1 2dgeo

```
int cmp(double x) {
  if(fabs(x)<EPS)return 0;
  return x<0?-1:1;}
struct PT{
  double x,y;
  PT(){x=y=0;}
  PT(double _x,double _y){x=_x,y=_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*(double a)const{
    return PT(x*a,y*a);}
  PT operator/(double a)const{
    return PT(x/a,y/a);}
  double val(){
    return sqrt(x*x+y*y);}
  void scan(){scanf("%lf%lf",&x,&y);}
  void print(){printf("(%.4f, %.4f)",x,y);}};
struct line{double a,b,c;};
double dist(PT a,PT b){return(a-b).val();}
double dist2(PT a,PT b){
  a=a-b;return a.x*a.x+a.y*a.y;}
double dot(PT a,PT b){
  return a.x*b.x+a.y*b.y;}
double cross(PT a,PT b){
  return a.x*b.y-a.y*b.x;}
PT RotateCCW90(PT p){
  return PT(-p.y,p.x);}
PT RotateCW90(PT p){
  return PT(p.y,-p.x);}
PT RotateCCW(PT p,double t){
  return PT(p.x*cos(t)-p.y*sin(t), p.x*sin(t)+p.y*cos(t));}
PT RotateCW(PT p,double t){
  return PT(p.x*cos(t)+p.y*sin(t), -p.x*sin(t)+p.y*cos(t));}
// project point c onto line segment through a and b
PT ProjectPointSegment(PT a,PT b,PT c){
  double r=dot(b-a,b-a);
  if(fabs(r)<EPS)return a;r=dot(c-a,b-a)/r;
  if (r<0)return a;if (r>1)return b;
  return a+(b-a)*r;}
// compute distance from c to segment between a and b
double DistancePointSegment(PT a,PT b,PT c){
  return sqrt(dist2(c, ProjectPointSegment(a, b, c)));}
// returns bisector of angle YXZ
line bisector(PT Y,PT X,PT Z){
  PT xy=(Y-X)/(Y-X).val();
  PT xz=(Z-X)/(Z-X).val();
  PT d=xy+xz;
  line ret{d.y,-d.x,d.x*X.y-d.y*X.x};
  return ret;}
vector<PT> CircleLineIntersection(PT a,PT b,PT c,double r){
  vector<PT>ret;
  b=b-a;a=a-c;
  double A=dot(b, b);double B=dot(a, b);
  double C=dot(a, a)-r*r;double 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;}
PT ComputeLineIntersection(PT a,PT b,PT c,PT d){
  double a1=a.y-b.y;double b1=b.x-a.x;
  double c1=cross(a, b);double a2=c.y-d.y;
  double b2=d.x-c.x;double c2=cross(c, d);
  double D=a1*b2-a2*b1;
  return PT((b1*c2-b2*c1)/D,(c1*a2-c2*a1)/D);}
```

2 A

```
#include <bits/stdc++.h>
using namespace std;
#define ull unsigned long long
```

```
#define uint  unsigned int
#define f64 double
#define f128 __float128
#define ll long long
#define lll __int128
#define ulll __uint128_t
#define Te template<class T>
#define pb push_back
#define gu getchar_unlocked
#define pu putchar_unlocked
#define vll vector<ll>
#define EPS 1e-9
#define pi acos(-1.0)
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template <typename T>
using ordered_set =
tree<T, null_type, less<T>, rb_tree_tag,
tree_order_statistics_node_update>;
#pragma GCC optimize("Ofast")
//#pragma GCC target("sse,sse2,sse3,ssse3,
//sse4,popcnt,abm,mmx,avx,avx2,fma")
#pragma GCC optimize("unroll-loops")
//Iterating through all masks with their submasks. Complexity O(3^n)
/*for (int m=0; m<(1<<n); ++m)
   for (int s=m; s; s=(s-1)&m)*/
```

3 AP

```
void articulationPointAndBridge(int u) {///av=articulation_vertex
dfs_low[u]=dfs_num[u]=dfsNumberCounter++;//dfs_low[u]<=dfs_num[u]
for(int j=0;j<(int)adj[u].size();j++) {
int v=adj[u][j];
if(dfs_num[v.first]==UNVISITED) {//a tree edge
dfs_parent[v.first]=u;
if(u==dfsRoot)rootChildren++;// special case if u is a root
articulationPointAndBridge(v.first);
if(dfs_low[v.first]>dfs_num[u])// for articulation point
av[u] = true; // store this information first
if(dfs_low[v.first]>dfs_num[u])//for bridge
printf("Edge (%d, %d) is a bridge\n",u,v.first);
dfs_low[u]=min(dfs_low[u],dfs_low[v.first]); }//update dfs_low[u]
else if(v.first!=dfs_parent[u])//a back edge and not direct cycle
dfs_low[u]=min(dfs_low[u],dfs_num[v.first]);}}//update dfs_low[u]
// inside int main()
dfsNumberCounter=0;dfs_num.assign(V,UNVISITED);dfs_low.assign(V,0);
dfs_parent.assign(V,0);av.assign(V,0);
printf("Bridges:\n");
for(int i=0;i<V;i++)
if(dfs_num[i]==UNVISITED) {
dfsRoot=i;rootChildren=0;articulationPointAndBridge(i);
av[dfsRoot] = (rootChildren > 1); } // special case
printf("Articulation Points:\n");
for(int i=0;i<V;i++)
if(av[i])printf(" Vertex %d\n", i);
```

4 BIT

```
const static int M=100010;int V[M];//1 based
Te inline void add(T *a,int p,T v){for(;p<=M;p+=p&(-p))a[p]+=v;}
Te inline T query(T *a,int p){T s=0;for(;p>0;p-=p&(-p))s+=a[p];return
   s;}
```

5 Burnside

```
for( int i=1;i<=n;i++ )ans=(ans+bigmod( k, __gcd(i,n) ))%mod;
ll inv= bigmod(n,mod-2);ans= (ans*inv)%mod;
```

6 Complete Graph

```
void bfs( int x )
{queue<int> q;q.push( x );unv.erase( unv.find(x) );
while(!q.empty())
{cc++;int u= q.front();q.pop();
int sz= adj[u].size();vector<int> temp;
for( it= unv.begin(); it!=unv.end(); it++ ){int v= (*it);
if( !binary_search(adj[u].begin(),adj[u].end(), v) )
{temp.push_back(v);q.push(v);}}
for(int i=0; i<temp.size(); i++)unv.erase(unv.find(temp[i]));}}
main()
{for(int i=1; i<=n;
    i++)unv.insert(i),sort(adj[i].begin(),adj[i].end());
for(int i=1; i<=n; i++)
{if( unv.find(i)==unv.end())continue;
cc= 0;bfs(i);v.push_back(cc);}
```

7 Factorization Class

```
ull pow(ull a, ull t, ull mod){ull r=1;
for(;t;t>>=1,a=(ulll)(a)*a%mod)if(t&1)r=(ulll)(r)*a%mod;return r;}
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=pow(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;}
ulll ctz(ulll x){
return (ull)(x)?__builtin_ctzll(x):__builtin_ctzll(x>>64)+64;}
ulll gcd(ulll a, ulll b){
    if (!a||!b)return a|b;int s=ctz(a|b);
    for(b>>=ctz(b);a;a-=b)if((a>>=ctz(a))<b)swap(a,b);
        return b<<s;}
ull f(ull x,ull c,ull m){return ((ulll)x*x%m+c)%m;}
ull brent(ull n) {
    if(!(n&1))return 2;if(isprime(n))return n;
    random_device rd;mt19937 rnd(rd());
  ull x =2+rnd()%(n-2),g=1,q=1,xs,y,c=2+rnd()%(n-2);
  int m=128,l=1;while(g==1){y=x;
      for(int i=1;i<l;i++)x=f(x,c,n);
      int k=0;while(k<l&&g==1){xs=x;
      for(int i=0;i<m&&i<l-k;i++){
        x=f(x,c,n);q=(ulll)(q)*llabs(y-x)%n;}
        g=gcd(q,n);k+=m;}l<<=1;}
        if(g==n){do{xs=f(xs,c,n);
g=gcd(llabs(xs-y),n);}while(g==1);}return g;}
```

