while(1)

{

int i;

for(i=mh[best];~i;i=list[i].next)

{

if(!vis[list[i].v])break;

else mh[best]=list[i].next;

}

if(~i)

{

int x=list[i].v;

q[j]=x;g[x]=j;

vis[x]=1;

for(i=eh[x];~i;i=list[i].next)if(!vis[list[i].v])

{

f[list[i].v]++;

add(mh,f[list[i].v],list[i].v);

best=max(best,f[list[i].v]);

}

break;

}

else best--;

}

}

}

#### 哈密顿回路

//充分条件：任意两点度数和大于等于n

const int maxn=500;

int ans[maxn],n,m;

bool con[maxn][maxn],vis[maxn];

inline void reverse(int s,int t)

{

while(s<t)

{

swap(ans[s],ans[t]);

s++,t--;

}

}

inline int extend(int &cur)

{

while(1)

{

bool flag=1;

rep(i,1,n)if(!vis[i] && con[i][ans[cur-1]])

{

vis[i]=1;

ans[cur++]=i;

flag=0;

break;

}

if(flag)break;

}

return ans[cur-1];

}

void Hamilton()

{

int s=1,t;

clr(vis,0);

int i;

for(i=2;i<=n;i++)if(con[s][i])break;

t=i;

int cur=2;

ans[0]=s;ans[1]=t;

vis[s]=vis[t]=1;

while(1)

{

t=extend(cur);

reverse(0,cur-1);

swap(s,t);

t=extend(cur);

if(!con[s][t])

{

int k=cur-1;

rep(i,1,k)if(con[ans[i]][t] && con[ans[i+1]][s])

{

reverse(i+1,k);

t=ans[k];

break;

}

}

if(cur==n)return;

rep(i,1,n)if(!vis[i])

{

bool flag=0;

rep0(j,cur)if(con[ans[j]][i])

{

reverse(0,j-1);

reverse(j,cur-1);

t=ans[cur++]=i;

vis[i]=1;

flag=1;

break;

}

if(flag)break;

}

}

}

#### 高精度模板

const int MAXN = 410;

char tmps[MAXN];

struct bign

{

int len,s[MAXN];

//bool minus;

bign(){memset(s,0,sizeof(s));len=1;}

bign(int num) {\*this=num;}

bign(const char \*num) {\*this=num;}

bign operator = (const int num)

{

sprintf(tmps,"%d",num);

\*this=tmps;

return \*this;

}

bign operator = (const char \*num)

{

len=strlen(num);

for(int i=0; i<len; i++)s[i]=num[len-i-1]-'0';

return \*this;

}

void read(){scanf("%s",tmps);\*this=tmps;}

void write(){for(int i=len-1;i>=0;i--)putchar(s[i]+'0');}

bign operator + (const bign &b) const

{

bign c;c.len=0;

for(int i=0,g=0;g || i<max(len,b.len);i++)

{

int x=g;

if(i<len)x+=s[i];

if(i<b.len)x+=b.s[i];

c.s[c.len++]=x%10;

g=x/10;

}

return c;

}

void clean(){while(len > 1 && !s[len-1])len--;}

bign operator \* (const bign &b) const

{

bign c;c.len=len+b.len;

for(int i=0;i<len;i++)for(int j=0;j<b.len;j++)c.s[i+j]+=s[i]\*b.s[j];

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

{

c.s[i+1]+=c.s[i]/10;

c.s[i]%=10;

}

c.clean();

return c;

}

bign operator - (const bign &b)

{

bign c;c.len=0;

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

{

int x=s[i]-g;

if(i<b.len)x-=b.s[i];

if(x>=0)g=0;

else g=1,x+=10;

c.s[c.len++]=x;

}

c.clean();

return c;

}

bign operator / (const bign &b)

{

bign c,f=0;

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

{

f=f\*10;

f.s[0]=s[i];

while(f>=b)f=f-b,c.s[i]++; //need >=

}

c.len=len;

c.clean();

return c;

}

bign operator % (const bign &b){return \*this-\*this/b\*b;}

bool operator < (const bign &b)

{

if(len!=b.len) return len<b.len;

for(int i=len-1;i>=0;i--)if(s[i]!=b.s[i])return s[i]<b.s[i];

return false;

