

Code Template for ACM-ICPC

Roundgod @ Nanjing University

November 14, 2018

Contents

1	DataStructures	1
1.1	Heap	1
1.2	Fenwick Tree	1
1.3	Mo's algorithm	2
1.4	Mo's algorithm with Queries	3
1.5	Heavy Light Decomposition	5
1.6	Segment Tree	8
1.7	Segment Tree Beats	10
1.8	Persistent Segment Tree	12
1.9	Persistent DSU	13
1.10	Persistent Trie	14
1.11	Monotone Stack	15
1.12	Monotone Deque	15
1.13	Splay	16
1.14	Treap	19
1.15	Link-Cut Tree	21
1.16	Union Set	22
1.17	Sparse Table	23
1.18	DSU on Tree	24
1.19	Virtual Tree	25
1.20	Centroid Decomposition	26
2	Geometry	28
2.1	Geometry All-in-One	28
2.2	Stereometry	41
3	Graph	48
3.1	Dijkstra	48
3.2	Floyd-Warshall	48
3.3	Korasaju	49
3.4	LCA with binary lifting	49
3.5	LCA with range minimum query	50
3.6	Dinic	52
3.7	Min-cost flow	53
3.8	Bipartite Matching	54
3.9	Common Matching	55
3.10	Kuhn-Munkres	58
3.11	Linear Programming	60
3.12	Dominator Tree	61
3.13	Hopcroft-Karp	62
3.14	Edge-connected Components	63
3.15	Vertex-connected Components	65
3.16	Minimum Arborescence	66
3.17	Chordal Graph	68
4	Math	69
4.1	BigNum	69
4.2	Belerkamp-Massey	76
4.3	Chinese Remainder Theorem	78
4.4	Matrix Determinant	79
4.5	Euler Sieve	80
4.6	Extended GCD	81
4.7	Fast Fourier Transform	82

4.8	Fast Walsh-Hadamard Transform	85
4.9	Gauss-Jordan	86
4.10	Linear Basis	87
4.11	Linear Congruence	88
4.12	Linear Recurrence	88
4.13	LU Decomposition	89
4.14	Matrix Operations	91
4.15	Miller-Rabin primality test	92
4.16	Mod-Combinatation and Mod-fact	93
4.17	Fast Number-Theoretic Transform	94
4.18	Pell's equation	96
4.19	Pollard-Rho	96
4.20	Polynomial Operations	99
4.21	Polynomial Summations	104
4.22	Prime Counting Function	106
4.23	Primitive Root	106
4.24	Segmented Sieve	107
4.25	Stirling number of the first kind	108
4.26	Stirling number of the second kind(multiple)	110
4.27	Stirling number of the second kind(single)	112
4.28	Prefix Sum of Miu	112
4.29	Prefix Sum of Phi	113
4.30	Tonelli-Shanks	115
4.31	Euclid	116
4.32	Simpson	117
4.33	Farey	117
5	Others	118
5.1	Convex Hull Trick	118
5.2	Dynamic Convex Hull Trick	118
5.3	Knuth's optimization	119
5.4	Multiple Backpack	120
5.5	Sum Over Subsets	121
5.6	Enumeration of Subsets	121
5.7	whatday	122
6	String	122
6.1	Trie	122
6.2	KMP	123
6.3	Hash Matching	124
6.4	Aho-Corasick Automaton	124
6.5	Suffix Array	127
6.6	SA-IS	128
6.7	Manacher	130
6.8	Suffix Automaton	130

1 DataStructures

1.1 Heap

```
#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 heap
{
    priority_queue<int> q1,q2;
    void push(int x) {q1.push(x);}
    void erase(int x) {q2.push(x);}
    int top()
    {
        while(q2.size()&&q1.top()==q2.top()) q1.pop(),q2.pop();
        return q1.top();
    }
    void pop()
    {
        while(q2.size()&&q1.top()==q2.top()) q1.pop(),q2.pop();
        q1.pop();
    }
    int size()
    {
        return q1.size()-q2.size();
    }
};
int main()
{
    return 0;
}
```

1.2 Fenwick Tree

```
#include<bits/stdc++.h>
#define MAXN 100000
#define MAXLOGN 20
#define INF 1000000000
using namespace std;
int bit[2*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 bisearch(int v)
{
    int sum=0,pos=0;
    for(int i=MAXLOGN;i>=0;i--)
    {
        if(pos+(1<<i)<=n&&sum+bit[pos+(1<<i)]<v)
        {
            sum+=bit[pos+(1<<i)];
            pos+=(1<<i);
        }
    }
    return pos+1;
}
int main()
{
    return 0;
}

```

1.3 Mo's algorithm

```

#include<bits/stdc++.h>
#define MAXN 100005
#define MAXM 100005
using namespace std;
struct query
{
    int l,r,id;
}save[MAXN];
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;
    if(x.l/block&1) return x.r>y.r; else return x.r<y.r;
}
void add(int pos)
{
    if(cnt[a[pos]]==a[pos]) ans--;
    cnt[a[pos]]++;
    if(cnt[a[pos]]==a[pos]) ans++;
}
void del(int pos)
{
    if(cnt[a[pos]]==a[pos]) ans--;
    cnt[a[pos]]--;
    if(cnt[a[pos]]==a[pos]) ans++;
}
void update(int cl,int cr,int l,int r)
{

```

```

        while(cr<r) add(++cr);
        while(cl>l) add(--cl);
        while(cl<l) del(cl++);
        while(cr>r) del(cr--);
    }
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;
}

```

1.4 Mo's algorithm with Queries

```

#include<bits/stdc++.h>
#define MAXN 10005
#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 query{int l,r,t,id;};
int n,m,c[MAXN],cnt[100*MAXN],res,cur,ans[MAXN];
query q[MAXN];
int tim[MAXN],pos[MAXN],pre[MAXN],val[MAXN];

```

```

int totq,totc,nowl,nowr;
const int blocks=462;
bool cmp(query x,query y)
{
    if(x.l/blocks!=y.l/blocks) return x.l<y.l;
    if(x.r/blocks!=y.r/blocks) return x.r<y.r;
    return x.t<y.t;
}
char ch[5];
void add(int p)
{
    if(!cnt[c[p]]) res++;
    cnt[c[p]]++;
}
void del(int p)
{
    cnt[c[p]]--;
    if(!cnt[c[p]]) res--;
}
void tadd(int cur)
{
    if(pos[cur]>=nowl&&pos[cur]<=nowr)
    {
        cnt[c[pos[cur]]]--;
        if(!cnt[c[pos[cur]]]) res--;
    }
    pre[cur]=c[pos[cur]];
    c[pos[cur]]=val[cur];
    if(pos[cur]>=nowl&&pos[cur]<=nowr)
    {
        if(!cnt[c[pos[cur]]]) res++;
        cnt[c[pos[cur]]]++;
    }
}
void tdel(int cur)
{
    if(pos[cur]>=nowl&&pos[cur]<=nowr)
    {
        cnt[c[pos[cur]]]--;
        if(!cnt[c[pos[cur]]]) res--;
    }
    c[pos[cur]]=pre[cur];
    if(pos[cur]>=nowl&&pos[cur]<=nowr)
    {
        if(!cnt[c[pos[cur]]]) res++;
        cnt[c[pos[cur]]]++;
    }
}
void tupd(int now)
{
    while(cur<totc&&tim[cur+1]<=now) tadd(++cur);
    while(cur>0&&tim[cur]>now) tdel(cur--);
}
void upd(int now,int l,int r)
{
    tupd(now);
    while(nowl>l) add(--nowl);
    while(nowr<r) add(++nowr);
    while(nowl<l) del(nowl++);
}

```

```

    while(nowr>r) del(nowr--);
}
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++) scanf("%d",&c[i]);
    for(int i=1;i<=m;i++)
    {
        scanf("%s",ch);
        if(ch[0]=='Q')
        {
            totq++;q[totq].id=totq;q[totq].t=i;
            scanf("%d%d",&q[totq].l,&q[totq].r);
        }
        else
        {
            totc++;tim[totc]=i;
            scanf("%d%d",&pos[totc],&val[totc]);
        }
    }
    sort(q+1,q+totq+1,cmp);
    nowl=1;nowr=0;cur=0;
    for(int i=1;i<=totq;i++)
    {
        upd(q[i].t,q[i].l,q[i].r);
        ans[q[i].id]=res;
    }
    for(int i=1;i<=totq;i++) printf("%d\n",ans[i]);
    return 0;
}

```

1.5 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<(int)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<(int)G[v].size();i++)
    {
        int to=G[v][i];
        if(to==p||to==wson[v]) continue;
        dfs2(to,v,to);
    }
}
void init()
{
    tot=0;
    memset(wson,0,sizeof(wson)); //important when multiple test cases!!!
    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-->0)
{
    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;
}

```

1.6 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;
}

```

1.7 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;
}

```

1.8 Persistent Segment Tree

```

#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 100005
#define MAXM 2000005
#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,q,tot,cnt,a[MAXN],root[MAXN];
int lson[MAXM],rson[MAXM],mx[MAXM];
void merge(int k)
{
    mx[k]=max(mx[lson[k]],mx[rson[k]]);
}
void build(int &k,int l,int r)
{
    k=++tot;
    if(l==r) {mx[k]=a[l]; return;}
    int mid=(l+r)/2;
    build(lson[k],l,mid);build(rson[k],mid+1,r);
    merge(k);
}
void insert(int &k,int last,int l,int r,int p,int v)
{
    k=++tot;
    mx[k]=mx[last];
    if(l==r) {mx[k]=v; return;}
    lson[k]=lson[last];rson[k]=rson[last];
    int mid=(l+r)/2;
    if(p<=mid) insert(lson[k],lson[last],l,mid,p,v);
    else insert(rson[k],rson[last],mid+1,r,p,v);
    merge(k);
}
int query(int &k,int l,int r,int x,int y)

```

```

{
    if(!k) return 0;
    if(l>y||r<x) return 0;
    if(l>=x&&r<=y) return mx[k];
    int mid=(l+r)/2;
    return max(query(lson[k],l,mid,x,y),query(rson[k],mid+1,r,x,y));
}
int main()
{
    scanf("%d%d",&n,&q);
    for(int i=1;i<=n;i++)
        scanf("%d",&a[i]);
    build(root[++cnt],1,n);
    for(int i=1;i<=q;i++)
    {
        int type,k,x,y;
        scanf("%d%d%d",&type,&k,&x,&y);
        if(type==1) insert(root[++cnt],root[k],1,n,x,y);
        else printf("%d\n",query(root[k],1,n,x,y));
    }
    return 0;
}

```

1.9 Persistent DSU

```

#include<bits/stdc++.h>
#define MAXN 100005
#define MAXM 2000005
#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,m,tot,root[MAXN];
int lson[MAXM],rson[MAXM],p[MAXM],rk[MAXM];
void build(int &k,int l,int r)
{
    k=++tot;
    if(l==r) {p[k]=l; return;}
    int mid=(l+r)/2;
    build(lson[k],l,mid);build(rson[k],mid+1,r);
}
void insert(int &k,int last,int l,int r,int pos,int val)
{
    k=++tot;
    if(l==r) {p[k]=val; rk[k]=rk[last]; return;}
    lson[k]=lson[last];rson[k]=rson[last];
    int mid=(l+r)/2;
    if(pos<=mid) insert(lson[k],lson[last],l,mid,pos,val);
    else insert(rson[k],rson[last],mid+1,r,pos,val);
}
int query(int k,int l,int r,int pos)
{
    if(l==r) return k;
    int mid=(l+r)/2;

```



