# Code Template for ACM-ICPC

Roundgod @ Nanjing University

April 15, 2021

# Contents


**1 DataStructures** **1**
- 1.1 Binary Indexed Tree . . . 1
- 1.2 Centroid Decomposition . . . 1
- 1.3 Descartes Tree . . . 3
- 1.4 Disjoint Set Union . . . 4
- 1.5 Heap . . . 4
- 1.6 Heavy Light Decomposition . . . 5
- 1.7 Lichao Segment Tree . . . 8
- 1.8 Link Cut Tree . . . 9
- 1.9 Long Chain Decomposition . . . 10
- 1.10 Mo's Algorithm . . . 12
- 1.11 Monotone Deque . . . 13
- 1.12 Monotone Stack . . . 14
- 1.13 Persistent Disjoint Set Union . . . 14
- 1.14 Persistent Segment Tree . . . 15
- 1.15 Persistent Trie . . . 17
- 1.16 Mo's Algorithm with Queries . . . 17
- 1.17 Mo's Algorithm with Queries on Tree . . . 19
- 1.18 Segment Tree . . . 22
- 1.19 Segment Tree Beats . . . 24
- 1.20 Segment Tree Merge . . . 26
- 1.21 Sparse Table . . . 27
- 1.22 Splay . . . 28
- 1.23 Treap . . . 31
- 1.24 DSU on Tree . . . 33
- 1.25 Mo's Algorithm on Tree . . . 34
- 1.26 Virtual Tree . . . 36
- 1.27 Young Tableaux . . . 38

**2 Geometry** **39**
- 2.1 Geometry All-in-One . . . 39
- 2.2 Convex Hull . . . 52
- 2.3 Stereometry . . . 53

**3 Graph** **60**
- 3.1 Bipartite Matching . . . 60
- 3.2 Block Cut Tree . . . 61
- 3.3 Bridge Tree . . . 62
- 3.4 Chordal Graph . . . 63
- 3.5 Common Matching(Blossom) . . . 64
- 3.6 Dijkstra . . . 67
- 3.7 Dinic . . . 67
- 3.8 Dominator Tree . . . 68
- 3.9 Dynamic Bridge . . . 70
- 3.10 Dynamic Connectivity . . . 78
- 3.11 Ear Decomposition . . . 81
- 3.12 Floyd-Warshall . . . 82
- 3.13 Ford-Fulkerson . . . 83
- 3.14 Gomory-Hu Tree . . . 84
- 3.15 Hopcrokt-Karp . . . 86
- 3.16 Kosaraju . . . 87
- 3.17 Kuhn-Munkres . . . 88

3.18 LCA with binary lifting . . . 89
3.19 LCA with range minimum query . . . 90
3.20 Min-cost flow(with Dijkstra) . . . 91
3.21 Min-cost flow(with SPFA) . . . 92
3.22 Minimum Arborescence . . . 93
3.23 Minimum Diameter Spanning Tree . . . 95
3.24 SPFA . . . 97
3.25 Square Counting . . . 98
3.26 Tarjan . . . 99
3.27 Tree Chain Intersection . . . 100
3.28 Triangle Counting . . . 101
3.29 Topological Sort . . . 102

**4 Math** **103**
4.1 Berlekamp-Massey . . . 103
4.2 BigNum . . . 105
4.3 Chinese Remainder Theorem . . . 112
4.4 Matrix Determinant . . . 113
4.5 DIVCNT1 . . . 114
4.6 Euclid . . . 115
4.7 Euler Sieve . . . 116
4.8 Extended GCD . . . 117
4.9 Extended Lucas Theorem . . . 118
4.10 Farey . . . 120
4.11 Fast Multiplication . . . 121
4.12 Fast Fourier Transform . . . 121
4.13 Fast Walsh-Hadamard Transform . . . 124
4.14 Gauss-Jordan . . . 125
4.15 Polynomial Interpolation . . . 126
4.16 Linear Basis . . . 127
4.17 Linear Congruence . . . 128
4.18 LU Decomposition . . . 128
4.19 Matrix Opertions . . . 129
4.20 Miller-Rabin primality test . . . 131
4.21 Mod-Combination and Mod-fact . . . 131
4.22 Fast Number-Theoretic Transform . . . 132
4.23 Pell's equation . . . 134
4.24 Pohlig-Hellman . . . 135
4.25 Points In a Circle . . . 136
4.26 Pollard Rho . . . 137
4.27 Polynomial Operations . . . 139
4.28 Polynomial Summations . . . 144
4.29 Power Tower . . . 146
4.30 Prime Counting Function . . . 147
4.31 Primitive Root . . . 148
4.32 Schreier-Sims . . . 149
4.33 Segmented Sieve . . . 151
4.34 Simpson Method . . . 151
4.35 Stirling number of the first kind . . . 152
4.36 Stirling number of the second kind(multiple) . . . 154
4.37 Subset Convolution . . . 155
4.38 Prefix Sum of Miu . . . 157
4.39 Prefix Sum of Phi . . . 157
4.40 Tonelli-Shanks . . . 159

**5 Others** **160**
5.1 Convex Hull Trick . . . 160
5.2 Dynamic Convex Hull Trick . . . 160
5.3 Dynamic Dynamic Programming . . . 162
5.4 Knuth's Optimization . . . 164
5.5 Matroid Intersection . . . 165
5.6 Multiple Backpack . . . 168
5.7 Nim Multiplication . . . 169
5.8 Simplex Method . . . 170
5.9 Sum Over Subset Dynamic Programming . . . 171
5.10 Subset Choosing(Gosper's Hack) . . . 172
5.11 Surreal Number . . . 172
5.12 Weighted Matroid Intersection . . . 173
5.13 Zeller's Formula . . . 177

**6 String** **177**
6.1 Aho-Corasick Automaton . . . 177
6.2 KMP . . . 180
6.3 Manacher . . . 180
6.4 Suffix Array . . . 181
6.5 SA-IS . . . 182
6.6 Suffix Automaton . . . 184
6.7 Trie . . . 185

**7 Miscellaneous** **186**
7.1 Stairways(Sqrt decomposition+Convex hull) . . . 186
7.2 Dynamic Tree Diameter(Centroid Decomposition+Segment Tree) . . . 189
7.3 Dynamic Tree Diameter(Bracket Sequence+Segment Tree) . . . 192
7.4 Logical Chain(Kosaraju+std::bitset) . . . 195
7.5 Key Array(Splay+reverse operation) . . . 197
7.6 Bimatching(Blossom) . . . 199
7.7 Solar Panels(number theory sqrt for more than one numbers) . . . 202
7.8 XOR Tree(Long Chain Decomposition) . . . 202
7.9 Coins 2(Knapsack+Period) . . . 204
7.10 Subsequence Queries(DP+Matrix) . . . 205
7.11 Distinct Integers(Novel use of segment tree) . . . 207
7.12 Forbidden Words(64 times faster) . . . 209
7.13 Reactor Cooling(Maximum flow with lowerbounds) . . . 210
7.14 Great Siberian Wall(Minimum Enclosing Triangle) . . . 212
7.15 Road Times(Simplex) . . . 217
7.16 Subset Sum(Set balancing) . . . 220
7.17 Longest Shortest Path(Dual Program) . . . 221

# 1 DataStructures

## 1.1 Binary Indexed 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;
}
```

## 1.2 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;
```

```cpp
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;
}
```

## 1.3 Descartes 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,h[MAXN];
int st[MAXN],t;
int fa[MAXN],ls[MAXN],rs[MAXN],root;
int main()
{
    for(int i=1;i<=n;i++)
    {
        while(t&&h[st[t-1]]>h[i]) ls[i]=st[--t];
        if(t) rs[st[t-1]]=i;
        st[t++]=i;
    }
    for(int i=1;i<=n;i++) fa[ls[i]]=fa[rs[i]]=i;
    for(int i=1;i<=n;i++) if(!fa[i]) root=i;
```

```
}
```

## 1.4 Disjoint Set Union

```
#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;
}
```

## 1.5 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;
}
```

## 1.6 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--)
    {
        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;
}
```

## 1.7 Lichao Segment Tree

```
#include<bits/stdc++.h>
#define MAXN 80005
#define MAXM 10000005
#define MAXT 1000000001
#define INF 1000000000000000000LL
#define MOD 1000000007
#define F first
#define S second
int n,q,tot,lson[MAXM],rson[MAXM];
bool has[MAXM];
P ans[MAXM];
ll f(P p,int x)
{
    return 1LL*p.F*x+p.S;
}
void insert(int &k,int l,int r,int x,int y,P p)
{
    if(l>y||x>r) return;
    k=++tot;
    has[k]=false;
    if(l>=x&&r<=y)
    {
        if(!has[k])
        {
            has[k]=true;
            ans[k]=p;
            return;
        }
        ll trl=f(ans[k],l),trr=f(ans[k],r);
        ll vl=f(p,l),vr=f(p,r);
        if(trl<=vl&&trr<=vr) return;
        if(trl>=vl&&trr>=vr) {ans[k]=p; return;}
        int mid=(l+r)/2;
        if(trl>=vl) swap(ans[k],p);
        if(f(ans[k],mid)<=f(p,mid)) insert(rson[k],mid+1,r,x,y,p);
        else swap(ans[k],p),insert(lson[k],l,mid,x,y,p);
        return;
    }
    int mid=(l+r)/2;
    insert(lson[k],l,mid,x,y,p); insert(rson[k],mid+1,r,x,y,p);
}
ll query(int &k,int l,int r,int x)
{
    if(!k) return INF;
    ll res=(!has[k]?INF:f(ans[k],x));
    if(l==r) return res;
    int mid=(l+r)/2;
    if(x<=mid) return min(res,query(lson[k],l,mid,x));
    else return min(res,query(rson[k],mid+1,r,x));
}
```

## 1.8 Link Cut Tree

```
#include <bits/stdc++.h>
#define MAXN 300005
#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[2*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);
        if(findr(y)!=x) t[x].fa=y;
    }
    inline void cut(int x, int y)
    {
        maker(x);
        if(findr(y)==x&&t[x].fa==y&&t[y].ch[0]==x&&!t[y].ch[1])
        {
            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) lct::link(x, y);
        if(op==2) lct::cut(x, y);
        if(op==3) lct::t[x].w=y,lct::splay(x);
    }
}
```

## 1.9 Long Chain Decomposition

```
//Yinchuan 2019 Regional E
#pragma GCC optimize(3)
#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 unsigned long long ull;
typedef pair<int,int> P;
int n,k,a[MAXN],len[MAXN],son[MAXN];
int st[MAXN],ed[MAXN];
int cnt[MAXN][4],save[MAXN][4];
ull ans[MAXN];
```

```
int curi,curj;
ull res;
vector<int> G[MAXN];
void dfs(int v,int p)
{
    for(auto to:G[v])
    {
        if(to==p) continue;
        dfs(to,v);
        if(len[to]>len[son[v]]) son[v]=to;
    }
    len[v]=len[son[v]]+1;
}
void dfs2(int v,int p)
{
    if(son[v])
    {
        st[son[v]]=st[v]+1;
        dfs2(son[v],v);
        ed[v]=ed[son[v]];
        for(int i=0;i<4;i++)
        {
            cnt[v][i]=cnt[son[v]][i];
            save[st[v]][i]=0;
        }
    }
    else
    {
        ed[v]=st[v];
        for(int i=0;i<4;i++)
        {
            cnt[v][i]=0;
            save[st[v]][i]=0;
        }
    }
    int x=(((a[v]>>curi)&1)<<1)+((a[v]>>curj)&1);
    save[st[v]][x]=1;
    cnt[v][x]++;
    while(ed[v]-st[v]>k)
    {
        for(int i=0;i<4;i++) cnt[v][i]-=save[ed[v]][i];
        ed[v]--;
    }
    for(auto to:G[v])
    {
        if(to==p||to==son[v]) continue;
        st[to]=ed[v]+1;
        dfs2(to,v);
        for(int i=st[to];i<=ed[to]&&i-st[to]<=k;i++)
            for(int j=0;j<4;j++)
            {
                save[i-st[to]+1+st[v]][j]+=save[i][j];
                cnt[v][j]+=save[i][j];
            }
    }
    ans[v]+=res*cnt[v][0]*cnt[v][3];
    ans[v]+=res*cnt[v][1]*cnt[v][2];
}
int main()
```

```
{
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    for(int i=2;i<=n;i++)
    {
        int f;
        scanf("%d",&f);
        G[f].push_back(i);
        G[i].push_back(f);
    }
    dfs(1,0);
    for(curi=0;curi<30;curi++)
        for(curj=0;curj<=curi;curj++)
        {
            if(curi==curj) res=1ULL<<(curi+curj); else res=1ULL<<(curi+curj+1);
            st[1]=1;
            dfs2(1,0);
        }
    for(int i=1;i<=n;i++) printf("%llu\n",ans[i]);
    return 0;
}
```

## 1.10 Mo's Algorithm

```
#include<bits/stdc++.h>
#define MAXN 100005
#define MAXM 100005
using namespace std;
struct query
{
    int l,r,id;
}save[MAXM];
int cnt[MAXN],a[MAXN],out[MAXN];
int n,m,ans,block;
bool cmp(query x,query y)
{
    if(x.l/block!=y.l/block) return x.l/block<y.l/block;
    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;
}
```

## 1.11 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;
}
```

## 1.12 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);
}
```

## 1.13 Persistent Disjoint Set Union

```cpp
#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[MAXN],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;
}
```

## 1.14 Persistent Segment Tree

```cpp
#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%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;
}
```

## 1.15 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;
}
```

## 1.16 Mo's Algorithm with Queries

```
#include<bits/stdc++.h>
```

```
#define MAXN 50005
#define MAXM 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;
const int blocks=1200;
int tot,tcnt,qid;
struct query
{
    int l,r,ti,id;
}Q[MAXN];
int n,q,cnt[MAXM],ans,a[MAXN];
P change[MAXN];
int res[MAXN];
bool cmp(query x,query y)
{
    if(x.l/blocks!=y.l/blocks) return x.l/blocks<y.l/blocks;
    if(x.r/blocks!=y.r/blocks) return x.r/blocks<y.r/blocks;
    if(x.r/blocks&1) return x.ti>y.ti; else return x.ti<y.ti;
}
void add(int pos)
{
    if(!cnt[a[pos]]) ans++;
    cnt[a[pos]]++;
}
void del(int pos)
{
    cnt[a[pos]]--;
    if(!cnt[a[pos]]) ans--;
}
void modify(int ti,int num)
{
    if(change[ti].F>=Q[num].l&&change[ti].F<=Q[num].r)
    {
        cnt[a[change[ti].F]]--;
        if(!cnt[a[change[ti].F]]) ans--;
        if(!cnt[change[ti].S]) ans++;
        cnt[change[ti].S]++;
    }
    swap(a[change[ti].F],change[ti].S);
}
char ch[2];
int main()
{
    scanf("%d%d",&n,&q);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    for(int i=1;i<=q;i++)
    {
        int l,r;
        scanf("%s%d%d",ch,&l,&r);
        if(ch[0]=='Q') Q[++tot]=(query){l,r,tcnt,++qid};
        else change[++tcnt]=P(l,r);
    }
    sort(Q+1,Q+tot+1,cmp);
    int l=1,r=0,ti=0;
```

```
    for(int i=1;i<=tot;i++)
    {
        while(l>Q[i].l) add(--l);
        while(l<Q[i].l) del(l++);
        while(r<Q[i].r) add(++r);
        while(r>Q[i].r) del(r--);
        while(ti<Q[i].ti) modify(++ti,i);
        while(ti>Q[i].ti) modify(ti--,i);
        res[Q[i].id]=ans;
    }
    for(int i=1;i<=qid;i++) printf("%d\n",res[i]);
    return 0;
}
```

## 1.17 Mo's Algorithm with Queries on Tree

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 100005
#define MAXM 100005
#define MAXLOGN 20
#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,q,tot,ctot,qtot,st[2*MAXN],ed[2*MAXN],loc[2*MAXN],val[MAXN];
int V[MAXN],W[MAXM],C[MAXN];
vector<int> dis;
vector<int> G[MAXN];
int spt[MAXLOGN+1][4*MAXN];
int vs[MAXN*2],depth[MAXN*2];
int id[MAXN],pos[2*MAXN],cnt[MAXN];
P change[MAXN];
ll res;
bool vis[MAXN];
vector<int> v;
void dfs(int v,int p,int d,int &k)
{
    st[v]=++tot; loc[tot]=v;
    id[v]=k;
    vs[k]=v;
    depth[k++]=d;
    for(auto to:G[v])
    {
        if(to==p) continue;
        dfs(to,v,d+1,k);
        vs[k]=v;
        depth[k++]=d;
    }
    ed[v]=++tot;
    loc[tot]=v;
}
void add_edge(int u,int v)
{
```

```
    G[u].push_back(v);
    G[v].push_back(u);
}
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) spt[0][i]=i;
    for(int i=1;1<<i<n;++i)
        for(int j=1;j+(1<<i)-1<=n;++j)
            spt[i][j]=getMin(spt[i-1][j],spt[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(spt[k][l],spt[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]))];
}

struct qry
{
    int l,r,z,ti,id;
}Q[MAXM];
bool cmp(qry a,qry b)
{
    if(pos[a.l]!=pos[b.l]) return pos[a.l]<pos[b.l];
    if(pos[a.r]!=pos[b.r]) return pos[a.r]<pos[b.r];
    if(pos[a.r]&1) return a.ti>b.ti; else return a.ti<b.ti;
}
void deal(int x)
{
    if(!vis[x])
    {
        cnt[C[x]]++;
        res+=1LL*W[cnt[C[x]]]*V[C[x]];
    }
    else
    {
        res-=1LL*W[cnt[C[x]]]*V[C[x]];
        cnt[C[x]]--;
    }
    vis[x]^=1;
}
void modify(int ti)
{
    int x=change[ti].F,y=change[ti].S;
```

```
    if(vis[x])
    {
        res-=1LL*W[cnt[C[x]]]*V[C[x]];
        cnt[C[x]]--;
        cnt[y]++;
        res+=1LL*W[cnt[y]]*V[y];
    }
    swap(C[change[ti].F],change[ti].S);
}
ll ans[MAXM];
const int blocks=2000;
int main()
{
    scanf("%d%d%d",&n,&m,&q);
    for(int i=1;i<=m;i++) scanf("%d",&V[i]);
    for(int i=1;i<=n;i++) scanf("%d",&W[i]);
    for(int i=0;i<n-1;i++)
    {
        int u,v;
        scanf("%d%d",&u,&v);
        add_edge(u,v);
    }
    for(int i=1;i<=n;i++) scanf("%d",&C[i]);
    init(n);
    for(int i=1;i<=tot;i++) pos[i]=i/blocks+1;
    for(int i=1;i<=q;i++)
    {
        int type,u,v;
        scanf("%d%d%d",&type,&u,&v);
        if(type==0) change[++ctot]=P(u,v);
        else
        {
            ++qtot;
            Q[qtot].id=qtot;
            Q[qtot].ti=ctot;
            if(st[u]>st[v]) swap(u,v);
            int z=lca(u,v);
            if(z==u) Q[qtot].l=st[u],Q[qtot].r=st[v];
            else Q[qtot].l=ed[u],Q[qtot].r=st[v],Q[qtot].z=z;
        }
    }
    sort(Q+1,Q+qtot+1,cmp);
    int l=1,r=0,ti=0;
    memset(cnt,0,sizeof(cnt));
    memset(vis,false,sizeof(vis));
    for(int i=1;i<=qtot;i++)
    {
        if(r<Q[i].r) {for(r++;r<=Q[i].r;r++) deal(loc[r]); r--;}
        if(r>Q[i].r) {for(;r>Q[i].r;r--) deal(loc[r]); }
        if(l<Q[i].l) {for(;l<Q[i].l;l++) deal(loc[l]); }
        if(l>Q[i].l) {for(l--;l>=Q[i].l;l--) deal(loc[l]); l++;}
        if(Q[i].z) deal(Q[i].z);
        while(ti<Q[i].ti) modify(++ti);
        while(ti>Q[i].ti) modify(ti--);
        ans[Q[i].id]=res;
        if(Q[i].z) deal(Q[i].z);
    }
    for(int i=1;i<=qtot;i++) printf("%lld\n",ans[i]);
    return 0;
```

```
}
```

## 1.18 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;
}
```

## 1.19 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;
}
```

## 1.20 Segment Tree Merge

```
#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,sz,tot,root[MAXN],a[MAXN],ans[MAXN];
int cnt[MAXM],lson[MAXM],rson[MAXM];
vector<int> G[MAXN];
vector<int> v;
//ask for how many a[j]<a[i] if j is in the subtree of i for every i from 1..n
//time complexity:O(nlogn)
void pushup(int k)
{
    cnt[k]=cnt[lson[k]]+cnt[rson[k]];
}
void build(int &k,int l,int r,int p)
{
    k=++tot;
    if(l==r)
    {
        cnt[k]=1;
        return;
    }
    int mid=(l+r)/2;
    if(p<=mid) build(lson[k],l,mid,p);
    else build(rson[k],mid+1,r,p);
    pushup(k);
}
int merge(int x,int y,int l,int r)
{
    if(!x) return y;
    if(!y) return x;
    if(l==r)
    {
        cnt[x]+=cnt[y];
        return x;
    }
    int mid=(l+r)/2;
    lson[x]=merge(lson[x],lson[y],l,mid);
    rson[x]=merge(rson[x],rson[y],mid+1,r);
    pushup(x);
    return x;
```

```
}
int query(int k,int l,int r,int x)
{
    if(k==0) return 0;
    if(r<x) return 0;
    if(l>=x) return cnt[k];
    int mid=(l+r)/2;
    return query(lson[k],l,mid,x)+query(rson[k],mid+1,r,x);
}
void dfs(int v,int p)
{
    for(auto to:G[v])
    {
        if(to==p) continue;
        dfs(to,v);
        root[v]=merge(root[v],root[to],1,sz);
    }
    ans[v]=query(root[v],1,sz,a[v]+1);
}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
        v.push_back(a[i]);
    }
    sort(v.begin(),v.end());
    v.erase(unique(v.begin(),v.end()),v.end());
    for(int i=1;i<=n;i++) a[i]=lower_bound(v.begin(),v.end(),a[i])-v.begin()+1;
    sz=(int)v.size();
    for(int i=2;i<=n;i++)
    {
        int p;scanf("%d",&p);
        G[p].push_back(i);G[i].push_back(p);
    }
    for(int i=1;i<=n;i++) build(root[i],1,sz,a[i]);
    dfs(1,0);
    for(int i=1;i<=n;i++) printf("%d\n",ans[i]);
    return 0;
}
```

## 1.21 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;
}
```

