Code Template for ACM-ICPC

wcysai @ NJU

June 2, 2018

Contents

1 Graph

1.1 Depth First Search

```
#include<bits/stdc++.h>
#define MAXV 100005
#define INF 1000000000
using namespace std;
int n;
vector<int> G[MAXV];
int depth[MAXV],pa[MAXV],pre[MAXV];
bool used[MAXV];
void dfs_visit(int v)
{
	d[v]=++t;
	color[v]=1;
	for(int i=0;i<G[v].size();i++)
	{
		int to=G[v][i];
		if(color[to]==0)
		{
			p[to]=v;
			dfs_visit(to);
		}
	}
	color[v]=2;
	order.push_front(v);
	f[v]=++t;
}
int solve(int v,int p)
{
	printf("%d ",v+1);
	for(int i=0;i<G[v].size();i++)
	{
		int to=G[v][i];
		if(to==p) continue;
		solve(to,v);
		printf("%d ",v+1);
	}

}
int main()
{
	scanf("%d",&n);
	for(int i=0;i<n-1;i++)
	{
		int x,y;
		scanf("%d%d",&x,&y);
		G[x-1].push_back(y-1);
		G[y-1].push_back(x-1);
	}
	dfs(0,-1,0);
	int v=0,res=-INF;
	for(int i=0;i<n;i++)
	{
		if(depth[i]>res)
		{
			res=depth[i];
```

```
                        v=i;
                }
        }
        dfs(v,-1,0);
        int u=0;res=-INF;
        for(int i=0;i<n;i++)
        {
                if(depth[i]>res)
                {
                        res=depth[i];
                        u=i;
                }
        }
        printf("%d\n",2*n-2-depth[u]);
        memset(used,false,sizeof(used));
        for(int i=u;i!=v;i=pa[i])
        {
                used[i]=true;
                printf("%d ",i+1);
                for(int j=0;j<G[i].size();j++)
                {
                        if(used[G[i][j]]||G[i][j]==pa[i]) continue;
                        solve(G[i][j],i);
                        printf("%d ",i+1);
                }
        }
        printf("%d\n",v+1);
        return 0;
}
```

1.2 Topological Sort

```
#include<bits/stdc++.h>
#define MAXV 100005
using namespace std;
int V,t;
vector<int> G[MAXV];
int d[MAXV],f[MAXV],p[MAXV],color[MAXV];
deque<int> order;
void dfs_visit(int v)
{
        d[v]=++t;
        color[v]=1;
        for(int i=0;i<G[v].size();i++)
        {
                int to=G[v][i];
                if(color[to]==0)
                {
                        p[to]=v;
                        dfs_visit(to);
                }
        }
        color[v]=2;
        order.push_front(v);
        f[v]=++t;
}
void toposort()
```

```
{
	t=0;
	memset(color,0,sizeof(color));
	memset(p,-1,sizeof(p));
	for(int i=0;i<V;i++)
		if(color[i]==0)
			dfs_visit(i);
}
int main()
{
	return 0;
}
```

1.3 Korasaju

```
#include<bits/stdc++.h>
#define MAXV 20000
using namespace std;
int V;
vector<int> G[MAXV];
vector<int> rG[MAXV];
vector<int> vs;
bool used[MAXV];
int cmp[MAXV];
void add_edge(int from,int to)
{
    G[from].push_back(to);
    rG[to].push_back(from);
}
void dfs(int v)
{
    used[v]=true;
    for(int i=0;i<G[v].size();i++)
        if(!used[G[v][i]]) dfs(G[v][i]);
    vs.push_back(v);
}
void rdfs(int v,int k)
{
    used[v]=true;
    cmp[v]=k;
    for(int i=0;i<rG[v].size();i++)
        if(!used[rG[v][i]]) rdfs(rG[v][i],k);
}
int scc()
{
    memset(used,0,sizeof(used));
    vs.clear();
    for(int v=0;v<V;v++)
    {
        if(!used[v]) dfs(v);
    }
    int k=0;
    memset(used,0,sizeof(used));
    for(int i=vs.size()-1;i>=0;i--)
    {
        if(!used[vs[i]]) rdfs(vs[i],k++);
    }
    return k;
}
int main()
{
    return 0;
}
```

1.4 Dijkstra

```
#include<bits/stdc++.h>
#define MAXV 1000
#define MAXE 10000
#define INF 1000000
using namespace std;
struct edge{int to,cost;};
typedef pair<int,int> P;
int V;
vector<edge> G[MAXV];
int d[MAXV];
void dijkstra(int s)
{
    priority_queue<P,vector<P>,greater<P> > que;
    fill(d,d+V,INF);
    d[s]=0;
    que.push(P(0,s));
    while(!que.empty())
    {
        P p=que.top(); que.pop();
        int v=p.second;
        if(d[v]<p.first) continue;
        for(int i=0;i<G[v].size();i++)
        {
            edge e=G[v][i];
            if(d[e.to]>d[v]+e.cost)
            {
                d[e.to]=d[v]+e.cost;
                que.push(P(d[e.to],e.to));
            }
        }
    }
}
int main()
{
    return 0;
}
```

1.5 SPFA

```
#include<bits/stdc++.h>
#define MAXV 1000
#define MAXE 10000
#define INF 1000000
using namespace std;
struct edge{int to,cost;};
typedef pair<int,int> P;
```

```
int V;
vector<edge> G[MAXV];
int d[MAXV];
bool inque[MAXV];
queue<int> que;
void spfa(int s)
{
    fill(d,d+V,INF);
    fill(inque,inque+V,false);
    d[s]=0;
    while(!que.empty()) que.pop();
    que.push(s);
    inque[s]=true;
    while(!que.empty())
    {
        int u=que.front();
        que.pop();
        for(int i=0;i<G[u].size();i++)
        {
            if(d[u]+G[u][i].cost<d[G[u][i].to])
            {
                d[G[u][i].to]=d[u]+G[u][i].cost;
                if(!inque[G[u][i].to])
                {
                    inque[G[u][i].to]=true;
                    que.push(G[u][i].to);
                }
            }
        }
        inque[u]=false;
    }
}
int main()
{
    return 0;
}
```

1.6 Floyd-Warshall

```
#include<bits/stdc++.h>
#define MAXV 10000
#define MAXE 1000
#define INF 1000000
using namespace std;
int d[MAXV][MAXV];
int V;
void floyd_warshall()
{
    for(int k=0;k<V;k++)
        for(int i=0;i<V;i++)
          for(int j=0;j<V;j++) d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
}
int main()
{
    return 0;
}
```

1.7 Prim

```cpp
#include<bits/stdc++.h>
#define MAXV 1000
#define MAXE 10000
#define INF 1000000
using namespace std;
int cost[MAXV][MAXV];
int mincost[MAXV];
bool used[MAXV];
int V;
int prim()
{
    for(int i=0;i<V;i++)
    {
        mincost[i]=INF;
        used[i]=false;
    }
    mincost[0]=0;
    int res=0;
    while(true)
    {
        int v=-1;
        for(int u=0;u<V;u++)
          if(!used[u]&&(v==-1||mincost[u]<mincost[v])) v=u;
        if(v==-1) break;
        used[v]=true;
        res+=mincost[v];
        for(int u=0;u<V;u++)
            mincost[u]=min(mincost[u],cost[v][u]);
    }
    return res;
}
int main()
{
    return 0;
}
```

1.8 Kruskal

```cpp
#include<bits/stdc++.h>
#define MAXV 10000
#define MAXE 1000
#define INF 1000000
#define MAXN 100000
using namespace std;
struct edge{int u,v,cost;};
bool cmp(const edge &e1,const edge &e2)
{
    return e1.cost<e2.cost;
}
edge es[MAXE];
int V,E;
int p[MAXN],r[MAXN];
void init(int n)
{
    for(int i=0;i<n;i++)
    {
        p[i]=i;
        r[i]=0;
    }
}
int find(int x)
{
    if(p[x]==x) return x;
    else return p[x]=find(p[x]);
}
void unite(int x,int y)
{
    x=find(x);
    y=find(y);
    if(x==y) return;
    if(r[x]<r[y]) p[x]=y;
    else
    {
        p[y]=x;
        if(r[x]==r[y]) r[x]++;
    }
}
bool same(int x,int y)
{
    return find(x)==find(y);
}
int kruskal()
{
    sort(es,es+E,cmp);
    init(V);
    int res=0;
    for(int i=0;i<E;i++)
    {
        edge e=es[i];
        if(!same(e.u,e.v))
        {
            unite(e.u,e.v);
            res+=e.cost;
        }
    }
    return res;
}
int main()
{
    return 0;
}
```

1.9 Dinic

```
#include<bits/stdc++.h>
#define MAXV 3005
#define MAXE 50000
#define INF 1000000
using namespace std;
struct edge{int to,cap,rev;};
int V;
```

```
vector<edge> G[MAXV];
int level[MAXV];
int iter[MAXV];
void add_edge(int from,int to,int cap)
{
    G[from].push_back((edge){to,cap,G[to].size()});
    G[to].push_back((edge){from,0,G[from].size()-1});
}
void bfs(int s)
{
    memset(level,-1,sizeof(level));
    queue<int> que;
    level[s]=0;
    que.push(s);
    while(!que.empty())
    {
        int v=que.front(); que.pop();
        for(int i=0;i<G[v].size();i++)
        {
            edge &e=G[v][i];
            if(e.cap>0&&level[e.to]<0)
            {
                level[e.to]=level[v]+1;
                que.push(e.to);
            }
        }
    }
}

int dfs(int v,int t,int f)
{
    if(v==t) return f;
    for(int &i=iter[v];i<G[v].size();i++)
    {
        edge &e=G[v][i];
        if(level[v]<level[e.to]&&e.cap>0)
        {
            int d=dfs(e.to,t,min(f,e.cap));
            if(d>0)
            {
                e.cap-=d;
                G[e.to][e.rev].cap+=d;
                return d;
            }
        }
    }
    return 0;
}
int max_flow(int s,int t)
{
    int flow=0;
    for(;;)
    {
        bfs(s);
        if(level[t]<0) return flow;
        memset(iter,0,sizeof(iter));
        int f;
        while((f=dfs(s,t,INF))>0)
          flow+=f;
    }
}
int main()
{
    scanf("%d",&V);
    for(int i=0;i<V;i++)
        for(int j=i+1;j<V;j++)
            add_edge(i,j,i^j);
    printf("%d\n",max_flow(0,V-1));
    return 0;
}
```

1.10 Mincost Flow

```
#include<bits/stdc++.h>
#define MAXV 1000
#define MAXE 10000
#define INF 1000000
using namespace std;
typedef pair<int,int> P;
struct edge{int to,cap,cost,rev;};
int dist[MAXV],h[MAXV],prevv[MAXV],preve[MAXV];
int V;
vector<edge> G[MAXV];
void add_edge(int from,int to,int cap,int cost)
{
    G[from].push_back((edge){to,cap,cost,G[to].size()});
    G[to].push_back((edge){from,0,-cost,G[from].size()-1});
}
int min_cost_flow(int s,int t,int f)
{
    int res=0;
    fill(h,h+V,0);
    while(f>0)
    {
        priority_queue<P,vector<P>,greater<P> >que;
        fill(dist,dist+V,INF);
        dist[s]=0;
        que.push(P(0,s));
        while(!que.empty())
        {
            P p=que.top(); que.pop();
            int v=p.second;
            if(dist[v]<p.first) continue;
            for(int i=0;i<G[v].size();i++)
            {
                edge &e=G[v][i];
                if(e.cap>0&&dist[e.to]>dist[v]+e.cost+h[v]-h[e.to])
                {
                    dist[e.to]=dist[v]+e.cost+h[v]-h[e.to];
                    prevv[e.to]=v;
                    preve[e.to]=i;
                    que.push(P(dist[e.to],e.to));
                }
            }
        }
        if(dist[t]==INF)
        {
            return -1;
        }
        for(int v=0;v<V;v++) h[v]+=dist[v];
        int d=f;
        for(int v=t;v!=s;v=prevv[v])
        {
            d=min(d,G[prevv[v]][preve[v]].cap);
        }
        f-=d;
        res+=d*h[t];
        for(int v=t;v!=s;v=prevv[v])
        {
            edge &e=G[prevv[v]][preve[v]];
            e.cap-=d;
            G[v][e.rev].cap+=d;
        }
    }
    return res;
}
int main()
{
    return 0;
}
```

1.11 Bipartite Matching

```
#include<cstdio>
#include<cmath>
#include<cstring>
#include<cstdlib>
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#define MAX_V 10000
#define MAXN 1000000
using namespace std;
int V;
vector<int> G[MAX_V];
int match[MAX_V];
bool used[MAX_V];
void add_edge(int u,int v)
{
    G[u].push_back(v);
    G[v].push_back(u);
}
bool dfs(int v)
{
    used[v]=true;
    for(int i=0;i<G[v].size();i++)
    {
        int u=G[v][i],w=match[u];
        if(w<0||!used[w]&&dfs(w))
        {
            match[v]=u;
            match[u]=v;
            return true;
        }
    }
    return false;
}
int bipartite_matching()
{
    int res=0;
    memset(match,-1,sizeof(match));
    for(int v=0;v<V;v++)
    {
        if(match[v]<0)
        {
            memset(used,0,sizeof(used));
            if(dfs(v))
            {
                res++;
            }
        }
    }
    return res;
}
int main()
{
    int p=sieve(1000000);
    return 0;
}
```

1.12 Common Matching

```
#include<bits/stdc++.h>
#define MAXN 500
int n,m,x,y,fore,rear,cnt,ans,father[MAXN],f[MAXN],path[MAXN],tra[MAXN],que[MAXN],match[MAXN];
bool a[MAXN][MAXN],check[MAXN],treec[MAXN],pathc[MAXN];
inline void push(int x)
{
    que[++rear]=x;
    check[x]=true;
    if(!treec[x])
    {
        tra[++cnt]=x;
        treec[x]=true;
    }
}
int root(int x){return f[x]?f[x]=root(f[x]):x;}

void clear()
{
    for(int i=1,j;i<=cnt;++i)
    {
        j=tra[i];
        check[j]=treec[j]=false;
        father[j]=0,f[j]=0;
    }
}

int lca(int u,int v)
```

```
{
    int len=0;
    for(;u;u=father[match[u]])
    {
        u=root(u);
        path[++len]=u;
        pathc[u]=true;
    }
    for(;;v=father[match[v]])
    {
        v=root(v);
        if(pathc[v]) break;
    }
    for(int i=1;i<=len;++i)
    {
        pathc[path[i]]=false;
    }
    return v;
}

void reset(int u,int p)
{
    for(int v;root(u)!=p;)
    {
        if(!check[v=match[u]]) push(v);
        if(f[u]==0) f[u]=p;
        if(f[v]==0) f[v]=p;
        u=father[v];
        if(root(u)!=p) father[u]=v;
    }
}

void flower(int u,int v)
{
    int p=lca(u,v);
    if(root(u)!=p) father[u]=v;
    if(root(v)!=p) father[v]=u;
    reset(u,p),reset(v,p);
}

bool find(int x)
{
    fore=rear=cnt=0,push(x);
    while(fore++<rear)
    {
        int i=que[fore];
        for(int j=1;j<=n;++j)
        {
            if(a[i][j]&&root(i)!=root(j)&&match[j]!=i)
              if(match[j]&&father[match[j]])
                 flower(i,j);
              else if(father[j]==0)
              {
                  father[j]=i;
                  tra[++cnt]=j;
                  treec[j]=true;
                  if(match[j])
                    push(match[j]);
                  else
```

```
                {
                    for(int k=i,l=j,p;k;l=p,k=father[l])
                    {
                        p=match[k];
                        match[k]=l;
                        match[l]=k;
                    }
                    return true;
                }
            }
        }
    }
    return false;
}

void matching()
{
    for(int i=1;i<=n;i++)
        if(match[i]==0)
        {
            if(find(i)) ans++;
            clear();
        }
}

int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++)
    {
      int x,y;
      scanf("%d%d",&x,&y);
      a[x][y]=a[y][x]=true;
    }
    matching();
    printf("%d\n",ans);
    return 0;
}
```

1.13 Hopcroft-Karp

```
#include<bits/stdc++.h>
#define MAXN 50030
using namespace std;
int n1,n2;
vector<int> G[MAXN];
int mx[MAXN],my[MAXN];
queue<int> que;
int dx[MAXN],dy[MAXN];
bool vis[MAXN];
bool find(int u)
{
    for(int i=0;i<G[u].size();i++)
    {
        if(!vis[G[u][i]]&&dy[G[u][i]]==dx[u]+1)
        {
            vis[G[u][i]]=true;
            if(!my[G[u][i]]||find(my[G[u][i]]))
            {
                mx[u]=G[u][i];
                my[G[u][i]]=u;
                return true;
            }
        }
    }
    return false;
}
int matching()
{
    memset(mx,0,sizeof(mx));
    memset(my,0,sizeof(my));
    int ans=0;
    while(true)
    {
        bool flag=false;
        while(!que.empty()) que.pop();
        memset(dx,0,sizeof(dx));
        memset(dy,0,sizeof(dy));
        for(int i=1;i<=n1;i++)
            if(!mx[i]) que.push(i);
        while(!que.empty())
        {
            int u=que.front();
            que.pop();
            for(int i=0;i<G[u].size();i++)
                if(!dy[G[u][i]])
                {
                    dy[G[u][i]]=dx[u]+1;
                    if(my[G[u][i]])
                    {
                        dx[my[G[u][i]]]=dy[G[u][i]]+1;
                        que.push(my[G[u][i]]);
                    }
                    else flag=true;
                }
        }
        if(!flag) break;
        memset(vis,0,sizeof(vis));
        for(int i=1;i<=n1;i++)
            if(!mx[i]&&find(i)) ans++;
    }
    return ans;
}
int main()
{
    return 0;
}
```

