xaphoenix 模板

目录

1. 数学
2. 欧几里得算法······························ 2
3. 扩展欧几里得算法···························· 2
4. 中国剩余定理······························ 2
5. 单变元线性模方程组··························· 2
6. 求原根································· 3
7. Baby step giant step···························· 3
8. 扩展Baby step giant step·························· 4
9. 平方剩余································ 4
10. N次剩余································ 5
11. Miller-rabin······························· 6
12. Rho·································· 6
13. 筛法·································· 7
14. 筛欧拉函数······························· 8
15. 筛逆元································· 8
16. Mobius································· 8
17. 高斯消元································ 9
18. 矩阵求逆································10
19. 快速幂·································11
20. 高精度·································11
21. 自适应simpson····························· 15
22. FFT···································15
23. NTT·································· 16
24. 图论
25. Fleury·································· 18
26. Topo···································19
27. Kruskal··································19
28. Prim···································20
29. Floyd···································20
30. Dinic···································20
31. Hungry··································22
32. 割点···································23
33. 桥····································24
34. 数据结构
35. 并查集··································24
36. Valuesplay·································24
37. Subsplay································· 28
38. Treap···································31
39. 划分树··································33
40. 左偏树··································34
41. 数学
42. 欧几里得算法

int gcd(int a,int b)

{

return b==0?a:gcd(b,a%b);

}

1. 扩展欧几里得算法

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

{

if (b==0)

{

x=1;

y=0;

return a;

}

else

{

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

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

return r;

}

}

1. 中国剩余定理

int CRT(int a[],int m[],int n)

{

int M=1;

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

M\*=m[i];

int ret=0;

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

{

int x,y;

int tm=M/m[i];

extended\_gcd(tm,m[i],x,y);

ret=(ret+tm\*x\*a[i])%M;

}

return (ret+M)%M;

}

1. 单变元线性模方程组

int China(int n)

{

int m1,r1,m2,r2,flag=0,i,d,x,y,c,t;

scanf("%d%d",&m1,&r1);

flag=0;

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

{

scanf("%d%d",&m2,&r2);

if(flag)continue;

gcd(m1,m2,d,x,y);

c=r2-r1;

if(c%d)

{

flag=1;

continue;

}

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;

}

1. 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;

}

}

1. Baby Step Giant Step

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;

}

1. 扩展Baby Step Giant Step

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;

}

1. 平方剩余

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;

}

1. N次剩余

vector <int> residue(int p,int N,int a)

{

int g=primitive\_root(p);

long long m=discrete\_log(g,a,p);

vector<int> ret;

if (a==0)

{

ret.push\_back(0);

return ret;

}

if (m==-1) return ret;

long long A=N,B=p-1,C=m,x,y;

long long d=extended\_gcd(A,B,x,y);

if (c%d!=0) return ret;

x=x\*(C/d)%B;

long long delta=B/d;

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

{

x=((x+delta)%B+B)%B;

ret.push\_back((int)pow\_mod(g,x,p));

}

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

ret.erase(unqinue(ret.begin(),ret.end()),ret.end());

return ret;

}

1. Miller-rabin

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;

}

1. Rho

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 (isprimes(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;

}

}

}

1. 筛法

#define maxn 1000000

bool valid[maxn];

void getPrime(int n,int &tot,int ans[maxn])

{

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;

}

}

}

1. 筛欧拉函数

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;

}

}

1. 筛逆元

inv[1]=1;

for(int i=2;i<=min(2\*n,p-1);i++)

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

1. Mobius

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;

}

}

}

}

1. 高斯消元

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;

}

1. 矩阵求逆

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];

}

}

}

1. 快速幂

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

{

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

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

temp=temp\*temp%n;

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

return temp;

}

1. 高精度

const int ten[4]={1,10,100,1000};

const int maxl=1000;

struct BigNumber

{

int d[maxl];

BigNumber (string s)

{

int len=s.size();

d[0]=(len-1)/4+1;

int i,j,k;

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

d[i]=0;

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

{

j=(len-i-1)/4+1;

k=(len-i-1)%4;

d[j]+=ten[k]\*(s[i]-'0');

}

while (d[0]>1&&d[d[0]]==0)

d[0]--;

}

BigNumber()

{

\*this=BigNumber(string("0"));

}

string toString()

{

string s("");

int i,j,temp;

for (i=3;i>=1;i--)

if (d[d[0]]>=ten[i]) break;

temp=d[d[0]];

for (j=i;j>=0;j--)

{

s=s+(char)(temp/ten[j]+'0');

temp%=ten[j];

}

for (i=d[0]-1;i>0;i--)

{

temp=d[i];

for (j=3;j>=0;j--)

{

s=s+(char)(temp/ten[j]+'0');

temp%=ten[j];

}

}

return s;

}

}zero("0"),d,temp,mid1[15];

bool operator < (const BigNumber &a, const BigNumber &b)

{

if (a.d[0]!=b.d[0]) return a.d[0]<b.d[0];

int i;

for (i=a.d[0];i>0;i--)

if (a.d[i]!=b.d[i]) return a.d[i]<b.d[i];

return false;

}

BigNumber operator +(const BigNumber &a,const BigNumber &b)

{

BigNumber c;

c.d[0]=max(a.d[0],b.d[0]);

int i,x=0;

for (i=1;i<=c.d[0];i++)

{

x=a.d[i]+b.d[i]+x;

c.d[i]=x%10000;

x/=10000;

}

while (x!=0)

{

c.d[++c.d[0]]=x%10000;

x/=10000;

}

return c;

}

BigNumber operator -(const BigNumber &a,const BigNumber &b)

{

BigNumber c;

c.d[0]=a.d[0];

int i,x=0;

for (i=l;i<=c.d[0];i++)

{

x=10000+a.d[i]-b.d[i]+x;

c.d[i]=x%10000;

x=x/10000-1;

}

while ((c.d[0]>1)&&(c.d[c.d[0]]==0)) c.d[0]--;

return c;

}

BigNumber operator \* (const BigNumber &a,const BigNumber &b)

{

BigNumber c;

c.d[0]=a.d[0]+b.d[0];

int i,j,x;

for (i=1;i<=a.d[0];i++)

{

x=0;

for (j=1;j<=b.d[0];j++)

{

x=a.d[i]\*b.d[j]+x+c.d[i+j-1];

c.d[i+j-1]=x%10000;

x/=10000;

}

c.d[i+b.d[0]]=x;