## 1.22 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;
        }
    }
```

```
}
```

## 1.23 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;
}
```

## 1.24 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);
}
```

## 1.25 Mo's Algorithm on Tree

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 200005
#define MAXM 200005
#define MAXLOGN 20
#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,st[2*MAXN],ed[2*MAXN],loc[2*MAXN],val[MAXN];
vector<int> dis;
vector<int> G[MAXN];
int spt[MAXLOGN+1][4*MAXN];
int vs[MAXN*2],depth[MAXN*2];
int id[MAXN],pos[2*MAXN],cnt[MAXN],now,sum;
bool vis[MAXN];
vector<int> v;
void dfs(int v,int p,int d,int &k)
{
    st[v]=++tot; loc[tot]=v;
    id[v]=k;
    vs[k]=v;
    depth[k++]=d;
    for(auto to:G[v])
    {
        if(to==p) continue;
        dfs(to,v,d+1,k);
        vs[k]=v;
        depth[k++]=d;
```

```
    }
    ed[v]=++tot;
    loc[tot]=v;
}
void add_edge(int u,int v)
{
    G[u].push_back(v);
    G[v].push_back(u);
}
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) spt[0][i]=i;
    for(int i=1;1<<i<n;++i)
        for(int j=1;j+(1<<i)-1<=n;++j)
            spt[i][j]=getMin(spt[i-1][j],spt[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(spt[k][l],spt[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]))];
}

struct qry
{
    int u,v;
    int l,r,z,id;
}Q[MAXM];
bool cmp(qry a,qry b)
{
    return pos[a.l]==pos[b.l]?a.r<b.r:pos[a.l]<pos[b.l];
}
void deal(int x)
{
    if(!vis[x])
    {
        if(!cnt[val[x]]) now++;
        cnt[val[x]]++;
        sum++;
    }
    else
    {
        cnt[val[x]]--;
        assert(cnt[val[x]]>=0);
```

```
        if(!cnt[val[x]]) now--;
        sum--;
    }
    vis[x]^=1;
}
int ans[MAXM];
const int blocks=200;
int main()
{
    scanf("%d%d",&n,&q);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&val[i]);
        dis.push_back(val[i]);
    }
    sort(dis.begin(),dis.end());
    dis.erase(unique(dis.begin(),dis.end()),dis.end());
    for(int i=1;i<=n;i++) val[i]=lower_bound(dis.begin(),dis.end(),val[i])-dis.begin();
    for(int i=0;i<n-1;i++)
    {
        int u,v;
        scanf("%d%d",&u,&v);
        add_edge(u,v);
    }
    init(n);
    assert(tot==2*n);
    for(int i=1;i<=tot;i++) pos[i]=i/blocks+1;
    for(int i=1;i<=q;i++)
    {
        Q[i].id=i;
        int u,v;
        scanf("%d%d",&u,&v);
        Q[i].u=u; Q[i].v=v;
        if(st[u]>st[v]) swap(u,v);
        int z=lca(u,v);
        if(z==u) Q[i].l=st[u],Q[i].r=st[v];
        else Q[i].l=ed[u],Q[i].r=st[v],Q[i].z=z;
    }
    sort(Q+1,Q+q+1,cmp);
    int l=1,r=0;
    memset(cnt,0,sizeof(cnt));
    memset(vis,false,sizeof(vis));
    for(int i=1;i<=q;i++)
    {
        if(r<Q[i].r) {for(r++;r<=Q[i].r;r++) deal(loc[r]); r--;}
        if(r>Q[i].r) {for(;r>Q[i].r;r--) deal(loc[r]); }
        if(l<Q[i].l) {for(;l<Q[i].l;l++) deal(loc[l]); }
        if(l>Q[i].l) {for(l--;l>=Q[i].l;l--) deal(loc[l]); l++;}
        if(Q[i].z) deal(Q[i].z);
        ans[Q[i].id]=now;
        if(Q[i].z) deal(Q[i].z);
    }
    for(int i=1;i<=q;i++) printf("%d\n",ans[i]);
    return 0;
}
```

## 1.26 Virtual Tree

```cpp
#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;
}
```

## 1.27 Young Tableaux

```
#include<bits/stdc++.h>
#define MAXN 5005
#define MAXM 305
#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];
int t[MAXN];
int young[MAXM][MAXM];
void ins(int v)
{
    for(int i=1;;i++)
    {
        if(t[i]==0||v>=young[i][t[i]])
        {
            young[i][++t[i]]=v;
            break;
        }
        int pos=upper_bound(young[i]+1,young[i]+t[i]+1,v)-young[i];
        swap(young[i][pos],v);
    }
}
```

```
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
        ins(a[i]);
    }
    int x=0;
    for(int i=1;;i++)
    {
        x+=t[i];
        printf("%d\n",x);
        if(x==n) break;
    }
    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 operator ^(const Point &b)const
    {
        return x*b.y - y*b.x;
    }
    //             (  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
    {
        if(angle!=ta.angle) return angle<ta.angle;
        return ((s - ta.s)^(ta.e - ta.s)) < 0;
    }
    //
```

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

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

//          /
double dis(const Point &_Off, bool isSegment = false)
{
    ///
    pton();
    //
    double td = (a * _Off.x + b * _Off.y + c) / sqrt(a * a + b * b);
    //
    if(isSegment)
    {
        double xp = (b * b * _Off.x - a * b * _Off.y - a * c) / ( a * a + b * b);
        double yp = (-a * b * _Off.x + a * a * _Off.y - b * c) / (a * a + b * b);
        double xb = max(s.x, e.x);
```

```
        double yb = max(s.y, e.y);
        double xs = s.x + e.x - xb;
        double ys = s.y + e.y - yb;
       if(xp > xb + eps || xp < xs - eps || yp > yb + eps || yp < ys - eps)
            td = min( getdis(_Off,s), getdis(_Off,e));
    }
    return fabs(td);
}

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

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

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

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

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

{
    double tmp = xmult( gsort, ta, tb);
```

```
    if( fabs( tmp) < eps)
        return getdis( gsort, ta) < getdis( gsort, tb);
    else if( tmp > 0)
        return 1;
    return 0;
}

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


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

```
        }
        else if( Line::isCrossSS( ln, l))
            ret++;
    }
    return ret&1;
}

//               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 = getdis(pt[i+1],pt[j+1]), ans =
            make_pair(pt[i+1],pt[j+1]);
    }
    return ans;
}

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

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

```
        for(i = 2; i < ln; i ++)
        {
            while(bot < top && judege(dq[top],dq[top-1],L[i]) > 0)
                top --;
            while(bot < top && judege(dq[bot],dq[bot+1],L[i]) > 0)
                bot ++;
            dq[++ top] = L[i];
        }
        while(bot < top && judege(dq[top],dq[top-1],dq[bot]) > 0)
            top --;
        while(bot < top && judege(dq[bot],dq[bot+1],dq[top]) > 0)
            bot ++;
        //
        //       if(top <= bot + 1)
        //           return 0;
        dq[++top] = dq[bot];
        for(i = bot; i < top; i ++)
            Line::crossLPt(dq[i],dq[i + 1],pt[n++]);
        return n;
    }
};


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    //
    //
    Point perpencenter()
    {
        Line u,v;
        u.s = c;
```

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

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

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

int main(void)
```

```
{

    return 0;
}
```

## 2.2 Convex Hull

```
#include<cstdio>
#include<cmath>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<algorithm>
#include<vector>
#define MAXN 50005
using namespace std;
double EPS= 1e-10;
double add(double a,double b)
{
    if(abs(a+b)<EPS*(abs(a)+abs(b))) return 0;
    return a+b;
}
struct P
{
    double x,y;
    P(){}
    P(double x,double y):x(x),y(y){}
    P operator +(P p)
    {
        return P(add(x,p.x),add(y,p.y));
    }
    P operator -(P p)
    {
        return P(add(x,-p.x),add(y,-p.y));
    }
    P operator *(double d)
    {
        return P(x*d,y*d);
    }
    double dot(P p)
    {
        return add(x*p.x,y*p.y);
    }
    double det(P p)
    {
        return add(x*p.y,-y*p.x);
    }
};
bool cmp_x(const P& p,const P& q)
{
    if (p.x!=q.x) return p.x<q.x;
    return p.y<q.y;
}
vector<P> convex_hull(P* ps,int n)
{
    sort(ps,ps+n,cmp_x);
    int k=0;
```

```
    vector<P> qs(n*2);
    for(int i=0;i<n;i++)
    {
        while(k>1&&(qs[k-1]-qs[k-2]).det(ps[i]-qs[k-1])<=0) k--;
        qs[k++]=ps[i];
    }
    for(int i=n-2,t=k;i>=0;i--)
    {
        while(k>t&&(qs[k-1]-qs[k-2]).det(ps[i]-qs[k-1])<=0) k--;
        qs[k++]=ps[i];
    }
    qs.resize(k-1);
    return qs;
}
double dist (P p,P q)
{
    return (p-q).dot(p-q);
}
int N;
P ps[MAXN];
int main()
{
    scanf("%d",&N);
    for(int i=0;i<N;i++)
        scanf("%lf %lf",&ps[i].x,&ps[i].y);
    vector<P> qs=convex_hull(ps,N);
    double res=0;
    for(int i=0;i<qs.size();i++)
    {
        for(int j=0;j<i;j++)
        {
            res=max(res,dist(qs[i],qs[j]));
        }
    }
    printf("%.0f",res);
}
```

## 2.3 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++)
         {
               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;//                                   (                              )                 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;


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.dis2(Q);ans2=p1.dis2(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 Bipartite Matching

```
#include<bits/stdc++.h>
#define MAXN 10005
using namespace std;
int V;
vector<int> G[MAXN];
int match[MAXN];
bool used[MAXN];
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=1;v<=V;v++)
    {
        if(match[v]<0)
        {
            memset(used,0,sizeof(used));
            if(dfs(v))
            {
                res++;
            }
        }
    }
```

```
    return res;
}
```

## 3.2 Block Cut Tree

```
#pragma GCC optimize(3)
#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;
int n,m,tot,t,bcc_cnt,mcnt;
vector<int> G[MAXN],bcc[MAXN];
int st[MAXN],dfn[MAXN],low[MAXN],bccno[MAXN];
bool art[MAXN];
vector<int> tree[MAXN];
int id[MAXN];
int N;
//block-cut tree:
//vertex-biconnected components are connected by their shared articulation point
void dfs(int v,int p,int &tot)
{
    dfn[v]=low[v]=++tot;
    st[t++]=v;
    for(auto to:G[v])
    {
        if(to==p) continue;
        if(!dfn[to])
        {
            dfs(to,v,tot);
            low[v]=min(low[v],low[to]);
            if(low[to]>=dfn[v])
            {
                art[v]=(dfn[v]>1||dfn[to]>2);
                bcc_cnt++;
                bcc[bcc_cnt].push_back(v); bccno[v]=bcc_cnt;
                while(bcc[bcc_cnt].back()!=v)
                {
                    bccno[st[t-1]]=bcc_cnt;
                    bcc[bcc_cnt].push_back(st[t-1]),t--;
                }
            }
        }
        else low[v]=min(low[v],dfn[to]);
    }
}
int tarjan()
{
    bcc_cnt=t=0;
    memset(dfn,0,sizeof(dfn));
    memset(art,false,sizeof(art));
    for(int i=1;i<=n;i++) if(!dfn[i]) dfs(i,-1,tot=0);
```

```
    return bcc_cnt;
}
void build_block_cut_tree()
{
    tarjan();N=0;
    for(int i=1;i<=n;i++) if(art[i]) id[i]=++N;
    for(int i=1;i<=bcc_cnt;i++)
    {
        N++;
        for(auto v:bcc[i])
        {
            if(!art[v]) id[v]=N;
            else
            {
                tree[id[v]].push_back(N);
                tree[N].push_back(id[v]);
            }
        }
    }
}
```

## 3.3 Bridge Tree

```
#pragma GCC optimize(3)
#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;
int n,m,tot,t,bcc_cnt,mcnt;
vector<int> G[MAXN],bcc[MAXN];
int st[MAXN],dfn[MAXN],low[MAXN],bccno[MAXN];
int U[MAXM],V[MAXM];
bool isbridge[MAXM];
vector<int> tree[MAXN];
//bridge tree:
//edge-biconnected components are connected by bridges
void add_edge(int u,int v)
{
    U[++mcnt]=u;V[mcnt]=v;
    G[u].push_back(mcnt);G[v].push_back(mcnt);
}
int adj(int u,int e)
{
    return U[e]==u?V[e]:U[e];
}
void dfs1(int v,int edge)
{
    dfn[v]=low[v]=++tot;
    st[t++]=v;
    for(auto e:G[v])
    {
```

```
        if(e==edge) continue;
        int to=adj(v,e);
        if(!dfn[to])
        {
            dfs1(to,e);
            low[v]=min(low[v],low[to]);
        }
        else low[v]=min(low[v],dfn[to]);
    }
    if(low[v]==dfn[v]&&edge!=-1) isbridge[edge]=true;
}
void dfs2(int v)
{
    dfn[v]=1;
    bccno[v]=bcc_cnt;
    bcc[bcc_cnt].push_back(v);
    for(auto e:G[v])
    {
        int to=adj(v,e);
        if(isbridge[e]) continue;
        if(!dfn[to]) dfs2(to);
    }
}
int tarjan()
{
    bcc_cnt=tot=0;
    memset(dfn,0,sizeof(dfn));
    memset(isbridge,false,sizeof(isbridge));
    for(int i=1;i<=n;i++) if(!dfn[i]) dfs1(i,-1);
    memset(dfn,0,sizeof(dfn));
    for(int i=1;i<=n;i++)
    {
        if(!dfn[i])
        {
            bcc_cnt++;
            dfs2(i);
        }
    }
    return bcc_cnt;
}
void build_bridge_tree()
{
    tarjan();
    for(int i=1;i<=mcnt;i++)
    {
        if(isbridge[i])
        {
            int u=bccno[U[i]],v=bccno[V[i]];
            tree[u].push_back(v);tree[v].push_back(u);
        }
    }
}
```

## 3.4 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],label[MAXN];
vector<int> st[MAXN];
bool vis[MAXN];
vector<int> peo;
void MCS()
{
    memset(vis,0,sizeof(vis));
    memset(h,0,sizeof(h));
    int cur=0;
    for(int i=1;i<=n;i++) st[0].push_back(i);
    for(int i=n;i>=1;i--)
    {
        while(1)
        {
            while(st[cur].size()==0) cur--;
            int now=st[cur].back();
            st[cur].pop_back();if(vis[now]) continue;
            vis[now]=true;label[now]=i;
            for(auto to:G[now])
            {
                if(vis[to]) continue;
                h[to]++; st[h[to]].push_back(to); cur=max(cur,h[to]);
            }
            break;
        }
    }
    reverse(peo.begin(),peo.end());
}
```

## 3.5 Common Matching(Blossom)

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

```
    {
        tra[++cnt]=x;
        treec[x]=true;
    }
}
int root(int x){return f[x]?f[x]=root(f[x]):x;}

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

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

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

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

bool find(int x)
```

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

void matching()
{
    ans=0;
    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;
}
```

## 3.6 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<(int)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;
}
```

## 3.7 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,(int)G[to].size()});
    G[to].push_back((edge){from,0,(int)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<(int)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<(int)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;
    }
}
```

## 3.8 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<(int)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]);
	}
}
void build_dominator_tree(int r)
{
    dfs(r);int N=t;
	for(int i=N;i>=1;i--)
	{
		for(int j=0;j<(int)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<(int)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]);
	}
    for(int i=1;i<=N;i++) bucket[i].clear(),rG[i].clear();
}
```

## 3.9 Dynamic Bridge

```
/**
 * Author: Sergey Kopeliovich (Burunduk30@gmail.com)
 * Total Time = O(mlogm)
 */

#include <ctime>
#include <cassert>
#include <cstdio>
#include <cstring>

#include <algorithm>
#include <map>

using namespace std;

#define forn(i, n) for (int i = 0; i < (int)(n); i++)
#define forit(i, a) for (__typeof((a).begin()) i = (a).begin(); i != (a).end(); i++)
#define mp make_pair

typedef pair <int, int> pii;

template <class T> inline void relaxMin( T &a, T b ) { a = min(a, b); }

/* Main part */

const int maxN = (int)1e5;
const int maxM = (int)1e5;

struct query
{
  int a, b, L, R;

  query() { }
  query( int _a, int _b, int _L, int _R ) : a(_a), b(_b), L(_L), R(_R) { }
};

int n, m, qn, res[maxM + 1];
query q[maxM];

const int maxMem = (int)1e7;
const int maxE = (int)1e6;

int mpos = 0, en, next[maxE], to[maxE], w[maxE];
char mem[maxMem];

template <class T> T *getMem( int n )
{
  char *r = mem + mpos;
  mpos += n * sizeof(T);
  assert(mpos <= maxMem);
  return (T *)r;
}
```

```
void addE( int a, int b, int x, int *head )
{
  assert(en + 2 <= maxE);
  next[en] = head[a], to[en] = b, w[en] = x, head[a] = en++;
  next[en] = head[b], to[en] = a, w[en] = x, head[b] = en++;
}

int curT, cc, used[maxN], T[maxN], minT[maxN];
int sp, ss[maxN];
int vnX, color[maxN];
int enX, ea[maxM], eb[maxM], ew[maxM];

// find components of edge-2-connectivity and bridges
void getComp( int old_sp )
{
  while (sp > old_sp)
    color[ss[--sp]] = vnX;
  vnX++;
}

void dfs( int v, int pr, int *head )
{
  int cnt = 0;
  ss[sp++] = v;
  used[v] = cc;
  minT[v] = T[v] = curT++;
  for (int e = head[v]; e != -1; e = next[e])
  {
    int x = to[e];
    if (x != pr || ++cnt > 1)
    {
      if (used[x] != cc)
      {
        int old_sp = sp;
        dfs(x, v, head);
        if (minT[x] > T[v])
          getComp(old_sp), ew[enX] = w[e], ea[enX] = v, eb[enX++] = x;
      }
      relaxMin(minT[v], minT[x]);
    }
  }
}

// determine important vertices in tree
int paint( int v, int *head )
{
  int num = 0;
  used[v] = cc;
  for (int e = head[v]; e != -1; e = next[e])
    if (used[to[e]] != cc)
      num += paint(to[e], head);
  color[v] |= (num >= 2);
  return color[v] || num;
}

// consolidate edges
int findEdges( int v, int start, int curLen, int *head )
{
  int ret = 0;
```

```
  used[v] = cc;
  if (color[v] && curLen > 0)
  {
    ea[enX] = v, eb[enX] = start, ew[enX++] = curLen;
    start = v, curLen = 0, ret = 1;
  }
  for (int e = head[v]; e != -1; e = next[e])
    if (used[to[e]] != cc && findEdges(to[e], start, curLen + w[e], head))
      w[e] = 0, w[e ^ 1] = 0, ret = 1;
  return ret;
}

// main procedure
void newGraph( int qn, query *q, int &vn, int *head, int &old_en )
{
  fill(head, head + vn, -1), en = old_en, vn = vnX;
  forn(i, enX)
    addE(color[ea[i]], color[eb[i]], ew[i], head);
  forn(i, qn)
    q[i].a = color[q[i].a], q[i].b = color[q[i].b];
}

void Solve( int L, int R, int qn2, query *q2, int vn2, int *head2, int have )
{
  query *q = getMem<query>(qn2);
  int old_en = en, old_mpos = mpos;
  int qn = 0, vn = vn2;
  int *head = getMem<int>(vn);

  /* Process all already obvious queries */
  memcpy(head, head2, sizeof(head2[0]) * vn);
  forn(i, qn2)
    if (q2[i].L < L && q2[i].R >= R)
      addE(q2[i].a, q2[i].b, 1, head);
    else if (q2[i].R >= L && q2[i].L < R)
      q[qn++] = q2[i];

  /* Consolidate components of edge-2-connectivity */
  cc++, curT = 0;
  vnX = enX = sp = 0;
  forn(i, vn)
    if (used[i] != cc)
      dfs(i, -1, head), getComp(0);
  newGraph(qn, q, vn, head, old_en);

  /* Determine all important edges */
  forn(i, vn)
    color[i] = 0;
  forn(i, qn)
    color[q[i].a] = color[q[i].b] = 1;
  cc++;
  forn(i, vn)
    if (used[i] != cc)
      paint(i, head);

  /* Reduce the graph */
  enX = 0, cc++;
  forn(i, vn)
    if (used[i] != cc && color[i])
```

```
      findEdges(i, i, 0, head);
  forn(i, vn)
    for (int e = head[i]; e != -1; e = next[e])
      if (w[e] > 0)
        have += w[e];
  vnX = 0;
  forn(i, vn)
    if (color[i])
      color[i] = vnX++;
  newGraph(qn, q, vn, head, old_en);

  /* Recursion continues... */
  mpos -= sizeof(int) * (vn2 - vn);
  if (L == R)
    res[L] = have / 2;
  else
  {
    int M = (L + R) / 2;
    Solve(L, M, qn, q, vn, head, have);
    Solve(M + 1, R, qn, q, vn, head, have);
  }
  en = old_en, mpos = old_mpos;
}

void Read()
{
  #define NAME "bridges3"
  assert(freopen(NAME ".in", "r", stdin));
  assert(freopen(NAME ".out", "w", stdout));

  assert(scanf("%d%d", &n, &m) == 2);
  assert(1 <= n && n <= maxN);
  assert(0 <= m && m <= maxM);

  map <pii, int> L;
  forn(i, m)
  {
    char com[9];
    int a, b;
    assert(scanf("%s", com) == 1);
    assert(scanf("%d%d", &a, &b) == 2);
    assert(1 <= a && a <= n && 1 <= b && b <= n && a != b);
    a--, b--;
    if (a > b)
      swap(a, b);

    pii p = mp(a, b);
    assert(!strcmp(com, "ADD") == !L.count(p));
    if (L.count(p))
    {
      q[qn++] = query(a, b, L[p], i);
      L.erase(p);
    }
    else
      L[p] = i;
  }
  forit(it, L)
    q[qn++] = query(it->first.first, it->first.second, it->second, m);
}
```