8 Hopcroft Karp

```
struct Hp{const int oo=1e9;int n;
    vector<int> mL,mR,d;vector<vector<int>>g;
    Hp(int n):n(n),mL(n+1),mR(n+1),d(n+1),g(n+1){}
    void edge(int u,int v){g[u].pb(v);/*both part diff 1..n*/}
    bool B(){queue<int> q;
    for(int u=1;u<=n;u++)if(mL[u])d[u]=oo;else d[u]=0,q.push(u);
    d[0]=oo;while(!q.empty()){int u=q.front();q.pop();
    for(auto v:g[u]){if(d[mR[v]]==oo)d[mR[v]]=d[u]+1,q.push(mR[v]);}}
    return d[0]!=oo;}
    bool D(int u){if(!u)return 1;for(auto
        v:g[u])if(d[mR[v]]==d[u]+1&&D(mR[v]))
    {mL[u]=v;mR[v]=u;return 1;}d[u]=oo;return false;}
    int A(){int r=0;while(B())for(int
        u=1;u<=n;u++)if(!mL[u]&&D(u))r++;return r;}};
```

9 Hungarian Matching

```
struct HM
{ll c[N][N],fx[N],fy[N],d[N];
    int mx[N],my[N],ar[N],tr[N];queue<int>q;
    int st,fin,n;
    ll oo=1e18;
    HM() {}HM(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 edge(int u, int v,ll cost)
    {c[u][v]=min(c[u][v],cost);}
    inline ll getC(int u,int v)
    {return c[u][v]-fx[u]-fy[v];}
    void B(){while(!q.empty())q.pop();q.push(st);
    for(int i=0;i<=n;++i)tr[i]=0;
    for(int v=1;v<=n;++v)d[v]=getC(st,v),ar[v]=st;}fin=0;}
    void Aug()
    {while(!q.empty())
    {int u=q.front();q.pop();
    for(int v=1;v<=n;++v)if (!tr[v])
    {ll w=getC(u, v);if(!w){tr[v]=u;
    if
        (!my[v]){fin=v;return;}q.push(my[v]);}if(d[v]>w)d[v]=w,ar[v]=u;}}}
    void sX_aY()
    {ll delta=inf;for(int v=1;v<=n;++v)
    if(tr[v]==0&&d[v]<delta)delta = d[v];// Rotate
    fx[st] += delta;
    for(int v=1;v<=n;++v)
    if(tr[v])int u = my[v],fy[v] -= delta,fx[u] += delta;
    else d[v] -= delta;
    for (int v = 1; v <= n; ++v)
    if (!tr[v] && !d[v])
    {tr[v] = ar[v];
    if (!my[v])
    {fin = v;return;}
    q.push(my[v]);}}
    void Enlarge()
    {do{int u = tr[fin];
    int nxt = mx[u];mx[u] = fin;
    my[fin] = u;fin = nxt;}while (fin);}
    ll 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)
    {st = u;B();
    while (!fin){Aug();if (!fin)subX_addY();}Enlarge();}
    ll ans = 0;
    for (int i = 1; i <= n; ++i)if(c[i][mx[i]]!=inf)
    ans += c[i][mx[i]];return ans;};
```

10 Manacher

```
const int mx=1010;
int d[2][mx],n;
int MK(string &S,bool T){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;}}}
```

11 Palindromic Tree

```
struct pt
{
```

```
    int next[MX][26],fail[MX];
    long long cnt[MX]; // Number of times the palindromic
    int ans; // sub string occurs in the string
    int num[MX]; // How many palindromic
    int st[MX],end[MX],len[MX],S[MX],last,n;//sub-string ends in that
       position
    int p; // total node(number of unique palindrome in string)
    int newnode(int l)
    {for(int i=0; i<N; i++)next[p][i]=0;cnt[p]=0;
    num[p]=0;len[p]=l;ans= max( ans, len[p] );return p++;}
    void init()
    {p=0;newnode(0);newnode(-1);last=0;n=0;S[n]=-1;fail[0]=1;ans= 1;}
    int get_fail(int x) // KMP
    {while(S[n-len[x]-1]!=S[n])x=fail[x];return x;}
    void add(int c,int i)
    {c-='a';S[++n]=c;int cur = get_fail(last);
    if(!next[cur][c])
    {int now = newnode(len[cur]+2);end[now]=i;st[now]=i-len[cur]-2+1;
    fail[now] = next[get_fail(fail[cur])][c];next[cur][c]=now;
    num[now]=num[fail[now]]+1;}last=next[cur][c];cnt[last]++;}
    void count()
    {for(int i=p-1; i>=0; --i)cnt[fail[i]]+=cnt[i];}
}tt;
int main(){tt.init();for(int i=0; i<totlen; i++)tt.add(str[i],i);}
```

12 Persistent Segment Tree

```
struct node{int l,r,val;node(){l=r=val=0;}
node(int _l,int _r,int _val){l=_l,r=_r,val=_val;}
}t[20*N]; ///size will be nlogn
int root[N],cnt,c,a[N];
map<ll,ll>ulta,mp;vector<int>adj[N];
int par[N][22], dep[N];
void build(int cur,int b,int e)
{if(b==e){t[cur]=node(0,0,0);return;}
int left,right,mid=(b+e)/2;
t[cur].l=left=++cnt;t[cur].r=right=++cnt;
build(left,b,mid);build(right,mid+1,e);
t[cur].val=t[left].val+t[right].val;}
void upd(int pre,int cur,int b,int e,int i,int v)
{if(i<b||i>e)return;
if(b==e){t[cur].val= t[pre].val+v;return;}
int left,right,mid=(b+e)/2;
if(i<=mid){t[cur].r=right=t[pre].r;t[cur].l=left=++cnt;
upd(t[pre].l,t[cur].l,b,mid,i,v);}
else{t[cur].l=left=t[pre].l;t[cur].r=right=++cnt;
upd(t[pre].r,t[cur].r,mid+1,e,i,v);}
t[cur].val=t[left].val+t[right].val;}
int query(int u, int v, int upor, int cent, int b, int e, int k)
{if(b==e)return b;int mid=(b+e)/2;
int cnt=t[t[u].l].val+t[t[v].l].val-t[t[cent].l].val-t[t[upor].l].val;
if(cnt>=k)return query(t[u].l,t[v].l,t[upor].l,t[cent].l,b,mid,k);
return query(t[u].r,t[v].r,t[upor].r,t[cent].r,mid+1,e,k-cnt);}
void dfs( int u, int pre )
{dep[u]=dep[pre]+1;root[u]=++cnt;
upd(root[pre],root[u],1,c,mp[a[u]],1);
for(auto v:adj[u]){if(v==pre)continue;
par[v][0]= u;dfs(v,u);}}
```

13 ahocorasic

```
char text[2000000],str[505][505];
vector<int>vc[400000];
int nwnode[400000][27],backnode[400000];
int cnt[400000],vis[400000],id = 0;
int newnode(){
  id++;
  for (int i=1;i<=26;++i)nwnode[id][i]=0;
  vis[id]=0;cnt[id]=0;
  vc[id].clear();return id;}
void build(int n){
  int root=newnode(),p;
```

```
  queue<int>q;
  for(int i=1;i<=n;++i){
    p=root;
    for(int j=0;str[i][j];j++){
      int c=str[i][j]-96;
      if(!nwnode[p][c])nwnode[p][c]=newnode();
      p=nwnode[p][c];}}
  for(int i=1;i<=26;++i){
    if(!nwnode[root][i])nwnode[root][i]=root;
    else{
      q.push(nwnode[root][i]);
      backnode[nwnode[root][i]] = 1;}}
  int u,v,w;
  while(!q.empty()){
    u=q.front();
    q.pop();
    for(int i=1;i<=26;++i){
      if(!nwnode[u][i])continue;
      w=backnode[u];v=nwnode[u][i];
      while(nwnode[w][i]==0){w=backnode[w];}
      int c=nwnode[w][i];
      backnode[v]=w=c;
      vc[w].push_back(v);
      q.push(v); }}}
void ahocorasic(){
  int p=1,c;
  for(int i=0;text[i];++i) {
    c=text[i]-96;
    while(!nwnode[p][c])p=backnode[p];
    p=nwnode[p][c];
    cnt[p]++;}}
int dfs(int p){
  if (vis[p]==1)return cnt[p];
  for (int i = 0;i<(int)vc[p].size();++i) {
    int w=vc[p][i];
    cnt[p]+=dfs(w);}
  vis[p]=1;return cnt[p];}
main(){
  id=0;cin>>text;
  for (int i=1;i<=n;++i)cin>>str[i];
  build(n);ahocorasic();
  for (int i=1;i<=n;++i) {
    int p=1;
    for (int j=0;str[i][j];++j) {
      int c=str[i][j]-96;
      p=nwnode[p][c];}
    printf("%d\n", dfs(p));}}
```