}

while ((c.d[0]>1)&&(c.d[c.d[0]]==0)) c.d[0]--;

return c;

}

bool smaller (const BigNumber &a,const BigNumber &b,int delta)

{

if (a.d[0]+delta!=b.d[0]) return a.d[0]+delta<b.d[0];

int i;

for (i=a.d[0];i>0;i--)

if (a.d[i]!=b.d[i+delta]) return a.d[i]<b.d[i+delta];

return truel

}

void Minus(BigNumber &a, const BigNumber &b,int delta)

{

int i,x=0;

for (i=1;i<=a.d[0]-delta;i++)

{

x=10000+a.d[i+delta]-b.d[i]+x;

a.d[i+delta]=x%10000;

x=x/10000-1;

}

while ((a.d[0]>1)&&(a.d[a.d[0]]==0) a.d[0]--;

}

BigNumber operator \*(const BigNumber &a,const int &k)

{

BigNumber c;

c.d[0]=a.d[0];

int i,x=0;

for (i=1;i<=a.d[0];i++)

{

x=a.d[i]\*k+x;

c.d[i]=x%10000;

x/=10000;

}

while (x>0)

{

c.d[++c.d[0]]=x%10000;

x/=10000;

}

while ((c.d[0]>1)&&(c.d[c.d[0]]==0) c.d[0]--;

return c;

}

BigNumber operator / (const BigNumber &a,const BigNumber &b)

{

BigNumber c;

d=a;

int i,j,temp;

mid1[0]=b;

for (i=1;i<=13;i++)

mid1[i]=mid1[i-1]\*2;

for (i=a.d[0]-b.d[0];i>=0;i--)

{

temp=8192;

for (j=13;j>=0;j--)

{

if (smaller(mid1[j],d,i))

{

Minus(d,mid1[j],i);

c.d[i+1]+=temp;

}

temp/=2;

}

}

c.d[0]=max(1,a.d[0]-b.d[0]+1);

while ((c.d[0]>1)&&(c.d[c.d[0]]==0) c.d[0]--;

return c;

}

bool operator ==(const BigNumber &a,cosnt BigNumber &b)

{

int i;

if (a.d[0]!=b.d[0]) return false;

for (i=1;i<=a.d[0];i++)

if (a.d[i]!=b.d[i]) return false;

return true;

}

1. 自适应simpson

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;

}

1. FFT

typedef long long Int64;

const int maxn=2000000;

const double pi=acos(-1.0);

typedef complex<double> Complex;

void build (Complex\_P[],Complex P[],int n,int m,int curr,int &cnt)

{

if (m==n) \_P[cnt++];

else

{

build(\_P,P,n,m\*2,curr,cnt);

build(\_P,P,n,m\*2,curr+m,cnt);

}

}

void FFT(Complex P[],int n,int oper)

{

static Complex \_P[maxn];

int cnt=0;

build(\_P,P,n,1,0,cnt);

copy(\_P,\_P+n,P);

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

{

int m=1<<d;

int m2=m\*2;

double p0=pi/m\*oper;

Complex unit\_p0=Complex(cos(p0),sin(p0));

for (int i=0;i<n;i+=m2)

{

Complex unit=1;

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

{

Complex &P1=P[i+j+m],&P2=P[i+j];

Complex t=unit\*P1;

P1=P2-t;

P2=P2+t;

unit=unit\*unit\_p0;

}

}

}

}

1. NTT

void init()

{

while((n+m)>=(1<<len))len++;

N=(1<<len);

inv=pow\_mod(N,mod-2,MOD);

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

{

int pos=0;

int temp=i;

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

{

pos<<=1;pos |= temp&1;temp>>=1;

}

rev[i]=pos;

}

}

void ntt(int \*a,int n,int re)

{

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

{

if(rev[i]>i)

{

swap(a[i],a[rev[i]]);

}

}

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

{

int mid=i>>1;

int wn=pow\_mod(G,(mod-1)/i,MOD);

if(re) wn=pow\_mod(wn,(mod-2),MOD);

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

{

int w=1;

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

{

int temp1=a[j+k];

int temp2=(ll)a[j+k+mid]\*w%mod;

a[j+k]=(temp1+temp2);if(a[j+k]>=mod)a[j+k]-=mod;

a[j+k+mid]=(temp1-temp2);if(a[j+k+mid]<0)a[j+k+mid]+=mod;

w=(ll)w\*wn%mod;

}

}

}

if(re)

{

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

{

a[i]=(ll)a[i]\*inv%mod;

}

}

}

int main()

{

ntt(a,N,0);

ntt(b,N,0);

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

{

c[i]=(ll)a[i]\*b[i]%mod;

}

ntt(c,N,1);

return 0;

}

1. 图论
2. Fleury

void dfs(int x)

{

S.push(x);

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

{

if(edge[x][i]>0)

{

edge[i][x]=edge[x][i]=0;

dfs(i);

break;

}

}

}

void Fleury(int x){

S.push(x);

while(!S.empty()){

int b=0;

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

if(edge[S.top()][i]>0){

b=1;

break;

}

}

if(b==0){

printf("%d",S.top());

S.pop();

}else {

int y=S.top();

S.pop();

dfs(y);

}

printf("\n");

}

1. Topo

void topo()

{

//priority\_queue<int,vector<int>,cmp> q; //点有序时

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

if(in[i]==0) q.push(i);

int flag=0;

while(!q.empty())

{

int x=q.top();

q.pop();

ans[++num]=x;

for(int i=0;i<way[x].size();i++)

{

in[way[x][i]]--;

if(in[way[x][i]]==0)

q.push(way[x][i]);

}

}

}

1. Kruskal

struct Edge

{

int start,end;

double length;

bool visit;

}edge[MAXN\*MAXN];

double kruskal(int n)

{

int i,j=0;

double sum=0;

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

sort(edge,edge+Count,cmp);

for (i=0;i<Count&&j<n;i++)

{

if (Union(edge[i].start,edge[i].end))

{

sum+=edge[i].length;

edge[i].visit=1;

j++;

}

}

return sum;

}

1. Prim

void Prim()

{

memset(v,0,sizeof v);

for (int i=1;i<=N;i++) dis[i]=INF;

dis[1]=0;

int ans=0;

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

{

int mark=-1;

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

if (!v[j])

{

if (mark==-1) mark=j;

else if (dis[j]<dis[mark]) mark=j;

if (mark==-1) break;

v[mark]=1;

ans+=dis[mark];

for (int j=0;j<g[mark].size();j++)

if (!v[g[mark][j].first])