```

        if(pos<=mid) return query(lson[k],l,mid,pos);
        else return query(rson[k],mid+1,r,pos);
    }
void add(int k,int l,int r,int pos)
{
    if(l==r) {rk[k]++; return;}
    int mid=(l+r)/2;
    if(pos<=mid) add(lson[k],l,mid,pos);
    else add(rson[k],mid+1,r,pos);
}
int find(int k,int x)
{
    int q=query(k,1,n,x);
    if(x==p[q]) return q;
    return find(k,p[q]);
}
int main()
{
    return 0;
}

```

1.10 Persistent Trie

```

#include<bits/stdc++.h>
#define MAXN 100005
#define MAXM 2000005
#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,k,a[MAXN],tot;
int trie[MAXM][2],root[MAXN],sz[MAXM];
int newnode()
{
    ++tot;
    trie[tot][0]=trie[tot][1]=0;
    return tot;
}
void insert(int u,int v,int x)
{
    int now1=root[u]=newnode(),now2=root[v];
    for(int i=18;i>=0;i--)
    {
        int id=(x>>i)&1;
        trie[now1][id]=newnode();
        trie[now1][!id]=trie[now2][!id];
        now1=trie[now1][id];now2=trie[now2][id];
        sz[now1]=sz[now2]+1;
    }
}
int query(int l,int r,int x)
{
    int res=0;
    int now1=root[r+1],now2=root[l];

```

```

for(int i=18;i>=0;i--)
{
    int id=(x>>i)&1;
    if(sz[trie[now1][!id]]-sz[trie[now2][!id]]>0)
    {
        res+=(1<<i);
        id=!id;
    }
    now1=trie[now1][id];now2=trie[now2][id];
}
return res;
}
int main()
{
    return 0;
}

```

1.11 Monotone Stack

```

#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);
}

```

1.12 Monotone Deque

```

#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;
}

```

1.13 Splay

```

#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;
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 pushup(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;
    pushup(old); pushup(x);
}
inline void splay(int x,int goal=0)
{
    for(int fa;(fa=f[x])!=goal;rotate(x))
        if (f[fa]!=goal) rotate((get(x)==get(fa))?fa:x);
    if(goal==0) 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]++; pushup(now); pushup(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;
            pushup(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 now,int k)
{
    while(now)
    {
        if(k<=size[ch[now][0]]) now=ch[now][0];
        else if(k<=size[ch[now][0]]+cnt[now]) return key[now];
        else k-=size[ch[now][0]]+cnt[now],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]--; pushup(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);
    pushup(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(root,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;
    }
}
}

```

1.14 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;
}

```

1.15 Link-Cut 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;
#define lc t[x].ch[0]
#define rc t[x].ch[1]
#define pa t[x].fa
typedef long long ll;
namespace lct
{
    struct meow{int ch[2], fa, rev, sum, w;} t[MAXN];
    inline int wh(int x) {return t[pa].ch[1] == x;}
    inline int isr(int x) {return t[pa].ch[0] != x && t[pa].ch[1] != x;}
    inline void update(int x) {t[x].sum = t[lc].sum ^ t[rc].sum ^ t[x].w;}
    inline void rever(int x) {t[x].rev ^= 1; swap(lc, rc);}
    inline void pushdown(int x)
    {
        if(t[x].rev) {
            if(lc) rever(lc);
            if(rc) rever(rc);
            t[x].rev = 0;
        }
    }
    void pd(int x) {if(!isr(x)) pd(pa); pushdown(x);}
    inline void rotate(int x) {
        int f=t[x].fa, g=t[f].fa, c=wh(x);
        if(!isr(f)) t[g].ch[wh(f)]=x; t[x].fa=g;
        t[f].ch[c] = t[x].ch[c^1]; t[t[f].ch[c]].fa=f;
        t[x].ch[c^1] = f; t[f].fa=x;
        update(f); update(x);
    }
    inline void splay(int x) {
        pd(x);
        for(; !isr(x); rotate(x))
            if(!isr(pa)) rotate( wh(pa)==wh(x) ? pa : x );
    }
}

```



```

inline void access(int x) {
    for(int y=0; x; y=x, x=pa) splay(x), rc=y, update(x);
}
inline void maker(int x) {
    access(x); splay(x); rever(x);
}
inline int findr(int x)
{
    access(x); splay(x);
    while(lc) pushdown(x), x=lc; return x;
}
inline void link(int x, int y) {
    maker(x); t[x].fa=y;
}
inline void cut(int x, int y) {
    maker(x); access(y); splay(y);
    t[x].fa = t[y].ch[0] = 0; update(y);
}
inline void split(int x, int y) {
    maker(x); access(y); splay(y);
}
} using lct::findr;

int n, Q, op, x, y;
int main()
{
    scanf("%d%d", &n, &Q);
    for(int i=1; i<=n; i++) scanf("%d", &lct::t[i].w);
    for(int i=1; i<=Q; i++)
    {
        scanf("%d%d%d", &op, &x, &y);
        if(op==0) lct::split(x, y), printf("%d\n", lct::t[y].sum);
        if(op==1) if(findr(x) != findr(y)) lct::link(x, y);
        if(op==2) if(findr(x) == findr(y)) lct::cut(x, y);
        if(op==3) lct::t[x].w = y, lct::splay(x);
    }
}

```

1.16 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;
}

```

1.17 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;
}

```

1.18 DSU on 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;
int n,t,c[MAXN],sz[MAXN],st[MAXN],ed[MAXN],cnt[MAXN],rev[MAXN];
vector<int> G[MAXN];
void dfs(int v,int p)
{
    st[v]=++t;rev[t]=v;
    sz[v]=1;
    for(int i=0;i<(int)G[v].size();i++)
    {
        if(G[v][i]==p) continue;
        dfs(G[v][i],v);
        sz[v]+=sz[G[v][i]];
    }
    ed[v]=t;
    return;
}
void dfs2(int v,int p,bool keep)
{
    int mx=-1,wson=-1;
    for(int i=0;i<(int)G[v].size();i++)
    {
        int to=G[v][i];
        if(to==p) continue;
        if(sz[to]>mx) {mx=sz[to]; wson=to;}
    }
    for(int i=0;i<(int)G[v].size();i++)
    {
        int to=G[v][i];
        if(to==p||to==wson) continue;
        dfs2(to,v,0);
    }
    if(wson!=-1) dfs2(wson,v,1);
    for(int i=0;i<(int)G[v].size();i++)
    {
        int to=G[v][i];
        if(to==p||to==wson) continue;
        for(int j=st[to];j<=ed[to];j++)

```

```

        cnt[c[rev[j]]]++;
    }
    cnt[c[v]]++;
    //answer queries here
    if(!keep)
    {
        for(int j=st[v];j<=ed[v];j++)
            cnt[c[rev[j]]]--;
    }
}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%d",&c[i]);
    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);
    }
    dfs(1,0);
}

```

1.19 Virtual Tree

```

#include<bits/stdc++.h>
#define MAXV 100005
#define INF 1000000000
#define MAXLOGV 20
using namespace std;
struct edge
{
    int to,cost;
};
vector<edge> G[MAXV];
vector<int> vt[MAXV];
int parent[MAXLOGV][MAXV];
int depth[MAXV],dfn[MAXV],dis[MAXV],st[MAXV];
int n,q,tot;
void add_edge(int from,int to)
{
    vt[from].push_back(to);
}
bool cmp(int x,int y)
{
    return dfn[x]<dfn[y];
}
void dfs(int v,int p,int d,int minx)
{
    dfn[v]=++tot;
    dis[v]=minx;
    parent[0][v]=p;
    depth[v]=d;
    for(int i=0;i<(int)G[v].size();i++)
        if(G[v][i].to!=p) dfs(G[v][i].to,v,d+1,min(minx,G[v][i].cost));
}
void init(int V)

```

```

{
    dfs(1,-1,0,INF);
    for(int k=0;k+1<MAXLOGV;k++)
    {
        for(int v=1;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 build_vtree(vector<int> &a)
{
    sort(a.begin(),a.end(),cmp);
    a.erase(unique(a.begin(),a.end()),a.end());
    assert(a.size()>0);
    int t=0;
    st[t++]=a[0];
    vector<int> newly;newly.clear();
    for(int i=1;i<(int)a.size();i++)
    {
        if(t==0) {st[t++]=a[i]; continue;}
        int l=lca(a[i],st[t-1]);
        while(t>1&&dfn[st[t-2]]>=dfn[l]) add_edge(st[t-2],st[t-1]),t--;
        if(l!=st[t-1]) {add_edge(l,st[t-1]),st[t-1]=l; newly.push_back(l);}
        st[t++]=a[i];
    }
    while(t>1) add_edge(st[t-2],st[t-1]),t--;
    for(auto it:newly) a.push_back(it);
    return st[0];
}
int main()
{
    return 0;
}

```

1.20 Centroid Decomposition

```

#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 edge{int to,cost;};
int N,K;
vector<edge> G[MAXN];
bool centroid[MAXN];
int sz[MAXN],deep[MAXN],d[MAXN];
int ans;
P getroot(int v,int p,int t)//search_centroid
{
    P res=P(INT_MAX,-1);
    int m=0;
    sz[v]=1;
    for(int i=0;i<(int)G[v].size();i++)
    {
        int to=G[v][i].to;
        if(to==p||centroid[to]) continue;
        res=min(res,getroot(to,v,t));
        m=max(m,sz[to]);
        sz[v]+=sz[to];
    }
    m=max(m,t-sz[v]);
    res=min(res,P(m,v));
    return res;
}
void getdeep(int v,int p)//enumerate path
{
    deep[++deep[0]]=d[v];
    for(int i=0;i<(int)G[v].size();i++)
    {
        int to=G[v][i].to;
        if(to==p||centroid[to]) continue;
        d[to]=d[v]+G[v][i].cost;
        getdeep(to,v);
    }
}
int cal(int v,int cost)
{
    d[v]=cost;deep[0]=0;
    getdeep(v,0);
    sort(deep+1,deep+deep[0]+1);
    int l=1,r=deep[0],sum=0;
    while(l<r)
    {
        if(deep[l]+deep[r]<=K)
        {
            sum+=r-l;
            l++;
        }
        else r--;
    }
}

```

```

        return sum;
    }
    void solve(int v)
    {
        ans+=cal(v,0);
        centroid[v]=true;
        for(int i=0;i<(int)G[v].size();i++)
        {
            int to=G[v][i].to,cost=G[v][i].cost;
            if(centroid[to]) continue;
            ans-=cal(to,cost);
            int rt=getroot(to,v,sz[to]).S;
            solve(rt);
        }
    }
    void ac()
    {
        ans=0;
        int rt=getroot(1,0,N).S;
        solve(rt);
        printf("%d\n",ans);
    }
    int main()
    {
        while(scanf("%d%d",&N,&K)==2)
        {
            if(!N&&!K) break;
            for(int i=1;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);
                G[x].push_back((edge){y,z});
                G[y].push_back((edge){x,z});
            }
            memset(centroid,false,sizeof(centroid));
            ac();
        }
        return 0;
    }
}