```
void TimeStamp( const char *s )
{
  fprintf(stderr, "[%05.2f] %s\n", (double)clock() / CLOCKS_PER_SEC, s);
}

int main()
{
  Read();

  TimeStamp("Data is read");

  int *head = getMem<int>(n);
  fill(head, head + n, -1);

  TimeStamp("Memory is allocated");

  Solve(0, m, qn, q, n, head, 0);

  TimeStamp("Problem is solved");

  forn(i, m)
    printf("%d\n", res[i + 1]);

  TimeStamp("Result is outputed");
  return 0;
}**
 * Author: Sergey Kopeliovich (Burunduk30@gmail.com)
 * Total Time = O(mlogm)
 */

#include <ctime>
#include <cassert>
#include <cstdio>
#include <cstring>

#include <algorithm>
#include <map>

using namespace std;

#define forn(i, n) for (int i = 0; i < (int)(n); i++)
#define forit(i, a) for (__typeof((a).begin()) i = (a).begin(); i != (a).end(); i++)
#define mp make_pair

typedef pair <int, int> pii;

template <class T> inline void relaxMin( T &a, T b ) { a = min(a, b); }

/* Main part */

const int maxN = (int)1e5;
const int maxM = (int)1e5;

struct query
{
  int a, b, L, R;

  query() { }
```

```
  query( int _a, int _b, int _L, int _R ) : a(_a), b(_b), L(_L), R(_R) { }
};

int n, m, qn, res[maxM + 1];
query q[maxM];

const int maxMem = (int)1e7;
const int maxE = (int)1e6;

int mpos = 0, en, next[maxE], to[maxE], w[maxE];
char mem[maxMem];

template <class T> T *getMem( int n )
{
  char *r = mem + mpos;
  mpos += n * sizeof(T);
  assert(mpos <= maxMem);
  return (T *)r;
}

void addE( int a, int b, int x, int *head )
{
  assert(en + 2 <= maxE);
  next[en] = head[a], to[en] = b, w[en] = x, head[a] = en++;
  next[en] = head[b], to[en] = a, w[en] = x, head[b] = en++;
}

int curT, cc, used[maxN], T[maxN], minT[maxN];
int sp, ss[maxN];
int vnX, color[maxN];
int enX, ea[maxM], eb[maxM], ew[maxM];

// find components of edge-2-connectivity and bridges
void getComp( int old_sp )
{
  while (sp > old_sp)
    color[ss[--sp]] = vnX;
  vnX++;
}

void dfs( int v, int pr, int *head )
{
  int cnt = 0;
  ss[sp++] = v;
  used[v] = cc;
  minT[v] = T[v] = curT++;
  for (int e = head[v]; e != -1; e = next[e])
  {
    int x = to[e];
    if (x != pr || ++cnt > 1)
    {
      if (used[x] != cc)
      {
        int old_sp = sp;
        dfs(x, v, head);
        if (minT[x] > T[v])
          getComp(old_sp), ew[enX] = w[e], ea[enX] = v, eb[enX++] = x;
      }
      relaxMin(minT[v], minT[x]);
```

```
    }
  }
}

// determine important vertices in tree
int paint( int v, int *head )
{
  int num = 0;
  used[v] = cc;
  for (int e = head[v]; e != -1; e = next[e])
    if (used[to[e]] != cc)
      num += paint(to[e], head);
  color[v] |= (num >= 2);
  return color[v] || num;
}

// consolidate edges
int findEdges( int v, int start, int curLen, int *head )
{
  int ret = 0;
  used[v] = cc;
  if (color[v] && curLen > 0)
  {
    ea[enX] = v, eb[enX] = start, ew[enX++] = curLen;
    start = v, curLen = 0, ret = 1;
  }
  for (int e = head[v]; e != -1; e = next[e])
    if (used[to[e]] != cc && findEdges(to[e], start, curLen + w[e], head))
      w[e] = 0, w[e ^ 1] = 0, ret = 1;
  return ret;
}

// main procedure
void newGraph( int qn, query *q, int &vn, int *head, int &old_en )
{
  fill(head, head + vn, -1), en = old_en, vn = vnX;
  forn(i, enX)
    addE(color[ea[i]], color[eb[i]], ew[i], head);
  forn(i, qn)
    q[i].a = color[q[i].a], q[i].b = color[q[i].b];
}

void Solve( int L, int R, int qn2, query *q2, int vn2, int *head2, int have )
{
  query *q = getMem<query>(qn2);
  int old_en = en, old_mpos = mpos;
  int qn = 0, vn = vn2;
  int *head = getMem<int>(vn);

  /* Process all already obvious queries */
  memcpy(head, head2, sizeof(head2[0]) * vn);
  forn(i, qn2)
    if (q2[i].L < L && q2[i].R >= R)
      addE(q2[i].a, q2[i].b, 1, head);
    else if (q2[i].R >= L && q2[i].L < R)
      q[qn++] = q2[i];

  /* Consolidate components of edge-2-connectivity */
  cc++, curT = 0;
```

```
  vnX = enX = sp = 0;
  forn(i, vn)
    if (used[i] != cc)
      dfs(i, -1, head), getComp(0);
  newGraph(qn, q, vn, head, old_en);

  /* Determine all important edges */
  forn(i, vn)
    color[i] = 0;
  forn(i, qn)
    color[q[i].a] = color[q[i].b] = 1;
  cc++;
  forn(i, vn)
    if (used[i] != cc)
      paint(i, head);

  /* Reduce the graph */
  enX = 0, cc++;
  forn(i, vn)
    if (used[i] != cc && color[i])
      findEdges(i, i, 0, head);
  forn(i, vn)
    for (int e = head[i]; e != -1; e = next[e])
      if (w[e] > 0)
        have += w[e];
  vnX = 0;
  forn(i, vn)
    if (color[i])
      color[i] = vnX++;
  newGraph(qn, q, vn, head, old_en);

  /* Recursion continues... */
  mpos -= sizeof(int) * (vn2 - vn);
  if (L == R)
    res[L] = have / 2;
  else
  {
    int M = (L + R) / 2;
    Solve(L, M, qn, q, vn, head, have);
    Solve(M + 1, R, qn, q, vn, head, have);
  }
  en = old_en, mpos = old_mpos;
}

void Read()
{
  #define NAME "bridges3"
  assert(freopen(NAME ".in", "r", stdin));
  assert(freopen(NAME ".out", "w", stdout));

  assert(scanf("%d%d", &n, &m) == 2);
  assert(1 <= n && n <= maxN);
  assert(0 <= m && m <= maxM);

  map <pii, int> L;
  forn(i, m)
  {
    char com[9];
    int a, b;
```

```
    assert(scanf("%s", com) == 1);
    assert(scanf("%d%d", &a, &b) == 2);
    assert(1 <= a && a <= n && 1 <= b && b <= n && a != b);
    a--, b--;
    if (a > b)
      swap(a, b);

    pii p = mp(a, b);
    assert(!strcmp(com, "ADD") == !L.count(p));
    if (L.count(p))
    {
      q[qn++] = query(a, b, L[p], i);
      L.erase(p);
    }
    else
      L[p] = i;
  }
  forit(it, L)
    q[qn++] = query(it->first.first, it->first.second, it->second, m);
}

void TimeStamp( const char *s )
{
  fprintf(stderr, "[%05.2f] %s\n", (double)clock() / CLOCKS_PER_SEC, s);
}

int main()
{
  Read();

  TimeStamp("Data is read");

  int *head = getMem<int>(n);
  fill(head, head + n, -1);

  TimeStamp("Memory is allocated");

  Solve(0, m, qn, q, n, head, 0);

  TimeStamp("Problem is solved");

  forn(i, m)
    printf("%d\n", res[i + 1]);

  TimeStamp("Result is outputed");
  return 0;
}
```

## 3.10 Dynamic Connectivity

```
#include<bits/stdc++.h>
#define MAXN 300005
#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,x,y;
char str[2];
vector<P> edges[4*MAXN];
bool ask[MAXN];
int p[MAXN],r[MAXN],sz[MAXN];
int ans[MAXN];
int num;
struct update
{
    int x,y;
    bool addrk;
};
update st[MAXN];
int t;
void init(int n)
{
    for(int i=1;i<=n;i++)
    {
        p[i]=i;
        r[i]=0;
    }
}
int find(int x)
{
   while(p[x]!=x) x=p[x];
   return x;
}
bool unite(int x,int y)
{
    x=find(x);
    y=find(y);
    if(x==y) return false;
    num--;
    if(r[x]<r[y])
    {
        p[x]=y;
        st[t++]=(update){x,y,false};
    }
    else
    {
        p[y]=x;
        st[t++]=(update){y,x,r[x]==r[y]};
        if(r[x]==r[y]) r[x]++;
    }
    return true;
}
void undo()
{
    assert(t);
    int x=st[t-1].x,y=st[t-1].y;
    //printf("undo %d %d %d\n",x,y,st[t-1].addrk);
    p[x]=x;p[y]=y;
    if(st[t-1].addrk) r[y]--;
    t--;num++;
}
bool same(int x,int y)
{
```

```
    return find(x)==find(y);
}
void add_edge(int k,int l,int r,int x,int y,int u,int v)
{
    if(x>r||l>y) return;
    if(l>=x&&r<=y)
    {
        edges[k].push_back(P(u,v));
        return;
    }
    int mid=(l+r)/2;
    add_edge(k*2,l,mid,x,y,u,v);add_edge(k*2+1,mid+1,r,x,y,u,v);
}
void solve(int k,int l,int r)
{
    if(l>r) return;
    int cnt=0;
    for(auto e:edges[k]) if(unite(e.F,e.S)) cnt++;
    if(l==r)
    {
        if(ask[l]) ans[l]=num;
        for(int i=0;i<cnt;i++) undo();
        return;
    }
    int mid=(l+r)/2;
    solve(k*2,l,mid);solve(k*2+1,mid+1,r);
    //printf("cnt %d %d %d\n",l,r,cnt);
    for(int i=0;i<cnt;i++) undo();
}
map<P,int> mp;
int main()
{
    scanf("%d%d",&n,&k);num=n;init(n);
    memset(ask,false,sizeof(ask));
    for(int i=1;i<=k;i++)
    {
        scanf("%s",str);
        if(str[0]=='?')
        {
            ask[i]=true;
            continue;
        }
        scanf("%d%d",&x,&y);
        if(x>y) swap(x,y);
        if(str[0]=='+') mp[P(x,y)]=i;
        else
        {
            add_edge(1,1,k,mp[P(x,y)],i-1,x,y);
            mp[P(x,y)]=-1;
        }
    }
    for(auto p:mp) if(p.S!=-1) add_edge(1,1,k,p.S,k,p.F.F,p.F.S);
    solve(1,1,k);
    for(int i=1;i<=k;i++) if(ask[i]) printf("%d\n",ans[i]);
    return 0;
}
```

## 3.11 Ear Decomposition

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
#include<ext/pb_ds/priority_queue.hpp>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
using namespace __gnu_pbds;
typedef long long ll;
typedef pair<int,int> P;
typedef tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update> ordered_set;
typedef __gnu_pbds::priority_queue<int,greater<int>,pairing_heap_tag> pq;
int n,m;
vector<int> G[MAXN];
int dep[MAXN];
vector<vector<int> > ears;
vector<int> path[MAXN];
vector<int> back[MAXN];
int low[MAXN];
int vis[MAXN];
bool f=true;
void cmin(int &a,int b)
{
    if(dep[b]<dep[a]) a=b;
}
void add_ear(int to,vector<int> &a)
{
    reverse(a.begin(),a.end());
    a.push_back(to);
    ears.push_back(a);
}
void dfs(int v,int p,int d)
{
    if(!f) return;
    vis[v]=1;dep[v]=d;low[v]=v;
    int used=0;
    for(auto to:G[v])
    {
        if(to==p) continue;
        if(!vis[to])
        {
            dfs(to,v,d+1);
            path[to].push_back(v);
            if(!used||dep[low[to]]<dep[low[v]])
            {
                if(used) add_ear(low[v],path[v]);
                swap(path[v],path[to]);
                low[v]=low[to];
                used=1;
            }
            else add_ear(low[to],path[to]);
        }
```

```
        else if(vis[to]==1) back[v].push_back(to);
    }
    vis[v]=2;
    for(int i=0;i<(int)back[v].size();i++)
    {
        int u=back[v][i];
        if(!used||dep[u]<dep[low[v]])
        {
            if(used) add_ear(low[v],path[v]);
            path[v].clear();path[v].push_back(v);
            low[v]=u;
            used=1;
        }
        else ears.emplace_back((vector<int>){v,u});
    }
   /* printf("%d %d %d\n",v,dep[v],low[v]);
    printf("chain %d\n",v);
    for(auto x:path[v]) printf("%d ",x);
    puts(""); */
    if(dep[low[v]]==dep[v]&&v!=1) f=false;
}
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);
    }
    dfs(1,0,0);
    if(!f) puts("-1");
    else
    {
        add_ear(1,path[1]);
        printf("%d\n",(int)ears.size());
        for(int i=0;i<(int)ears.size();i++)
        {
            printf("%d ",(int)ears[i].size()-1);
            for(int j=0;j<(int)ears[i].size();j++)
                printf("%d ",ears[i][j]);
            puts("");
        }
    }
    return 0;
}
```

## 3.12 Floyd-Warshall

```
#include<bits/stdc++.h>
#define MAXN 505
using namespace std;
int n,d[MAXN][MAXN];
void floyd_warshall()
{
    for(int k=1;k<=n;k++)
        for(int i=1;i<=n;i++)
```

```
        for(int j=1;j<=n;j++) d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
}
```

## 3.13 Ford-Fulkerson

```
#include<bits/stdc++.h>
#define MAXV 1000
#define MAXE 10000
#define INF 1000000
using namespace std;
struct edge{int to,cap,rev;};
bool used[MAXV];
int V;
vector<edge> G[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});
}
int dfs(int v,int t,int f)
{
    if(v==t) return f;
    used[v]=true;
    for(int i=0;i<G[v].size();i++)
    {
        edge &e=G[v][i];
        if(!used[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(;;)
    {
        memset(used,0,sizeof(used));
        int f=dfs(s,t,INF);
        if(f==0) return flow;
        flow+=f;
    }
}
int main()
{
    return 0;
}
```

## 3.14 Gomory-Hu Tree

```
#include<bits/stdc++.h>
#define MAXV 3005
#define MAXE 50000
#define INF 1000000000
using namespace std;
typedef pair<int,int> P;
typedef long long ll;
struct edge{int to,cap,rev,id;};//id=1 positive edge, id=0 reverse edge
struct edge2{int to,cost;};
struct edge3{int from,to,cap;};
int V,E;
vector<edge> G[MAXV];
vector<edge2> gh[MAXV];
vector<edge3> edges;
int level[MAXV];
int iter[MAXV];
void add_edge(int from,int to,int cap)
{
    edges.push_back((edge3){from,to,cap});
}
void add_all()
{
    for(auto e:edges)
    {
        G[e.from].push_back((edge){e.to,e.cap,(int)G[e.to].size(),1});
        G[e.to].push_back((edge){e.from,0,(int)G[e.from].size()-1,0});
    }
}
void clear_all()
{
    for(int i=1;i<=V;i++) G[i].clear();
}
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<(int)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<(int)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;
    }
}
//0-based!!!
void build_gomory_hu_tree()
{
    vector<int> p(V+1,1),cap(V+1,0);
    for(int s=2;s<=V;s++)
    {
        add_all();
        int t=p[s];
        cap[s]=max_flow(s,t);
        vector<bool> in_cut(V+1,0);
        queue<int> que({s});
        in_cut[s]=true;
        while(!que.empty())
        {
            int v=que.front();
            que.pop();
            for(auto e:G[v])
            {
                if(e.cap>0&&!in_cut[e.to])
                {
                    in_cut[e.to]=true;
                    que.push(e.to);
                }
            }
        }
        for(int v=1;v<=V;v++)
            if(v!=s&&in_cut[v]&&p[v]==t)
                p[v]=s;
        if(in_cut[p[t]])
        {
```

```
            p[s]=p[t];
            p[t]=s;
            swap(cap[s],cap[t]);
        }
        clear_all();
    }
    for(int v=2;v<=V;v++)
    {
        gh[p[v]].push_back((edge2){v,cap[v]});
        gh[v].push_back((edge2){p[v],cap[v]});
    }
}
int main()
{
    scanf("%d%d",&V,&E);
    for(int i=0;i<E;i++)
    {
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        add_edge(u,v,w);
        add_edge(v,u,w);
    }
    build_gomory_hu_tree();
    return 0;
}
```

## 3.15 Hopcrokt-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;
}
```

## 3.16 Kosaraju

```
#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;
}
```

## 3.17 Kuhn-Munkres

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 1005
#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;
int w[MAXN][MAXN];
//minimum weight bipartite matching
ll km(int n,int m)
{
    vector<int> u(n+1),v(m+1),p(m+1),way(m+1);
    for(int i=1;i<=n;i++)
    {
        p[0]=i;
        int j0=0;
        vector<int> minv(m+1,INF);
        vector<char> used(m+1,false);
        do
        {
            used[j0]=true;
            int i0=p[j0],delta=INF,j1;
            for(int j=1;j<=m;++j)
                if(!used[j])
                {
                    int cur=w[i0][j]-u[i0]-v[j];
                    if(cur<minv[j]) minv[j]=cur,way[j]=j0;
                    if(minv[j]<delta) delta=minv[j],j1=j;
                }
```

```
            for(int j=0;j<=m;++j) if(used[j]) u[p[j]]+=delta,v[j]-=delta; else minv[j]-=delta;
            j0=j1;
        }while(p[j0]!=0);
        do
        {
            int j1=way[j0];
            p[j0]=p[j1];
            j0=j1;
        }while(j0);
    }
    ll res=0;
    for(int i=1;i<=m;i++) res+=w[p[i]][i];
    return res;
}
int main()
{
    return 0;
}
```

## 3.18 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,-1,0);
    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)];
}
```

## 3.19 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+1][4*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;
}
```

## 3.20 Min-cost flow(with Dijkstra)

```
#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,(int)G[to].size()});
    G[to].push_back((edge){from,0,-cost,(int)G[from].size()-1});
}
int min_cost_flow(int s,int t,int f)
{
    int res=0;
    fill(h+1,h+V+1,0);
    while(f>0)
    {
        priority_queue<P,vector<P>,greater<P> >que;
        fill(dist+1,dist+V+1,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=1;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;
}
```

## 3.21 Min-cost flow(with SPFA)

```
#include<bits/stdc++.h>
#define MAXV 500005
#define MAXE 1000005
#define INF 1000000000000000000LL
using namespace std;
typedef long long ll;
typedef pair<ll,int> P;
struct edge{int to,cap; ll cost; int rev;};
int n,m,V,s[MAXV],e[MAXV];
ll w[MAXV];
ll dist[MAXV];
```

```
int prevv[MAXV],preve[MAXV];
vector<edge> G[MAXV];
bool inque[MAXV];
void add_edge(int from,int to,int cap,ll cost)
{
    G[from].push_back((edge){to,cap,cost,(int)G[to].size()});
    G[to].push_back((edge){from,0,-cost,(int)G[from].size()-1});
}
ll min_cost_flow(int s,int t,int f)
{
    ll res=0;
    while(f>0)
    {
        queue<int>que;
        fill(dist+1,dist+V+1,INF);
        fill(inque+1,inque+V+1,false);
        dist[s]=0;
        que.push(s);
        while(!que.empty())
        {
            int u=que.front(); que.pop();
            for(int i=0;i<(int)G[u].size();i++)
            {
                if(G[u][i].cap>0&&dist[u]+G[u][i].cost<dist[G[u][i].to])
                {
                    dist[G[u][i].to]=dist[u]+G[u][i].cost;
                    prevv[G[u][i].to]=u;
                    preve[G[u][i].to]=i;
                    if(!inque[G[u][i].to])
                    {
                        inque[G[u][i].to]=true;
                        que.push(G[u][i].to);
                    }
                }
            }
            inque[u]=false;
        }
        if(dist[t]==INF) return -1;
        int d=f;
        for(int v=t;v!=s;v=prevv[v]) d=min(d,G[prevv[v]][preve[v]].cap);
        f-=d;
        res+=1LL*d*dist[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;
}
```

## 3.22 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;
    }
}
```

## 3.23 Minimum Diameter Spanning Tree

```
#include<bits/stdc++.h>
#define MAXN 505
#define MAXM 200005
#define INF 1000000000000000000LL
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<ll,ll> P;
typedef pair<double,ll> PP;
const double eps=1e-2;
ll n,m,d[MAXN][MAXN],save[MAXN][MAXN],pre[MAXN];
double d2[MAXN];
ll u[MAXM],v[MAXM],w[MAXM];
bool used[MAXN];
vector<P> dist[MAXN];
vector<P> MDST;
void floyd_warshall()
{
    for(ll k=1;k<=n;k++)
        for(ll i=1;i<=n;i++)
          for(ll j=1;j<=n;j++) d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
}
pair<P,double> absolute_center()
{
    ll ans=INF;
    ll uu=-1,vv=-1;
    double res=0.0;
    for(ll i=1;i<=n;i++)
    {
```