{

int x=g[mark][j].first;

dis[x]=min(dis[x],g[mark][j].second);

}

}

}

return ans;

}

1. Floyd

const int MaxN=111;

const int INF=100000000;

int N,g[MaxN][MaxN];

void floyd()

{

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

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

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

g[i][j]=min(g[i][j],g[i][k]+g[k][j]);

}

1. dinic

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;

}

1. Hungry

const int MAXN=1000;

int uN,vN; //u,v数目

int g[MAXN][MAXN];//编号是0~n-1的

int linker[MAXN];

bool used[MAXN];

bool dfs(int u)

{

int v;

for(v=0;v<vN;v++)

if(g[u][v]&&!used[v])

{

used[v]=true;

if(linker[v]==-1||dfs(linker[v]))

{

linker[v]=u;

return true;

}

}

return false;

}

int hungary()

{

int res=0;

int u;

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

for(u=0;u<uN;u++)

{

memset(used,0,sizeof(used));

if(dfs(u)) res++;

}

return res;

}

1. 割点

void tarjan(int x,int par)//BCC & Cut Vertex

{ int i,dc=0;

dfn[x]=low[x]=++T;

st.push(x);

for (i=last[x];i!=0;i=pre[i])

if (dfn[e[i]]==0)

{ ++dc;

tarjan(e[i],x);

low[x]=min(low[x],low[e[i]]);

if (low[e[i]]>=dfn[x])

{ cut[x]=1;

++bcn;

while (!st.empty()&&st.top()!=e[i])

{ bcc[bcn].push\_back(st.top());

st.pop();

}

bcc[bcn].push\_back(e[i]);

st.pop();

bcc[bcn].push\_back(x);

}

}

else low[x]=min(low[x],dfn[e[i]]);

if (par==0&&dc==1)

cut[x]=0;

}

1. 桥

void tarjan(int x)//Cut Edge

{ int i;

low[x]=dfn[x]=++T;

for (i=last[x];i!=0;i=pre[i])

if (dfn[e[i]]==0)

{ epar[e[i]]=((i-1)>>1)+1;

tarjan(e[i]);

low[x]=min(low[x],low[e[i]]);

}

else

if (epar[x]!=(((i-1)>>1)+1))

low[x]=min(low[x],dfn[e[i]]);

if (low[x]==dfn[x])

cut[epar[x]]=1;

}

1. 数据结构
2. 并查集

int find(int x)

{

if (father[x]==x) return x;

father[x]=find(father[x]);

return father[x];

}

bool Union(int x,int y)

{

int f1=find(x);

int f2=find(y);

if (f1==f2) return false;

if (f1<f2) father[f1]=f2;

else father[f2]=f1;

return true;

}

1. ValueSplay

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;

}

1. Subsplay

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;

}

1. Treap

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;

1. 划分树

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);

}

}

1. 左偏树

struct lf

{

int dist;

LL key;

int left,right,ll,rr,way;

}nd[maxn];

int merge(int a,int b)

{

int c;

if ((a==0)||((b!=0)&&(nd[a].key<nd[b].key))) swap(a,b);

if (b==0) return a;

nd[a].right=merge(nd[a].right,b);

if (nd[nd[a].left].dist<nd[nd[a].right].dist) swap(nd[a].left,nd[a].right);

nd[a].dist=nd[nd[a].right].dist+1;

return a;

}

**图论**

struct node

{ int v;

long long d;

};

void swap(int x,int y)

{ struct node t;

t=heap[x],heap[x]=heap[y],heap[y]=t;

pos[heap[x].v]=x,pos[heap[y].v]=y;

}

void heapify(int k)

{ int i,j;

i=k,j=2\*i;

while(j<=hl)

{ if (j+1<=hl&&heap[j+1].d<heap[j].d)

++j;

if (heap[i].d<heap[j].d)

break ;

else swap(i,j),i=j,j=i\*2;

}

}

void upward(int x)

{ int i;

i=x;

while (i>0)

{ if (heap[i].d<heap[i/2].d)

swap(i,i/2),i/=2;

else break ;

}

}

void del(int x)

{ swap(x,hl);

--hl;

heapify(1);

}

void dijkstra(long long dis[],int src)

{ int i,u;

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

{ heap[i].d=INF;

heap[i].v=i;

pos[i]=i;

}

hl=n;

swap(src,1);

heap[1].d=0;

while (hl>0)

{ u=heap[1].v;

del(1);

for (i=last[u];i!=0;i=pre[i])

if (pos[e[i]]<=hl&&heap[pos[u]].d+w[i]<heap[pos[e[i]]].d)

{ heap[pos[e[i]]].d=heap[pos[u]].d+w[i];

upward(pos[e[i]]);

}

}

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

dis[i]=heap[pos[i]].d;

}

//Dijkstra

priority\_queue<pair<int,int>,vector<pair<int,int> >,greater<pair<int,int> > > q;

int dijkstra()

{ int i,u;

memset(in,0,sizeof(in));

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

dis[i]=INF;

dis[1]=0;

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

q.push(make\_pair(dis[i],i));

while (!q.empty())

{ u=q.top().second;

q.pop();

if (in[u])

continue ;

in[u]=1;

for (i=last[u];i!=0;i=pre[i])

if (dis[e[i]]>dis[u]+w[i])

{ dis[e[i]]=dis[u]+w[i];

q.push(make\_pair(dis[e[i]],e[i]));

}

}

return dis[n];

}

void tarjan(int x,int par)//BCC & Cut Vertex

{ int i,dc=0;

dfn[x]=low[x]=++T;

st.push(x);

for (i=last[x];i!=0;i=pre[i])

if (dfn[e[i]]==0)

{ ++dc;

tarjan(e[i],x);

low[x]=min(low[x],low[e[i]]);

if (low[e[i]]>=dfn[x])

{ cut[x]=1;

++bcn;

while (!st.empty()&&st.top()!=e[i])

{ bcc[bcn].push\_back(st.top());

st.pop();

}

bcc[bcn].push\_back(e[i]);

st.pop();

bcc[bcn].push\_back(x);

}

}

else low[x]=min(low[x],dfn[e[i]]);

if (par==0&&dc==1)

cut[x]=0;

}

void tarjan(int x)//Cut Edge

{ int i;

low[x]=dfn[x]=++T;

for (i=last[x];i!=0;i=pre[i])

if (dfn[e[i]]==0)

