1. Aho

```
char s[N];int node[N][27];int vis[N];int backnode[N];int cnt[N];pii endpos[N];int id;char st[N];int ans[N];vector<int>pos[N];int len[N];int BIG(int k){string ss(s),t(st);ss=t+"#"+ss;int n=ss.size(),m=t.size();vector<int>p(n);for(int i=1;i<n;i++){int j=p[i-1];while(j>0 && ss[i]!=ss[j]){j=p[j-1];}if(ss[i]==ss[j])j++;p[i]=j;}vector<int>v;for(int i=m+m;i<n;i++){if(p[i]==m) v.emplace_back(i);}if(v.size()<k)return-1;int ans=1e9;for(int i=k-1;i<v.size();i++){ans=min(ans,v[i]-v[i-k+1]);}return ans+m;}inline void init(){id=0;for(int i=0;i<26;i++)node[id][i]=0;}inline int newnode(){id++;for(int i=0;i<26;i++){node[id][i]=0;}backnode[id]=0;cnt[id]=0;endpos[id]=pii(-1,-1);return id;}inline void Insert(int id,int k){int u=0;int n=strlen(st);for(int i=0;i<n;i++){int x=st[i]-'a';if(!node[u][x])node[u][x]=newnode();u=node[u][x];}endpos[u]=pii(id,k);}inline void AhoCorasik(){queue<int>q;for(int i=0;i<26;i++){if(node[0][i]){q.push(node[0][i]);backnode[node[0][i]]=0;}}while(!q.empty()){int u=q.front();int w=backnode[u];q.pop();for(int i=0;i<26;i++){int v=node[u][i];if(v){q.push(v);backnode[v]=node[backnode[u]][i];}else {node[u][i]=node[backnode[u]][i];}}}}void f(){int u=0;for(int i=0;s[i];i++){int v=u=node[u][s[i]-'a'];while(v!=0){if(endpos[v].ff!=-1)pos[v].push_back(i);v=backnode[v];}}for(int i=1;i<=id;i++){if(endpos[i].ff!=-1){int idx=endpos[i].ff;int k=endpos[i].ss;int val=INT_MAX;if(pos[i].size()<k)ans[idx]=-1;else{for(int a=0,b=k-1;b<pos[i].size();a++,b++)val=min(val,pos[i][b]-pos[i][a]+len[idx]);ans[idx]=val;}}}}
```

2. BIGINT

```
struct Bigint{string a;int sign;Bigint(){} Bigint(stringb){(*this)=b;}int size(){return a.size();}Bigint inverseSign(){sign*=-1;return(*this);}Bigint normalize(int newSign){for(int i=a.size()-1;i>0&&a[i]=='0';i--)a.erase(a.begin()+i);sign=(a.size()==1&&a[0]=='0')?1:newSign;return(*this);}void operator=(string b){ a=b[0]=='-'?b.substr(1):b;reverse(a.begin(),a.end());this->normalize(b[0]=='-'?-1:1);}bool operator<(const Bigint& b)const{if(sign!=b.sign)return sign<b.sign;if(a.size()!=b.a.size())return sign==1?a.size()< b.a.size():a.size()>b.a.size();for(int i=a.size()-1;i>=0;i--)if(a[i]!=b.a[i])return sign==1?a[i]<b.a[i]:a[i]>b.a[i];return false;}bool operator==(constBigint&b)const{return a==b.a && sign== b.sign;}Bigint operator+(Bigint b){if(sign!=b.sign)return(* this)-b.inverseSign();Bigint c;for(int i=0,carry=0;i<a.size()||i<b.size()||carry;i++){carry+=(i<a.size()?a[i]-48:0)+(i<b.a.size()?b.a[i]-48:0);c.a+=(carry%10+48);carry/=10;}return c.normalize(sign);}Bigint operator-(Bigint b){if(sign!=b.sign)return(*this)+b.inverseSign();int s=sign;sign=b.sign=1;if((*this)<b)return ((b-(*this)).inverseSign()).normalize(-s);Bigint c;for(int i=0,borrow=0;i<a.size();i++){borrow=a[i]-borrow-(i<b.size()?b.a[i]:48);c.a+=borrow>=0?borrow+48:borrow+58;borrow=borrow>=0?0:1;}return c.normalize(s);}Bigint operator*(Bigint b){Bigint c("0");for(int i=0,k=a[i]-48;i<a.size();i++,k=a[i]-48){while(k--)c=c+b;b.a.insert(b.a.begin(),'0');}return c.normalize(sign*b.sign);}Bigint operator/(Bigint b){if(b.size()==1&&b.a[0]=='0')b.a[0]/=(b.a[0]-48);Bigint c("0"),d;for(int j=0;j<a.size();j++)d.a+="0";int dSign=sign*b.sign;b.sign=1;for(int i=a.size()-1;i>=0;i--){c.a.insert(c.a.begin(),'0');bc=c+a.substr(i,1);while(!(c<b))c=c-b,d.a[i]++;}return d.normalize(dSign);}Bigint operator%(Bigint b){if(b.size()==1&& b.a[0]=='0')b.a[0]/=(b.a[0]-48);Bigint c("0");b.sign=1;for(int i=a.size()-1;i>=0;i--){c.a.insert(c.a.begin(),'0');c=c+a.substr(i,1);while(!(c<b))c=c-b;}return c.normalize(sign);}void print(){if(sign==-1)putchar('-');for(int i=a.size()-1;i>=0;i--)putchar(a[i]);}};
```

3. CHT

```
/*add lines with -m and -b and return -ans to make this code working for minimums.*/const ll is_query=-(1LL<<62);struct Line{ll m,b;Mutable function<const Line*()>succ;bool operator<(const Line&rhs)const{if(rhs.b !=is_query)return m<rhs.m;const Line*s=succ();if(!s)return 0;ll x=rhs.m;return b-s->b<(s->m-m)*x;}};struct HullDynamic:public multiset<Line>{bool bad(iterator y){auto z=next(y);if(y==begin()){if(z==end())return 0;return y->m==z->m&&y->b<=z->b;}auto x=prev(y);if(z==end())return y->m==x->m&&y->b<=x->b;return((1.0*(x->b-y->b)*(z->m-y->m))>=(1.0*(y->b-z->b)*(y->m-x->m)));}void insert_line(ll m,ll b){auto y=insert({m,b});y->succ=[=]{return next(y)==end()?0:&*next(y);};if(bad(y)){erase(y);return;}while(next(y)!=end()&&bad(next(y)))erase(next(y));while(y!=begin()&&bad(prev(y)))erase(prev(y));}ll eval(ll x){auto l=*lower_bound((Line){x,is_query});return l.m*x+l.b;}};
```

4. Dinic

```
struct FlowEdge{int v,u;long long cap,flow=0;FlowEdge(int v,int u,long long cap):v(v),u(u),cap(cap){}};struct Dinic{const long long flow_inf=1e18;vector<FlowEdge>edges;vector<vector<int>>adj,usedInFlow;int n,m=0;int s,t;vector<int>level,ptr;queue<int>q;Dinic(int n,int s,int t):n(n),s(s),t(t){adj.resize(n);level.resize(n);ptr.resize(n);usedInFlow.resize(n);}void add_edge(int v,int u,long long ca
```

```
p){edges.emplace_back(v,u,cap);edges.emplace_back(u,v,0);adj[v].push_back(m);adj[u].push_back(m+1);mp[v][u]=m;m+=2;}bool bfs(){while(!q.empty()){int v=q.front();q.pop();for(int id:adj[v]){if(edges[id].cap-edges[id].flow<1)continue;if(level[edges[id].u]!=-1)continue;level[edges[id].u]=level[v]+1;q.push(edges[id].u);}}return level[t]!=-1;}long long dfs(int v,long long pushed){if(pushed==0)return 0;if(v==t)return pushed;for(int&cid=ptr[v];cid<(int)adj[v].size();cid++){int id=adj[v][cid];int u=edges[id].u;if(level[v]+1!=level[u]||edges[id].cap-edges[id].flow<1)continue;long long tr=dfs(u,min(pushed,edges[id].cap-edges[id].flow));if(tr==0)continue;edges[id].flow+=tr;edges[id^1].flow-=tr;return tr;}return 0;}long long flow(){long long f=0;while(true){fill(level.begin(),level.end(),-1);level[s]=0;q.push(s);if(!bfs())break;fill(ptr.begin(),ptr.end(),0);while(long long pushed=dfs(s,flow_inf)){f+=pushed;}}return f;}};
```

5. Dynamic Hull

```
const int N=3e5+9;typedef set<pair<int,int>>::iterator iter;struct PT{int x,y;PT():x(0),y(0){}PT(int x,int y):x(x),y(y){}PT(constPT&rhs):x(rhs.x),y(rhs.y){}PT(constiter&p):x(p->first),y(p->second){}PT operator-(const PT& rhs)const{return PT(x-rhs.x,y-rhs.y);}};long long cross(PT a,PT b){return(long long)a.x* b.y-(long long)a.y*b.x;}inline int inside(set<pair<int,int>>&hull,const PT&p){//border inclusive int x=p.x,y=p.y;iter p1=hull.lower_bound(make_pair(x,y));if(p1==hull.end())return 0;if(p1->first==x)return p1!= hull.begin()&&y<=p1->second;if(p1==hull.begin())return 0;iter p2(p1--);return cross(p-PT(p1),PT(p2)-p)>=0;}inline void del(set<pair<int,int>>&hull,iter it,long long&scross){if(hull.size()==1){hull.erase(it);return;}if(it==hull.begin()){iter p1=it++;scross-=cross(p1,it);hull.erase(p1);return;}iter p1=--it,p2=++it;if(++it==hull.end()){scross-=cross(p1,p2);hull.erase(p2);return;}scross-=cross(p1,p2)+cross(p2,it)-cross(p1,it);hull.erase(p2);}inline void add(set<pair<int,int>>&hull,iter it,long long&scross){if(hull.size()==1)return;if(it==hull.begin()){iter p1=it++;scross+=cross(p1,it);return;}iter p1=-it,p2=++it;if(++it==hull.end()){scross+=cross(p1,p2);return;}scross+=cross(p1,p2)+ cross(p2,it)-cross(p1,it);}inline void add(set<pair<int,int>>&hull,const PT&p,long long&scross){//nocollinearPTs if(inside(hull,p))return;int x=p.x,y=p.y;iter pnt= hull.insert(make_pair(x,y)).first,p1,p2;add(hull,pnt,scross);for(;;del(hull,p2,scross)){p1 =pnt;if(++p1==hull.end())break;p2=p1;if(++p1==hull.end())break;if(cross(PT(p2)-p,PT(p1)-p)<0)break;}for(;;del(hull,p2,scross)){if((p1=pnt)==hull.begin())break;if(--p1==hull.begin())break;p2=p1--;if(cross(PT(p2)-p,PT(p1)-p)>0)break;}}int main(){long long ucross=0,dcross=0;set<pair<int,int>>uhull,dhull;PT p[]={PT(0,0),PT(3,0),PT(3,3),PT(0,3),PT(0, 1),PT(0,2),PT(3,1),PT(3, 2)};for(int i=0;i<5;i++){add(uhull,PT(+p[i].x,+p[i].y),ucross);add(dhull,PT(-p[i].x,-p[i].y),dcross);}cout<<fixed<<setprecision(1)<<"Area:"<<fabs(ucross+dcross)/2.0<<"\n";return 0;}
```

6. MCMF

```
struct MCMF{using T=ll;static const T inf=1e9+7;int n,m,s,t;bool vis[N];ll par[N],pos[N];T pot[N],dis[N],left[N];priority_queue<pair<T,int>>q;struct edge{int to,rev;T cap,cost,flow;int id;};vector<edge>ed[N];MCMF(){}MCMF(int _n){n=_n+2;for(int i=1;i<=n;i++)ed[i].clear();}void add_edge(int u,int v,T cap,T cost,int id=-1){edge a={v,(int)ed[v].size(),cap,cost,0,id};edge b={u,(int)ed[u].size(),0,-cost,0,-1};ed[u].push_back(a);ed[v].push_back(b);}T BellmanFord(int u){if(vis[u])return pot[u];if(u==s){pot[s]=0;return 0;}vis[u]=true;pot[u]=inf;for(edge e:ed[u]){edge r=ed[e.to][e.rev];if(r.flow<r.cap)pot[u]=min(pot[u],BellmanFord(e.to)+r.cost);}return pot[u];}booldijkstra(){memset(vis,0,sizeofvis);for(int i=1;i<=n;i++)dis[i]=left[i]=inf;dis[s]=0;q.push({0,s});int u,v;while(!q.empty()){u=q.top().second;q.pop();if(vis[u])continue;vis[u]=true;int ptr=0;for(edge e:ed[u]){v=e.to;T cost=e.cost+pot[u]-pot[v];if(e.flow<e.cap&&dis[u]+cost<dis[v]){dis[v]=dis[u]+cost;par[v]=u;pos[v]=ptr;left[v]=min(left[u],e.cap-e.flow);q.push(make_pair(-dis[v],v));}++ptr;}}for(int i=1;i<=n;i++)dis[i]+=(pot[i]-pot[s]);return vis[t];}pair<T,T>solve(int _s,int _t,T goal=inf){s=_s;t=_t;memset(pot,0,sizeof pot);memset(vis,0,sizeof vis);intu,v;T ans=0,cost=0, f;while(ans<goal&&dijkstra()){u=t;f=left[t];while(u!=s){v=par[u];ed[v][pos[u]].flow+=left[t];ed[u][ed[v][pos[u]].rev].flow-=left[t];u=v;}T need=goal-ans;f=min(f,need);ans+=f;cost+=f*dis[t];memcpy(pot,dis,sizeofdis);}return make_pair(ans,cost);}};
```

7. FFT

```
namespace ntt{struct num{double x,y;num(){x=y=0;}num(double x,double y):x(x),y(y){}};inline num operator+(num a,num b){return num(a.x+b.x,a.y+b.y);}inline num operator-(num a,num b){return num(a.x-b.x,a.y-b.y);}inline num operator*(num a,num b){return num(a.x*b.x-a.y*b.y,a.x*b.y+a.y*b.x);}inline num conj(num a){return num(a.x,-a.y);}int base=1;vector<num>roots={{0,0},{1,0}};vector<int>rev={0,1};const double PI=acosl(-1.0);void ensure_base(int nbase){if(nbase<=base)return;rev.resize(1<<nbase);for(int i=0;i<(1<<nbase);i++)rev[i]=(rev[i>>1]>>1)+((i&1)<<(nbase-1));roots.resize(1<<nbase);while(base<nbase){double angle=2*PI/(
```

```
1<<(base+1));for(int i=1<<(base-1);i<(1<<base);i++){roots[i<<1]=roots[i];double angle_i=angle*(2*i+1-(1<<base));roots[(i<<1)+1]=num(cos(angle_i),sin(angle_i));}base++;}}void fft(vector<num>&a,int n=-1){if(n==-1)n=a.size();assert((n&(n-1))==0);int zeros=__builtin_ctz(n);ensure_base(zeros);int shift=base-zeros;for(int i=0;i<n;i++)if(i<(rev[i]>>shift))swap(a[i],a[rev[i]>>shift]);for(int k=1;k<n;k<<=1){for(int i=0;i<n;i+=2*k){for(int j=0;j<k;j++){num z=a[i+j+k]*roots[j+k];a[i+j+k]=a[i+j]-z;a[i+j]=a[i+j]+z;}}}}vector<num>fa,fb;vector<ll>multiply(vector<ll>&a,vector<ll>&b){ll need=a.size()+b.size()-1;ll nbase=0;while((1<<nbase)<need)nbase++;ensure_base(nbase);ll sz=1<<nbase;if(sz>(ll)fa.size())fa.resize(sz);for(ll i=0;i<sz;i++){ll x=(i<(ll)a.size()?a[i]:0);ll y=(i<(ll)b.size()?b[i]:0);fa[i]=num(x,y);}fft(fa,sz);num r(0,-0.25/sz);for(ll i=0;i<=(sz>>1);i++){ll j=(sz-i)&(sz-1);num z=(fa[j]*fa[j]-conj(fa[i]*fa[i]))*r;if(i!=j)fa[j]=(fa[i]* fa[i]-conj(fa[j]*fa[j]))*r;fa[i]=z;}fft(fa,sz);vector<ll>res(need);for(ll i=0;i<need;i++)res[i]=fa[i].x+0.5;return res;}vector<int>multiply(vector<int>&a,vector<int>&b,int m,int eq=0){int need=a.size()+b.size()-1;int nbase=0;while((1<<nbase)<need)nbase++;ensure_base(nbase);int sz=1<<nbase;if(sz>(int)fa.size())fa.resize(sz);for(int i=0;i<(int)a.size();i++){int x=(a[i]%m+m)%m;fa[i]=num(x&((1<<15)-1),x>>15);}fill(fa.begin()+a.size(),fa.begin()+sz,num{0,0});fft(fa,sz);if(sz>(int)fb.size())fb.resize(sz);if(eq)copy(fa.begin(), fa.begin()+sz,fb.begin());else{for(int i=0;i<(int)b.size();i++){int x=(b[i]%m+m)%m;fb[i]=num(x&((1<<15)-1),x>>15);}fill(fb.begin()+b.size(),fb.begin()+sz,num{0,0});fft(fb, sz);}double ratio=0.25/sz;num r2(0,-1),r3(ratio,0),r4(0,-ratio),r5(0, 1);for(int i=0;i<=(sz>>1);i++){int j=(sz-i)&(sz-1);num a1=(fa[i]+conj(fa[j]));num a2=(fa[i]-conj(fa[j]))*r2;num b1=(fb[i]+conj(fb[j]))*r3;num b2=(fb[i]-conj(fb[j]))*r4;if(i!=j){num c1=(fa[j]+conj(fa[i]));num c2=(fa[j]-conj(fa[i]))*r2;num d1=(fb[j]+conj(fb[i]))*r3;num d2=(fb[j]-conj(fb[i]))*r4;fa[i]=c1*d1+c2* d2*r5;fb[i]=c1*d2+c2*d1;}fa[j]=a1*b1+a2*b2*r5;fb[j]=a1*b2+a2*b1;}fft(fa,sz);fft(fb,sz);vector<int>res(need);for(int i=0;i<need;i++){ll aa=fa[i].x+0.5;ll bb=fb[i].x+0.5;ll cc=fa[i].y+0.5;res[i]=((aa+(bb%m)<<15)+((cc%m)<<30))%m;}return res;}}
```

8. Manacher

```
int d[2][5007];void Man(string&S,bool T){int n=S.size();for(int i=0,l=0,r=-1;i<n;i++){int k=(i>r)?T:min(d[T][l+r-i+!T],r-i+1);while(0<=i-k-!T&&i+k<n&&S[i-k-!T]==S[i+k])k++;d[T][i]=k--;if(i+k>r){l=i-k-!T;r=i+k;}}}
```

