

Template for XCPC

SUNCHAOYI

2025-2026

Contents

1	Compile	3
1.1	Fast I/O	3
1.2	Run.bash	3
1.3	Run.ps1	3
2	Graph	4
2.1	Diameter of a Tree	4
2.2	Centroid of a Tree	4
2.3	Minimum Spanning Tree	5
2.3.1	Prim	5
2.3.2	Kruskal	5
2.4	LCA	5
2.5	Topological Sorting	6
2.6	Shortest Path	7
2.6.1	Dijkstra	7
2.6.2	SPFA	7
2.7	DCC	8
2.8	Tarjan	9
2.9	Bipartite Graph Matchings	10
2.10	Flow	10
2.10.1	Edmonds-Karp	10
2.10.2	Dinic	11
3	Data Structure	13
3.1	Segment Tree	13
3.2	Fenwick Tree	13
3.3	Heavy-Light Decomposition	13
3.4	Splay Tree	13
3.5	Sparse Table	13
3.6	Persistent Data Structure	13
3.7	Linear-Basis	13
4	Math	15
4.1	Elementary Arithmetic Bucket	15
4.2	Combination	15
4.3	The Sieve of Primes Euler's Totient Function Möbius Inversion	16
4.4	Exgcd	16
4.5	CRT	16
4.6	Gaussian Elimination	16
4.7	FFT	17
5	Computational Geometry	17

6	String	22
6.1	Hash	22
6.2	KMP	22
6.3	Manacher	22
6.4	Trie	23
6.4.1	Trie	23
6.4.2	01-Trie	23
6.5	Aho-Corasick Automaton	24
6.6	SA/SAM	24
6.7	Border/Fail Tree	25

1 Compile

1.1 Fast I/O

```
#include <bits/stdc++.h>
#define init(x) memset (x,0,sizeof (x))
#define ll long long
#define ull unsigned long long
#define INF 0x3f3f3f3f
#define pii pair <int,int>
using namespace std;
const int MAX = 1e5 + 5;
const int MOD = 1e9 + 7;
char s[200];
inline int read ();
inline void output ();
int main ()
{
    //freopen (".in","r",stdin);
    //freopen (".out","w",stdout);
    return 0;
}
inline int read ()
{
    int s = 0;int f = 1;
    char ch = getchar ();
    while ((ch < '0' || ch > '9') && ch != EOF)
    {
        if (ch == '-') f = -1;
        ch = getchar ();
    }
    while (ch >= '0' && ch <= '9')
    {
        s = s * 10 + ch - '0';
        ch = getchar ();
    }
    return s * f;
}
inline void output (int x)
{
    if (x < 0) putchar ('-');
    x = (x > 0) ? x : -x;
    int cnt = 0;
    while (x)
    {
        s[cnt++] = x % 10 + '0';
        x /= 10;
    }
    while (cnt) putchar (s[--cnt]);
}
```

1.2 Run.bash

```
#!/bin/bash
g++ -std=c++17 -O2 -Wall "$1" -o main
./main < in.txt > out.txt
```

1.3 Run.ps1

```
# Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
# new file : 'type nul > filename.cpp'
# run : '.\run.ps1 filename.cpp'
g++ -std=c++17 -O2 -Wall $args[0] -o main
cat in.txt | .\main | Out-File -FilePath out.txt
```

2 Graph

2.1 Diameter of a Tree

```
template <typename T>
class Diameter
{
    int n,p;
    vector <T> dis;
    vector <vector <pair <int,T>>> ve;
    void dfs (int u,int fa)
    {
        for (const auto& [v,w] : ve[u])
        {
            if (v == fa) continue;
            dis[v] = dis[u] + w;
            if (dis[v] > dis[p]) p = v;
            dfs (v,u);
        }
    }
public:
    Diameter (int n) : n(n),ve(n + 1) {}
    void add (int u,int v,T w = 1) {ve[u].push_back ({v,w});ve[v].push_back ({u,w});}
    T calc ()
    {
        dis.assign (n + 1,0);
        p = 1;dfs (1,0);
        dis[p] = 0;dfs (p,0);
        return dis[p];
    }
};
```

2.2 Centroid of a Tree

```
class Centroid
{
    int n;
    vector <int> sz,w,cen;
    vector <vector <int>> ve;
    void dfs (int u,int fa)
    {
        sz[u] = 1;w[u] = 0;
        for (auto v : ve[u])
        {
            if (v == fa) continue;
            dfs (v,u);
            sz[u] += sz[v];
            w[u] = max (w[u],sz[v]);
        }
        w[u] = max (w[u],n - sz[u]);
        if (w[u] <= n / 2) cen.push_back (u);
    }
public:
    Centroid (int n) : n(n),ve (n + 1),sz (n + 1),w (n + 1) {}
    void add (int u,int v) {ve[u].push_back (v);ve[v].push_back (u);}
    auto calc ()
    {
        cen.clear ();
        dfs (1, 0);
        sort (cen.begin (),cen.end ());
        return cen;
    }
};
```

```
};
```