```

2 Geometry

2.1 Geometry All-in-One

```

#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;
}
// [ ] 0 (n)
bool ahas( Point &_Off)
{
    int ret = 0;
    double inv = 1e20;
    Line l = Line( _Off, Point( -inv ,_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;
}

// 0 (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 = getdis2(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          [          ]
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;
}

//-----
//
Circle operator-(Circle &_Off) const [_Off]
{
    //
    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;
}

//
Circle operator+(Circle &_Off) const [_Off]
{
    //
    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), ang0 = atan2(_Off.y - c.y, _Off.x - c.x);
    Point pl = Point(c.x + r * cos(ang0 + angp), c.y + r * sin(ang0 + angp)),

```

```

        pr = Point(c.x + r * cos(angp - angp), c.y + r * sin(angp - 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 angp = atan2(yp - c.y, xp - c.x);
        double angp = acos(td / r);

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

```

2.2 Stereometry

```

#pragma GCC optimize(3)
#include <bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long double T;
typedef long double db;
typedef long long ll;
typedef pair<int,int> P;
const T PI=acos(-1.0);
const T eps=1e-10;

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

class Point
{
public:
    T x,y,z;
    Point(){}
    Point(T tx,T ty,T tz) {x=tx,y=ty,z=tz;}
    db dist2(Point p) {return (x-p.x)*(x-p.x)+(y-p.y)*(y-p.y)+(z-p.z)*(z-p.z);}
}

```

```

    db dist(Point p) {return sqrt(dist2(p));}
    Point operator+(Point p) {return {x+p.x,y+p.y,z+p.z};}
    Point operator-(Point p) {return {x-p.x,y-p.y,z-p.z};}
    Point operator*(T d) {return {x*d,y*d,z*d};}
    Point operator/(T d) {return {x/d,y/d,z/d};}
    bool operator==(Point p) {return tie(x,y,z)==tie(p.x,p.y,p.z);}
    bool operator!=(Point p) {return !operator==(p);}
    const bool operator<(Point &p) const {return tie(x,y,z)<tie(p.x,p.y,p.z);}
};

Point zero{0,0,0};
T operator|(Point v, Point w) {return v.x*w.x + v.y*w.y + v.z*w.z;}
T sq(Point v) {return v|v;}
db abs(Point v) {return sqrt(sq(v));}
Point unit(Point v) {return v/abs(v);}
db angle(Point v, Point w)
{
    db cosTheta=(v|w)/abs(v)/abs(w);
    return acos(max((db)-1.0,min((db)1.0,cosTheta)));
}
Point operator*(Point v,Point w) {return {v.y*w.z-v.z*w.y,v.z*w.x-v.x*w.z,v.x*w.y-v.y*w.x};}
T orient(Point p, Point q, Point r, Point s) {return (q-p)*(r-p)|(s-p);}
T orientByNormal(Point p, Point q, Point r, Point n) {return (q-p)*(r-p)|n;}
class Plane
{
public:
    Point n; T d;
    Plane(){}
    Plane(Point n,T d) : n(n), d(d) {}
    Plane(Point n, Point p) : n(n), d(n|p) {}
    Plane(Point p, Point q, Point r) : Plane((q-p)*(r-p), p) {}
    T side(Point p) {return n|p-d;}
    db dist(Point p) {return abs(side(p))/abs(n);}
    Plane translate(Point t) {return {n,d+n|t};}
    Plane shiftup(db dist) {return {n,d+dist*abs(n)};}
    Point proj(Point p) {return p-n*side(p)/sq(n);}
    Point refl(Point p) {return p-n*2*side(p)/sq(n);}
};

class Line
{
public:
    Point d,o;
    Line(){}
    Line(Point p,Point q):d(q-p),o(p){}
    Line(Plane p1,Plane p2)
    {
        d=p1.n*p2.n;
        o=(p2.n*p1.d-p1.n*p2.d)*d/sq(d);
    }
    db dist2(Point p) {return sq(d*(p-o))/sq(d);}
    db dist(Point p) {return sqrt(dist2(p));}
    bool cmpProj(Point p,Point q) {return (d|p)<(d|q);}
    Point proj(Point p) {return o+d*(d|(p-o))/sq(d);}
    Point refl(Point p) {return proj(p)*2-p;}
    Point inter(Plane p) {return o-d*p.side(o)/(d|p.n);}
};

db dist(Line l1,Line l2)
{

```

```

    Point n=l1.d*l2.d;
    if(n==zero) return l1.dist(l2.o);
    return abs((l2.o-l1.o)|n)/abs(n);
}

Point closestOnL1(Line l1,Line l2)
{
    Point n2 = l2.d*(l1.d*l2.d);
    return l1.o+l1.d*((l2.o-l1.o)|n2)/(l1.d|n2);
}

db angle(Plane p1,Plane p2)
{
    return angle(p1.n,p2.n);
}

bool is_parallel(Plane p1,Plane p2)
{
    return p1.n*p2.n==zero;
}

bool is_perpendicular(Plane p1,Plane p2)
{
    return (p1.n|p2.n)==0;
}

db angle(Line l1,Line l2)
{
    return angle(l1.d,l2.d);
}

bool is_parallel(Line l1,Line l2)
{
    return l1.d*l2.d==zero;
}

bool is_perpendicular(Line l1,Line l2)
{
    return (l1.d|l2.d)==0;
}

db angle(Plane p, Line l)
{
    return PI/2-angle(p.n,l.d);
}

bool is_parallel(Plane p,Line l)
{
    return (p.n|l.d)==0;
}

bool is_perpendicular(Plane p,Line l)
{
    return p.n*l.d==zero;
}

Line perpthrough(Plane p,Point o) {return Line(o,o+p.n);}

Plane perpthrough(Line l,Point o) {return Plane(l.d,o);}

```

```

Point vectorArea2(vector<Point> p)
{
    Point S=zero;
    for(int i=0,n=p.size();i<n;i++) S=S+p[i]*p[(i+1)%n];
    return S;
}

db area(vector<Point> p) {return abs(vectorArea2(p))/2.0;}

class Polyhedron
{
public:
    Polyhedron(){}
    vector<vector<Point> > faces;
    void clear(){faces.clear();}
    db surface_area()
    {
        db S=0;
        for(auto f:faces) S=S+area(f);
        return S;
    }

    struct edge{int v;bool same;};
    void reorient()
    {
        int n=faces.size();
        vector<vector<edge> > G(n);
        map<pair<Point,Point>, int> es;
        for(int u=0;u<n;u++)
        {
            for(int i=0,m=(int)faces[u].size();i<m;i++)
            {
                Point p=faces[u][i],q=faces[u][(i+1)%m];
                if(es.count({p,q}))
                {
                    int v=es[{p,q}];
                    G[u].push_back({v,true});G[v].push_back({u,true});
                }
                else if(es.count({q,p}))
                {
                    int v=es[{q,p}];
                    G[u].push_back({v,false});G[v].push_back({u,false});
                }
                else es[{p,q}]=u;
            }
        }
        vector<bool> vis(n,false),flip(n);
        flip[0]=false;
        queue<int> q;q.push(0);
        while(!q.empty())
        {
            int u=q.front();q.pop();
            for(edge e:G[u])
            {
                if(!vis[e.v])
                {
                    vis[e.v]=true;
                    flip[e.v]=flip[u]^e.same;
                }
            }
        }
    }
};

```

```

        q.push(e.v);
    }
}
}
for(int u=0;u<n;u++)
    if(flip[u])
        reverse(faces[u].begin(),faces[u].end());
}

db volume()
{
    double ans=0.0;
    for(auto f:faces) ans+=(vectorArea2(f)|f[0]);
    return abs(ans)/6.0;
}
};

struct fac
{
    int a,b,c;
    bool ok;
};

struct T3dhull
{
    int n;
    Point ply[MAXN];
    int trianglecnt;
    fac tri[MAXN];
    int vis[MAXN][MAXN];
    double area(Point a,Point b,Point c){return abs((b-a)*(c-a));}
    double volume(Point a,Point b,Point c,Point d){return (b-a)*(c-a)|(d-a);}
    double ptoplane(Point &p,fac &f)
    {
        Point m=ply[f.b]-ply[f.a],n=ply[f.c]-ply[f.a],t=p-ply[f.a];
        return (m*n)|t;
    }
    void deal(int p,int a,int b)
    {
        int f=vis[a][b];
        fac add;
        if(tri[f].ok)
        {
            if((ptoplane(ply[p],tri[f]))>eps) dfs(p,f);
            else
            {
                add.a=b,add.b=a,add.c=p,add.ok=1;
                vis[p][b]=vis[a][p]=vis[b][a]=trianglecnt;
                tri[trianglecnt++]=add;
            }
        }
    }
    void dfs(int p,int cnt)
    {
        tri[cnt].ok=0;
        deal(p,tri[cnt].b,tri[cnt].a);
        deal(p,tri[cnt].c,tri[cnt].b);
        deal(p,tri[cnt].a,tri[cnt].c);
    }
    bool same(int s,int e)
    {

```



```

    Point a=ply[tri[s].a],b=ply[tri[s].b],c=ply[tri[s].c];
    return fabs(volume(a,b,c,ply[tri[e].a]))<eps
        &&fabs(volume(a,b,c,ply[tri[e].b]))<eps
        &&fabs(volume(a,b,c,ply[tri[e].c]))<eps;
}
void construct()
{
    int i,j;
    trianglecnt=0;
    if(n<4) return ;
    bool tmp=true;
    for(i=1;i<n;i++) if((abs(ply[0]-ply[i]))>eps){ swap(ply[1],ply[i]); tmp=false; break;}
    if(tmp) return;
    tmp=true;
    for(i=2;i<n;i++)if((abs((ply[0]-ply[1])*(ply[1]-ply[i])))>eps){swap(ply[2],ply[i]);
        tmp=false; break;}
    if(tmp) return ;
    tmp=true;
    for(i=3;i<n;i++)
        if(fabs((ply[0]-ply[1])*(ply[1]-ply[2])|(ply[0]-ply[i]))>eps){swap(ply[3],ply[i]);
            tmp=false; break;}
    if(tmp) return ;
    fac add;
    for(i=0;i<4;i++)// (4 ply [0],ply[1],ply[2],ply[3])
    {
        add.a=(i+1)%4,add.b=(i+2)%4,add.c=(i+3)%4,add.ok=1;
        if((ptoplane(ply[i],add))>0)
            swap(add.b,add.c);//
        vis[add.a][add.b]=vis[add.b][add.c]=vis[add.c][add.a]=trianglecnt;//
        tri[trianglecnt++]=add;
    }
    for(i=4;i<n;i++)//
    {
        for(j=0;j<trianglecnt;j++)// 3 ( i , j )
        {
            if(tri[j].ok&&(ptoplane(ply[i],tri[j]))>eps)//
            {
                dfs(i,j);
                break;// ( )
            }
        }
    }
    int
        cnt=trianglecnt;// tri [i].ok=0
    trianglecnt=0;
    for(i=0;i<cnt;i++)
    {
        if(tri[i].ok)
            tri[trianglecnt++]=tri[i];
    }
}
double area()//
{
    double ret=0;
    for(int i=0;i<trianglecnt;i++)
        ret+=area(ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]);
    return ret/2;
}
double volume()//