9. GEO

```
const double inf=1e100;const double eps=1e-9;const double PI=acos((double)-1.0);int sign(double x){return(x>eps)-(x<-eps);}struct PT{double x,y;PT(){x=0,y=0;}PT(double x,double y):x(x),y(y){}PT(const PT &p):x(p.x),y(p.y){}PT operator+(const PT &a)const{return PT(x+a.x,y+a.y);}PT operator-(const PT &a)const{return PT(x-a.x,y-a.y);}PT operator*(const double a)const{return PT(x*a,y*a);}friend PT operator*(const double &a,const PT &b){return PT(a*b.x,a*b.y);}PT operator/(const double a)const{return PT(x/a,y/a);}bool operator==(PT a)const{return sign(a.x-x)==0
&&sign(a.y-y)==0;}bool operator!=(PT a)const{return !(*this==a);}bool operator<(PT a)const{return sign(a.x-x)==0?y<a.y:x<a.x;}bool operator>(PT a)const{return sign(a.x-x)==0?y>a.y:x>a.x;}double norm(){return sqrt(x*x+y*y);}double norm2(){return x*x+y*y;}PT perp(){return PT(-y,x);}double arg(){return atan2(y,x);}PT truncate(double r) { // returns a vector with norm r and having same direction
double k=norm();if(!sign(k))return *this;r/=k;return PT(x*r,y*r);}};inline double dot(PT a,PT b){return a.x*b.x+a.y*b.y;}inline double dist2(PT a,PT b){return dot(a-b,a-b);}inline double dist(PT a,PT b){return sqrt(dot(a-b,a-b));}inline double cross(PT a,PT b){return a.x*b.y-a.y*b.x;}inline double cross2(PT a,PT b,PT c){return cross(b-a,c-a);}inline int orientation(PT a,PT b,PT c){return sign(cross(b-a,c-a));}PT perp(PT a){return PT(-a.y,a.x);}PT rotateccw90(PT a){return PT(-a.y,a.x);}PT rotatecw90(PT a){return PT(a.y,-a.x);}PT rotateccw(PT a,double t){return PT(a.x*cos(t)-a.y*sin(t),a.x*sin(t)+a.y*cos(t));}PT rotatecw(PT a,double t){return PT(a.x*cos(t)+a.y*sin(t),-a.x*sin(t)+a.y*cos(t));}double SQ(double x){return x*x;}double rad_to_deg(double r){return(r*180.0/PI);}double deg_to_rad(double d){return(d*PI/180.0);}double get_angle(PT a,PT b){double costheta=dot(a,b)/a.norm()/b.norm();return acos(max((double)-1.0,min((double)1.0,costheta)));}bool is_point_in_angle(PT b,PT a,PT c,PT p) { // does point p lie in angle <bac
assert(orientation(a,b,c)!=0);if(orientation(a,c,b)<0)swap(b,c);return orientation(a,c,p)>=0&&orientatio
```

```
n(a,b,p)<=0;}bool half(PT
p){return
p.y>0.0||(p.y==0.0&&p.x<0.0);}voi
d polar_sort(vector<PT> &v) { //
sort points in counterclockwise
sort(v.begin(),v.end(),[](PT a,PT
b){return
make_tuple(half(a),0.0,a.norm2())
<make_tuple(half(b),cross(a,b),b.
norm2());});}void
polar_sort(vector<PT> &v,PT o) {
// sort points in
counterclockwise with respect to
point o
sort(v.begin(),v.end(),[&](PT
a,PT b){return
make_tuple(half(a-o),0.0,(a-o).no
rm2())<make_tuple(half(b-o),cross
(a-o,b-o),(b-o).norm2());});}stru
ct line{PT a,b;// goes through
points a and b
PT v;double c;/*line form:
direction vec [cross] (x,y) = c*/
line(){}/*direction vector v and
offset c*/
line(PT v,double
c):v(v),c(c){auto
p=get_points();a=p.first;b=p.seco
nd;}/*equation ax + by + c = 0*/
line(double _a,double _b,double
_c):v({_b,-_a}),c(-_c){auto
p=get_points();a=p.first;b=p.seco
nd;}// goes through points p and
q
line(PT p,PT
q):v(q-p),c(cross(v,p)),a(p),b(q)
{}pair<PT,PT> get_points() {
//extract any two points from
this line
PT p,q;double a = -v.y,b = v.x;//
ax + by = c
if(sign(a)==0){p=PT(0,c/b);q=PT(1
,c/b);}else
if(sign(b)==0){p=PT(c/a,0);q=PT(c
/a,1);}else{p=PT(0,c/b);q=PT(1,(c
-a)/b);}return {p,q};}//ax + by +
c = 0
array<double,3> get_abc(){double
a=-v.y,b=v.x;return {a,b,c};}// 1
if on the left,-1 if on the
right,0 if on the line
int side(PT p){return
sign(cross(v,p)-c);}// line that
is perpendicular to this and goes
through point p
line perpendicular_through(PT
p){return {p,p+perp(v)};}//
translate the line by vector t
i.e. shifting it by vector t
line translate(PT t){return
{v,c+cross(v,t)};}// compare two
points by their orthogonal
projection on this line
// a projection point comes
before another if it comes first
according to vector v
bool cmp_by_projection(PT p,PT
q){return dot(v,p)<dot(v,q);}line
shift_left(double d){PT
z=v.perp().truncate(d);return
line(a+z,b+z);}};// find a point
from a through b with distance d
PT point_along_line(PT a,PT
b,double d){assert(a!=b);return
a+(((b-a)/(b-a).norm())*d);}//
projection point c onto line
through a and b  assuming a != b
PT project_from_point_to_line(PT
a,PT b,PT c){return
a+(b-a)*dot(c-a,b-a)/(b-a).norm2(
);}// reflection point c onto
line through a and b  assuming a
!= b
PT
reflection_from_point_to_line(PT
a,PT b,PT c){PT
p=project_from_point_to_line(a,b,
c);return p+p-c;}// minimum
distance from point c to line
through a and b
double dist_from_point_to_line(PT
a,PT b,PT c){return
fabs(cross(b-a,c-a)/(b-a).norm())
;}// returns true if  point p is
on line segment ab
bool is_point_on_seg(PT a,PT b,PT
p){if(fabs(cross(p-b,a-b))<eps){i
f(p.x<min(a.x,b.x)||p.x>max(a.x,b
.x))return
false;if(p.y<min(a.y,b.y)||p.y>ma
x(a.y,b.y))return false;return
true;}return false;}// minimum
distance point from point c to
segment ab that lies on segment
ab
PT project_from_point_to_seg(PT
a,PT b,PT c){double
r=dist2(a,b);if(sign(r)==0)return
a;r=dot(c-a,b-a)/r;if(r<0)return
a;if(r>1)return b;return
a+(b-a)*r;}// minimum distance
from point c to segment ab
double dist_from_point_to_seg(PT
a,PT b,PT c){return
dist(c,project_from_point_to_seg(
a,b,c));}// 0 if not parallel,1
if parallel,2 if collinear
int is_parallel(PT a,PT b,PT c,PT
d){double
k=fabs(cross(b-a,d-c));if(k<eps){
if(fabs(cross(a-b,a-c))<eps&&fabs
(cross(c-d,c-a))<eps)return
2;else return 1;}else return
0;}// check if two lines are same
bool are_lines_same(PT a,PT b,PT
c,PT
d){if(fabs(cross(a-c,c-d))<eps&&f
abs(cross(b-c,c-d))<eps)return
true;return false;}// bisector
vector of <abc
PT angle_bisector(PT &a,PT &b,PT
&c){PT p=a-b,q=c-b;return
p+q*sqrt(dot(p,p)/dot(q,q));}// 1
if point is ccw to the line,2 if
point is cw to the line,3 if
point is on the line
int point_line_relation(PT a,PT
b,PT p){int
c=sign(cross(p-a,b-a));if(c<0)ret
urn 1;if(c>0)return 2;return
3;}// intersection point between
ab and cd assuming unique
intersection exists
bool line_line_intersection(PT
a,PT b,PT c,PT d,PT &ans){double
a1=a.y-b.y,b1=b.x-a.x,c1=cross(a,
b);double
a2=c.y-d.y,b2=d.x-c.x,c2=cross(c,
d);double
det=a1*b2-a2*b1;if(det==0)return
0;ans=PT((b1*c2-b2*c1)/det,(c1*a2
-a1*c2)/det);return 1;}//
intersection point between
segment ab and segment cd
assuming unique intersection
exists
```

```
bool seg_seg_intersection(PT a,PT b,PT c,PT d,PT &ans){double oa=cross2(c,d,a),ob=cross2(c,d,b);double oc=cross2(a,b,c),od=cross2(a,b,d);if(oa*ob<0&&oc*od<0){ans=(a*ob-b*oa)/(ob-oa);return 1;}else return 0;}// intersection point between segment ab and segment cd assuming unique intersection may not exists
// se.size()==0 means no intersection
// se.size()==1 means one intersection
// se.size()==2 means range intersection
set<PT> seg_seg_intersection_inside(PT a, PT b, PT c, PT d){PT ans;if(seg_seg_intersection(a,b,c,d,ans))return {ans};set<PT> se;if(is_point_on_seg(c,d,a))se.insert(a);if(is_point_on_seg(c,d,b))se.insert(b);if(is_point_on_seg(a,b,c))se.insert(c);if(is_point_on_seg(a,b,d))se.insert(d);return se;}// intersection  between segment ab and line cd
// 0 if do not intersect,1 if proper intersect,2 if segment intersect
int seg_line_relation(PT a,PT b,PT c,PT d){double p=cross2(c,d,a);double q=cross2(c,d,b);if(sign(p)==0&&sign(q)==0)return 2;else if(p*q<0)return 1;else return 0;}// intersection between segament ab and line cd assuming unique intersection exists
bool seg_line_intersection(PT a,PT b,PT c,PT d,PT &ans){bool k=seg_line_relation(a,b,c,d);assert(k!=2);if(k)line_line_intersection(a,b,c,d,ans);return k;}// minimum distance from segment ab to segment cd
double dist_from_seg_to_seg(PT a,PT b,PT c,PT d){PT dummy;if(seg_seg_intersection(a,b,c,d,dummy))return 0.0;else return min({dist_from_point_to_seg(a,b,c),dist_from_point_to_seg(a,b,d),dist_from_point_to_seg(c,d,a),dist_from_point_to_seg(c,d,b)});}// minimum distance from point c to ray (starting point a and direction vector b)
double dist_from_point_to_ray(PT a,PT b,PT c){b=a+b;double r=dot(c-a,b-a);if(r<0.0)return dist(c,a);return dist_from_point_to_line(a,b,c);}// starting point as and direction vector ad
bool ray_ray_intersection(PT as,PT ad,PT bs,PT bd){double dx=bs.x-as.x,dy=bs.y-as.y;double det=bd.x*ad.y-bd.y*ad.x;if(fabs(det)<eps)return 0;double u=(dy*bd.x-dx*bd.y)/det;double v=(dy*ad.x-dx*ad.y)/det;if(sign(u)>=0&&sign(v)>=0)return 1;else return 0;}double ray_ray_distance(PT as,PT ad,PT bs,PT bd){if(ray_ray_intersection(as,ad,bs,bd))return 0.0;double ans=dist_from_point_to_ray(as,ad,bs);ans=min(ans,dist_from_point_to_ray(bs,bd,as));return ans;}struct circle{PT p;double r;circle(){}circle(PT _p,double _r): p(_p),r(_r){};// center (x,y) and radius r
circle(double x,double y,double _r): p(PT(x,y)),r(_r){};// circumcircle of a triangle
// the three points must be unique
circle(PT a,PT b,PT c){b=(a+b)*0.5;c=(a+c)*0.5;line_line_intersection(b,b+rotatecw90(a-b),c,c+rotatecw90(a-c),p);r=dist(a,p);}// inscribed circle of a triangle
circle(PT a,PT b,PT c,bool t){line u,v;double m=atan2(b.y-a.y,b.x-a.x),n=atan2(c.y-a.y,c.x-a.x);u.a=a;u.b=u.a+(PT(cos((n+m)/2.0),sin((n+m)/2.0)));v.a=b;m=atan2(a.y-b.y,a.x-b.x),n=atan2(c.y-b.y,c.x-b.x);v.b=v.a+(PT(cos((n+m)/2.0),sin((n+m)/2.0)));line_line_intersection(u.a,u.b,v.a,v.b,p);r=dist_from_point_to_seg(a,b,p);}bool operator==(circle v){return p==v.p&&sign(r-v.r)==0;}double area(){return PI*r*r;}double circumference(){return 2.0*PI*r;}};//0 if outside,1 if on circumference,2 if inside circle
int circle_point_relation(PT p,double r,PT b){double d=dist(p,b);if(sign(d-r)<0)return 2;if(sign(d-r)==0)return 1;return 0;}// 0 if outside,1 if on circumference,2 if inside circle
int circle_line_relation(PT p,double r,PT a,PT b){double d=dist_from_point_to_line(a,b,p);if(sign(d-r)<0)return 2;if(sign(d-r)==0)return 1;return 0;}//compute intersection of line through points a and b with
//circle centered at c with radius r > 0
vector<PT> circle_line_intersection(PT c,double r,PT a,PT b){vector<PT> ret;b=b-a;a=a-c;double A=dot(b,b),B=dot(a,b);double C=dot(a,a)-r*r,D=B*B-A*C;if(D<-eps)return ret;ret.push_back(c+a+b*(-B+sqrt(D+eps))/A);if(D>eps)ret.push_back(c+a+b*(-B-sqrt(D))/A);return ret;}//5 - outside and do not intersect
//4 - intersect outside in one point
//3 - intersect in 2 points
//2 - intersect inside in one point
//1 - inside and do not intersect
int circle_circle_relation(PT a,double r,PT b,double R){double d=dist(a,b);if(sign(d-r-R)>0)return 5;if(sign(d-r-R)==0)return 4;double l=fabs(r-R);if(sign(d-r-R)<0&&sign(d-l)>0)return
```

```
3;if(sign(d-l)==0)return
2;if(sign(d-l)<0)return
1;assert(0);return -1;}vector<PT>
circle_circle_intersection(PT
a,double r,PT b,double
R){if(a==b&&sign(r-R)==0)return
{PT(1e18,1e18)};vector<PT>
ret;double d=sqrt(dist2(a,
b));if(d>r+R||d+min(r, R)<max(r,
R))return ret;double
x=(d*d-R*R+r*r)/(2*d);double
y=sqrt(r*r-x*x);PT
v=(b-a)/d;ret.push_back(a+v*x+rot
ateccw90(v)*y);if(y>0)ret.push_ba
ck(a+v*x-rotateccw90(v)*y);return
ret;}// returns two circle c1,c2
through points a,b and of radius
r
// 0 if there is no such circle,1
if one circle,2 if two circle
int get_circle(PT a,PT b,double
r,circle &c1,circle
&c2){vector<PT>
v=circle_circle_intersection(a,r,
b,r);int t=v.size();if(!t)return
0;c1.p=v[0],c1.r=r;if(t==2)c2.p=v
[1],c2.r=r;return t;}// returns
two circle c1,c2 which is tangent
to line u, goes through
// point q and has radius r1;0
for no circle,1 if c1 = c2 ,2 if
c1 != c2
int get_circle(line u,PT q,double
r1,circle &c1,circle &c2){double
d=dist_from_point_to_line(u.a,u.b
,q);if(sign(d-r1*2.0)>0)return
0;if(sign(d)==0){cout<<u.v.x<<'
'<<u.v.y<<'\n';c1.p=q+rotateccw90
(u.v).truncate(r1);c2.p=q+rotatec
w90(u.v).truncate(r1);c1.r=c2.r=r
1;return 2;}line
u1=line(u.a+rotateccw90(u.v).trun
cate(r1),u.b+rotateccw90(u.v).tru
ncate(r1));line
u2=line(u.a+rotatecw90(u.v).trunc
ate(r1),u.b+rotatecw90(u.v).trunc
ate(r1));circle
cc=circle(q,r1);PT
p1,p2;vector<PT>
v;v=circle_line_intersection(q,r1
,u1.a,u1.b);if(!v.size())v=circle
_line_intersection(q,r1,u2.a,u2.b
);v.push_back(v[0]);p1=v[0],p2=v[
1];c1=circle(p1,r1);if(p1==p2){c2
=c1;return
1;}c2=circle(p2,r1);return 2;}//
returns area of intersection
between two circles
double circle_circle_area(PT
a,double r1,PT b,double
r2){double
d=(a-b).norm();if(r1+r2<d+eps)ret
urn 0;if(r1+d<r2+eps)return
PI*r1*r1;if(r2+d<r1+eps)return
PI*r2*r2;double
theta_1=acos((r1*r1+d*d-r2*r2)/(2
*r1*d)),theta_2=acos((r2*r2+d*d-r
1*r1)/(2*r2*d));return
r1*r1*(theta_1-sin(2*theta_1)/2.)
+r2*r2*(theta_2-sin(2*theta_2)/2.
);}// tangent lines from point q
to the circle
int tangent_lines_from_point(PT
p,double r,PT q,line &u,line
&v){int x=sign(dist2(p,q)-r*r);if
(x < 0) return 0;// point in
cricle
if (x == 0) { // point on circle
u=line(q,q+rotateccw90(q-p));v=u;
return 1;}double
d=dist(p,q);double l=r*r/d;double
h=sqrt(r*r-l*l);u=line(q,p+((q-p)
.truncate(l)+(rotateccw90(q-p).tr
uncate(h))));v=line(q,p+((q-p).tr
uncate(l)+(rotatecw90(q-p).trunca
te(h))));return 2;}// returns
outer tangents line of two
circles
// if inner == 1 it returns inner
tangent lines
int tangents_lines_from_circle(PT
c1,double r1,PT c2,double r2,bool
inner,line &u,line
&v){if(inner)r2=-r2;PT
d=c2-c1;double
dr=r1-r2,d2=d.norm(),h2=d2-dr*dr;
if(d2==0||h2<0){assert(h2!=0);ret
urn
0;}vector<pair<PT,PT>>out;for(int
tmp: {- 1,1}){PT
v=(d*dr+rotateccw90(d)*sqrt(h2)*t
mp)/d2;out.push_back({c1+v*r1,c2+
v*r2});}u=line(out[0].first,out[0
].second);if(out.size()==2)v=line
(out[1].first,out[1].second);retu
rn 1+(h2>0);}//O(n^2 log n)
struct CircleUnion{int n;double
x[2020],y[2020],r[2020];int
covered[2020];vector<pair<double,
double>>seg,cover;double
arc,pol;inline int sign(double
x){return x<-eps?-1:x>eps;}inline
int sign(double x,double
y){return sign(x-y);}inline
double SQ(const double x){return
x*x;}inline double dist(double
x1,double y1,double x2,double
y2){return
sqrt(SQ(x1-x2)+SQ(y1-y2));}inline
double angle(double A,double
B,double C){double
val=(SQ(A)+SQ(B)-SQ(C))/(2*A*B);i
f(val<-1)val=-1;if(val>+1)val=+1;
return
acos(val);}CircleUnion(){n=0;seg.
clear(),cover.clear();arc=pol=0;}
void
init(){n=0;seg.clear(),cover.clea
r();arc=pol=0;}void add(double
xx,double yy,double
rr){x[n]=xx,y[n]=yy,r[n]=rr,cover
ed[n]=0,n++;}void getarea(int
i,double lef,double
rig){arc+=0.5*r[i]*r[i]*(rig-lef-
sin(rig-lef));double
x1=x[i]+r[i]*cos(lef),y1=y[i]+r[i
]*sin(lef);double
x2=x[i]+r[i]*cos(rig),y2=y[i]+r[i
]*sin(rig);pol+=x1*y2-x2*y1;}doub
le solve(){for(int
i=0;i<n;i++){for(int
j=0;j<i;j++){if(!sign(x[i]-x[j])&
&!sign(y[i]-y[j])&&!sign(r[i]-r[j
]))){r[i]=0.0;break;}}}for(int
i=0;i<n;i++){for(int
j=0;j<n;j++){if(i!=j&&sign(r[j]-r
[i])>=0&&sign(dist(x[i],y[i],x[j]
,y[j])-(r[j]-r[i]))<=0){covered[i
]=1;break;}}}for(int
i=0;i<n;i++){if(sign(r[i])&&!cove
red[i]){seg.clear();for(int
j=0;j<n;j++){if(i!=j){double
d=dist(x[i],y[i],x[j],y[j]);if(si
gn(d-(r[j]+r[i]))>=0||sign(d-abs(
r[j]-r[i]))<=0){continue;}double
alpha=atan2(y[j]-y[i],x[j]-x[i]);
```

```
double beta=angle(r[i],d,r[j]);pair<double,double> tmp(alpha-beta,alpha+beta);if(sign(tmp.first)<=0&&sign(tmp.second)<=0){seg.push_back(pair<double,double>(2*PI+tmp.first,2*PI+tmp.second));}else if(sign(tmp.first)<0){seg.push_back(pair<double,double>(2*PI+tmp.first,2*PI));seg.push_back(pair<double,double>(0,tmp.second));}else{seg.push_back(tmp);}}}sort(seg.begin(),seg.end());double rig=0;for(vector<pair<double,double> >::iterator iter=seg.begin();iter!=seg.end();iter++){if(sign(rig-iter->first)>=0){rig=max(rig,iter->second);}else{getarea(i,rig,iter->first);rig=iter->second;}}if(!sign(rig)){arc+=r[i]*r[i]*PI;}else{getarea(i,rig,2*PI);}}}return pol/2.0+arc;}}CU;double area_of_triangle(PT a,PT b,PT c){return fabs(cross(b-a,c-a)*0.5);}// -1 if strictly inside,0 if on the polygon,1 if strictly outside
int is_point_in_triangle(PT a,PT b,PT c,PT p){if(sign(cross(b-a,c-a))<0)swap(b,c);int c1=sign(cross(b-a,p-a));int c2=sign(cross(c-b,p-b));int c3=sign(cross(a-c,p-c));if(c1<0||c2<0||c3<0)return 1;if(c1+c2+c3!=3)return 0;return -1;}double perimeter(vector<PT> &p){double ans=0;int n=p.size();for(int i=0;i<n;i++)ans+=dist(p[i],p[(i+1)%n]);return ans;}double area(vector<PT> &p){double ans=0;int n=p.size();for(int i=0;i<n;i++)ans+=cross(p[i],p[(i+1)%n]);return fabs(ans)*0.5;}// centroid of a (possibly non-convex) polygon,