14 bellmanford

```
bool bellmanford(){
       memset(vis,0,sizeof vis);
       for (int i = 1; i <= n; ++i) dis[i] = -INF;
       queue<int>q;q.push(1);dis[1]=1.;
       while(!q.empty()) {
              int u=q.front();q.pop();vis[u] = 0;
              for (auto p:adj[u]) {
                     int v=p.ff;f64 w=p.ss;
                     if (dis[v]<dis[u]*w) {
                            dis[v]=dis[u]*w;
                            if (!vis[v]){vis[v]=1;q.push(v);}
                            if (v==1)return true;}}}///negative cycle
       return false;}
```

15 centroid

```
void dfs0(int from,int u,int dep){
  T[u]=from;L[u]=dep;
  for(auto v:adj[u]) {
    if(v==from)continue;
    dfs0(u,v,dep+1); } }
void init(){//declare int n, m, tot globally
```

```
  for(int i=1;i<=n;i++)
    for(int j=0;1<<j<n;j++)P[i][j]=-1;
  for(int i=1;i<=n;i++)P[i][0]=T[i];
  for(int j=1;1<<j<n;j++)
    for(int i=1;i<=n;i++)
      if(P[i][j-1]!=-1)P[i][j]=P[P[i][j-1]][j-1];}
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 decompose(int root,int p){
  tot=0;dfs1(root,root);
  int centroid=dfs2(root,root);
  if(p==-1)p=centroid;
  par[centroid]=p;dead[centroid]=1;
  for(auto v:adj[centroid]){
    if(dead[v])continue;
    decompose(v,centroid);}
  adj[centroid].clear();}
int lca(int p,int q){
  int tmp,log,i;
  if(L[p]<L[q])tmp=p,p=q,q=tmp;
  for(log=1;1<<log<=L[p];log++);
  log--;
  for(i=log;i>=0;i--)
    if(L[p]-(1<<i)>=L[q])p=P[p][i];
  if(p==q)return p;
  for(i=log;i>=0;i--)
    if(P[p][i]!=-1&&P[p][i]!=P[q][i]){p=P[p][i],q=P[q][i];}
  return T[p];}
int dist(int u,int v){
  return L[u]+L[v]-2*L[lca(u, v)];}
void update(int u){
  int x=u;
  while(1) {
    ans[x]=min(ans[x],dist(x, u));
    if(x==par[x])break;
    x=par[x]; } }
int query(int u){
  int x=u;
  int ret=100000;
  while(1) {
    ret = min(ret,ans[x]+dist(x,u));
    if(x==par[x])break;
    x=par[x]; }
  return ret; }
```

16 centroid2

```
void dfs3(int u,int p,int dep,int dis,int add) {
  cnt[dis][dep]+=add;
  for(auto v:adj[u]) {
    if(v==p||dead[v])continue;
    dfs3(v,u,dep,dis+1,add);}}
ll dfs4(int u,int p,int dep,int dis) {
  ll ret=0;
  for(int i=0; i<(int)primes.size();++i) {
    if(primes[i]-dis<0)continue;
    if(!cnt[primes[i]-dis][dep])break;
    if(primes[i]!=dis)ret+=cnt[primes[i]-dis][dep];
    else ret+=2ll;}
  for(auto v:adj[u]) {
    if(v==p||dead[v]) continue;
    ret+=dfs4(v,u,dep,dis+1);}
  return ret;}
void decompose(int root,int dep) {
  tot=0; dfs1(root,root);
  int centroid=dfs2(root,root);
  dfs3(centroid,centroid,dep,0,1);
```

```
  ll c=0; int u=centroid;
  for(auto v:adj[u]) {
    if(dead[v]) continue;
    dfs3(v,u,dep,1,-1);
    c+=dfs4(v,u,dep,1);
    dfs3(v,u,dep,1,1); }
  ans+=(c>>1); dead[u]=1;
  for(auto v:adj[u]) {
    if(dead[v])continue;
    decompose(v,dep+1); }
  for(int i=0;i<maxn&&cnt[i][dep];++i)cnt[i][dep]=0; }
```

17 chinese graph

```
const int N=1010,E=1000010;
int F[N],T[E*2],P[E*2],C=0,V[N];
void add(int u,int v){P[C]=F[u],F[u]=C,T[C++]=v;}
// P[C] = F[u], F[u] = C, T[C][1]=w ,T[C++][0] = v;
void dfs(int u)//init before graph
{V[u]=1;for(int i=F[u];~i;i=P[i]){int v=T[i];if(!V[v]){dfs(v);}}}
void init(){C=0,memset(F,-1,sizeof F);memset(V,0,sizeof V);}
```

18 crt

```
/**A CRT solver which works even when moduli are not pairwise coprime
    1. Add equations using addEquation() method
    2. Call solve() to get {x, N} pair, where x is the unique
       solution modulo N.
    Assumptions:1. LCM of all mods will fit into long long.*/
class ChineseRemainderTheorem {
    typedef pair<ll,ll> pll;
    vector<pll> equations;
public:/** CRT Equations stored as pairs of vector. See addEqation()*/
    void clear() {equations.clear();}
    /** Add equation of the form x = r (mod m)*/
    void addEquation(ll r,ll m){ equations.push_back({r, m});}
    pll solve() {
        if (equations.size() == 0) return {-1,-1}; /// No equations to
            solve
        ll a1 = equations[0].first;
        ll m1 = equations[0].second;
        a1 %= m1;
        /** Initially x = a_0 (mod m_0)*/
        /** Merge the solution with remaining equations */
        for ( int i = 1; i < equations.size(); i++ ) {
            ll a2 = equations[i].first;
            ll m2 = equations[i].second;
            ll g = __gcd(m1, m2);
            if ( a1 % g != a2 % g ) return {-1,-1}; /// Conflict in
                equations
            /** Merge the two equations*/
            ll p, q;
            ext_gcd(m1/g, m2/g, &p, &q);
            ll mod = m1 / g * m2;
            ll
                x=((lll)a1*(m2/g)%mod*q%mod+(lll)a2*(m1/g)%mod*p%mod)%mod;
            /** Merged equation*/
            a1 = x;
            if ( a1 < 0 ) a1 += mod;
            m1 = mod;
        }
        return {a1, m1};}};
```

19 dc3

```
#define N 200005
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[N],wb[N],wv[N],ww[N],R[N],L[N],sa[3*N],r[3*N];
#define Fo(n) for(i=0;i<n;i++)
```

```
int c0(int *y,int a,int b){
return y[a]==y[b]&&y[a+1]==y[b+1]&&y[a+2]==y[b+2];}
int c12(int k,int *y,int a,int b){if(k^2)
    return y[a]<y[b]||y[a]==y[b]&&wv[a+1]<wv[b+1];
return y[a]<y[b]||y[a]==y[b]&&c12(1,y,a+1,b+1);}
void sort(int *r,int *a,int *b,int n,int m){int i;
Fo(n)wv[i]=r[a[i]];Fo(m)ww[i]=0;Fo(n)ww[wv[i]]++;
Fo(m-1)ww[i+1]+=ww[i];for(i=n-1;i>=0;i--)b[--ww[wv[i]]]=a[i];}
void dc3(int *r,int *sa,int n,int m)
{int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;Fo(n)if(i%3!=0)wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if(p<tbc)dc3(rn,san,tbc,p);else Fo(tbc) san[rn[i]]=i;
    Fo(tbc) if(san[i]<tb) wb[ta++]=san[i]*3;
    if(n%3==1) wb[ta++]=n-1;sort(r,wb,wa,ta,m);
    Fo(tbc) wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0;i<ta && j<tbc;p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
for(;i<ta;p++)sa[p]=wa[i++];for(;j<tbc;p++)sa[p]=wb[j++];}
void get_L(int n){int i,j,k=0;
    Fo(n+1)R[sa[i]]=i;for(i=0;i<n;L[R[i++]]=k)
    for(k?k--:0,j=sa[R[i]-1];r[i+k]==r[j+k];k++);}
void SA(char *a,int l){int i;Fo(l+1)r[i]=a[i];
    dc3(r,sa,l+1,256);get_L(l);}
```