```

```

{
    Point p(0,0,0);
    double ret=0;
    for(int i=0;i<trianglecnt;i++)
        ret+=volume(p,ply[tri[i].a],ply[tri[i].b],ply[tri[i].c]);
    return fabs(ret/6);
}
int facetri() {return trianglecnt;}//
int facepolygon()//
{
    int ans=0,i,j,k;
    for(i=0;i<trianglecnt;i++)
    {
        for(j=0,k=1;j<i;j++)
        {
            if(same(i,j)) {k=0;break;}
        }
        ans+=k;
    }
    return ans;
}
}hull;

```

```

-----

        tmljs1988 CSDN
        https          ://blog.csdn.net/tmljs1988/article/details/7268944?utm_source=copy
T point_to_segment(Point &p1,Point &p2,Point &p3)
{
    T l=0.0,r=1.0,ans1,ans2;
    while(r-l>1e-14)
    {
        T dis=(r-l)/3.0;
        T lmid=l+dis,rmid=l+2.0*dis;
        Point Q=p2+((p3-p2)*lmid),R=p2+((p3-p2)*rmid);
        ans1=p1|Q;ans2=p1|R;
        if(ans1<ans2) r=rmid; else l=lmid;
    }
    return sqrt(min(ans1,ans2));
}
T segment_dist(Point &p1, Point &p2, Point &p3, Point &p4)
{
    T l=0.0,r=1.0,ans1,ans2;
    while(r-l>1e-14)
    {
        T dis=(r-l)/3.0;
        T lmid=l+dis,rmid=l+2.0*dis;
        Point p=p1+((p2-p1)*lmid),q=p1+((p2-p1)*rmid);
        ans1=point_to_segment(p,p3,p4);ans2=point_to_segment(q,p3,p4);
        if(ans1<ans2) r=rmid; else l=lmid;
    }
    return min(ans1,ans2);
}
int main()
{
    return 0;
}

```

3 Graph

3.1 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;
}
```

3.2 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;
}
```

3.3 Korasaju

```
#include<bits/stdc++.h>
#define MAXN 100005
using namespace std;
int n;
vector<int> G[MAXN];
vector<int> rG[MAXN];
vector<int> vs;
bool used[MAXN];
int cmp[MAXN];
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<(int)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<(int)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=1;v<=n;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;
}
```

3.4 LCA with binary lifting

```
#include<bits/stdc++.h>
#define MAXN 100005
#define MAXLOGN 20
```

```

using namespace std;
vector<int> G[MAXN];
int pa[MAXLOGN][MAXN];
int depth[MAXN];
int n,q;
void dfs(int v,int p,int d)
{
    pa[0][v]=p;
    depth[v]=d;
    for(int i=0;i<(int)G[v].size();i++)
        if(G[v][i]!=p) dfs(G[v][i],v,d+1);
}
void init(int V)
{
    dfs(1,0,0);
    for(int k=0;k+1<MAXLOGN;k++)
    {
        for(int v=1;v<=V;v++)
        {
            if(pa[k][v]<0) pa[k+1][v]=-1;
            else pa[k+1][v]=pa[k][pa[k][v]];
        }
    }
}
int get(int v,int x)
{
    for(int k=0;k<MAXLOGN;k++)
        if((x>>k)&1)
            v=pa[k][v];
    return v;
}
int lca(int u,int v)
{
    if(depth[u]>depth[v]) swap(u,v);
    v=get(v,depth[v]-depth[u]);
    if(u==v) return u;
    for(int k=MAXLOGN-1;k>=0;k--)
    {
        if(pa[k][u]!=pa[k][v])
        {
            u=pa[k][u];
            v=pa[k][v];
        }
    }
    return pa[0][u];
}
int dis(int u,int v)
{
    return depth[u]+depth[v]-2*depth[lca(u,v)];
}
int main()
{
    return 0;
}

```

3.5 LCA with range minimum query

```

#include<bits/stdc++.h>
#define MAXN 100005
#define MAXLOGN 22
#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,q;
int st[MAXLOGN][2*MAXN];
vector<int> G[MAXN];
int vs[MAXN*2-1];
int depth[MAXN*2-1];
int id[MAXN];
void dfs(int v,int p,int d,int &k)
{
    id[v]=k;
    vs[k]=v;
    depth[k++]=d;
    for(int i=0;i<(int)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=1;i<=n;++i) st[0][i]=i;
    for(int i=1;1<<i<n;++i)
        for(int j=1;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(1,0,0,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()
{
    return 0;
}

```

3.6 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;
}

```

3.7 Min-cost 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;
}

```

3.8 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[MAXN_V];
int match[MAXN_V];
bool used[MAXN_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;
}

```

3.9 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=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;
}

```

3.10 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;
}

```

3.11 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 + 1]);
    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;
}

```

3.12 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;
    label[t]=t;sdom[t]=t;dsu[t]=t;
    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);
    }
    dfs(1);n=t;
    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;
}

```

3.13 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;
}

```

3.14 Edge-connected Components

```

#include<bits/stdc++.h>
#define MAXN 100005
#define MAXM 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 Edge
{
    int u,v;
    Edge(int u=0,int v=0):u(u),v(v){}
}e[MAXM];
int n,m,stamp,dfn[MAXN],low[MAXN],bccno[MAXN],bcc_cnt;
vector<int> vec[MAXN],bcc[MAXN];
bool g[MAXN][MAXN],isbridge[MAXM];

void tarjan(int tot,int fa)
{
    int tmp;
    dfn[tot]=low[tot]=++stamp;
    for(int i=0;i<(int)vec[tot].size();i++)
    {
        tmp=e[vec[tot][i]].v;
        if(!dfn[tmp])
        {
            tarjan(tmp,tot);
            low[tot]=min(low[tot],low[tmp]);
            if(low[tmp]>dfn[tot])
                isbridge[vec[tot][i]]=isbridge[vec[tot][i]^1]=1;
        }
        else if(dfn[tmp]<dfn[tot] && tmp!=fa)
        {
            low[tot]=min(low[tot], dfn[tmp]);
        }
    }
}

void dfs(int tot)
{
    dfn[tot]=1;
    bccno[tot]=bcc_cnt;
    for(int i=0;i<(int)vec[tot].size();i++)
    {
        int tmp=vec[tot][i];
        if(isbridge[tmp])
            continue;
        if(!dfn[e[tmp].v])
        {
            dfs(e[tmp].v);
        }
    }
}

void find_ebcc(){
    bcc_cnt=stamp=0;
    memset(dfn,0,sizeof(dfn));
    memset(low,0,sizeof(low));
    memset(isbridge,0,sizeof(isbridge));
    memset(bccno,0,sizeof(bccno));
    memset(bcc,0,sizeof(bcc));
    for(int i=1;i<=n;i++)

```

```

        if(!dfn[i])
            tarjan(i, -1);
memset(dfn,0,sizeof(dfn));
for(int i=1;i<=n;i++)
{
    if(!dfn[i])
    {
        bcc_cnt++;
        dfs(i);
    }
}
}
int main()
{
    return 0;
}

```

3.15 Vertex-connected Components

```

#include<bits/stdc++.h>
#define MAXN 100005
#define MAXM 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 Edge{
    int u,v;
    Edge(int u=0,int v=0):u(u),v(v){}
}e[MAXN];
int n,m,stamp,dfn[MAXN],low[MAXN],iscut[MAXN],bccno[MAXN];
int scnt,st[MAXN],bcc_cnt;
vector<int> vec[MAXN],bcc[MAXN];

void tarjan(int tot,int fa)
{
    int child=0,tmp;
    dfn[tot]=low[tot]=++stamp;
    for(int i=0;i<vec[tot].size();i++)
    {
        tmp=e[vec[tot][i]].v;
        if(!dfn[tmp])
        {
            st[++scnt]=vec[tot][i],child++;
            tarjan(tmp,tot);
            low[tot]=min(low[tot],low[tmp]);
            if(low[tmp]>=dfn[tot])
            {
                iscut[tot]=1;
                bcc[++bcc_cnt].clear();
                while(1)
                {
                    int num=st[scnt--];
                    if(bccno[e[num].u]!=bcc_cnt)

```

```

        {
            bcc[bcc_cnt].push_back(e[num].u);
            bccno[e[num].u]=bcc_cnt;
        }
        if(bccno[e[num].v]!=bcc_cnt)
        {
            bcc[bcc_cnt].push_back(e[num].v);
            bccno[e[num].v]=bcc_cnt;
        }
        if(e[num].u==tot && e[num].v==tmp)
            break;
    }
}
}
else if(dfn[tmp]<dfn[tot] && tmp!=fa)
{
    st[++scnt]=vec[tot][i];
    low[tot]=min(low[tot], dfn[tmp]);
}
}
if(fa<0 && child==1)
    iscut[tot]=0;
}

void find_bcc()
{
    //      bccno
    memset(dfn,0,sizeof(dfn));
    memset(low,0,sizeof(low));
    memset(iscut,0,sizeof(iscut));
    memset(bccno,0,sizeof(bccno));
    memset(bcc,0,sizeof(bcc));
    stamp=scnt=bcc_cnt=0;
    for(int i=1;i<=n;i++)
        if(!dfn[i])
            tarjan(i,-1);
}
int main()
{
    return 0;
}

```

3.16 Minimum Arborescence

```

#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,k,a[MAXN];
namespace ZL
{
    const int N=100010,M=100010,inf=1e9;

```

```

struct edge
{
    int u,v,w,use,id;
}b[M],a[2000100];
int n,m,ans,pre[N],id[N],vis[N],root,In[N],h[N],len,way[M];
void init(int _n,int _root)
{
    n=_n; m=0; b[0].w=inf; root=_root; ans=0;
}
void add(int u,int v,int w)
{
    b[++m]=(edge){u,v,w,0,m};
    a[m]=b[m];
}
int work()
{
    len=m;
    for (;;)
    {
        for (int i=1;i<=n;i++){pre[i]=0; In[i]=inf; id[i]=0; vis[i]=0; h[i]=0;}
        for (int i=1;i<=m;i++)
            if (b[i].u!=b[i].v&&b[i].w<In[b[i].v])
            {
                pre[b[i].v]=b[i].u; In[b[i].v]=b[i].w; h[b[i].v]=b[i].id;
            }
        for (int i=1;i<=n;i++) if (pre[i]==0&&i!=root) return 0;
        int cnt=0; In[root]=0;
        for (int i=1;i<=n;i++)
        {
            if (i!=root) a[h[i]].use++;
            int now=i; ans+=In[i];
            while (vis[now]==0&&now!=root)
            {
                vis[now]=i; now=pre[now];
            }
            if (now!=root&&vis[now]==i)
            {
                cnt++; int kk=now;
                while (1)
                {
                    id[now]=cnt; now=pre[now];
                    if (now==kk) break;
                }
            }
        }
        if (cnt==0) return 1;
        for (int i=1;i<=n;i++) if (id[i]==0) id[i]=++cnt;
        for (int i=1;i<=m;i++)
        {
            int k1=In[b[i].v]; int k2=b[i].v;
            b[i].u=id[b[i].u]; b[i].v=id[b[i].v];
            if (b[i].u!=b[i].v)
            {
                b[i].w-=k1; a[++len].u=b[i].id; a[len].v=h[k2];
                b[i].id=len;
            }
        }
        n=cnt;
        root=id[root];
    }
}