1.14 Kuhn-Munkres

```
#include<bits/stdc++.h>
#define MAXN 505
#define INF 1000000000
using namespace std;
```

```
int w[MAXN][MAXN],x[MAXN],y[MAXN];
int prev_x[MAXN],prev_y[MAXN],son_y[MAXN],slack[MAXN],par[MAXN];
int lx,ly,pop;
void adjust(int v)
{
    son_y[v]=prev_y[v];
    if(prev_x[son_y[v]]!=2)
        adjust(prev_x[son_y[v]]);
}
bool find(int v)
{
    for(int i=0;i<pop;i++)
    {
        if(prev_y[i]==-1)
        {
            if(slack[i]>x[v]+y[i]-w[v][i])
            {
                slack[i]=x[v]+y[i]-w[v][i];
                par[i]=v;
            }
            if(x[v]+y[i]==w[v][i])
            {
                prev_y[i]=v;
                if(son_y[i]==-1)
                {
                    adjust(i);
                    return true;
                }
                if(prev_x[son_y[i]]!=-1)
                    continue;
                prev_x[son_y[i]]=i;
                if(find(son_y[i]))
                    return true;
            }
        }
    }
    return false;
}
int km()
{
    int m;
    for(int i=0;i<pop;i++)
    {
        son_y[i]=-1;
        y[i]=0;
    }
    for(int i=0;i<pop;i++)
    {
        x[i]=0;
        for(int j=0;j<pop;j++)
            x[i]=max(x[i],w[i][j]);
    }
    bool flag;
    for(int i=0;i<pop;i++)
    {
        for(int j=0;j<pop;j++)
        {
            prev_x[j]=prev_y[j]=-1;
            slack[j]=INF;
        }
        prev_x[i]=-2;
        if(find(i)) continue;
        flag=false;
        while(!flag)
        {
            m=INF;
            for(int j=0;j<pop;j++)
                if(prev_y[j]==-1)
                    m=min(m,slack[j]);
            for(int j=0;j<pop;j++)
            {
                if(prev_x[j]!=-1)
                    x[j]-=m;
                if(prev_y[j]!=-1)
                    y[j]+=m;
                else
                    slack[j]-=m;
            }
            for(int j=0;j<pop;j++)
            {
                if(prev_y[j]==-1&&!slack[j])
                {
                    prev_y[j]=par[j];
                    if(son_y[j]==-1)
                    {
                        adjust(j);
                        flag=true;
                        break;
                    }
                    prev_x[son_y[j]]=j;
                    if(find(son_y[j]))
                    {
                        flag=true;
                        break;
                    }
                }
            }
        }
    }
    int ans=0;
    for(int i=0;i<pop;i++)
        ans+=w[son_y[i]][i];
    return ans;
}
int main()
{
    return 0;
}
```

1.15 Lowest Common Ancestor(binary search)

```
#include<bits/stdc++.h>
#define MAXV 100005
#define MAXLOGV 20
using namespace std;
vector<int> G[MAXV];
```

```
int root;
int parent[MAXLOGV][MAXV];
int depth[MAXV];
int n,q;
void dfs(int v,int p,int d)
{
    parent[0][v]=p;
    depth[v]=d;
    for(int i=0;i<G[v].size();i++)
        if(G[v][i]!=p) dfs(G[v][i],v,d+1);
}
void init(int V)
{
    dfs(root,-1,0);
    for(int k=0;k+1<MAXLOGV;k++)
    {
        for(int v=0;v<V;v++)
        {
            if(parent[k][v]<0) parent[k+1][v]=-1;
            else parent[k+1][v]=parent[k][parent[k][v]];
        }
    }
}
int lca(int u,int v)
{
    if(depth[u]>depth[v]) swap(u,v);
    for(int k=0;k<MAXLOGV;k++)
    {
        if((depth[v]-depth[u])>>k&1)
            v=parent[k][v];
    }
    if(u==v) return u;
    for(int k=MAXLOGV-1;k>=0;k--)
    {
        if(parent[k][u]!=parent[k][v])
        {
            u=parent[k][u];
            v=parent[k][v];
        }
    }
    return parent[0][u];
}
int dis(int u,int v)
{
    return depth[u]+depth[v]-2*depth[lca(u,v)];
}
int main()
{
    return 0;
}
```

1.16 Lowest Common Ancestor(rmq)

```
#include<bits/stdc++.h>
#define MAXV 100005
#define MAXLOGV 32
using namespace std;
```

```
int N,M,Q;
int st[MAXLOGV][MAXV];
vector<int> G[MAXV];
int root;
int vs[MAXV*2-1];
int depth[MAXV*2-1];
int id[MAXV];
void dfs(int v,int p,int d,int &k)
{
    id[v]=k;
    vs[k]=v;
    depth[k++]=d;
    for(int i=0;i<G[v].size();i++)
    {
        if(G[v][i]!=p)
        {
            dfs(G[v][i],v,d+1,k);
            vs[k]=v;
            depth[k++]=d;
        }
    }
}
int getMin(int x, int y)
{
    return depth[x]<depth[y]?x:y;
}

void rmq_init(int n)
{
    for(int i=0;i=n;++i) st[0][i]=i;
    for(int i=1;1<<i<n;++i)
        for(int j=0;j+(1<<i)-1<n;++j)
            st[i][j]=getMin(st[i-1][j],st[i-1][j+(1<<(i-1))]);
}
void init(int V)
{
    int k=0;
    dfs(root,-1,0,k);
    rmq_init(V*2-1);
}
int query(int l, int r)
{
    int k=31-__builtin_clz(r-l+1);
    return getMin(st[k][l],st[k][r-(1<<k)+1]);
}
int lca(int u,int v)
{
    if(u==v) return u;
    return vs[query(min(id[u],id[v]),max(id[u],id[v]))];
}
int dis(int u,int v)
{
    return depth[id[u]]+depth[id[v]]-2*depth[id[lca(u,v)]];
}
int main()
{
    scanf("%d%d",&N,&M);
    for(int i=0;i<M;i++)
    {
        int x,y;
        scanf("%d%d",&x,&y);
        G[x].push_back(y);
        G[y].push_back(x);
    }
    root=0;
    init(N);
    scanf("%d",&Q);
    while(Q--)
    {
        int x,y;
        scanf("%d%d",&x,&y);
        printf("%d\n",lca(x,y));
    }
    return 0;
}
```

1.17 Dominator Tree

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
vector<int> G[MAXN],rG[MAXN],dt[MAXN],bucket[MAXN];
int sdom[MAXN],idom[MAXN],arr[MAXN],rev[MAXN],par[MAXN],dsu[MAXN],label[MAXN];
int n,m,t;
int find(int u,int x=0)
{
        if(u==dsu[u]) return x?-1:u;
        int v=find(dsu[u],x+1);
        if(v<0) return u;
        if(sdom[label[dsu[u]]]<sdom[label[u]])
                label[u]=label[dsu[u]];
        dsu[u]=v;
        return x?v:label[u];
}
void unite(int u,int v)
{
        dsu[v]=u;
}
void dfs(int v)
{
        t++;arr[v]=t;rev[t]=v;
        for(int i=0;i<G[v].size();i++)
        {
                int to=G[v][i];
                if(!arr[to]) dfs(to),par[arr[to]]=arr[v];
                rG[arr[to]].push_back(arr[v]);
        }
}
int main()
{
        scanf("%d%d",&n,&m);
        for(int i=1;i<=m;i++)
        {
                int u,v;
                scanf("%d%d",&u,&v);
                G[u].push_back(v);
        }
        for(int i=1;i<=n;i++)
                sdom[i]=i,idom[i]=0,label[i]=i,dsu[i]=i;
        dfs(1);
        for(int i=n;i>=1;i--)
        {
                for(int j=0;j<rG[i].size();j++)
                        sdom[i]=min(sdom[i],sdom[find(rG[i][j])]);
                if(i>1) bucket[sdom[i]].push_back(i);
                for(int j=0;j<bucket[i].size();j++)
                {
                        int w=bucket[i][j],v=find(w);
                        if(sdom[v]==sdom[w]) idom[w]=sdom[w];
                        else idom[w]=v;
                }
                if(i>1) unite(par[i],i);
        }
        for(int i=2;i<=n;i++)
        {
                if(idom[i]!=sdom[i]) idom[i]=idom[idom[i]];
                dt[rev[idom[i]]].push_back(rev[i]);
                printf("%d %d\n",rev[i],rev[idom[i]]);
        }
        return 0;
}
```

2 DataStructures

2.1 Fenwick Tree

```
#include<bits/stdc++.cpp>
#define MAXN 100000
using namespace std;
int bit[MAXN+1],n;
int sum(int i)
{
    int s=0;
    while(i>0)
    {
        s+=bit[i];
        i-=i&-i;
    }
    return s;
}
void add(int i,int x)
{
    while(i<=n)
    {
        bit[i]+=x;
        i+=i&-i;
    }
}
int main()
{
    return 0;
}
```

2.2 Mo's algorithm

```
#include<bits/stdc++.h>
#define MAXN 100005
#define MAXM 100005
using namespace std;
struct query
{
    int l,r,id;
}save[MAXM];
int cnt[MAXN],a[MAXN],out[MAXN];
int n,m,ans,block;
bool cmp(query x,query y)
{
    if(x.l/block!=y.l/block) return x.l/block<y.l/block;
    return x.r<y.r^(x.l/block&1);
}
void add(int pos)
{
    if(cnt[a[pos]]==a[pos]) ans--;
    cnt[a[pos]]++;
    if(cnt[a[pos]]==a[pos]) ans++;
    return;
}
void del(int pos)
{
    if(cnt[a[pos]]==a[pos]) ans--;
    cnt[a[pos]]--;
    if(cnt[a[pos]]==a[pos]) ans++;
    return;
}
void update(int cl,int cr,int l,int r)
{
    while(cl<l)
    {
        del(cl);
        cl++;
    }
    while(cl>l)
    {
        cl--;
        add(cl);
    }
    while(cr>r)
    {
        del(cr);
        cr--;
    }
    while(cr<r)
    {
        cr++;
        add(cr);
    }
    return;
}
int main()
{
    scanf("%d %d",&n,&m);
    block=(int)sqrt(n);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
        if(a[i]>100000) a[i]=100001;
    }
    for(int i=0;i<m;i++)
    {
        save[i].id=i;
        scanf("%d %d",&save[i].l,&save[i].r);
    }
    sort(save,save+m,cmp);
    memset(cnt,0,sizeof(cnt));
    ans=0;
    for(int i=save[0].l;i<=save[0].r;i++)
    {
        if(cnt[a[i]]==a[i]) ans--;
        cnt[a[i]]++;
        if(cnt[a[i]]==a[i]) ans++;
    }
    out[save[0].id]=ans;
    int cl=save[0].l,cr=save[0].r;
    for(int i=1;i<m;i++)
    {
        update(cl,cr,save[i].l,save[i].r);
        out[save[i].id]=ans;
        cl=save[i].l;
        cr=save[i].r;
    }
    for(int i=0;i<m;i++)
        printf("%d\n",out[i]);
    return 0;
}
```

2.3 Heavy Light Decomposition

```
#include<bits/stdc++.h>
#define MAXN 400005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
struct node
{
    int l,r,maxi,sum;
};
```

```
int tot,q,n,k,a[MAXN];
int pa[MAXN],dep[MAXN],sz[MAXN],wson[MAXN],top[MAXN],spos[MAXN],tpos[MAXN];
struct segtree
{
    node seg[4*MAXN];
    int id[MAXN];
    void build(int k,int l,int r)
    {
        seg[k].l=l;seg[k].r=r;
        if(l==r)
        {
            seg[k].maxi=seg[k].sum=a[tpos[l]];
            id[l]=k;
            return;
        }
        int mid=(l+r)/2;
        build(k*2,l,mid);build(k*2+1,mid+1,r);
        seg[k].maxi=max(seg[k*2].maxi,seg[k*2+1].maxi);
        seg[k].sum=seg[k*2].sum+seg[k*2+1].sum;
    }
    void update(int k,int x)
    {
        k=id[k];
        seg[k].maxi=seg[k].sum=x;
        while(k>1)
        {
            k=k/2;
            seg[k].maxi=max(seg[k*2].maxi,seg[k*2+1].maxi);
            seg[k].sum=seg[k*2].sum+seg[k*2+1].sum;
        }
    }
    int query1(int k,int l,int r)
    {
        if(seg[k].l>r||seg[k].r<l) return -INF;
        if(seg[k].l>=l&&seg[k].r<=r) return seg[k].maxi;
        return max(query1(k*2,l,r),query1(k*2+1,l,r));
    }
    int query2(int k,int l,int r)
    {
        if(seg[k].l>r||seg[k].r<l) return 0;
        if(seg[k].l>=l&&seg[k].r<=r) return seg[k].sum;
        return query2(k*2,l,r)+query2(k*2+1,l,r);
    }
    int query_max(int l,int r)
    {
        return query1(1,l,r);
    }
    int query_sum(int l,int r)
    {
        return query2(1,l,r);
    }
}tree;
vector<int> G[MAXN];
void dfs1(int v,int p,int d)
{
    dep[v]=d;pa[v]=p;sz[v]=1;
    for(int i=0;i<G[v].size();i++)
    {
        int to=G[v][i];
        if(to==p) continue;
        dfs1(to,v,d+1);
        if(sz[to]>sz[wson[v]]) wson[v]=to;
        sz[v]+=sz[to];
    }
}
void dfs2(int v,int p,int num)
{
    top[v]=num;
    spos[v]=++tot;
    tpos[tot]=v;
    if(wson[v]) dfs2(wson[v],v,num);
    for(int i=0;i<G[v].size();i++)
    {
        int to=G[v][i];
        if(to==p||to==wson[v]) continue;
        dfs2(to,v,to);
    }
}
void init()
{
    tot=0;dfs1(1,1,1);
    dfs2(1,0,1);
    tree.build(1,1,n);
}
void update(int k,int x)
{
    tree.update(spos[k],x);
}
int query_max(int u,int v)
{
    int res=-INF;
    while(top[u]!=top[v])
    {
        if(dep[top[u]]<dep[top[v]]) swap(u,v);
        res=max(res,tree.query_max(spos[top[u]],spos[u]));
        u=pa[top[u]];
    }
    if(dep[u]<dep[v]) swap(u,v);
    res=max(res,tree.query_max(spos[v],spos[u]));
    return res;
}
int query_sum(int u,int v)
{
    int res=0;
    while(top[u]!=top[v])
    {
        if(dep[top[u]]<dep[top[v]]) swap(u,v);
        res+=tree.query_sum(spos[top[u]],spos[u]);
        u=pa[top[u]];
    }
    if(dep[u]<dep[v]) swap(u,v);
    res+=tree.query_sum(spos[v],spos[u]);
    return res;
}
char str[10];
int x,y;
int main()
{
    scanf("%d",&n);
    for(int i=0;i<n-1;i++)
    {
        int u,v;
        scanf("%d%d",&u,&v);
        G[u].push_back(v);G[v].push_back(u);
    }
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    init();
    scanf("%d",&q);
    while(q--)
    {
        scanf("%s%d%d",str,&x,&y);
        if(str[1]=='H') update(x,y);
        if(str[1]=='M') printf("%d\n",query_max(x,y));
        if(str[1]=='S') printf("%d\n",query_sum(x,y));
    }
    return 0;
}
```

2.4 Segment Tree

```
#include<bits/stdc++.h>
#define MAXN 500030
using namespace std;
int n,m,h[MAXN],c[MAXN];
struct node
{
    int l,r,left,right,lazy;
}seg[4*MAXN];
bool cmp(int x,int y)
{
    return x>y;
}
void build(int k,int l,int r)
{
    seg[k].l=l;
    seg[k].r=r;
    seg[k].lazy=0;
    if(l==r)
    {
        seg[k].left=seg[k].right=h[l];
        return;
    }
    int mid=(l+r)/2;
    build(k*2,l,mid);
    build(k*2+1,mid+1,r);
    seg[k].left=seg[k*2].left;
    seg[k].right=seg[k*2+1].right;
}
void Lazy(int k)
{
    if(seg[k].l==seg[k].r)
    {
        seg[k].lazy=0;
        return;
    }
    seg[k*2].left-=seg[k].lazy;
    seg[k*2].right-=seg[k].lazy;
    seg[k*2+1].left-=seg[k].lazy;
    seg[k*2+1].right-=seg[k].lazy;
    seg[k*2].lazy+=seg[k].lazy;
    seg[k*2+1].lazy+=seg[k].lazy;
    seg[k].lazy=0;
}
bool update(int k,int l,int r)
{
    if(r<l) return true;
    if(seg[k].l>r||seg[k].r<l) return true;
    if(seg[k].l>=l&&seg[k].r<=r)
    {
        seg[k].lazy++;
        seg[k].left--;
        seg[k].right--;
        return (seg[k].left>=0&&seg[k].right>=0);
    }
    if(seg[k].lazy) Lazy(k);
    bool f1=update(k*2,l,r);
    bool f2=update(k*2+1,l,r);
    seg[k].left=seg[k*2].left;
    seg[k].right=seg[k*2+1].right;
    return(f1&&f2);
}
int findval(int k,int l,int r,int x)
{
    if(seg[k].lazy) Lazy(k);
    if(l==r) return seg[k].left;
    int mid=(l+r)/2;
    if(x>mid) return findval(k*2+1,mid+1,r,x);
    return findval(k*2,l,mid,x);
}
int findleft(int k,int l,int r,int x)
{
    if(seg[k].lazy) Lazy(k);
    if(l==r) return l;
    int mid=(l+r)/2;
    if(seg[k*2].right<=x) return findleft(k*2,l,mid,x);
    return findleft(k*2+1,mid+1,r,x);
}
int findright(int k,int l,int r,int x)
{
    if(seg[k].lazy) Lazy(k);
    if(l==r) return r;
    int mid=(l+r)/2;
    if(seg[k*2].lazy) Lazy(k*2);
    if(seg[k*2+1].lazy) Lazy(k*2+1);
    if(seg[k*2+1].left>=x) return findright(k*2+1,mid+1,r,x);
    return findright(k*2,l,mid,x);
}
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
        scanf("%d",&h[i]);
    sort(h+1,h+n+1,cmp);
    for(int i=0;i<m;i++)
```