{ epar[e[i]]=((i-1)>>1)+1;

tarjan(e[i]);

low[x]=min(low[x],low[e[i]]);

}

else

if (epar[x]!=(((i-1)>>1)+1))

low[x]=min(low[x],dfn[e[i]]);

if (low[x]==dfn[x])

cut[epar[x]]=1;

}

void tarjan(int x)//SCC

{ int i;

dfn[x]=low[x]=++T;

st.push(x);

instack[x]=1;

for (i=last[x];i!=0;i=pre[i])

if (dfn[e[i]]==0)

{ tarjan(e[i]);

low[x]=min(low[x],low[e[i]]);

}

else

if (instack[e[i]])

low[x]=min(low[x],dfn[e[i]]);

if (low[x]==dfn[x])

{ ++scc\_cnt;

while (!st.empty()&&st.top()!=x)

{ scc[st.top()]=scc\_cnt;

instack[st.top()]=0;

st.pop();

}

st.pop();

scc[x]=scc\_cnt;

instack[x]=0;

}

}

#define targ(p,x) (qr[p].first==x?qr[p].second:qr[p].first)

void tarjan(int u)//LCA

{ int i;

for (i=last[u];i!=0;i=pre[i])

{ tarjan(e[i]);

f[find(e[i])]=u;

}

c[u]=1;

for (i=0;i<inq[u].size();++i)

if (c[targ(inq[u][i],u)]&&res[inq[u][i]]==-1)

res[inq[u][i]]=trie[find(targ(inq[u][i],u))].cod;

}

int findpath(int x)

{ int i;

lh[x]=1;

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

if (lx[x]+ly[i]==g[x][i]&&rh[i]==0)

{ rh[i]=1;

if (match[i]==0||findpath(match[i]))

{ match[i]=x;

return 1;

}

}

else slack[i]=min(slack[i],lx[x]+ly[i]-g[x][i]);

return 0;

}

void update()

{ int i,j,delta=INF;

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

if (!rh[i])

delta=min(delta,slack[i]);

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

{ if (lh[i])

lx[i]-=delta;

if (rh[i])

ly[i]+=delta;

}

}

int km()

{ int i,j,ans=0;

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

{ lx[i]=-INF;

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

lx[i]=max(lx[i],g[i][j]);

}

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

{ fill(slack+1,slack+n+1,INF);

while (1)

{ memset(lh,0,sizeof(lh));

memset(rh,0,sizeof(rh));

if (findpath(i))

break ;

else update();

}

}

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

ans+=g[match[i]][i];

return -ans;

}

//KM

void dfs(int x)

{ int i;

dfn[x]=++tstamp;

pos[dfn[x]]=x;

for (i=0;i<e[x].size();++i)

if (!dfn[e[x][i]])

{ par[e[x][i]]=x;

dfs(e[x][i]);

}

}

int find(int x)

{ if (f[x]==0)

return x;

int y=find(f[x]);

if (semi[dlink[x]]>semi[dlink[f[x]]])

dlink[x]=dlink[f[x]];

return f[x]=y;

}

void tarjan()

{ int i,x,y,z;

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

semi[i]=dlink[i]=i;

for (y=n;y>1;--y)

{ x=dfn[par[pos[y]]];

for (i=0;i<pre[pos[y]].size();++i)

{ z=dfn[pre[pos[y]][i]];

find(z);

semi[y]=min(semi[y],semi[dlink[z]]);

}

dom[semi[y]].push\_back(y);

f[y]=x;

for (i=0;i<dom[x].size();++i)

{ z=dom[x][i];

find(z);

if (semi[dlink[z]]<x)

idom[z]=dlink[z];

else idom[z]=x;

}

dom[x].clear();

}

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

{ if (idom[i]!=semi[i])

idom[i]=idom[idom[i]];

dom[idom[i]].push\_back(i);

}

idom[1]=0;

}

//Dominator Tree

void init\_dist()

{ int i,h,t;

q[h=t=1]=dest;

d[dest]=1;

while (h<=t)

{ for (i=last[q[h]];i!=0;i=pre[i])

if (d[e[i]]==0)

{ d[e[i]]=d[q[h]]+1;

++gap[d[e[i]]];

q[++t]=e[i];

}

++h;

}

}

int update()

{ int i,delta=INF;

for (i=dest;i!=src;i=e[rev(r[i])])

delta=min(delta,f[r[i]]);

for (i=dest;i!=src;i=e[rev(r[i])])

{ f[r[i]]-=delta;

f[rev(r[i])]+=delta;

}

return delta;

}

void relabel(int& p)

{ int i,t=n+3;

for (i=last[p];i!=0;i=pre[i])

if (f[i]>0)

t=min(t,d[e[i]]+1);

--gap[d[p]];

++gap[t];

d[p]=t;

if (p!=src)

p=e[rev(r[p])];

}

int sap()

{ int i,j,ans;

init\_dist();

memcpy(cur,last,sizeof(cur));

i=src,ans=0;

while (d[src]<n)

{ if (i==dest)

{ ans+=update();

i=src;

}

for (j=cur[i];j!=0;j=pre[j])

if (f[j]>0&&d[e[j]]+1==d[i])

{ cur[i]=j;

r[e[j]]=j;

i=e[j];

break ;

}

if (j==0)

{ if (gap[d[i]]==1)

break ;

cur[i]=last[i];

relabel(i);

}

}

return ans;

}

//ISAP

bool find\_ring()

{ int i,j,k,c=0;

memset(bel,0,sizeof(bel));

memset(pre,0,sizeof(pre));

memset(h,0,sizeof(h));

fill(min\_adv+1,min\_adv+n+1,INF);

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

if (e[i].w<min\_adv[e[i].v]&&e[i].u!=e[i].v)

{ pre[e[i].v]=e[i].u;

min\_adv[e[i].v]=e[i].w;

}

min\_adv[root]=0;

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

if (min\_adv[i]>INF-2)

throw(int(0));

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

{ ans+=min\_adv[i];

for (j=i;h[j]!=i&&bel[j]==0&&j!=root;j=pre[j])

h[j]=i;

if (j!=root&&bel[j]==0)

{ bel[j]=++c;

for (k=pre[j];k!=j;k=pre[k])

bel[k]=c;

}

}

if (c==0)

return false;

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

if (bel[i]==0)

bel[i]=++c;

n=c;

return true;

}

double edmonds()

{ double dv;

int i;

root=1;

ans=0;

while (find\_ring())