```
        int sz=(int)dist[i].size();
        if(dist[i][sz-1].F+dist[i][sz-2].F<ans)
        {
            ans=dist[i][sz-1].F+dist[i][sz-2].F;
            uu=vv=i; res=0.0;
        }
    }
    for(ll i=0;i<m;i++)
    {
        memset(used,false,sizeof(used));
        ll now=(int)dist[v[i]].size()-1;
        for(ll j=0;j<(int)dist[u[i]].size();j++)
        {
            used[dist[u[i]][j].S]=true;
            while(now>0&&used[dist[v[i]][now].S]) now--;
            double pos=(dist[u[i]][j].F+dist[v[i]][now].F+w[i])/2.0-dist[u[i]][j].F;
            if(pos<eps||pos-w[i]>eps) continue;
            if(dist[u[i]][j].F+dist[v[i]][now].F+w[i]<ans)
            {
                ans=dist[u[i]][j].F+dist[v[i]][now].F+w[i];
                uu=u[i]; vv=v[i]; res=pos;
            }
        }
    }
    printf("%lld\n",ans);
    return make_pair(P(uu,vv),res);
}
void minimum_diameter_spanning_tree()
{
    MDST.clear();
    auto p=absolute_center();
    fill(d2,d2+n+1,INF); memset(pre,-1,sizeof(pre));
    priority_queue<PP,vector<PP>,greater<PP> > que;
    d2[p.F.F]=p.S; d2[p.F.S]=d[p.F.F][p.F.S]-p.S;
    que.push(PP(d2[p.F.F],p.F.F)); if(p.F.F!=p.F.S) que.push(PP(d2[p.F.S],p.F.S));
    while(!que.empty())
    {
        PP p=que.top(); que.pop();
        ll v=p.S;
        if(d2[v]<p.F) continue;
        for(ll to=1;to<=n;to++)
        {
            if(d2[to]>d2[v]+save[v][to])
            {
                d2[to]=d2[v]+save[v][to];
                pre[to]=v;
                que.push(PP(d2[to],to));
            }
        }
    }
    if(p.F.F!=p.F.S) MDST.push_back(P(p.F.F,p.F.S));
    for(ll i=1;i<=n;i++) if(pre[i]!=-1) MDST.push_back(P(pre[i],i));
}
int main()
{
    scanf("%lld%lld",&n,&m);
    for(ll i=1;i<=n;i++)
        for(ll j=1;j<=n;j++)
            d[i][j]=save[i][j]=(i==j?0:INF);
```

```
    for(ll i=0;i<m;i++)
    {
        scanf("%lld%lld%lld",&u[i],&v[i],&w[i]);
        d[u[i]][v[i]]=d[v[i]][u[i]]=save[u[i]][v[i]]=save[v[i]][u[i]]=w[i];
    }
    floyd_warshall();
    for(ll i=1;i<=n;i++)
    {
        for(ll j=1;j<=n;j++) dist[i].push_back(P(d[i][j],j));
        sort(dist[i].begin(),dist[i].end());
    }
    minimum_diameter_spanning_tree();
    for(auto p:MDST) printf("%lld %lld\n",p.F,p.S);
    return 0;
}
```

## 3.24 SPFA

```
#include<bits/stdc++.h>
#define MAXV 1000
#define MAXE 10000
#define INF 1000000
using namespace std;
struct edge{int to,cost;};
typedef pair<int,int> P;
int V;
vector<edge> G[MAXV];
int d[MAXV];
bool inque[MAXV];
queue<int> que;
void spfa(int s)
{
    fill(d,d+V,INF);
    fill(inque,inque+V,false);
    d[s]=0;
    while(!que.empty()) que.pop();
    que.push(s);
    inque[s]=true;
    while(!que.empty())
    {
        int u=que.front();
        que.pop();
        for(int i=0;i<G[u].size();i++)
        {
            if(d[u]+G[u][i].cost<d[G[u][i].to])
            {
                d[G[u][i].to]=d[u]+G[u][i].cost;
                if(!inque[G[u][i].to])
                {
                    inque[G[u][i].to]=true;
                    que.push(G[u][i].to);
                }
            }
        }
        inque[u]=false;
    }
}
```

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

## 3.25 Square Counting

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
int n,m,deg[MAXN],a[MAXN],cnt[MAXN],r[MAXN];
vector<int> G[MAXN];
vector<int> gr[MAXN];
bool cmp(int x,int y)
{
    return deg[x]<deg[y];
}
int main()
{
    memset(cnt,0,sizeof(cnt));
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++) a[i]=i;
    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);
        deg[v]++;deg[u]++;
    }
    sort(a+1,a+n+1,cmp);
    for(int i=1;i<=n;i++) r[a[i]]=i;
    for(int i=1;i<=n;i++)
        for(auto to:G[i]) if(r[to]>r[i]) gr[i].push_back(to);
    int ans=0;
    for(int i=1;i<=n;i++)
    {
        for(auto u:G[i])
        {
            for(auto to:gr[u])
            {
                if(r[to]>r[i])
                {
                    ans+=cnt[to];
                    cnt[to]++;
                }
            }
        }
        for(auto u:G[i])
        {
            for(auto to:gr[u])
            {
                if(r[to]>r[i]) cnt[to]--;
            }
        }
```

```
        printf("%d\n",ans);
    }
    printf("%d\n",ans);
    return 0;
}
```

## 3.26 Tarjan

```
#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;
vector<int> G[MAXN];
int n,dfn[MAXN],low[MAXN],st[MAXN];
int vis[MAXN];
int cmp[MAXN],cnt,tot,t;
void dfs(int v)
{
    dfn[v]=low[v]=++tot;
    vis[v]=1;
    st[t++]=v;
    for(auto to:G[v])
    {
        if(!vis[to])
        {
            dfs(to);
            low[v]=min(low[v],low[to]);
        }
        else if(vis[to]==1) low[v]=min(low[v],dfn[to]);
    }
    if(dfn[v]==low[v])
    {
        int u;
        do
        {
            u=st[t-1]; t--;
            cmp[u]=cnt;
            vis[u]=2;
        }while(u!=v);
        cnt++;
    }
}
int tarjan()
{
    t=tot=cnt=0;
    memset(vis,0,sizeof(vis));
    for(int i=1;i<=n;i++) if(!dfn[i]) dfs(i);
    return cnt;
}
```

## 3.27 Tree Chain Intersection

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 10005
#define MAXLOGN 14
#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 T,n,q,k,a[MAXN];
vector<int> G[MAXN];
int pa[MAXLOGN][MAXN];
int depth[MAXN];
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)];
}
bool cmp(int x,int y)
{
    return depth[x]>depth[y];
}
int main()
{
    scanf("%d",&T);
    int tot=0;
    while(T--)
    {
        scanf("%d",&n);
        for(int i=1;i<=n;i++) G[i].clear();
        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);
        }
        init(n);
        printf("Case %d:\n",++tot);
        scanf("%d",&q);
        for(int i=0;i<q;i++)
        {
            int u,v;
            scanf("%d",&k);
            scanf("%d%d",&u,&v);
            bool f=true;
            for(int j=0;j<k-1;j++)
            {
                int uu,vv;
                scanf("%d%d",&uu,&vv);
                if(!f) continue;
                int l1=lca(u,v),l2=lca(uu,vv);
                int t1=lca(l1,uu),t2=lca(l1,vv),t3=lca(l2,u),t4=lca(l2,v);
                f=false;
                if((t1==l1&&depth[t1]>=depth[l2])||(t2==l1&&depth[t2]>=depth[l2])) f=true;
                if((t3==l2&&depth[t3]>=depth[l1])||(t4==l2&&depth[t4]>=depth[l1])) f=true;
                if(!f) continue;
                int a[4];
                a[0]=lca(u,vv);a[1]=lca(u,uu);a[2]=lca(v,vv);a[3]=lca(v,uu);
                sort(a,a+4,cmp);
                u=a[0],v=a[1];
            }
            if(!f) puts("0"); else printf("%d\n",dis(u,v)+1);
        }
    }
    return 0;
}
```

## 3.28 Triangle Counting

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

```
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
int n,m,deg[MAXN],a[MAXN],cnt[MAXN],r[MAXN];
vector<int> G[MAXN];
vector<int> gr[MAXN];
bool cmp(int x,int y)
{
    return deg[x]<deg[y];
}
int main()
{
    memset(cnt,0,sizeof(cnt));
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++) a[i]=i;
    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);
        deg[v]++;deg[u]++;
    }
    sort(a,a+n+1,cmp);
    for(int i=1;i<=n;i++) r[a[i]]=i;
    for(int i=1;i<=n;i++)
        for(auto to:G[i]) if(r[to]>r[i]) gr[i].push_back(to);
    int ans=0;
    for(int i=1;i<=n;i++)
    {
        for(auto u:gr[i]) cnt[u]++;
        for(auto u:gr[i])
            for(auto to:gr[u]) ans+=cnt[to];
        for(auto u:gr[i]) cnt[u]--;
    }
    printf("%d\n",ans);
    return 0;
}
```

## 3.29 Topological Sort

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

```
                        p[to]=v;
                        dfs_visit(to);
                }
        }
        color[v]=2;
        order.push_front(v);
        f[v]=++t;
}
void toposort()
{
        t=0;
        memset(color,0,sizeof(color));
        memset(p,-1,sizeof(p));
        for(int i=0;i<V;i++)
                if(color[i]==0)
                        dfs_visit(i);
}
```

# 4 Math

## 4.1 Berlekamp-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((1ll<<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;
}
```

## 4.2 BigNum

```
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;
    }
};
```

## 4.3 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 DIVCNT1

```
#include<bits/stdc++.h>
#define MAXN 10000005
#define INF 100000000
#define MOD 100000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef __int128 lll;
struct frac
{
    ll x,y;
    frac(ll _x=0,ll _y=0) {x=_x; y=_y;}
    frac operator +(const frac &t) const
    {
        return frac(x+t.x,y+t.y);
    }
}st[MAXN],L,R,M;

int T;
ll n;
bool inR(ll x,ll y) {return x*y<=n;}
double slope(ll x) {return (double)n/x/x;}

inline void write(lll x){
    if(x>=10)write(x/10);
    putchar(x%10+'0');
}

inline void writeln(const lll &x){
    write(x);
    putchar('\n');
}

lll solve(ll n)
{
    lll ret=0;
    int t=0,rt=cbrt(n);
    st[++t]=frac(1,0);
    st[++t]=frac(1,1);
    ll m=sqrt(n),x=n/m,y=m+1;
    while(true)
    {
        for(L=st[t--];!inR(x+L.x,y-L.y);x+=L.x,y-=L.y)
            ret+=x*L.y+(L.y+1)*(L.x-1)/2;
        if(y<=rt) break;
        for(R=st[t];inR(x+R.x,y-R.y);R=st[--t]) L=R;
        while(true)
        {
            M=L+R;
```

```
            if(!inR(x+M.x,y-M.y)) st[++t]=(R=M);
            else
            {
                if(slope(x+M.x)<=(double)R.y/R.x) break;
                L=M;
            }
        }
    }
    for(int i=1;i<y;i++) ret+=n/i;
    return ret*2-1LL*m*m;
}
int main()
{
    scanf("%d",&T);
    while(T--)
    {
        scanf("%lld",&n);
        writeln(solve(n));
    }
    return 0;
}
```

## 4.6 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}^{n}i\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;
}
```

## 4.7 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;
}
```

## 4.8 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;
}
```

## 4.9 Extended Lucas Theorem

```
#include<bits/stdc++.h>
#define MAXN 1000005
#define MAXM 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;
ll n,k;
bool isprime[MAXN];
int m;
ll r[MAXM][MAXM],x[MAXM];
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;
}
int mod_inverse(int a,int m)
{
    ll x,y;
    extgcd(a,m,x,y);
```

```
    return (m+x%m)%m;
}
int solve(vector<P> &v)
{
    int n=(int)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]=(1LL*r[j][i]*(x[i]-x[j])%v[i].S+v[i].S)%v[i].S;
    }
    int base=1;
    for(int i=0;i<n;i++)
    {
        ans=(ans+1LL*base*x[i])%m;
        base=1LL*base*v[i].S%m;
    }
    return ans;
}
int pow_mod(ll a,ll i,int m)
{
    a%=m;
    int s=1;
    while(i)
    {
        if(i&1) s=1LL*s*a%m;
        a=a*a%m;
        i>>=1;
    }
    return s;
}
int get_mod(ll n,int p,int pk)
{
    if(n==0) return 1;
    int ans=1;
    for(int i=1;i<=pk;i++) if(i%p) ans=1LL*ans*i%pk;
    ans=pow_mod(ans,n/pk,pk);
    for(int i=1;i<=n%pk;i++) if(i%p) ans=1LL*ans*i%pk;
    ans=1LL*ans*get_mod(n/p,p,pk)%pk;
    return ans;
}
int comb(ll n,ll k,int p,int pk)
{
    if(n<k) return 0;
    int res1=get_mod(n,p,pk),res2=get_mod(k,p,pk),res3=get_mod(n-k,p,pk);
    int cnt=0;
    ll x=n;
    while(x) cnt+=x/p,x/=p;
    x=k;
    while(x) cnt-=x/p,x/=p;
    x=n-k;
    while(x) cnt-=x/p,x/=p;
    int ans=1LL*res1*mod_inverse(res2,pk)%pk*mod_inverse(res3,pk)%pk*pow_mod(p,cnt,pk)%pk;
    return ans;
}
vector<P> v;
```

```
int main()
{
    scanf("%lld%lld%d",&n,&k,&m);
    memset(isprime,true,sizeof(isprime));
    for(int i=2;i<=1000000;i++)
        for(int j=2*i;j<=1000000;j+=i)
            isprime[j]=false;
    int tmp=m;
    for(int i=2;i<=1000000;i++)
    {
        if(!isprime[i]) continue;
        if(tmp%i) continue;
        int p=i,pk=1;
        while(tmp%i==0) tmp/=i,pk*=i;
        v.push_back(make_pair(comb(n,k,p,pk),pk));
    }
    printf("%d\n",solve(v));
    return 0;
}
```

## 4.10 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;
}
```

## 4.11 Fast Multiplication

```
#include<bits/stdc++.h>
typedef long long ll;
ll mul(ll A,ll B,ll mod)
{
    return (A*B-(ll)((long double)A*B/mod)*mod+mod)%mod;
}
```

## 4.12 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);
    }
};
```

## 4.13 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;
}
```

## 4.14 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;
}
```

## 4.15 Polynomial Interpolation

```
#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;
}
```

## 4.16 Linear Basis

```
#include<bits/stdc++.h>
#define MAXN 1000
using namespace std;
int a[MAXN],bas[62];
int n;
int main()
{
    for(int i=1;i<=n;i++)
    {
        int x=a[i];
        for(int j=60;j>=0;j--)
        {
            if(x&(1ll<<j))
            {
                if(!bas[j])
                {
                    bas[j]=x;
                    break;
                }
                x^=bas[j];
            }
        }
    }
}
```

## 4.17 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);
}
```

## 4.18 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");
    }
}
```

## 4.19 Matrix Opertions

```
#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");
    }
}
```

## 4.20 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;
}
```

## 4.21 Mod-Combination 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()
{
    inv[1] = 1;
        for (int i = 2; i < MOD; i++)
                inv[i] = (MOD - MOD / i) * inv[ MOD % i] % MOD;
    printf("%d\n",mod_inverse(22,31));
    return 0;
}
```

## 4.22 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;
}
```

## 4.23 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;
}
```

## 4.24 Pohlig-Hellman

```
#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;
int T;
ll a,b,p;
ll mul(ll A,ll B,ll mod)
{
    return (A*B-(ll)((long double)A*B/mod)*mod+mod)%mod;
}
ll pow_mod(ll a,ll i,ll mod)
{
    if(i==0) return 1;
     ll s=1;
    while(i>0)
```

```
    {
        if(i&1) s=mul(s,a);
        a=mul(a,a);
        i>>=1;
    }
    return s;
}

vector<P> fact;

ll Pohlig_Hellman(ll g,ll x,ll p,vector<P> fact)
{
    ll q=p-1,now=g,res=1,ker=0;
    for(int i=0;i<(int)fact.size();i++)
    {
        for(int j=0;j<fact[i].S;j++)
        {
            q/=fact[i].F;
            //find dlg modulo current prime, code below is for the prime 2
            if(pow_mod(x,q,p)!=1)
            {
                x=mul(x,now,p);
                res+=ker;
            }
            now=pow_mod(now,fact[i].F,p);
            ker*=fact[i].F;
        }
    }
    return (p-1-res)%(p-1);
}
```

## 4.25 Points In a Circle

```
#include <bits/stdc++.h>

using namespace std;
using i64 = int64_t;
int T;

i64 solve_fast(i64 N) {
  auto inside = [N] (i64 x, i64 y) {
    return x * x + y * y <= N;
  };
  auto cut = [] (i64 x, i64 y, int dx1, int dy1) {
    return dx1 * x >= dy1 * y;
  };

  const i64 v = sqrtl(N / 2), w = sqrtl(N);
  i64 x = v;
  i64 y = i64(sqrtl(max<i64>(0, N - (v + 1) * (v + 1)))) + 1;

  auto stac = stack< pair<int, int> >({{0, 1}, {1, 1}});

  i64 ret = 0;
  while (1) {
    int dx1, dy1; tie(dx1, dy1) = stac.top(); stac.pop();
    while (inside(x + dx1, y - dy1))
```

```
    {
        x += dx1; y -= dy1;
        ret += i64(dx1) * (y - 1)
          + ((i64(dx1 + 1) * (dy1 + 1)) >> 1) - dy1;
    }

    int dx2 = dx1, dy2 = dy1;
    while (!stac.empty()) {
      tie(dx1, dy1) = stac.top();
      if (inside(x + dx1, y - dy1)) break;
      stac.pop();
      dx2 = dx1, dy2 = dy1;
    }
    if (stac.empty()) break;

    while (1) {
      int dx12 = dx1 + dx2, dy12 = dy1 + dy2;
      if (inside(x + dx12, y - dy12)) {
        stac.emplace(dx1 = dx12, dy1 = dy12);
      } else {
        if (cut(x + dx12, y - dy12, dx1, dy1)) break;
        dx2 = dx12, dy2 = dy12;
      }
    }
  }
  ret = ret * 2 + i64(v) * v;
  ret = ret * 4 + 4 * i64(w) + 1;
  return ret;
}

int main()
{
    i64 N = 1e18;
    // printf("%llu\n", solve_naive(N));
    printf("%llu\n", solve_fast(N));
    return 0;
}
```

## 4.26 Pollard Rho

```
#include<bits/stdc++.h>
#define MAXN 1005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef unsigned long long ULL;
//to achieve best running time, sieve until 2/3 prime factors remaining
namespace pollardrho
{
    ULL gcd(ULL a, ULL b) {return b ? gcd(b, a % b) : a;}

    ULL mulmod(ULL x,ULL y,ULL p)
    {
        ULL z=(long double)x/p*y;
        ULL res=(ULL)x*y-(ULL)z*p;
```

```
    return (res+p)%p;
}

ULL powmod(ULL b, ULL e, ULL m)
{
    ULL r = 1;
    while (e)
    {
        if (e & 1) r = mulmod(r, b, m);
        b = mulmod(b, b, m);
        e >>= 1;
    }
    return r;
}

bool test(ULL n)
{
    if (n < 3) return n==2;
    // ! The array a[] should be modified if the range of x changes.
    static const ULL a[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, ULLONG_MAX};//works for 1e18
    ULL r = 0, d = n-1, x;
    while (~d & 1) d >>= 1, r++;
    for (int i=0; a[i] < n; i++)
    {
        x = powmod(a[i], d, n);
        if (x == 1 || x == n-1) goto next;
        for(int i=0;i<r;i++)
        {
            x = mulmod(x, x, n);
            if (x == n-1) goto next;
        }
        return false;
        next:;
    }
    return true;
}

map<ULL, int> mp;
mt19937_64 gen(time(NULL));

void PollardRho(ULL n)
{
    ULL c, x, y, d;
    while (n % 2 == 0)
    {
        mp[2]++;
        n /= 2;
    }
    if (n == 1) return;

    if (test(n))
    {
        mp[n]++;
        return;
    }

    d = n;
    static int counter = 0;
    while (d == n)
```

```
        {
            x = y = 2;
            d = 1;
            c = gen() % (n - 1) + 1;
            while (d == 1)
            {
                counter++;
                x = (mulmod(x, x, n) + c) % n;
                y = (mulmod(y, y, n) + c) % n;
                y = (mulmod(y, y, n) + c) % n;
                d = gcd(x > y ? x - y : y - x, n);
            }
        }
        PollardRho(d);
        PollardRho(n / d);
    }

    void work(ULL n,int id)
    {
        PollardRho(n);
        for(auto p:mp) fact[id].push_back(p);
        mp.clear();
    }
}
```

## 4.27 Polynomial Operations

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 998244353
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
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 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;
}
ll 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);
    ll 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(ll 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=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);
    }
};
namespace poly
{
    int inv(int x) {return pow_mod(x,MOD-2,MOD);}
    vector<int> fa,fb,fc,fd,fe,ff,fg,Inv;
    void get_inv(vector<int> &a,int n,vector<int> &ret)
    {
        assert(a[0]!=0);
        if(n==1)
        {
            ret.resize(1);
            ret[0]=inv(a[0]);
```

```
        return;
    }
    get_inv(a,(n+1)>>1,ret);
    fa=a; fb=ret;
    fa=fft::multiply_mod(fb,fb,MOD,1);
    fa=fft::multiply_mod(fa,a,MOD);
    fa.resize(n); fb.resize(n); ret.resize(n);
    for(int i=0;i<n;i++)
    {
        ret[i]=inc(fb[i],fb[i]);
        ret[i]=dec(ret[i],fa[i]);
    }
    fa.clear(); fb.clear();
}
void get_sqrt(vector<int> &a,int n,vector<int> &ret)
{
    if(n==1)
    {
        ret.resize(1);
        int x=Tonelli_Shanks(a[0],MOD);
        assert(x!=-1);
        ret[0]=x;
        return;
    }
    get_sqrt(a,(n+1)>>1,ret);
    get_inv(ret,n,fc);
    ret=fft::multiply_mod(ret,ret,MOD,1);
    ret.resize(n);
    for(int i=0;i<n;i++) fc[i]=1LL*fc[i]*((MOD+1)/2)%MOD;
    for(int i=0;i<n;i++) ret[i]=inc(ret[i],a[i]);
    ret=fft::multiply_mod(ret,fc,MOD);
    ret.resize(n);
}
void diff(vector<int> &a,int n,vector<int> &ret)
{
    ret.resize(n);
    for(int i=1;i<n;i++) ret[i-1]=1LL*a[i]*i%MOD;
    ret[n-1]=0;
}
void intg(vector<int> &a,int n,vector<int> &ret)
{
    ret.resize(n); Inv.resize(n);
    if(n>1) Inv[1]=1;
    for(int i=2;i<=n-1;i++) Inv[i]=dec(MOD,1LL*Inv[MOD%i]*(MOD/i)%MOD);
    for(int i=n-1;i>=1;i--) ret[i]=1LL*a[i-1]*Inv[i]%MOD;
    ret[0]=0;
}
void get_ln(vector<int> &a,int n,vector<int> &ret)
{
    assert(a[0]==1);
    diff(a,n,fc);
    get_inv(a,n,fd);
    fc=fft::multiply_mod(fc,fd,MOD);
    intg(fc,n,ret);
    ret.resize(n);
    fc.clear(); fd.clear();
}
void get_exp(vector<int> &a,int n,vector<int> &ret)
{
```

```
        if(n==1)
        {
            ret.resize(1); ret[0]=1;
            return;
        }
        get_exp(a,(n+1)>>1,ret); ret.resize(n);
        get_ln(ret,n,ff);
        for(int i=0;i<n;i++) ff[i]=dec(MOD,ff[i]);
        ff[0]+=1; if(ff[0]>=MOD) ff[0]-=MOD;
        for(int i=0;i<n;i++) ff[i]=inc(ff[i],a[i]);
        ret=fft::multiply_mod(ret,ff,MOD); ret.resize(n);
        ff.clear();
    }
    void division(vector<int> &a,vector<int> &b,vector<int> &q,vector<int> &r)
    {
        int n=(int)a.size(),m=(int)b.size();
        if(n<m) {q.resize(1); q[0]=0; r=a; return;}
        vector<int> tmp=b; reverse(tmp.begin(),tmp.end());
        get_inv(tmp,n-m+1,tmp);
        vector<int> rev=a; reverse(rev.begin(),rev.end());
        q=fft::multiply_mod(tmp,rev,MOD); q.resize(n-m+1);
        reverse(q.begin(),q.end());
        vector<int> t=fft::multiply_mod(b,q,MOD);
        r.resize(m-1);
        for(int i=0;i<m-1;i++) r[i]=dec(a[i],t[i]);
    }
}
int n,m,k;
vector<int> f,g,a,b;
int main()
{
    scanf("%d",&n);
    f.resize(n);
    for(int i=0;i<n;i++) scanf("%d",&f[i]);
    vector<int> expf;
    poly::get_exp(f,n,expf);
    for(int i=0;i<n;i++) printf("%d ",expf[i]);
    return 0;
}
```