// assuming that the coordinates are listed in a clockwise or
// counterclockwise fashion.
Note that the centroid is often known as
// the "center of gravity" or "center of mass".
PT centroid(vector<PT> &p){int n=p.size();PT c(0,0);double sum=0;for(int i=0;i<n;i++)sum+=cross(p[i],p[(i+1)%n]);double scale=3.0*sum;for(int i=0;i<n;i++){int j=(i+1)%n;c=c+(p[i]+p[j])*cross(p[i],p[j]);}return c/scale;}// 0 if cw,1 if ccw
bool get_direction(vector<PT> &p){double ans=0;int n=p.size();for(int i=0;i<n;i++)ans+=cross(p[i],p[(i+1)%n]);if(sign(ans)>0)return 1;return 0;}// it returns a point such that the sum of distances
// from that point to all points in p  is minimum
// O(n log^2 MX)
PT geometric_median(vector<PT> p){auto tot_dist=[&](PT z){double res=0;for(int i=0;i<p.size();i++)res+=dist(p[i],z);return res;};auto findY=[&](double x){double yl=-1e5,yr=1e5;for(int i=0;i<60;i++){double ym1=yl+(yr-yl)/3;double ym2=yr-(yr-yl)/3;double d1=tot_dist(PT(x,ym1));double d2=tot_dist(PT(x,ym2));if(d1<d2)yr=ym2;else yl=ym1;}return pair<double,double>(yl,tot_dist(PT(x,yl)));};double xl=-1e5,xr=1e5;for(int i=0;i<60;i++){double xm1=xl+(xr-xl)/3;double xm2=xr-(xr-xl)/3;double y1,d1,y2,d2;auto z=findY(xm1);y1=z.first;d1=z.second;z=findY(xm2);y2=z.first;d2=z.second;if(d1<d2)xr=xm2;else xl=xm1;}return {xl,findY(xl).first };}vector<PT> convex_hull(vector<PT> &p){if(p.size()<=1)return p;vector<PT> v=p;sort(v.begin(),v.end());vector<PT> up,dn;for(auto& p:v){while(up.size()>1&&orientation(up[up.size()-2],up.back(),p)>=0){up.pop_back();}while(dn.size()>1&&orientation(dn[dn.size()-2],dn.back(),p)<=0){dn.pop_back();}up.push_back(p);dn.push_back(p);}v=dn;if(v.size()>1)v.pop_back();reverse(up.begin(),up.end());up.pop_back();for(auto& p:up){v.push_back(p);}if(v.size()==2&&v[0]==v[1])v.pop_back();return v;}//checks if convex or not
bool is_convex(vector<PT> &p){bool s[3];s[0]=s[1]=s[2]=0;int n=p.size();for(int i=0;i<n;i++){int j=(i+1)%n;int k=(j+1)%n;s[sign(cross(p[j]-p[i],p[k]-p[i]))+1]=1;if(s[0]&&s[2])return 0;}return 1;}// -1 if strictly inside,0 if on the polygon,1 if strictly outside
// it must be strictly convex,otherwise make it strictly convex first
int is_point_in_convex(vector<PT> &p,const PT& x) { // O(log n)
int n=p.size();assert(n>=3);int a=orientation(p[0],p[1],x),b=orientation(p[0],p[n-1],x);if(a<0||b>0)return 1;int l=1,r=n-1;while(l+1<r){int mid=l+r>>1;if(orientation(p[0],p[mid],x)>=0)l=mid;else r=mid;}int k=orientation(p[l],p[r],x);if(k<=0)return -k;if(l==1&&a==0)return 0;if(r==n-1&&b==0)return 0;return -1;}bool is_point_on_polygon(vector<PT> &p,const PT& z){int n=p.size();for(int i=0;i<n;i++){if(is_point_on_seg(p[i],p[(i+1)%n],z))return 1;}return 0;}// returns 1e9 if the point is on the polygon
int winding_number(vector<PT> &p,const PT& z) { // O(n)
if(is_point_on_polygon(p,z))return 1e9;int n=p.size(),ans=0;for(int
```

```
i=0;i<n;++i){int j=(i+1)%n;bool
below=p[i].y<z.y;if(below!=(p[j].
y<z.y)){auto
orient=orientation(z,p[j],p[i]);i
f(orient==0)return
0;if(below==(orient>0))ans+=below
?1:-1;}}return ans;}// -1 if
strictly inside,0 if on the
polygon,1 if strictly outside
int
is_point_in_polygon(vector<PT>
&p,const PT& z) { // O(n)
int k=winding_number(p,z);return
k==1e9?0:k==0?1:-1;}// id of the
vertex having maximum dot product
with z
// polygon must need to be convex
// top - upper right vertex
// for minimum dot prouct negate
z and return -dot(z,p[id])
int extreme_vertex(vector<PT>
&p,const PT &z,const int top) {
// O(log n)
int n=p.size();if(n==1)return
0;double ans=dot(p[0],z);int
id=0;if(dot(p[top],z)>ans)ans=dot
(p[top],z),id=top;int
l=1,r=top-1;while(l<r){int
mid=l+r>>1;if(dot(p[mid+1],z)>=do
t(p[mid],z))l=mid+1;else
r=mid;}if(dot(p[l],z)>ans)ans=dot
(p[l],z),id=l;l=top+1,r=n-1;while
(l<r){int
mid=l+r>>1;if(dot(p[(mid+1)%n],z)
>=dot(p[mid],z))l=mid+1;else
r=mid;}l %=
n;if(dot(p[l],z)>ans)ans=dot(p[l]
,z),id=l;return id;}double
diameter(vector<PT> &p){int
n=(int)p.size();if(n==1)return
0;if(n==2)return
dist(p[0],p[1]);double ans=0;int
i=0,j=1;while(i<n){while(cross(p[
(i+1)%n]-p[i],p[(j+1)%n]-p[j])>=0
){ans=max(ans,dist2(p[i],p[j]));j
=(j+1)%n;}ans=max(ans,dist2(p[i],
p[j]));i++;}return
sqrt(ans);}double
width(vector<PT> &p){int
n=(int)p.size();if(n<=2)return
0;double ans=inf;int
i=0,j=1;while(i<n){while(cross(p[
(i+1)%n]-p[i],p[(j+1)%n]-p[j])>=0
)j=(j+1)%n;ans=min(ans,dist_from_
point_to_line(p[i],p[(i+1)%n],p[j
]));i++;}return ans;}// minimum
perimeter
double
minimum_enclosing_rectangle(vecto
r<PT> &p){int
n=p.size();if(n<=2)return
perimeter(p);int mndot=0;double
tmp=dot(p[1]-p[0],p[0]);for(int
i=1;i<n;i++){if(dot(p[1]-p[0],p[i
])<=tmp){tmp=dot(p[1]-p[0],p[i]);
mndot=i;}}double ans=inf;int
i=0,j=1,mxdot=1;while(i<n){PT
cur=p[(i+1)%n]-p[i];while(cross(c
ur,p[(j+1)%n]-p[j])>=0)j=(j+1)%n;
while(dot(p[(mxdot+1)%n],cur)>=do
t(p[mxdot],cur))mxdot=(mxdot+1)%n
;while(dot(p[(mndot+1)%n],cur)<=d
ot(p[mndot],cur))mndot=(mndot+1)%
n;ans=min(ans,2.0*((dot(p[mxdot],
cur)/cur.norm()-dot(p[mndot],cur)
/cur.norm())+dist_from_point_to_l
ine(p[i],p[(i+1)%n],p[j])));i++;}
return ans;}// given n
points,find the minimum enclosing
circle of the points
// call convex_hull() before this
for faster solution
// expected O(n)
circle
minimum_enclosing_circle(vector<P
T>
&p){random_shuffle(p.begin(),p.en
d());int n=p.size();circle
c(p[0],0);for(int
i=1;i<n;i++){if(sign(dist(c.p,p[i
])-c.r)>0){c=circle(p[i],0);for(i
nt
j=0;j<i;j++){if(sign(dist(c.p,p[j
])-c.r)>0){c=circle((p[i]+p[j])/2
,dist(p[i],p[j])/2);for(int
k=0;k<j;k++){if(sign(dist(c.p,p[k
])-c.r)>0){c=circle(p[i],p[j],p[k
]);}}}}}return c;}// returns a
vector with the vertices of a
polygon with everything
// to the left of the line going
from a to b cut away.
vector<PT> cut(vector<PT> &p,PT
a,PT b){vector<PT> ans;int
n=(int)p.size();for(int
i=0;i<n;i++){double
c1=cross(b-a,p[i]-a);double
c2=cross(b-a,p[(i+1)%n]-a);if(sig
n(c1)>=0)ans.push_back(p[i]);if(s
ign(c1*c2)<0){if(!is_parallel(p[i
],p[(i+1)%n],a,b)){PT
tmp;line_line_intersection(p[i],p
[(i+1)%n],a,b,tmp);ans.push_back(
tmp);}}}return ans;}// not
necessarily convex,boundary is
included in the intersection
// returns total intersected
length
double
polygon_line_intersection(vector<
PT> p,PT a,PT b){int
n=p.size();p.push_back(p[0]);line
l=line(a,b);double
ans=0.0;vector<
pair<double,int>>vec;for(int
i=0;i<n;i++){int
s1=sign(cross(b-a,p[i]-a));int
s2=sign(cross(b-a,p[i+1]-a));if(s
1==s2)continue;line
t=line(p[i],p[i+1]);PT
inter=(t.v*l.c-l.v*t.c)/cross(l.v
,t.v);double
tmp=dot(inter,l.v);int
f;if(s1>s2)f=s1&&s2?2:1;else
f=s1&&s2?-2:-1;vec.push_back(make
_pair(tmp,f));}sort(vec.begin(),v
ec.end());for(int
i=0,j=0;i+1<(int)vec.size();i++){
j+=vec[i].second;if(j)ans+=vec[i+
1].first-vec[i].first;}ans=ans/sq
rt(dot(l.v,l.v));p.pop_back();ret
urn ans;}pair<PT,PT>
convex_line_intersection(vector<P
T> &p,PT a,PT b){return
{{0,0},{0,0}};}// minimum
distance from a point to a convex
polygon
// it assumes point does not lie
strictly inside the polygon
double
dist_from_point_to_polygon(vector
<PT> &v,PT p) { // O(log n)
int
n=(int)v.size();if(n<=3){double
ans=inf;for(int
i=0;i<n;i++)ans=min(ans,dist_from
```

```
_point_to_seg(v[i],v[(i+1)%n],p));return ans;}PT bscur,bs=angle_bisector(v[n-1],v[0],v[1]);int ok, i, pw=1, ans=0, sgncur, sgn=sign(cross(bs,p-v[0]));while(pw<=n)pw <<= 1;while((pw >>= 1)){if((i=ans+pw)<n){bscur=angle_bisector(v[i-1],v[i],v[(i+1)%n]); sgncur=sign(cross(bscur,p-v[i])); ok=sign(cross(bs,bscur))>=0?(sgn>=0||sgncur<=0):(sgn>=0&&sgncur<=0);if(ok)ans=i,bs=bscur,sgn=sgncur;}}return dist_from_point_to_seg(v[ans],v[(ans+1)%n],p);}// minimum distance from convex polygon p to line ab
// returns 0 is it intersects with the polygon
// top - upper right vertex
double dist_from_polygon_to_line(vector<PT> &p,PT a,PT b,int top) {
//O(log n)
PT orth=(b-a).perp();if(orientation(a,b,p[0])>0)orth=(a-b).perp();int id=extreme_vertex(p,orth,top);if (dot(p[id] - a,orth) > 0) return 0.0;//if orth and a are in the same half of the line,then poly and line intersects
return dist_from_point_to_line(a,b,p[id]);//does not intersect
}// minimum distance from a convex polygon to another convex polygon
double dist_from_polygon_to_polygon(vector<PT> &p1,vector<PT> &p2) { // O(n log n)
double ans=inf;for(int i=0;i<p1.size();i++){ans=min(ans,dist_from_point_to_polygon(p2,p1[i]));}for(int i=0;i<p2.size();i++){ans=min(ans,dist_from_point_to_polygon(p1,p2[i]));}return ans;}// maximum distance from a convex polygon to another convex polygon
double maximum_dist_from_polygon_to_polygon(vector<PT> &u,vector<PT> &v){
//O(n)
int n=(int)u.size(),m=(int)v.size();double ans=0;if(n<3||m<3){for(int i=0;i<n;i++){for(int j=0;j<m;j++)ans=max(ans,dist2(u[i],v[j]));}return sqrt(ans);}if(u[0].x>v[0].x)swap(n,m),swap(u,v);int i=0,j=0,step=n+m+10;while(j+1<m&&v[j].x<v[j+1].x)j++;while(step--){if(cross(u[(i+1)%n]-u[i],v[(j+1)%m]-v[j])>=0)j=(j+1)%m;else i=(i+1)%n;ans=max(ans,dist2(u[i],v[j]));}return sqrt(ans);}pair<PT,int> point_poly_tangent(vector<PT> &p,PT Q,int dir,int l,int r){while(r-l>1){int mid=(l+r)>>1;bool pvs=orientation(Q,p[mid],p[mid-1])!=-dir;bool nxt=orientation(Q,p[mid],p[mid+1])!=-dir;if(pvs&&nxt)return {p[mid],mid};if(!(pvs||nxt)){auto p1=point_poly_tangent(p,Q,dir,mid+1,r);auto p2=point_poly_tangent(p,Q,dir,l,mid-1);return orientation(Q,p1.first,p2.first)==dir?p1:p2;}if(!pvs){if(orientation(Q,p[mid],p[l])==dir)r=mid-1;else if(orientation(Q,p[l],p[r])==dir)r=mid-1;else l=mid+1;}if(!nxt){if(orientation(Q,p[mid],p[l])==dir)l=mid+1;else if(orientation(Q,p[l],p[r])==dir)r=mid-1;else l=mid+1;}}pair<PT,int> ret={p[l],l};for(int i=l+1;i<=r;i++)ret=orientation(Q,ret.first,p[i])!=dir?make_pair(p[i],i):ret;return ret;}// (cw,ccw) tangents from a point that is outside this convex polygon
// returns indexes of the points
pair<int,int> tangents_from_point_to_polygon(vector<PT> &p,PT Q){int cw=point_poly_tangent(p,Q,1,0,(int)p.size()-1).second;int ccw=point_poly_tangent(p,Q,-1,0,(int)p.size()-1).second;return make_pair(cw,ccw);}// calculates the area of the union of n polygons (not necessarily convex).
// the points within each polygon must be given in CCW order.
// complexity: O(N^2),where N is the total number of points
double rat(PT a,PT b,PT p){return !sign(a.x-b.x)?(p.y-a.y)/(b.y-a.y):(p.x-a.x)/(b.x-a.x);};double polygon_union(vector<vector<PT>> &p){int n=p.size();double ans=0;for(int i=0;i<n;++i){for(int v=0;v<(int)p[i].size();++v){PT a=p[i][v],b=p[i][(v+1)%p[i].size()];vector<pair<double,int>> segs;segs.emplace_back(0,0),segs.emplace_back(1,0);for(int j=0;j<n;++j){if(i!=j){for(size_t u=0;u<p[j].size();++u){PT c=p[j][u],d=p[j][(u+1)%p[j].size()];int sc=sign(cross(b-a,c-a)),sd=sign(cross(b-a,d-a));if(!sc&&!sd){if(sign(dot(b-a,d-c))>0&&i>j){segs.emplace_back(rat(a,b,c),1),segs.emplace_back(rat(a,b,d),-1);}}else{double sa=cross(d-c,a-c),sb=cross(d-c,b-c);if(sc>=0&&sd<0)segs.emplace_back(sa/(sa-sb),1);else if(sc<0&&sd>=0)segs.emplace_back(sa/(sa-sb),-1);}}}}sort(segs.begin(),segs.end());double pre=min(max(segs[0].first,0.0),1.0),now,sum=0;int cnt=segs[0].second;for(int j=1;j<segs.size();++j){now=min(max(segs[j].first,0.0),1.0);if(!cnt)sum+=now-pre;cnt+=segs[j].second;pre=now;}ans+=cross(a,b)*sum;}}return ans*0.5;}// contains all points p such that: cross(b - a,p
```

```
- a) >= 0
struct HP{PT a,b;HP(){}HP(PT a,PT b):a(a),b(b){}HP(const HP& rhs):a(rhs.a),b(rhs.b){}int operator<(const HP& rhs)const{PT p=b-a;PT q=rhs.b-rhs.a;int fp=(p.y<0||(p.y==0&&p.x<0));int fq=(q.y<0||(q.y==0&&q.x<0));if(fp!=fq)return fp==0;if(cross(p,q))return cross(p,q)>0;return cross(p,rhs.b-a)<0;}PT line_line_intersection(PT a,PT b,PT c,PT d){b=b-a;d=c-d;c=c-a;return a+b*cross(c,d)/cross(b,d);}PT intersection(const HP &v){return line_line_intersection(a,b,v.a,v.b);}};int check(HP a,HP b,HP c){return cross(a.b - a.a,b.intersection(c) - a.a) > -eps;//-eps to include polygons of zero area (straight lines,points)
}// consider half-plane of counter-clockwise side of each line
// if lines are not bounded add infinity rectangle
// returns a convex polygon,a point can occur multiple times though
// complexity: O(n log(n))
vector<PT> half_plane_intersection(vector<HP> h){sort(h.begin(),h.end());vector<HP> tmp;for(int i=0;i<h.size();i++){if(!i||cross(h[i].b-h[i].a,h[i-1].b-h[i-1].a)){tmp.push_back(h[i]);}}h=tmp;vector<HP> q(h.size()+10);int qh=0,qe=0;for(int i=0;i<h.size();i++){while(qe-qh>1&&!check(h[i],q[qe-2],q[qe-1]))qe--;while(qe-qh>1&&!check(h[i],q[qh],q[qh+1]))qh++;q[qe++]=h[i];}while(qe-qh>2&&!check(q[qh],q[qe-2],q[qe-1]))qe--;while(qe-qh>2&&!check(q[qe-1],q[qh],q[qh+1]))qh++;vector<HP> res;for(int i=qh;i<qe;i++)res.push_back(q[i]);vector<PT> hull;if(res.size()>2){for(int i=0;i<res.size();i++){hull.push_back(res[i].intersection(res[(i+1)%((int)res.size())]));}}return hull;}// a and b are strictly convex polygons of DISTINCT points
// returns a convex hull of their minkowski sum with distinct points
vector<PT> minkowski_sum(vector<PT> &a,vector<PT> &b){int n=(int)a.size(),m=(int)b.size();int i = 0,j = 0;//assuming a[i] and b[j] both are (left,bottom)-most points
vector<PT> c;c.push_back(a[i]+b[j]);while(i+1<n||j+1<m){PT p1=a[i]+b[(j+1)%m];PT p2=a[(i+1)%n]+b[j];int t=orientation(c.back(),p1,p2);if(t>=0)j=(j+1)%m;if(t<=0)i=(i+1)%n,p1=p2;if(t==0)p1=a[i]+b[j];if(p1==c[0])break;c.push_back(p1);}return c;}// system should be translated from circle center
double triangle_circle_intersection(PT c,double r,PT a,PT b){double sd1=dist2(c,a),sd2=dist2(c,b);if(sd1>sd2)swap(a,b),swap(sd1,sd2);double sd=dist2(a,b);double d1=sqrtl(sd1),d2=sqrtl(sd2),d=sqrt(sd);double x=abs(sd2-sd-sd1)/(2*d);double h=sqrtl(sd1-x*x);if(r>=d2)return h*d/2;double area=0;if(sd+sd1<sd2){if(r<d1)area=r*r*(acos(h/d2)-acos(h/d1))/2;else{area=r*r*(acos(h/d2)-acos(h/r))/2;double y=sqrtl(r*r-h*h);area+=h*(y-x)/2;}}else{if(r<h)area=r*r*(acos(h/d2)+acos(h/d1))/2;else{area+=r*r*(acos(h/d2)-acos(h/r))/2;double y=sqrtl(r*r-h*h);area+=h*y/2;if(r<d1){area+=r*r*(acos(h/d1)-acos(h/r))/2;area+=h*y/2;}else area+=h*x/2;}}return area;}// intersection between a simple polygon and a circle
double polygon_circle_intersection(vector<PT> &v,PT p,double r){int n=v.size();double ans=0.00;PT org={0,0};for(int i=0;i<n;i++){int x=orientation(p,v[i],v[(i+1)%n]);if(x==0)continue;double area=triangle_circle_intersection(org,r,v[i]-p,v[(i+1)%n]-p);if(x<0)ans-=area;else ans+=area;}return abs(ans);}// find a circle of radius r that contains as many points as possible
// O(n^2 log n);
double maximum_circle_cover(vector<PT> p,double r,circle &c){int n=p.size();int ans=0;int id=0;double th=0;for(int i=0;i<n;++i){// maximum circle cover when the circle goes through this point
vector<pair<double,int>> events={{-PI,+1},{PI,-1}};for(int j=0;j<n;++j){if(j==i)continue;double d=dist(p[i],p[j]);if(d>r*2)continue;double dir=(p[j]-p[i]).arg();double ang=acos(d/2/r);double st=dir-ang,ed=dir+ang;if(st>PI)st-=PI*2;if(st<=-PI)st+=PI*2;if(ed>PI)ed-=PI*2;if(ed<=-PI)ed+=PI*2;events.push_back({st - eps,+1});// take care of precisions!
events.push_back({ed,-1});if(st>ed){events.push_back({-PI,+1});events.push_back({+PI,-1});}}sort(events.begin(),events.end());int cnt=0;for(auto &&e: events){cnt+=e.second;if(cnt>ans){ans=cnt;id=i;th=e.first;}}}PT w=PT(p[id].x+r*cos(th),p[id].y+r*sin(th));c = circle(w,r);//best_circle
return ans;}// radius of the maximum inscribed circle in a convex polygon
```

```
double
maximum_inscribed_circle(vector<P
T> p){int
n=p.size();if(n<=2)return
0;double
l=0,r=20000;while(r-l>eps){double
mid=(l+r)*0.5;vector<HP> h;const
int
L=1e9;h.push_back(HP(PT(-L,-L),PT
(L,-L)));h.push_back(HP(PT(L,-L),
PT(L,L)));h.push_back(HP(PT(L,L),
PT(-L,L)));h.push_back(HP(PT(-L,L
),PT(-L,-L)));for(int
i=0;i<n;i++){PT
z=(p[(i+1)%n]-p[i]).perp();z=z.tr
uncate(mid);PT
y=p[i]+z,q=p[(i+1)%n]+z;h.push_ba
ck(HP(p[i]+z,p[(i+1)%n]+z));}vect
or<PT>
nw=half_plane_intersection(h);if(
!nw.empty())l=mid;else
r=mid;}return l;}
```