```
        scanf("%d",&c[i]);
    build(1,1,n);
    int cnt=0;
    while(true)
    {
        if(c[cnt]>n) break;
        int x=findval(1,1,n,c[cnt]);
        int a=findleft(1,1,n,x);
        int b=findright(1,1,n,x);
        bool f1=update(1,1,a-1),f2=update(1,b-c[cnt]+a,b);
        if(!(f1&&f2)) break;
        cnt++;
        if(cnt>=m) break;
    }
    printf("%d\n",cnt);
    return 0;
}
```

2.5 Segment Tree Beats

```
#include<bits/stdc++.h>
#define MAXN 1000005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
struct node
{
        ll l,r,sum,maxx,secx,maxnum,lazy;
}seg[4*MAXN];
ll t,n,m,a[MAXN];
void Lazy(ll k)
{
        if(seg[k].l==seg[k].r||seg[k].lazy==INT_MAX) return;
        if(seg[k*2].lazy>=seg[k].lazy&&seg[k*2].maxx>seg[k].lazy)
        {
                seg[k*2].sum-=(seg[k*2].maxx-seg[k].lazy)*seg[k*2].maxnum;
                seg[k*2].maxx=seg[k].lazy;
                seg[k*2].lazy=seg[k].lazy;
        }
        if(seg[k*2+1].lazy>=seg[k].lazy&&seg[k*2+1].maxx>seg[k].lazy)
        {
                seg[k*2+1].sum-=(seg[k*2+1].maxx-seg[k].lazy)*seg[k*2+1].maxnum;
                seg[k*2+1].maxx=seg[k].lazy;
                seg[k*2+1].lazy=seg[k].lazy;
        }
        seg[k].lazy=INT_MAX;
        return;
}
void merge(ll k)
{
        seg[k].sum=seg[k*2].sum+seg[k*2+1].sum;
        seg[k].maxx=max(seg[k*2].maxx,seg[k*2+1].maxx);
        ll res=0,ans=-1;
        if(seg[k*2].maxx==seg[k].maxx) res+=seg[k*2].maxnum;
        if(seg[k*2+1].maxx==seg[k].maxx) res+=seg[k*2+1].maxnum;
        seg[k].maxnum=res;
        if(seg[k*2].maxx!=seg[k].maxx) ans=max(ans,seg[k*2].maxx);
        if(seg[k*2].secx!=seg[k].maxx) ans=max(ans,seg[k*2].secx);
        if(seg[k*2+1].maxx!=seg[k].maxx) ans=max(ans,seg[k*2+1].maxx);
        if(seg[k*2+1].secx!=seg[k].maxx) ans=max(ans,seg[k*2+1].secx);
        seg[k].secx=ans;
        //printf("l=%lld r=%lld maxx=%lld secx=%lld maxnum=%lld sum=%lld
            lazy=%lld\n",seg[k].l,seg[k].r,seg[k].maxx,seg[k].secx,seg[k].maxnum,seg[k].sum,seg[k].lazy);
}
void build(ll k,ll l,ll r)
{
        seg[k].l=l;seg[k].r=r;seg[k].lazy=INT_MAX;
        if(l==r)
        {
                seg[k].maxx=seg[k].sum=a[l];
                seg[k].maxnum=1;
                seg[k].secx=-1;
                return;
        }
        ll mid=(l+r)/2;
        build(k*2,l,mid);build(k*2+1,mid+1,r);
        merge(k);
}
void update(ll k,ll l,ll r,ll x)
{
        if(seg[k].l>r||seg[k].r<l||seg[k].maxx<=x) return;
        if(seg[k].l>=l&&seg[k].r<=r&&seg[k].secx<x)
        {
                seg[k].sum-=(seg[k].maxx-x)*seg[k].maxnum;
                seg[k].maxx=x;
                seg[k].lazy=x;
                return;
        }
        Lazy(k);
        update(k*2,l,r,x);update(k*2+1,l,r,x);
        merge(k);
}
ll query1(ll k,ll l,ll r)
{
        if(seg[k].l>r||seg[k].r<l) return 0;
        if(seg[k].l>=l&&seg[k].r<=r) return seg[k].maxx;
        Lazy(k);
        return max(query1(k*2,l,r),query1(k*2+1,l,r));
}
ll query2(ll k,ll l,ll r)
{
        if(seg[k].l>r||seg[k].r<l) return 0;
        if(seg[k].l>=l&&seg[k].r<=r) return seg[k].sum;
        Lazy(k);
        return query2(k*2,l,r)+query2(k*2+1,l,r);
}
int main()
{
        scanf("%lld",&t);
        while(t--)
        {
                scanf("%lld%lld",&n,&m);
            for(ll i=1;i<=n;i++) scanf("%lld",&a[i]);
            build(1,1,n);
            for(ll i=1;i<=m;i++)
            {
                  ll type,x,y,z;
                  scanf("%lld",&type);
                  if(type==0)
                  {
                        scanf("%lld%lld%lld",&x,&y,&z);
                        update(1,x,y,z);
                  }
                  else if(type==1)
                  {
                        scanf("%lld%lld",&x,&y);
                        printf("%lld\n",query1(1,x,y));
                  }
                  else
                  {
                        scanf("%lld%lld",&x,&y);
                        printf("%lld\n",query2(1,x,y));
                  }
            }
      }
      return 0;
}
```

2.6 Splay

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
#define MAXN 1000000
int ch[MAXN][2],f[MAXN],size[MAXN],cnt[MAXN],key[MAXN];
int sz,root;
inline void clear(int x){
   ch[x][0]=ch[x][1]=f[x]=size[x]=cnt[x]=key[x]=0;
}
inline bool get(int x){
   return ch[f[x]][1]==x;
}
inline void update(int x){
   if (x){
      size[x]=cnt[x];
      if (ch[x][0]) size[x]+=size[ch[x][0]];
      if (ch[x][1]) size[x]+=size[ch[x][1]];
   }
}
inline void rotate(int x){
   int old=f[x],oldf=f[old],whichx=get(x);
   ch[old][whichx]=ch[x][whichx^1]; f[ch[old][whichx]]=old;
   ch[x][whichx^1]=old; f[old]=x;
   f[x]=oldf;
   if (oldf)
      ch[oldf][ch[oldf][1]==old]=x;
   update(old); update(x);
}
```

```
inline void splay(int x){
    for (int fa;fa=f[x];rotate(x))
      if (f[fa])
        rotate((get(x)==get(fa))?fa:x);
    root=x;
}
inline void insert(int x){
    if (root==0){sz++; ch[sz][0]=ch[sz][1]=f[sz]=0; root=sz; size[sz]=cnt[sz]=1; key[sz]=x;
         return;}
    int now=root,fa=0;
    while(1){
        if (x==key[now]){
            cnt[now]++; update(now); update(fa); splay(now); break;
        }
        fa=now;
        now=ch[now][key[now]<x];
        if (now==0){
            sz++;
            ch[sz][0]=ch[sz][1]=0;
            f[sz]=fa;
            size[sz]=cnt[sz]=1;
            ch[fa][key[fa]<x]=sz;
            key[sz]=x;
            update(fa);
            splay(sz);
            break;
        }
    }
}
inline int find(int x){
    int now=root,ans=0;
    while(1){
        if (x<key[now])
          now=ch[now][0];
        else{
            ans+=(ch[now][0]?size[ch[now][0]]:0);
            if (x==key[now]){
                splay(now); return ans+1;
            }
            ans+=cnt[now];
            now=ch[now][1];
        }
    }
}
inline int findx(int x){
    int now=root;
    while(1){
        if (ch[now][0]&&x<=size[ch[now][0]])
          now=ch[now][0];
        else{
            int temp=(ch[now][0]?size[ch[now][0]]:0)+cnt[now];
            if (x<=temp) return key[now];
            x-=temp; now=ch[now][1];
        }
    }
}
inline int pre(){
    int now=ch[root][0];
    while (ch[now][1]) now=ch[now][1];
    return now;
}
inline int next(){
    int now=ch[root][1];
    while (ch[now][0]) now=ch[now][0];
    return now;
}
inline void del(int x){
    int whatever=find(x);
    if (cnt[root]>1){cnt[root]--; update(root); return;}
    if (!ch[root][0]&&!ch[root][1]) {clear(root); root=0; return;}
    if (!ch[root][0]){
        int oldroot=root; root=ch[root][1]; f[root]=0; clear(oldroot); return;
    }
    else if (!ch[root][1]){
        int oldroot=root; root=ch[root][0]; f[root]=0; clear(oldroot); return;
    }
    int leftbig=pre(),oldroot=root;
    splay(leftbig);
    ch[root][1]=ch[oldroot][1];
    f[ch[oldroot][1]]=root;
    clear(oldroot);
    update(root);
}
int main(){
    int n,opt,x;
    scanf("%d",&n);
    for (int i=1;i<=n;++i){
        scanf("%d%d",&opt,&x);
        switch(opt){
            case 1: insert(x); break;
            case 2: del(x); break;
            case 3: printf("%d\n",find(x)); break;
            case 4: printf("%d\n",findx(x)); break;
            case 5: insert(x); printf("%d\n",key[pre()]); del(x); break;
            case 6: insert(x); printf("%d\n",key[next()]); del(x); break;
        }
    }
}
```

2.7 Treap

```
#include<bits/stdc++.h>
#define MAXN 50030
#define INF 1000000000
using namespace std;
struct treap
{
    int root,treapcnt,key[MAXN],priority[MAXN],childs[MAXN][2],cnt[MAXN],size[MAXN];
    treap()
    {
        root=0;
        treapcnt=1;
        priority[0]=INF;
        size[0]=0;
    }
```

```
void update(int x)
{
    size[x]=size[childs[x][0]]+cnt[x]+size[childs[x][1]];
}

void rotate(int &x,int t)
{
    int y=childs[x][t];
    childs[x][t]=childs[y][1-t];
    childs[y][1-t]=x;
    update(x);
    update(y);
    x=y;
}

void _insert(int &x,int k)
{
    if(x)
    {
        if(key[x]==k)
        {
            cnt[x]++;
        }
        else
        {
            int t=key[x]<k;
            _insert(childs[x][t],k);
            if(priority[childs[x][t]]<priority[x])
                {
                rotate(x,t);
            }
        }
    }
    else
    {
        x=treapcnt++;
        key[x]=k;
        cnt[x]=1;
        priority[x]=rand();
        childs[x][0]=childs[x][1]=0;
    }
    update(x);
}

void _erase(int &x,int k)
{
    if(key[x]==k)
    {
        if(cnt[x]>1)
        {
            cnt[x]--;
        }
        else
        {
            if(childs[x][0]==0&&childs[x][1]==0)
            {
                x=0;
                return;
                }
                int t=priority[childs[x][0]]>priority[childs[x][1]];
                rotate(x,t);
                _erase(x,k);
            }
        }
        else
        {
            _erase(childs[x][key[x]<k],k);
        }
        update(x);
    }

    int _getKth(int &x,int k)
    {
        if(k<=size[childs[x][0]])
        {
            return _getKth(childs[x][0],k);
        }
        k-=size[childs[x][0]]+cnt[x];
        if(k<=0)
        {
            return key[x];
        }
        return _getKth(childs[x][1],k);
    }

    void insert(int k)
    {
        _insert(root,k);
    }

    void erase(int k)
    {
        _erase(root,k);
    }

    int getKth(int k)
    {
        return _getKth(root,k);
    }
};

int main()
{
    return 0;
}
```

2.8 Union Set

```
#include<bits/stdc++.h>
#define MAXN 100000
using namespace std;
int p[MAXN],r[MAXN];
void init(int n)
{
    for(int i=0;i<n;i++)
    {
        p[i]=i;
        r[i]=0;
    }
}
int find(int x)
{
    if(p[x]==x) return x;
    else return p[x]=find(p[x]);
}
void unite(int x,int y)
{
    x=find(x);
    y=find(y);
    if(x==y) return;
    if(r[x]<r[y]) p[x]=y;
    else
    {
        p[y]=x;
        if(r[x]==r[y]) r[x]++;
    }
}
bool same(int x,int y)
{
    return find(x)==find(y);
}
int main()
{
    return 0;
}
```

2.9 Sparse Table

```
#include<bits/stdc++.h>
#define MAXN 100000
using namespace std;
int N,Q;
int a[MAXN];
int st[MAXN][32];
int pre[MAXN];
void init(int n,int *arr)
{
    pre[1]=0;
    for(int i=2;i<=n;i++)
    {
        pre[i]=pre[i-1];
        if ((1<<pre[i]+1)==i) ++pre[i];
    }
    for(int i=n-1;i>=0;--i)
    {
        st[i][0]=arr[i];
        for(int j=1;(i+(1<<j)-1)<n;++j)
            st[i][j]=min(st[i][j-1],st[i+(1<<j-1)][j-1]);
    }
}
int query(int l,int r)
{
    int len=r-l+1,k=pre[len];
    return min(st[l][k],st[r-(1<<k)+1][k]);
}
int main()
{
    scanf("%d",&N);
    for(int i=0;i<N;i++)
        scanf("%d",&a[i]);
    init(N,a);
    scanf("%d",&Q);
    while(Q--)
    {
        int x,y;
        scanf("%d%d",&x,&y);
        printf("%d\n",query(x,y));
    }
    return 0;
}
```