## 4.28 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;
}
```

## 4.29 Power Tower

```
#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,q,a[MAXN];
unordered_map<int,int> phi;
int f(ll x,int m) {return x<m?x:x%m+m;}
int pow_mod(int a,int i,int m)
{
    int s=1%m;
    while(i)
    {
        if(i&1) s=f(1LL*s*a,m);
        a=f(1LL*a*a,m);
        i>>=1;
    }
    return s;
}
int getphi(int n)
{
    if(phi.find(n)!=phi.end()) return phi[n];
    int res=n;
    for(int i=2;i*i<=n;i++)
    {
        if(n%i==0)
```

```
        {
            res=res/i*(i-1);
            for(;n%i==0;n/=i);
        }
    }
    if(n!=1) res=res/n*(n-1);
    return phi[n]=res;
}
//calculate power tower in [l,r] modulo m
//need to modulo m outside the recursion
int solve(int l,int r,int m)
{
    if(m==1||l>r||a[l]==1) return 1;
    return pow_mod(f(a[l],m),solve(l+1,r,getphi(m)),m);
}
```

## 4.30 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;
}
```

## 4.31 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;
}
```

## 4.32 Schreier-Sims

```
#include <cstdio>
#include <vector>
#include <cassert>

using namespace std;

typedef vector<int> perm;
typedef long long llong;

perm operator *(const perm& a, const perm& b) {
    assert(a.size() == b.size());
    perm c(a.size());
    for (int i = 0; i < a.size(); i++) {
        c[i] = a[b[i]];
    }
    return c;
}

perm inv(const perm& a) {
    perm c(a.size());
    for (int i = 0; i < a.size(); i++)
        c[a[i]] = i;
    return c;
}

perm identity(int n) {
    perm c(n);
    for (int i = 0; i < n; i++)
        c[i] = i;
    return c;
}

void DFS(const perm& cur, const vector<perm>& generators, vector<perm>& sigma) {
    sigma[cur[0]] = cur;
    for (const perm& g : generators) {
        perm y = g * cur;
        if (sigma[y[0]].empty()) {
            DFS(y, generators, sigma);
        }
    }
}

void reduceGenerators(vector<perm>& generators) {
    if (generators.empty())
        return;
    int n = generators.front().size();
    int pt = 0;
```

```
    for (int i = 0; i < n; i++) {
        vector<int> posByFirst(n, -1);
        for (int j = pt; j < generators.size(); j++) {
            perm& g = generators[j];
            assert(g[i] >= i);
            if (g[i] == i)
                continue;
            else if (posByFirst[g[i]] == -1) {
                posByFirst[g[i]] = pt;
                g.swap(generators[pt]);
                pt++;
            } else {
                g = inv(generators[posByFirst[g[i]]]) * g;
            }
        }
    }
    assert(pt <= n * (n - 1) / 2);
    generators.resize(pt);
}

llong calc(vector<perm> generators) {
    if (generators.empty())
        return 1ll;
    int n = generators.front().size();
    if (n == 0)
        return 1ll;

    vector<perm> sigma(n, perm());
    DFS(identity(n), generators, sigma);
    vector<perm> invSigma(n, perm());
    for (int i = 0; i < n; i++)
        invSigma[i] = inv(sigma[i]);

    int nSigma = 0;

    vector<perm> newGenerators;
    for (int i = 0; i < n; i++) {
        if (sigma[i].empty())
            continue;
        nSigma++;
        for (const perm& g : generators) {
            perm x = g * sigma[i];
            assert(!invSigma[x[0]].empty());
            newGenerators.emplace_back(invSigma[x[0]] * x);
        }
    }

    reduceGenerators(newGenerators);
    for (perm& g : newGenerators) {
        assert(g[0] == 0);
        g.erase(g.begin() + 0);
        for (int& x : g)
            --x;
    }

    return nSigma * calc(newGenerators);
}
```

```
int main() {
    int k, n;
    scanf("%d %d", &k, &n);
    vector<perm> generators;
    for (int i = 0; i < k; i++) {
        perm x(n);
        for (int j = 0; j < n; j++)
            scanf("%d", &x[j]), --x[j];
        generators.emplace_back(x);
    }
    llong ans = calc(generators);
    printf("%lld\n", ans);
    return 0;
}
```

## 4.33 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);
}
```

## 4.34 Simpson Method

```
#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));
}
```

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

```
    return 0;
}
```

## 4.36 Stirling number of the second kind(multiple)

```
#include<bits/stdc++.h>
#define MAXN 100005
#define MOD 998244353
#define INF 1000000000
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int g=3;
int two[32];
int dbit(int x)
{
    while((x&-x)!=x) x+=x&-x;
    return x;
}
int pow_mod(int a,int i)
{
    if(i==0) return 1;
    int s=1;
    while(i>0)
     {
         if(i&1) s=(1LL*s*a)%MOD;
         a=(1LL*a*a)%MOD;
         i>>=1;
     }
     return s;
}
int rev(int x,int r)
{
    int ans=0;
    for(int i=0;i<r;i++)
        if(x&(1<<i)) ans+=1<<(r-i-1);
    return ans;
}
void ntt(int n,int A[],int on)
{
    int r=0,cnt=0,t=n;
    while(t>1) {cnt++; t/=2;}
    for(;;r++) if((1<<r)==n) break;
    for(int i=0;i<n;i++)
    {
        int tmp=rev(i,r);
        if(i<tmp) swap(A[i],A[tmp]);
    }
    for(int s=1;s<=r;s++)
    {
        int m=1<<s;
        int wn=pow_mod(g,(MOD-1)/m);
        for(int k=0;k<n;k+=m)
        {
            int w=1;
```

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

## 4.37 Subset Convolution

```
#include<bits/stdc++.h>
#define MAXN 2000005
```

```
#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 f[MAXN],g[MAXN];
int fhat[MAXLOGN][MAXN],ghat[MAXLOGN][MAXN],h[MAXLOGN][MAXN];
int fog[MAXN];
int n;
void add(int &a,int b) {a+=b; if(a>=MOD) a-=MOD;}
void dec(int &a,int b) {a-=b; if(a<0) a+=MOD;}
void subset_convolution()
{

    for(int mask=0;mask<(1<<n);mask++)
    {
        fhat[__builtin_popcount(mask)][mask]=f[mask];
        ghat[__builtin_popcount(mask)][mask]=g[mask];
    }
    for(int i=0;i<=n;i++)
    {
        for(int j=0;j<n;j++)
        {
            for(int mask=0;mask<(1<<n);mask++)
            {
                if((mask&(1<<j))!=0)
                {
                    add(fhat[i][mask],fhat[i][mask^(1<<j)]);
                    add(ghat[i][mask],ghat[i][mask^(1<<j)]);
                }
            }
        }
    }
    for(int mask=0;mask<(1<<n);mask++)
        for(int i=0;i<=n;i++)
            for(int j=0;j<=i;j++)
                add(h[i][mask],1LL*fhat[j][mask]*ghat[i-j][mask]%MOD);
    for(int i=0;i<=n;i++)
        for(int j=0;j<n;j++)
            for(int mask=0;mask<(1<<n);mask++)
                if((mask&(1<<j))!=0)
                    dec(h[i][mask],h[i][mask^(1<<j)]);
    for(int mask=0;mask<(1<<n);mask++) fog[mask]=h[__builtin_popcount(mask)][mask];
}
int main()
{
    scanf("%d",&n);
    for(int i=0;i<(1<<n);i++) scanf("%d",&f[i]);
    for(int i=0;i<(1<<n);i++) scanf("%d",&g[i]);
    subset_convolution();
    for(int i=0;i<(1<<n);i++) printf("%d ",fog[i]);
    puts("");
    return 0;
}
```

## 4.38 Prefix Sum of Miu

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

## 4.39 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;
}
```

## 4.40 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;
}
```

# 5 Others

## 5.1 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<ll,ll> P;
ll N,tot,t,now;
P st[MAXN];
void add(ll u,ll v)
{
    P p=P(u,v);
    while(t-now>1&&(st[t-1].F-st[t-2].F)*(p.S-st[t-1].S)<=(st[t-1].F-p.F)*(st[t-2].S-st[t-1].S))
        t--;
    st[t++]=p;
}
bool cmp(P x,P y)
{
    if(x.S!=y.S) return x.S<y.S;
    return x.F<y.F;
}
ll query(ll x)
{
    ll l=-1,r=t-1;
    while(r-l>1)
    {
        ll mid=(l+r)/2;
        if(st[mid].F*x+st[mid].S<=st[mid+1].F*x+st[mid+1].S) l=mid;
        else r=mid;
    }
    return st[r].F*x+st[r].S;
}
int main()
{
}
```

## 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 Dynamic Dynamic Programming

```
//luogu 4719 dynamic maximum weight vertex cover
#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 tot,n,q;
int dp[MAXN][2];
int ldp[MAXN][2];
int
    pa[MAXN],a[MAXN],dep[MAXN],sz[MAXN],wson[MAXN],top[MAXN],st[MAXN],ed[MAXN],tpos[MAXN],w[MAXN],bot[MAXN];
struct mat
{
    int v[2][2];
    mat(){v[0][0]=v[0][1]=v[1][0]=v[1][1]=-INF;}
};
mat mul(mat a,mat b)
{
    mat c;
    for(int i=0;i<2;i++)
        for(int j=0;j<2;j++)
            for(int k=0;k<2;k++)
                c.v[i][j]=max(c.v[i][j],a.v[i][k]+b.v[k][j]);
    return c;
}
mat unit;
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; bot[num]=v;
    st[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);
    }
    ed[v]=tot;
}
struct segtree
{
    mat val[4*MAXN];
    void pushup(int k)
    {
        val[k]=mul(val[k*2],val[k*2+1]);
    }
    void build(int k,int l,int r)
    {
        if(l==r)
        {
            int v=tpos[l];
            val[k].v[0][0]=val[k].v[0][1]=ldp[v][0];
            val[k].v[1][0]=ldp[v][1];
            val[k].v[1][1]=-INF;
            return;
        }
        int mid=(l+r)/2;
        build(k*2,l,mid); build(k*2+1,mid+1,r);
        pushup(k);
    }
    void update(int k,int l,int r,int p,int v1,int v2)
    {
        if(l==r)
        {
            val[k].v[0][0]=val[k].v[0][1]=v1;
            val[k].v[1][0]=v2;
            val[k].v[1][1]=-INF;
            return;
        }
        int mid=(l+r)/2;
        if(p<=mid) update(k*2,l,mid,p,v1,v2);
        else update(k*2+1,mid+1,r,p,v1,v2);
        pushup(k);
    }
    mat query(int k,int l,int r,int x,int y)
    {
        if(x>r||l>y) return unit;
        if(l>=x&&r<=y) return val[k];
        int mid=(l+r)/2;
        return mul(query(k*2,l,mid,x,y),query(k*2+1,mid+1,r,x,y));
    }
}tree;
void init()
{
    tot=0;
    memset(wson,0,sizeof(wson));//important when multiple test cases!!!
    dfs1(1,0,1);
    dfs2(1,0,1);
    tree.build(1,1,n);
}
```

```
void update(int v,int x)
{
    ldp[v][1]+=(x-w[v]); w[v]=x;
    while(v!=0)
    {
        int l=st[top[v]],r=st[bot[top[v]]];
        //mat tmp1(2,vec(1)),tmp2(2,vec(1));
        //tmp1[0][0]=tmp1[1][0]=tmp2[0][0]=tmp2[1][0]=0;
        mat past=tree.query(1,1,n,l,r);
        tree.update(1,1,n,st[v],ldp[v][0],ldp[v][1]);
        mat now=tree.query(1,1,n,l,r);
        v=pa[top[v]];
        ldp[v][0]+=max(now.v[0][0],now.v[1][0])-max(past.v[0][0],past.v[1][0]);
        ldp[v][1]+=now.v[0][0]-past.v[0][0];
    }
}
int main()
{
    unit.v[0][0]=unit.v[1][1]=0; unit.v[0][1]=unit.v[1][0]=-INF;
    scanf("%d%d",&n,&q);
    for(int i=1;i<=n;i++) scanf("%d",&a[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);
    }
    init();
    for(int i=1;i<=n;i++) update(i,a[i]);
    for(int i=0;i<q;i++)
    {
        int x,y;
        scanf("%d%d",&x,&y);
        update(x,y);
        int l=1,r=st[bot[1]];
        mat A=tree.query(1,1,n,l,r);
        printf("%d\n",max(A.v[0][0],A.v[1][0]));
    }
    return 0;
}
```

## 5.4 Knuth's Optimization

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

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

## 5.5 Matroid Intersection

```
#include<bits/stdc++.h>
#define MAXN 65
#define MAXM 6005
#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 color[MAXM];
ll val[MAXM];
int n,m,tot,tot2;
struct LinearMatroid
{
    ll basis[62];
    void clear()
    {
        memset(basis,0,sizeof(basis));
    }
    void add(ll x)
    {
        for(int j=60;j>=0;j--)
        {
            if(!(x&(1LL<<j))) continue;
            if(!basis[j])
            {
```

```
                basis[j]=x;
                return;
            }
            else x^=basis[j];
        }
    }
    bool test(ll x)
    {
        for(int j=60;j>=0;j--)
        {
            if(!(x&(1LL<<j))) continue;
            if(!basis[j]) return true; else x^=basis[j];
        }
        return false;
    }
};
struct ColorfulMatroid
{
    int cnt[125];
    void clear()
    {
        memset(cnt,0,sizeof(cnt));
    }
    void add(int x)
    {
        cnt[x]++;
    }
    bool test(int x)
    {
        return (cnt[x]==0);
    }
};


template <typename MT1, typename MT2>
struct MatroidIntersection
{
    int n;
    MatroidIntersection(int _n):n(_n){}
    int pre[MAXM],id[MAXM];
    bool vis[MAXM],sink[MAXM],has[MAXM];
    queue<int> que;
    void clear_all()
    {
        memset(vis,false,sizeof(vis));
        memset(sink,false,sizeof(sink));
        memset(pre,0,sizeof(pre));
        while(que.size()) que.pop();
    }
    vector<int> getcur()
    {
        vector<int> ret;
        for(int i=1;i<=n;i++) if(has[i]) ret.push_back(i);
        return ret;
    }
    void enqueue(int v,int p)
    {
        vis[v]=true; pre[v]=p;
        que.push(v);
```

```
}
vector<int> run()
{
    MT1 mt1; MT2 mt2;
    memset(has,false,sizeof(has));
    while(true)
    {
        vector<int> cur=getcur();
        int cnt=0;
        for(int i=1;i<=n;i++) if(has[i]) id[i]=cnt++;
        MT1 allmt1; MT2 allmt2; allmt1.clear(); allmt2.clear();
        vector<MT1> vmt1(cur.size()); vector<MT2> vmt2(cur.size());
        for(auto &x:vmt1) x.clear(); for(auto &x:vmt2) x.clear();
        clear_all();
        for(auto x:cur) allmt1.add(val[x]),allmt2.add(color[x]);
        for(int i=0;i<(int)cur.size();i++)
            for(int j=0;j<(int)cur.size();j++)
            {
                if(i==j) continue;
                vmt1[i].add(val[cur[j]]);
                vmt2[i].add(color[cur[j]]);
            }
        for(int i=1;i<=n;i++)
        {
            if(has[i]) continue;
            if(allmt1.test(val[i])) {que.push(i); vis[i]=true;}
        }
        for(int i=1;i<=n;i++)
        {
            if(has[i]) continue;
            if(allmt2.test(color[i])) sink[i]=true;
        }
        int last=-1;
        while(que.size())
        {
            int v=que.front(); que.pop();
            if(sink[v]) {last=v; break;}
            for(int i=1;i<=n;i++)
            {
                if(vis[i]) continue;
                if(has[i]==has[v]) continue;
                if(has[v])
                {
                    if(vmt1[id[v]].test(val[i])) enqueue(i,v);
                }
                else
                {
                    if(vmt2[id[i]].test(color[v])) enqueue(i,v);
                }
            }
        }
        if(last==-1) return cur;
        while(last)
        {
            has[last]^=1;
            last=pre[last];
        }
    }
}
```

```
};
//Pick Your Own Nim
//In real cases, Linear Matroid Need Optimization to Pass
int main()
{
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        ll x;
        scanf("%lld",&x);
        val[++tot]=x; color[tot]=++tot2;
    }
    scanf("%d",&m);
    for(int i=0;i<m;i++)
    {
        int k;
        scanf("%d",&k);
        tot2++;
        for(int j=0;j<k;j++)
        {
            ll x;
            scanf("%lld",&x);
            val[++tot]=x; color[tot]=tot2;
        }
    }
    MatroidIntersection<LinearMatroid,ColorfulMatroid> matint(tot);
    vector<int> res=matint.run();
    if(res.size()<n+m) {puts("-1"); return 0;}
    else
    {
        vector<ll> ans;
        int last=n;
        for(auto x:res)
        {
            if(color[x]>last)
            {
                ans.push_back(val[x]);
                last=color[x];
            }
        }
        for(auto x:ans) printf("%lld\n",x);
    }
    return 0;
}
```

## 5.6 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]);
}
```

## 5.7 Nim Multiplication

```
#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,sg[2][2]={0,0,0,1};
int nim_mult_pow(int x,int y)
{
    if(x<2)
        return sg[x][y];
    int a=0;
    for(;;a++)
        if(x>=(1<<(1<<a))&&x<(1<<(1<<(a+1))))
            break;
    int m=1<<(1<<a);
    int p=x/m,s=y/m,t=y%m;
    int d1=nim_mult_pow(p,s);
    int d2=nim_mult_pow(p,t);
    return (m*(d1^d2))^nim_mult_pow(m/2,d1);
}

int nim_mult(int x,int y)
{
    if(x<y)
        return nim_mult(y,x);
    if(x<2)
        return sg[x][y];
    int a=0;
    for(;;a++)
        if(x>=(1<<(1<<a))&&x<(1<<(1<<(a+1))))
            break;
    int  m=1<<(1<<a);
    int p=x/m,q=x%m,s=y/m,t=y%m;
    int c1=nim_mult(p,s);
```

```
    int c2=nim_mult(p,t)^nim_mult(q,s);
    int c3=nim_mult(q,t);
    return (m*(c1^c2))^c3^nim_mult_pow(m/2,c1);
}
int main()
{
    int x,y;
    while(scanf("%d%d",&x,&y)==2) printf("%d\n",nim_mult(x,y));
}
```

## 5.8 Simplex Method

```
#include<bits/stdc++.h>
#define MAXN 105
#define MAXM 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;
typedef double db;
typedef vector<db> vec;
typedef vector<vec> mat;
//usage:
//minimize cx, where x is a vector of length n, with m constraints
//feed in m+1 lines
//first line contains (0,c)
//each of the next lines describe an inequality ax<=b in the form of (b,a)
const db eps=1e-8;
bool eq(db a,db b)
{
    return fabs(a-b)<eps;
}
bool ls(db a,db b)
{
    return a<b&&!eq(a,b);
}
vec simplex(mat a)
{
    int n=(int)a.size()-1;
    int m=(int)a[0].size()-1;
    vec left(n+1),up(m+1);
    iota(up.begin(),up.end(),0);
    iota(left.begin(),left.end(),m);
    auto pivot=[&](int x,int y)
    {
        swap(left[x],up[y]);
        db k=a[x][y];
        a[x][y]=1;
        vector<int> vct;
        for(int j=0;j<=m;j++)
        {
            a[x][j]/=k;
            if(!eq(a[x][j],0)) vct.push_back(j);
        }
```

```
        for(int i=0;i<=n;i++)
        {
            if(eq(a[i][y],0)||i==x) continue;
            k=a[i][y];
            a[i][y]=0;
            for(int j:vct) a[i][j]-=k*a[x][j];
        }
    };
    while(1)
    {
        int x=-1;
        for(int i=1;i<=n;i++) if(ls(a[i][0],0)&&(x==-1||a[i][0]<a[x][0])) x=i;
        if(x==-1) break;
        int y=-1;
        for(int j=1;j<=m;j++) if(ls(a[x][j],0)&&(y==-1||a[x][j]<a[x][y])) y=j;
        assert(y!=-1);
        pivot(x,y);
    }
    while(1)
    {
        int y=-1;
        for(int j=1;j<m;j++) if(ls(0,a[0][j])&&(y==-1||a[0][j]>a[0][y])) y=j;
        if(y==-1) break;
        int x=-1;
        for(int i=1;i<=n;i++) if(ls(0,a[i][y])&&(x==-1||a[i][0]/a[i][y]<a[x][0]/a[x][y])) x=i;
        assert(x!=-1);
        pivot(x,y);
    }
    vector<double> ans(m+1);
    for(int i=1;i<=n;i++) if(left[i]<=m) ans[left[i]]=a[i][0];
    ans[0]=-a[0][0];
    return ans;
}
```