20 discrete log

```
struct DiscreteLogarithm
{
        int pow(){/*...*/}
        int PrimitiveRoot(int p){/*in ntt*/}
        int DiscreteLog(int a, int b, int m)
        {///find any integer x such that a^x = b (mod m)
                int n = (int) sqrt (m + .0) + 1;
                int an = 1;
                for (int i = 0; i < n; ++i) an = (1LL * an * a) % m;
                unordered_map<int, int> vals;
                for (int p = 1, cur = an; p <= n; ++p) {
                        if (!vals.count(cur)) vals[cur] = p;
                        cur = (1LL * cur * an) % m;
                }
                for (int q = 0, cur = b; q <= n; ++q) {
                        if (vals.count(cur)) {
                                int ans = vals[cur] * n - q;
                                if (pow(a, ans, m) == b % m) return ans;}
                                cur = (1LL * cur * a) % m;
                }
                return -1;
        }
///returns any or all numbers x such that x^k = a (mod n)
        int DiscreteRoot(int k, int a, int n) {
                if (a == 0) return 1;
                int g = PrimitiveRoot(n);
                int phi = n - 1;
                int sq = (int) sqrt (n + .0) + 1;
                vector < pair<int, int> > dec (sq);
                for (int i = 1; i <= sq; ++i) dec[i - 1] = make_pair
                    (pow (g, 1LL * i *sq % phi * k % phi, n), i);
                sort (dec.begin(), dec.end());
                int any_ans = -1;
                for (int i = 0; i < sq; ++i) {
                        int my = pow (g, 1LL * i * k % phi, n) * 1LL *
                            a % n;
                        auto it = lower_bound (dec.begin(), dec.end(),
                            make_pair (my,0));
                        if (it != dec.end() && it->first == my) {
                                any_ans = it->second * sq - i;
                                break;
                        }
                }
                if (any_ans == -1) return -1;
```

```
            int delta = (n - 1) / __gcd (k, n - 1);
            for (int cur = any_ans % delta; cur < n - 1; cur +=
                delta) return (pow (g, cur, n));
///for all possible answers
///int delta = (n-1) / __gcd(k, n-1);
///vector<int> ans;
///for (int cur = any_ans % delta; cur < n-1; cur += delta)
// ans.push_back(pow(g, cur, n));
///sort(ans.begin(), ans.end());
///return ans;
        }
} d;
```

21 dsuontree

```
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=sz[v];bigchild=v;}}
  for (auto v:adj[u]) {
    if (v!=p&&v!=bigchild)dsu(v,u,0);}
  if (bigchild!=-1) {
    dsu(bigchild,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)
```

22 dynamix convex hull

```
#define Lop(t) {while(k>t&&Sg(R[k-2],R[k-1],p[i])^2)k--;R[k++]=p[i];}
#define SV(x) if(x==1)x=-1;
struct P{ll x,y;}base;
vector<P> R(200007), p;
int Sg(P p,P q,P r){ll v=(q.y-p.y)*(r.x-q.x)-(q.x-p.x)*(r.y-q.y);
    return v?v>0?1:2:0;}
ll D(P a,P b){return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);}
bool cmp(P a,P b){return a.x^b.x?a.x<b.x:a.y<b.y;}
void bord(){int k=0,n=p.size();sort(p.begin(),p.end(),cmp);
    for(int i=0;i<n;i++)Lop(1)int t=k;
    for(int i=n-2;i>=0;i--)Lop(t)R.resize(k-1);}
bool inside(P a){int n=R.size();int L=1,H=n-1;
while(H-L>1){int m=(L+H)/2;if(Sg(R[0],R[m],a)^1)L=m;else H=m;}
int v1=Sg(R[0],R[L],a),v2=Sg(R[L],R[H],a),v3=Sg(R[H],R[0],a);
SV(v1)SV(v2)SV(v3)return !(v1*v2<0||v2*v3<0||v3*v1<0);}
void add(P a,vector<P> H){int I=0,n=H.size();ll pt=1e18;
    for(int i=0;i<n;i++){ll d=D(a,H[i]);if(d<pt)pt=d,I=i;}
    int u=I;bool G=0;while(Sg(a,H[u],H[(u+1)%n])^2)u=(u+1)%n,G=1;
    if(!G){while(Sg(a,H[u],H[(n+u-1)%n])^2)u=(n+u-1)%n;}
    G=0;int L=I;while(Sg(a,H[L],H[(n+L-1)%n])^1)L=(n+L-1)%n,G=1;
    if(!G){while(Sg(a,H[L],H[(L+1)%n])^1)L=(L+1)%n,G=1;}
    int Cr=u;if(u==L)Cr=0,L=n-1,u=0;R.resize(0);R.pb(H[u]);
    while(Cr^L){Cr=(Cr+1)%n;R.pb(H[Cr]);}R.pb(a);}
```

23 ex euclid

```
int ext_gcd(int A,int B,int &X,int &Y){
int x2=1,y2=0,x1=0,y1=1,x,y,r2,r1,q,r;
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;}
int Ext_gcd(int a,int b,int & x,int & y){if(a==0){x=0,y=1;
return b;}int x1,y1,d=Ext_gcd(b%a,a,x1,y1);
x=y1-(b/a)*x1;y=x1;return d;}
```

24 fasterIO

```
Te inline int scan(T *n) {
        bool N=0;*n=0;register int c=gu();
        while(!isdigit(c)&&~c&&c^45)c=gu();
        if(!~c)return c;if(!(c^45)){N=1;c=gu();}
        for(;isdigit(c);c=gu())*n=(*n)*10+c-48;
        if(N)*n=-1*(*n);return 1;}
Te inline void Rec(T n){
        if(!n){return;}
        if(n<0){pu('-');n*=-1;}Rec(n/10);
        pu(n%10+0x30);}
Te inline void print(T n){if(!n)pu('0');
else Rec(n);return;}
```

25 fft

```
using cd = complex<double>;
const double PI = acos(-1);

void fft(vector<cd> & a, bool invert) {
        int n = a.size();

        for (int i = 1, j = 0; i < n; i++) {
                int bit = n >> 1;
                for (; j & bit; bit >>= 1)
                        j ^= bit;
                j ^= bit;

                if (i < j)
                        swap(a[i], a[j]);
        }

        for (int len = 2; len <= n; len <<= 1) {
                double ang = 2 * PI / len * (invert ? -1 : 1);
                cd wlen(cos(ang), sin(ang));
                for (int i = 0; i < n; i += len) {
                        cd w(1);
                        for (int j = 0; j < len / 2; j++) {
                                cd u = a[i + j], v = a[i + j + len / 2]
                                    * w;
                                a[i + j] = u + v;
                                a[i + j + len / 2] = u - v;
                                w *= wlen;
                        }
                }
        }

        if (invert) {
                for (cd & x : a)
                        x /= n;
        }
}
vector<int> multiply(vector<int> const& a, vector<int> const& b) {
    vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.end());
    int n = 1;
    while (n < a.size() + b.size())
        n <<= 1;
    fa.resize(n);
    fb.resize(n);
```

```
    fft(fa, false);
    fft(fb, false);
    for (int i = 0; i < n; i++)
        fa[i] *= fb[i];
    fft(fa, true);

    vector<int> result(n);
    for (int i = 0; i < n; i++)
        result[i] = round(fa[i].real());
    return result;
}
```