3 Geometry

3.1 Geometric Template

```
#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>


using namespace std;
const double PI = acos(-1.0);
const double eps = 1e-10;

/****************          ***************/
//           tatb
int sgn( double ta, double tb)
{
    if(fabs(ta-tb)<eps)return 0;
    if(ta<tb)  return -1;
    return 1;
}

//
class Point
{
public:

    double x, y;

    Point(){}
    Point( double tx, double ty){ x = tx, y = ty;}

    bool operator < (const Point &_se) const
    {
        return x<_se.x || (x==_se.x && y<_se.y);
    }
    friend Point operator + (const Point &_st,const Point &_se)
    {
        return Point(_st.x + _se.x, _st.y + _se.y);
    }
    friend Point operator - (const Point &_st,const Point &_se)
    {
        return Point(_st.x - _se.x, _st.y - _se.y);
    }
    //              (   double   )
    bool operator == (const Point &_off)const
    {
        return sgn(x, _off.x) == 0 && sgn(y, _off.y) == 0;
    }

};

/****************          ***************/
//
double dot(const Point &po,const Point &ps,const Point &pe)
{
    return (ps.x - po.x) * (pe.x - po.x) + (ps.y - po.y) * (pe.y - po.y);
}
//
double xmult(const Point &po,const Point &ps,const Point &pe)
{
    return (ps.x - po.x) * (pe.y - po.y) - (pe.x - po.x) * (ps.y - po.y);
}
//
double getdis2(const Point &st,const Point &se)
{
    return (st.x - se.x) * (st.x - se.x) + (st.y - se.y) * (st.y - se.y);
}
//
double getdis(const Point &st,const Point &se)
{
    return sqrt((st.x - se.x) * (st.x - se.x) + (st.y - se.y) * (st.y - se.y));
}

//
class Line
{
public:

    Point s, e;//                   [s]        [e]
    double a, b, c;//       ,ax+by+c=0
    double angle;//               [-pi,pi]

    Line(){}
    Line( Point ts, Point te):s(ts),e(te){}//get_angle();}
    Line(double _a,double _b,double _c):a(_a),b(_b),c(_c){}

    //
    bool operator < (const Line &ta)const
    {
        return angle<ta.angle;
    }
    //
    friend double operator / ( const Line &_st, const Line &_se)
    {
```

```
    return (_st.e.x - _st.s.x) * (_se.e.y - _se.s.y) - (_st.e.y - _st.s.y) * (_se.e.x -
        _se.s.x);
}
//
friend double operator *( const Line &_st, const Line &_se)
{
    return (_st.e.x - _st.s.x) * (_se.e.x - _se.s.x) - (_st.e.y - _st.s.y) * (_se.e.y -
        _se.s.y);
}
//
//a=y2-y1,b=x1-x2,c=x2*y1-x1*y2
bool pton()
{
    a = e.y - s.y;
    b = s.x - e.x;
    c = e.x * s.y - e.y * s.x;
    return true;
}
//
//                                                      ,                        =
friend bool operator < (const Point &_Off, const Line &_Ori)
{
    return (_Ori.e.y - _Ori.s.y) * (_Off.x - _Ori.s.x)
        < (_Off.y - _Ori.s.y) * (_Ori.e.x - _Ori.s.x);
}
//
double get_angle( bool isVector = true)
{
    angle = atan2( e.y - s.y, e.x - s.x);
    if(!isVector && angle < 0)
        angle += PI;
    return angle;
}

//                      1:               2   s   ,       e
bool has(const Point &_Off, bool isSegment = false) const
{
    bool ff = sgn( xmult( s, e, _Off), 0) == 0;
    if( !isSegment) return ff;
    return ff
        && sgn(_Off.x - min(s.x, e.x), 0) >= 0 && sgn(_Off.x - max(s.x, e.x), 0) <= 0
        && sgn(_Off.y - min(s.y, e.y), 0) >= 0 && sgn(_Off.y - max(s.y, e.y), 0) <= 0;
}

//           /
double dis(const Point &_Off, bool isSegment = false)
{
    ///
    pton();
    //
    double td = (a * _Off.x + b * _Off.y + c) / sqrt(a * a + b * b);
    //
    if(isSegment)
    {
        double xp = (b * b * _Off.x - a * b * _Off.y - a * c) / ( a * a + b * b);
        double yp = (-a * b * _Off.x + a * a * _Off.y - b * c) / (a * a + b * b);
        double xb = max(s.x, e.x);
        double yb = max(s.y, e.y);
        double xs = s.x + e.x - xb;
        double ys = s.y + e.y - yb;
       if(xp > xb + eps || xp < xs - eps || yp > yb + eps || yp < ys - eps)
           td = min( getdis(_Off,s), getdis(_Off,e));
    }
    return fabs(td);
}

//
Point mirror(const Point &_Off)
{
    ///
    Point ret;
    double d = a * a + b * b;
    ret.x = (b * b * _Off.x - a * a * _Off.x - 2 * a * b * _Off.y - 2 * a * c) / d;
    ret.y = (a * a * _Off.y - b * b * _Off.y - 2 * a * b * _Off.x - 2 * b * c) / d;
    return ret;
}
//
static Line ppline(const Point &_a,const Point &_b)
{
    Line ret;
    ret.s.x = (_a.x + _b.x) / 2;
    ret.s.y = (_a.y + _b.y) / 2;
    //
    ret.a = _b.x - _a.x;
    ret.b = _b.y - _a.y;
    ret.c = (_a.y - _b.y) * ret.s.y + (_a.x - _b.x) * ret.s.x;
    //
    if(fabs(ret.a) > eps)
    {
        ret.e.y = 0.0;
        ret.e.x = - ret.c / ret.a;
        if(ret.e == ret. s)
        {
            ret.e.y = 1e10;
            ret.e.x = - (ret.c - ret.b * ret.e.y) / ret.a;
        }
    }
    else
    {
        ret.e.x = 0.0;
        ret.e.y = - ret.c / ret.b;
        if(ret.e == ret. s)
        {
            ret.e.x = 1e10;
            ret.e.y = - (ret.c - ret.a * ret.e.x) / ret.b;
        }
    }
    return ret;
}

//------------                       -------------
//             t
Line& moveLine( double t)
{
    Point of;
    of = Point( -( e.y - s.y), e.x - s.x);
    double dis = sqrt( of.x * of.x + of.y * of.y);
    of.x= of.x * t / dis, of.y = of.y * t / dis;
```

```
        s = s + of, e = e + of;
        return *this;
    }
    //
    static bool equal(const Line &_st,const Line &_se)
    {
        return _st.has( _se.e) && _se.has( _st.s);
    }
    //
    static bool parallel(const Line &_st,const Line &_se)
    {
        return sgn( _st / _se, 0) == 0;
    }
    //
    //      -1                  01
    static bool crossLPt(const Line &_st,const Line &_se, Point &ret)
    {
        if(parallel(_st,_se))
        {
            if(Line::equal(_st,_se)) return 0;
            return -1;
        }
        ret = _st.s;
        double t = ( Line(_st.s,_se.s) / _se) / ( _st / _se);
        ret.x += (_st.e.x - _st.s.x) * t;
        ret.y += (_st.e.y - _st.s.y) * t;
        return 1;
    }
    //------------                       ----------
    //
    //              [_st],     [_se]
    friend bool crossSL( Line &_st, Line &_se)
    {
        return sgn( xmult( _st.s, _se.s, _st.e) * xmult( _st.s, _st.e, _se.e), 0) >= 0;
    }

    //                      (     eps     )
    static bool isCrossSS( const Line &_st, const Line &_se)
    {
        //1.
        //2.                 0
        return
            max(_st.s.x, _st.e.x) >= min(_se.s.x, _se.e.x) &&
            max(_se.s.x, _se.e.x) >= min(_st.s.x, _st.e.x) &&
            max(_st.s.y, _st.e.y) >= min(_se.s.y, _se.e.y) &&
            max(_se.s.y, _se.e.y) >= min(_st.s.y, _st.e.y) &&
            sgn( xmult( _se.s, _st.s, _se.e) * xmult( _se.s, _se.e, _st.s), 0) >= 0 &&
            sgn( xmult( _st.s, _se.s, _st.e) * xmult( _st.s, _st.e, _se.s), 0) >= 0;
    }
};

//      graham
Point gsort;
bool gcmp( const Point &ta, const Point &tb)///

{
    double tmp = xmult( gsort, ta, tb);
    if( fabs( tmp) < eps)
        return getdis( gsort, ta) < getdis( gsort, tb);
    else if( tmp > 0)
        return 1;
    return 0;
}

class Polygon
{
public:
    const static int maxpn = 5e4+7;
    Point pt[maxpn];//
    Line dq[maxpn]; //
    int n;//


    //
    double area()
    {
        double ans = 0.0;
        for(int i = 0; i < n; i ++)
        {
            int nt = (i + 1) % n;
            ans += pt[i].x * pt[nt].y - pt[nt].x * pt[i].y;
        }
        return fabs( ans / 2.0);
    }
    //
    Point gravity()
    {
        Point ans;
        ans.x = ans.y = 0.0;
        double area = 0.0;
        for(int i = 0; i < n; i ++)
        {
            int nt = (i + 1) % n;
            double tp = pt[i].x * pt[nt].y - pt[nt].x * pt[i].y;
            area += tp;
            ans.x += tp * (pt[i].x + pt[nt].x);
            ans.y += tp * (pt[i].y + pt[nt].y);
        }
        ans.x /= 3 * area;
        ans.y /= 3 * area;
        return ans;
    }
    //                                  [        ] O (n)
    bool ahas( Point &_Off)
    {
        int ret = 0;
        double infv = 1e20;//
        Line l = Line( _Off, Point( -infv ,_Off.y));
        for(int i = 0; i < n; i ++)
        {
            Line ln = Line( pt[i], pt[(i + 1) % n]);
            if(fabs(ln.s.y - ln.e.y) > eps)
            {
                Point tp = (ln.s.y > ln.e.y)? ln.s: ln.e;
                if( ( fabs( tp.y - _Off.y) < eps && tp.x < _Off.x + eps) || Line::isCrossSS( ln, l))
                    ret++;
            }
            else if( Line::isCrossSS( ln, l))
                ret++;
        }
        return ret&1;
    }

    //                 O                 (logn)
    bool bhas( Point & p)
    {
        if( n < 3)
            return false;
        if( xmult( pt[0], p, pt[1]) > eps)
            return false;
        if( xmult( pt[0], p, pt[n-1]) < -eps)
            return false;
        int l = 2,r = n-1;
        int line = -1;
        while( l <= r)
        {
            int mid = ( l + r) >> 1;
            if( xmult( pt[0], p, pt[mid]) >= 0)
                line = mid,r = mid - 1;
            else l = mid + 1;
        }
        return xmult( pt[line-1], p, pt[line]) <= eps;
    }



    //
    Polygon split( Line &_Off)
    {
        //
        Polygon ret;
        Point spt[2];
        double tp = 0.0, np;
        bool flag = true;
        int i, pn = 0, spn = 0;
        for(i = 0; i < n; i ++)
        {
            if(flag)
                pt[pn ++] = pt[i];
            else
                ret.pt[ret.n ++] = pt[i];
            np = xmult( _Off.s, _Off.e, pt[(i + 1) % n]);
            if(tp * np < -eps)
            {
                flag = !flag;
                Line::crossLPt( _Off, Line(pt[i], pt[(i + 1) % n]), spt[spn++]);
            }
            tp = (fabs(np) > eps)?np: tp;
        }
        ret.pt[ret.n ++] = spt[0];
        ret.pt[ret.n ++] = spt[1];
        n = pn;
        return ret;
    }


    /**                             _p_n                             **/
```

```
void ConvexClosure( Point _p[], int _n)
{
    sort( _p, _p + _n);
    n = 0;
    for(int i = 0; i < _n; i++)
    {
        while( n > 1 && sgn( xmult( pt[n-2], pt[n-1], _p[i]), 0) <= 0)
            n--;
        pt[n++] = _p[i];
    }
    int _key = n;
    for(int i = _n - 2; i >= 0; i--)
    {
        while( n > _key && sgn( xmult( pt[n-2], pt[n-1], _p[i]), 0) <= 0)
            n--;
        pt[n++] = _p[i];
    }
    if(n>1)  n--;//
}
/******       graham               ********************/
/******         _p       ,       _n       ****************/

void graham( Point _p[], int _n)
{
    int cur=0;
    for(int i = 1; i < _n; i++)
        if( sgn( _p[cur].y, _p[i].y) > 0 || ( sgn( _p[cur].y, _p[i].y) == 0 && sgn( _p[cur].x,
             _p[i].x) > 0) )
            cur = i;
    swap( _p[cur], _p[0]);
    n = 0, gsort = pt[n++] = _p[0];
    if( _n <= 1) return;
    sort( _p + 1, _p+_n ,gcmp);
    pt[n++] = _p[1];
    for(int i = 2; i < _n; i++)
    {
        while(n>1 && sgn( xmult( pt[n-2], pt[n-1], _p[i]), 0) <= 0)//
                          n
            n--;
        pt[n++] = _p[i];
    }
}
//                (                                        )
//
pair<Point,Point> rotating_calipers()
{
    int i = 1 % n;
    double ret = 0.0;
    pt[n] = pt[0];
    pair<Point,Point>ans=make_pair(pt[0],pt[0]);
    for(int j = 0; j < n; j ++)
    {
        while( fabs( xmult( pt[i+1], pt[j], pt[j + 1])) > fabs( xmult( pt[i], pt[j], pt[j +
             1])) + eps)
            i = (i + 1) % n;
        //pt[i] pt [j],pt[i + 1] pt [j + 1]
        if(ret < getdis2(pt[i],pt[j])) ret = getdis2(pt[i],pt[j]), ans = make_pair(pt[i],pt[j]);
        if(ret < getdis2(pt[i+1],pt[j+1])) ret = getdis(pt[i+1],pt[j+1]), ans =
             make_pair(pt[i+1],pt[j+1]);
    }
    return ans;
}

//                (                           )
//
double rotating_calipers( Polygon &_Off)
{
    int i = 0;
    double ret = 1e10;//inf
    pt[n] = pt[0];
    _Off.pt[_Off.n] = _Off.pt[0];
    //                pt                 [0]
    while( _Off.pt[i + 1].y > _Off.pt[i].y)
        i = (i + 1) % _Off.n;
    for(int j = 0; j < n; j ++)
    {
        double tp;
        //              >,
        while((tp = xmult(_Off.pt[i + 1],pt[j], pt[j + 1]) - xmult(_Off.pt[i], pt[j], pt[j +
            1])) > eps)
            i = (i + 1) % _Off.n;
        //(pt[i],pt[i+1]) (_Off.pt[j],_Off.pt[j + 1])
        ret = min(ret, Line(pt[j], pt[j + 1]).dis(_Off.pt[i], true));
        ret = min(ret, Line(_Off.pt[i], _Off.pt[i + 1]).dis(pt[j + 1], true));
        if(tp > -eps)//                    TLE
        {
            ret = min(ret, Line(pt[j], pt[j + 1]).dis(_Off.pt[i + 1], true));
            ret = min(ret, Line(_Off.pt[i], _Off.pt[i + 1]).dis(pt[j], true));
        }
    }
    return ret;
}

//-----------          -------------
//         :O(nlog2(n))
//
//                    [1]              [ln];(
//          n          [                           ]                                      0
int judege( Line &_lx, Line &_ly, Line &_lz)
{
    Point tmp;
    Line::crossLPt(_lx,_ly,tmp);
    return sgn(xmult(_lz.s,tmp,_lz.e),0);
}
int halfPanelCross(Line L[], int ln)
{
    int i, tn, bot, top;
    for(int i = 0; i < ln; i++)
        L[i].get_angle();
    sort(L, L + ln);
    //
    for(i = tn = 1; i < ln; i ++)
        if(fabs(L[i].angle - L[i - 1].angle) > eps)
            L[tn ++] = L[i];
    ln = tn, n = 0, bot = 0, top = 1;
    dq[0] = L[0], dq[1] = L[1];
    for(i = 2; i < ln; i ++)
    {
            while(bot < top && judege(dq[top],dq[top-1],L[i]) > 0)
                top --;
            while(bot < top && judege(dq[bot],dq[bot+1],L[i]) > 0)
                bot ++;
            dq[++ top] = L[i];
        }
        while(bot < top && judege(dq[top],dq[top-1],dq[bot]) > 0)
            top --;
        while(bot < top && judege(dq[bot],dq[bot+1],dq[top]) > 0)
            bot ++;
        //
        //        if(top <= bot + 1)
        //            return 0;
        dq[++top] = dq[bot];
        for(i = bot; i < top; i ++)
            Line::crossLPt(dq[i],dq[i + 1],pt[n++]);
        return n;
    }
};


class Circle
{
public:
    Point c;//
    double r;//
    double db, de;//                                  (    0    -360)

    //-------  ---------

    //
    bool inside( Polygon &_Off)
    {
        if(_Off.ahas(c) == false)
            return false;
        for(int i = 0; i < _Off.n; i ++)
        {
            Line l = Line(_Off.pt[i], _Off.pt[(i + 1) % _Off.n]);
            if(l.dis(c, true) < r - eps)
                return false;
        }
        return true;
    }

    //
    bool has( Polygon &_Off)
    {
        for(int i = 0; i < _Off.n; i ++)
            if( getdis2(_Off.pt[i],c) > r * r - eps)
                return false;
        return true;
    }

    //-------     -------
    //                                        [_Off]
    Circle operator-(Circle &_Off) const
    {
        //
        double d2 = getdis2(c,_Off.c);
    double d = getdis(c,_Off.c);
    double ans = acos((d2 + r * r - _Off.r * _Off.r) / (2 * d * r));
    Point py = _Off.c - c;
    double oans = atan2(py.y, py.x);
    Circle res;
    res.c = c;
    res.r = r;
    res.db = oans + ans;
    res.de = oans - ans + 2 * PI;
    return res;
}
//                                        [_Off]
Circle operator+(Circle &_Off) const
{
    //
    double d2 = getdis2(c,_Off.c);
    double d = getdis(c,_Off.c);
    double ans = acos((d2 + r * r - _Off.r * _Off.r) / (2 * d * r));
    Point py = _Off.c - c;
    double oans = atan2(py.y, py.x);
    Circle res;
    res.c = c;
    res.r = r;
    res.db = oans - ans;
    res.de = oans + ans;
    return res;
}

//
//           [_Off](             ),                   (      s_Off       ,     e     )
pair<Line, Line> tangent( Point &_Off)
{
    double d = getdis(c,_Off);
    //
    double angp = acos(r / d), ango = atan2(_Off.y - c.y, _Off.x - c.x);
    Point pl = Point(c.x + r * cos(ango + angp), c.y + r * sin(ango + angp)),
        pr = Point(c.x + r * cos(ango - angp), c.y + r * sin(ango - angp));
    return make_pair(Line(_Off, pl), Line(_Off, pr));
}

//
//              [_Off](       )
pair<Point, Point> cross(Line _Off)
{
    _Off.pton();
    //
    double td = fabs(_Off.a * c.x + _Off.b * c.y + _Off.c) / sqrt(_Off.a * _Off.a + _Off.b *
        _Off.b);

    //
    double xp = (_Off.b * _Off.b * c.x - _Off.a * _Off.b * c.y - _Off.a * _Off.c) / ( _Off.a *
        _Off.a + _Off.b * _Off.b);
    double yp = (- _Off.a * _Off.b * c.x + _Off.a * _Off.a * c.y - _Off.b * _Off.c) / (_Off.a *
        _Off.a + _Off.b * _Off.b);

    double ango = atan2(yp - c.y, xp - c.x);
    double angp = acos(td / r);

    return make_pair(Point(c.x + r * cos(ango + angp), c.y + r * sin(ango + angp)),
            Point(c.x + r * cos(ango - angp), c.y + r * sin(ango - angp)));
    }
};

class triangle
{
public:
    Point a, b, c;//
    triangle(){}
    triangle(Point a, Point b, Point c): a(a), b(b), c(c){}

    //
    double area()
    {
        return fabs( xmult(a, b, c)) / 2.0;
    }

    //
    //
    Point circumcenter()
    {
        double pa = a.x * a.x + a.y * a.y;
        double pb = b.x * b.x + b.y * b.y;
        double pc = c.x * c.x + c.y * c.y;
        double ta = pa * ( b.y - c.y) - pb * ( a.y - c.y) + pc * ( a.y - b.y);
        double tb = -pa * ( b.x - c.x) + pb * ( a.x - c.x) - pc * ( a.x - b.x);
        double tc = a.x * ( b.y - c.y) - b.x * ( a.y - c.y) + c.x * ( a.y - b.y);
        return Point( ta / 2.0 / tc, tb / 2.0 / tc);
    }

    //
    //
    Point incenter()
    {
        Line u, v;
        double m, n;
        u.s = a;
        m = atan2(b.y - a.y, b.x - a.x);
        n = atan2(c.y - a.y, c.x - a.x);
        u.e.x = u.s.x + cos((m + n) / 2);
        u.e.y = u.s.y + sin((m + n) / 2);
        v.s = b;
        m = atan2(a.y - b.y, a.x - b.x);
        n = atan2(c.y - b.y, c.x - b.x);
        v.e.x = v.s.x + cos((m + n) / 2);
        v.e.y = v.s.y + sin((m + n) / 2);
        Point ret;
        Line::crossLPt(u,v,ret);
        return ret;
    }

    //
    //
    Point perpencenter()
    {
        Line u,v;
        u.s = c;
        u.e.x = u.s.x - a.y + b.y;
        u.e.y = u.s.y + a.x - b.x;
        v.s = b;
        v.e.x = v.s.x - a.y + c.y;
        v.e.y = v.s.y + a.x - c.x;
        Point ret;
        Line::crossLPt(u,v,ret);
        return ret;
    }

    //
    //
    //
    //
    Point barycenter()
    {
        Line u,v;
        u.s.x = (a.x + b.x) / 2;
        u.s.y = (a.y + b.y) / 2;
        u.e = c;
        v.s.x = (a.x + c.x) / 2;
        v.s.y = (a.y + c.y) / 2;
        v.e = b;
        Point ret;
        Line::crossLPt(u,v,ret);
        return ret;
    }

    //
    //
    Point fermentPoint()
    {
        Point u, v;
        double step = fabs(a.x) + fabs(a.y) + fabs(b.x) + fabs(b.y) + fabs(c.x) + fabs(c.y);
        int i, j, k;
        u.x = (a.x + b.x + c.x) / 3;
        u.y = (a.y + b.y + c.y) / 3;
        while (step > eps)
        {
            for (k = 0; k < 10; step /= 2, k ++)
            {
                for (i = -1; i <= 1; i ++)
                {
                    for (j =- 1; j <= 1; j ++)
                    {
                        v.x = u.x + step * i;
                        v.y = u.y + step * j;
                        if (getdis(u,a) + getdis(u,b) + getdis(u,c) > getdis(v,a) + getdis(v,b) +
                            getdis(v,c))
                            u = v;
                    }
                }
            }
        }
        return u;
    }
};

int main(void)
{
    return 0;
}
```