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

{ dv=min\_adv[e[i].v];

e[i].u=bel[e[i].u];

e[i].v=bel[e[i].v];

if (e[i].u!=e[i].v)

e[i].w-=dv;

}

root=bel[root];

}

return ans;

}

//Minimum arborescence

**字符串：**

void getnext()

{ int i,p,h,t;

q[h=t=1]=0;

while (h<=t)

{ p=trie[trie[q[h]].par].next;

do

{ if (trie[p].ch[trie[q[h]].chr-'a']&&p!=trie[q[h]].par)

{ trie[q[h]].next=trie[p].ch[trie[q[h]].chr-'a'];

break ;

}

else p=trie[p].next;

} while (p);

if (trie[p].ch[trie[q[h]].chr-'a']&&p!=trie[q[h]].par)

trie[q[h]].next=trie[p].ch[trie[q[h]].chr-'a'];

if (q[h])

add(trie[q[h]].next,q[h]);

for (i=0;i<26;++i)

if (trie[q[h]].ch[i])

q[++t]=trie[q[h]].ch[i];

++h;

}

}

//AC Automaton & Fail Tree

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;

}

}

}

//Extended KMP

void getnext()

{ int i,j;

i=0,j=1;

next[1]=0;

while (j<=m)

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

{ ++i,++j;

next[j]=i;

}

else i=next[i];

}

int kmp(char s[],int n)

{ int i,j,ans=0;

i=j=1;

while (i<=n&&j<=m)

{ if (j==0||s[i]==st[j])

++i,++j;

else j=next[j];

if (j>m)

++ans,j=next[j];

}

return ans;

}

//KMP

void init()

{ int i;

st[1]='#';

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

{ st[i\*2+1]='#';

st[i\*2]=t[i];

}

n=n\*2+1;

st[0]='%',st[n+1]='$';

}

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;

}

//Manacher

template<typename T>

void radix(int a[],int b[],T s[],int m,int n)

{ int i;

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

h[i]=0;

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

++h[s[a[i]]];

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

h[i]+=h[i-1];

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

b[h[s[a[i]]]--]=a[i];

}

void init\_sa()

{ int i,j;

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

rank[i]=i;

radix(rank,sa,st,255,n);

rank[sa[1]]=1;

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

if (st[sa[i]]!=st[sa[i-1]])

rank[sa[i]]=rank[sa[i-1]]+1;

else rank[sa[i]]=rank[sa[i-1]];

for (i=1;i<=n;i\*=2)

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

{ a[j]=rank[j];

if (i+j<=n)

b[j]=rank[i+j];

else b[j]=0;

sa[j]=j;

}

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

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

rank[sa[1]]=1;

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

if (a[sa[j]]!=a[sa[j-1]]||b[sa[j]]!=b[sa[j-1]])

rank[sa[j]]=rank[sa[j-1]]+1;

else rank[sa[j]]=rank[sa[j-1]];

if (rank[sa[n]]==n)

break ;

}

}

void calc\_height()

{ int i,p=0;

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

{ if (p)

--p;

if (rank[i]!=1)

while (st[i+p]==st[sa[rank[i]-1]+p]&&st[i+p]!='|')

++p;

height[rank[i]]=p;

}

}

//Suffix Array

**计算几何**

bool fequal(double a,double b)

{ return fabs(a-b)<prec;

}

int fcmp(double x)

{ if (fabs(x)<prec)

return 0;

return x>0?1:-1;

}

struct point

{ double x,y;

point(double x=0,double y=0):x(x),y(y) {}

double length()

{ return sqrt(x\*x+y\*y);

}

};

struct line

{ point p1,p2;

double a,b,c;

double ang;

line() {}

line(point px,point py)

{ a=py.y-px.y;

b=px.x-py.x;

c=-a\*px.x-b\*px.y;

p1=px,p2=py;

point v=p2-p1;

ang=atan2(v.y,v.x);

}

void angle()

{ point px=p2-p1;

ang=atan2(px.y,px.x);

}

point vec()

{ return p2-p1;

}

};

struct polygon

{ vector<point> pt;

vector<line> li;

double ar;

double area();

void getline();

polygon()

{ pt.clear();

li.clear();

ar=0;

}

};

/\* Operators \*/

point operator +(point a,point b)

{ return point(a.x+b.x,a.y+b.y);

}

point operator -(point a,point b)

{ return point(a.x-b.x,a.y-b.y);

}

point operator \*(point a,double b)

{ return point(a.x\*b,a.y\*b);

}

point operator \*(double b,point a)

{ return point(a.x\*b,a.y\*b);

}

point operator /(point a,double b)

{ return point(a.x/b,a.y/b);

}

bool operator <(point a,point b)

{ return a.x<b.x||(fcmp(a.x-b.x)==0&&a.y<b.y);

}

bool operator ==(point a,point b)

{ return (fcmp(a.x-b.x)==0&&fcmp(a.y-b.y)==0);

}

bool operator <(line a,line b)

{ return a.angle()<b.angle();

}

/\* Common operations \*/

vector normal(vector x)

{ double l=x.length();

return vector(-x.y/l,x.x/l);

}

double cp(point a,point b)

{ return a.x\*b.y-a.y\*b.x;

}

double dp(point a,point b)

{ return a.x\*b.x+a.y\*b.y;

}

double dist2(point a,point b)

{ return sqr(a.x-b.x)+sqr(a.y-b.y);

}

double dist(point a,point b)

{ return sqrt(sqr(a.x-b.x)+sqr(a.y-b.y));

}

double torad(double ang)//degree to radian

{ return pi\*(ang/180);

}

point rotate(point a,double rad)

{ return point(a.x\*cos(rad)-a.y\*sin(rad),a.x\*sin(rad)+a.y\*cos(rad));

}

point project(point& p,point &a,point &b)//Projection of p on line ab

{ vector v=b-a;

return a+v\*(dp(v,p-a)/dp(v,v));

}

polygon make\_polygon(point p[],int n)

{ polygon poly;

int i;

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

poly.pt.push\_back(p[i]);

poly.getline();

return poly;

}

double polygon::area()

{ int i;

double area=0;

for (i=1;i<pt.size();++i)

area+=cp(pt[i-1],pt[i]);

area+=cp(pt[i-1],pt[0]);

ar=area;

return area/2;

}

void polygon::getline()

{ int i;

for (i=1;i<pt.size();++i)

li.push\_back(line(pt[i-1],pt[i]));

li.push\_back(line(pt[pt.size()-1],pt[0]));

}

void make\_polygon(line poly[],int n)//Generate the points through the lines,use after half-plane intersection

{ line lt[605];

int i;

memcpy(lt,poly,sizeof(lt));

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

poly[i].p2=poly[i+1].p1=intersect\_point(lt[i],lt[i+1]);

poly[n].p2=poly[1].p1=intersect\_point(lt[n],lt[1]);

}

point unit\_vec(point a,point b)

{ point p=b-a;

return p/p.length();

}

/\* Relations \*/

double get\_parameter(point a,point b,point p)//parameter of P on vector AB

{ if (a==p)

return 0;

if (b==p)

return 1;

if (fcmp(a.x-b.x)==0)

return (p.y-a.y)/(b.y-a.y);

else return (p.x-a.x)/(b.x-a.x);

}

point intersect\_point(point p,point v,point q,point w)//of lines. Ensure the point exist.