26 gaussian

```
/*1. Set row and col of mat
2. Call rank() to perform gauss-elimination and find rank
3. Call isValid() to find if solution exists.
Careful about int a[x][x]. If mod^2 crosses int, take ll
If mod is 2, it is better to use XOR since it a lot faster.*/
struct GAUSS{
  int row, col;
  ll a[x][x];
  int mod;
       bool valid;
       GAUSS(){mod=xyz;}
  void clear(){memset(a,0,sizeof a);}
       void isValid(int st){
              int i;valid=true;
              for (i=st;i<row;i++){
          if (a[i][col-1]){valid=false;return;}}}
  ///Return Rank of Matrix
  ///Free variable = Variable - Rank or Col - Rank - 1
  int rank(){
    int i=0,j=0,k,r,u;
       while(i<row&&j<col-1){
              r=i;
              for(k=i;k<row;k++)
       if(a[k][j]){
         r=k;break;}///Find non-zero coefficient
              if(a[r][j]){
                     if(r!=i) ///Swap row if required
                            for(k=0;k<col;k++)
                                   swap(a[r][k],a[i][k]);
///Neutralize if required. Depends on whether double or modular
    division
         ll v=a[i][j];
         v=modInv(v,mod);
         for(u=j;u<col;u++){
           a[i][u]=(a[i][u]*v)%mod;}
           /*double v = a[i][j];
           for ( u = j; u < col; u++ ) {
             a[i][u] /= v;
           }*/
                     for(u=i+1;u<row;u++)
                            if(a[u][j]){ ///Eliminate
           int v = a[u][j];
                            for(k=j;k<col;k++) {
                                   a[u][k]=((a[i][k]*v)-a[u][k])%mod;
           if (a[u][k]<0)a[u][k] += mod;
                                   }}i++;}j++;}
       return i;}
  void print() {
    FOR(i,0,row-1){
      FOR(j,0,col-1){
        printf("%d ",a[i][j]);
      }nl;}}
}mat;
```

27 hackenbush

```
// Consider a two player game on a graph with a specified
    vertex(root).
```

```
// In each turn, a player eliminates one edge.
// Then, if a subgraph that is disconnected from the root, it
    isremoved.
// If a player cannot select an edge (i.e., the graph is singleton),
// he will lose.
// Compute the Grundy number of the given graph.
// Algorithm:
// We use two principles:
// 1. Colon Principle: Grundy number of a tree is the xor of
// Grundy number of child subtrees.
// 2. Fusion Principle: Consider a pair of adjacent vertices u, v
// that has another path (i.e., they are in a cycle). Then,
// we can contract u and v without changing Grundy number.
// We first decompose graph into two-edge connected components.
// Then, by contracting each components by using Fusion Principle,
// we obtain a tree (and many self loops) that has the same Grundy
// number to the original graph. By using Colon Principle, we can
// compute the Grundy number.
// Complexity: O(m + n)
struct hackenbush{
int n;
vector<vector<int>>adj;
hackenbush(int n):n(n),adj(n) { }
void add_edge(int u,int v) {
adj[u].push_back(v);
if(u != v) adj[v].push_back(u);}
// r is the only root connecting to the ground
int grundy(int r){
vector<int> num(n),low(n);int t=0;
function<int(int,int)>dfs=[&](int p,int u) {
num[u]=low[u]=++t;int ans=0;
for(int v:adj[u]) {
if(v==p) {p+=2*n;continue;}
if(num[v]==0) {int res=dfs(u,v);
low[u]=min(low[u],low[v]);
if(low[v]>num[u]) ans^=(1+res)^1; // bridge
else ans^=res;}// non bridge
else low[u]=min(low[u],num[v]);}
if(p>n)p-=2*n;
for (int v:adj[u])
if(v!=p&&num[u]<=num[v])ans^=1;
return ans;};
return dfs(-1, r);}};
main(){
int cases;scanf("%d",&cases);
for (int icase = 0;icase<cases;++icase) {
int n;scanf("%d", &n);
vector<int> ground(n);int r;
for (int i=0;i<n;++i) {
scanf("%d",&ground[i]);
if (ground[i] =1)r=i;}
int ans=0;
hackenbush g(n);
for (int i=0;i<n-1;++i) {
int u,v;scanf("%d%d",&u,&v);
--u;--v;
if(ground[u])u=r;if(ground[v])v=r;
if(u == v) ans^=1;
else g.add_edge(u,v);}
int res = ans ^ g.grundy(r);
printf("%d\n", res != 0);}}
```

28 hld

```
//define root 0
vector<int> adj[N],costs[N],indexx[N];
int baseArray[N], ptr,tree[N<<2];
int chainNo,chainInd[N],chainHead[N],posInBase[N];
int depth[N],par[lN][N],otherEnd[N],subsize[N]; ;
//insert segment tree
int query_up(int u,int v) {
  if(u==v) return 0; //update this for nodes
  int uchain,vchain=chainInd[v],ans=-1;
  while(1){///segq=query_tree
```

```
    uchain=chainInd[u];
    if(uchain==chain) {
      if(u==v)break;//update here for nodes
      int qt=segq(1,0,ptr,posInBase[v]+1,posInBase[u]);
      if(qt>ans)ans=qt;
      break;}int qt=
segq(1,0,ptr,posInBase[chainHead[uchain]],posInBase[u]);
    if(qt>ans)ans=qt;
    u=chainHead[uchain];
    u=par[0][u];}
  return ans;}
int LCA(int u,int v) {
  if(depth[u]<depth[v])swap(u,v);
  int diff=depth[u]-depth[v];
  for(int i=0;i<lmaxn;i++)if((diff>>i)&1)u=par[i][u];
  if(u==v)return u;
  for(int i=lmaxn-1;i>=0;i--)if(par[i][u]!=par[i][v]) {
    u=par[i][u];v=par[i][v];}
  return par[0][u];}
void query(int u,int v) {
  int lca = LCA(u, v);
  int ans = query_up(u, lca);int temp = query_up(v, lca);
  if(temp>ans) ans=temp;printf("%d\n",ans);}
void change(int i,int val) {
  int u=otherEnd[i];
  update_tree(1, 0, ptr, posInBase[u], val);
}
void HLD(int curNode, int cost, int prev) {
  if(chainHead[chainNo]==-1){chainHead[chainNo]=curNode;}
  chainInd[curNode]=chainNo;
  posInBase[curNode]=ptr;baseArray[ptr++]=cost;
  int sc=-1,ncost;
  for(int i=0;i<adj[curNode].size();i++)if(adj[curNode][i]!=prev) {
    if(sc==-1||subsize[sc]<subsize[adj[curNode][i]]) {
      sc=adj[curNode][i];ncost=costs[curNode][i];}}
  if(sc!=-1) HLD(sc,ncost,curNode);
  for(int i=0;i<adj[curNode].size();i++)if(adj[curNode][i]!=prev) {
    if(sc!=adj[curNode][i]) {
      chainNo++;
      HLD(adj[curNode][i], costs[curNode][i], curNode);}}}

void dfs(int cur,int prev,int _depth=0) {
  par[0][cur] = prev;depth[cur]=_depth;subsize[cur] = 1;
  for(int i=0;i<adj[cur].size();i++)
    if(adj[cur][i]!=prev) {
      otherEnd[indexx[cur][i]]=adj[cur][i];
      dfs(adj[cur][i],cur,_depth+1);
      subsize[cur]+=subsize[adj[cur][i]];}}
main(){
ptr = 0;
for(int i=0; i<n; i++) {
  chainHead[i] = -1; //clean in case of test cases
  for(int j=0; j<lmaxn; j++) par[j][i] = -1;}
for(int i=1; i<n; i++) {
  costs[u].push_back(c);
  indexx[u].push_back(i-1);
  costs[v].push_back(c);
  indexx[v].push_back(i-1);
}
chainNo = 0;dfs(root, -1);HLD(root, -1, -1);build_tree(1, 0, ptr);
for(int i=1; i<lmaxn; i++)
  for(int j=0; j<n; j++)
    if(par[i-1][j] != -1)par[i][j] = par[i-1][par[i-1][j]];
}
```

29 josephus

```
Te T jos(T n,T k,T m){m=n-m;if(k<=1)return n-m;T i=m;
while(i<n){T r=(i-m+k-2)/(k-1);if(i+r>n)r=n-i;else if(!r)r=1;
i+=r;m=(m+r*k)%i;}return m+1;}//for k=2;2,4...will die
```