4 Math

4.1 BigNum

```
#include<iostream>
#include<string>
#include<cstdio>
#include<cstring>
#include<cmath>
#include<cstdlib>
#include<vector>
#include<iomanip>
#include<algorithm>
using namespace std;

#define MAXN 9999
#define MAXSIZE 10
#define DLEN 4

class BigNum
{
public:
    int a[500];   //
    int len;      //
    BigNum(){ len = 1;memset(a,0,sizeof(a)); } //
    BigNum(const int);     //           int
    BigNum(const char*);   //
    BigNum(const BigNum &); //
    BigNum &operator=(const BigNum &); //

    friend istream& operator>>(istream&, BigNum&); //
    friend ostream& operator<<(ostream&, BigNum&); //

    BigNum operator+(const BigNum &) const; //
    BigNum operator-(const BigNum &) const; //
    BigNum operator*(const BigNum &) const; //
    BigNum operator/(const int &) const;  //

    BigNum operator^(const int &) const; //      n
    int   operator%(const int &) const;  //               int
    bool  operator>(const BigNum & T)const; //
    bool  operator>(const int & t)const;   //              int

    void print();     //
};
BigNum::BigNum(const int b)  //           int
{
    int c,d = b;
    len = 0;
    memset(a,0,sizeof(a));
    while(d > MAXN)
    {
        c = d - (d / (MAXN + 1)) * (MAXN + 1);
        d = d / (MAXN + 1);
        a[len++] = c;
    }
    a[len++] = d;
}
BigNum::BigNum(const char*s)  //
{
    int t,k,index,l,i;
    memset(a,0,sizeof(a));
    l=strlen(s);
    len=l/DLEN;
    if(l%DLEN)
        len++;
    index=0;
    for(i=l-1;i>=0;i-=DLEN)
    {
        t=0;
        k=i-DLEN+1;
        if(k<0)
            k=0;
        for(int j=k;j<=i;j++)
            t=t*10+s[j]-'0';
        a[index++]=t;
    }
}
BigNum::BigNum(const BigNum & T) : len(T.len) //
{
    int i;
    memset(a,0,sizeof(a));
    for(i = 0 ; i < len ; i++)
        a[i] = T.a[i];
}
BigNum & BigNum::operator=(const BigNum & n) //
{
    int i;
    len = n.len;
    memset(a,0,sizeof(a));
    for(i = 0 ; i < len ; i++)
        a[i] = n.a[i];
    return *this;
}
istream& operator>>(istream & in, BigNum & b) //
{
    char ch[MAXSIZE*4];
    int i = -1;
    in>>ch;
    int l=strlen(ch);
    int count=0,sum=0;
    for(i=l-1;i>=0;)
    {
        sum = 0;
        int t=1;
        for(int j=0;j<4&&i>=0;j++,i--,t*=10)
        {
            sum+=(ch[i]-'0')*t;
        }
        b.a[count]=sum;
        count++;
    }
    b.len =count++;
    return in;

}
ostream& operator<<(ostream& out, BigNum& b) //
{
    int i;
    cout << b.a[b.len - 1];
    for(i = b.len - 2 ; i >= 0 ; i--)
    {
        cout.width(DLEN);
        cout.fill('0');
        cout << b.a[i];
    }
    return out;
}

BigNum BigNum::operator+(const BigNum & T) const //
{
    BigNum t(*this);
    int i,big;    //
    big = T.len > len ? T.len : len;
    for(i = 0 ; i < big ; i++)
    {
        t.a[i] +=T.a[i];
        if(t.a[i] > MAXN)
        {
            t.a[i + 1]++;
            t.a[i] -=MAXN+1;
        }
    }
    if(t.a[big] != 0)
        t.len = big + 1;
    else
        t.len = big;
    return t;
}
BigNum BigNum::operator-(const BigNum & T) const //
{
    int i,j,big;
    bool flag;
    BigNum t1,t2;
    if(*this>T)
    {
        t1=*this;
        t2=T;
        flag=0;
    }
    else
    {
        t1=T;
        t2=*this;
        flag=1;
    }
    big=t1.len;
    for(i = 0 ; i < big ; i++)
    {
        if(t1.a[i] < t2.a[i])
        {
            j = i + 1;
            while(t1.a[j] == 0)
                j++;
            t1.a[j--]--;
            while(j > i)
                t1.a[j--] += MAXN;
            t1.a[i] += MAXN + 1 - t2.a[i];
        }
        else
            t1.a[i] -= t2.a[i];
    }
    t1.len = big;
    while(t1.a[t1.len - 1] == 0 && t1.len > 1)
    {
        t1.len--;
        big--;
    }
    if(flag)
        t1.a[big-1]=0-t1.a[big-1];
    return t1;
}

BigNum BigNum::operator*(const BigNum & T) const //
{
    BigNum ret;
    int i,j,up;
    int temp,temp1;
    for(i = 0 ; i < len ; i++)
    {
        up = 0;
        for(j = 0 ; j < T.len ; j++)
        {
            temp = a[i] * T.a[j] + ret.a[i + j] + up;
            if(temp > MAXN)
            {
                temp1 = temp - temp / (MAXN + 1) * (MAXN + 1);
                up = temp / (MAXN + 1);
                ret.a[i + j] = temp1;
            }
            else
            {
                up = 0;
                ret.a[i + j] = temp;
            }
        }
        if(up != 0)
            ret.a[i + j] = up;
    }
    ret.len = i + j;
    while(ret.a[ret.len - 1] == 0 && ret.len > 1)
        ret.len--;
    return ret;
}
BigNum BigNum::operator/(const int & b) const //
{
    BigNum ret;
    int i,down = 0;
    for(i = len - 1 ; i >= 0 ; i--)
    {
        ret.a[i] = (a[i] + down * (MAXN + 1)) / b;
        down = a[i] + down * (MAXN + 1) - ret.a[i] * b;
    }
    ret.len = len;
    while(ret.a[ret.len - 1] == 0 && ret.len > 1)
        ret.len--;
    return ret;
}
int BigNum::operator %(const int & b) const //             int
{
    int i,d=0;
    for (i = len-1; i>=0; i--)
    {
        d = ((d * (MAXN+1))% b + a[i])% b;
    }
    return d;
}
BigNum BigNum::operator^(const int & n) const //     n
{
    BigNum t,ret(1);
    int i;
    if(n<0)
        exit(-1);
    if(n==0)
        return 1;
    if(n==1)
        return *this;
    int m=n;
    while(m>1)
    {
        t=*this;
        for( i=1;i<<1<=m;i<<=1)
        {
            t=t*t;
        }
        m-=i;
        ret=ret*t;
        if(m==1)
            ret=ret*(*this);
    }
    return ret;
}
bool BigNum::operator>(const BigNum & T) const //
{
    int ln;
    if(len > T.len)
        return true;
    else if(len == T.len)
    {
        ln = len - 1;
        while(a[ln] == T.a[ln] && ln >= 0)
            ln--;
        if(ln >= 0 && a[ln] > T.a[ln])
            return true;
        else
            return false;
    }
    else
        return false;
```

```
}
bool BigNum::operator >(const int & t) const //            int
{
    BigNum b(t);
    return *this>b;
}

void BigNum::print()  //
{
    int i;
    cout << a[len - 1];
    for(i = len - 2 ; i >= 0 ; i--)
    {
        cout.width(DLEN);
        cout.fill('0');
        cout << a[i];
    }
    cout << endl;
}
int main(void)
{
    BigNum x=BigNum(1);
    for(int i=2;i<=100;i++)
        x=x*BigNum(i);
    int sum=0;
    x.print();
    for(int i=0;i<500;i++)
    {
        while(x.a[i]>0)
        {
            sum+=x.a[i]%10;
            x.a[i]/=10;
        }
    }
    printf("%d\n",sum);
    return 0;
}
```

4.2 Determinant

```
#include<bits/stdc++.h>
#define MAXN 505
using namespace std;
typedef vector<int> vec;
typedef vector<vec> mat;
int n;
int det_mod(mat A,int M)
{
        int n=A.size();
        for(int i=0;i<n;i++)
                for(int j=0;j<n;j++)
                        A[i][j]%=M;
        int ans=1;
        for(int i=0;i<n;i++)
        {
                for(int j=i+1;j<n;j++)
                {
                    while(A[j][i]!=0)
                    {
                        int t=A[i][i]/A[j][i];
                        for(int k=0;k<n;k++)
                        {
                            A[i][k]=A[i][k]-A[j][k]*t;
                            swap(A[i][k],A[j][k]);
                        }
                        ans=-ans;
                    }
                    if(A[i][i]==0) return 0;
                }
                ans=ans*A[i][i];
        }
        return (ans%M+M)%M;
}
int main()
{
        scanf("%d",&n);
        mat A(n,vec(n));
        for(int i=0;i<n;i++)
                for(int j=0;j<n;j++)
                        scanf("%d",&A[i][j]);
        printf("%d\n",det_mod(A,3));
        return 0;
}
```

4.3 Eratonthenes Sieve

```
#include<bits/stdc++.h>
#define MAXN 100005
#define MOD 1000000007
#define INF 1000000000
using namespace std;
typedef long long ll;
int prime[MAXN];
bool is_prime[MAXN];
int sieve(int n)
{
    int p=0;
    for(int i=0;i<=n;i++) is_prime[i]=true;
    is_prime[0]=is_prime[1]=false;
    for(int i=2;i<=n;i++)
    {
        if(is_prime[i])
        {
            prime[p++]=i;
            for(int j=2*i;j<=n;j+=i) is_prime[j]=false;
        }
    }
    return p;
}
```

4.4 Euler Sieve

```
#include<bits/stdc++.h>
#define MAXN 100005
#define MOD 1000000007
#define INF 1000000000
using namespace std;
typedef long long ll;
int prime[MAXN],phi[MAXN],miu[MAXN];
bool is_prime[MAXN];
int sieve(int n)
{
    int p=0;
    for(int i=0;i<=n;i++) is_prime[i]=true;
    is_prime[0]=is_prime[1]=false;
    for(int i=2;i<=n;i++)
    {
        if(is_prime[i]) prime[p++]=i;
        for(int j=0;j<p;j++)
        {
            if(prime[j]*i>n) break;
            is_prime[prime[j]*i]=false;
            if(i%prime[j]==0) break;
        }
    }
    return p;
}
void genphi(int n)
{
    int p=0;
    memset(phi,0,sizeof(phi));
    phi[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(is_prime[i]) {p++; phi[i]=i-1;}
        for(int j=0;j<p;j++)
        {
            if(prime[j]*i>n) break;
            phi[i*prime[j]]=phi[i]*(i%prime[j]?prime[j]-1:prime[j]);
            if(i%prime[j]==0) break;
        }
    }
}
void genmiu(int n)
{
    int p=0;
    memset(miu,0,sizeof(miu));
    miu[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(is_prime[i]) {p++; miu[i]=-1;}
        for(int j=0;j<p;j++)
        {
            if(prime[j]*i>n) break;
            miu[i*prime[j]]=i%prime[j]?-miu[i]:0;
            if(i%prime[j]==0) break;
        }
    }
}
int main()
```

```
{
    sieve(100000);
    genphi(100000);
    genmiu(100000);
    for(int i=1;i<=10;i++)
        printf("%d\n",miu[i]);
    return 0;
}
```

4.5 Miller-Rabin

```
#include<bits/stdc++.h>
using namespace std;
int pow_mod(int a,int i,int n)
{
    if(i==0) return 1%n;
    int temp=pow_mod(a,i>>1,n);
      temp=temp*temp%n;
    if(i&1) temp=(long long) temp*a%n;
    return temp;
}
bool test(int n,int a,int d)
{
    if(n==2) return true;
    if(n==a) return true;
    if((n&1)==0) return false;
    while(!(d&1)) d=d>>1;
    int t=pow_mod(a,d,n);
    while((d!=n-1)&&(t!=1)&&(t!=n-1))
    {
        t=(long long)t*t%n;
        d=d<<1;
    }
    return(t==n-1||(d&1)==1);
}
bool isPrime(int n)
{
    if(n<2) return false;
    int a[]={2,3,61};
    for(int i=0;i<=2;++i) if(!test(n,a[i],n-1)) return false;
    return true;
}
int main()
{
    return 0;
}
```

4.6 Matrix Operations

```
#include<bits/stdc++.h>
#define MAXN 1000
using namespace std;
typedef vector<double> vec;
typedef vector<vec> mat;
typedef long long ll;
```

```
int n;
mat mul(mat A,mat B)
{
    mat C(A.size(),vec(B[0].size()));
    for(int i=0;i<A.size();i++)
        for(int k=0;k<B.size();k++)
            for(int j=0;j<B[0].size();j++)
                C[i][j]=(C[i][j]+A[i][k]*B[k][j]);
    return C;
}
mat pow(mat A,ll n)
{
    mat B(A.size(),vec(A.size()));
    for(int i=0;i<A.size();i++)
        B[i][i]=1;
    while(n>0)
    {
        if(n&1) B=mul(B,A);
        A=mul(A,A);
        n>>=1;
    }
    return B;
}
int main()
{
    scanf("%d",&n);
    mat A(n,vec(n));
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            scanf("%lf",&A[i][j]);
    mat L(n,vec(n));
    mat U(n,vec(n));
    for(int i=1;i<n;i++)
        for(int j=0;j<i;j++)
            U[i][j]=0;
    for(int i=0;i<n;i++)
        L[i][i]=1;
    for(int i=0;i<n;i++)
        for(int j=i+1;j<n;j++)
            L[i][j]=0;
    for(int i=0;i<n;i++)
    {
        U[i][i]=A[i][i];
        for(int j=i+1;j<n;j++)
        {
            L[j][i]=A[j][i]/U[i][i];
            U[i][j]=A[i][j];
        }
        for(int j=i+1;j<n;j++)
            for(int k=i+1;k<n;k++)
                A[j][k]=A[j][k]-L[j][i]*U[i][k];
    }
    printf("L=\n");
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
            printf("%6lf ",L[i][j]);
        printf("\n");
    }
    printf("U=\n");
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
            printf("%6lf ",U[i][j]);
        printf("\n");
    }
}
```

4.7 LU

```
#include<bits/stdc++.h>
#define MAXN 1000
using namespace std;
typedef vector<double> vec;
typedef vector<vec> mat;
typedef long long ll;
int n;
mat mul(mat A,mat B)
{
    mat C(A.size(),vec(B[0].size()));
    for(int i=0;i<A.size();i++)
        for(int k=0;k<B.size();k++)
            for(int j=0;j<B[0].size();j++)
                C[i][j]=(C[i][j]+A[i][k]*B[k][j]);
    return C;
}
mat pow(mat A,ll n)
{
    mat B(A.size(),vec(A.size()));
    for(int i=0;i<A.size();i++)
        B[i][i]=1;
    while(n>0)
    {
        if(n&1) B=mul(B,A);
        A=mul(A,A);
        n>>=1;
    }
    return B;
}
int main()
{
    scanf("%d",&n);
    mat A(n,vec(n));
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            scanf("%lf",&A[i][j]);
    mat L(n,vec(n));
    mat U(n,vec(n));
    for(int i=1;i<n;i++)
        for(int j=0;j<i;j++)
            U[i][j]=0;
    for(int i=0;i<n;i++)
        L[i][i]=1;
    for(int i=0;i<n;i++)
        for(int j=i+1;j<n;j++)
            L[i][j]=0;
    for(int i=0;i<n;i++)
    {
        U[i][i]=A[i][i];
        for(int j=i+1;j<n;j++)
        {
            L[j][i]=A[j][i]/U[i][i];
            U[i][j]=A[i][j];
        }
        for(int j=i+1;j<n;j++)
            for(int k=i+1;k<n;k++)
                A[j][k]=A[j][k]-L[j][i]*U[i][k];
    }
    printf("L=\n");
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
            printf("%6lf ",L[i][j]);
        printf("\n");
    }
    printf("U=\n");
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
            printf("%6lf ",U[i][j]);
        printf("\n");
    }
}
```

4.8 Gauss-Jordan

```
#include<bits/stdc++.h>
#define MAXN 105
using namespace std;
const double eps=1e-8;
typedef vector<double> vec;
typedef vector<vec> mat;
int sz;
vec gauss_jordan(const mat& A, const vec& b)
{
    int n=A.size();
    mat B(n,vec(n+1));
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            B[i][j]=A[i][j];

    for(int i=0;i<n;i++) B[i][n]=b[i];
    for(int i=0;i<n;i++)
    {
        int pivot=i;
        for(int j=i;j<n;j++)
            if(abs(B[j][i])>abs(B[pivot][i])) pivot=j;
        swap(B[i],B[pivot]);
        if(abs(B[i][i])<eps) return vec();
        for(int j=i+1;j<=n;j++) B[i][j]/=B[i][i];
        for(int j=0;j<n;j++)
        {
            if(i!=j)
            {
                for(int k=i+1;k<=n;k++)
                    B[j][k]-=B[j][i]*B[i][k];
            }
        }
    }
    vec x(n);
    for(int i=0;i<n;i++)
        x[i]=B[i][n];
    return x;
}
int main()
{
    scanf("%d",&sz);
    mat A(sz,vec(sz));
    vec b(sz);
    for(int i=0;i<sz;i++)
        for(int j=0;j<sz;j++)
            A[i][j]=0;
    for(int i=0;i<sz;i++)
    {
        double x;
        int cnt=0;
        while(scanf("%lf",&x)==1)
        {
            if(x==-1) break;
            A[x-1][i]=1.0;
        }
    }
    for(int i=0;i<sz;i++)
        b[i]=1.0;
    vec res=gauss_jordan(A,b);
    if(res==vec()) printf("No solution\n");
    else
    {
        for(int i=0;i<sz;i++)
            if(res[i]>0) printf("%d ",i+1);
        printf("\n");
    }
    return 0;
}
```