10. Hopcroft Carp

```
struct Hp{static const int inf=1e
9;int n;vector mL,mR,d;vector>g;H
p(int n):n(n),mL(n+1),mR(n+1),d(n
+1),g(n+1){} void edge(int u,int
v){ g[u].push_back(v);/*left Part
(1..n indexed)-right part(1..n in
dexed)*/}bool bfs(){queue q;for(i
nt u=1;u<=n;u++){if(!mL[u])d[u]=0
,q.push(u);else d[u]=inf;}d[0]=in
f;while(!q.empty()){int u=q.front
();q.pop();for(auto v:g[u]){if(d[
mR[v]]==inf)d[mR[v]]=d[u]+1,q.pus
h(mR[v]);}}return(d[0]!=inf);}boo
l dfs(int u){if(!u)return 1;for(a
uto v:g[u]){if(d[mR[v]]==d[u]+1&&
dfs(mR[v])){mL[u]=v;mR[v]=u;retur
n 1;}}d[u]=inf;return false;}int
ans(){int ret=0;while(bfs())for(i
nt u=1;u<=n;u++)if(!mL[u])if(dfs(
u))ret++;return ret;}};
```

11. KMP and Z

```
/*max prefix match suffix*/vector
<int>prefix_function(string s){in
t n=(int)s.length();vector<int>pi
(n);for(int i=1;i<n;i++){int j=pi
[i-1];while(j>0&&s[i]!=s[j])j=pi[
j-1];if(s[i]==s[j])j++;pi[i]=j;}r
eturn pi;}/*max prefix match with
 prefix*/vector<int>z_function(st
ring s){int n=(int)s.length();vec
tor<int>z(n);for(int i=1,l=0,r=0;
i<n;++i){if(i<=r)z[i]=min(r-i+1,z
[i-l]);while(i+z[i]<n&& s[z[i]]==
s[i+z[i]])++z[i];if(i+z[i]-1>r)l=
i,r=i+z[i]-1;}return z;}
```

12. HLD

```
template<int size,int lg,typename
seg_t=long long>struct
hld{vector<int> edges[size];int
bigchild[size],sz[size],depth[siz
e],chain[size];int
label[size],label_time,par[size];
int lca_lift[size][lg];seg_t
seg_tree[4*size];seg_t
seg_lazy[4*size];int N;/* CHANGE
THIS SECTION BY PROBLEM */inline
seg_t seg_combine(seg_t a,seg_t
b){return a+b;}inline seg_t
seg_lazy_apply(seg_t
lazy_val,seg_t new_val){return
new_val;}inline seg_t
seg_lazy_func(seg_t cur_val,seg_t
lazy_val,int l,int r){return
lazy_val;}const seg_t
seg_sentinel=0;const seg_t
seg_lazy_sentinel=-1;const seg_t
seg_init_val=0;/* END SECTION
*/seg_t seg_query_header(int
l,int r){return
seg_query_rec(0,0,N-1,l,r);}seg_t
seg_query_rec(int i,int tl,int
tr,int ql,int
qr){seg_eval_lazy(i,tl,tr);if(ql<
=tl&&tr<=qr)return
seg_tree[i];if(tl>tr||tr<ql||qr<t
l)return seg_sentinel;int
mid=(tl+tr)/2;seg_t
a=seg_query_rec(2*i+1,tl,mid,ql,q
r);seg_t
b=seg_query_rec(2*i+2,mid+1,tr,ql
,qr);return
seg_combine(a,b);}void
seg_update_header(int l,int
r,seg_t
v){seg_update_rec(0,0,N-1,l,r,v);
}seg_t seg_update_rec(int i,int
tl,int tr,int ql,int qr,seg_t
v){seg_eval_lazy(i,tl,tr);if(tl>t
r||tr<ql||qr<tl)return
seg_tree[i];if(ql<=tl&&tr<=qr){se
g_lazy[i]=seg_lazy_apply(seg_lazy
[i],v);seg_eval_lazy(i,tl,tr);ret
urn seg_tree[i];}if(tl==tr)return
seg_tree[i];int
mid=(tl+tr)/2;seg_t
a=seg_update_rec(2*i+1,tl,mid,ql,
qr,v);seg_t
b=seg_update_rec(2*i+2,mid+1,tr,q
l,qr,v);return
seg_tree[i]=seg_combine(a,b);}voi
d seg_eval_lazy(int i,int l,int
r){if(seg_lazy[i]==seg_lazy_senti
nel)return;seg_tree[i]=seg_lazy_f
unc(seg_tree[i],seg_lazy[i],l,r);
if(l!=r){seg_lazy[i*2+1]=seg_lazy
_apply(seg_lazy[i*2+1],seg_lazy[i
]);seg_lazy[i*2+2]=seg_lazy_apply
(seg_lazy[i*2+2],seg_lazy[i]);}se
g_lazy[i]=seg_lazy_sentinel;}/* -
init phase 1- *//* initialize
segtree,clear edges,etc. */void
init_arrays(int n){N=n;for(int
i=0;i<N;i++){edges[i].clear();cha
in[i]=i;}for(int
i=0;i<4*N;i++){seg_tree[i]=seg_in
it_val;seg_lazy[i]=seg_lazy_senti
nel;}}/* put edges in */void
add_edge(int u,int
v){edges[u].push_back(v);edges[v]
.push_back(u);}/* build the
lca&&hld stuff */void
init_tree(seg_t* arr,int root=0){
lca_dfs(root,-1);dfs_size(root,-1
,0);dfs_chains(root,-1);
label_time=0;dfs_labels(arr,root,
-1);}void lca_dfs(int v,int
par){lca_lift[v][0]=par;for(int
i=1;i<lg;i++){if(lca_lift[v][i-1]
==-1)lca_lift[v][i]=-1;else
lca_lift[v][i]=lca_lift[lca_lift[
v][i-1]][i-1];}for(int x:
edges[v]){if(x!=par){lca_dfs(x,v)
;}}}void dfs_size(int v,int p,int
d){sz[v]=1;depth[v]=d;par[v]=p;in
t bigc=-1,bigv=-1;for(int x:
edges[v]){if(x!=p){dfs_size(x,v,d
+1);sz[v]+=sz[x];if(sz[x]>bigv){b
igc=x;bigv=sz[x];}}}bigchild[v]=b
igc;}void dfs_chains(int v,int
p){if(bigchild[v]!=-1){chain[bigc
hild[v]]=chain[v];}for(int x:
```

```
edges[v]){if(x!=p){dfs_chains(x,v);}}}void dfs_labels(seg_t* arr,int v,int p){label[v]=label_time++;seg_update_header(label[v],label[v],arr[v]);if(bigchild[v]!=-1){dfs_labels(arr,bigchild[v],v);}for(int x:edges[v]){if(x!=p&&x!=bigchild[v]){dfs_labels(arr,x,v);}}}/* usage */int lca(int a,int b){if(depth[a]<depth[b])swap(a,b);int d=depth[a]-depth[b];int v=get_kth_ancestor(a,d);if(v==b){return v;}else{for(int i=lg-1;i>=0;i--){if(lca_lift[v][i]!=lca_lift[b][i]){v=lca_lift[v][i];b=lca_lift[b][i];}}return lca_lift[b][0];}}int get_kth_ancestor(int v,int k){for(int i=lg-1;i>=0;i--){if(v==-1)return v;if((1<<i)<=k){v=lca_lift[v][i];k-=(1<<i);}}return v;}/*excludes p */seg_t query_chain(int v,int p){seg_t val=seg_sentinel;while(depth[p]<depth[v]){int top=chain[v];if(depth[top]<=depth[p]){int diff=depth[v]-depth[p];top=get_kth_ancestor(v,diff-1);}val=seg_combine(val,seg_query_header(label[top],label[v]));v=par[top];}return val;}seg_t query(int u,int v){int lc=lca(u,v);seg_t val=seg_combine(query_chain(u,lc),query_chain(v,lc));return seg_combine(val,0LL);}/* excludes p */void update_chain(int v,int p,seg_t val){while(depth[p]<depth[v]){int top=chain[v];if(depth[top]<=depth[p]){int diff=depth[v]-depth[p];top=get_kth_ancestor(v,diff-1);}seg_update_header(label[top],label[v],val);v=par[top];}}void update(int u,int v,seg_t val){int lc=lca(u,v);update_chain(u,lc,val);update_chain(v,lc,val);seg_update_header(label[lc],label[lc],val);}};
```

14.MO

```
inline int64_t gilbertOrder(int x,int y, int pow,int rotate){
if(pow==0)return 0;int hpow=1<<(pow-1);int seg=(x<hpow)? ((y<hpow)?0:3):((y<hpow)?1:2);
seg=(seg+rotate)&3;
const int rotateDelta[4] = {3,0,0,1};int nx=x& (x^hpow),ny=y& (y^hpow);
int nrot=(rotate+rotateDelta[seg])&3;
int64_t subSquareSize= int64_t(1)<<(2*pow-2);
int64_t ans=seg*subSquareSize;
int64_t add=gilbertOrder (nx, ny,pow-1,nrot);
ans+=(seg==1||seg==2) ?add:(subSquareSize-add-1);
return ans;}struct query{
int l,r,id;ll ord;void calcord(){ord=gilbertOrder (l,r,21,0);}}qq[mx1];
bool cmp(query &a,query &b) {return a.ord<b.ord;}for (int i=1;i<=q;i++)sf(qq[i].l) ,sf(qq[i].r), qq[i].id=i, qq[i].calcord();sort(qq+1, qq+q+1,cmp);int l=0,r=0;
for(int i=1;i<=q;i++){
while(l<qq[i].l) baad(l++);
while(r<qq[i].r) jog(++r);
while(l>qq[i].l) jog(--l);
while(r>qq[i].r) baad(r--);
ans[ qq[i].id]=sum;}
```