## 5.9 Sum Over Subset Dynamic Programming

```
#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;
}
```

## 5.10 Subset Choosing(Gosper's Hack)

```
#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;
}
```

## 5.11 Surreal Number

```
double getans(double l,double r)
{
    assert(l<r);
    assert(!(l==0&&r==0));
    if(l==-INF&&r==INF) return 0;
    if(l==-INF) return r-1;
    if(r==INF) return l+1;
    if(l<0&&r>0) return 0.0;
    if(l<0&&r<=0) return -getans(-r,-l);
```

```
    double res=1.0;
    while(true)
    {
        double t=0.0;
        while(t<=l) t+=res;
        if(t<r) return t;
        res/=2.0;
    }
    assert(0);
}
```

## 5.12 Weighted Matroid Intersection

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 85
#define MAXM 205
#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 c[MAXN],k[MAXN],color[MAXM],u[MAXM],v[MAXM],w[MAXM],cost[MAXM];
ll val[MAXM];
int T,n,m,tot,tot2;
struct LinearMatroid
{
    ll basis[62];
    void clear()
    {
        memset(basis,0,sizeof(basis));
    }
    void add(ll x)
    {
        for(int j=60;j>=0;j--)
        {
            if(!(x&(1LL<<j))) continue;
            if(!basis[j])
            {
                basis[j]=x;
                return;
            }
            else x^=basis[j];
        }
    }
    bool test(ll x)
    {
        for(int j=60;j>=0;j--)
        {
            if(!(x&(1LL<<j))) continue;
            if(!basis[j]) return true; else x^=basis[j];
        }
        return false;
    }
};
```

```
struct ColorfulMatroid
{
    int cnt[125];
    void clear()
    {
        memset(cnt,0,sizeof(cnt));
    }
    void add(int x)
    {
        cnt[x]++;
    }
    bool test(int x)
    {
        return (cnt[x]==0);
    }
};

struct GraphMatroid
{
    vector<int> G[MAXN];
    bool vis[MAXN];
    bool exist[MAXN];
    void dfs(int v)
    {
        vis[v]=true;
        for(auto to:G[v]) if(!vis[to]) dfs(to);
    }
    bool test(vector<int> &vec)
    {
        for(int i=1;i<=n+1;i++) G[i].clear();
        memset(vis,false,sizeof(vis));
        memset(exist,true,sizeof(exist));
        for(auto x:vec) exist[x]=false;
        for(int i=1;i<=tot;i++)
        {
            if(exist[i])
            {
                G[u[i]].push_back(v[i]);
                G[v[i]].push_back(u[i]);
            }
        }
        dfs(1);
        for(int i=1;i<=n+1;i++) if(!vis[i]) return false;
        return true;
    }
};

struct PartitionMatroid
{
    int cnt[125];
    bool test(vector<int> &vec)
    {
        memset(cnt,0,sizeof(cnt));
        for(auto x:vec) cnt[color[x]]++;
        for(int i=1;i<=m;i++) if(cnt[i]>c[i]-k[i]) return false;
        return true;
    }
};
```

```
template <typename MT1, typename MT2>
struct MatroidIntersection
{
    int n,S,T;
    MatroidIntersection(int _n):n(_n){}
    int pre[MAXM],id[MAXM],d[MAXM];
    bool inque[MAXM],sink[MAXM],has[MAXM];
    vector<int> g[MAXN];
    queue<int> que;
    void clear_all()
    {
        for(int i=1;i<=n+2;i++)
        {
            inque[i]=false;
            sink[i]=false;
            pre[i]=0;
            d[i]=-INF;
            if(has[i]) cost[i]=w[i]; else cost[i]=-w[i];
            g[i].clear();
        }
        while(que.size()) que.pop();
    }
    void add_edge(int u,int v)
    {
        g[u].push_back(v);
    }
    vector<int> getcur()
    {
        vector<int> ret;
        for(int i=1;i<=n;i++) if(has[i]) ret.push_back(i);
        return ret;
    }
    void enqueue(int v,int p)
    {
        pre[v]=p;
        if(!inque[v])
        {
            inque[v]=true;
            que.push(v);
        }
    }
    pair<vector<int>,ll> run()
    {
        ll ans=0;
        MT1 mt1; MT2 mt2;
        memset(has,false,sizeof(has));
        S=n+1; T=n+2;
        while(true)
        {
            clear_all();
            for(int i=1;i<=n;i++)
            {
                if(!has[i])
                {
                    cost[i]=w[i];
                    has[i]^=1;
                    vector<int> tmp=getcur();
                    if(mt1.test(tmp)) add_edge(S,i);
```

```
                if(mt2.test(tmp)) add_edge(i,T);
                has[i]^=1;
            }
            else cost[i]=-w[i];
        }
        for(int i=1;i<=n;i++)
        {
            if(!has[i])
            {
                for(int j=1;j<=n;j++)
                {
                    if(has[j])
                    {
                        has[i]^=1; has[j]^=1;
                        vector<int> tmp=getcur();
                        if(mt1.test(tmp)) add_edge(j,i);
                        if(mt2.test(tmp)) add_edge(i,j);
                        has[i]^=1; has[j]^=1;
                    }
                }
            }
        }
        d[S]=0; que.push(S); inque[S]=true;
        cost[S]=cost[T]=0;
        int counter=0;
        while(que.size())
        {
            counter++;
            int u=que.front(); que.pop();
            for(auto to:g[u])
                if(d[to]<d[u]+cost[to])
                {
                    d[to]=d[u]+cost[to];
                    enqueue(to,u);
                }
            inque[u]=false;
        }
        if(!pre[T]) return make_pair(getcur(),ans);
        ans+=d[T];
        int last=pre[T];
        while(last!=S)
        {
            has[last]^=1;
            last=pre[last];
        }
    }
}
};
//hdu 6636 Milk Candy
int main()
{
    scanf("%d",&T);
    while(T--)
    {
        tot=0;
        scanf("%d%d",&n,&m);
        int sum=0;
        ll ans=0;
        for(int i=1;i<=m;i++)
```

```
        {
            scanf("%d%d",&c[i],&k[i]);
            sum+=c[i]-k[i];
            for(int j=1;j<=c[i];j++)
            {
                int l,r,cost;
                scanf("%d%d%d",&l,&r,&cost);
                color[++tot]=i; u[tot]=l; v[tot]=r+1; w[tot]=cost;
                ans+=cost;
            }
        }
        MatroidIntersection<GraphMatroid,PartitionMatroid> matint(tot);
        auto res=matint.run();
        GraphMatroid gm; PartitionMatroid pm;
        if((int)res.F.size()!=sum||!gm.test(res.F)||!pm.test(res.F)) puts("-1"); else
            printf("%lld\n",ans-res.S);
    }
    return 0;
}
```

### 5.13 Zeller's Formula

```
#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 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;
}
```

## 6.2 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 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]);
}
```

## 6.4 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;
}
```

## 6.5 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;
}
```

## 6.6 Suffix Automaton

```
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
struct SuffixAutomaton
{
    vector<map<char,int>> edges;
    vector<int> link;
    vector<int> length;
    int last;
    SuffixAutomaton(string s)
    {
        edges.push_back(map<char,int>());
        link.push_back(-1);
        length.push_back(0);
        last=0;
        for(int i=0;i<s.size();i++)
```

```
        {
            edges.push_back(map<char,int>());
            length.push_back(i+1);
            link.push_back(0);
            int r=edges.size()-1;
            int p=last;
            while(p>=0 && edges[p].find(s[i])==edges[p].end())
            {
                edges[p][s[i]]=r;
                p=link[p];
            }
            if(p!=-1)
            {
                int q=edges[p][s[i]];
                if(length[p]+1==length[q]) link[r]=q;
                else
                {
                    edges.push_back(edges[q]); // copy edges of q
                    length.push_back(length[p]+1);
                    link.push_back(link[q]); // copy parent of q
                    int qq=edges.size()-1;
                    // add qq as the new parent of q and r
                    link[q]=qq;
                    link[r]=qq;
                    // move short classes pointing to q to point to q'
                    while(p>=0 && edges[p][s[i]]==q)
                    {
                        edges[p][s[i]]=qq;
                        p=link[p];
                    }
                }
            }
            last=r;
        }
        vector<int> terminals;
        int p=last;
        while(p>0)
        {
            terminals.push_back(p);
            p=link[p];
        }
    }
};
int main()
{
    return 0;
}
```

## 6.7 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;
}
```

# 7 Miscellaneous

## 7.1 Stairways(Sqrt decomposition+Convex hull)

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000000000LL
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
typedef long double db;
const int blocks=300;
vector<int> f;
int maxi[MAXN];
vector<int> convex_hull[MAXN];
int n,sz,a[MAXN];
ll dp[MAXN],add1[MAXN],add2[MAXN];
bool need[MAXN];
bool check(int x,int y,int z)
{
    return (db)1.0*(dp[z]-dp[y])/(f[z]-f[y])<=(db)1.0*(dp[y]-dp[x])/(f[y]-f[x]);
```

```
}
void build(int id)
{
    convex_hull[id].clear();
    int st=id*blocks,ed=min(sz,(id+1)*blocks);
    if(add1[id])
    {
        for(int i=st;i<ed;i++) dp[i]+=add1[id];
        add1[id]=0;
    }
    if(add2[id])
    {
        for(int i=st;i<ed;i++) dp[i]+=1LL*add2[id]*f[i];
        add2[id]=0;
    }
    for(int i=id*blocks;i<min(sz,(id+1)*blocks);i++)
    {
        int t=(int)convex_hull[id].size();
        while(t>=2&&check(convex_hull[id][t-2],convex_hull[id][t-1],i))
        {
            convex_hull[id].pop_back();
            t--;
        }
        convex_hull[id].push_back(i); t++;
        while(t>=2&&dp[convex_hull[id][t-2]]<=dp[convex_hull[id][t-1]])
        {
            convex_hull[id].pop_back();
            t--;
        }
    }
    need[id]=false;
}
void update(int x,ll v)
{
    int block_id=x/blocks;
    dp[x]=v-add1[block_id]-1LL*add2[block_id]*f[x];
    need[block_id]=true;
}
ll get_val(int x)
{
    int block_id=x/blocks;
    return dp[x]+add1[block_id]+1LL*add2[block_id]*f[x];
}
ll query_block(int block_id)
{
    if(need[block_id]) build(block_id);
    assert(convex_hull[block_id].size());
    int x=convex_hull[block_id].back();
    return get_val(x);
}
ll query(int l,int r)
{
    ll ret=INF;
    while(l<=r&&l%blocks!=0) {ret=min(ret,get_val(l)); l++;}
    //printf("ret=%lld\n",ret);
    while(l<=r&&r%blocks!=blocks-1) {ret=min(ret,get_val(r)); r--;}
    while(l<r) {ret=min(ret,query_block(l/blocks)); l+=blocks;}
    return ret;
}
```

```
void add(int l,int r,int v)
{
    while(l<=r&&l%blocks!=0) {dp[l]+=v; need[l/blocks]=true; l++;}
    while(l<=r&&r%blocks!=blocks-1) {dp[r]+=v; need[r/blocks]=true; r--;}
    while(l<r) {add1[l/blocks]+=v; l+=blocks;}
}
void addf(int l,int r)
{
    while(l<=r&&l%blocks!=0) {dp[l]+=f[l]; need[l/blocks]=true; l++; }
    while(l<=r&&r%blocks!=blocks-1) {dp[r]+=f[r]; need[r/blocks]=true; r--;}
    while(l<r)
    {
        int block_id=l/blocks;
        if(need[block_id]) build(block_id);
        add2[l/blocks]++;
        int t=(int)convex_hull[block_id].size();
        while(t>=2&&get_val(convex_hull[block_id][t-2])<=get_val(convex_hull[block_id][t-1]))
        {
            convex_hull[block_id].pop_back();
            t--;
        }
        l+=blocks;
    }
}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
        f.push_back(a[i]);
    }
    f.push_back(0);
    sort(f.begin(),f.end()); f.erase(unique(f.begin(),f.end()),f.end());
    for(int i=1;i<=n;i++) a[i]=lower_bound(f.begin(),f.end(),a[i])-f.begin();
    for(int i=1;i<=n;i++) maxi[i]=max(maxi[i-1],a[i]);
    sz=(int)f.size();
    for(int i=0;i<sz;i++) dp[i]=INF;
    dp[0]=0;
    for(int i=0;i<(sz-1)/blocks;i++) build(i);
    memset(need,false,sizeof(need));
    for(int i=1;i<=n;i++)
    {
        if(a[i]==maxi[i]) continue;
        ll val=query(0,a[i]);
        update(a[i],val);
        ll tmp=f[maxi[i]]-f[a[i]];
        add(0,a[i]-1,tmp);
        add(a[i]+1,sz-1,-f[a[i]]);
        addf(a[i]+1,sz-1);
    }
    ll ans=INF;
    for(int i=0;i<sz;i++) ans=min(ans,get_val(i));
    printf("%lld\n",ans);
    return 0;
}
```

## 7.2 Dynamic Tree Diameter(Centroid Decomposition+Segment Tree)

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000000000000LL
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
struct edge{ll to,cost;};
ll n,q,w;
vector<edge> G[MAXN];
bool centroid[MAXN];
unordered_map<ll,ll> st[MAXN],ed[MAXN],whichson[MAXN];
ll rk[MAXN],cursz,curv;
ll sz[MAXN],deep[MAXN],d[MAXN],fa[MAXN];
ll root,ans,tot;
multiset<ll> save;

struct edge2
{
    ll u,v,cost;
}edges[MAXN];
void add_edge(ll u,ll v,ll cost)
{
    G[u].push_back((edge){v,cost});
    G[v].push_back((edge){u,cost});
}
P getroot(ll v,ll p,ll t)//search_centroid
{
    P res=P(INT_MAX,-1);
        ll m=0;
    sz[v]=1;
    for(ll i=0;i<(int)G[v].size();i++)
    {
        ll 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;
}
struct segtree
{
    vector<ll> maxi,lazy;
    multiset<ll> ms;
    void resize(ll sz)
    {
        maxi.clear(); lazy.clear();
        maxi.resize(4*sz+1); lazy.resize(4*sz+1);
    }
    void pushup(ll k)
```

```
    {
        maxi[k]=max(maxi[k*2],maxi[k*2+1]);
    }
    void pushdown(ll k)
    {
        if(!lazy[k]) return;
        for(ll i=k*2;i<=k*2+1;i++)
        {
            maxi[i]+=lazy[k];
            lazy[i]+=lazy[k];
        }
        lazy[k]=0;
    }
    void update(ll k,ll l,ll r,ll x,ll y,ll v)
    {
        if(l>y||x>r) return;
        if(l>=x&&r<=y)
        {
            lazy[k]+=v;
            maxi[k]+=v;
            return;
        }
        pushdown(k);
        ll mid=(l+r)/2;
        update(k*2,l,mid,x,y,v); update(k*2+1,mid+1,r,x,y,v);
        pushup(k);
    }
    ll query(ll k,ll l,ll r,ll x,ll y)
    {
        if(l>y||x>r) return -INF;
        if(l>=x&&r<=y) return maxi[k];
        pushdown(k);
        ll mid=(l+r)/2;
        return max(query(k*2,l,mid,x,y),query(k*2+1,mid+1,r,x,y));
    }
    ll get_ans()
    {
        if(ms.size()==0) return 0;
        else if(ms.size()==1) return max(0LL,*(--ms.end()));
        else
        {
            auto it=ms.end();
            ll ret=0;
            it--; if(*it>0) ret+=*it;
            it--; if(*it>0) ret+=*it;
            return ret;
        }
    }
}seg[MAXN];
void getsz(ll v,ll p)
{
    sz[v]=1;
    for(ll i=0;i<(int)G[v].size();i++)
    {
        ll to=G[v][i].to;
        if(to==p||centroid[to]) continue;
        getsz(to,v);
        sz[v]+=sz[to];
    }
```

```
}
void dfs(ll v,ll p,ll c,ll son)
{
    whichson[curv][v]=son;
    st[curv][v]=++tot;
    for(ll i=0;i<(int)G[v].size();i++)
    {
        ll to=G[v][i].to,cost=G[v][i].cost;
        if(to==p||centroid[to]) continue;
        dfs(to,v,cost,son);
    }
    ed[curv][v]=tot;
    seg[curv].update(1,1,cursz,st[curv][v],ed[curv][v],c);
}
void build_segtree(ll v)
{
    seg[v].resize(sz[v]);
    tot=1;
    st[v][v]=1; ed[v][v]=sz[v];
    curv=v;
    cursz=sz[v];
    for(ll i=0;i<(int)G[v].size();i++)
    {
        ll to=G[v][i].to,cost=G[v][i].cost;
        if(centroid[to]) continue;
        dfs(to,v,cost,to);
    }
    for(ll i=0;i<(int)G[v].size();i++)
    {
        ll to=G[v][i].to;
        if(centroid[to]) continue;
        ll val=seg[v].query(1,1,cursz,st[curv][to],ed[curv][to]);
        seg[v].ms.insert(val);
    }
    save.insert(seg[v].get_ans());
}
void solve(ll v,ll p,ll tot)
{
    rk[v]=tot;
    fa[v]=p;
    centroid[v]=true;
    getsz(v,0);
    build_segtree(v);
    for(ll i=0;i<(int)G[v].size();i++)
    {
        ll to=G[v][i].to,cost=G[v][i].cost;
        if(centroid[to]) continue;
        root=getroot(to,v,sz[to]).S;
        solve(root,v,tot+1);
    }
    root=v;
}
void centroid_decomposition()
{
    memset(centroid,false,sizeof(centroid));
    root=getroot(1,0,n).S;
    solve(root,0,0);
}
ll update_edge(ll id,ll cost)
```

```
{
    ll u=edges[id].u,v=edges[id].v;
    if(rk[v]<rk[u]) swap(u,v);
    ll tmp=u,d=cost-edges[id].cost;
    while(tmp)
    {
        ll vert=(st[tmp][v]<=st[tmp][u]?u:v);
        save.erase(save.find(seg[tmp].get_ans()));
        ll affected=whichson[tmp][vert];
        ll val=seg[tmp].query(1,1,sz[tmp],st[tmp][affected],ed[tmp][affected]);
        seg[tmp].ms.erase(seg[tmp].ms.find(val));
        seg[tmp].update(1,1,sz[tmp],st[tmp][vert],ed[tmp][vert],d);
        val=seg[tmp].query(1,1,sz[tmp],st[tmp][affected],ed[tmp][affected]);
        seg[tmp].ms.insert(val);
        save.insert(seg[tmp].get_ans());
        tmp=fa[tmp];
    }
    edges[id].cost=cost;
}
int main()
{
    scanf("%lld%lld%lld",&n,&q,&w);
    for(ll i=0;i<n-1;i++)
    {
        ll u,v,cost;
        scanf("%lld%lld%lld",&u,&v,&cost);
        edges[i].u=u; edges[i].v=v; edges[i].cost=cost;
        add_edge(u,v,cost);
    }
    centroid_decomposition();
    ll res=*(--save.end());
    ll last=0,d,e;
    for(ll i=0;i<q;i++)
    {
        scanf("%lld%lld",&d,&e);
        d=(d+last)%(n-1);
        e=(e+last)%w;
        update_edge(d,e);
        last=*(--save.end());
        printf("%lld\n",last);
    }
    return 0;
}
```

## 7.3 Dynamic Tree Diameter(Bracket Sequence+Segment Tree)

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000000000000LL
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
struct edge{ll to,cost;};
```

```
ll n,q,w;
ll st[MAXN],ed[MAXN];
vector<edge> G[MAXN];
struct edge2
{
    ll u,v,cost;
}edges[MAXN];
ll tot,dep[2*MAXN];
void add_edge(ll u,ll v,ll cost)
{
    G[u].push_back((edge){v,cost});
    G[v].push_back((edge){u,cost});
}
struct node
{
    node() {val[0]=val[1]=val[2]=val[3]=val[4]=-INF;}
    ll val[5];
    /*  0: a
        1: -2b
        2: a-2b
        3: -2b+c
        4: a-2b+c */
    void init(ll x)
    {
        val[0]=x;
        val[1]=-2*x;
        val[2]=val[3]=-x;
        val[4]=0;
    }
    void add(ll x)
    {
        val[0]+=x; val[1]-=2*x; val[2]-=x; val[3]-=x;
    }
    void update(const node &lhs,const node &rhs)
    {
        for(ll i=0;i<5;i++) val[i]=max(lhs.val[i],rhs.val[i]);
        val[2]=max(val[2],lhs.val[0]+rhs.val[1]);
        val[3]=max(val[3],lhs.val[1]+rhs.val[0]);
        val[4]=max(val[4],max(lhs.val[0]+rhs.val[3],lhs.val[2]+rhs.val[0]));
    }
};
void dfs(ll v,ll p,ll d)
{
    dep[++tot]=d;
    st[v]=tot;
    for(auto e:G[v])
    {
        if(e.to==p) continue;
        dfs(e.to,v,d+e.cost);
        dep[++tot]=d;
    }
    ed[v]=tot;
}
struct segtree
{
    node maxi[8*MAXN];
    ll lazy[8*MAXN];
    void pushup(ll k)
    {
```

```
        maxi[k].update(maxi[k*2],maxi[k*2+1]);
    }
    void pushdown(ll k)
    {
        if(!lazy[k]) return;
        for(ll i=k*2;i<=k*2+1;i++)
        {
            maxi[i].add(lazy[k]);
            lazy[i]+=lazy[k];
        }
        lazy[k]=0;
    }
    void build(ll k,ll l,ll r)
    {
        if(l==r)
        {
            maxi[k].init(dep[l]);
            return;
        }
        ll mid=(l+r)/2;
        build(k*2,l,mid); build(k*2+1,mid+1,r);
        pushup(k);
    }
    void update(ll k,ll l,ll r,ll x,ll y,ll v)
    {
        if(l>y||x>r) return;
        if(l>=x&&r<=y)
        {
            maxi[k].add(v);
            lazy[k]+=v;
            return;
        }
        pushdown(k);
        ll mid=(l+r)/2;
        update(k*2,l,mid,x,y,v); update(k*2+1,mid+1,r,x,y,v);
        pushup(k);
    }
    node query(ll k,ll l,ll r,ll x,ll y)
    {
        if(l>y&&x>r) return node();
        if(l>=x&&r<=y) return maxi[k];
        pushdown(k);
        ll mid=(l+r)/2;
        node ret;
        ret.update(query(k*2,l,mid,x,y),query(k*2+1,mid+1,r,x,y));
        return ret;
    }
}seg;
void update_edge(ll id,ll cost)
{
    ll delta=cost-edges[id].cost;
    ll v=(st[edges[id].u]<st[edges[id].v]?edges[id].v:edges[id].u);
    seg.update(1,1,tot,st[v],ed[v],delta);
    edges[id].cost=cost;
}
int main()
{
    scanf("%lld%lld%lld",&n,&q,&w);
    for(ll i=0;i<n-1;i++)
```

```
    {
        ll u,v,cost;
        scanf("%lld%lld%lld",&u,&v,&cost);
        edges[i].u=u; edges[i].v=v; edges[i].cost=cost;
        add_edge(u,v,cost);
    }
    dfs(1,0,0);
    assert(tot==2*n-1);
    seg.build(1,1,tot);
    ll last=0,d,e;
    for(ll i=0;i<q;i++)
    {
        scanf("%lld%lld",&d,&e);
        d=(d+last)%(n-1);
        e=(e+last)%w;
        update_edge(d,e);
        last=seg.query(1,1,tot,1,tot).val[4];
        printf("%lld\n",last);
    }
    return 0;
}
```