```

```

    }
    return 1;
}
void getway()
{
    for (int i=1;i<=m;i++) way[i]=0;
    for (int i=len;i>m;i--)
    {
        a[a[i].u].use+=a[i].use; a[a[i].v].use-=a[i].use;
    }
    for (int i=1;i<=m;i++) way[i]=a[i].use;
}
}
int main()
{
    return 0;
}

```

3.17 Chordal Graph

```

/*****
> File Name: ChordalGraph.cpp
> Author: Roundgod
> Mail: wcysai@foxmail.com
> Created Time: 2018-10-31 15:49:59
*****/

#pragma GCC optimize(3)
#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,m;
vector<int> G[MAXN];
int h[MAXN];
bool vis[MAXN];
vector<int> peo;
priority_queue<P> pq;
void MCS()
{
    memset(vis,0,sizeof(vis));
    memset(h,0,sizeof(h));
    for(int i=1;i<=n;i++) pq.push(P(0,i));
    for(int i=n;i>=1;i--)
    {
        while(1)
        {
            P p=pq.top();pq.pop();
            if(vis[p.S]) continue;
            peo.push_back(p.S);vis[p.S]=true;
            for(auto to:G[p.S])
            {

```

```

        if(vis[to]) continue;
        h[to]++;pqe.push(P(h[to],to));
    }
    break;
}
}
reverse(peo.begin(),peo.end());
}
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=0;i<m;i++)
    {
        int u,v;scanf("%d%d",&u,&v);
        G[u].push_back(v);G[v].push_back(u);
    }
    MCS();
    return 0;
}

```

4 Math

4.1 BigInt

```

const int base = 1000000000;
const int base_digits = 9;
struct bigint {
    vector<int> a;
    int sign;
    /*<arpa>*/
    int size(){
        if(a.empty())return 0;
        int ans=(a.size()-1)*base_digits;
        int ca=a.back();
        while(ca)
            ans++,ca/=10;
        return ans;
    }
    bigint operator ^(const bigint &v){
        bigint ans=1,a=*this,b=v;
        while(!b.isZero()){
            if(b%2)
                ans*=a;
            a*=a,b/=2;
        }
        return ans;
    }
    string to_string(){
        stringstream ss;
        ss << *this;
        string s;
        ss >> s;
        return s;
    }
    int sumof(){
        string s = to_string();

```



```

    int ans = 0;
    for(auto c : s) ans += c - '0';
    return ans;
}
/*</arpa>*/
bigint() :
    sign(1) {
}

bigint(long long v) {
    *this = v;
}

bigint(const string &s) {
    read(s);
}

void operator=(const bigint &v) {
    sign = v.sign;
    a = v.a;
}

void operator=(long long v) {
    sign = 1;
    a.clear();
    if (v < 0)
        sign = -1, v = -v;
    for (; v > 0; v = v / base)
        a.push_back(v % base);
}

bigint operator+(const bigint &v) const {
    if (sign == v.sign) {
        bigint res = v;

        for (int i = 0, carry = 0; i < (int) max(a.size(), v.a.size()) || carry; ++i) {
            if (i == (int) res.a.size())
                res.a.push_back(0);
            res.a[i] += carry + (i < (int) a.size() ? a[i] : 0);
            carry = res.a[i] >= base;
            if (carry)
                res.a[i] -= base;
        }
        return res;
    }
    return *this - (-v);
}

bigint operator-(const bigint &v) const {
    if (sign == v.sign) {
        if (abs() >= v.abs()) {
            bigint res = *this;
            for (int i = 0, carry = 0; i < (int) v.a.size() || carry; ++i) {
                res.a[i] -= carry + (i < (int) v.a.size() ? v.a[i] : 0);
                carry = res.a[i] < 0;
                if (carry)
                    res.a[i] += base;
            }
            res.trim();
        }
    }
}

```

```

        return res;
    }
    return -(v - *this);
}
return *this + (-v);
}

void operator*=(int v) {
    if (v < 0)
        sign = -sign, v = -v;
    for (int i = 0, carry = 0; i < (int) a.size() || carry; ++i) {
        if (i == (int) a.size())
            a.push_back(0);
        long long cur = a[i] * (long long) v + carry;
        carry = (int) (cur / base);
        a[i] = (int) (cur % base);
        //asm("divl %%ecx" : "=a"(carry), "=d"(a[i]) : "A"(cur), "c"(base));
    }
    trim();
}

bigint operator*(int v) const {
    bigint res = *this;
    res *= v;
    return res;
}

void operator*=(long long v) {
    if (v < 0)
        sign = -sign, v = -v;
    for (int i = 0, carry = 0; i < (int) a.size() || carry; ++i) {
        if (i == (int) a.size())
            a.push_back(0);
        long long cur = a[i] * (long long) v + carry;
        carry = (int) (cur / base);
        a[i] = (int) (cur % base);
        //asm("divl %%ecx" : "=a"(carry), "=d"(a[i]) : "A"(cur), "c"(base));
    }
    trim();
}

bigint operator*(long long v) const {
    bigint res = *this;
    res *= v;
    return res;
}

friend pair<bigint, bigint> divmod(const bigint &a1, const bigint &b1) {
    int norm = base / (b1.a.back() + 1);
    bigint a = a1.abs() * norm;
    bigint b = b1.abs() * norm;
    bigint q, r;
    q.a.resize(a.a.size());

    for (int i = a.a.size() - 1; i >= 0; i--) {
        r *= base;
        r += a.a[i];
        int s1 = r.a.size() <= b.a.size() ? 0 : r.a[b.a.size()];
        int s2 = r.a.size() <= b.a.size() - 1 ? 0 : r.a[b.a.size() - 1];
    }
}

```

```

        int d = ((long long) base * s1 + s2) / b.a.back();
        r -= b * d;
        while (r < 0)
            r += b, --d;
        q.a[i] = d;
    }

    q.sign = a1.sign * b1.sign;
    r.sign = a1.sign;
    q.trim();
    r.trim();
    return make_pair(q, r / norm);
}

bigint operator/(const bigint &v) const {
    return divmod(*this, v).first;
}

bigint operator%(const bigint &v) const {
    return divmod(*this, v).second;
}

void operator/=(int v) {
    if (v < 0)
        sign = -sign, v = -v;
    for (int i = (int) a.size() - 1, rem = 0; i >= 0; --i) {
        long long cur = a[i] + rem * (long long) base;
        a[i] = (int) (cur / v);
        rem = (int) (cur % v);
    }
    trim();
}

bigint operator/(int v) const {
    bigint res = *this;
    res /= v;
    return res;
}

int operator%(int v) const {
    if (v < 0)
        v = -v;
    int m = 0;
    for (int i = a.size() - 1; i >= 0; --i)
        m = (a[i] + m * (long long) base) % v;
    return m * sign;
}

void operator+=(const bigint &v) {
    *this = *this + v;
}

void operator-=(const bigint &v) {
    *this = *this - v;
}

void operator*=(const bigint &v) {
    *this = *this * v;
}

void operator/=(const bigint &v) {
    *this = *this / v;
}

```

```

}

bool operator<(const bigint &v) const {
    if (sign != v.sign)
        return sign < v.sign;
    if (a.size() != v.a.size())
        return a.size() * sign < v.a.size() * v.sign;
    for (int i = a.size() - 1; i >= 0; i--)
        if (a[i] != v.a[i])
            return a[i] * sign < v.a[i] * sign;
    return false;
}

bool operator>(const bigint &v) const {
    return v < *this;
}

bool operator<=(const bigint &v) const {
    return !(v < *this);
}

bool operator>=(const bigint &v) const {
    return !(*this < v);
}

bool operator==(const bigint &v) const {
    return !(*this < v) && !(v < *this);
}

bool operator!=(const bigint &v) const {
    return *this < v || v < *this;
}

void trim() {
    while (!a.empty() && !a.back())
        a.pop_back();
    if (a.empty())
        sign = 1;
}

bool isZero() const {
    return a.empty() || (a.size() == 1 && !a[0]);
}

bigint operator-() const {
    bigint res = *this;
    res.sign = -sign;
    return res;
}

bigint abs() const {
    bigint res = *this;
    res.sign *= res.sign;
    return res;
}

long long longValue() const {
    long long res = 0;
    for (int i = a.size() - 1; i >= 0; i--)
        res = res * base + a[i];
    return res * sign;
}

```

```

friend bigint gcd(const bigint &a, const bigint &b) {
    return b.isZero() ? a : gcd(b, a % b);
}
friend bigint lcm(const bigint &a, const bigint &b) {
    return a / gcd(a, b) * b;
}

void read(const string &s) {
    sign = 1;
    a.clear();
    int pos = 0;
    while (pos < (int) s.size() && (s[pos] == '-' || s[pos] == '+')) {
        if (s[pos] == '-')
            sign = -sign;
        ++pos;
    }
    for (int i = s.size() - 1; i >= pos; i -= base_digits) {
        int x = 0;
        for (int j = max(pos, i - base_digits + 1); j <= i; j++)
            x = x * 10 + s[j] - '0';
        a.push_back(x);
    }
    trim();
}

friend istream& operator>>(istream &stream, bigint &v) {
    string s;
    stream >> s;
    v.read(s);
    return stream;
}

friend ostream& operator<<(ostream &stream, const bigint &v) {
    if (v.sign == -1)
        stream << '-';
    stream << (v.a.empty() ? 0 : v.a.back());
    for (int i = (int) v.a.size() - 2; i >= 0; --i)
        stream << setw(base_digits) << setfill('0') << v.a[i];
    return stream;
}

static vector<int> convert_base(const vector<int> &a, int old_digits, int new_digits) {
    vector<long long> p(max(old_digits, new_digits) + 1);
    p[0] = 1;
    for (int i = 1; i < (int) p.size(); i++)
        p[i] = p[i - 1] * 10;
    vector<int> res;
    long long cur = 0;
    int cur_digits = 0;
    for (int i = 0; i < (int) a.size(); i++) {
        cur += a[i] * p[cur_digits];
        cur_digits += old_digits;
        while (cur_digits >= new_digits) {
            res.push_back(int(cur % p[new_digits]));
            cur /= p[new_digits];
            cur_digits -= new_digits;
        }
    }
    res.push_back((int) cur);
}