15.Palindromic Tree

```
/*-> diff(v)=len(v)-len (link(v))-> series link will lead from the vertex v to the vertex u corresponding to the maximum suffix palindrome of v which satisfies diff(v)!= diff(u)-> path within series links to the root contains only O(log n)vertices->cnt contains the number of palindromic suffixes of the node*/struct PalindromicTree{struct node{
int nxt[26],len,st,en,link, diff,slink,cnt,oc;};string s;
vector<node> t;int sz,last;
PalindromicTree(){}
PalindromicTree(string _s){
s=_s;int n=s.size();t.clear()
;t.resize(n+9);sz=2,last=2;
t[1].len=-1,t[1].link=1;
t[2].len=0,t[2].link=1;
t[1].diff=t[2].diff=0;t[1].slink=1;t[2].slink=2;}int
extend(int pos) { // returns 1 if it creates a new palindrome
int cur=last,curlen=0;int ch=s[pos]-'a';while(1){curlen=t[cur].len;if(pos-1-curlen>=0&&s[pos-1-curlen]==s[pos])break;cur=t[cur].link;}if(t[cur].nxt[ch]){last=t[cur].nxt[ch];t[last].oc++;return 0;}sz++;last=sz;t[sz].oc=1;t[sz].len=t[cur].len+2;t[cur].nxt[ch]=sz;t[sz].en=pos;t[sz].st=pos-t[sz].len+1;if(t[sz].len==1){t[sz].link=2;t[sz].cnt=1;t[sz].diff=1;t[sz].slink=2;return 1;}
while(1){cur=t[cur].link;curlen=t[cur].len;if(pos-1-curlen>=0&&s[pos-1-curlen]==s[pos]){t[sz].link=t[cur].nxt[ch];break;}}t[sz].cnt=1+T[t[sz].link].cnt;t[sz].diff=t[sz].len-t[t[sz].link].len;if(t[sz].diff==t[t[sz].link].diff)t[sz].slink=t[t[sz].link].slink;else t[sz].slink=t[sz].link;return 1;}void
calc_occurrences(){for(int i=sz;i>=3;i--)t[t[i].link].oc+=t[i].oc;}vector<array<int,2>>minimum_partition() {
//(even,odd),1 indexed
int n=s.size();vector<array<int,2>>ans(n+1,{0,0}), series_ans(n+5,{0,0});
ans[0][1]=series_ans[2][1]=1e9;for(int i=1;i<=n;i++){extend(i-1);for(int k=0;k<2;k++){ans[i][k]=1e9;for(int v=last;t[v].len>0;v=t[v].slink){series_ans[v][!k]=ans[i-(t[t[v].slink].len+t[v].diff)][!k];if(t[v].diff==t[t[v].link].diff)
```

```
series_ans[v][!k]=min(
Series_ans[v][!k],series_ans
[t[v].link][!k]);ans[i][k]=
min(ans[i][k],series_ans
[v][!k]+1);}}}return ans;}}t;
16.PBDS
#include
<ext/pb_ds/assoc_container.
hpp>
#include
<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef tree<int,null_type,less
<int>,rb_tree_tag,
tree_order_statistics_node
_update> ordered_set;
typedef tree<int,null_type,less_
equal<int>,rb_tree_tag,
tree_order_statistics_node
_update> ordered_multiset;
X.insert(16);
*X.find_by_order(0)
X.order_of_key(-5)
#include<ext/rope>//header with
rope using namespace std;
using namespace __gnu_cxx;
//namespace with rope and some
additional stuff rope<int>v;
//use as usual STL container
for(rope<int>::iterator it=v.
mutable_begin();it!= v.mutable_
end();++it)cout<<*it<<" ";
for(int i=1;i<=n;++i)v.push_back
(i);for(int i=0;i<m;++i){cin>>l>>
r;--l,--r;rope<int>cur=v.substr(l
,r-l+1);v.erase(l,r-l+1);v.insert
(v.mutable_begin(),cur);}
17.CUSTOM HASH
struct custom_hash{static
uint64_t splitmix64(uint64_t x){
x+=0x9e3779b97f4a7c15;
x=(x^(x>>32))*0xbf58476d1ce4e5b9;
x=(x^(x>>29))*0x94d049bb133111eb;
return x^(x>>34);}size_t
operator()(uint64_t x)const{
static const uint64_t
FIXED_RANDOM=chrono::steady_clock
::now().time_since_epoch().
count();return splitmix64(x +
FIXED_RANDOM);}};
18.Pollard Rho and miller rabin
ull mulmod(ull a,ull t,ull
mm){ull r=1;for(;t;t>>=1
,a=(ulll)(a)* a%mm)if(t&1)
r=(ulll)(r)*a%mm;return r;
}ll mul(ll a,ll b,ll p){ll
tmp=(a*b-(ll)((long double
)a/p*b+1e-8)*p);return tmp<
0?tmp+p:tmp;}bool isPrime
(ull n){static const int
jp[]={2,3,5,7,11,13,17,19};
if(n==1)return 0;for(int
p:jp)if(n%p==0)return n==p;
ull r=n-1,x,y;int e=0;while
(~r&1)r>>=1,++e;for(int p:jp)
{x=mulmod(p,r,n);for(int t=0;
t<e&&x>1;++t){y=(lll)x*x%n;
if(y==1&&x^(n-1))return 0;x=y
;}if(x^1)return 0;}return 1;}
ll rho(ll n,ll c){ll k=2,x=
rand()%n,y=x,p=1;for(ll i=1;
p==1;i++){x=(mul(x,x,n)+c)%n;
p=y>x?y-x:x-y;p=__gcd(n,p);
if(i==k) y=x,k+=k;}return p;}
unordered_map<ll,int>mp;
void solve(ll n){
if(n==1) return;if(isPrime(n
)){mp[n]++;return;}ll t=n;
while(t==n)
t=rho(n,rand()%(n-1)+1);
solve(t);solve(n/t);}
19.Suffix Array
int Maxn=200005;int M_Log=20;
int arr[Maxn],n;int kinds=
256;///maximum ASCII value of
any character of the string
int nn;char str[M];int K,
buc[M],r[M],sa[M],X[M],Y[M],
high[M];bool cmp(int *r,int
a,int b,int x){return(r[a]
==r[b]&&r[a+x]==r[b+x]);}
vector<int>saa;int lcp[M];
void suffix_array_DA(int n,
int m){int *x=X,*y=Y,i,j,k=0
,l;memset(buc,0,sizeof(buc));
for(i=0;i<n;i++)buc[x[i]=
str[i] ]++;for(i=1;i<m;i++)
buc[i]+=buc[i-1];for(i=n-1;
i>=0;i--)sa[--buc[x[i]]]=i;
for(l=1,j=1;j<n;m=j,l<<=1){
j=0;for(i=n-l;i<n;i++)y[j++]
=i;for(i=0;i<n;i++)if(sa[i]
>=l)y[j++]=sa[i]-l;for(i=0;
i<m;i++)buc[i]=0;for(i=0;i<n;
i++)buc[x[y[i]]]++;for(i=1;
i<m;i++)buc[i]+=buc[i-1];for
(i=n-1;i>=0;i--)sa[--buc[
x[y[i]]]]=y[i];for(swap(x,y),
x[sa[0]]=0,i=1,j=1;i<n;i++)
x[sa[i]]=cmp(y,sa[i-1],sa[i]
,l)?j-1:j++;}for(i=1;i<n;i++)
r[sa[i]]=i;for(i=0;i<n-1;high
[r[i++]]=k)for(k?k--:0,j=sa[r
[i]-1];str[i+k]==str[j+k];k++)
;}void
suffix_array_construction
(string s){int n=s.size();for
(int i=0;i<n;i++)str[i]
=s[i];str[n]='\0';
suffix_array_DA(n+1,kinds);
for(int i=1;i<=n;i++)
saa.push_back(sa[i]);}void
lcp_construction(string const
&s,vector<int> const& p){int
n=s.size();vector<int>rank(n,
0);for(int i=0;i<n;i++)
rank[p[i]]=i;int k=0;for(int
i=0;i<n;i++){if(rank[i]==n-1)
{k=0;continue;}int j=p
[rank[i]+1];while(i+k<n&&j+k
<n &&s[i+k]==s[j+k])k++;lcp[
rank[i]]=k;if(k)k--;}}string
ss;int
combinedStringErKoiShuru[M],
combinedErKonIndexSuffixAraay
Tekoi[M],stringErShuruSuffix
ArrayTeKothay[M],sz[M];struct
wtree{int s[Maxn],tree
[M_Log][Maxn],L[M_Log][Maxn];
long long ls[M_Log][Maxn],SL;
void build(int l,int r,int
d){if(l==r)return;int
mid=(l+r)>>1,cnt=0,lc=l,rc=
mid+1,ec=0;for(int i= l;i<=r;
i++)if(tree[d][i]<s[mid])
cnt++;for(int i=l;i<=r;i++)
{if((tree[d][i]<s[mid])||(
tree[d][i]==s[mid]&&ec<(mid-
l+1-cnt))){tree[d+1][lc++]=
tree[d][i];ls[d][i]=ls[d][i
-1]+ tree[d][i];if(tree[
d][i]==s[mid])ec++;} else{
tree[d+1][rc++]=tree[d][i];
ls[d][i]=ls[d][i-1];}L[d][i]=
L[d][l-1]+lc-l;}build(l,mid,
d+1);build(mid+1,r,d+1);}void
```

```
init(int *arr,int n){for(
int i=1;i<=n;i++){tree[0][i]=
arr[i];s[i]=tree[0][i];}sort(s+1,
s+n+1);build(1,n,0);}int kth(int
l,int r,int d,int x,int y,int k)
//1,n,0,x,y,k
if(l==r){SL+=tree[d][l];return
tree[d][l];}int cnt=L[d][y]
-L[d][x-1];int mid=(l+r)>>1;if(
cnt>=k){int newx=L[d][x-1]-L[d]
[l-1];int newy=L[d][y]-L[d][l-1]
;return kth(l,mid,d+1,l+newx,l
+newy-1,k);}else{int newx=x-l-(L
[d][x-1]-L[d][l-1]);int newy=y-l+
1-(L[d][y]-L[d][l-1]);SL+=ls[d][y
]-ls[d][x-1];return kth(mid+1,r,
d+1,mid+1+newx,mid+newy,k-cnt);}}
int LTE(int l,int r,int d,int
x,int y,int k) //1,n,0,x,y,k
if(l==r){if(l>y||l<x)return 0;//
SL += tree[d][l]*(tree[d][l]<=k);
return tree[d][l]<=k;}int cnt=L[d
][y]-L[d][x -1],mid=(l+r)>>1;if(
s[mid]<=k){int newx=x-l-(L[d]
[x-1]-L[d][l-1]);int newy=y-l+1-
(L[d][y]-L[d][l-1]);// SL
+=ls[d][y]-ls[d][x-1];return
cnt+LTE(mid+1,r,d+1,mid+1+newx,mi
d +newy,k);}else {int
newx=L[d][x-1]-L[d][l-1];int
newy=L[d][y]-L[d][l-1];return
LTE(l,mid,d+1,l+newx,l+newy-1,k);
}}}tree;ll ans[M];int occ[M];int
last[M],idx[M];int
t[M][21],lg2[M];void f(int n){int
i,j,k,m,l,r,q;lg2[1]=0;for(i=2;i<
=n;i++)lg2[i]=lg2[i/2]+1;for(i=1;
i<=n;i++)t[i][0]=lcp[i];for(j=1;j
<=20;j++)for(i=1;i+(1<<(j-1))<=n;
i++)t[i][j]=min(t[i][j-1],t[i+(1<
<(j-1))][j-1]);}int squery(int
l,int r ){int k=lg2[r-l+1];int
ans=min(t[l][k],t[r
-(1<<k)+1][k]);return ans;}int
main(){
ios_base::sync_with_stdio(0);
cin.tie(0);int n,k;cin>>n>>k;
string s;int
combinedStringErIndex=0;
std::vector<int>
shuruIndexErList;for(int
i=1;i<=n;i++){stringt;cin>>t;
sz[i]=t.size();if(i>1)s+="$";
s+=t;combinedStringErKoiShuru
[i]=combinedStringErIndex;
shuruIndexErList.push_back(
combinedStringErIndex);
combinedStringErIndex+=
t.size()+1;}
suffix_array_construction(s);
lcp_construction(s,saa);
nn=saa.size();reverse(saa.
begin(),saa.end());saa.
push_back(-1);reverse(saa.
begin(),saa.end();for(int
i=nn-1;i>=1;i--)lcp[i]=
lcp[i-1];lcp[0]= 0;f(nn);for
(int i=1;i<=nn;i++)
combinedErKonIndexSuffixAraay
Tekoi[saa[i]]=i;for(int i=1;
i<=n;i++)
stringErShuruSuffixArrayTe
Kothay[i]=
combinedErKonIndexSuffixAraay
Tekoi[
combinedStringErKoiShuru[i]];
sort(shuruIndexErList.begin()
,shuruIndexErList.end());int
ii=1;for(int i=nn;i>=1;
i--){if(s[saa[i]]=='$')
continue;int p= upper_bound(
shuruIndexErList.begin(),
shuruIndexErList.end(),saa[i]
)-shuruIndexErList.begin();
idx[i]=p;if(last[p])occ[i]=
last[p];elseocc[i]=nn+7;
last[p]= i;}tree.init
(occ,nn);for(int i=1;i<=nn;
i++){if(s[saa[i]]=='$')
continue;int id= idx[i];int
l= 1,r= sz[id]-(saa[i]-
combinedStringErKoiShuru[id]
+1)+1;int len=0;int A=i,B=i;
while(l<=r){int mid=(l+r)/2;
int l1=1,r1=i-1;int a=i,b=
i-1;while(l1<=r1){int mid1=(
l1+r1)/2;int rmq=squery
(mid1,i-1);if(rmq<mid)l1=
mid1+1;else r1=mid1-1,a=mid1
;}l1=i,r1=nn;while(l1<=r1){
int mid1=(l1+r1)/2;int rmq=
squery(i,mid1);if(rmq<mid)r1=
mid1-1;else b= mid1,l1=
mid1+1;}b++;A= a,B= b;int
unq=tree.LTE(1,nn,0,a,b,b);
unq=b-a+1-unq;if(unq>=k)len=
mid,l=mid+1;else
r=mid-1;}ans[id]+=len;}
20.Monotonic queue
deque<pii>dq;void notun(int
baad,int i,int j){while(dq
.size()&&dq.front().first<
baad)dq.pop_front();while(
dq.size()and dq.back().second
<=dp[i][j])dq.pop_back();
dq.push_back({i,dp[i][j]});}
int main(){int n,k,x;cin>>n
>>k>>x;if( x>n||k>n||(n+2*K-
2)/(2*k-1)>x)return cout<<"
-1\n",0;for(int i=1;i<=n;i++
)cin>>arr[i];for(int i=0;i<=n
;i++)for(int j=0;j<=x;j++)
dp[i][j]=-inf;dp[0][0]=0;
for(int j=1;j<=x;j++){
dq.clear();for(int i=1;i<=n;
i++){notun( i-k,i-1,j-1);ll
boro= dq.front().second;
dp[i][j]=boro+arr[i];}}}/*
With reason:*/
deque<pair<int,int>>dq;/*
<id,value>*/for(int i=1,j=1;
i<=n-k+1;++i){/*pop:remove
expired elements*/while
(!dq.empty()&& dq.front().
first<=i-k)dq.pop_front();/*
push:add new element */
while(j<=n&&j<=i+k-1){/*We want
to put A_j into the deque *//*
First,remove elements thatis
useless */while(!dq.empty()
&&A[j]<=dq.back().second)dq
.pop_back();/*Then,putitin */
dq.push_back({j,A[j]}),++j;}}
21.Hungarian
struct HungarianMatching{long
long c[N][N],fx[N],fy[N],d[N];
int mx[N],my[N],arg[N],trace
[N];queue<int> q;int start,
finish,n;const long long inf
=1e18;HungarianMatching(){}
HungarianMatching(int n):n(n)
{for(int i=1;i<=n;++i){fy[i]=
mx[i]=my[i]=0;for(int j=1;j<=n
;++j)c[i][j]=inf;}}void
addedge(int u,int v,long
long cost){
```

```
c[u][v]=min(c[u][v],
cost);}inline long long getC(int
u,int v){
return c[u][v]-fx[u]-fy[v];}void
initBFS(){while(!q.empty())q.pop(
);q.push(start);for(int i=0;i<=n;
++i)trace[i]=0;for(int v=1;v<=n;
++v){d[v]=getC(start,v);arg[v]=st
art;}finish=0;}void findAugPath()
{while(!q.empty()){int u=
q.front();q.pop();for(int v=1;v<=
n;++v)if(!trace[v]){long long w=
getC(u,v);if(!w){ trace[v]=u;if(
!my[v]){finish=v;return;} q.push
(my[v]);}if(d[v]>w){d[v]=w;arg[v]
=u;}}}}void subX_addY(){long long
delta=inf;for(int v=1;v<=n;++v)if
(trace[v]==0&&d[v]<delta){delta=d
[v];}/* Rotate*/fx[start]+=delta;
for(int v=1;v<=n;++v)if(trace[v])
{int u=my[v];fy[v]-=delta;fx[u]+=
delta;}else d[v]-=delta;for(int v
=1;v<=n;++v)if(!trace[v]&&!d[v])
trace[v]=arg[v];if(!my[v]){finish
=v;return;}q.push(my[v]);}}void
enlarge(){do{int u=trace[finish]
;int nxt=mx[u];mx[u]=finish;my[
finish]=u;finish=nxt;}while(
finish);}long long matching
(){for(int u=1;u<=n;++u){fx[u]=
c[u][1];for(int v=1;v<=n;++v){
fx[u]=min(fx[u],c[u][v]);}}for(
int v=1;v<=n;++v){fy[v]=c[1][v]
-fx[1];for(int u=1;u<=n;++u){
fy[v]=min(fy[v],c[u][v]-fx[u]);}}
for(int u=1;u<=n;++u){start=
u;initBFS();while(!finish){findAu
gPath();if(!finish)subX_addY();}E
nlarge();}long long ans=0;for(int
i=1;i<=n;++i)if(c[i][mx[i]]!=inf)
ans+=c[i][mx[i]];return ans;}};
22.DSU on Tree
void dfs(int u,int
p){sz[u]=1;for(auto
v:adj[u]){if(v==p)continue;dfs(v,
u);sz[u]+=sz[v];}}void add(int
u,int p,int
x){cnt[col[u]]+=x;if(cnt[col[u]]>
maxi){maxi=cnt[col[u]];sum=col[u]
;}else
if(cnt[col[u]]==maxi){sum+=col[u]
;}for(auto
v:adj[u]){if(v!=p&&big[v]!=1)add(
v,u,x);}}void dsu(int u,int
p,bool keep){int
bigchild=-1,mx=-1;for(auto
v:adj[u]){if(v!=p&&sz[v]>mx){mx=s
z[v];bigchild=v;}}for(auto
v:adj[u]){if(v!=p&&v!=bigchild)ds
u(v,u,0);}if(bigchild!=-1){dsu(bi
gchild,u,1);big[bigchild]=1;}add(
u,p,1);ans[u]=sum;if(bigchild!=-1
)big[bigchild]=0;if(keep==0){add(
u,p,-1);maxi=0;sum=0;}}//dfs(1,1)
dsu(1,1,1)
23.Automata
#define ll long long #define mx
400007 using namespace std;ll
sum[mx];struct SuffixAutomaton{
struct node{int len,link,firstpos
;map<char,int>nxt;};int sz,last;
vector<node>t;vector<int>terminal
;vector<bool> vis1,vis2;
vector<long long>dp;vector<long
long>cnt1,cnt2;vector<vector<
int>>g;SuffixAutomaton(){}
SuffixAutomaton(int n){
t.resize(2*n);terminal.resize
(2*n,0);dp.resize(2*n,-1);sz=1;
last=0;g.resize(2*n);cnt1.resize
(2*n,0);cnt2.resize(2*n,0);vis1
.resize(2*n,0);vis2.resize
(2*n,0);t[0].len=0;t[0].link=
-1;t[0].firstpos=0;}void extend
(char c){int p=last;if(t[p].nxt
.count(c)){int q=t[p].nxt[c];
if(t[q].len==t[p].len+1){last=q;
return;}int clone=sz++;t[clone]=
t[q];t[clone].len=t[p].len+1;
t[q].link=clone;last=clone;
while(p!=-1&&t[p].nxt[c]==q){t[p]
.nxt[c]=clone;p=t[p].link;}return
;}int cur=sz++;t[cur].len=t[last]
.len+1;t[cur].firstpos=t[cur].len
;p=last;while(p!=-1&&!t[p].nxt.
count(c)){t[p].nxt[c]=cur;p
=t[p].link;}if(p==-1)
t[cur].link=0;else{int q=t[p].
nxt[c];if(t[p].len+1==t[q].len)
t[cur].link=q;else{int clone=sz++
;t[clone]=t[q]; t[clone].len
=t[p].len+1;while(p!=-1&&t[p].
nxt[c]==q){t[p].nxt[c]=clone;p=t
[p].link;}t[q].link=t[cur].link
=clone;}}last=cur;}void
build_tree(){for(int i=1;i<sz;i++
)g[t[i].link].push_back(i);}void
build(string&s){for(autox:s){exte
nd(x);// val[last]= something
terminal[last]=1;}build_tree();}l
ong long cnt(int i){ //number of
times i-th node occurs in the
string
if(dp[i]!=-1)returndp[i];long
long ret=terminal[i];for(auto
&x:g[i])ret+=cnt(x);return dp[i]
=ret;}void dfs(int u){if(vis1[u])
return;vis1[u]=1;for(auto v:
t[u].nxt){if(v.first==
'#')cnt1[u]++;else
if(v.first=='$')Cnt2[u]
++;else{dfs(v.second);cnt1[u]
+=cnt1[v.second];cnt2[u]+=
cnt2[v.second];}}}};int32_t
main(){SuffixAutomaton
sa(s.size());for(auto c:
s)sa.extend(c);}
24.suffix array
const int MAX=200010;ll
txt[MAX];int iSA[MAX],
SA[MAX];int cnt[MAX],nexto[MAX];
bool bh[MAX],b2h[MAX];bool
smaller_first_char(int a,int
b){return txt[a]<txt[b];}void
suffixSort(int n){for(int i=0
;i<n;++i){SA[i]=i;}sort(SA,SA
+n,smaller_first_char);for(
int i=0;i<n;++i){bh[i]=i==0
||txt[SA[i]]!=txt[SA[i-1]];
b2h[i]=false;}for(
int h=1;h<n;h<<=1)
{int buckets=0;
for(int i=0,j;i<n;i=j){j=i+1;
while(j<n&&!bh[j])j++;nexto[i]=j;
Buckets++;}if(buckets==n)break;
for(int i=0;i<n;i=nexto[i]){
cnt[i]=0;for(int j=i;j<nexto[i];
++j){iSA[SA[j]]=i;}}cnt[iSA[n-h]]
++;b2h[iSA[n-h]]=true;for(int i=0
;i<n;i=nexto[i]){for(int j=i;j<
nexto[i];++j){int s=SA[j]-h;
if(s>=0){int head=iSA[s];iSA[s]=
head+cnt[head]++;b2h[iSA[s]]=true
;}}for(int j=i;j<nexto[i];++j){
int s=SA[j]-h;if(s>=0&&
b2h[iSA[s]]){for(int k=iSA[s]
```

```
+1;!bh[k]&&b2h[k];k++)b2h[k]=
false;}}}for(int i=0;i<n;++i)
{SA[iSA[i]]=i;bh[i]|=b2h[i];}}for
(int i=0;i<n;++i){iSA[SA[i]]=i;
}}int lcp[MAX],ara[MAX],rev[MAX]
,rev2[MAX],dp[MAX],root[MAX];void
getlcp(int n){for(int i=0;i<n;
++i){iSA[SA[i]]=i;}lcp[0]=0;for(
int i=0;i<=2;i++)
{//assert(txt[SA[i]]==big);}for(
int i=0,h=0;i<n;++i){if(iSA[i]
>0){int j=SA[iSA[i]-1];
while(i+h<n&&j+h<n&&txt[i+h]==txt
[j+h])h++;lcp[iSA[i]]=h;if(h>0)h-
-;}}}
25.Rotating Calipers
n=ret.size();for(int i=0;i<n;i++)
{int k=(i+1)%n;for(int j=i+1;j<n;
j++){ll last= 0;while((k+1)%n!=i
&& area(ret[i],ret[j],ret[k])<=
area(ret[i],ret[j], ret[(k+1)%n])
)k=(k+1)%n;ans=max(ans,area(ret[i
],ret[j],ret[k]));}
26.Bitset trick
bs._Find_first() bs._Find_next()
27.prime counting
Prime Countingnamespace pcf{/*
Prime-Counting Function
initialize once by calling
init()Legendre(n)&&Lehmer(n)
returns the number of primes
less than||equal to n Lehmer
(n)is faster*/
#define MAXN 1000010
/* initial sieve limit*/
#define MAX_PRIMES 1000010
/* max size of the primearray
for sieve*/
#define PHI_N 100000
#define PHI_K 100
unsigned int
ar[(MAXN>>6)+5]={0};int len=0;
/* total number of primes
generated by sieve*/int
primes[MAX_PRIMES];int
counter[MAXN];/* counter[m]
--> number of primes<=i*/int
phi_dp[PHI_N][PHI_K];/*precal
of phi(n,k)*/bitset isComp;
/*bool isComp[MAXN];*/void
Sieve(int N){int i,j,sq=
sqrt(N);isComp[1]=true;for(
i=4;i<=N;i+=2)isComp[i]=true;
for(i=3;i<=sq;i+=2){if(!
isComp[i]){for(j=i*i;j<=N;
j+=i+i)isComp[j]=1;}}for(i=1;
i<=N;i++){if(!isComp[i])
primes[len++]=i;counter[i]=
len;}}void
init(){Sieve(MAXN-1);/*
precalculation of phi up to
size(PHI_N,PHI_K)*/int
k,n,res;for(n=0;n<PHI_N;n++)
phi_dp[n][0]=n;for(k=1;k<
PHI_K;k++){for(n=0;n<PHI_N;
n++){phi_dp[n][k]=phi_dp[n]
[k-1]-phi_dp[n/primes[k-1]]
[k-1];}}}/* returns number of
integers less||equal nwhich
are not divisible byany of the
first k primes recurrence -->
phi( n ,k)=phi(
n,k-1)-phi(n/p_k,k-1)*/long
long phi(long long n,int
k){if(n<PHI_N&&k<PHI_K)return
phi_dp[n][k];if(k==1)return(
(++n)>>1);if(primes[k-1]>=n)
return 1;return
phi(n,k-1)-phi(n/primes[k-1],
k-1);}long long Legendre(
longlong n){if(n<MAXN) return
counter[n];int lim=sqrt(n)+1;
int k=upper_bound(
primes,primes+len,lim)-primes
;return phi(n,k)+(k
-1);}/*complexity:n^(2/3).
(logn)^(1/3)*/long long
Lehmer(long long
n){if(n<MAXN)return
counter[n];long long w,res=0;
int i,j,a,b,c,lim;
b=sqrt(n),c=Lehmer(cbrt(n)),
a=Lehmer(sqrt(b)),b=Lehmer(b)
;res=phi(n,a)+(((b+a-2)*
(b-a+1))>>1);for(i=a;i<b;i++)
{w=n/primes[i];lim
=Lehmer(sqrt(w)),res-=Lehmer
(w);if(i<=c){for(j=i;j<lim;
j++){res+=j;res-=Lehmer(w/
primes[j]);}}}return res;}}
28.mt19937 rng
(chrono::
steady_clock::now().
time_since_epoch().
count());
29.Articulationbridge
void find_articulation_bridge
(ll node,ll
par){dis[node]=low[node]=
++tim;int children=0;for(int
u:v[node]){if(u==par)continue
;if(!dis[u]){
find_articulation_bridge(u,
node); if(dis[node]<low[u]){
bridge.push_back({u,node});}
low[node]=min(low[u],low[node
]);/* articlation point
if(low[u]>=dis[node]&&
par!=-1)point */children++;}
else low[node]=min
(low[node],dis[u]);}/*
articlationPoint if(par==
-1&&children>1)point*/}
30.PHI
//find number of co prime
with i from 1 to mx
int arr[sz];void oiler(ll n){
for(int i=2;i<=n;i++)arr[i]=i;
for(int i=2;i<=n;i++){if(
arr[i]==i){for(int j=i;j<=n;
j+=i)arr[j]-=arr[j]/i;}}}
31.mobius function
void mobius(int N){mob[1]=1;
for(int i=2;i<N;i++){mob[i]--;
for(int j=i+i;j<N;j+=i)
{mob[j]-=mob[i];}}}
32.TREAP
#include<bits/stdc++.h>
#define ll long long
#define inf 1000000000000000LL
#define _max(x) (x?
x->mx:(int)-1e9)
using namespace std;random_device
rd;mt19937 rr(rd());typedef
struct item*pitem;struct item{int
prior,cnt;ll
value,pref,suff,tot,best;item(){}
item( int a,int b){value=a;
prior=b;pref=suff=best=tot=
value;l=NULL;r=NULL;cnt=0;}
pitem l,r;};pitem root=NULL;
int cnt(pitem it){return it?
it->cnt:0;}void combine(pitem
t){// update data from t
first,then l and r
if(!t)return;t->tot= t->value;
```

```
if(t->l)t->tot+=t->l->tot;if(t->r
)t->tot+=t->r->tot;ll prefL=
-inf,sumL=-inf,suffL=-inf;ll
prefR=-inf,sumR=-inf,suffR=-inf;
ll bestL=-inf,bestR= -inf;
if(t->l){prefL=t->l->pref;suffL=t
->l->suff;sumL=t->l->tot;bestL=t-
>l->best;}if(t->r){prefR=t->r->pr
ef;suffR=t->r->suff;sumR=t->r->to
t;bestR=t->r->best;}if(prefL!=-in
f)t->pref=max({
prefL,sumL+t->value,sumL,sumL+t->
value+sumR,sumL+t->value+prefR
});else t-> pref=max({
t->value,t->value+sumR,t->value+p
refR});if(suffR!=-inf)t->
suff=max({suffR,sumR+
t->value,sumR,sumR+t->value+
sumL,sumR+t->value+suffL});else
t->suff=max({t->value,t->value+su
mL,t->value+suffL});t->best=max({
bestR,bestL,t->pref,t->suff,t->to
t,suffL+t->value,t->value+prefR,s
uffL+t->value+prefR,t->value,suff
L,prefR,sumR,sumL});}void
upd_cnt(pitem it){if(it)it->
cnt=cnt(it->l)+cnt(it->r)+1;//if(
it)//it->mx=max(it->value,max(_ma
x(it->l),_max(it->r)));}void
merge(pitem&t,pitem l,pitem
r){if(!l||!r)t=l?l: r;else
if(l->prior>r->prior)merge(
l->r,l->r,r),t=l;else merge
(r->l,l,r->l),t=r;upd_cnt(t);
combine(t);}void split(pitem
t,pitem&l,pitem &r,int key,
int add=0){if(!t)return
void(l=r=0);int
cur_key=add+cnt(t->l);if(key
<=cur_key)split(t->l,l,t->l,
key,add),r=t;elsesplit(t->r,
t->r,r,key,add+1+cnt(t->l)),
l=t;upd_cnt(t);combine(t);}
void print_array(pitem
t){if(!t)return;print_array(
t->l);printf("%d ",t->
value);print_array(t->r);}void
insert_at_pos(int y,int pos)
{pitem it=new item(y,rr());
pitem l,r,mid;split(root,
l,mid,pos);merge(root,l,it );
merge(root,root,mid);}void
range_query(int x,int y){
pitem l,r,mid;split(root,l,
mid,x-1);split(mid,mid,
r,y-x+1);//cout<<mid->mx<<"\n";
merge(root,l,mid);merge(root,
root,r);}void delete_pos(int
pos){pitem l,r,mid;split(
root,l,mid,pos);split(mid,mid
,r,1); free(mid);merge(root,
l,r);}void replace_at_pos(int
x,int pos){delete_pos(pos);
insert_at_pos(x,pos);}ll
range_maxsum_query(int x,int
y){pitem l,r,mid;split(root,l,
mid,x);split(mid,mid,r,y-x+1);
ll ret=mid->best; merge(mid,
mid,r);merge(root,l,mid)
;return ret;}int main(){
ios_base::sync_with_stdio(0);
cin.tie(0);int n;cin>>n;for
(int i=0;i<n;i++){ll
p;cin>>p;insert_at_pos(p,i);}
int q;cin>>q;while(q--){char
c;cin>>c;if(c=='I'){int
x,y;cin>>x>>y;insert_at_pos(
y,x-1 );}else if(c=='D'){int
x;cin>>x;delete_pos(x-1);}else
if(c=='R'){int x,y;cin>>x>>y;
replace_at_pos(y,x-1);}else{
int x,y;cin>>x>>y;x--,y--;ll
ans=range_maxsum_query(x,y);
cout<<ans<<"\n";}}return 0;}
33.Blossom
mt19937 rnd(chrono::steady
_clock::now().time_since_
epoch().count());struct
Blossom{int vis[N],par[N],
orig[N],match[N],aux[N],t;int
n;bool ad[N];vector<int>
g[N];queue<int> Q;
Blossom(){}Blossom(int _n){
n=_n;t=0;for(int i=0;i<=_n;
++i){g[i].clear();match[i]
=aux[i]=par[i]=vis[i]=aux[i]
=ad[i]=orig[i]=0;}}void
add_edge(int u,int
v){g[u].push_back(v);g[v].
push_back(u);}void augment(
int u,int v){int pv=v,nv;do
{pv=par[v];nv=match[pv];match
[v]=pv;match[pv]=v;v=nv;}
while(u!=pv);}int lca(int v,
int w){++t;while(true){if(v){
if(aux[v]==t)return v;aux[v]
=t;v=orig[par[match[v]]];}
swap(v,w);}}void blossom(int
v,int w,int a){while(orig[v]
!=a){par[v]=w;w=match[v];
ad[v]=true;if(vis[w]==1)Q.
push(w),vis[w]=0;orig[v]=
orig[w]=a;v=par[w];}}//it
finds an augmented path
starting from u
bool bfs(int u){fill(
vis+1,vis+n+1,-1);iota(orig+1
,orig+n+1,1);Q=queue<int>();
Q.push(u);vis[u]=0;while(
!Q.empty()){int v=Q.front()
;Q.pop();ad[v]=true;for(int
x:g[v]){if(vis[x]==-1){par[x]
=v;vis[x]=1;if(!match[x])
return augment(u,x),true;
Q.push(match[x]);vis[match
[x]]=0;}else if(vis[x]==0
&&orig[v]!= orig[x]){int
a=lca(orig[v],orig[x]);
blossom(x,v,a);blossom(v,x,a)
;}}}return false;}int
maximum_matching(){int ans=0;
vector<int> p(n-1);iota
(p.begin(),p.end(),1);shuffle
(p.begin(),p.end(),rnd);for
(int i=1;i<=n;i++)
shuffle(g[i].begin(),g[i].end
(),rnd);for(auto u:p){if
(!match[u]){for(auto v:g[u]){
if(!match[v]){match[u]=v,match
[v]=u;++ans;break;}}}}for(int
i=1;i<= n;++i)if(!
match[i]&&bfs(i))++ans;return
ans;}} M;int32_t main()
{ios_base::sync_with_stdio(0)
;cin.tie(0);int t;
cin>>t;
while(t--){int n,m;cin>>n>>m;
M=Blossom(n);while(m--){int u
,v;cin>>u>>v;M.add_edge(u,v);}
int ans=M.maximum_matching();
if(ans*2==n)cout<<0<<'\n';else
{memset(M.ad,0,sizeof M.ad);
for(int i=1;i<=n;i++)if(M.
match[i]==0)M.bfs(i);int
ans=0;for(int i=1;i<=n;i++)
ans+= M.ad[i];//nodes which are
```