{ point u=p-q;

double t=cp(w-q,u)/cp(v-p,w-q);

return p+(v-p)\*t;

}

point intersect\_point(half\_plane a,half\_plane b)

{ vector u=a.p1-b.p1;

double t=cp(b.dir(),u)/cp(a.dir(),b.dir());

return a.p1+a.dir()\*t;

}

line midnormal(point a,point b)

{ point mid=(a+b)/2;

return line(mid,mid+rotate(mid-a,pi/2));

}

double directed\_dis(line li,point p)

{ return cp(p-li.p1,li.p2-li.p1)/li.length();

}

bool in\_ang(point pl,point p0,point pr,point x)

{ return point\_on\_ray(x,p0,pl)||point\_on\_ray(x,p0,pr)||(point\_onleft(p0,pr,x)&&!point\_onleft(p0,pl,x));

}

bool point\_on\_left(point a,point b,point p)

{ return fcmp(cp(b-a,p-a))>0;

}

bool point\_on\_right(point a,point b,point p)

{ return fcmp(cp(b-a,p-a))<0;

}

bool point\_on\_seg(point p,point a,point b)

{ if (fcmp(cp(p-a,p-b))==0)

{ if (fcmp(p.x-min(a.x,b.x))>=0)

if (fcmp(p.x-max(a.x,b.x))<=0)

if (fcmp(p.y-min(a.y,b.y))>=0)

if (fcmp(p.y-max(a.y,b.y))<=0)

return 1;

return 0;

}

return 0;

}

bool point\_on\_ray(point p,point a,point b)

{ if (p==a)

return 1;

if (p==b)

return 1;

if (cp(p-a,p-b)==0)

{ if (b.x>a.x&&p.x>a.x)

return 1;

if (b.x<a.x&&p.x<a.x)

return 1;

return 0;

}

return 0;

}

point sym\_point(point p,line l)

{ point p2;

double d;

d=l.a\*l.a+l.b\*l.b;

p2.x=(l.b\*l.b\*p.x-l.a\*l.a\*p.x-2\*l.a\*l.b\*p.y-2\*l.a\*l.c)/d;

p2.y=(l.a\*l.a\*p.y-l.b\*l.b\*p.y-2\*l.a\*l.b\*p.x-2\*l.b\*l.c)/d;

return p2;

}

int seg\_colinear(point a1,point a2,point b1,point b2)

{ if (fcmp(cp(a1-b1,a1-b2))==0&&fcmp(cp(a2-b1,a2-b2))==0)

return 1;

else return 0;

}

int seg\_inprop\_intersect(point a1,point a2,point b1,point b2)

{ double c1,c2,c3,c4;

if (seg\_colinear(a1,a2,b1,b2))

{ if (fcmp(max(a1.x,a2.x)-min(b1.x,b2.x))<0)

return 0;

if (fcmp(max(b1.x,b2.x)-min(a1.x,a2.x))<0)

return 0;

return 1;

}

c1=cp(a2-a1,b1-a1);

c2=cp(a2-a1,b2-a1);

c3=cp(b2-b1,a1-b1);

c4=cp(b2-b1,a2-b1);

return fcmp(c1)\*fcmp(c2)<=0&&fcmp(c3)\*fcmp(c4)<=0;

}

bool seg\_prop\_intersect(point& a1,point& a2,point& b1,point& b2)

{ double c1,c2,c3,c4;

c1=cp(a2-a1,b1-a1);

c2=cp(a2-a1,b2-a1);

c3=cp(b2-b1,a1-b1);

c4=cp(b2-b1,a2-b1);

return fcmp(c1)\*fcmp(c2)<0&&fcmp(c3)\*fcmp(c4)<0;

}

int seg\_intersect\_term(point a1,point a2,point b1,point b2)

{ if (seg\_colinear(a1,a2,b1,b2))

{ if (a1==b1&&!point\_on\_seg(a1,b1,b2))

return 1;

if (a2==b1&&!point\_on\_seg(a2,b1,b2))

return 1;

if (a1==b2&&!point\_on\_seg(a1,b1,b2))

return 1;

if (a2==b2&&!point\_on\_seg(a2,b1,b2))

return 1;

return 0;

}

if (seg\_inprop\_intersect(a1,a2,b1,b2))

return 1;

return 0;

}

bool point\_in\_polygon(point &p,polygon &g)

{ int i,wn=0,k,d1,d2;

int n=g.pt.size();

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

{ point &p1=g.pt[i];

point &p2=g.pt[(i+1)%n];

if (p1==p||p2==p||point\_on\_seg(p,p1,p2))

return true;

k=cp(p2-p1,p-p1);

d1=p1.y-p.y;

d2=p2.y-p.y;

if (k>0&&d1<=0&&d2>0)

++wn;

if (k<0&&d2<=0&&d1>0)

--wn;

}

if (wn!=0)

return true;

return false;

}

bool point\_in\_polygon(point &p,line g[],int n)

{ int i,wn=0;

double k,d1,d2;

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

{ point &p1=g[i].p1;

point &p2=g[i].p2;

if (p1==p||p2==p||point\_on\_seg(p,p1,p2))

return true;

k=cp(p2-p1,p-p1);

d1=p1.y-p.y;

d2=p2.y-p.y;

if (fcmp(k)>0&&fcmp(d1)<=0&&fcmp(d2)>0)

++wn;

if (fcmp(k)<0&&fcmp(d2)<=0&&fcmp(d1)>0)

--wn;

}

if (wn!=0)

return true;

return false;

}

bool point\_in\_circle(point p,double r,point center=point())//center located at (0,0) as default