30 kd tree

```
const int MAXN = 300000+5;
const int MAXD = 2;
inline int64_t sq(int x) { return x * 1ll * x; }
struct point
{
	int c[MAXD];
	point() { }
};
struct cmp
{
	int current_d;
	cmp() { current_d = 0; }
	cmp(int d) { current_d = d; }
	bool operator() (const point& a, const point& b) { return
	    a.c[current_d] < b.c[current_d]; }
};

int64_t sq_dist(point a, point b, int d)
{
	int64_t answer = 0;
	for(int i = 0; i < d; i++)
		answer += sq(a.c[i] - b.c[i]);

	return answer;
}

struct kd_tree
{
	struct node
	{
		point p;
		int L, R, axis;
		node() { L = -1; R = -1; }
		node(point _p) { L = -1; R = -1; p = _p; }
	};

	int psz = 0, D, root;
	node tr[MAXN << 2];

	kd_tree() { D = 0; psz = 0; }
	kd_tree(int d) { D = d; psz = 0; }

	int new_node() { return psz++; }

	int build(point *from, point *to, int axis)
	{
		if(to - from == 0)
			return -1;

		point *mid = from + (to - from) / 2;

		nth_element(from, mid, to, cmp(axis));

		int c_node = new_node();
		tr[c_node] = node(*mid);

		tr[c_node].axis = axis;

		tr[c_node].L = build(from, mid, (axis + 1) % D);
		tr[c_node].R = build(mid + 1, to, (axis + 1) % D);

		return c_node;
	}

	void init(point *from, point *to, int d)
	{
		D = d;
		random_shuffle(from, to);
		root = build(from, to, 0);
	}
	long long tol;
	long long cntt;
	void query(int idx, point q, int64_t answer)
	{
```

```
        if(cntt>tol)return;
        if (idx == -1) return;
        long long ttt=sq_dist(q, tr[idx].p, D);
        if(ttt<=answer)cntt++;
        if(tr[idx].p.c[tr[idx].axis] <= q.c[tr[idx].axis])
        {
                query(tr[idx].R, q, answer);
                if(tr[idx].L != -1 && q.c[tr[idx].axis] -
                    sqrt(answer) <= tr[idx].p.c[tr[idx].axis])
                        query(tr[idx].L, q, answer);
        }
        else
        {
                query(tr[idx].L, q, answer);
                if(tr[idx].R != -1 && q.c[tr[idx].axis] +
                    sqrt(answer) >= tr[idx].p.c[tr[idx].axis])
                        query(tr[idx].R, q, answer);
        }
}
long long inCircle(point q,long long rad)
{
        cntt=0;
        query(root, q, rad*rad);
        return cntt;
}
void U(int idx, point q,int axis)
{
        // if(idx == -1)
        // {
        //      int c_node = new_node();
        //      tr[c_node] = node(q);

        //      tr[c_node].axis = axis;
        // }
        if(tr[idx].p.c[tr[idx].axis] <= q.c[tr[idx].axis])
        {
                if(tr[idx].R==-1)
                {
                        int c_node = new_node();
                        tr[c_node] = node(q);

                        tr[c_node].axis = axis;
                        tr[idx].R=c_node;
                }
                else
                U(tr[idx].R, q,(axis+1)%D);
        }
        else
        {
                if(tr[idx].L==-1)
                {
                        int c_node = new_node();
                        tr[c_node] = node(q);

                        tr[c_node].axis = axis;
                        tr[idx].L=c_node;
                }
                else
                U(tr[idx].L, q,(axis+1)%D);
        }
    }
    void update(point q)
    {
            U(root,q,0);
    }
};

int n, d=2;
point li[MAXN];
kd_tree t;
int main()
{int n,q ;
        cin >> n;
        cin>>q;
        for(int x = 0; x < d; x++)
                for(int i = 0; i < n; i++)
```

```
                cin >> li[i].c[x];

        t.init(li, li + n, d);
        for (int i = 0; i < q; i++)
        {
                point pp;
                for(int x=0;x<d;x++)cin>>pp.c[x];
                long long rad;
                cin>>rad>>t.tol;
                long long ans=t.inCircle(pp,rad);
                if(ans>t.tol)cout<<"Too Many!!!\n";
                else
                        cout<<ans<<endl;
                if(ans<=t.tol)t.update(pp);
        }
}
```

31 kuhn

```
vector<int>adj[N]; int vis[N],match[N],iter;
int dfs(int u){
  if(vis[u]==iter)return 0;
  vis[u] = iter;// Left nodes - 0, 1, ..., n - 1. Right rest.
  for(int v : adj[u]) {
    if(match[v] < 0 || dfs(match[v])) {
      match[u] = v, match[v] = u; return 1;}
  }return 0;}
int kuhn(){
  memset(match, -1, sizeof match);
  int ans = 0;
  for(int i = 0; i < n; i++) ++iter; ans += dfs(i);
  return ans; }
```

32 linear recurrance

```
ll linearRec(vll S, vll tr, ll k) {ll n=S.size();
if(!n)return 0;auto C=[&](vll a,vll b){
vll R(n*2+1);for(ll i=0;i<n+1;i++)for(ll j=0;j<n+1;j++)
R[i+j]=(R[i+j]+a[i]*b[j])%mod;for(ll i=2*n;i>n;--i)
for(ll j=0;j<n;j++)R[i-1-j]=(R[i-1-j]+R[i]*tr[j])%mod;
R.resize(n+1);return R;};vll p(n+1),e(p);p[0]=e[1]=1;
for(++k;k;k/=2){if(k&1)p=C(p,e);e=C(e,e);}
ll R=0;for(ll i=0;i<n;i++)R=(R+p[i+1]*S[i])%mod;return R;}
```

33 linear seive

```
const int M = 1000010;//size,P=prime
vector<int>Pr;bool P[M];int F[M],C[M];
#define D(p,x) ((Pow(p,x)/p)*(p-1))
void sieve() {//F=function,C=cnt
    F[1]=1;//D(p,x)=f(p^x)
    for(int i=2;i<M;++i){if(!P[i]){Pr.pb(i);F[i]=D(i,1);C[i]=1;}
    for(int j=0;j<Pr.size()&&i*Pr[j]<M;++j){P[i*Pr[j]]=1;
    if(i%Pr[j]^0){F[i*Pr[j]]=F[i]*F[Pr[j]];C[i*Pr[j]]=1;}
    else{F[i*Pr[j]]=F[i]*(D(Pr[j],C[i]+1)/D(Pr[j],C[i]));
    C[i*Pr[j]]=C[i]+1;break;}}}}
```

34 math

```
#define D(m,n) (n%m==0)
ll intersectionOfDiagonal(ll n)
{return ((n-3)*(n-2)*(n-1)*n/24)
       +((-5*n*n*n+45*n*n-70*n+24)/24)*D(2,n)
       -((3*n)/2)*D(4,n)+((-45*n*n+262*n)/6)*D(6,n)
       +42*n*D(12,n)+60*n*D(18,n)+35*n*D(24,n)
       -38*n*D(30,n)-82*n*D(42,n)-330*n*D(60,n)-144*n*D(84,n)
```

```
      -96*n*D(90,n)-144*n*D(120,n)-96*n*D(210,n);}
double picksArea(ll side_e,ll vitore){vitore+(side_e/2.0)+1.0}
// Let D(n) be the last non-zero digit in n!
// If tens digit (or second last digit) of n is odd
//    D(n) = 4 * D(floor(n/5)) * D(Unit digit of n)
// If tens digit (or second last digit) of n is even
//    D(n) = 6 * D(floor(n/5)) * D(Unit digit of n)
string min_cyclic_string(string s) {
    s += s;
    int n = s.size();
    int i = 0, ans = 0;
    while (i < n / 2) {
        ans = i;
        int j = i + 1, k = i;
        while (j < n && s[k] <= s[j]) {
            if (s[k] < s[j])
                k = i;
            else
                k++;
            j++;
        }
        while (i <= k)
            i += j - k;
    }
    return s.substr(ans, n / 2);
}
```

35 maxflow

```
///make idd = 0 at the start of test case
struct Node {vector<int> adj;}graf[N];
struct Edge {int u,v,cap,flow;};
vector<Edge> E;
int v,e,s,t,dist[N],upTo[N],idd = 0;
inline bool BFS() {
  for (int i=1;i<=v;i++)dist[i]=-1;
  queue<int>bfs_queue;
  bfs_queue.push(s);dist[s] = 0;
  while (!bfs_queue.empty()) {
    int xt=bfs_queue.front();
    bfs_queue.pop();///cId=currID
    for (int i=0;i<graf[xt].adj.size();i++){
      int cId=graf[xt].adj[i]; int xt1=E[cId].v;
      if (dist[xt1]==-1&&E[cId].flow<E[cId].cap){
        bfs_queue.push(xt1); dist[xt1]=dist[xt] + 1;}}}
  return (dist[t] != -1);}
inline int DFS(int xt,int mC) {
  if (!mC)return 0;///mC=minCap
  if (xt==t)return mC;
  while (upTo[xt]<graf[xt].adj.size()) {
    int cId=graf[xt].adj[upTo[xt]];
    int xt1=E[cId].v;
    if (dist[xt1]!=dist[xt]+1){
      upTo[xt]++;continue;}
    int aug=DFS(xt1,min(mC,E[cId].cap-E[cId].flow));
    if (aug>0){
      E[cId].flow+=aug;
      if (cId&1)cId--;else cId++;
      E[cId].flow-=aug;
      return aug;}
    upTo[xt]++;}
  return 0;}
inline int Dinic(){
  int flow=0;
  while (true) {
    if (!BFS()) break;
    for (int i=1;i<=v;i++)upTo[i]=0;
    while (true){
      int cF=DFS(s, INF);if (!cF)break;
      flow += cF;}}
  return flow;}
inline void addEdge(int u,int v,int cap) {
  Edge E1,E2;
  E1.u=u,E1.v=v,E1.cap=cap,E1.flow=0;
```

```
  E2.u=v,E2.v=u,E2.cap=0,E2.flow=0;
  graf[u].adj.push_back(idd++);
  E.push_back(E1);
  graf[v].adj.push_back(idd++);
  E.push_back(E2);}
```