## 7.4 Logical Chain(Kosaraju+std::bitset)

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 255
#define MAXM 25005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
typedef bitset<250> bs;
int n,m;
string str;
bs g[MAXN],rg[MAXN];
bs used;
int ans,cnt;
vector<int> vs;
void dfs(int v)
{
    if(used[v]) return;
    used.set(v);
    bs vis=(~used)&g[v];
    while(vis.any())
    {
        int i=vis._Find_first();
        dfs(i);
        vis=(~used)&g[v];
    }
    vs.push_back(v);
}
void rdfs(int v)
{
```

```
    if(!used[v]) cnt++; else return;
    used.set(v);
    bs vis=(~used)&rg[v];
    while(vis.any())
    {
        int i=vis._Find_first();
        rdfs(i);
        vis=(~used)&rg[v];
    }
}
int korasaju()
{
    ans=0; used.reset();
    vs.clear();
    for(int i=0;i<n;i++) if(!used[i]) dfs(i);
    cnt=0;
    used.reset();
    for(int i=(int)vs.size()-1;i>=0;i--)
        if(!used[vs[i]])
        {
            cnt=0;
            rdfs(vs[i]);
            ans+=cnt*(cnt-1)/2;
        }
    return ans;
}
void modify(int u,int v)
{
    g[u].flip(v); rg[v].flip(u);
}
int main()
{
    scanf("%d%d ",&n,&m);
    for(int i=0;i<n;i++)
    {
        cin>>str;
        for(int j=0;j<n;j++)
            if(str[j]=='1')
            {
                g[i].set(j);
                rg[j].set(i);
            }
    }
    for(int i=0;i<m;i++)
    {
        int x;scanf("%d",&x);
        for(int j=0;j<x;j++)
        {
            int u,v; scanf("%d%d",&u,&v);
            u--;v--;
            modify(u,v);
        }
        printf("%d\n",korasaju());
    }
    return 0;
}
```

## 7.5 Key Array(Splay+reverse operation)

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 100005
#define INF 1000000000
#define MOD 1000000007
#define F first
#define S second
#define lc t[x].ch[0]
#define rc t[x].ch[1]
#define pa t[x].fa
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
int n,q,a[MAXN],root,tot;
struct node
{
    int ch[2];
    int fa;
    int rev,add;
    int sz,w,maxi,mini;
    ll sum;
}t[MAXN];
inline int get(int x) {return t[pa].ch[1]==x;}
inline void pushup(int x)
{
    t[x].sum=t[x].maxi=t[x].mini=t[x].w;
    t[x].sz=1;
    if(lc)
    {
        t[x].maxi=max(t[x].maxi,t[lc].maxi);
        t[x].mini=min(t[x].mini,t[lc].mini);
        t[x].sum+=t[lc].sum;
        t[x].sz+=t[lc].sz;
    }
    if(rc)
    {
        t[x].maxi=max(t[x].maxi,t[rc].maxi);
        t[x].mini=min(t[x].mini,t[rc].mini);
        t[x].sum+=t[rc].sum;
        t[x].sz+=t[rc].sz;
    }
}
inline void rever(int x)
{
    t[x].rev^=1;
    swap(lc,rc);
}
inline void add(int x,int v)
{
    t[x].sum+=1LL*t[x].sz*v;
    t[x].maxi+=v;
    t[x].mini+=v;
    t[x].add+=v;
    t[x].w+=v;
}
inline void pushdown(int x)
```

```
{
    if(t[x].rev)
    {
        if(lc) rever(lc);
        if(rc) rever(rc);
        t[x].rev=0;
    }
    if(t[x].add)
    {
        if(lc) add(lc,t[x].add);
        if(rc) add(rc,t[x].add);
        t[x].add=0;
    }
}
inline void rotate(int x)
{
    int old=pa,oldf=t[old].fa,whichx=get(x);
    t[old].ch[whichx]=t[x].ch[whichx^1]; t[t[old].ch[whichx]].fa=old;
    t[x].ch[whichx^1]=old; t[old].fa=x;
    pa=oldf;
    if(oldf) t[oldf].ch[t[oldf].ch[1]==old]=x;
    pushup(old); pushup(x);
}
inline void splay(int x,int goal=0)
{
    for(int f;(f=t[x].fa)!=goal;rotate(x))
        if(t[f].fa!=goal) rotate((get(x)==get(f))?f:x);
    if(goal==0) root=x;
}
inline int findx(int x,int k)
{
    while(x)
    {
        pushdown(x);
        int cnt=t[lc].sz+1;
        if(k==cnt) return x;
        if(k>cnt) x=rc,k-=cnt;
        else x=lc;
    }
}
void build(int &x,int l,int r,int fa)
{
    if(l>r) return;
    int mid=(l+r)/2;
    x=++tot; pa=fa;
    t[x].w=a[mid];if(mid>=1&&mid<=n) t[x].sz=1;
    build(lc,l,mid-1,x);build(rc,mid+1,r,x);
    pushup(x);
}
void debug()
{
    for(int i=1;i<=n+2;i++)
    {
        printf("%d %d %d %d %d %lld %d %d
            %d\n",i,t[i].fa,t[i].ch[0],t[i].ch[1],t[i].sz,t[i].sum,t[i].maxi,t[i].mini,t[i].add);
    }
}
int adjust(int l,int r)
{
```

```
    int x=findx(root,l-1);
    int y=findx(root,r+1);
    splay(x);splay(y,root);
    return t[y].ch[0];
}
char str[10];
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    build(root,0,n+1,0);
    scanf("%d",&q);
    for(int i=0;i<q;i++)
    {
        int l,r;
        scanf("%s",str);
        scanf("%d%d",&l,&r);l++;r++;
        int pos=adjust(l,r);
        if(str[0]=='c')
        {
            int v;
            scanf("%d",&v);
            add(pos,v);pushup(t[pos].fa);pushup(t[t[pos].fa].fa);
        }
        else if(str[0]=='r')
        {
            rever(pos);
            pushup(t[pos].fa);pushup(t[t[pos].fa].fa);
        }
        else
        {
            printf("%lld %d %d\n",t[pos].sum,t[pos].mini,t[pos].maxi);
        }
    }
    return 0;
}
```

## 7.6 Bimatching(Blossom)

```
#include<bits/stdc++.h>
#define MAXN 1005
#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 V,n,m,x,y,fore,rear,cnt,ans,father[MAXN],f[MAXN],path[MAXN],tra[MAXN],que[MAXN],match[MAXN];
bool a[MAXN][MAXN],check[MAXN],treec[MAXN],pathc[MAXN];
inline void push(int x)
{
    que[++rear]=x;
    check[x]=true;
    if(!treec[x])
    {
        tra[++cnt]=x;
```

```
        treec[x]=true;
    }
}
int root(int x) {return f[x]?f[x]=root(f[x]):x;}
void clear()
{
    for(int i=1,j;i<=cnt;++i)
    {
        j=tra[i];
        check[j]=treec[j]=false;
        father[j]=0,f[j]=0;
    }
}
int lca(int u,int v)
{
    int len=0;
    for(;u;u=father[match[u]])
    {
        u=root(u);
        path[++len]=u;
        pathc[u]=true;
    }
    for(;;v=father[match[v]])
    {
        v=root(v);
        if(pathc[v]) break;
    }
    for(int i=1;i<=len;++i)
    {
        pathc[path[i]]=false;
    }
    return v;
}
void reset(int u,int p)
{
    for(int v;root(u)!=p;)
    {
        if(!check[v=match[u]]) push(v);
        if(f[u]==0) f[u]=p;
        if(f[v]==0) f[v]=p;
        u=father[v];
        if(root(u)!=p) father[u]=v;
    }
}
void flower(int u,int v)
{
    int p=lca(u,v);
    if(root(u)!=p) father[u]=v;
    if(root(v)!=p) father[v]=u;
    reset(u,p);reset(v,p);
}
bool find(int x)
{
    fore=rear=cnt=0,push(x);
    while(fore++<rear)
    {
        int i=que[fore];
        for(int j=1;j<=V;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;
}
int t;
void matching()
{
    for(int i=1;i<=V;i++)
        if(match[i]==0)
        {
            if(find(i)) ans++;
            clear();
        }
}
//1..m: ladies
//m+1..m+n cavalier 1
//m+n+1..m+2n cavalier 2
void add_edge(int u,int v) {a[u][v]=a[v][u]=true;}
char mp[200][200];
int main()
{
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d%d",&n,&m);V=2*n+m;
        memset(a,false,sizeof(a));
        for(int i=1;i<=n;i++) add_edge(m+i,m+n+i);
        for(int i=1;i<=n;i++)
        {
            scanf("%s",mp[i]+1);
            for(int j=1;j<=m;j++)
            {
                int x;
                if(mp[i][j]=='1') {add_edge(j,m+i); add_edge(j,m+n+i);}
            }
        }
        cnt=0;fore=0;rear=0;
        memset(match,0,sizeof(match));
        memset(father,0,sizeof(father));
```

```
        memset(f,0,sizeof(f));
        memset(path,0,sizeof(path));
        memset(tra,0,sizeof(tra));
        memset(que,0,sizeof(que));
        ans=0;
        matching();
        printf("%d\n",ans-n);
    }
    return 0;
}
```

## 7.7 Solar Panels(number theory sqrt for more than one numbers)

```
#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,b,c,d;
int get(int x,int y)
{
    if(y>x) return INF;
    return x/(x/y);
}
int main()
{
    scanf("%d",&n);
    while(n--)
    {
        scanf("%d%d%d%d",&a,&b,&c,&d);
        a--;c--;
        int last=-1,ans=0;
        for(int i=1;i<=max(b,d);i=last+1)
        {
            last=min(min(get(a,i),get(b,i)),min(get(c,i),get(d,i)));
            if((b/i-a/i>0)&&(d/i-c/i>0)) ans=max(ans,last);
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

## 7.8 XOR Tree(Long Chain Decomposition)

```
#pragma GCC optimize(3)
#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 unsigned long long ull;
typedef pair<int,int> P;
int n,k,a[MAXN],len[MAXN],son[MAXN];
int st[MAXN],ed[MAXN];
int cnt[MAXN][4],save[MAXN][4];
ull ans[MAXN];
int curi,curj;
ull res;
vector<int> G[MAXN];
void dfs(int v,int p)
{
    for(auto to:G[v])
    {
        if(to==p) continue;
        dfs(to,v);
        if(len[to]>len[son[v]]) son[v]=to;
    }
    len[v]=len[son[v]]+1;
}
void dfs2(int v,int p)
{
    if(son[v])
    {
        st[son[v]]=st[v]+1;
        dfs2(son[v],v);
        ed[v]=ed[son[v]];
        for(int i=0;i<4;i++)
        {
            cnt[v][i]=cnt[son[v]][i];
            save[st[v]][i]=0;
        }
    }
    else
    {
        ed[v]=st[v];
        for(int i=0;i<4;i++)
        {
            cnt[v][i]=0;
            save[st[v]][i]=0;
        }
    }
    int x=(((a[v]>>curi)&1)<<1)+((a[v]>>curj)&1);
    save[st[v]][x]=1;
    cnt[v][x]++;
    while(ed[v]-st[v]>k)
    {
        for(int i=0;i<4;i++) cnt[v][i]-=save[ed[v]][i];
        ed[v]--;
    }
    for(auto to:G[v])
    {
        if(to==p||to==son[v]) continue;
        st[to]=ed[v]+1;
        dfs2(to,v);
        for(int i=st[to];i<=ed[to]&&i-st[to]<=k;i++)
            for(int j=0;j<4;j++)
            {
```

```
                save[i-st[to]+1+st[v]][j]+=save[i][j];
                cnt[v][j]+=save[i][j];
            }
    }
    ans[v]+=res*cnt[v][0]*cnt[v][3];
    ans[v]+=res*cnt[v][1]*cnt[v][2];
}
int main()
{
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    for(int i=2;i<=n;i++)
    {
        int f;
        scanf("%d",&f);
        G[f].push_back(i);
        G[i].push_back(f);
    }
    dfs(1,0);
    for(curi=0;curi<30;curi++)
        for(curj=0;curj<=curi;curj++)
        {
            if(curi==curj) res=1ULL<<(curi+curj); else res=1ULL<<(curi+curj+1);
            st[1]=1;
            dfs2(1,0);
        }
    for(int i=1;i<=n;i++) printf("%llu\n",ans[i]);
    return 0;
}
```

## 7.9 Coins 2(Knapsack+Period)

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 100005
#define MAXW 5500000
#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 n,a[MAXN],s;
bool dp[MAXW];
ll gcd(ll a,ll b) {return b==0?a:gcd(b,a%b);}
ll lcm(ll a,ll b) {return a*b/gcd(a,b);}
bool deq[MAXW];
int main()
{
    while(scanf("%lld",&n)==1)
    {
        ll m=1;
        for(ll i=1;i<=n;i++) m=lcm(m,i);
        s=0;
        for(ll i=1;i<=n;i++)
        {
```

```
            scanf("%lld",&a[i]);
            s+=i*a[i];
        }
        fill(dp,dp+n*m+1,false);
        dp[0]=true;
        for(ll i=1;i<=n;i++)
        {
            for(ll j=0;j<i;j++)
            {
                ll s=0,t=0,cnt=0;
                for(ll k=0;k*i+j<n*m;k++)
                {
                    bool val=dp[k*i+j];
                    deq[t++]=val;
                    if(val) cnt++;
                    if(cnt) dp[k*i+j]=true; else dp[k*i+j]=false;
                    if(t-s>a[i])
                    {
                        if(deq[s]) cnt--;
                        s++;
                    }
                }
            }
        }
        ll ans=0;
        for(ll i=0;i<n*m;i++)
        {
            if(dp[i])
            {
                ans++;
                if(s-i>=n*m) ans++;
            }
        }
        for(ll i=1;i<=m;i++)
        {
            if(dp[n*m-i])
            {
                ll l=n*m-i+m,r=s-n*m;
                if(r>=l) ans+=(r-l)/m+1;
            }
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```

## 7.10 Subsequence Queries(DP+Matrix)

```
#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;
```

```
typedef vector<int> vec;
typedef vector<vec> mat;
char str[MAXN];
int n;
vec S[MAXN],D[MAXN];
mat id(53,vec(53));
void add(int &a,int b) {a+=b; if(a>=MOD) a-=MOD;}
void dec(int &a,int b) {a-=b; if(a<0) a+=MOD;}
int getnum(char ch)
{
    if(ch>='a'&&ch<='z') return ch-'a';
    else return ch-'A'+26;
}
void compute_S()
{
    for(int i=0;i<=n;i++) S[i].resize(53);
    for(int i=0;i<53;i++) S[0][i]=1;
    mat u=id;
    for(int i=1;i<=n;i++)
    {
        int x=getnum(str[i]);
        for(int j=0;j<53;j++)
        {
            add(S[i][j],2LL*S[i-1][j]%MOD);
            dec(S[i][j],u[x][j]);
            int dif=S[i][j]; dec(dif,S[i-1][j]);
            add(u[x][j],dif);
        }
    }
}
void compute_D()
{
    for(int i=0;i<=n;i++) D[i].resize(53);
    mat u=id;
    for(int i=1;i<=n;i++)
    {
        int x=getnum(str[i]);
        for(int j=0;j<53;j++) D[i][j]=u[x][j];
        for(int j=0;j<53;j++)
        {
            u[x][j]=2LL*u[x][j]%MOD;
            dec(u[x][j],D[i-1][j]);
        }
    }
}
int Q,a0,b0,p,q,r;
int get_ans(int l,int r)
{
    vec tmp(53);
    tmp[52]=1;
    for(int i=0;i<53;i++) dec(tmp[i],D[l-1][i]);
    int ans=0;
    for(int i=0;i<53;i++) add(ans,1LL*S[r][i]*tmp[i]%MOD);
    return ans;
}
int main()
{
    for(int i=0;i<53;i++) id[i][i]=1;
    scanf("%s",str+1);
```

```
    n=strlen(str+1);
    compute_S(); compute_D();
    scanf("%d%d%d%d%d%d",&Q,&a0,&b0,&p,&q,&r);
    int a,b,x,y,z;
    a=a0; b=b0; z=0;
    for(int i=0;i<Q;i++)
    {
        int ta=(1LL*p*a+1LL*q*b+z+r)%MOD;
        int tb=(1LL*p*b+1LL*q*a+z+r)%MOD;
        x=min(ta%n,tb%n)+1;
        y=max(ta%n,tb%n)+1;
        z=get_ans(x,y);
        a=ta; b=tb;
    }
    printf("%d\n",z);
}
```

## 7.11 Distinct Integers(Novel use of segment tree)

```
//Problem Statement: Single Point Modification,
//Range Query, each query asks how many subintervals containing distinct integers in the interval
#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;
int n,q,z,a[MAXN],nxt[MAXN];
ll ans;
set<int> s[MAXN];
struct segtree
{
    int sz[4*MAXN],minx[4*MAXN];
    ll ls[4*MAXN];
    ll get(int k,int x)
    {
        if(sz[k]==1) return min(x,minx[k]);
        else if(minx[k*2+1]>=x) return 1LL*x*sz[k*2+1]+get(k*2,x);
        else return ls[k]+get(k*2+1,x);
    }
    void pushup(int k)
    {
        minx[k]=min(minx[k*2],minx[k*2+1]);
        ls[k]=get(k*2,minx[k*2+1]);
    }
    void build(int k,int l,int r)
    {
        sz[k]=r-l+1;
        if(l==r)
        {
            minx[k]=nxt[l];
            return;
        }
        int mid=(l+r)/2;
```

```
        build(k*2,l,mid); build(k*2+1,mid+1,r);
        pushup(k);
    }
    void update(int k,int l,int r,int p,int v)
    {
        if(l==r)
        {
            minx[k]=v;
            return;
        }
        int mid=(l+r)/2;
        if(p<=mid) update(k*2,l,mid,p,v); else update(k*2+1,mid+1,r,p,v);
        pushup(k);
    }
    void query(int k,int l,int r,int x,int y)
    {
        if(l>y||x>r) return;
        if(l>=x&&r<=y)
        {
            ans+=get(k,z);
            z=min(z,minx[k]);
            return;
        }
        int mid=(l+r)/2;
        query(k*2+1,mid+1,r,x,y); query(k*2,l,mid,x,y);
    }
}seg;
int main()
{
    scanf("%d%d",&n,&q);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&a[i]);
        a[i]++;
        s[a[i]].insert(i);
    }
    for(int i=1;i<=n;i++)
    {
        auto it=s[a[i]].find(i); ++it;
        if(it!=s[a[i]].end()) nxt[i]=*it; else nxt[i]=n+1;
    }
    seg.build(1,1,n);
    for(int i=0;i<q;i++)
    {
        int type,x,y;
        scanf("%d%d%d",&type,&x,&y);
        x++; y++;
        if(type==1)
        {
            ans=0; y--;
            z=y+1;
            seg.query(1,1,n,x,y);
            ans-=1LL*(x+y)*(y-x+1)/2;
            printf("%lld\n",ans);
        }
        else
        {
            auto it=s[a[x]].find(x); assert(it!=s[a[x]].end());
            if(it!=s[a[x]].begin())
```

```
            {
                it--;
                int id=*it;
                it++; it++;
                if(it==s[a[x]].end()) seg.update(1,1,n,id,n+1); else seg.update(1,1,n,id,*it);
                it--;
            }
            s[a[x]].erase(it);
            a[x]=y; s[a[x]].insert(x);
            it=s[a[x]].find(x);
            if(it!=s[a[x]].begin())
            {
                it--;
                int id=*it;
                seg.update(1,1,n,id,x);
                it++;
            }
            it++;
            if(it!=s[a[x]].end()) seg.update(1,1,n,x,*it); else seg.update(1,1,n,x,n+1);
        }
    }
    return 0;
}
```

## 7.12 Forbidden Words(64 times faster)

```
//Problem Statement: given a dictionary of n nonempty words of lowercase English letters
//Two players take turns mentioning letters
//A player loses when it is possible to form a word after she mentions a letter
//Determine who wins
#include<bits/stdc++.h>
#define MAXN 3000005
#define MAXM (1<<21)
#define M (1<<5)
#define F first
#define S second
using namespace std;
typedef unsigned int uint;
typedef long long ll;
typedef pair<int,int> P;
int n;
uint block[MAXM],win[MAXM];
const int L=26-5;
const int LL=(1<<(L+5))-1;
string str;
const uint mask[5]=
{
    0xffff0000,
    0xff00ff00,
    0xf0f0f0f0,
    0xcccccccc,
    0xaaaaaaaa,
};
const uint mask2[5]=
{
    0x0000ffff,
    0x00ff00ff,
```