```

```

        while (!res.empty() && !res.back())
            res.pop_back();
        return res;
    }

typedef vector<long long> vll;

static vll karatsubaMultiply(const vll &a, const vll &b) {
    int n = a.size();
    vll res(n + n);
    if (n <= 32) {
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                res[i + j] += a[i] * b[j];
        return res;
    }

    int k = n >> 1;
    vll a1(a.begin(), a.begin() + k);
    vll a2(a.begin() + k, a.end());
    vll b1(b.begin(), b.begin() + k);
    vll b2(b.begin() + k, b.end());

    vll a1b1 = karatsubaMultiply(a1, b1);
    vll a2b2 = karatsubaMultiply(a2, b2);

    for (int i = 0; i < k; i++)
        a2[i] += a1[i];
    for (int i = 0; i < k; i++)
        b2[i] += b1[i];

    vll r = karatsubaMultiply(a2, b2);
    for (int i = 0; i < (int) a1b1.size(); i++)
        r[i] -= a1b1[i];
    for (int i = 0; i < (int) a2b2.size(); i++)
        r[i] -= a2b2[i];

    for (int i = 0; i < (int) r.size(); i++)
        res[i + k] += r[i];
    for (int i = 0; i < (int) a1b1.size(); i++)
        res[i] += a1b1[i];
    for (int i = 0; i < (int) a2b2.size(); i++)
        res[i + n] += a2b2[i];
    return res;
}

bigint operator*(const bigint &v) const {
    vector<int> a6 = convert_base(this->a, base_digits, 6);
    vector<int> b6 = convert_base(v.a, base_digits, 6);
    vll a(a6.begin(), a6.end());
    vll b(b6.begin(), b6.end());
    while (a.size() < b.size())
        a.push_back(0);
    while (b.size() < a.size())
        b.push_back(0);
    while (a.size() & (a.size() - 1))
        a.push_back(0), b.push_back(0);
    vll c = karatsubaMultiply(a, b);
    bigint res;

```

```

        res.sign = sign * v.sign;
        for (int i = 0, carry = 0; i < (int) c.size(); i++) {
            long long cur = c[i] + carry;
            res.a.push_back((int) (cur % 1000000));
            carry = (int) (cur / 1000000);
        }
        res.a = convert_base(res.a, 6, base_digits);
        res.trim();
        return res;
    }
};

```

4.2 Belerkamp-Massey

```

#include<bits/stdc++.h>
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
using namespace std;
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll mod=1000000007;
ll pow_mod(ll a,ll i)
{
    ll s=1;
    while(i)
    {
        if(i&1) s=s*a%mod;
        a=a*a%mod;
        i>>=1;
    }
    return s;
}
namespace linear_seq
{
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];
    vector<ll> Md;
    void mul(ll *a,ll *b,int k)
    {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if(a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for(int i=k+k-1;i>=k;i--)
            if(_c[i]) rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    int solve(ll n,VI a,VI b)//a:coefficient b:initial value b[n+1]=a[0]*b[n]+...
    {
        ll ans=0,pnt=0;
        int k=SZ(a);
        assert(SZ(a)==SZ(b));
    }
}

```

```

rep(i,0,k) _md[k-1-i]=-a[i];
_md[k]=1;
Md.clear();
rep(i,0,k) if(_md[i]!=0) Md.push_back(i);
rep(i,0,k) res[i]=base[i]=0;
res[0]=1;
while((lll<<pnt)<=n) pnt++;
for(int p=pnt;p>=0;p--)
{
    mul(res,res,k);
    if((n>>p)&1)
    {
        for(int i=k-1;i>=0;i--) res[i+1]=res[i];
        res[0]=0;
        rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
    }
}
rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
if(ans<0) ans+=mod;
return ans;
}
VI BM(VI s)
{
    VI C(1,1),B(1,1);
    int L=0,m=1,b=1;
    rep(n,0,SZ(s))
    {
        ll d=0;
        rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
        if(d==0) ++m;
        else if(2*L<=n)
        {
            VI T=C;
            ll c=mod-d*pow_mod(b,mod-2)%mod;
            while(SZ(C)<SZ(B)+m) C.pb(0);
            rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
            L=n+1-L;B=T;b=d;m=1;
        }
        else
        {
            ll c=mod-d*pow_mod(b,mod-2)%mod;
            while(SZ(C)<SZ(B)+m) C.pb(0);
            rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
            ++m;
        }
    }
    return C;
}
int gao(VI a,ll n)
{
    VI c=BM(a);
    c.erase(c.begin());
    rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
    return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
}
}
int main()
{
    ll n;

```



```

while(scanf("%lld",&n)!=EOF)
{
    printf("%d\n",linear_seq::gao(VI{1,5,11,36,95,281,781,2245,6336},n-1));
}
return 0;
}

```

4.3 Chinese Remainder Theorem

```

#include<bits/stdc++.h>
#define MAXN 105
#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,k;
int r[MAXN][MAXN],x[MAXN];
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 solve(vector<P> &v)
{
    int n=v.size();
    for(int i=0;i<n;i++)
        for(int j=i+1;j<n;j++)
            r[i][j]=mod_inverse(v[i].S,v[j].S);
    int ans=0;
    for(int i=0;i<n;i++)
    {
        x[i]=v[i].F;
        for(int j=0;j<i;j++)
        {
            x[i]=r[j][i]*(x[i]-x[j]);
            x[i]=x[i]%v[i].S;
            if(x[i]<0) x[i]+=v[i].S;
        }
    }
}

```

```

    }
    int base=1;
    for(int i=0;i<n;i++)
    {
        ans+=base*x[i];
        base*=v[i].S;
    }
    return ans;
}
int main()
{
    vector<P> v;
    v.push_back(P(4,7));
    v.push_back(P(3,13));
    printf("%d\n",solve(v));
    return 0;
}

```

4.4 Matrix 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;
}

```

4.5 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;
}

```

4.6 Extended GCD

```

#include<bits/stdc++.h>
using namespace std;
typedef __int64 ll;
ll extgcd(ll a,ll b,ll &x,ll &y)
{
    ll d=a;
    if(b!=0)
    {
        d=extgcd(b,a%b,y,x);
        y-=(a/b)*x;
    }
    else
    {
        x=1;
        y=0;
    }
    return d;
}
ll a,b,x,y;
int main()
{
    while(scanf("%I64d%I64d",&a,&b)==2)
    {
        if(extgcd(a,b,x,y)==1)
        {
            while(x<0)
            {
                x+=b;
                y-=a;
            }
            printf("%I64d %I64d\n",x,y);
        }
        else puts("sorry");
    }
    return 0;
}

```

4.7 Fast Fourier Transform

```
#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;
    }
    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;
}

```

4.8 Fast Walsh-Hadamard Transform

```

#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];
                    //xor:
                    a[i+j]=(x+y)%MOD,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];
            //xor:
            a[i+j]=1LL*(x+y)*REV%MOD,a[i+j+d]=(1LL*(x-y)*REV%MOD+MOD)%MOD;
            //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;
}

```

4.9 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;
}

```

4.10 Linear Basis

```

#include<bits/stdc++.h>
#define MAXN 1000
using namespace std;
int a[MAXN];
int n;
int cal()
{
    int k=1;
    for(int j=63;j>=0;j--)
    {
        int t=0;
        for(int i=k;i<=n;i--)
        {
            if((a[i]>>j)&1)
            {
                t=i;
                break;
            }
        }
    }
}

```

```

        if(t)
        {
            swap(a[t],a[k]);
            for(int i=1;i<=n;i++)
                if(i!=k&&((a[i]>>j)&1)) a[i]^=a[k];
            k++;
        }
    }
    return k-1;
}

```

4.11 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);
}

```

4.12 Linear Recurrence

```

// Calculating kth term of linear recurrence sequence
// Complexity: init  $O(n^2 \log)$  query  $O(n^2 \log k)$ 
// Requirement: const LOG const MOD
// Input(constructor): vector<int> - first n terms
//                      vector<int> - transition function
// Output(calc(k)): int - the kth term mod MOD
// Example: In: {1, 1} {2, 1}  $a_n = 2a_{n-1} + a_{n-2}$ 
//          Out: calc(3) = 3, calc(10007) = 71480733 (MOD 1e9+7)

struct LinearRec {
    int n;
    vector<int> first, trans;
    vector<vector<int>> > bin;

    vector<int> add(vector<int> &a, vector<int> &b) {
        vector<int> result(n * 2 + 1, 0);
        //You can apply constant optimization here to get a ~10x speedup
        for (int i = 0; i <= n; ++i) {
            for (int j = 0; j <= n; ++j) {
                if ((result[i + j] += (long long)a[i] * b[j] % MOD) >= MOD) {
                    result[i + j] -= MOD;
                }
            }
        }
    }
}

```

```

        }
    }
    for (int i = 2 * n; i > n; --i) {
        for (int j = 0; j < n; ++j) {
            if ((result[i - 1 - j] += (long long)result[i] * trans[j] % MOD) >=
                MOD) {
                result[i - 1 - j] -= MOD;
            }
        }
        result[i] = 0;
    }
    result.erase(result.begin() + n + 1, result.end());
    return result;
}

LinearRec(vector<int> &first, vector<int> &trans):first(first), trans(trans) {
    n = first.size();
    vector<int> a(n + 1, 0);
    a[1] = 1;
    bin.push_back(a);
    for (int i = 1; i < LOG; ++i) {
        bin.push_back(add(bin[i - 1], bin[i - 1]));
    }
}

int calc(int k) {
    vector<int> a(n + 1, 0);
    a[0] = 1;
    for (int i = 0; i < LOG; ++i) {
        if (k >> i & 1) {
            a = add(a, bin[i]);
        }
    }
    int ret = 0;
    for (int i = 0; i < n; ++i) {
        if ((ret += (long long)a[i + 1] * first[i] % MOD) >= MOD) {
            ret -= MOD;
        }
    }
    return ret;
}
};

```

4.13 LU Decomposition

```

#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");
    }
}

```

```

    }
}

```

4.14 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");
}
}

```

4.15 Miller-Rabin primality test

```

#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;
}

```

4.16 Mod-Combinatation and Mod-fact

```

#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;
}

```


}

4.17 Fast Number-Theoretic Transform

```
#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[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 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;
}

```

4.18 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;
}

```

4.19 Pollard-Rho

```

#include<stdio.h>

```

```

#include<string.h>
#include<stdlib.h>
#include<time.h>
#include<iostream>
#include<algorithm>
using namespace std;

//*****
// Miller_Rabin
//                               <2^63
//*****
const int S=20;//                               S

//      (a*b)%c.  a,  blong                               long
// a,b,c <2^63
long long mult_mod(long long a,long long b,long long c)
{
    a%=c;
    b%=c;
    long long ret=0;
    while(b)
    {
        if(b&1){ret+=a;ret%=c;}
        a<<=1;
        if(a>=c)a%=c;
        b>>=1;
    }
    return ret;
}

//      x^n %c
long long pow_mod(long long x,long long n,long long mod)//x^n%c
{
    if(n==1)return x%mod;
    x%=mod;
    long long tmp=x;
    long long ret=1;
    while(n)
    {
        if(n&1) ret=mult_mod(ret,tmp,mod);
        tmp=mult_mod(tmp,tmp,mod);
        n>>=1;
    }
    return ret;
}

//      a      ,n-1=x*2^t      a^(n-1)=1(mod n)      n
//      true      ,      false
bool check(long long a,long long n,long long x,long long t)
{
    long long ret=pow_mod(a,x,n);