unmatched in some maximummatching
```
cout<<ans<<'\n';}}return 0;}
```
34. Sublime build
```
{"shell_cmd": "g++ -std=c++17
-Wshadow -Wall \"${file}\" -o
\"${file_path}/${file_base_name}\
" -O2 -Wno-unused-result &&
timeout 5s
\"${file_path}/${file_base_name}\
" < \"${file_path}/input.txt\" >
\"${file_path}/output.txt\" "
}
Compile: g++ -O2 -std=c++17
-Wno-unused-result -Wshadow -Wall
-o "%e" "%f"
Build: g++ -DLOCAL -std=c++17
-Wshadow -Wall -o "%e" "%f"
-fsanitize=address
-fsanitize=undefined
-D_GLIBCXX_DEBUG -g
```
35.Centroid decomposition
```
int n,tot,cs=1;vector<int>
adj[maxn];ll cnt[4][K];int
sz[maxn],col[maxn],val[maxn];bool
dead[maxn],f;ll ans;void dfs0(int
u,int p,int
c){if(col[u]){c++;val[u]=0;}else
val[u]=c;for(auto v:
adj[u]){if(v==p)continue;dfs0(v,u
,c);}}void dfs1(int u,int
p){sz[u]=1;tot++;for(auto v:
adj[u]){if(v==p||dead[v])continue
;dfs1(v,u);sz[u]+=sz[v];}}int
dfs2(int u,int p){for(auto v:
adj[u]){if(v!=p&&!dead[v]&&sz[v]>
tot/2){return dfs2(v,u);}}return
u;}void dfs3(int u,int p,int
lev,int
add){if(!col[u]&&val[u]<=2)cnt[va
l[u]][lev]+=add;for(auto v:
adj[u]){if(v==p||dead[v])continue
;dfs3(v,u,lev,add);}}ll dfs4(int
u,int p,int dep){ll
ret=0;if(!col[u]&&val[u]<=2)ret+=
cnt[2-val[u]][dep];for(auto v:
adj[u]){if(v==p||dead[v])continue
;ret+=dfs4(v,u,dep);}return
ret;}void decompose(int root,int
dep){tot=0;dfs1(root,root);int
centroid=dfs2(root,root);f=1;int
u=centroid;dfs0(u,-1,0);dfs3(u,u,
dep,1);ll add=0;//dead[u] = 1;
for(auto v:
adj[u]){if(dead[v])continue;dfs3(
v,u,dep,-1);add+=dfs4(v,u,dep);df
s3(v,u,dep,+1);}ans+=add;dead[u]=
1;for(auto v:
adj[u]){if(dead[v])continue;decom
pose(v,dep+1);}for(int
i=0;i<=2;++i)cnt[i][dep]=0;}main(
){while(tc--){scanf("%d",&n);for(
int
i=1;i<=n;++i)dead[i]=0;val[i]=0;a
ns=0;decompose(1,0);for(int
i=1;i<=n;++i)adj[i].clear();memse
t(cnt,0,sizeof cnt);}}
```
36.Polar Sort
```
bool half(point p){return p.y
>0||(p.y==0&&p.x<0);}
void polar_sort(vector<point>
&v){sort(v.begin(),v.end(),
[](point a,point b){return
make_tuple(half(a-base),
0,(a-base)|(a-base))
<make_tuple(half(b-base),
(a-base)*(b-base),(b-base)|
(b-base));});}
```
37.prime list:
```
100001183    100001987
1000002457   1000004249
100000002317
100000003559
```
38.Btst wt substr match
```
bitset<N>bs[26],oc;int main()
{int i,j,k,n,q,l,r;string s,p;
cin>>s;for(i=0;s[i];i++)bs[s
[i]-'a'][i]=1;cin>>q;while
(q--){cin>>p;oc.set();for(i=
0;p[i];i++)oc&=(bs[p[i]-'a']
>>i);cout<<oc.count()<<endl;//
number of occurences
int ans=N,sz=p.size();int
pos=oc._Find_first();v.
push_back(pos);pos=oc._Find_
next(pos);while(pos<N){v.
push_back(pos);pos=oc._Find_
next(pos);}for(auto x:v)
cout<<x<<' ';// position of
occurences
cout<<endl;v.clear();cin>>l
>>r;//number of occurences
from l to r,where l and r is
1-indexed
if(sz>r-l+1)cout<<0<<endl;
else cout<<(oc>>(l- 1)).
count()-(oc >>(r-sz+1))
.count()<< endl;}return 0;
```
39.extended euclid
```
ll extended_euclid(ll a
,ll b,ll &x,ll &y){if
(b==0){x=1;y=0;return
a;}ll x1,y1;ll d =
extended_euclid(b,a%b,
x1,y1);x=y1;y=x1-y1*
(a/b);return d;}
```
40. 3D GEO
```
struct p3{double x,y,z;p3()
{x=0,y=0;z=0;}p3(double x,
double y,double z):x(x),y(y),
z(z){}p3(const p3 &p):x(p.x),
y(p.y),z(p.z){}void scan()
{cin>>x>>y>>z;}p3 operator+
(const p3 &a)const{return
p3(x+a.x,y+a.y,z+a.z);}p3
operator-(const p3 &a)const
{return p3(x-a.x,y-a.y,z-a.z)
;}p3 operator*(const double
a)const{return p3(x*a,y*a,
z*a);}friend p3 operator*(
const double &a,const p3 &b)
{return p3(a*b.x,a*b.y,a*b.z)
;}p3 operator/(const double
a)const{return p3(x/a,y/a,
z/a);}bool operator==(p3
a)const{return sign(a.x-x)
==0&&sign(a.y-y)==0&&sign
(a.z-z)==0;}bool operator!=
(p3 a)const{return !(*this==a
);}double abs(){return
sqrt(x*x+y*y+z*z);}double
sq(){return x*x+y*y+z*z;}p3
unit(){return *this/abs();}}
zero(0,0,0);double operator
| (p3 v,p3 w) { //dot product
return v.x*w.x+v.y*w.y+v.z
*w.z;}p3 operator*(p3 v,p3 w)
{//cross product
return
{v.y*w.z-v.z*w.y,v.z*w.x-v.
x*w.z,v.x*w.y-v.y*w.x};}
double sq(p3 v){return
v|v;}double abs(p3 v){return
sqrt(sq(v));}p3 unit(p3 v){
return v/abs(v);}inline double
dot(p3 a,p3 b){return
a.x*b.x+a.y*b.y+a.z*b.z;}
```

```
inline double dist2(p3 a,p3
b){return dot(a-b,a-b);}inline
double dist(p3 a,p3 b){return
sqrt(dot(a-b,a-b));}inline p3
cross(p3 a,p3 b){return
p3(a.y*b.z-a.z*b.y,a.z*b.x-
a.x*b.z,a.x*b.y-a.y*b.x);}
double orient(p3 p,p3 q,p3 r,
p3 s){return(q-p)*(r-p)|(s-p)
;}double orient_by_normal(p3
p,p3 q,p3 r,p3 n){return
(q-p)*(r-p)|n;}double
get_angle(p3 a,p3 b){double
costheta=dot(a,b)/a.abs()
/b.abs();return acos(max
((double)-1.0,min((double)1.0
,costheta)));}double
small_angle(p3 v,p3 w){return
acos(min(fabs(v|w)/abs(v)/abs
(w),(double)1.0));}struct
plane{p3 n;double d;//(n|p)=d
// From normal n and offset d
plane(p3 n,double
d):n(n),d(d){}// From normal n
and point P
plane(p3 n,p3 p):n(n),d(n|p)
{}// From three non-collinear
points P,Q,R
plane(p3 p,p3 q,p3
r):plane((q-p)*(r-p),p){}
double side(p3
p){return(n|p)-d;}double dist
(p3 p){return
fabs(side(p))/abs(n);}plane
translate(p3 t){return
{n,d+(n|t)};}plane shiftUp(
double dist){return
{n,d+dist*abs(n)};}p3 proj(p3
p){return p-n*side(p)/sq(n);}
p3 refl(p3 p){return
p-n*2*side(p)/sq(n);}};struct
coords{p3 o,dx,dy,dz;//From
three points P,Q,R on the
plane:
//build an orthonormal 3D basis
coords(p3 p,p3 q,p3
r):o(p){dx=unit(q-p);dz=unit(
dx*(r-p));dy=dz*dx;}//From
four points P,Q,R,S: take
directions PQ,PR,PS as is
//it allows us to keep using
integer coordinates but has
some pitfalls.
coords(p3 p,p3 q,p3 r,p3
s):o(p),dx(q-p),dy(r-p),dz(s-p){}
PT pos2d(p3 p){return
{(p-o)|dx,(p-o)|dy};}p3 pos3d(p3
p){return
{(p-o)|dx,(p-o)|dy,(p-o)|dz};}p3
pos3d(PT p){ //original position
vector
return o+dx*p.x+dy*p.y;}};struct
line3d{p3 d,o;// p = o + k * d
// From two points P,Q
line3d(p3 p,p3 q):d(q-p),o(p){}//
From two planes p1,p2 (requires T
= double,planes are not parallel)
line3d(plane p1,plane
p2){d=p1.n*p2.n;o=(p2.n*p1.d-p1.n
*p2.d)*d/sq(d);}double dist2(p3
p){return
sq(d*(p-o))/sq(d);}double dist(p3
p){return sqrt(dist2(p));}bool
cmp_proj(p3 p,p3
q){return(d|p)<(d|q);}p3 proj(p3
p){return o+d*(d|(p-o))/sq(d);}p3
refl(p3 p){return proj(p)*2-p;}p3
inter(plane p) { return o - d *
p.side(o) / (d | p.n);} //
assuming plane and line are not
parallel
};double dist(line3d l1,line3d
l2){p3 n=l1.d*l2.d;if (n == zero)
return l1.dist(l2.o);// parallel
return
fabs((l2.o-l1.o)|n)/abs(n);}p3
closest_on_l1(line3d l1,line3d
l2){p3 n2=l2.d*(l1.d*l2.d);
return l1.o+l1.d*((l2.o
-l1.o)|n2)/(l1.d|n2);}double
get_angle(line3d l1,line3d
l2){return
small_angle(l1.d,l2.d);}bool
is_parallel(line3d l1,line3d
l2){return l1.d*l2.d==zero;}
bool is_perpendicular(line3d
l1,line3d l2){return
sign((l1.d|l2.d))==0;}double
get_angle(plane p1,plane
p2){return
small_angle(p1.n,p2.n);}bool
is_parallel(plane p1,plane
p2){return p1.n*p2.n==zero;}
bool is_perpendicular(plane
p1,plane p2){return
sign((p1.n|p2.n))==0;}double
get_angle(plane p,line3d
l){return PI/2-
small_angle(p.n,l.d);}
bool is_parallel(plane p,
line3d l){return sign(
(p.n|l.d))==0;}bool
is_perpendicular(plane p,
line3d l){return p.n*l.d==
zero;}//returns two points on
intesection line of two planes
formed by points
//a1,b1,c1 and a2,b2,c2
respectively
pair<p3,p3>
plane_plane_intersection(p3
a1,p3 b1,p3 c1,p3 a2,p3 b2,p3
c2){p3 n1=(b1-a1)*(c1-a1);p3
n2=(b2-a2)*(c2-a2);double
d1=n1|a1,d2=n2|a2;p3
d=n1*n2;if(d==zero)return
make_pair(zero,zero);p3
o=(n2*d1-n1*d2)*d/(d|d);return
make_pair(o,o+d);}//returns
center of circle passing
through three
// non-colinear and co-planer
points a,b and c
p3 circle_center(p3 a,p3 b,p3
c){p3 v1=b-a,v2=c-a;double
v1v1=v1|v1,v2v2=v2|v2,v1v2=
v1|v2;double
base=0.5/(v1v1*v2v2-v1v2*
v1v2);double
k1=base*v2v2*(v1v1-v1v2);
double k2=base*v1v1*(v2v2-
v1v2);
return a+v1*k1+v2*k2;}//segment
ab to point c
double distance_from_segment_
to_point(p3 a,p3 b,p3
c){if(sign(dot(b-a,c-a))<0)
return
dist(a,c);if(sign(dot(a-b,c-
b))<0)return dist(b,c);return
fabs(cross((b-a).unit(),c-a)
.abs());}double
distance_from_triangle_to_
point(p3 a,p3 b,p3 c,p3 d){
plane P(a,b,c);p3
proj=P.proj(d);double
```

```
dis=min(distance_from_segment
_to_point(a,b,d),min(distance
_from_segment_to_point(b,c,d
),distance_from_segment_to_
point(c,a,d)));int o=
sign(orient_by_normal(a,b,
proj,P.n));int
inside=o==sign(orient_by_
normal(b,c,proj,P.n));
inside&=o==sign(orient_by_
normal(c,a,proj,P.n));if(
inside)return(d-proj).abs();
return dis;}double
distance_from_triangle_to_
segment(p3 a,p3 b,p3 c,p3 d,
p3 e){double l=0.0,r=1.0;int
cnt=100;double ret =
1e12;//beware!
while(cnt--){double
mid1=l+(r-l)/3.0,mid2=r-(r-
l)/3.0;double
x=distance_from_triangle_to
_point(a,b,c,d+(e-d)*mid1);
double y=
distance_from_triangle_to_
point(a,b,c,d+(e-d)*mid2);
if(x<y){r=mid2;ret=x;}else{
ret=y;l=mid1;}}return ret;}
double distance_from_triangle
_to_
triangle(p3 a,p3 b,p3 c,p3 d
,p3 e,p3 f){double ret =
1e12;// beware!
ret=min(ret,distance_from_
triangle_to_segment(a,b,c,
d,e));ret=min(ret,distance_
from_triangle_to_segment(a,b
,c,e,f));ret=min(ret,distance
_from_triangle_to_segment(a,
b,c,f,d));ret=min(ret,
Distance_from_triangle_to_
segment(d,e,f,a,b));ret=min(
Ret,distance_from_triangle_to
_segment(d,e,f,b,c));ret=min
(ret,distance_from_triangle
_to_segment(d,e,f,c,a));
return ret;}bool operator<
(p3 p,p3 q){return
tie(p.x,p.y,p.z)<tie(q.x,
q.y,q.z);}struct edge{int v;
bool same;// is the common
edge in the same order?
};// Given a series of faces
(lists of points),reverse some of
them
// so that their orientations
are consistent
vector<vector<p3>>
reorient(vector<vector<p3>>
fs){int n=fs.size();// Find
the common edges and create
the resulting graph
vector<vector<edge>>
g(n);map<pair<p3,p3>,int> es;
for(int u=0;u<n;u++){for(int
i=0,m=fs[u].size();i<m;i++){p3
a=fs[u][i],b=fs[u][(i+1)%m];
// Let's look at edge [AB]
if(es.count({a,b})) { //
seen in same order
int v=es[{a,b}];g[u].push_
back({v,true});g[v].push_back
({u,true});}else if(es.count
({
b,a})) { // seen in different
order
int v=es[
{b,a}];g[u].push_back({v,false});
g[v].push_back({u,false});}else {
// not seen yet
es[{a,b}]=u;}}}// Perform BFS to
find which faces should be
flipped
vector<bool>
vis(n,false),flip(n);flip[0]=fals
e;queue<int>
q;q.push(0);while(!q.empty()){int
u=q.front();q.pop();for(edge
e:g[u]){if(!vis[e.v]){vis[e.v]=tr
ue;// If the edge was in the same
order,
// exactly one of the two should
be flipped
flip[e.v]=(flip[u]^e.same);q.push
(e.v);}}}// Actually perform the
flips
for(int
u=0;u<n;u++)if(flip[u]){reverse(f
s[u].begin(),fs[u].end());}return
fs;}struct CH3D{struct face{int
a,b,c;// the number of three
points on one face of the convex
hull
bool ok;// whether the face
belongs to the face on the final
convex hull
};int n;// initial vertex number
vector<p3> P;int num;// convex
hull surface triangle number
vector<face> F;// convex
surface triangles
vector<vector<int>> g;void
init(vector<p3>
p){P=p;n=p.size();F.resize(
8*n+1);g.resize(n+1,vector<
int>(n+1));}double len(p3
a){return sqrt(a|a);}p3
cross(const p3 &a,const p3
&b,const p3
&c){return(b-a)*(c-a);}double
area(p3 a,p3 b,p3 c){return
len((b-a)*(c-a));}double
volume(p3 a,p3 b,p3 c,p3
d){return(b-a)*(c-a)|(d-a);}
// positive: p3 in the same
direction
double dblcmp(p3 &p,face &f)
{p3 m=P[f.b]-P[f.a];p3
n=P[f.c]-P[f.a];p3
t=p-P[f.a];return(m*n)|t;}
void deal(int p,int a,int b)
{int f = g[a][b];// search for
another plane adjacent to the
edge
face add;if(F[f].ok){if
(dblcmp
(P[p],F[f])>eps)
dfs(p,f);else{add.a=b;add.b
=a;add.c = p;// pay attention
 to the order here,to be
right-handed
add.ok=true;g[p][b]=g[a][p]
=g[b][a]=num;F[num++]=add
;}}}// recursively search all
faces that should be removed
from the convex hull
void dfs(int p,int
now){F[now].ok=0;deal(p,F[
now].b,F[now].a);deal(p,F[
now].c,F[now].b);deal(p,F[
now].a,F[now].c);}bool same
(int s,int t){p3 &a=P[F[s].
a];p3 &b=P[F[s].b];p3
&c=P[F[s].c];return
fabs(volume(a,b,c,P[F[t].a]))
<eps&&fabs(volume(a,b,c,
```

```
P[F[t].b]))<eps&&fabs(volume
(a,b,c,P[F[t].c]))<eps;}//
building a 3D convex hull
void create_hull(){int
i,j,tmp;face
add;num=0;if(n<4)return;//
ensure that the first four
points are not coplanar
bool flag=true;for(i=1;i<n;
i++){if(len(P[0]-P[i])>eps)
{swap(P[1],P[i]);flag=false;
break;}}if(flag)return;
flag=true;// make the first
three points not collinear
for(i=2;i<n;i++){if(len((
P[0]-P[1])*(P[1]-P[i]))>eps)
{swap(P[2],P[i]);flag=false;
break;}}if(flag)return;flag=
true;// make the first four
points not coplanar
for(int i=3;i<n;i++)
{if(fabs((P[0]-P[1])*(P[1]-
P[2])|(P[0]-P[i]))>eps){
swap(P[3],P[i]);flag=false;
break;}}if(flag)return;for(
i=0;i<4;i++){add.a=(i+1)%4;
add.b=(i+2)%4;add.c=(i+3)%4;
add.ok=true;if(dblcmp(P[i],
add)>0)swap(add.b,add.c);
g[add.a][add.b]=g[add.b][
add.c]=g[add.c][add.a]=num;
F[num++]=add;}for(i=4;i<n;
i++){for(j=0;j<num;j++){if(
F[j].ok&&dblcmp(P[i],F[j])
>eps){dfs(i,j);break;}}}tmp
=num;for(i=num=0;i<tmp;i++)
if(F[i].ok)F[num++]=F[i];}
double surface_area(){double
res=0;if(n==3){p3
p=cross(P[0],P[1],P[2]);res
=len(p)/2.0;return res;}for
(int i=0;i<num;i++){res+=
area(P[F[i].a],P[F[i].b],
P[F[i].c]);}return
res/2.0;}double volume(){
double res=0;p3 tmp(0,0,0)
;for(int i=0;i<num;i++){
res+=volume(tmp,P[F[i].a],
P[F[i].b],P[F[i].c]);}return
fabs(res/6.0);}int
number_of_triangles() { //
number of surface triangles
return num;}int
number_of_polygons() {//number of
surface polygons
int i,j,res,flag;for(i=res=0;
i<num;i++){flag=1;for(
j=0;j<i;j++){if(same(i,j)){
flag=0;break;}}res+=flag;}
return res;}p3 centroid() {//
center of gravity
p3 ans(0,0,0),o(0,0,0);double
all=0;for(int
i=0;i<num;i++){double
vol=volume(o,P[F[i].a],P[F[i]
.b],P[F[i].c]);ans=ans+(o+P
[F[i].a]+P[F[i].b]+P[F[i].c]
)/4.0*vol;all+=vol;}ans=ans
/all;return ans;}double
point_to_face_distance(p3 p,
int i){return
fabs(volume(P[F[i].a],P[F
[i].b],P[F[i].c],p)/len((P[
F[i].b]-P[F[i].a])*(P[F[i].c]
-P[F[i].a])));}};p3
get_sphere(double r,double
lat,double
lon){lat*=PI/180,lon*=PI/180;
return {r*cos(lat)*
cos(lon),r*cos(lat)*sin(lon),
r*sin(lat)};}int
sphere_line_intersection(p3
o,double r,line3d l,pair<p3,
p3> &out){double
h2=r*r-l.dist2(o);if (h2 < 0)
return 0;// the line doesn't
touch the sphere
p3 p=l.proj(o);p3 h = l.d *
sqrt(h2)/abs(l.d);// vector
parallel to l,of length h
out={p-h,p+h};return
1+(h2>0);}//The shortest
distance between two points A
and B on a sphere (O,r) is
//given by travelling along
plane OAB. It is called the
great-circle distance
double greatCircleDist(p3
o,double r,p3 a,p3 b){return
r*get_angle(a-o,b-o);}// Assume
that the sphere is centered at
the origin
// We will call a segment [AB]
valid if A and B are not
// opposite each other on the
sphere
bool validSegment(p3 a,p3
b){return
a*b!=zero||(a|b)>0;}bool
proper_intersection(p3 a,p3 b,p3
c,p3 d,p3 &out){p3 ab = a * b,cd
= c * d;// normals of planes OAB
and OCD
int
oa=sign(cd|a),ob=sign(cd|b),oc=si
gn(ab|c),od=sign(ab|d);out = ab *
cd * od;// four multiplications
=> careful with overflow!
return(oa!=ob&&oc!=od&&oa!=oc);}/
/ Assume that the sphere is
centered at the origin
bool point_on_sphere_segment(p3
a,p3 b,p3 p){p3
n=a*b;if(n==zero)return
a*p==zero&&(a|p)>0;return(n|p)==0
&&(n|a*p)>=0&&(n|b*p)<=0;}struct
Set:vector<p3>{using
vector::vector;// import
constructors
void insert(p3 p){for(p3
q:*this)if(p*q==zero)return;push_
back(p);}};// Assume that the
sphere is centered at the origin
Set
segment_segment_intersection_on_s
phere(p3 a,p3 b,p3 c,p3
d){assert(validSegment(a,b)
&&validSegment(c,d));p3
out;if(proper_intersection
(a,b,c,d,out))return {out};
Set s;if(point_on_
sphere_segment(c,d,a))
s.insert(a);if(point_
on_sphere_segment(c,d,b))s.
insert(b);if(point_on_sphere
_segment(a,b,c))s.insert(c);
if(point_on_sphere_segment(a,
b,d))s.insert(d);return s;}
double angle_on_sphere(p3 a,
p3 b,p3 c){return
get_angle(a*b,a*c);}// Assume
that the sphere is centered at
the origin
double
oriented_angle_on_sphere(p3 a,
p3 b,p3 c){if((a*b|c)>=0)
```

```
return angle_on_sphere(a,b,c)
;else return
2*PI-angle_on_sphere(a,b,c);}
// Assume that the sphere is
centered at the origin
double
area_on_the_surface_of_the_
sphere(double r,vector<p3>
p){int n=p.size();double
sum=-(n-2)*PI;for(int
i=0;i<n;i++){sum+=oriented_
angle_on_sphere(p[(i+1)%n],
p[(i+2)%n],p[i]);}return
r*r*sum;}// 0 if O is outside
the polyhedron
// 1 if O is inside the
polyhedron,and the vector
areas of the faces are
oriented towards the outside;
// -1 if O is inside the
polyhedron,and the vector
areas of the faces are
oriented towards the inside.
// Assume that O is the origin
int winding_number_3D(vector<
vector<p3>> fs){double
sum=0;for(vector<p3>
f:fs){sum+=remainder(area_on
_the_surface_of_the_sphere
(1,f),4*PI);}return
round(sum/(4*PI));}int32_t
main(){ios_base::sync_with
_stdio(0);cin.tie(0);int
n;while(cin>>n){vector<p3>
p(n);for(int
i=0;i<n;i++)p[i].scan();CH3D
hull;hull.init(p);hull.create
_hull();cout<<hull.number_of
_polygons()<<'\n';}return 0;}
41.L-R Flow
const int N=3e5+9;const long
long inf=1LL<<61;struct
Dinic{struct edge{int to,rev;
long long flow,w;int id;};int
n,s,t,mxid;vector<int>
d,flow_through;vector<int>
done;vector<vector<edge>>
g;Dinic(){}Dinic(int
_n){n=_n+10;mxid=0;g.resize(n)
;}void add_edge(int u,int v,
long long w,int id=-1){edge
a={v,(int)g[v].size(),0,w,id}
;edge b = {u,(int)g[u].
size(),0,0,-1};//for
bidirectional edges cap(b)=w
g[u].emplace_back(a);g[v].
emplace_back(b);mxid=max(
mxid,id);}bool
bfs(){d.assign(n,-1);d[s]=0;
queue<int> q;q.push(s);
while(!q.empty()){int
u=q.front();q.pop();for(auto
&e:g[u]){int
v=e.to;if(d[v]==-1&&e.flow<
e.w)d[v]=d[u]+1,q.push(v);}}
return d[t]!=-1;}long long
dfs(int u,long long
flow){if(u==t)return flow;for
(int &i=done[u];
i<(int)g[u].size();i++){edge
&e=g[u][i];if(e.w<=e.flow)
continue;int
v=e.to;if(d[v]==d[u]+1){long
long nw=dfs(v,min(
flow,e.w-e.flow));if(nw>0)
{e.flow+=nw;g[v][e.rev].flow
-=nw;return nw;}}}return 0;}
long long max_flow(int _s,int
_t){s=_s;t=_t;long long
flow=0;while(bfs()){done.
assign(n,0);while(long long
nw=dfs(s,inf))flow+=nw;}flow
_through.assign(mxid+10,0);
for(int i=0;i<n;i++)for(auto
e:g[i])if(e.id>=0)flow_through
[e.id]=e.flow;return
flow;}};//flow_through[i]=
extra flow beyond 'low' sent
through edge i
struct LR_Flow{Dinic F;int
n,s,t;struct edge{int
u,v,l,r,id;};vector<edge>
edges;LR_Flow(){}LR_Flow(int
_n){n=_n+10;s=n-2,t=n-1;;
edges.clear();}void add_edge(
int u,int v,int l,
int r,int
id=-1){assert(0<=l&&l<=r);
edges.push_back({u,v,l,r,id
});}bool feasible(int _s=-1,
int _t=-1,int L=-1,int
R=-1){if(L!=-1)edges.push_back
({_t,_s,L,R,-1});F=Dinic(n);
long long target=0;for(auto
e:edges){int
u=e.u,v=e.v,l=e.l,r=e.r,id=e.
id;if(l!=0){F.add_edge(s,v,l);
F.add_edge(u,t,l);target+=l;}
F.add_edge(u,v,r-l,id);}auto
ans=F.max_flow(s,t);if(L!=-1)
edges.pop_back();if (ans <
target) return 0;//not
Feasible return 1;}
int max_flow(int _s,int _t){
//-1 means flow is not feasible
int
mx=1e5+9;if(!feasible(_s,_t,0,
mx))return -1;return
F.max_flow(_s,_t);}int
min_flow(int _s,int _t) { //-1
means flow is not feasible
int mx=1e9;int
ans=-1,l=0,r=mx;while(l<=r){int
mid=l+r>>1;if(feasible(_s,_t,0,mi
d))ans=mid,r=mid-1;else
l=mid+1;}return ans;}};int
get_id(map<int,int> &mp,int
k){if(mp.find(k)==mp.end())mp[k],
mp[k]=mp.size();return mp[k];}int
Lx[N],Rx[N],Ly[N],Ry[N],degx[N],d
egy[N];int32_t
main(){ios_base::sync_with_stdio(
0);cin.tie(0);int
n,m;cin>>n>>m;LR_Flow
F(2*n+10);int r,b;cin>>r>>b;int
sp=r>b;if(sp)swap(r,b);map<int,in
t> mx,my;for(int
i=1;i<=n;i++){int
x,y;cin>>x>>y;if(sp)swap(x,y);F.a
dd_edge(get_id(mx,x),get_id(my,y)
+n,0,1,i);degx[mx[x]]++;degy[my[y
]]++;}for(int
i=1;i<=mx.size();i++)Lx[i]=0,Rx[i
]=degx[i];for(int
i=1;i<=my.size();i++)Ly[i]=0,Ry[i
]=degy[i];while(m--){int
ty,x,d;cin>>ty>>x>>d;ty--;ty^=sp;
if(ty==0){if(mx.find(x)!=mx.end()
){int i=mx[x];int p=degx[i];int
l=(p-d+1)/2,r=(p+d)/2;l=max(0,l);
r=min(p,r);Lx[i]=max(Lx[i],l);Rx[
i]=min(Rx[i],r);}}else{if(my.find
(x)!=my.end()){int i=my[x];int
p=degy[i];int
l=(p-d+1)/2,r=(p+d)/2;l=max(0,l);
r=min(p,r);Ly[i]=max(Ly[i],l);Ry[
```

```
i]=min(Ry[i],r);}}}int
s=2*n+2,t=s+1;for(int
i=1;i<=mx.size();i++){if(Lx[i]>Rx
[i])return
cout<<-1<<'\n',0;F.add_edge
(s,i,Lx[i],Rx[i]);}for(int
i=1;i<=my.size();i++){if(
Ly[i]>Ry[i])return
cout<<-1<<'\n',0;F.add_edge
(i+n,t,Ly[i],Ry[i]);}int
c=F.max_flow(s,t);if(c==-1)
return cout<<-1<<'\n',0;long
long ans=1LL*c*r+1LL*
(n-c)*b;cout<<ans<<'\n';for
(int i=1;i<=n;i++){cout<<
"br"[F.F.flow_through[i]^sp];}cou
t<<'\n';return 0;}
```