4.9 Mod-inverse and Mod-combination

```
#include<bits/stdc++.h>
#define MAXN 100000
#define MAXP 1005
using namespace std;
int gcd(int a,int b)
{
    if(b==0) return a;
    return gcd(b,a%b);
}
int extgcd(int a,int b,int &x,int &y)
{
    int d=a;
    if(b!=0)
    {
        d=extgcd(b,a%b,y,x);
        y-=(a/b)*x;
    }
    else
    {
        x=1;
        y=0;
    }
    return d;
}
int mod_inverse(int a,int m)
{
    int x,y;
    extgcd(a,m,x,y);
    return (m+x%m)%m;
}
int fact[MAXP];
int mod_fact(int n,int p,int &e)
{
    e=0;
    if(n==0) return 1;
    int res=mod_fact(n/p,p,e);
    e+=n/p;
    if(n/p%2!=0) return res*(p-fact[n%p])%p;
    return res*fact[n%p]%p;
}
int mod_comb(int n,int k,int p)
{
    if(n<0||k<0||n<k) return 0;
    int e1,e2,e3;
    int a1=mod_fact(n,p,e1),a2=mod_fact(k,p,e2),a3=mod_fact(n-k,p,e3);
    if(e1>e2+e3) return 0;
    return a1*mod_inverse(a2*a3%p,p)%p;
}
int main()
{
    printf("%d\n",mod_inverse(22,31));
    return 0;
}
```

4.10 Primitive Root

```
#include<cstdio>
#include<cmath>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<algorithm>
#include<vector>
#include<queue>
#include<deque>
#include<stack>
#include<map>
#define MAXN 1005000
using namespace std;
typedef long long ll;
```

```
vector<ll> a;
ll pow_mod(ll a,ll i,ll mod)
{
    if(i==0) return 1;
     ll s=1;
    while(i>0)
     {
         if(i&1) s=(s*a)%mod;
         a=(a*a)%mod;
         i>>=1;
     }
     return s;
}
bool g_test(ll g,ll p)
{
    for(ll i=0;i<a.size();i++)
        if(pow_mod(g,(p-1)/a[i],p)==1)
            return 0;
    return 1;
}
ll primitive_root(ll p)
{
    ll tmp=p-1;
    for(ll i=2;i<=tmp/i;i++)
        if(tmp%i==0)
        {
            a.push_back(i);
            while(tmp%i==0)
                tmp/=i;
        }
    if(tmp!=1)
    {
        a.push_back(tmp);
    }
    ll g=1;
    while(true)
    {
        if(g_test(g,p))
            return g;
        ++g;
    }
}
int main()
{
    ll n;
    while(scanf("%lld",&n)==1)
        printf("%lld\n",primitive_root(n));
    return 0;
}
```

4.11 Pell's equation

```
#include<bits/stdc++.h>
#define MAXN 10005
#define F first
#define S second
using namespace std;
```

```
typedef pair<int,int> P;
P Pell(int N)
{
        int p0=0,p1=1,q0=1,q1=0;
        int a0=(int)sqrt(N),a1=a0,a2=a0;
        if(a0*a0==N) return P(-1,-1);
        int g1=0,h1=1;
        while(true)
        {
                int g2=-g1+a1*h1;
                int h2=(N-g2*g2)/h1;
                a2=(g2+a0)/h2;
                int p2=a1*p1+p0;
                int q2=a1*q1+q0;
                if(p2*p2-N*q2*q2==1) return P(p2,q2);
                a1=a2;g1=g2;h1=h2;p0=p1;p1=p2;q0=q1;q1=q2;
        }
}
int main()
{
        int n;
        while(scanf("%d",&n)==1)
        {
                P p=Pell(n);
                printf("%d %d\n",p.F,p.S);
        }
        return 0;
}
```

4.12 Linear Basis

```
#include<bits/stdc++.h>
#define MAXN 1000
using namespace std;
int p[63],a[MAXN];
int n;
int cal()
{
    for(int i=1;i<=n;i++)
    {
        for(int j=62;j>=0;j--)
        {
            if(!p[j]) {p[j]=a[i]; break;}
            else a[i]^=p[j];
        }
    }
    for(int j=0;j<=62;j++) if(p[j]) r++;
    return r;
}
```

4.13 Linear Congruence

```
#include<bits/stdc++.h>
#define MAXN 10000
using namespace std;
```

```
pair<int,int> linear_congruence(const vector<int>&A, const vector<int>&B, const vector<int>&M)
{
    int x=0,m=1;
    for(int i=0;i<A.size();i++)
    {
        int a=A[i]*m,b=B[i]-A[i]*x,d=gcd(M[i],a);
        if(b%d!=0) return make_pair(0,-1);
        int t=b/d*mod_inverse(a/d,M[i]/d)%(M[i]/d);
        x=x+m*t;
        m*=M[i]/d;
    }
    return make_pair(x%m,m);
}
```

4.14 Fast Fourier Transformation

```
#include <bits/stdc++.h>
#define MAXN 400005
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const double PI=acos(-1.0);
namespace fft
{
    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<int> multiply(vector<int> &a, vector<int> &b)
    {
        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<sz;i++)
        {
            int x=(i<(int)a.size()?a[i]:0);
            int y=(i<(int)b.size()?b[i]:0);
            fa[i]=num(x,y);
        }
        fft(fa,sz);
        num r(0,-0.25/sz);
        for(int i=0;i<=(sz>>1);i++)
        {
            int 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<int> res(need);
        for(int i=0;i<need;i++) res[i]=fa[i].x+0.5;
        return res;
```

```
}

vector<int> multiply_mod(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;
    }
```

```cpp
        return res;
    }
    vector<int> square_mod(vector<int> &a,int m)
    {
        return multiply_mod(a,a,m,1);
    }
};
string s1,s2;
int main()
{
    cin>>s1;
    cin>>s2;
    int len1=(int)s1.size();
    vector<int> v1(len1);
    for(int i=0;i<len1;i++)
        v1[i]=(int)(s1[len1-1-i]-'0');
    int len2=(int)s2.size();
    vector<int> v2(len2);
    for(int i=0;i<len2;i++)
        v2[i]=(int)(s2[len2-1-i]-'0');
    vector<int> ans;
    ans=fft::multiply(v1,v2);
    int carry=0;
    for(int i=0;i<(int)ans.size();i++)
    {
        carry+=ans[i];
        ans[i]=carry%10;
        carry/=10;
    }
    while(carry>0)
    {
        ans.push_back(carry%10);
        carry/=10;
    }
    while((int)ans.size()>1&&ans.back()==0) ans.pop_back();
    for(int i=(int)ans.size()-1;i>=0;i--)
        printf("%d",ans[i]);
    return 0;
}
```

4.15 Fast Number Theoretic Transformation

```cpp
#include<bits/stdc++.h>
#define MAXN 100005
#define MOD 998244353
#define INF 1000000000
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int g=3;
int two[32];
int pow_mod(int a,int i)
{
    if(i==0) return 1;
    int s=1;
    while(i>0)
     {
         if(i&1) s=(1LL*s*a)%MOD;
         a=(1LL*a*a)%MOD;
         i>>=1;
     }
     return s;
}
int rev(int x,int r)
{
    int ans=0;
    for(int i=0;i<r;i++)
        if(x&(1<<i)) ans+=1<<(r-i-1);
    return ans;
}
void ntt(int n,int A[],int on)
{
    int r=0,cnt=0,t=n;
    while(t>1) {cnt++; t/=2;}
    for(;;r++) if((1<<r)==n) break;
    for(int i=0;i<n;i++)
    {
        int tmp=rev(i,r);
        if(i<tmp) swap(A[i],A[tmp]);
    }
    for(int s=1;s<=r;s++)
    {
        int m=1<<s;
        int wn=pow_mod(g,(MOD-1)/m);
        for(int k=0;k<n;k+=m)
        {
            int w=1;
            for(int j=0;j<m/2;j++)
            {
                int t,u;
                t=1LL*w*A[k+j+m/2]%MOD;
                u=A[k+j];
                A[k+j]=(u+t);
                if(A[k+j]>=MOD) A[k+j]-=MOD;
                A[k+j+m/2]=u+MOD-t;
                if(A[k+j+m/2]>=MOD) A[k+j+m/2]-=MOD;
                w=1LL*w*wn%MOD;
            }
        }
    }
    if(on==-1)
    {
        for(int i=1;i<n/2;i++)
            swap(A[i],A[n-i]);
        for(int i=0;i<n;i++)
            A[i]=1LL*A[i]*two[cnt]%MOD;
    }
}
int A[MAXN],B[MAXN],ans[MAXN];
int main()
{
    int n,m;
    for(int i=1;i<=30;i++)
        two[i]=pow_mod(1<<i,MOD-2);
    string s1;
    string s2;
    while(cin>>s1>>s2)
    {
        n=s1.size();
        m=s2.size();
        memset(A,0,sizeof(A));
        memset(B,0,sizeof(B));
        for(int i=n-1; i>=0 ; i--)
            A[i]=s1[n-i-1]-'0';
        for(int i=m-1; i>=0; i--)
            B[i]=s2[m-i-1]-'0';
        int tmp=1;
        while(tmp<max(n,m))
            tmp*=2;
        n=tmp;
        ntt(2*n,A,1);
        ntt(2*n,B,1);
        for(int i=0; i<2*n; i++)
            A[i]=1LL*A[i]*B[i]%MOD;
        ntt(2*n,A,-1);
        memset(ans,0,sizeof ans);
        for(int i=0;i<2*n;i++)
        {
            ans[i]+=A[i];
            if(ans[i]>=10)
            {
                ans[i+1]+=ans[i]/10;
                ans[i]%=10;
            }
        }
        int e=0;
        for(int i=2*n-1;i>=0;i--)
        {
            if(ans[i])
            {
                e=i;
                break;
            }
        }
        for(int i=e;i>=0;i--)
        {
            printf("%d",ans[i]);
        }
        printf("\n");
    }
    return 0;
}
```

4.16 Fast Walsh-Hadamard Transformation

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define REV 500000004
#define F first
```

```
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
void FWT(int a[],int n)
{
    for(int d=1;d<n;d<<=1)
        for(int m=d<<1,i=0;i<n;i+=m)
            for(int j=0;j<d;j++)
            {
                int x=a[i+j],y=a[i+j+d];
                a[i+j]=(x+y)%MOD,a[i+j+d]=(x-y+MOD)%MOD;
                //xor:a[i+j]=x+y,a[i+j+d]=(x-y+MOD)%MOD;
                //and:a[i+j]=x+y;
                //or:a[i+j+d]=x+y;
            }
}

void UFWT(int a[],int n)
{
    for(int d=1;d<n;d<<=1)
        for(int m=d<<1,i=0;i<n;i+=m)
            for(int j=0;j<d;j++)
            {
                int x=a[i+j],y=a[i+j+d];
                a[i+j]=1LL*(x+y)*REV%MOD,a[i+j+d]=(1LL*(x-y)*REV%MOD+MOD)%MOD;
                //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
                //and:a[i+j]=x-y;
                //or:a[i+j+d]=y-x;
            }
}
void solve(int a[],int b[],int n)
{
    FWT(a,n);
    FWT(b,n);
    for(int i=0;i<n;i++) a[i]=1LL*a[i]*b[i]%MOD;
    UFWT(a,n);
}
int main()
{
    return 0;
}
```

4.17 Polynomial Inverse

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 998244353
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int g=3;
int two[31];
int dbit(int x)
```

```
{
    while(x!=(x&-x)) x+=(x&-x);
    return x;
}
int pow_mod(int a,int i)
{
    if(i==0) return 1;
    int s=1;
    while(i>0)
     {
         if(i&1) s=(1LL*s*a)%MOD;
         a=(1LL*a*a)%MOD;
         i>>=1;
     }
     return s;
}
int rev(int x,int r)
{
    int ans=0;
    for(int i=0;i<r;i++)
        if(x&(1<<i)) ans+=1<<(r-i-1);
    return ans;
}
void ntt(int n,int A[],int on)
{
    int r=0,cnt=0,t=n;
    while(t>1) {cnt++; t/=2;}
    for(;;r++) if((1<<r)==n) break;
    for(int i=0;i<n;i++)
    {
        int tmp=rev(i,r);
        if(i<tmp) swap(A[i],A[tmp]);
    }
    for(int s=1;s<=r;s++)
    {
        int m=1<<s;
        int wn=pow_mod(g,(MOD-1)/m);
        for(int k=0;k<n;k+=m)
        {
            int w=1;
            for(int j=0;j<m/2;j++)
            {
                int t,u;
                t=1LL*w*A[k+j+m/2]%MOD;
                u=A[k+j];
                A[k+j]=(u+t);
                if(A[k+j]>=MOD) A[k+j]-=MOD;
                A[k+j+m/2]=u+MOD-t;
                if(A[k+j+m/2]>=MOD) A[k+j+m/2]-=MOD;
                w=1LL*w*wn%MOD;
            }
        }
    }
    if(on==-1)
    {
        for(int i=1;i<n/2;i++)
            swap(A[i],A[n-i]);
        for(int i=0;i<n;i++)
            A[i]=1LL*A[i]*two[cnt]%MOD;
    }
}
int n,A[MAXN],B[MAXN],C[MAXN];
void find_inverse(int A[],int n)
{
        if(n==1) {B[0]=pow_mod(A[0],MOD-2); return;}
        find_inverse(A,(n+1)/2);
        int len=dbit(n)*2;
        for(int i=0;i<n;i++)
                C[i]=A[i];
        for(int i=n;i<len;i++) C[i]=0;
        ntt(len,C,1);ntt(len,B,1);
        for(int i=0;i<len;i++)
                C[i]=1LL*B[i]*B[i]%MOD*C[i]%MOD;
        ntt(len,C,-1);ntt(len,B,-1);
        for(int i=0;i<n;i++)
                B[i]=((2*B[i]-C[i])%MOD+MOD)%MOD;
}
int main()
{
        for(int i=1;i<=30;i++)
        two[i]=pow_mod(1<<i,MOD-2);
    for(int i=0;i<2;i++)
        A[i]=1;
    find_inverse(A,3);
    for(int i=0;i<4;i++) printf("%d ",B[i]);
    return 0;
}
```

4.18 Polynomial Square Root

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 998244353
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int g=3;
int two[31];
int dbit(int x)
{
    while(x!=(x&-x)) x+=(x&-x);
    return x;
}
int pow_mod(int a,int i)
{
    if(i==0) return 1;
    int s=1;
    while(i>0)
     {
         if(i&1) s=(1LL*s*a)%MOD;
         a=(1LL*a*a)%MOD;
         i>>=1;
     }
    return s;
}
int rev(int x,int r)
{
    int ans=0;
    for(int i=0;i<r;i++)
        if(x&(1<<i)) ans+=1<<(r-i-1);
    return ans;
}
void ntt(int n,int A[],int on)
{
    int r=0,cnt=0,t=n;
    while(t>1) {cnt++; t/=2;}
    for(;;r++) if((1<<r)==n) break;
    for(int i=0;i<n;i++)
    {
        int tmp=rev(i,r);
        if(i<tmp) swap(A[i],A[tmp]);
    }
    for(int s=1;s<=r;s++)
    {
        int m=1<<s;
        int wn=pow_mod(g,(MOD-1)/m);
        for(int k=0;k<n;k+=m)
        {
            int w=1;
            for(int j=0;j<m/2;j++)
            {
                int t,u;
                t=1LL*w*A[k+j+m/2]%MOD;
                u=A[k+j];
                A[k+j]=(u+t);
                if(A[k+j]>=MOD) A[k+j]-=MOD;
                A[k+j+m/2]=u+MOD-t;
                if(A[k+j+m/2]>=MOD) A[k+j+m/2]-=MOD;
                w=1LL*w*wn%MOD;
            }
        }
    }
    if(on==-1)
    {
        for(int i=1;i<n/2;i++)
            swap(A[i],A[n-i]);
        for(int i=0;i<n;i++)
            A[i]=1LL*A[i]*two[cnt]%MOD;
    }
}
int n,A[MAXN],B[MAXN],C[MAXN],D[MAXN];
void find_inverse(int A[],int n)
{
        if(n==1) {B[0]=pow_mod(A[0],MOD-2); return;}
        find_inverse(A,(n+1)/2);
        int len=dbit(n);
        for(int i=0;i<n;i++)
                C[i]=A[i];
        for(int i=n;i<len;i++) C[i]=0;
        ntt(len,C,1);ntt(len,B,1);
        for(int i=0;i<len;i++)
                C[i]=1LL*B[i]*B[i]%MOD*C[i]%MOD;
        ntt(len,C,-1);ntt(len,B,-1);
        for(int i=0;i<n;i++)
                B[i]=((2*B[i]-C[i])%MOD+MOD)%MOD;
}
void find_sqr(int A[],int n)
{
    if(n==1)
    {
        D[0]=A[0];
        return;
    }
    find_sqr(A,(n+1)/2);
    memset(B,0,sizeof(B));
    find_inverse(D,(n+1)/2);
    for(int i=0;i<(n+1)/2;i++)
        B[i]=1LL*B[i]*((MOD+1)/2)%MOD;
    int len=dbit(n)*2;
    ntt(len,D,1);
    for(int i=0;i<len;i++)
        D[i]=1LL*D[i]*D[i]%MOD;
    ntt(len,D,-1);
    for(int i=0;i<n;i++)
        D[i]=(D[i]+A[i])%MOD;
    ntt(len,D,1);ntt(len,B,1);
    for(int i=0;i<len;i++)
        D[i]=1LL*D[i]*B[i]%MOD;
    ntt(len,D,-1);
    for(int i=n;i<2*n;i++) D[i]=0;
}
int main()
{
        for(int i=1;i<=30;i++)
        two[i]=pow_mod(1<<i,MOD-2);
    A[0]=1;
    A[1]=MOD-2;
    A[2]=1;
    find_sqr(A,4);
    for(int i=0;i<4;i++) printf("%d ",D[i]);
    return 0;
}
```

4.19 Stirling number of the first kind