```

```

    long long last=ret;
    for(int i=1;i<=t;i++)
    {
        ret=mult_mod(ret,ret,n);
        if(ret==1&&last!=1&&last!=n-1) return true;//
        last=ret;
    }
    if(ret!=1) return true;
    return false;
}

// Miller_Rabin()
//      true      .(
//      false     ;

bool Miller_Rabin(long long n)
{
    if(n<2)return false;
    if(n==2)return true;
    if((n&1)==0) return false;//
    long long x=n-1;
    long long t=0;
    while((x&1)==0){x>>=1;t++;}
    for(int i=0;i<S;i++)
    {
        long long a=rand()%(n-1)+1;//rand() stdlib . h
        if(check(a,n,x,t))
            return false;//
    }
    return true;
}

//*****
//pollard_rho
//*****
long long factor[100];//
int tol;// 0

long long gcd(long long a,long long b)
{
    if(a==0)return 1;//??????
    if(a<0) return gcd(-a,b);
    while(b)
    {
        long long t=a%b;
        a=b;
        b=t;
    }
    return a;
}

long long Pollard_rho(long long x,long long c)
{
    long long i=1,k=2;
    long long x0=rand()%x;
    long long y=x0;
    while(1)
    {

```

```

        i++;
        x0=(mult_mod(x0,x0,x)+c)%x;
        long long d=gcd(y-x0,x);
        if(d!=1&&d!=x) return d;
        if(y==x0) return x;
        if(i==k){y=x0;k+=k;}
    }
}
//          n
void findfac(long long n)
{
    if(Miller_Rabin(n))//
    {
        factor[tol++]=n;
        return;
    }
    long long p=n;
    while(p>=n)p=Pollard_rho(p,rand()%(n-1)+1);
    findfac(p);
    findfac(n/p);
}

int main()
{
    //srand(time(NULL));// time . h // POJG ++
    long long n;
    while(scanf("%I64d",&n)!=EOF)
    {
        tol=0;
        findfac(n);
        for(int i=0;i<tol;i++)printf("%I64d ",factor[i]);
        printf("\n");
        if(Miller_Rabin(n))printf("Yes\n");
        else printf("No\n");
    }
    return 0;
}

```

4.20 Polynomial Operations

```

#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 10000000000
#define MOD 998244353
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
int n,k,a[MAXN];
int pow_mod(int a,int i,int m)
{
    int s=1;
    while(i)
    {
        if(i&1) s=1LL*s*a%m;
    }
}

```

```

        a=1LL*a*a%m;
        i>>=1;
    }
    return s;
}
inline int inc(int a,int b) {a+=b; return a>=MOD?a-MOD:a;}
inline int dec(int a,int b) {a-=b; return a<0?a+MOD:a;}
int Tonelli_Shanks(int n,ll p)
{
    if(p==2) return (n&1)?1:-1;
    if(pow_mod(n,p>>1,p)!=1) return -1;
    if(p&2) return pow_mod(n,(p+1)>>2,p);
    int s=__builtin_ctzll(p^1);
    int q=p>>s,z=2;
    for(;pow_mod(z,p>>1,p)==1;++z);
    int c=pow_mod(z,q,p),r=pow_mod(n,(q+1)>>1,p),t=pow_mod(n,q,p),tmp;
    for(int m=s,i;t!=1;)
    {
        for(i=0,tmp=t;tmp!=1;++i) tmp=tmp*tmp%p;
        for(;i<--m;) c=c*c%p;
        r=r*c%p;c=c*c%p;t=t*c%p;
    }
    return r;
}
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=acos(-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;
    }
    return res;
}
vector<int> square_mod(vector<int> &a,int m)

```

```

    {
        return multiply_mod(a,a,m,1);
    }
};
namespace poly
{
    int inv(int x) {return pow_mod(x,MOD-2,MOD);}
    vector<int> fa,fb,fc,fd;
    vector<int> get_inv(vector<int> &a,int n)
    {
        assert(a[0]!=0);
        if(n==1)
        {
            fa.resize(1);
            fa[0]=inv(a[0]);
            return fa;
        }
        fa=get_inv(a,(n+1)>>1);
        fb=fft::multiply_mod(fa,fa,MOD,1);
        fb=fft::multiply_mod(fb,a,MOD);
        fa.resize(n);
        for(int i=0;i<n;i++)
        {
            fa[i]=inc(fa[i],fa[i]);
            fa[i]=dec(fa[i],fb[i]);
        }
        return fa;
    }
    vector<int> get_sqrt(vector<int> &a,int n)
    {
        if(n==1)
        {
            fc.resize(1);
            int x=Tonelli_Shanks(a[0],MOD);
            assert(x!=-1);
            fc[0]=x;return fc;
        }
        fd=get_sqrt(a,(n+1)>>1);
        fc=get_inv(fd,(n+1)>>1);
        fd=fft::multiply_mod(fd,fd,MOD,1);
        for(int i=0;i<(n+1)/2;i++) fc[i]=1LL*fc[i]*((MOD+1)/2)%MOD;
        for(int i=0;i<n;i++) fd[i]=inc(fd[i],a[i]);
        fd=fft::multiply_mod(fd,fc,MOD);
        fd.resize(n);return fd;
    }
    vector<int> diff(vector<int> &a)
    {
        for(int i=1;i<(int)a.size();i++) a[i-1]=1LL*a[i]*i%MOD;
        if(a.size()>=1) a.resize((int)a.size()-1);
        return a;
    }
    vector<int> intg(vector<int> &a)
    {
        int sz=(int)a.size();
        a.resize(sz+1);
        static vector<int> Inv(sz+1);
        Inv[1]=1;
        for(int i=2;i<=sz;i++) Inv[i]=dec(MOD,1LL*Inv[MOD%i]*(MOD/i)%MOD);
        for(int i=sz;i>=1;i--) a[i]=1LL*a[i-1]*Inv[i]%MOD;
    }
}

```

```

        a[0]=0;
        return a;
    }
};
int main()
{
    vector<int> res(20);
    res[0]=1;res[1]=2;res[2]=1;
    res=poly::get_sqrt(res,6);
    for(int i=0;i<(int)res.size();i++) printf("%d ",res[i]);
    return 0;
}

```

4.21 Polynomial Summations

```

#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;
ll pow_mod(ll a,ll i)
{
    ll s=1;
    while(i)
    {
        if(i&1) s=s*a%MOD;
        a=a*a%MOD;
        i>>=1;
    }
    return s;
}
ll gcd(ll a,ll b)
{
    if(b==0) return a;
    return gcd(b,a%b);
}
namespace polysum
{
    const int D=100005;
    ll a[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],c[D];
    ll calcn(int d,ll *a,ll n)
    {
        if(n<=d) return a[n];
        p1[0]=p2[0]=1;
        for(int i=0;i<=d;i++)
        {
            ll t=(n-i+MOD)%MOD;
            p1[i+1]=p1[i]*t%MOD;
        }
        for(int i=0;i<=d;i++)
        {
            ll t=(n-d+i+MOD)%MOD;
            p2[i+1]=p2[i]*t%MOD;
        }
    }
}

```

```

    }
    ll ans=0;
    for(int i=0;i<=d;i++)
    {
        ll t=g[i]*g[d-i]%MOD*p1[i]%MOD*p2[d-i]%MOD*a[i]%MOD;
        if((d-i)&1) ans=(ans-t+MOD)%MOD;
        else ans=(ans+t)%MOD;
    }
    return ans;
}
void init(int M)
{
    f[0]=f[1]=g[0]=g[1]=1;
    for(int i=2;i<=M+4;i++) f[i]=f[i-1]*i%MOD;
    g[M+4]=pow_mod(f[M+4],MOD-2);
    for(int i=M+3;i>=1;i--) g[i]=g[i+1]*(i+1)%MOD;
}
ll polysum(ll n,ll *a,ll m) //a[0]..a[m] \sum_{i=0}^{n-1} a[i]
{
    a[m+1]=calcn(m,a,m+1);
    for(int i=1;i<=m+1;i++) a[i]=(a[i-1]+a[i])%MOD;
    return calcn(m+1,a,n-1);
}
ll qpolysum(ll R,ll n,ll *a,ll m) //a[0]..a[m] \sum_{i=0}^{n-1} a[i]*R^i
{
    if(R==1) return polysum(n,a,m);
    a[m+1]=calcn(m,a,m+1);
    ll r=pow_mod(R,MOD-2),p3=0,p4=0,c,ans;
    h[0][0]=0;h[0][1]=1;
    for(int i=1;i<=m+1;i++)
    {
        h[i][0]=(h[i-1][0]+a[i-1])*r%MOD;
        h[i][1]=h[i-1][1]*r%MOD;
    }
    for(int i=0;i<=m+1;i++)
    {
        ll t=g[i]*g[m+1-i]%MOD;
        if(i&1) p3=((p3-h[i][0]*t)%MOD+MOD)%MOD,p4=((p4-h[i][1]*t)%MOD+MOD)%MOD;
        else p3=(p3+h[i][0]*t)%MOD,p4=(p4+h[i][1]*t)%MOD;
    }
    c=pow_mod(p4,MOD-2)*(MOD-p3)%MOD;
    for(int i=0;i<=m+1;i++) h[i][0]=(h[i][0]+h[i][1]*c)%MOD;
    for(int i=0;i<=m+1;i++) C[i]=h[i][0];
    ans=(calcn(m,C,n)*pow_mod(R,n)-c)%MOD;
    if(ans<0) ans+=MOD;
    return ans;
}
}
ll a[MAXN];
int main()
{
    a[0]=1;a[1]=100;a[2]=0;
    polysum::init(1000);
    printf("%lld\n",polysum::qpolysum(2,4,a,1));
    return 0;
}

```

4.22 Prime Counting Function

```
#include<bits/stdc++.h>
#define MAXN 1000005// MAXN=sqrt(upper_bound)
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
ll f[MAXN],g[MAXN],n,k; //f[i]:pi(n/i),g[i]:pi(i)

ll PrimeCount(ll n)
{
    ll i,j,m=0;
    for(m=1;m*m<=n;m++) f[m]=n/m-1;
    for(i=2;i<=m;i++) g[i]=i-1;
    for(i=2;i<=m;i++)
    {
        if(g[i]==g[i-1]) continue;
        for(j=1;j<=min(m-1,n/i/i);++j)
        {
            if(i*j<m) f[j]-=f[i*j]-g[i-1];
            else f[j]-=g[n/i/j]-g[i-1];
        }
        for(j=m;j>=i*i;j--) g[j]-=g[j/i]-g[i-1];
    }
    return f[1];
}

int main()
{
    while(scanf("%lld",&n)==1)
    {
        printf("%lld\n",PrimeCount(n));
    }
    return 0;
}
```

4.23 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;
}

```

4.24 Segmented Sieve

```

#include<bits/stdc++.h>
#define MAXL 1000005
#define MAXSQRTB 47000
#define INF 1000000000
using namespace std;
typedef long long ll;

```