}

bool operator > (const bign &b)

{

if(len!=b.len) return len>b.len;

for(int i=len-1;i>=0;i--)if(s[i]!=b.s[i])return s[i]>b.s[i];

return false;

}

bool operator == (const bign &b)

{

return !(\*this>b) && !(\*this<b);

}

bool operator != (const bign &b)

{

return !(\*this==b);

}

bool operator <= (const bign &b)

{

return !(\*this>b);

}

bool operator >= (const bign &b)

{

return !(\*this<b);

}

};

#### 读写优化

inline void read(int &x) {

char c;int f = 1;x = 0;

while(((c=getchar()) < '0' || c > '9') && c != '-');

if(c == '-') f = -1;else x = c-'0';

while((c=getchar()) >= '0' && c <= '9') x = x\*10+c-'0';

x \*= f;

}

int outn;

char out[];

inline void write(int x) {

if(x < 0) out[outn++] = '-', x = -x;

if(x) {

int tmpn = 0, tmp[20];

while(x) {

tmp[tmpn++] = x%10+'0';

x /= 10;

}

while(tmpn)

out[outn++] = tmp[--tmpn];

}

else out[outn++] = '0';

}

#### LINUX对拍

while true; do

./generate > input

./run < input > output

./std < input > answer

if diff output answer; then

echo OK

else

echo WA

break

fi

done

#### kdtree

const int inf=0x7fffffff;

const int maxn=2000010;

int rt,curd;

int qd[2],qans;

struct node

{

int d[2],l,r,mx[2],mi[2];

}p[maxn];

inline bool cmp(const node &a,const node &b)

{

return a.d[curd]==b.d[curd]?a.d[curd^1]<b.d[curd^1]:a.d[curd]<b.d[curd];

}

inline void up(node &x,node &y)

{

rep0(i,2)x.mi[i]=min(x.mi[i],y.mi[i]);

rep0(i,2)x.mx[i]=max(x.mx[i],y.mx[i]);

}

int build(int l,int r,int dd)

{

int mid=l+r>>1;

curd=dd;

nth\_element(p+l,p+mid,p+r+1,cmp);

rep(i,0,1)p[mid].mx[i]=p[mid].mi[i]=p[mid].d[i];

if(l!=mid)p[mid].l=build(l,mid-1,dd^1);else p[mid].l=0;

if(r!=mid)p[mid].r=build(mid+1,r,dd^1);else p[mid].r=0;

if(p[mid].l)up(p[mid],p[p[mid].l]);

if(p[mid].r)up(p[mid],p[p[mid].r]);

return mid;

}

inline void ins(int x)

{

int dd=0,cur=rt;

p[x].l=p[x].r=0;

rep(i,0,1)p[x].mi[i]=p[x].mx[i]=p[x].d[i];

while(1)

{

up(p[cur],p[x]);

if(p[x].d[dd]>=p[cur].d[dd])

{

if(p[cur].r==0){p[cur].r=x;return;}

else cur=p[cur].r;

}

else

{

if(p[cur].l==0){p[cur].l=x;return;}

else cur=p[cur].l;

}

dd^=1;

}

}

inline int dist(int x)

{

int ret=0;

rep(i,0,1)

{

if(qd[i]<p[x].mi[i])ret+=p[x].mi[i]-qd[i];

if(qd[i]>p[x].mx[i])ret+=qd[i]-p[x].mx[i];

}

return ret;

}

inline void qry(int x)

{

int dl=inf,dr=inf,d0=0;

rep(i,0,1)d0+=abs(p[x].d[i]-qd[i]);

qans=min(qans,d0);

if(p[x].l)dl=dist(p[x].l);

if(p[x].r)dr=dist(p[x].r);

if(dl<dr)

{

if(dl<qans)qry(p[x].l);

if(dr<qans)qry(p[x].r);

}

else

{

if(dr<qans)qry(p[x].r);

if(dl<qans)qry(p[x].l);

}

}

#### random

#include<random>

typedef uniform\_int\_distribution<int> uidint;

default\_random\_engine g(time(NULL));

uidint d1(1, 8);

d1(g)