{ return fcmp(dist2(p,center)-r\*r)<=0;

}

bool point\_on\_circle(point p,double r,point center=point())//center located at (0,0) as default

{ return fcmp(dist2(p,center)-r\*2)==0;

}

bool line\_circle\_inter(point p1,point p2,double r,point ct=point())//both p1 and p2 are not inside the circle(other situation untested)

{ point p=point\_project\_on\_line(p1,p2,ct);

if (point\_in\_circle(p,r,ct))

return true ;

return false ;

}

bool seg\_circle\_inter(point p1,point p2,double r)//center located at (0,0)

{ if (point\_in\_circle(p1,r)!=point\_in\_circle(p2,r))

return true ;

point p=point\_project\_on\_line(p1,p2,point());

if (!point\_in\_circle(p,r))

return false ;

if (fcmp(cp(p1,p)\*cp(p2,p))<=0)

return true ;

return false ;

}

bool seg\_circle\_prop\_inter(point p1,point p2,point pt,double r)

{ if (point\_prop\_in\_circle(p1,r,pt)!=point\_prop\_in\_circle(p2,r,pt))

return true;

point p=point\_project\_on\_line(p1,p2,pt);

if (!point\_in\_circle(p,r,pt))

return false;

if (fcmp(cp(p1,p)\*cp(p2,p))<=0)

return true;

return false;

}

pair<point,point> line\_circle\_inter\_point(point p1,point p2,double r,point ct=point())//intersect guaranteed,ordered as p1,ret.first,ret.second,p2 by polar angle

{ point p=point\_project\_on\_line(p1,p2,ct);

point v=unit\_vec(p1,p2);

double l=sqrt(sqr(r)-sqr(dist(ct,p)));

return make\_pair(p-l\*v,p+l\*v);

}

pair<point,point> seg\_circle\_inter\_point(point p1,point p2,double r,point ct=point())//intersect guaranteed,ordered as p1,ret.first,ret.second,p2 by polar angle

{ point p=point\_project\_on\_line(p1,p2,ct);

point v=unit\_vec(p1,p2);

double l=sqrt(sqr(r)-sqr(dist(ct,p)));

point pa=p-l\*v,pb=p+l\*v;

if (point\_in\_circle(p1,r,ct)!=point\_in\_circle(p2,r,ct))

if (point\_on\_seg(pa,p1,p2))

if ((pa==p1||pa==p2)&&point\_in\_circle(pb,r,ct)&&point\_on\_seg(pb,p1,p2))

return make\_pair(pb,pb);

else return make\_pair(pa,pa);

else return make\_pair(pb,pb);

if (fcmp(cp(pa,p)\*cp(pb,p))<=0)

return make\_pair(pa,pb);

else throw(make\_pair(pa,pb));

}

void circle\_inter\_point(point p1,double r1,point p2,double r2,vector<point>& v)

{ double d=dist(p1,p2);

if (fcmp(d-r1-r2)>0)

return ;

if (fcmp(d-r1-r2)==0)

{ v.push\_back((p2-p1)\*r1/d+p1);

return ;

}

if (r1<r2)

swap(r1,r2),swap(p1,p2);

if (fcmp(d-(r1-r2))<0)

return ;

if (fcmp(d-(r1-r2))==0)

{ v.push\_back((p2-p1)\*r2/r1+p2);

return ;

}

double c=(sqr(r1)+sqr(d)-sqr(r2))/(2\*d);

double h=sqrt(sqr(r1)-sqr(c));

point p=p1+(p2-p1)\*(c/d);

v.push\_back(p+normal(p-p1)\*h);

v.push\_back(p-normal(p-p1)\*h);

}

double sector\_area(point p1,point p2,double r)//undirected area, center located at (0,0)

{ p1=p1\*r/dist(p1,point());

p2=p2\*r/dist(p2,point());

double ang=abs(atan2(p1)-atan2(p2));

if (fcmp(ang-pi)>0)

ang=2\*pi-ang;

if (fcmp(ang+pi)<0)

ang=-2\*pi-ang;

return abs(r\*r\*ang/2);

}

double triangle\_circle\_inter(point p1,point p2,double r)//triangle formed by (0,0),p1,p2

{ pair<point,point> inter\_pair;

point p3,p4;

if (point\_in\_circle(p1,r)&&point\_in\_circle(p2,r))

return cp(p1,p2)/2;

if (!point\_in\_circle(p1,r)&&!point\_in\_circle(p2,r))

if (!line\_circle\_inter(p1,p2,r))

return sector\_area(p1,p2,r)\*fcmp(cp(p1,p2));

else

{ if (!seg\_circle\_inter(p1,p2,r))

return sector\_area(p1,p2,r)\*fcmp(cp(p1,p2));

inter\_pair=line\_circle\_inter\_point(p1,p2,r);

p3=inter\_pair.first,p4=inter\_pair.second;

return cp(p3,p4)/2+(sector\_area(p1,p3,r)+sector\_area(p4,p2,r))\*fcmp(cp(p1,p2));

}

p3=seg\_circle\_inter\_point(p1,p2,r).first;

if (point\_in\_circle(p1,r))

return cp(p1,p3)/2+sector\_area(p3,p2,r)\*fcmp(cp(p1,p2));

else return sector\_area(p1,p3,r)\*fcmp(cp(p1,p2))+cp(p3,p2)/2;

}

double polygon\_circle\_inter(polygon poly,double r)//center located at (0,0)

{ double ans=0;

int i;

for (i=0;i<poly.li.size();++i)

ans+=triangle\_circle\_inter(poly.li[i].p1,poly.li[i].p2,r);

return abs(ans);

}

int convex\_hull(point p[],int n,point ch[])

{ int i,k,m=0;

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

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

{ while (m>=2&&cp(ch[m]-ch[m-1],p[i]-ch[m-1])<=0)

--m;

ch[++m]=p[i];

}

k=m;

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

{ while (m>k&&cp(ch[m]-ch[m-1],p[i]-ch[m-1])<=0)

--m;

ch[++m]=p[i];

}

if (n>1)

--m;

return m;

}

int half\_plane\_intersect(line l[],int n,line poly[])

{ point p[605];

line q[605];

int i,h,t;

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

h=t=1;

q[1]=l[1];

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

{ while (h<t&&!point\_on\_left(l[i],p[t-1]))

--t;

while (h<t&&!point\_on\_left(l[i],p[h]))