```

bool is_prime_small[MAXSQRTB];
bool is_prime[MAXL];
vector<ll> prime;
void segment_sieve(ll a, ll b)
{
    for(ll i=0; (ll)i*i<=b; i++) is_prime_small[i]=true;
    for(ll i=0; i<b-a; i++) is_prime[i]=true;
    for(ll i=2; (ll)i*i<=b; i++)
    {
        if(is_prime_small[i])
        {
            for(ll j=2*i; (ll)j*j<=b; j+=i) is_prime_small[j]=false;
            for(ll j=max(2LL, (a+i-1)/i)*i; j<b; j+=i) is_prime[j-a]=false;
        }
    }
    for(ll i=0; i<b-a; i++)
        if(is_prime[i]&& a+i!=1) prime.push_back(a+i);
}

```

4.25 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;
int dbit(int 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]=1; 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;
}

```

4.26 Stirling number of the second kind(multiple)

```

#include<bits/stdc++.h>
#define MAXN 100005
#define MOD 998244353
#define INF 10000000000
#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;
}

```

4.27 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);
}

```

4.28 Prefix Sum of Miu

```

#include<bits/stdc++.h>
#define MAXN 500005
#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;
}

```

4.29 Prefix Sum of Phi

```

#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;
}

```

4.30 Tonelli-Shanks

```

#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,k,a[MAXN];
ll pow_mod(ll a,ll i,ll m)
{
    ll s=1;
    while(i)
    {
        if(i&1) s=s*a%m;
        a=a*a%m;
        i>>=1;
    }
    return s;
}
ll Tonelli_Shanks(ll n,ll p)
{
    if(p==2) return (n&1)?1:-1;
    if(pow_mod(n,p>>1,p)!=1) return -1;
    if(p&2) return pow_mod(n,(p+1)>>2,p);
    int s=__builtin_ctzll(p^1);
    ll q=p>>s,z=2;
    for(;pow_mod(z,p>>1,p)==1;++z);
    ll c=pow_mod(z,q,p),r=pow_mod(n,(q+1)>>1,p),t=pow_mod(n,q,p),tmp;
    for(int m=s,i;t!=1;)
    {
        for(i=0,tmp=t;tmp!=1;++i) tmp=tmp*tmp%p;
        for(;i<--m;) c=c*c%p;
        r=r*c%p;c=c*c%p;t=t*c%p;
    }
    return r;
}
int main()
{
    ll n,p;
    while(scanf("%lld%lld",&n,&p)==2) printf("%lld\n",Tonelli_Shanks(n,p));
    return 0;
}

```

}

4.31 Euclid

```
#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;
ll inv2,inv6,a,b,c,l,r;//need initialize
struct E
{
    ll f,g,h;
    E(){ }
    E(ll _f,ll _g,ll _h){f=_f,g=_g,h=_h;}
};
ll pow_mod(ll a,ll i)
{
    ll s=1;
    while(i)
    {
        if(i&1) s=s*a%MOD;
        a=a*a%MOD;
        i>>=1;
    }
    return s;
}
// f:\sum_{i=0}^n\lfloor\frac{ai+b}{c}\rfloor
// g:\sum_{i=0}^ni\times\lfloor\frac{ai+b}{c}\rfloor
// g:\sum_{i=0}^n\lfloor\frac{ai+b}{c}\rfloor^2
E cal(ll a,ll b,ll c,ll n)
{
    if(!a) return E(0,0,0);
    E x,y;
    if(a>=c||b>=c)
    {
        x=cal(a%c,b%c,c,n);
        y.f=(a/c*n%MOD*(n+1)%MOD*inv2+b/c*(n+1)+x.f)%MOD;
        y.g=(a/c*n%MOD*(n+1)%MOD*(n*2+1)%MOD*inv6+b/c*(n+1)%MOD*n%MOD*inv2+x.g)%MOD;
        y.h=a/c*(a/c)%MOD*n%MOD*(n+1)%MOD*(n*2+1)%MOD*inv6%MOD;
        (y.h+=b/c*(b/c)%MOD*(n+1))%=MOD;
        (y.h+=a/c*(b/c)%MOD*n%MOD*(n+1))%=MOD;
        (y.h+=2LL*(a/c)%MOD*x.g)%=MOD;
        (y.h+=2LL*(b/c)%MOD*x.f)%=MOD;
        y.f=(y.f+MOD)%MOD;y.g=(y.g+MOD)%MOD;y.h=(y.h+MOD)%MOD;
        return y;
    }
    ll m=(a*n+b)/c;
    x=cal(c,c-b-1,a,m-1);
    y.f=(n*m-x.f)%MOD;
    y.g=y.g*inv2%MOD;
    y.h=(n*m%MOD*(m+1)-2LL*x.g-2LL*x.f-y.f)%MOD;
    y.f=(y.f+MOD)%MOD;y.g=(y.g+MOD)%MOD;y.h=(y.h+MOD)%MOD;
```

```

    return y;
}
int main()
{
    inv2=pow_mod(2,MOD-2);inv6=pow_mod(6,MOD-2);
    return 0;
}

```

4.32 Simpson

```

#include<bits/stdc++.h>
using namespace std;
double simpson(double a,double b)
{
    double c=a+(b-a)/2;
    return (F(a)+4*F(c)+F(b))*(b-a)/6;
}
double asr(double a,double b,double eps,double A)
{
    double c=a+(b-a)/2;
    double L=simpson(a,c),R=simpson(c,b);
    if(fabs(L+R-A)<=15*eps) return L+R+(L+R-A)/15.0;
    return asr(a,c,eps/2,L)+asr(c,b,eps/2,R);
}
double asr(double a,double b,double eps)
{
    return asr(a,b,eps,simpson(a,b));
}

```

4.33 Farey

```

#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<ll,ll> P;
ll a,b,c,d,t;
P cal(ll a,ll b,ll c,ll d)
{
    //printf("%lld %lld %lld %lld\n",a,b,c,d);
    ll x=a/b+1;if(x*d<c) return P(x,1);
    if(!a) return P(1,d/c+1);
    if(a<=b&&c<=d)
    {
        P t=cal(d,c,b,a);
        swap(t.F,t.S); return t;
    }
    x=a/b;P t=cal(a-b*x,b,c-d*x,d);
    t.F+=t.S*x;return t;
}
int main()

```



```

{
    while(~scanf("%lld%lld%lld%lld",&a,&b,&c,&d))
    {
        t=__gcd(a,b),a/=t,b/=t,t=__gcd(c,d),c/=t,d/=t;
        printf("%lld %lld %lld %lld\n",a,b,c,d);
        P p=cal(a,b,c,d);
        printf("%lld/%lld\n",p.F,p.S);
    }
    return 0;
}

```

5 Others

5.1 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;
}

```

5.2 Dynamic Convex Hull Trick

```

#pragma GCC optimize(3)
#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,k,a[MAXN];
bool qu=0;
struct line
{
    long long m,b;

```

```

mutable function<const line*> succ;
bool operator<(const line& rhs) const
{
    if (!qu) return m<rhs.m;
    const line* s=succ();
    if (!s)
        return 0;
    return b-s->b<rhs.m*(s->m-m);
}
};
struct hull:public multiset<line>
{
    bool bad(iterator y)
    {
        auto z=next(y);
        if (y==begin())
        {
            if (z==end())
                return 0;
            return (y->m==z->m && y->b<=z->b);
        }
        auto x=prev(y);
        if (z==end())
            return (y->m==x->m && y->b<=x->b);
        return 1.0*(x->b-y->b)*(z->m-y->m)>=1.0*(y->b-z->b)*(y->m-x->m);
    }
    void add(long long m,long long b)
    {
        auto it=insert({m,b});
        it->succ=[=] { return (next(it)==end())? 0:&*next(it); };
        if (bad(it))
        {
            erase(it);
            return;
        }
        while (next(it)!=end() && bad(next(it))) erase(next(it));
        while (it!=begin() && bad(prev(it))) erase(prev(it));
    }
    long long eval(long long x)
    {
        if (empty()) return -(1LL<<60);
        qu=1;line l=*lower_bound((line){x,0});qu=0;
        return l.m*x+l.b;
    }
};
int main()
{
    return 0;
}

```

5.3 Knuth's optimization

```

#include<bits/stdc++.h>
#define MAXN 2005
#define INF 1000000000
using namespace std;
typedef long long ll;

```

```

11 a[MAXN];
11 n,k;
11 dp[MAXN][MAXN],knuth[MAXN][MAXN];
int main()
{
    while(scanf("%lld %lld",&n,&k)==2)
    {
        a[0]=0;
        for(11 i=1;i<=k;i++)
            scanf("%lld",&a[i]);
        a[k+1]=n;
        for(11 i=0;i<=k+1;i++)
            for(11 j=0;j<=k+1;j++)
                dp[i][j]=INF;
        for(11 i=0;i<=k;i++)
            dp[i][i+1]=0;
        for(11 l=3;l<=k+2;l++)
            for(11 i=0;i<=k+2-l;i++)
            {
                if(l==3)
                {
                    dp[i][i+1-1]=a[i+1-1]-a[i];
                    knuth[i][i+1-1]=i+1;
                }
                else
                    for(11 j=knuth[i][i+1-2];j<=knuth[i+1][i+1-1];j++)
                        if(dp[i][j]+dp[j][i+1-1]+a[i+1-1]-a[i]<dp[i][i+1-1])
                        {
                            dp[i][i+1-1]=dp[i][j]+dp[j][i+1-1]+a[i+1-1]-a[i];
                            knuth[i][i+1-1]=j;
                        }
            }
        printf("%lld\n",dp[0][k+1]);
    }
    return 0;
}

```

5.4 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]);
}

```

5.5 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;
}

```

5.6 Enumeration of 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,k,a[MAXN];
void solve1(int sup)//all subsets
{
    int sub=sup;

```

```

do
{
    //operation here
    sub=(sub-1)&sup;
}while(sub!=sup);
}
void solve2(int n,int k) //all subsets of (1<<n) of size k
{
    int comb=(1<<k)-1;
    while(comb<1<<n)
    {
        //operation here
        int x=comb&-comb,y=comb+x;
        comb=((comb&~y)/x>>1)|y;
    }
}
int main()
{
    return 0;
}

```

5.7 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;
}

```

6 String

6.1 Trie

```

#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 tot=1,n;
int trie[MAXN][26];
bool ed[MAXN];
void insert(char *s,int rt)
{
    for(int i=0;s[i];i++)
    {
        int x=s[i]-'a';
        if(trie[rt][x]==0) trie[rt][x]=++tot;
        rt=trie[rt][x];
    }
    ed[rt]=true;
}
bool find(char *s,int rt)
{
    for(int i=0;s[i];i++)
    {
        int x=s[i]-'a';
        if(trie[rt][x]==0) return false;
        rt=trie[rt][x];
    }
    return ed[rt];
}
int main()
{
    memset(ed,false,sizeof(ed));
    return 0;
}

```

6.2 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;
}

```

6.3 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;
}

```

6.4 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;
}

```

6.5 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;
}

```

6.6 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;
}

```

6.7 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]);
}

```

6.8 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;
}

```