42.CRT and Lucas

```
// Use int128 instead of ll
ll lucas(int n, int k, int p){
inv[0]= inv[1]=fac[0]=fac[1]=1;
for( int i=2;i<=p;i++) {fac[i]=
(fac[i-1]*i)%p;inv[i]=((ll)
(p-p/i)*inv[p%i])%p;}int x, y;
ll res=1;while(k){x=n%p;y=k%p;
if(x<y)return 0;res*=
(fac[x]*inv[fac[y]]*inv[fac[x -
y]])%p;res %=p;n/=p;k/=p;}
return res;}ll ext_gcd(ll A,
ll B,ll *X,ll *Y){ll x2,y2,x1,
y1,x,y,r2,r1,q,r;x2=1;y2=0;
x1=0;y1=1;for(r2=A,r1=B;r1!=0;
r2 = r1, r1 = r, x2 = x1, y2
= y1, x1 = x, y1 = y ) {
q = r2 / r1; r = r2 % r1;
x = x2 - (q * x1);
y = y2 - (q * y1); }
*X = x2; *Y = y2;
return r2;}
ll crt(ll *A, ll *M, ll n) {
ll a1 = A[0];ll m1 = M[0];
for ( ll i = 1; i < n; i++ ) {
ll a2 = A[i]; ll m2 = M[i];
ll g = __gcd(m1, m2);
if ( a1 % g != a2 % g ) return
-1;
ll p, q;
ext_gcd(m1/g, m2 / g, &p, &q);
ll mod = m1 / g * m2;
ll x=(a1*(m2/g)*q+a2*(m1/g)*p)
% mod; a1 = x;
if (a1 < 0) a1 += mod;
m1 = mod; } return a1; }
int main() { int n,r,m;
cin>>n>>r>>m;
std::vector<int> v;
for( int i=2;i<=50;i++ ){
if(m%i)continue;
v.push_back(i);
while(m%i==0)m/= i; }
ll arr[v.size()+5],
mods[v.size()+5]; int cc= 0;
for( auto x: v ){ arr[cc]= x;
mods[cc++]= lucas(n,r,x);}
ll ans= crt( mods, arr, cc );}
```