```
    0x0f0f0f0f,
    0x33333333,
    0x55555555,
};
uint mask3[40];
int main()
{
    freopen("forbidden-words.in","r",stdin);
    freopen("forbidden-words.out","w",stdout);
    for(int i=0;i<32;i++)
        for(int j=0;j<5;j++)
            if((i&(1<<j))==0) mask3[i]|=1<<(i^(1<<j));
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        cin>>str;
        uint m=0;
        for(int j=0;j<(int)str.size();j++) m|=(1<<((int)str[j]-'a'));
        m=((~m)&((1u<<26)-1));
        block[m/M]|=(1<<(m%M));
    }
    for(int j=MAXM-1;j>=0;j--)
    {
        for(int i=0;i<21;i++) if(!(j&(1<<i))) block[j]|=block[j^(1<<i)];
        for(int i=0;i<5;i++) block[j]|=(block[j]>>(1<<(4-i)))&mask2[i];
    }
    for(int i=0;i<MAXM;i++)
    {
        win[i]=block[i];
        for(int j=0;j<21;j++)
            if((i&(1<<j))) win[i]|=~(win[i^(1<<j)]);
        for(int j=0;j<32;j++) if((win[i]&(1<<j))==0) win[i]|=mask3[j];
    }
    if(win[MAXM-1]&0x80000000) puts("First"); else puts("Second");
    return 0;
}
```

## 7.13 Reactor Cooling(Maximum flow with lowerbounds)

```
#include<bits/stdc++.h>
#define MAXN 20005
#define MAXM 1000005
#define INF 10000000000000000LL
#define MOD 1000000007
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
struct edge {ll to,cap,rev,id;};
ll n,m,S,T;
vector<edge> G[MAXN];
ll level[MAXN];
ll iter[MAXN];
ll u[MAXM],v[MAXM],l[MAXM],r[MAXM],ans[MAXM],w[MAXN];
void add_edge(ll from,ll to,ll cap,ll id)
{
```

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

ll dfs(ll v,ll t,ll f)
{
    if(v==t) return f;
    for(ll &i=iter[v];i<(int)G[v].size();i++)
    {
        edge &e=G[v][i];
        if(level[v]<level[e.to]&&e.cap>0)
        {
            ll 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;
}


ll max_flow(ll s,ll t)
{
    ll flow=0;
    for(;;)
    {
        bfs(s);
        if(level[t]<0) return flow;
        memset(iter,0,sizeof(iter));
        ll f;
        while((f=dfs(s,t,INF))>0) flow+=f;
    }
}

int main()
```

```
{
    freopen("cooling.in","r",stdin);
    freopen("cooling.out","w",stdout);
    scanf("%lld%lld",&n,&m);
    S=n+1,T=n+2;
    ll sum=0;
    for(ll i=1;i<=m;i++)
    {
        scanf("%lld%lld%lld%lld",&u[i],&v[i],&l[i],&r[i]);
        w[u[i]]+=l[i]; w[v[i]]-=l[i];
        add_edge(u[i],v[i],r[i]-l[i],i);
    }
    for(ll i=1;i<=n;i++)
    {
        if(w[i]>0) {add_edge(i,T,w[i],0); sum+=w[i];}
        else add_edge(S,i,-w[i],0);
    }
    ll f=max_flow(S,T);
    if(f!=sum) {puts("NO"); return 0;}
    puts("YES");
    for(ll i=1;i<=n+2;i++)
    {
        for(auto &e:G[i])
        {
            if(!e.id) continue;
            ans[e.id]=l[e.id]+(r[e.id]-l[e.id]-e.cap);
        }
    }
    for(ll i=1;i<=m;i++) printf("%lld\n",ans[i]);
    //add_edge(S,s,INF); add_edge(s,S,INF);
}
```

## 7.14 Great Siberian Wall(Minimum Enclosing Triangle)

```
#include<bits/stdc++.h>
#define MAXN 85
#define INF 1000000000
#define MOD 1000000000
#define F first
#define S second
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
typedef double db;
const db PI=acos(-1.0);
const db eps=1e-8;
int sgn(db ta,db tb)
{
    if(fabs(ta-tb)<eps) return 0;
    if(ta<tb) return -1;
    return 1;
}
struct Point
{
    db x,y;
    Point(){}
    Point(db tx,db ty){x=tx,y=ty;}
```

```
    void print()
    {
        printf("%.10f %.10f\n",x,y);
    }
    bool operator<(const Point &q) const
    {
        return x<q.x||(x==q.x&&y<q.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 &p,const Point &q)
    {
        return Point(p.x-q.x,p.y-q.y);
    }
    friend Point operator /(const Point &p,const db v)
    {
        return Point(p.x/v,p.y/v);
    }
    friend Point operator *(const Point &p,const db v)
    {
        return Point(p.x*v,p.y*v);
    }
    db operator ^ (const Point &b) const
    {
        return x*b.y-y*b.x;
    }
    bool operator == (const Point &_off) const
    {
        return sgn(x,_off.x)==0&&sgn(y,_off.y)==0;
    }
};
db 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);
}
db 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));
}
db tri(const Point &a,const Point &b,const Point &c)
{
    return (b-a)^(c-a);
}
struct Line
{
    Point s,e;
    db a,b,c;
    db ang;
    Line(){}
    Line(Point ts,Point te):s(ts),e(te){}
    Line(db _a,db _b,db _c):a(_a),b(_b),c(_c){}

    bool pton()
    {
        a=e.y-s.y;
        b=s.x-e.x;
        c=e.x*s.y-e.y*s.x;
```

```
        return true;
    }

    friend db 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);
    }

};
static bool parallel(const Line &_st,const Line &_se)
{
    return sgn(_st/_se,0)==0;
}
static bool crossLPt(const Line &_st,const Line &_se,Point &ret)
{
    db ta=tri(_st.s,_st.e,_se.s);
    db tb=tri(_st.s,_st.e,_se.e);
    if(!sgn(ta,0)&&!sgn(tb,0)) return 0;
    if(!sgn(ta-tb,0)) return 0;
    ret=(_se.s*tb-_se.e*ta)/(tb-ta);
    return 1;
}
static bool isCrossSS(const Line &_st,const Line &_se,bool f=false)
{
    if(parallel(_st,_se)) return false;
    if(sgn(max(_st.s.x, _st.e.x) , min(_se.s.x, _se.e.x))<=0 ||
        sgn(max(_se.s.x, _se.e.x) , min(_st.s.x, _st.e.x))<=0 ||
        sgn(max(_st.s.y, _st.e.y) , min(_se.s.y, _se.e.y))<=0 ||
        sgn(max(_se.s.y, _se.e.y) , min(_st.s.y, _st.e.y))<=0 ) return false;
if(f)printf("xmult %lf %lf\n",xmult(_st.s,_se.s,_st.e)*xmult( _st.s, _se.e, _st.e),xmult( _se.s,
     _st.s, _se.e)*xmult( _se.s, _st.e, _se.e));
    return

        xmult( _st.s, _se.s, _st.e) * xmult( _st.s, _se.e, _st.e) < -1e-4 &&
         xmult( _se.s, _st.s, _se.e) * xmult( _se.s, _st.e, _se.e) < -1e-4;
};
Point gsort;
bool gcmp(const Point &ta,const Point &tb)
{
    db tmp=xmult(gsort,ta,tb);
    if(fabs(tmp)<eps) return getdis(gsort,ta)<getdis(gsort,tb);
    else if(tmp>0) return 1;
    return 0;
}
struct Polygon
{
    const static int maxpn=80;
    Point pt[maxpn];
    int n;
    db area()
    {
        db 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);
    }
```

```
    void orient()
    {
        db 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;
        }
        if(ans>0) reverse(pt,pt+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--;
            pt[n++]=_p[i];
        }
    }
    bool has(const Point &p)
    {
        //return false;
        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=0;
        while(l<=r)
        {
            int mid=(l+r)/2;
            if(xmult(pt[0],p,pt[mid])<-eps) line=mid,r=mid-1;
            else l=mid+1;
        }
        return xmult(pt[line-1],p,pt[line])<-eps;
    }
};
Point pt[MAXN];
Polygon pg;
int n;
db ans=INF;
Point pa,pb,pc;
db triangle_area(const Point &a,const Point &b,const Point &c)
{
    db ans=0.0;
    ans+=a.x*b.y-b.x*a.y;
    ans+=b.x*c.y-c.x*b.y;
    ans+=c.x*a.y-a.x*c.y;
    return fabs(ans/2.0);
};
bool check_valid(Point a,Point b,Point c)
{
```

```
    bool f=false;
    if(pg.has(a)||pg.has(b)||pg.has(c)) return false;
    //if(f) assert(0);
    if(sgn(tri(a,b,c),pg.area())<0) return false;
    //if(f) assert(0);
    for(int i=0;i<pg.n;i++)
    {
        if(f) printf("id=%d\n",i);
        int nxt=(i+1)%pg.n;
        Line l=Line(pg.pt[i],pg.pt[nxt]);
        if(isCrossSS(l,Line(a,b),f)) return false;
        if(isCrossSS(l,Line(b,c),f)) return false;
        if(isCrossSS(l,Line(c,a),f)) return false;
    }
    //if(f) assert(0);
    return true;
}
void update_ans(const Point &a,const Point &b,const Point &c,bool checkstatus=true)
{
    //if (checkstatus) {if(!check_valid(a,b,c)) return;}
    db area=tri(a,b,c);
    if(area>eps&&area<ans)
    {
        ans=area;
        pa=a; pb=b; pc=c;
    }
}
int main()
{
    freopen("wall.in","r",stdin);
    freopen("wall.out","w",stdout);
    scanf("%d",&n);
    for(int i=0;i<n;i++) scanf("%lf%lf",&pt[i].x,&pt[i].y);
    pg.graham(pt,n);
    //three lines of polygon
    int nn=pg.n;
    //printf("%d\n",nn);
    for(int i=0;i<nn;i++)
        for(int j=i+1;j<nn;j++)
            for(int k=j+1;k<nn;k++)
            {
                Line la,lb,lc;
                la=Line(pg.pt[i],pg.pt[(i+1)%nn]);
                lb=Line(pg.pt[j],pg.pt[(j+1)%nn]);
                lc=Line(pg.pt[k],pg.pt[(k+1)%nn]);
                Point a,b,c;
                if(!crossLPt(la,lb,a)) continue;
                if(!crossLPt(lb,lc,b)) continue;
                if(!crossLPt(lc,la,c)) continue;
                /*if(!sgn(a.x,-871.4382322842)||!sgn(b.x,-871.4382322842)||!sgn(c.x,-871.4382322842))
                {
                    a.print(); b.print(); c.print();
                    printf("\n");
                }*/
                //printf("%f\n",tri(a,b,c));
                if(tri(a,b,c)>0&&tri(a,b,c)<ans)
                {
                    ans=tri(a,b,c);
                    pa=a; pb=b; pc=c;
```

```
            }
        }
    //two lines and a vertex
    for(int i=0;i<nn;i++)
        for(int j=i+1;j<nn;j++)
        {
            Line la,lb;
            Point pp,q1,q2;
            la=Line(pg.pt[i],pg.pt[(i+1)%nn]);
            lb=Line(pg.pt[j],pg.pt[(j+1)%nn]);
            if(!crossLPt(la,lb,pp)) continue;
            if(getdis(pg.pt[i],pp)>getdis(pg.pt[(i+1)%nn],pp)) q1=pg.pt[i]; else q1=pg.pt[(i+1)%nn];
            if(getdis(pg.pt[j],pp)>getdis(pg.pt[(j+1)%nn],pp)) q2=pg.pt[j]; else q2=pg.pt[(j+1)%nn];
            int g=sgn(tri(q1,q2,pp),0);
            for(int k=0;k<nn;k++)
            {
                if(!sgn(tri(pp,q1,pg.pt[k]),0)||!sgn(tri(pp,q2,pg.pt[k]),0)||sgn(tri(q1,q2,pg.pt[k])*g,0)>=0)
                    continue;
                Point p2=pg.pt[k]+pg.pt[k]-pp;
                Point p3=p2+q2-pp;
                Point p4,p5;
                crossLPt(Line(pp,q1),Line(p2,p3),p4);
                crossLPt(Line(pp,q2),Line(p4,pg.pt[k]),p5);
                if(sgn(tri(p4,p5,pg.pt[(k+1)%nn])*g,0)>=0&&sgn(tri(p4,p5,pg.pt[(k-1+nn)%nn])*g,0)>=0)
                {
                    db d=fabs(tri(pp,p4,p5));
                    if(d<ans)
                    {
                        ans=d;
                        pa=pp; pb=p4; pc=p5;
                    }
                }
            }
        }
    printf("%.10f %.10f \n%.10f %.10f\n%.10f %.10f\n",pa.x,pa.y,pb.x,pb.y,pc.x,pc.y);
    return 0;
}
```

## 7.15 Road Times(Simplex)

```
#pragma GCC optimize(3)
#include<bits/stdc++.h>
#define MAXN 105
#define MAXM 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;
typedef double db;
typedef vector<db> vec;
typedef vector<vec> mat;
const db eps=1e-8;
//minimize
bool eq(db a,db b)
```

```
{
    return fabs(a-b)<eps;
}
bool ls(db a,db b)
{
    return a<b&&!eq(a,b);
}
vec simplex(mat a)
{
    int n=(int)a.size()-1;
    int m=(int)a[0].size()-1;
    vec left(n+1),up(m+1);
    iota(up.begin(),up.end(),0);
    iota(left.begin(),left.end(),m);
    auto pivot=[&](int x,int y)
    {
        swap(left[x],up[y]);
        db k=a[x][y];
        a[x][y]=1;
        vector<int> vct;
        for(int j=0;j<=m;j++)
        {
            a[x][j]/=k;
            if(!eq(a[x][j],0)) vct.push_back(j);
        }
        for(int i=0;i<=n;i++)
        {
            if(eq(a[i][y],0)||i==x) continue;
            k=a[i][y];
            a[i][y]=0;
            for(int j:vct) a[i][j]-=k*a[x][j];
        }
    };
    while(1)
    {
        int x=-1;
        for(int i=1;i<=n;i++) if(ls(a[i][0],0)&&(x==-1||a[i][0]<a[x][0])) x=i;
        if(x==-1) break;
        int y=-1;
        for(int j=1;j<=m;j++) if(ls(a[x][j],0)&&(y==-1||a[x][j]<a[x][y])) y=j;
        assert(y!=-1);
        pivot(x,y);
    }
    while(1)
    {
        int y=-1;
        for(int j=1;j<=m;j++) if(ls(0,a[0][j])&&(y==-1||a[0][j]>a[0][y])) y=j;
        if(y==-1) break;
        int x=-1;
        for(int i=1;i<=n;i++) if(ls(0,a[i][y])&&(x==-1||a[i][0]/a[i][y]<a[x][0]/a[x][y])) x=i;
        assert(x!=-1);
        pivot(x,y);
    }
    vector<double> ans(m+1);
    for(int i=1;i<=n;i++) if(left[i]<=m) ans[left[i]]=a[i][0];
    ans[0]=-a[0][0];
    return ans;
}
int n,m;
```

```
int main()
{
    scanf("%d",&n);
    mat e(n,vec(n)),id(n,vec(n,-1));
    vec len;
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
        {
            scanf("%lf",&e[i][j]);
            if(e[i][j]>0)
            {
                id[i][j]=m++;
                len.push_back(e[i][j]);
            }
        }
    auto getPath=[&](int s,int f)
    {
        vector<int> d(n,INF);
        vector<pair<int,int> > pr(n);
        vector<char> vis(n);
        d[s]=0;
        for(int it=0;it<n;it++)
        {
            int v=-1;
            for(int j=0;j<n;j++) if(!vis[j]&&(v==-1||d[j]<d[v])) v=j;
            vis[v]=1;
            for(int to=0;to<n;to++)
            {
                if(e[v][to]==-1) continue;
                if(d[to]>d[v]+e[v][to])
                {
                    d[to]=d[v]+e[v][to];
                    pr[to]={v,id[v][to]};
                }
            }
        }
        vector<db> res(m);
        while(f!=s)
        {
            res[pr[f].S]=1;
            f=pr[f].F;
        }
        return res;
    };
    mat a;
    int r;
    scanf("%d",&r);
    for(int i=0;i<m;i++)
    {
        vec cur(m+1);
        cur[0]=2*len[i]; cur[1+i]=1; a.push_back(cur);
        cur[0]=-len[i]; cur[1+i]=-1; a.push_back(cur);
    }
    for(int i=0;i<r;i++)
    {
        int s,f,t;
        scanf("%d%d%d",&s,&f,&t);
        auto cur=getPath(s,f);
        cur.insert(cur.begin(),t);
```

```
        a.push_back(cur);
        for(db &x:cur) x*=-1;
        a.push_back(cur);
    }
    int q;
    scanf("%d",&q);
    for(int i=0;i<q;i++)
    {
        int s,f;
        scanf("%d%d",&s,&f);
        auto cur=getPath(s,f);
        cur.insert(cur.begin(),0);
        auto ca=a;
        ca.insert(ca.begin(),cur);
        db res1=simplex(ca)[0];
        for(auto &x:ca[0]) x*=-1;
        db res2=-simplex(ca)[0];
        printf("%d %d %.15f %.15f\n",s,f,res2,res1);
    }
    return 0;
}
```

## 7.16 Subset Sum(Set balancing)

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

#define rep(i, n) for (int i = 0; i < int(n); i++)
#define Rep(i, n) for (int i = 1; i <=int(n); i++)
#define range(x) begin(x), end(x)

const int MAXN = 20005;

int n, c;
int wt[MAXN];
int _dp[2][MAXN * 2 + 1];

inline
int& dp(int x, int y) {
    return _dp[x&1][y - c + MAXN];
}

int main() {
    int T; scanf("%d", &T);
    while (T--) {
        scanf("%d%d", &n, &c);
        Rep (i, n) scanf("%d", wt + i);
        int totw = 0, ptr;
        for (ptr = 1; ptr <= n; ptr++) {
            totw += wt[ptr];
            if (totw > c) break;
        }
        if (totw <= c) {
            printf("%d\n", totw);
            continue;
        }
```

```
        memset(_dp, 0, sizeof _dp);
       // for(int w = c +MAXN-1; w>=c;w--) dp(ptr-1,w)=1;
        dp(ptr-1, totw - wt[ptr]) = ptr;
        for (int t = ptr; t <= n; t++) {
            memset(_dp[t&1], 0, sizeof _dp[0]);
            for(int w=c-MAXN+1;w<=c+MAXN;w++) dp(t,w)=dp(t-1,w);
            // balanced adding
            for (int w = c - MAXN+1; w <= c; w++)
            {
                dp(t, w) = max(dp(t,w), dp(t-1,w));
                dp(t, w + wt[t]) = max(dp(t-1, w), dp(t, w + wt[t]));
            }
            // balanced deleting
            for (int w = c + MAXN; w > c; w--)
            {
                for (int j = dp(t, w)-1 ; j >= dp(t-1, w); j--)
                    dp(t, w - wt[j]) = max(dp(t, w - wt[j]), j);
            }
            //for(int w= c-MAXN+1;w<=c+MAXN;w++) printf("%d %d %d\n",t,w,dp(t,w));
        }
        for (totw = c; dp(n, totw) == 0; totw--); // cout << totw << ' ' << dp(n, totw) << endl;
        printf("%d\n", totw);
    }
}
```

## 7.17 Longest Shortest Path(Dual Program)

```
//Problem: given graph and s,t, each edge e has initial length d_e and cost c_e
//one can pay x*c_e to make its length d_e+x(x can be noninteger)
//target: maximize shortest path from s to t within cost P
#include <bits/stdc++.h>
using namespace std;

#define rep(i, n) for (int i = 0; i < int(n); i++)
#define Rep(i, n) for (int i = 1; i <=int(n); i++)
#define range(x) begin(x), end(x)
#ifdef __LOCAL_DEBUG__
#define _debug(fmt, ...) fprintf(stderr, "[%s] " fmt "\n", __func__, ##__VA_ARGS__)
#else
#define _debug(...) ((void) 0)
#endif
typedef long long LL;
typedef unsigned long long ULL;

struct edge{
    int from, to;
    int cap, flow;
    LL cost;
};

const LL INF = LLONG_MAX / 2;
const int MAXN = 5005;
struct MCMF {
    int s, t, n, m;
    vector<edge> edges;
    vector<int> G[MAXN];
    bool inq[MAXN]; // queue
```

```
    LL d[MAXN];   // distance
    int p[MAXN];  // previous
    int a[MAXN];  // improvement

    void add_edge(int from, int to, int cap, LL cost) {
        edges.push_back(edge{from, to, cap, 0, cost});
        edges.push_back(edge{to, from, 0, 0, -cost});
        m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }

    bool spfa(){
        queue<int> q;
        fill(d, d + MAXN, INF); d[s] = 0;
        memset(inq, 0, sizeof(inq));
        q.push(s); inq[s] = true;
        p[s] = 0; a[s] = INT_MAX;
        while (!q.empty()){
            int u = q.front(); q.pop(); inq[u] = false;
            for (int i : G[u]) {
                edge& e = edges[i];
                if (e.cap > e.flow && d[e.to] > d[u] + e.cost){
                    d[e.to] = d[u] + e.cost;
                    p[e.to] = i;
                    a[e.to] = min(a[u], e.cap - e.flow);
                    if (!inq[e.to]) q.push(e.to), inq[e.to] = true;
                }
            }
        }
        return d[t] != INF;
    }

    void augment(){
        int u = t;
        while (u != s){
            edges[p[u]].flow += a[t];
            edges[p[u]^1].flow -= a[t];
            u = edges[p[u]].from;
        }
    }

    double solve(int s, int t, int p) {
        this->s = s; this->t = t;
        int flow = 0;
        LL cost = 0;
        double ans = DBL_MAX;
        while (spfa()) {
            augment();
            flow += a[t]; cost += a[t] * d[t];
            ans = min(ans, double(cost + p) / flow);
        }
        return ans;
    }
} mcmf;

int n, m, p, s, t;

int main() {
```

```
    scanf("%d%d%d%d%d", &n, &m, &p, &s, &t);
    rep (_, m) {
        int u, v, d, c; scanf("%d%d%d%d", &u, &v, &d, &c);
        mcmf.add_edge(u, v, c, d);
    }
    printf("%.12f\n", mcmf.solve(s, t, p));
    return 0;
}
```