```
#include<bits/stdc++.h>
#define MAXN 500005
#define MOD 998244353
#define INF 1000000000
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int g=3;
int tot=1;
ll dbit(ll x)
{
    while((x&-x)!=x) x+=x&-x;
    return x;
}
int two[32];
int pow_mod(int a,int i)
{
    if(i==0) return 1;
    int s=1;
    while(i>0)
     {
         if(i&1) s=(1LL*s*a)%MOD;
         a=(1LL*a*a)%MOD;
         i>>=1;
     }
     return s;
}
int rev(int x,int r)
{
    int ans=0;
    for(int i=0;i<r;i++)
        if(x&(1<<i)) ans+=1<<(r-i-1);
    return ans;
}
void ntt(int n,int A[],int on)
{
    int r=0,cnt=0,t=n;
    while(t>1) {cnt++; t/=2;}
    for(;;r++) if((1<<r)==n) break;
    for(int i=0;i<n;i++)
    {
        int tmp=rev(i,r);
        if(i<tmp) swap(A[i],A[tmp]);
    }
    for(int s=1;s<=r;s++)
    {
        int m=1<<s;
        int wn=pow_mod(g,(MOD-1)/m);
        for(int k=0;k<n;k+=m)
        {
            int w=1;
            for(int j=0;j<m/2;j++)
            {
                int t,u;
                t=1LL*w*A[k+j+m/2]%MOD;
                u=A[k+j];
                A[k+j]=(u+t);
                if(A[k+j]>=MOD) A[k+j]-=MOD;
                A[k+j+m/2]=u+MOD-t;
                if(A[k+j+m/2]>=MOD) A[k+j+m/2]-=MOD;
                w=1LL*w*wn%MOD;
            }
        }
    }
    if(on==-1)
    {
        for(int i=1;i<n/2;i++)
            swap(A[i],A[n-i]);
        for(int i=0;i<n;i++)
            A[i]=1LL*A[i]*two[cnt]%MOD;
    }
```

```
}
int A[MAXN],B[MAXN],C[10000000];
struct atom
{
    int l,r;
};
atom solve(int l,int r)
{
    if (l>r){ C[++tot]=1; return (atom){tot,tot};}
    if (l==r){ C[++tot]=l; C[++tot]=1; return (atom){tot-1,tot};}
    int mid=(l+r)/2; atom k1=solve(l,mid),k2=solve(mid+1,r);
    int n=max(mid-l+1,r-mid),sz=1;
    while (sz<=(n<<1)) sz*=2;
    for (int i=0;i<sz;i++){A[i]=0; B[i]=0;}
    for (int i=k1.l;i<=k1.r;i++) A[i-k1.l]=C[i];
    for (int i=k2.l;i<=k2.r;i++) B[i-k2.l]=C[i];
    ntt(sz,A,1); ntt(sz,B,1);
    for (int i=0;i<sz;i++) A[i]=1LL*A[i]*B[i]%MOD;
    ntt(sz,A,-1);
    atom ans; ans.l=tot+1;
    for (int i=0;i<=r-l+1;i++) C[++tot]=A[i]; ans.r=tot;
    return ans;
}
int n;
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=30;i++)
        two[i]=pow_mod(1<<i,MOD-2);
    atom ans=solve(0,n-1);
    for(int i=ans.l;i<=ans.r;i++)
        printf("%d ",C[i]);
    return 0;
}
```

4.20 Stirling number of the second kind(single)

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
int fact[MAXN];
int pow_mod(int a,int i)
{
    if(i==0) return 1;
    int s=1;
    while(i>0)
     {
        if(i&1) s=(1LL*s*a)%MOD;
        a=(1LL*a*a)%MOD;
        i>>=1;
     }
     return s;
}
int inv(int x)
{
       return pow_mod(x,MOD-2);
}
int n,m;
int main()
{
       scanf("%d%d",&n,&m);
       fact[0]=1;
       for(int i=1;i<=n;i++)
              fact[i]=1LL*fact[i-1]*i%MOD;
       int ans=0;
       for(int k=0;k<=m;k++)
       {
              int res=((1LL*fact[m]*inv(fact[k])%MOD)*inv(fact[m-k])%MOD)*pow_mod(m-k,n)%MOD;
              if(!(k&1)) ans=(ans+res)%MOD; else ans=(ans+MOD-res)%MOD;
       }
       ans=1LL*ans*(inv(fact[m]))%MOD;
       printf("%d\n",ans);
}
```

4.21 Stirling number of the second kind(multiple)

```
#include<bits/stdc++.h>
#define MAXN 100005
#define MOD 998244353
#define INF 1000000000
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int g=3;
int two[32];
int dbit(int x)
{
    while((x&-x)!=x) x+=x&-x;
    return x;
}
int pow_mod(int a,int i)
{
    if(i==0) return 1;
    int s=1;
    while(i>0)
     {
         if(i&1) s=(1LL*s*a)%MOD;
         a=(1LL*a*a)%MOD;
         i>>=1;
     }
     return s;
}
int rev(int x,int r)
{
    int ans=0;
    for(int i=0;i<r;i++)
```

```
        if(x&(1<<i)) ans+=1<<(r-i-1);
    return ans;
}
void ntt(int n,int A[],int on)
{
    int r=0,cnt=0,t=n;
    while(t>1) {cnt++; t/=2;}
    for(;;r++) if((1<<r)==n) break;
    for(int i=0;i<n;i++)
    {
        int tmp=rev(i,r);
        if(i<tmp) swap(A[i],A[tmp]);
    }
    for(int s=1;s<=r;s++)
    {
        int m=1<<s;
        int wn=pow_mod(g,(MOD-1)/m);
        for(int k=0;k<n;k+=m)
        {
            int w=1;
            for(int j=0;j<m/2;j++)
            {
                int t,u;
                t=1LL*w*A[k+j+m/2]%MOD;
                u=A[k+j];
                A[k+j]=(u+t);
                if(A[k+j]>=MOD) A[k+j]-=MOD;
                A[k+j+m/2]=u+MOD-t;
                if(A[k+j+m/2]>=MOD) A[k+j+m/2]-=MOD;
                w=1LL*w*wn%MOD;
            }
        }
    }
    if(on==-1)
    {
        for(int i=1;i<n/2;i++)
            swap(A[i],A[n-i]);
        for(int i=0;i<n;i++)
            A[i]=1LL*A[i]*two[cnt]%MOD;
    }
}
int fact[MAXN],inv[MAXN],A[MAXN],B[MAXN];
int main()
{
    int n;
    for(int i=1;i<=30;i++)
        two[i]=pow_mod(1<<i,MOD-2);
    scanf("%d",&n);
    fact[0]=1,inv[0]=1;
    for(int i=1;i<=n;i++)
    {
        fact[i]=1LL*fact[i-1]*i%MOD;
        inv[i]=pow_mod(fact[i],MOD-2);
    }
    int sz=dbit(n)*2;
    //printf("%d\n",sz);
    memset(A,0,sizeof(A));
    memset(B,0,sizeof(B));
    for(int i=0;i<=n;i++)
    {
        if(i&1) A[i]=MOD-inv[i]; else A[i]=inv[i];
        B[i]=1LL*inv[i]*pow_mod(i,n)%MOD;
        //printf("%d %d\n",A[i],B[i]);
    }
    ntt(sz,A,1);ntt(sz,B,1);
    for(int i=0;i<sz;i++)
        A[i]=1LL*A[i]*B[i]%MOD;
    ntt(sz,A,-1);
    for(int i=0;i<=n;i++)
        printf("%d ",A[i]);
    return 0;
}
```

4.22 SumPhi

```
#include<bits/stdc++.h>
#define MAXN 5000005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
bool is_prime[MAXN];
ll cnt,phi[MAXN],prime[MAXN];
ll n,f[MAXN];
map<ll,ll> mp;
ll mul_mod(ll a,ll i)
{
        ll s=0;a%=MOD;
        while(i)
        {
                if(i&1) s=(s+a)%MOD;
                a=(a+a)%MOD;
                i>>=1;
        }
        return s;
}

ll pow_mod(ll a,ll i)
{
        ll s=1;
        while(i)
        {
                if(i&1) s=mul_mod(s,a);
                a=mul_mod(a,a);
                i>>=1;
        }
        return s;
}
void genphi(ll n)
{
    ll p=0;
    memset(phi,0,sizeof(phi));
    phi[1]=1;
```

```
     for(ll i=0;i<=n;i++) is_prime[i]=true;
    is_prime[0]=is_prime[1]=false;
    for(ll i=2;i<=n;i++)
    {
        if(is_prime[i]) {prime[p++]=i; phi[i]=i-1;}
        for(ll j=0;j<p;j++)
        {
            if(prime[j]*i>n) break;
            is_prime[prime[j]*i]=false;
            phi[i*prime[j]]=phi[i]*(i%prime[j]?prime[j]-1:prime[j]);
            if(i%prime[j]==0) break;
        }
    }
    for(ll i=1;i<=n;i++) f[i]=(f[i-1]+phi[i])%MOD;
}
ll calc(ll x)
{
        if(x<=5000000) return f[x];
        if(mp.find(x)!=mp.end()) return mp[x];
        ll ans=mul_mod(mul_mod(x,x+1),pow_mod(2,MOD-2));
        for(ll i=2,r;i<=x;i=r+1)
        {
                r=x/(x/i);
                ans=(ans-calc(x/i)*((r-i+1)%MOD)%MOD+MOD)%MOD;
        }
        return mp[x]=ans;
}
int main()
{
        genphi(5000000);
        scanf("%lld",&n);
        printf("%lld\n",calc(n));
        return 0;
}
```

4.23 SumMiu

```
#include<bits/stdc++.h>
#define MAXN 5000005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
bool is_prime[MAXN];
int cnt,miu[MAXN],prime[MAXN];
ll n,m,f[MAXN];
map<ll,ll> mp;
void genmiu(int n)
{
    int p=0;
    for(int i=0;i<=n;i++) is_prime[i]=true;
    is_prime[0]=is_prime[1]=false;
    memset(miu,0,sizeof(miu));
    miu[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(is_prime[i]) {prime[p++]=i; miu[i]=-1;}
        for(int j=0;j<p;j++)
        {
            if(prime[j]*i>n) break;
            is_prime[prime[j]*i]=false;
            miu[i*prime[j]]=i%prime[j]?-miu[i]:0;
            if(i%prime[j]==0) break;
        }
    }
    for(int i=1;i<=n;i++) f[i]=f[i-1]+miu[i];
}
ll calc(ll x)
{
        if(x<=5000000) return f[x];
        if(mp.find(x)!=mp.end()) return mp[x];
        ll ans=1;
        for(ll i=2,r;i<=x;i=r+1)
        {
                r=x/(x/i);
                ans-=calc(x/i)*(r-i+1);
        }
        return mp[x]=ans;
}
int main()
{
        genmiu(5000000);
        scanf("%lld%lld",&n,&m);
        printf("%lld\n",calc(m)-calc(n-1));
        return 0;
}
```

5 String

5.1 Trie

```
#include<bits/stdc++.h>
#define MAXN 50020
using namespace std;
struct trie
{
    trie* next[26];
};
trie* thead;
char str[MAXN][1001];
inline trie* newnode()
{
    trie* t;
    t=(trie*)malloc(sizeof(trie));
    memset(t,0,sizeof(trie));
    return t;
}
void insert(char x[])
{
    int i;
    trie* s=thead;
    trie* t;
    for(i=0;x[i];i++)
    {
        if(s->next[x[i]-'a']) {s=s->next[x[i]-'a'];}
        else
        {
            t=newnode();
            s->next[x[i]-'a']=t;
            s=t;
        }
    }
    return;
}
bool find(char x[])
{
    trie* s=thead;
    int i;
    for(i=0;x[i];i++)
    {
        if(s->next[x[i]-'a']==NULL) return false;
        s=s->next[x[i]-'a'];
    }
    return true;
}
void deltrie(trie* s)
{
    int i;
    for(i=0;i<26;i++)
    {
        if(s->next[i])
        deltrie(s->next[i]);
    }
    free(s);
    s=NULL;
}
int main()
{
    int i=0;
    thead=newnode();
    while(scanf("%s",str[i])==1)
    {
        if(str[i][0]=='1') break;
        insert(str[i]);
        i++;
    }
    char x[20];
    while(scanf("%s",x)==1)
        printf(find(x)?"yes\n":"no\n");
    deltrie(thead);
    return 0;
}
```

5.2 KMP

```
#include<bits/stdc++.h>
using namespace std;
```

```
vector<int> kmp(string a,string b) // a=pattern, b=text
{
    int n=a.size();
    vector<int> next(n+1,0);
    for(int i=1;i<n;++i)
    {
        int j=i;
        while(j>0)
        {
            j=next[j];
            if(a[j]==a[i])
            {
                next[i+1]=j+1;
                break;
            }
        }
    }
    vector<int> p;//p=positions
    int m=b.size();
    for(int i=0,j=0;i<m;++i)
    {
        if(j<n&&b[i]==a[j])
        {
            j++;
        }
        else
        {
            while(j>0)
            {
                j=next[j];
                if(b[i]==a[j])
                {
                    j++;
                    break;
                }
            }
        }
        if(j==n)
        {
            p.push_back(i-n+1);
        }
    }
    return p;
}
int main()
{
    return 0;
}
```

5.3 Hash Matching

```
#include<bits/stdc++.h>
#define MAXN 100005
using namespace std;
typedef unsigned long long ull;
const ull B=1000000007;
bool contain(string a,string b)
```

```
{
        int al=a.length(),bl=b.length();
        if(al>bl) return false;
        ull t=1;
        for(int i=0;i<al;i++)
                t*=B;
        ull ah=0,bh=0;
        for(int i=0;i<al;i++) ah=ah*B+a[i];
        for(int i=0;i<al;i++) bh=bh*B+b[i];
        for(int i=0;i+al<=bl;i++)
        {
                if(ah==bh) return true;
                if(i+al<bl) bh=bh*B+b[i+al]-b[i]*t;
        }
        return false;
}
```

5.4 Aho-Corasick Automaton

```
#include<bits/stdc++.h>
#define MAXN 50020
using namespace std;
struct trie
{
    trie* next[26];
    trie* fail;
    bool mark;
};
trie* thead;
char str[MAXN][1001];
inline trie* newnode()
{
    trie* t;
    t=(trie*)malloc(sizeof(trie));
    t->fail=NULL;
    t->mark=false;
    memset(t,0,sizeof(trie));
    return t;
}
void insert(char x[])
{
    int i;
    trie* s=thead;
    trie* t;
    for(i=0;x[i];i++)
    {
        if(s->next[x[i]-'a']) {s=s->next[x[i]-'a'];}
        else
        {
            t=newnode();
            s->next[x[i]-'a']=t;
            s=t;
        }
    }
    s->mark=true;
    return;
}
```

```
trie* g(trie* s, char x)
{
    if(s->next[x-'a']) return s->next[x-'a'];
    else if(s==thead) return thead;
    else return NULL;
}

void bfs()
{
    trie* s=thead;
    queue<trie*> que;
    for(int i=0;i<26;i++)
        if(s->next[i]){s->next[i]->fail=thead; que.push(s->next[i]);}
    while(!que.empty())
    {
        trie* t=que.front();
        que.pop();
        for(int i=0;i<26;i++)
            if(g(t,(char)('a'+i))!=NULL)
            {
                que.push(t->next[i]);
                trie* v=t->fail;
                while(g(v,(char)('a'+i))==NULL) v=v->fail;
                t->next[i]->fail=g(v,(char)('a'+i));
            }
    }
    return;
}
int match(char x[])
{
    trie* s=thead;
    int cnt=0;
    for(int i=0;x[i];i++)
    {
        while(g(s,x[i])==NULL)
        {
            s=s->fail;
            if(s->mark) cnt++;
        }
        s=g(s,x[i]);
        if(s->mark) cnt++;
    }
     while(s->fail!=thead)
     {
        s=s->fail;
        if(s->mark) cnt++;
     }
    return cnt;
}
bool find(char x[])
{
    trie* s=thead;
    for(int i=0;x[i];i++)
    {
        if(s->next[x[i]-'a']==NULL) return false;
        s=s->next[x[i]-'a'];
    }
    return true;
}
```

```
void deltrie(trie* s)
{
    int i;
    for(i=0;i<26;i++)
    {
        if(s->next[i])
        deltrie(s->next[i]);
    }
    free(s);
    s=NULL;
}
int main()
{
    int i=0;
    thead=newnode();
    while(scanf("%s",str[i])==1)
    {
        if(str[i][0]=='1') break;
        insert(str[i]);
        i++;
    }
    bfs();
    char p[100];
    scanf("%s",p);
    printf("%d\n",match(p));
    deltrie(thead);
    return 0;
}
```

5.5 Suffix Array

```
#include<bits/stdc++.h>
#define MAXN 1005
using namespace std;
int n,k;
int r[MAXN+1];
int sa[MAXN],lcp[MAXN];
int c[MAXN],t1[MAXN],t2[MAXN];
string S;
void construct_sa(string S,int *sa)
{
    int n=S.length()+1;
    int m=130;
    int i,*x=t1,*y=t2;
    for(i=0;i<m;i++) c[i]=0;
    for(i=0;i<n;i++) c[x[i]=S[i]]++;
    for(i=1;i<m;i++) c[i]+=c[i-1];
    for(i=n-1;i>=0;i--) sa[--c[x[i]]]=i;
    for(int k=1;k<=n;k<<=1) {
        int p=0;
        for(i=n-k;i<n;i++) y[p++]=i;
        for(i=0;i<n;i++) if(sa[i]>=k) y[p++]=sa[i]-k;
        for(i=0;i<m;i++) c[i]=0;
        for(i=0;i<n;i++) c[x[y[i]]]++;
        for(i=0;i<m;i++) c[i]+=c[i-1];
        for(i=n-1;i>=0;i--) sa[--c[x[y[i]]]]=y[i];
        swap(x,y);
        p=1; x[sa[0]]=0;
        for(i=1;i<n;i++)
            x[sa[i]]=y[sa[i]]==y[sa[i-1]] && y[sa[i]+k]==y[sa[i-1]+k]?p-1:p++;
        if(p>=n) break;
        m=p;
    }
}
void construct_lcp(string S,int *sa,int *lcp)
{
    int n=S.length();
    for(int i=0;i<=n;i++) r[sa[i]]=i;
    int h=0;
    lcp[0]=0;
    for(int i=0;i<n;i++)
    {
        int j=sa[r[i]-1];
        if(h>0) h--;
        for(;j+h<n&&i+h<n;h++)
        {
            if(S[j+h]!=S[i+h]) break;
        }
        lcp[r[i]-1]=h;
    }
}
int main()
{
    cin>>S;
    n=S.size();
    construct_sa(S,sa);
    construct_lcp(S,sa,lcp);
    int cnt=0;
    return 0;
}
```