++h;

q[++t]=l[i];

if (fcmp(cp(q[t].p2-q[t].p1,q[t-1].p2-q[t-1].p1))==0)

{ --t;

if (point\_on\_left(q[t],l[i].p1))

q[t]=l[i];

}

if (h<t)

p[t-1]=intersect\_point(q[t-1],q[t]);

}

while (h<t&&!point\_on\_left(q[h],p[t-1]))

--t;

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

poly[i-h+1]=q[i];

return t-h+1;

}//better use make\_polygon() to get the points

map<point,map<int,int> > pos;

map<point,vector<int> > h;

void prep\_edge()

{ int i;

map<point,vector<int> >::iterator it;

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

h[li[i].p1].push\_back(i);

for (it=h.begin();it!=h.end();++it)

{ sort(it->second.begin(),it->second.end(),cmp\_li);

for (i=0;i<it->second.size();++i)

pos[it->first][li[(it->second)[i]].pos]=i;

}

}

void dfs\_subdiv(int x)

{ int next;

vis[x]=cur;

poly[cur].pt.push\_back(li[x].p1);

poly[cur].li.push\_back(li[x]);

if (pos[li[x].p2][li[x].pos]==0)

next=h[li[x].p2][h[li[x].p2].size()-1];

else next=h[li[x].p2][pos[li[x].p2][li[x].pos]-1];

if (vis[next]==cur)

return ;

else dfs\_subdiv(next);

}

void planar\_subdiv()

{ int i;

prep\_edge();

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

if (vis[i]==0)

{ ++cur;

dfs\_subdiv(i);

}

n=0;

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

if (poly[i].area()>0)

subdiv[++n]=poly[i];

}

//PSLG

**数论**

inline long long multi\_mod(long long a,long long b,long long c)

{ long long ret=0;

if (a>=c)

a%=c;

if (b>=c)

b%=c;

while (b)

{ if (b&1)

{ ret+=a;

if (ret>=c)

ret-=c;

}

a<<=1;

if (a>=c)

a-=c;

b>>=1;

}

return ret;

}

void ex\_gcd(long long a,long long b,long long &x,long long &y)

{ if (b==0)

{ x=1,y=0;

return ;

}

else

{ ex\_gcd(b,a%b,y,x);

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

}

}

inline long long inverse(long long x)

{ long long ans,y;

ex\_gcd(x,lim,ans,y);

if (ans<0)

ans+=lim;

return ans;

}

inline bool miller\_rabin(long long a,long long n)

{ long long d,t;

if (n==2)

return true;

if (n==1||(n&1)==0)

return false;

d=n-1;

while ((d&1)==0)

d>>=1;

t=qp(a,d,n);

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

{ t=multi\_mod(t,t,n);

d<<=1;

}

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

}

inline bool prime\_test(long long n)

{ int i;

for (i=0;i<prime\_base\_size;++i)

{ if (prime\_base[i]==n)

return true;

if (!miller\_rabin(prime\_base[i],n))

return false;

}

return true;

}

inline long long pollard\_rho(long long n,long long c)

{ long long x,y,d;

int loop\_cnt=0;

x=2,y=2,d=1;

while (d==1&&loop\_cnt<1000)

{ x=(multi\_mod(x,x,n)+c)%n;

y=(multi\_mod(y,y,n)+c)%n;

y=(multi\_mod(y,y,n)+c)%n;

d=gcd(llabs(x-y),n);

++loop\_cnt;

}

return d;

}

void sieve()

{ int i,j;

phi[1]=1,miu[1]=1;

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

{ if (!h[i])

{ prime[++pc]=i;

phi[i]=i-1;

miu[i]=-1;

}

for (j=1;j<=pc&&prime[j]\*i<=n;++j)

{ h[i\*prime[j]]=1;

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

{ phi[i\*prime[j]]=phi[i]\*prime[j];

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

break ;

}

else

{ phi[i\*prime[j]]=phi[i]\*phi[prime[j]];

miu[i\*prime[j]]=-miu[i];

}

}

}

}

LL work(LL n,LL m)

{ LL i,p,ans=0;

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

{ p=min(n/(n/i),m/(m/i));

ans+=(sum[p]-sum[i-1])\*(LL)(n/i)\*(m/i);

}

return ans;

}

LL crt(LL a[],const int m[],int n)

{ LL Mi,ans=0,x,y,M;

int i;

M=1;

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

M\*=m[i];

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

{ Mi=M/m[i];

ex\_gcd(m[i],Mi,x,y);

ans+=y\*Mi\*a[i];

ans%=M;

}

return (ans+M)%M;

}

pair<LL,LL> fact(LL n)

{ if (n<=1)

return make\_pair(1LL,0LL);

pair<LL,LL> ret;

LL m=n/lim,s,x,i;

ret=fact(m);

ret.second+=m\*ep;

for (i=lim\*m+1;i<=n;++i)

{ x=i;

while (x%p==0)

++ret.second,x/=p;

ret.first=ret.first\*x%lim;

}

s=1;

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

{ x=i;

while (x%p==0)

ret.second+=m,x/=p;

s=s\*x%lim;

}

s=qp(s,m);

ret.first=ret.first\*s%lim;

return ret;

}

LL comb(LL k,LL n)

{ pair<LL,LL> a,b,c,ret;

LL cnt=0;

a=fact(n);

b=fact(k);

c=fact(n-k);

cnt+=a.second;

cnt-=b.second;

cnt-=c.second;

if (cnt>=ep)

return 0;

else return a.first\*qp(p,cnt)%lim\*qp((LL)b.first\*c.first%lim,lim/p\*(p-1)-1)%lim;

}

LL work()

{ int i,x=lim;

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

if (x%i==0)

{ sdiv[++dc]=i;

val[dc]=1;

while (x%i==0)

x/=i,++cnt[dc],val[dc]\*=i;

}

if (x>1)

sdiv[++dc]=x,cnt[dc]=1,val[dc]=x;

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

{ lim=val[i];

p=sdiv[i];

ep=cnt[i];

res[i]=comb(k,n);

}

return crt(res,val,dc);

}

//Binomial modulo lim

inline long long c(long long m,long long n)

{ return fact[m]\*qp(fact[n]\*fact[m-n]%lim,lim-2)%lim;

}

long long lucas(long long m,long long n)

{ long long ans=1;

int i;

while (m!=0)

{ if (m%lim<n%lim)

return 0;

ans\*=c(m%lim,n%lim);

ans%=lim;

m/=lim,n/=lim;

}

return ans%lim;

}

//Lucas