36 multipoint

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
#define pb push_back
#define mp make_pair
#define ip pair<ll,ll>
#define ff first
#define ss second
#define MAXN 200005
#define mod 786433 //

ll root = 1000;
//ll root_1 = 235930;
ll root_pw = 1<<18;

void fft (vector<ll> &a, bool invert) {
	ll n = a.size();

	for (ll i=1, j=0; i<n; ++i) {
		ll bit = n >> 1;
		for (; j>=bit; bit>>=1)
			j -= bit;
		j += bit;
		if (i < j)
			swap (a[i], a[j]);
	}

	for (ll len=2; len<=n; len<<=1) {
		ll wlen = root;
		for (ll i=len; i<root_pw; i<<=1)
			wlen = ll (wlen * 1ll * wlen % mod);
		for (ll i=0; i<n; i+=len) {
			ll w = 1;
			for (ll j=0; j<len/2; ++j) {
				ll u = a[i+j], v = ll (a[i+j+len/2] *
				    1ll * w % mod);
				a[i+j] = u+v < mod ? u+v : u+v-mod;
				a[i+j+len/2] = u-v >= 0 ? u-v : u-v+mod;
				w = ll (w * 1ll * wlen % mod);
			}
		}
	}
}

vector<ll> ans(mod,0);
void multipointEval(vector<ll> &v)
{
	vector<ll> a(root_pw,0);
	vector<ll> b(root_pw,0);
	vector<ll> c(root_pw,0);
	ll p = 1;
	ll q = 1;
	for(int i=0;i<v.size();i++){
		a[i]=v[i];
		b[i] = (a[i]*p)%mod;
		c[i] = (a[i]*q)%mod;
		p = p*10%mod;
		q = q*100%mod;
	}

	ans[0] = a[0];
	fft(a,0);
	fft(b,0);
	fft(c,0);

	p = 1;
```

```
        for(int i=0;i<root_pw;i++){
                ans[p] = a[i];
                ans[p*10%mod] = b[i];
                ans[p*100%mod] = c[i];
                p = (p*root)%mod;
        }
}
int main(){
        ios_base::sync_with_stdio(0);cin.tie(0);

        ll n,p,q;
        cin>>n;
        vector<ll> v;
        for(int i=0;i<=n;i++){
                cin>>p;
                v.push_back(p);
        }
        multipointEval(v);
        cin>>q;
        while(q--){
                cin>>p;
                cout<<ans[p]<<endl;
        }
}
```

37 ntt

```
const ll mod = 998244353;
const ll root = 15311432;
const ll root_1 = 469870224;
const ll root_pw = 1 << 23;
void pre(){
        int k=0,c=mod-1;
        while(!(c&1))k++,c>>=1;
        vector<int> F;
    int Ph=mod-1,n=Ph;
    for(int i=2;i*i<=n;++i)
        if(n%i==0){F.pb(i);
        while(n%i==0)n/=i;}
    if(n^1)F.pb(n);int g;
    for(g=2;g<=mod;++g) {
bool o=1;for(size_t i=0;i<F.size()&&o;++i)
o &=pow(g,Ph/F[i],mod)!=1;if(o) break;}
   int rtt=pow(g,c,mod);
   cout<<"const ll root = "<<rtt<<";\n";
   int rti=pow(rtt,mod-2,mod);
   cout<<"const ll root_1 = "<<rti<<";\n";
        cout<<"const ll root_pw = 1 << "<<k<<";\n";}
void fft(vll & a, bool invert) {
        ll n = a.size();
        for (ll i = 1, j = 0; i < n; i++) {
                ll bit = n >> 1;
                for (; j & bit; bit >>= 1)
                        j ^= bit;j ^= bit;
                if (i < j)swap(a[i], a[j]);}
        for (ll len = 2; len <= n; len <<= 1) {
                ll wlen = invert ? root_1 : root;
                for (ll i = len; i < root_pw; i <<= 1)
                        wlen = (ll)(1LL * wlen * wlen % mod);
                for (ll i = 0; i < n; i += len) {
                        ll w = 1;
                        for (ll j=0;j<len/2;j++){
                                ll
                                    u=a[i+j],v=(ll)(1LL*a[i+j+len/2]*w%mod);
                                a[i+j]=u+v<mod?u+v:u+v-mod;
                                a[i+j+len/2]=u-v>=0?u-v:u-v+mod;
                                w = (ll)(1LL * w * wlen % mod);}}}
        if (invert) {
                ll n_1 = pow(n, mod - 2, mod);
                for (ll & x : a)
                        x = (ll)(1LL * x * n_1 % mod);}}
vll multiply(vll const& a, vll const& b) {
        vll fa(a.begin(), a.end()), fb(b.begin(), b.end());
        ll sz = a.size() + b.size() - 1;
```

```
	ll n = 1;
	while (n < a.size() + b.size())
		n <<= 1;
	fa.resize(n);
	fb.resize(n);

	fft(fa, false);
	fft(fb, false);
	for (ll i = 0; i < n; i++)
		fa[i] = (fa[i] * fb[i] + mod) % mod;
	fft(fa, true);

	vll result(sz);
	for (ll i = 0; i < sz; i++)
		result[i] = fa[i];
	return result;}
ll iv[(1 << 20) + 1];
void inverse_of_number()//calculate (1/i) modulo mod
{iv[1] = 1;for (int i = 2; i <= 1 << 20; ++i)
	iv[i] = mod - mod / i * iv[mod % i] % mod;}
vll translation(vll v, ll t) {
	vll a(v.size()), b(v.size()), r(v.size());
	ll f = 1, fi = 1, tp = 1;
	for (int i = 0; i < v.size(); ++i) {
		if (i) {f = f * i % mod;
			fi = fi * iv[i] % mod;
			tp = tp * t % mod;}
		a[v.size() - 1 - i] = f * v[i] % mod;
		b[i] = tp * fi % mod;}
	vll c = multiply(a, b); //multiply a,b
	fi = 1;
	for (int i = 0; i < v.size(); ++i) {
		if (i)fi = fi * iv[i] % mod;
		r[i] = c[v.size() - 1 - i] * fi % mod;}
	return r;}
vll rising_factorial(int r)
{vll v{1}, tt;
	int j = 0;
	while ((r >> j))j++; j--;
	for (int k = 0; j >= 0; j--)
	{tt = translation(v, mod - k);
		v = multiply(v, tt);k <<= 1;
		if (r & (1 << j)) {k |= 1;v.pb(0);
			for (int i = v.size() - 2; ~i; --i) {
				v[i + 1] += v[i];
				if (v[i + 1] >= mod)
					v[i + 1] -= mod;
				v[i]=v[i]*(mod-k)%mod;}}}return v;}
```

37.1 Lagrange polynomial interpolation

given set of pairs (x_i, y_i):

$$\boxed{A(x) = \sum_{i=1}^{n} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}}$$

$\sum_{k=0}^{n} \binom{n-k}{k} = F_{n+1}$
$\sum_{i=0}^{k} (-1)^i \binom{n}{i} = (-1)^k \binom{n-1}{k}$
$\sum_{i=0}^{k} \binom{n+i}{i} = \binom{n+k+1}{k}$
$\sum_{i=0}^{k} \binom{i}{n} = \binom{k+1}{n+1}$
$\sum_{i=0}^{n} i \times \binom{n}{i} = n \times 2^{n-1}$
$\sum_{i=0}^{n} i^2 \times \binom{n}{i} = (n + n^2) \times 2^{n-2}$
$\sum_{k=0}^{r} \binom{m}{k} \binom{n}{r-k} = \binom{m+n}{r}$
$\sum_{k=r}^{n} \binom{k}{r} = \binom{n+1}{r+1}$
$\sum_{i=0}^{k} \binom{k}{i}^2 = \binom{2k}{k}$
$\sum_{i=m}^{n} \binom{n}{i} \binom{i}{m} = 2^{n-m} \binom{n}{m}$
$\sum_{i=0}^{n} 3^i \binom{n}{i} = 4^i$
$D(n) = (n-1) \times (D(n-1) + D(n-2))$

38 prime counting