43. Mo on tree

```
inline void toggle(int id)
{if(arr[base[id]]<mx and active[
base[id]]){cnt[ arr[ base[
id]]]--;
if(!cnt[ arr[ base[ id ] ] ] )
nai[ arr[ base[ id ] ] ]= 1; }
else if(arr[ base[ id ] ]<mx){
cnt[ arr[ base[ id ] ] ]++;
if(cnt[arr[ base[ id ]]]==1)
nai[ arr[ base[ id ] ] ]=
0;}active[ base[ id ] ]^= 1;}
for(int i=1; i<=q; i++ )
{sf(a);sf(b);int L= lca(a,b);
if( st[a]>st[b] ) swap(a,b);
if( a==L ) { qq[i].l= st[ a ]+1;
 qq[i].r= st[ b ];}
else{qq[i].l= en[ a ];
qq[i].r= st[ b ];}qq[i].id= i;
 qq[i].calcord();}
for( int i=1; i<=q; i++ ){
 while(l<qq[i].l) toggle(l++);
 while(r<qq[i].r) toggle(++r);
while(l>qq[i].l) toggle(--l);
while(r>qq[i].r) toggle(r--);
ans[qq[i].id]=nai._Find_first();
```

44. Discrete Log

```
ll a,b; ll ans= INT_MAX;
scanf("%lld %lld", &a, &b);
for( ll i=sq;i>=1;i-- )
mp[ bigmod( a, i*sq )%mod ]= i;
for( ll i=0;i<=sq;i++ )
{ll num= (b*bigmod( a, i ))%mod;
if( mp.count( num ) )
ans= min(ans,mp[ num ]*sq-i);
```

45. Bridge Tree

```
vector<int>tree[N],g[N];//edge
list representation of graph int
U[M],V[M],vis[N],arr[N],T,dsu[N];
bool isbridge[M];//if i'th edge
is a bridge edge or not
int adj(int u,int e){return
U[e]^V[e]^u;}int f(int x){return
dsu[x]=(dsu[x]==x?x:f(dsu[x]));}v
oid merge(int a,int
b){dsu[f(a)]=f(b);}int dfs0(int
u,int edge) { //mark bridges
vis[u]=1;arr[u]=T++;int
dbe=arr[u];for(auto e:g[u]){int
w=adj(u,e);if(!vis[w])dbe=min(dbe
,dfs0(w,e));else
if(e!=edge)dbe=min(dbe,arr[w]);}i
f(dbe==arr[u]&&edge!=-1)isbridge[
edge]=true;else
if(edge!=-1)merge(U[edge],V[edge]
);return dbe;}void
buildBridgeTree(int n,int
m){for(int
i=1;i<=n;i++)dsu[i]=i;for(int
i=1;i<=n;i++)if(!vis[i])dfs0(i,-1
);for(int
i=1;i<=m;i++)if(f(U[i])!=f(V[i]))
{tree[f(U[i])].push_back(f(V[i]))
;tree[f(V[i])].push_back(f(U[i]))
;}}
```

46. Pragma

```
#pragma
comment(linker,"/stack:200000000"
)#pragma GCC
optimize("Ofast")#pragma GCC
target("sse,sse2,sse3,ssse3,sse4,
popcnt,abm,mmx,avx")#pragma GCC
optimize("unroll-loops")
```

47. Rolling Hash

```
struct HASH{size_t operator()(
const pair<int,int>&x)const{
return hash<long long>()(((long
long)x.first)^(((long long)
x.second)<<32));}};int
inv[SIZE];inv[1]=1;for(int
i=2;i<=n;i++){inv[i]=(-(m/i)*inv[
m%i])%m;inv[i]=inv[i]+m;}
mt19937 mrand(random_device{}());
int rnd(int x){return
mrand()%x;}typedef pair<int,int>
hashv;const ll
mod1=1000000007;const ll
mod2=1000000009;hashv
operator+(hashv a,hashv b){int
c1=a.ff+b.ff,c2=a.ss+b.ss;if(c1>=
```

```
mod1)c1-=mod1;if(c2>=mod2)c2-=mod
2;return {c1,c2};}hashv
operator-(hashv a,hashv b){int
c1=a.ff-b.ff,c2=a.ss-b.ss;if(c1<0
)c1+=mod1;if(c2<0)c2+=mod2;return
{c1,c2};}hashv operator*(hashv
a,hashv b){return
{1LL*a.ff*b.ff%mod1,1LL*a.ss*b.ss
%mod2};}int main(){for(int
i='A';i<='C';++i)ff[i]=hashv(rnd(
mod1),rnd(mod2));base=hashv(13331
,23333);pw[0]=hashv(1,1);for(int
i=1;i<=n;++i)pw[i]=pw[i-1]*base;h
s2[0]=hashv(0,0);for(int
i=0;i<n;++i){hs2[0]=hs2[0]*base+f
f[s2[i]];}//hashv val = ths[i] -
ths[i-sz[j]]*pw[sz[j]];}
```

48. SOS DP

```
for(int mask=0;mask<(1<<N);++mask
){dp[mask][-1] = A[mask];for(int
i=0;i<N;++i){if(mask&(1<<i))dp[ma
sk][i]=dp[mask][i-1]+dp[mask^(1<<
i)][i-1];else dp[mask][i]=dp[mask
][i-1];}F[mask]=dp[mask][N-1];}fo
r(int i=0;i<(1<<N);++i)F[i]=A[i];
for(int i=0;i<N;++i)for(int mask=
0;mask<(1<<N);++mask){if(mask&(1<
<i))F[mask]+=F[mask^(1<<i)];}
```

49. SCC

```
void dfs1(int u){vis[u]=true;for(
auto v: adj[u]){if(!vis[v])dfs1(v
);}vec.push_back(u);}vector<int>
comp;void dfs2(int u){
comp.push_back(u);vis[u]=true;for
(auto v: rev[u]){if(!vis[v])
dfs2(v);}}main(){for(int i=1;i<=n
;++i)if(!vis[i])dfs1(i);reverse(v
ec.begin(),vec.end());for(int i=1
;i<=n;++i)vis[i]=false;int scc=0;
for(auto u: vec){if(!
vis[u]){comp.clear();dfs2(u);scc+
+;for(auto x: comp)idx[x]=scc;}}}
```

50. Gauess eliminition

```
#include<bits/stdc++.h>
using namespace std;const int
N=3e5+9;const double eps=1e-9;int
Gauss(vector<vector<double>>
a,vector<double> &ans){int
n=(int)a.size(),m=(int)a[0].size(
)-1;vector<int> pos(m,-1);double
det=1;int rank=0;for(int
col=0,row=0;col<m&&row<n;++col){i
nt mx=row;for(int
i=row;i<n;i++)if(fabs(a[i][col])>
fabs(a[mx][col]))mx=i;if(fabs(a[m
x][col])<eps){det=0;continue;}for
(int
i=col;i<=m;i++)swap(a[row][i],a[m
x][i]);if(row!=mx)det=-det;det*=a
[row][col];pos[col]=row;for(int
i=0;i<n;i++){if(i!=row&&fabs(a[i]
[col])>eps){double
c=a[i][col]/a[row][col];for(int
j=col;j<=m;j++)a[i][j]-=a[row][j]
*c;}}++row;++rank;}ans.assign(m,0
);for(int
i=0;i<m;i++){if(pos[i]!=-1)ans[i]
=a[pos[i]][m]/a[pos[i]][i];}for(i
nt i=0;i<n;i++){double
sum=0;for(int
j=0;j<m;j++)sum+=ans[j]*a[i][j];i
f(fabs(sum - a[i][m]) > eps)
return -1;//no solution
}for(int i = 0;i < m;i++)
if(pos[i] == -1) return
2;//infinte solutions
return 1;//unique solution
}int main(){int
n,m;cin>>n>>m;vector<
vector<double>>v(n);for(int
i=0;i<n;i++){for(int
j=0;j<=m;j++){double
x;cin>>x;v[i].push_back(x);}}vect
or<double> ans;int
k=Gauss(v,ans);if(k)for(int
i=0;i<n;i++)cout<<fixed<<setpreci
sion(5)<<ans[i]<<' ';else
cout<<"no solution\n";return 0;}
```

51. Matexpon

```
#include<bits/stdc++.h>using
namespace
std;const int mod=10007;#define
MODMATEXPO mod #define ll  long
long
template<typename intex,int
N>struct matrix{intex
x[N+1][N+1];matrix(intex
v=0){memset(x,0,sizeof x);for(int
i=1;
i<=N;i++){x[i][i]=v;}}intex*opera
tor[]
(int a){return x[a];}inline
matrix operator*(const matrix
&r){matrix<intex,N> p(0);for(int
k=1;k<=N;++k){for(int
i=1;i<=N;++i){if(x[i][k]==0)conti
nue;for(int
j=1;j<=N;++j){p.x[i][j]+=x[i][k]*
r.x[k][j];#ifdef
MODMATEXPOp.x[i][j] %=
MODMATEXPO;#endif}}}return
p;}inline matrix mul(const matrix
&r,const int col){matrix<intex,N>
p(0);for(int
k=1;k<=N;++k){for(int
i=1;i<=N;++i){if(x[i][k]==0)conti
nue;for(int
j=1;j<=col;++j){p.x[i][j]+=x[i][k
]*r.x[k][j];#ifdef
MODMATEXPOp.x[i][j] %=
MODMATEXPO;#endif}}}return
p;}inline matrix pow(ll p){matrix
r(1),a=*this;for(;p>0 ;p >>=
1){if(p&1)r=r*a;a=a*a;}return
r;}intex determinant()const{intex
r=1,det[N+1][N+1];for(int
i=1;i<=N;++i)memcpy(det[i],x[i],s
izeof(x[i]));for(int
i=1;i<=N;++i){for(int
j=i+1;j<=N;++j){while(det[j][i]!=
0){intex
ratio=det[i][i]/det[j][i];for(int
k=i;k<=N;++k){det[i][k]-=ratio*de
t[j][k];swap(det[i][k],det[j][k])
;}r=-r;}}r=r*det[i][i];}return
r;}void
print(){cout<<N<<'\n';for(int
i=1;i<=N;i++){for(int
j=1;j<=N;j++)cout<<x[i][j]<<"
\n"[j==N];}}};int main(int
argc,char const *argv[]){int
n;cin>>n;matrix<int,3>
mat;for(int i=1;i<=n;i++){for(int
j=1;j<=n;j++){cin>>mat.x[i][j];//
cin >> mat[i][j];
}}mat.print();// int t;
// scanf("%d",&t);
// for(int I=1;I<=t;I++)
// {
//  printf("Case %d: ",I);
//  int a,b,c,d,n;
//  scanf("%d %d %d
%d",&n,&a,&b,&c);
//  matrix<int,4> arr;
//  matrix<int,4> brr;
//  int br[]={0,0,0,c};
```

```
//  brr.init(br,4,1);
//  int
ar[]={a,0,b,1,1,0,0,0,0,1,0,0,0,0
,0,1};
//  arr.init(ar,4,4);
//  arr=arr.pow(n);
//  arr=arr*brr;lca
//  //
arr=arr.mul(brr,1);//matrix,col
//  printf("%d\n",arr.x[3][1]);
// }
return 0;}
```

52. Wavelet tree

```
#include<bits/stdc++.h>
using namespace std;//careful for
multiple query
//delete pointer properly
const int MAXN=(int)3e5+9;const
int MAXV = (int)1e9 + 9;//maximum
value of any element in array
//array values can be negative
too,use appropriate minimum and
maximum value
struct wavelet_tree{int
lo,hi;wavelet_tree *l,*r;int
*b,*c,bsz,csz;// c holds the
prefix sum of elements
wavelet_tree(){lo=1;hi=0;bsz=0;cs
z=0,l=NULL;r=NULL;}void init(int
*from,int *to,int x,int
y){lo=x,hi=y;if(from>=to)return;i
nt mid=(lo+hi)>>1;auto
f=[mid](int x){return
x<=mid;};b=(int*)malloc((to-from+
2)*sizeof(int));bsz=0;b[bsz++]=0;
c=(int*)malloc((to-from+2)*sizeof
(int));csz=0;c[csz++]=0;for(auto
it=from;it!=to;it++){b[bsz]=(b[bs
z-1]+f(*it));c[csz]=(c[csz-1]+(*i
t));bsz++;csz++;}if(hi==lo)return
;auto
pivot=stable_partition(from,to,f)
;l=new
wavelet_tree();l->init(from,pivot
,lo,mid);r=new
wavelet_tree();r->init(pivot,to,m
id+1,hi);}//kth smallest element
in [l,r]
//for array [1,2,1,3,5] 2nd
smallest is 1 and 3rd smallest is
2
int kth(int l,int r,int
k){if(l>r)return
0;if(lo==hi)return lo;int
inLeft=b[r]-b[l-1],lb=b[l-1],rb=b
[r];if(k<=inLeft)return
this->l->kth(lb+1,rb,k);return
this->r->kth(l-lb,r-rb,k-inLeft);
}//count of numbers in [l,r] Less
than or equal to k
int LTE(int l,int r,int
k){if(l>r||k<lo)return
0;if(hi<=k)return r-l+1;int
lb=b[l-1],rb=b[r];return
this->l->LTE(lb+1,rb,k)+this->r->
LTE(l-lb,r-rb,k);}//count of
numbers in [l,r] equal to k
int count(int l,int r,int
k){if(l>r||k<lo||k>hi)return
0;if(lo==hi)return r-l+1;int
lb=b[l-1],rb=b[r];int
mid=(lo+hi)>>1;if(k<=mid)return
this->l->count(lb+1,rb,k);return
this->r->count(l-lb,r-rb,k);}//su
m of numbers in [l ,r] less than
or equal to k
int sum(int l,int r,int
k){if(l>r||k<lo)return
0;if(hi<=k)return c[r]-c[l-1];int
lb=b[l-1],rb=b[r];return
this->l->sum(lb+1,rb,k)+this->r->
sum(l-lb,r-rb,k);}~wavelet_tree()
{delete l;delete
r;}};wavelet_tree t;int
a[MAXN];int main(){int
i,j,k,n,m,q,l,r;cin>>n;for(i=1;i<
=n;i++)cin>>a[i];t.init(a+1,a+n+1
,-MAXV,MAXV);//beware! after the
init() operation array a[] will
not be same
cin>>q;while(q--){int
x;cin>>x;cin>>l>>r>>k;if(x==0){/*
kth smallest*/
cout<<t.kth(l,r,k)<<endl;}else
if(x==1){//less than or equal to
K
cout<<t.LTE(l,r,k)<<endl;}else
if(x==2){//count occurence of K
in [l,r]
cout<<t.count(l,r,k)<<endl;}if(x=
=3){//sum of elements less than
or equal to K in [l,r]
cout<<t.sum(l,r,k)<<endl;}}return
0;}
```

53. Persistent segment tree

```
const int MAX=5e5+5;struct
vertex{int
l,r,val;vertex(){l=r=-1;val=0;}};
vector<vertex> tre;void
get_new(int
&x){x=tre.size();tre.emplace_back
();}void combine(int
n){if(~tre[n].l){tre[n].val=tre[t
re[n].l].val+tre[tre[n].r].val;}}
void init(int n,int l,int
r){if(l==r){return;}int
mid=(l+r)/2;get_new(tre[n].l);get
_new(tre[n].r);init(tre[n].l,l,mi
d);init(tre[n].r,mid+1,r);}int
update(int n,int l,int r,int
i,int val){if(i>r||i<l){return
n;}int
cur;get_new(cur);if(l==r&&r==i){t
re[cur].val=tre[n].val+val;return
cur;}int mid=(l+r)/2;int
lf=update(tre[n].l,l,mid,i,val);i
nt
rg=update(tre[n].r,mid+1,r,i,val)
;tre[cur].l=lf;tre[cur].r=rg;comb
ine(cur);return cur;}int
query(int n,int l,int r,int i,int
j){if(i<=l&&j>=r){return
tre[n].val;}if(i>r||j<l)return
0;int mid=(l+r)/2;return
query(tre[n].l,l,mid,i,j)+query(t
re[n].r,mid+1,r,i,j);}int get(int
nr,int nl,int l,int r,int
k){if(l==r){return l;}// cout <<
l << ' ' << r << '\n';
int mid=(l+r)/2;int
l1=tre[nr].l;int
l2=tre[nl].l;if((mid-l+1)-(tre[l1
].val-tre[l2].val)<k)return
get(tre[nr].r,tre[nl].r,mid+1,r,k
-((mid-l+1)-(tre[l1].val-tre[l2].
val)));else return
get(l1,l2,l,mid,k);}int
root[MAX];int main()
{ios_base::sync_with_stdio(0);cin
.tie(0);get_new(root[0]);int
mx=1000006;init(root[0],1,mx);int
n;cin>>n;for(int
i=1;i<=n;i++){int
a;cin>>a;root[i]=update(root[i-1]
,1,mx,a,1);}int
q;cin>>q;while(q--){int
```

```
l,r,k;cin>>l>>r>>k;if(l>r)swap(l,
r);if(mx-(r-l+1)<k)cout<<k+r-l+1<
<"\n";else
cout<<get(root[r],root[l-1],1,mx,
k)<<"\n";}}
```