5.6 SA-IS

```
#include<bits/stdc++.h>
#define MAXN 1000000
#define L_TYPE 0
#define S_TYPE 1
using namespace std;
inline bool is_lms_char(int *type, int x) {
    return x > 0 && type[x] == S_TYPE && type[x - 1] == L_TYPE;
}
inline bool equal_substring(int *S, int x, int y, int *type) {
    do {
        if (S[x] != S[y])
            return false;
        x++, y++;
    } while (!is_lms_char(type, x) && !is_lms_char(type, y));

    return S[x] == S[y];
}
inline void induced_sort(int *S, int *SA, int *type, int *bucket, int *lbucket,int *sbucket, int
     n, int SIGMA)
{
    for (int i = 0; i <= n; i++)
```

```
        if (SA[i] > 0 && type[SA[i] - 1] == L_TYPE)
            SA[lbucket[S[SA[i] - 1]]++] = SA[i] - 1;
    for (int i = 1; i <= SIGMA; i++)
        sbucket[i] = bucket[i] - 1;
    for (int i = n; i >= 0; i--)
        if (SA[i] > 0 && type[SA[i] - 1] == S_TYPE)
            SA[sbucket[S[SA[i] - 1]]--] = SA[i] - 1;
}
static int *SAIS(int *S, int length, int SIGMA)
{
    int n = length - 1;
    int *type = new int[n + 1];
    int *position = new int[n + 1];
    int *name = new int[n + 1];
    int *SA = new int[n + 1];
    int *bucket = new int[SIGMA];
    int *lbucket = new int[SIGMA];
    int *sbucket = new int[SIGMA];
    memset(bucket, 0, sizeof(int) * (SIGMA + 1));
    for (int i = 0; i <= n; i++)
        bucket[S[i]]++;
    for (int i = 1; i <= SIGMA; i++)
    {
        bucket[i] += bucket[i - 1];
        lbucket[i] = bucket[i - 1];
        sbucket[i] = bucket[i] - 1;
    }
    type[n] = S_TYPE;
    for (int i = n - 1; i >= 0; i--)
    {
        if (S[i] < S[i + 1])
            type[i] = S_TYPE;
        else if (S[i] > S[i + 1])
            type[i] = L_TYPE;
        else
            type[i] = type[i + 1];
    }
    int cnt = 0;
    for (int i = 1; i <= n; i++)
        if (type[i] == S_TYPE && type[i - 1] == L_TYPE)
            position[cnt++] = i;
    fill(SA, SA + n + 1, -1);
    for (int i = 0; i < cnt; i++)
        SA[sbucket[S[position[i]]]--] = position[i];
    induced_sort(S, SA, type, bucket, lbucket, sbucket, n, SIGMA);
    fill(name, name + n + 1, -1);
    int lastx = -1, namecnt = 1;
    bool flag = false;
    for (int i = 1; i <= n; i++)
    {
        int x = SA[i];

        if (is_lms_char(type, x)) {
            if (lastx >= 0 && !equal_substring(S, x, lastx, type))
                namecnt++;
            if (lastx >= 0 && namecnt == name[lastx])
                flag = true;

            name[x] = namecnt;
            lastx = x;
        }
    }
    name[n] = 0;
    int *S1 = new int[cnt];
    int pos = 0;
    for (int i = 0; i <= n; i++)
        if (name[i] >= 0)
            S1[pos++] = name[i];

    int *SA1;
    if (!flag)
    {
        SA1 = new int[cnt + 1];
        for (int i = 0; i < cnt; i++)
            SA1[S1[i]] = i;
    }
    else
        SA1 = SAIS(S1, cnt, namecnt);
    lbucket[0] = sbucket[0] = 0;
    for (int i = 1; i <= SIGMA; i++)
    {
        lbucket[i] = bucket[i - 1];
        sbucket[i] = bucket[i] - 1;
    }
    fill(SA, SA + n + 1, -1);
    for (int i = cnt - 1; i >= 0; i--)
        SA[sbucket[S[position[SA1[i]]]]--] = position[SA1[i]];
    induced_sort(S, SA, type, bucket, lbucket, sbucket, n, SIGMA);
    return SA;
}
int main()
{
    return 0;
}
```

5.7 Manacher

```
#include<bits/stdc++.h>
#define MAXN 10000
using namespace std;
void manacher(char str[],int len[],int n)
{
    len[0]=1;
    for(int i=1,j=0;i<(n<<1)-1;++i)
    {
        int p=i>>1,q=i-p,r=((j+1)>>1)+len[j]-1;
        len[i]=r<q?0:min(r-q+1,len[(j<<1)-i]);
        while(p>len[i]-1&&q+len[i]<n&&str[p-len[i]]==str[q+len[i]])
            ++len[i];
        if(q+len[i]-1>r)
            j=i;
    }
}
int a[MAXN];
char str[MAXN];
int main()
```

```
{
    scanf("%s",str);
    int x=strlen(str);
    manacher(str,a,strlen(str));
    for(int i=0;i<2*x-1;i++)
      printf("%d ",a[i]);
}
```

5.8 Suffix Automaton

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
struct SuffixAutomaton
{
    vector<map<char,int>> edges;
    vector<int> link;
    vector<int> length;
    int last;
    SuffixAutomaton(string s)
    {
        edges.push_back(map<char,int>());
        link.push_back(-1);
        length.push_back(0);
        last=0;
        for(int i=0;i<s.size();i++)
        {
            edges.push_back(map<char,int>());
            length.push_back(i+1);
            link.push_back(0);
            int r=edges.size()-1;
            int p=last;
            while(p>=0 && edges[p].find(s[i])==edges[p].end())
            {
                edges[p][s[i]]=r;
                p=link[p];
            }
            if(p!=-1)
            {
                int q=edges[p][s[i]];
                if(length[p]+1==length[q]) link[r]=q;
                else
                {
                    edges.push_back(edges[q]); // copy edges of q
                    length.push_back(length[p]+1);
                    link.push_back(link[q]); // copy parent of q
                    int qq=edges.size()-1;
                    // add qq as the new parent of q and r
                    link[q]=qq;
                    link[r]=qq;
                    // move short classes pointing to q to point to q'
                    while(p>=0 && edges[p][s[i]]==q)
                    {
                        edges[p][s[i]]=qq;
                        p=link[p];
                    }
                }
            }
            last=r;
        }
        vector<int> terminals;
        int p=last;
        while(p>0)
        {
            terminals.push_back(p);
            p=link[p];
        }
    }
};
int main()
{
    return 0;
}
```

6 Others

6.1 Largest Rectangle

```
#include<bits/stdc++.h>
#define MAXN 100000
using namespace std;
int n;
int h[MAXN];
int L[MAXN],R[MAXN];
int st[MAXN];
void solve()
{
        int t=0;
        for(int i=0;i<n;i++)
        {
                while(t>0&&h[st[t-1]]>=h[i]) t--;
                L[i]=t==0?0:(st[t-1]+1);
                st[t++]=i;
        }
        t=0;
        for(int i=n-1;i>=0;i--)
        {
                while(t>0&&h[st[t-1]]>=h[i]) t--;
                R[i]=t==0?n:st[t-1];
                st[t++]=i;
        }
        long long res=0;
        for(int i=0;i<n;i++)
        {
                res=max(res,(long long)h[i]*(R[i]-L[i]));
        }
        printf("%lld\n",res);
}
```

6.2 Sliding Minimum

```
#include<bits/stdc++.h>
#define MAXN 100005
using namespace std;
int n,k;
int a[MAXN];
int b[MAXN];
int deq[MAXN];
void solve()
{
        int s=0,t=0;
        for(int i=0;i<n;i++)
        {
                while(s<t&&a[deq[t-1]]>=a[i]) t--;
                deq[t++]=i;
                if(i-k+1>=0)
                {
                        b[i-k+1]=a[deq[s]];
                        if(deq[s]==i-k+1)
                        {
                                s++;
                        }
                }
        }
        for(int i=0;i<=n-k;i++)
        {
                printf("%d%c",b[i],i==n-k?'\n':' ');
        }
}
int main()
{
    scanf("%d %d",&n,&k);
    for(int i=0;i<n;i++)
        scanf("%d",&a[i]);
    solve();
    return 0;
}
```

6.3 Multiple Backpack

```
#include<bits/stdc++.h>
#define MAXN 100005
int w[MAXN],v[MAXN],m[MAXN];
int dp[MAXW+1];
int deq[MAXW+1];
int deqv[MAXW+1];
void solve()
{
    for(int i=0;i<n;i++)
    {
        for(int a=0;a<w[i];a++)
        {
            int s=0,t=0;
            for(int j=0;j*w[i]+a<=W;j++)
            {
                int val=dp[j*w[i]+a]-j*v[i];
                while(s<t&&deqv[t-1]<=val) t--;
                deq[t]=j;
                deqv[t++]=val;
                dp[j*w[i]+a]=deqv[s]+j*v[i];
                if(deq[s]==j-m[i]) s++;
            }
        }
    }
    printf("%d\n",dp[W]);
}
```

6.4 Convex Hull Trick

```
#include <bits/stdc++.h>
#define ll long long
const int N=100050;
ll dp[N],b[N],a[N],T[N],t,p,n,i;
ll Get(int u, int v){ return (dp[u]-dp[v]+b[v]-b[u]-1)/(b[v]-b[u]);}
int main()
{
        scanf("%I64d",&n);
        for(i=1;i<=n;i++) scanf("%I64d",&a[i]);
        for(i=1;i<=n;i++) scanf("%I64d",&b[i]);
        T[t++]=1;
        for(i=2;i<=n;i++)
        {
                while(t-p>1 && Get(T[p],T[p+1])<=a[i]) p++;
                dp[i]=a[i]*b[T[p]]+dp[T[p]];
                while(t-p>1 && Get(T[t-1],i)<=Get(T[t-1],T[t-2])) t--;
                T[t++]=i;
        }
        printf("%I64d\n",dp[n]);
        return 0;
}
```

6.5 Knuth's optimization

```
#include<bits/stdc++.h>
#define MAXN 2005
#define INF 1000000000
using namespace std;
typedef long long ll;
ll a[MAXN];
ll n,k;
ll dp[MAXN][MAXN],knuth[MAXN][MAXN];
int main()
{
    while(scanf("%lld %lld",&n,&k)==2)
    {
        a[0]=0;
        for(ll i=1;i<=k;i++)
```

```
            scanf("%lld",&a[i]);
        a[k+1]=n;
        for(ll i=0;i<=k+1;i++)
            for(ll j=0;j<=k+1;j++)
                dp[i][j]=INF;
        for(ll i=0;i<=k;i++)
            dp[i][i+1]=0;
        for(ll l=3;l<=k+2;l++)
            for(ll i=0;i<=k+2-l;i++)
            {
                if(l==3)
                {
                    dp[i][i+l-1]=a[i+l-1]-a[i];
                    knuth[i][i+l-1]=i+1;
                }
                else
                    for(ll j=knuth[i][i+l-2];j<=knuth[i+1][i+l-1];j++)
                        if(dp[i][j]+dp[j][i+l-1]+a[i+l-1]-a[i]<dp[i][i+l-1])
                        {
                            dp[i][i+l-1]=dp[i][j]+dp[j][i+l-1]+a[i+l-1]-a[i];
                            knuth[i][i+l-1]=j;
                        }
            }
        printf("%lld\n",dp[0][k+1]);
    }
    return 0;
}
```

6.6 Centroid Decomposition

```
#include<bits/stdc++.h>
#define MAXN 10005
using namespace std;
struct edge{int to,length;};
int N,K;
vector<edge> G[MAXN];
bool centroid[MAXN];
int subtree_size[MAXN];
int ans;
int compute_subtree_size(int v,int p)
{
    int c=1;
    for(int i=0;i<G[v].size();i++)
    {
        int w=G[v][i].to;
        if(w==p||centroid[w]) continue;
        c+=compute_subtree_size(G[v][i].to,v);
    }
    subtree_size[v]=c;
    return c;
}
pair<int,int> search_centroid(int v,int p,int t)
{
    pair<int,int> res=make_pair(INT_MAX,-1);
    int s=1,m=0;
    for(int i=0;i<G[v].size();i++)
    {
        int w=G[v][i].to;
        if(w==p||centroid[w]) continue;
        res=min(res,search_centroid(w,v,t));
        m=max(m,subtree_size[w]);
        s+=subtree_size[w];
    }
    m=max(m,t-s);
    res=min(res,make_pair(m,v));
    return res;
}
void enumerate_paths(int v,int p,int d,vector<int> &ds)
{
    ds.push_back(d);
    for(int i=0;i<G[v].size();i++)
    {
        int w=G[v][i].to;
        if(w==p||centroid[w]) continue;
        enumerate_paths(w,v,d+G[v][i].length,ds);
    }
}
int count_pairs(vector<int> &ds)
{
    int res=0;
    sort(ds.begin(),ds.end());
    int j=ds.size();
    for(int i=0;i<ds.size();i++)
    {
        while(j>0&&ds[i]+ds[j-1]>K) j--;
        res+=j-(j>i?1:0);
    }
    return res/2;
}
void solve_subproblem(int v)
{
    compute_subtree_size(v,-1);
    int s=search_centroid(v,-1,subtree_size[v]).second;
    centroid[s]=true;
    for(int i=0;i<G[s].size();i++)
    {
        if(centroid[G[s][i].to]) continue;
        solve_subproblem(G[s][i].to);
    }
    vector<int> ds;
    ds.push_back(0);
    for(int i=0;i<G[s].size();i++)
    {
        if(centroid[G[s][i].to]) continue;
        vector<int> tds;
        enumerate_paths(G[s][i].to,s,G[s][i].length,tds);
        ans-=count_pairs(tds);
        ds.insert(ds.end(),tds.begin(),tds.end());
    }
    ans+=count_pairs(ds);
    centroid[s]=false;
}
void solve()
{
    ans=0;
    solve_subproblem(0);
    printf("%d\n",ans);
}
int main()
{
    int M;
    while(scanf("%d%d",&N,&K)==2)
    {
        if(!N&&!K) break;
        for(int i=0;i<N;i++)
            G[i].clear();
        for(int i=0;i<N-1;i++)
        {
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            x--;
            y--;
            G[x].push_back((edge){y,z});
            G[y].push_back((edge){x,z});
        }
        memset(centroid,false,sizeof(centroid));
        solve();
    }
    return 0;
}
```

6.7 Linear Programming

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;

const int N = 23;
const double eps = 1e-8;

double a[N][N], ans[N];
int n, m, t, id[N << 1];

void pivot(int l, int e)
{
    swap(id[e], id[n + l]);
    double r = a[l][e]; a[l][e] = 1;
    for (int j = 0; j <= n; ++j)
        a[l][j] /= r;
    for (int i = 0; i <= m; ++i)
        if (i != l) {
            r = a[i][e]; a[i][e] = 0;
            for (int j = 0; j <= n; ++j)
                a[i][j] -= r * a[l][j];
        }
}

int main()
{
    scanf("%d%d", &n, &m);
    int i, j, l, e; double k, kk;
    for (j = 1; j <= n; ++j) scanf("%lf", &a[0][j]), id[j] = j;
    for (i = 1; i <= m; ++i)
    {
        for (j = 1; j <= n; ++j)
            scanf("%lf", &a[i][j]);
        scanf("%lf", &a[i][0]);
    }

    while (true)
    {
        l = e = 0; k = -eps;
        for (i = 1; i <= m; ++i)
            if (a[i][0] < k) {
                k = a[i][0];
                l = i;
            }
        if (!l) break;
        k = -eps;
        for (j = 1; j <= n; ++j)
            if (a[l][j] < k && (!e || (rand() & 1))) {
                k = a[l][j];
                e = j;
            }
        if (!e) {puts("Infeasible"); return 0;}
        pivot(l, e);
    }

    while (true) {
        for (j = 1; j <= n; ++j)
            if (a[0][j] > eps)
                break;
        if ((e = j) > n) break;
        k = 1e18; l = 0;
        for (i = 1; i <= m; ++i)
            if (a[i][e] > eps && (kk = (a[i][0] / a[i][e])) < k) {
                k = kk;
                l = i;
            }
        if (!l) {puts("Unbounded"); return 0;}
        pivot(l, e);
    }

    printf("%.10lf\n", -a[0][0]);
    for (i = 1; i <= m; ++i) ans[id[n + i]] = a[i][0];
    for (i = 1; i <= n; ++i) printf("%.10lf ", ans[i]);
    return 0;
}
```

6.8 Sum Over Subsets

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
int n,a[MAXN],f[MAXN];
int main()
{
        scanf("%d",&n);
        for(int i=0;i<(1<<n);i++)
                scanf("%d",&a[i]);
        for(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)];
        }
        for(int i=0;i<(1<<n);i++)
                printf("%d ",f[i]);
        puts("");
        return 0;
}
```

6.9 whatday

```
#include<bits/stdc++.h>
using namespace std;
int whatday(int d,int m,int y)
{
    int ans;
    if(m==1||m==2)
        m+=12,y--;
    if((y<1752)||(y==1752&&m<9)||(y==1752&&m==9&&d<3))
        ans=(d+2*m+3*(m+1)/5+y+y/4+5)%7;
    else
        ans=(d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)%7;
    return ans;
}
int main()
{
    return 0;
}
```