```
const int M=1000010,Mp=1000010,Pn=100000,Pk=100;
uint ar[(M>>6)+5]={0};//Mp=max size of the prime
int z=0,Pr[Mp],C[M],dp[Pn][Pk];
void Sieve(int N){/*calc sieve+C+Pr*/}
void init(){Sieve(M);int k,n,res;
for(n=0;n<Pn;n++)dp[n][0]=n;//C=number of prime<=i
for(k=1;k<Pk;k++)for(n=0;n<Pn;n++)
dp[n][k]=dp[n][k-1]-dp[n/Pr[k-1]][k-1];}
ll phi(ll n,int k){if(n<Pn&&k<Pk)return dp[n][k];
if(k==1)return ((++n)>>1);if(Pr[k-1]>=n) return 1;
return phi(n,k-1)-phi(n/Pr[k-1],k-1);}
ll Legendre(ll n){if(n<M)return C[n];
int L=sqrt(n)+1;int k=upper_bound(Pr,Pr+z,L)-Pr;
return phi(n,k)+(k-1);}//returns number of integers
ll Lh(ll n){if(n<M)return C[n];// less or equal n
ll w,R=0;// which are not divisible by any of
int i,j,a,b,c,L;//the first k prime
b=sqrt(n),c=Lh(cbrt(n)),a=Lh(sqrt(b)),b=Lh(b);
R=phi(n,a)+(((b+a-2)*(b-a+1))>>1);
for(i=a;i<b;i++){w=n/Pr[i];L=Lh(sqrt(w)),R-=Lh(w);
if(i<=c){for(j=i;j<L;j++){R+=j;R-=Lh(w/Pr[j]);}}}
return R;}//f(n,k)=f(n,k-1)-f(n/(p_k-1),k-1)*p_k-1
```

39 propagate

```
void propagate(int node,int b,int e){
  tree[node]+=lazy[node]*(e-b+1);
  if(b!=e){
    lazy[node*2]+=lazy[node];
    lazy[node*2+1]+=lazy[node];}
  lazy[node]=0ll;}
void update(int node,int b,int e,int l,int r,ll val){
  propagate(node,b,e);//important
  if(b>r||e<l) return;
  if(b>=l && e<=r){
    tree[node]+=val*(e-b+1);
    if(b!=e){
      lazy[node*2]+=val;
      lazy[node*2+1]+=val;}
    return;}
  update(node*2,b,mid,l,r,val);
  update(node*2+1,mid+1,e,l,r,val);
  tree[node]=tree[node*2]+tree[node*2+1];}
//for query propagate first
```

40 scc

```
// finding Strongly Connected Components.
// nS holds the number of strongly connected components
// fill adj with the graph before running tarjanSCC alg.
int dL[maxn],dN[maxn],vis[maxn],nsn[maxn];
vector<vector<int>>scc;///nS=numSCC
vector<int>S;///dL=dfslow,dN=dfsnum
int dCnt,nS;///nsn=nodesSccNum
vector<vector<int>>adj,rev;
void tarjanSCC(int u) {
  dL[u]=dN[u]=dCnt++;
  S.push_back(u);vis[u]=1;//dL[u]<=dN[u]
  for(int j=0;j<(int)adj[u].size();j++) {
    int v=adj[u][j];if(dN[v]==0)tarjanSCC(v);
    if(vis[v])dL[u] = min(dL[u], dL[v]);}
  if(dL[u]==dN[u]) {//if this is a root (start) of an SCC
    nS++;
    while(1) {
      int v=S.back();S.pop_back();vis[v]=0;
      scc[nS].push_back(v),nsn[v]=nS;
      if (u==v)break;}}}
void dfs(vector<int>&vis,int i,vector<vector<int>> &graph){
  vis[i]=1;
  for(auto &e:graph[i])if(!vis[e])
    dfs(vis,e,graph);}
```

```
main(){
  adj.assign(n,vector<int>()),rev.assign(n,vector<int>());
  memset(nsn,0,sizeof nsn);
  scc.assign(n+10,vector<int>());
  memset(dL,0,sizeof dL);
  memset(dN,0,sizeof dN);
  memset(vis,0,sizeof vis);
  dCnt=nS=0;
  for (int i=0;i<n;i++)
    if (dN[i]==0)tarjanSCC(i);
}
```

41 sublime

```
{"shell_cmd": "g++ -DLOCAL -std=c++14 -Wshadow -Wall \"$file_name\"
    -o \"$file_base_name\"
        -fsanitize=address -fsanitize=undefined -D_GLIBCXX_DEBUG -g
        && gnome-terminal -e 'bash -c
            \"\"${file_path}/${file_base_name}\";
        echo;echo; echo Press ENTER to continue; read line;exit; exec
            bash\"'",}
class ChainCommand(sublime_plugin.WindowCommand):
    def run(self, commands):
        for a in commands:
            self.window.run_command(a[0], *a[1:])
[{"keys": ["f9"], "command": "chain", "args": {
"commands": [["focus_group",{"group":0}],["build"]]}},]
```

42 topsort

```
const static int M=1010;
vector<int> A[M];
bool S[M],V[M],G=0;
vector<int>T;
void D(int x,int p){if(G)return;V[x]=1;S[x]=1;
    for(auto u:A[x]){if(u^p&&V[u]&&S[u]){G=1;return;}
    if(!V[u]){D(u,x);}}S[x]=0;T.pb(x);}
```

43 trie

```
const int M=100010*20+10,L=27;
int T[M][L],z;
void I(string &p){int C=0;
       for(auto k:p){k-='a';
       if(!T[C][k])T[C][k]=++z;C=T[C][k];}
       T[C][L-1]=1;}
bool S(string &p){int C=0;
       for(auto k:p){k-='a';
       if(!T[C][k])return 0;
       C=T[C][k];}return T[C][L-1];}
void st(){memset(T,0,sizeof(T));z=0;}
```

44 wavelettree

```
const int MAXN = (int)1e5+9;///I checked only kth and LTE functions
const int MAXV = (int)1e9;///maximum value of any element in array
///array values can be negative too, use appropriate minimum and
    maximum value
struct wavelet_tree{
  wavelet_tree *l, *r;
  int *b, bsz,csz, lo, hi;///c holds the prefix sum of elements
  ll *c;
  wavelet_tree(){lo=1;hi=0;bsz=0;csz=0,l=NULL;r=NULL;}
  void init(ll *from,ll *to,int x,int y) {
  lo=x,hi=y;if(from>=to)return;
  int 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=(ll*)malloc((to-from+2)*sizeof(ll));csz = 0;
c[csz++]=0;
for(auto it=from;it!=to;it++){
b[bsz]=(b[bsz-1]+f(*it));
c[csz]=(c[csz-1]+(*it));
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, mid+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);}
///sum of numbers in [l ,r] less than or equal to k
ll 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;
ll a[N];
t.init(a+1,a+n+1,0,MAXV);
```

45 zkmp

```
void build_lps(int n) {
       lps[0]=0;
       for(int i=1;i<n;i++) {
              int j=lps[i-1];
              while(j>0&&s[i]!=s[j])j=lps[j-1];
              if(s[i]==s[j])j++;
              lps[i]=j;}}
int find_occ(int n, int m) {
       build_lps(n);//n=pat len
       int i=0,j=0,c=0;
       while(j<m){
              if(s[i]==ss[j]){
                     i++,j++;
                     if(i==n)c++,i=lps[i-1];}
              else{
                     if(i==0)j++;
                     else i=lps[i-1];}}
       return c;}

vector<int>z_function(string s) {
  int n=s.length();vector<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;}
```