2.3 Minimum Spanning Tree

2.3.1 Prim

```
template <typename T>
class MST
{
    int n;T ans;
    vector <int> vis;vector <vector <int>> g;vector <T> dis;
public :
    MST (int n) : n (n),vis (n + 1,0),g (n + 1,vector <int> (n + 1,INF)),dis (n + 1,INF) {dis[1] = ans = 0;vis[1] = 1;}
    void add (int u,int v,T w) {g[u][v] = g[v][u] = w;}
    T calc ()
    {
        for (int i = 2;i <= n;++i) dis[i] = g[1][i];
        for (int i = 1;i < n;++i)
        {
            int k = 0;
            for (int j = 1;j <= n;++j)
                if (!vis[j] && dis[j] < dis[k]) k = j;
            vis[k] = 1;
            ans += dis[k];
            for (int j = 1;j <= n;++j)
                if (!vis[j] && g[k][j] < dis[j]) dis[j] = g[k][j];
        }
        return ans;
    }
};
```

2.3.2 Kruskal

```
template <typename T>
class MST
{
    int n,m,e_cnt,cnt;T ans;
    struct node {int u,v;T w;};
    vector <int> fa;vector <node> g;
public:
    MST (int n,int m) : n (n),m(m),fa (n + 1,0),g (m + 1) {cnt = e_cnt = ans = 0;}
    void add (int u,int v,int w) {g[++e_cnt].u = u,g[e_cnt].v = v,g[e_cnt].w = w;}
    int getfa (int u) {return fa[u] == u ? u : fa[u] = getfa (fa[u]);}
    T calc ()
    {
        sort (g.begin (),g.end (),[](auto &x,auto &y) {return x.w < y.w;});
        for (int i = 1;i <= n;++i) fa[i] = i;
        for (int i = 1;cnt < n && i <= m;++i)
        {
            int dx = getfa (g[i].u),dy = getfa (g[i].v);
            if (dx == dy) continue;
            ans += g[i].w;fa[dx] = dy;++cnt;
        }
        return ans;
    }
};
```

2.4 LCA

```
class LCA
{
```

```

static constexpr int lg = 20;
int n;
vector<int> dep;
vector<vector<int>> f, ve;
public:
LCA (int n) : n (n), ve (n + 1), dep (n + 1), f (n + 1, vector<int> (lg + 1, 0))
{}
void add (int u, int v) {ve[u].push_back (v); ve[v].push_back (u);}
void pre (int u, int fa)
{
    f[u][0] = fa; dep[u] = dep[fa] + 1;
    for (int i = 0; i < lg; ++i) f[u][i + 1] = f[f[u][i]][i];
    for (auto v : ve[u])
        if (v != fa) pre (v, u);
}
int query (int u, int v)
{
    if (dep[u] < dep[v]) swap (u, v);
    for (int i = lg; ~i; --i)
    {
        if (dep[f[u][i]] >= dep[v]) u = f[u][i];
        if (u == v) return u;
    }
    for (int i = lg; ~i; --i)
        if (f[u][i] != f[v][i]) u = f[u][i], v = f[v][i];
    return f[u][0];
}
};

```

2.5 Topological Sorting

```

class Topo
{
    int n;
    vector<int> deg;
    vector<vector<int>> ve;

public:
Topo (int n) : n (n), ve (n + 1), deg (n + 1, 0) {}
void add (int u, int v)
{
    ve[u].push_back (v);
    ++deg[v];
}
vector<int> calc ()
{
    queue<int> q;
    vector<int> lst;
    for (int i = 1; i <= n; ++i)
        if (!deg[i]) q.push (i);
    while (!q.empty ())
    {
        int u = q.front (); q.pop ();
        lst.push_back (u);
        for (auto v : ve[u])
        {
            --deg[v];
            if (!deg[v]) q.push (v);
        }
    }
    return lst;
}
};

```

2.6 Shortest Path

2.6.1 Dijkstra

```
class dijkstra
{
    int n,m,cnt;
    vector <int> head,to,nxt,val,vis;
    vector <ll> dis;
public:
    dijkstra (int n,int m) :
        n (n),m (m),vis (n + 1,0),head (n + 1,0),dis (n + 1,INF),
        to (2 * m + 1,0),nxt (2 * m + 1,0),val (2 * m + 1,0) {cnt = 0;}
    void add (int u,int v,int w)
    {
        to[++cnt] = v;val[cnt] = w;nxt[cnt] = head[u];head[u] = cnt;
        to[++cnt] = u;val[cnt] = w;nxt[cnt] = head[v];head[v] = cnt;
    }
    auto calc (int s)
    {
        priority_queue <pii> q;
        for (int i = 1;i <= n;++i) vis[i] = 0,dis[i] = INF;
        q.push ({0,s});
        dis[s] = 0;
        while (!q.empty ())
        {
            int u = q.top ().second;q.pop ();
            if (vis[u]) continue;
            vis[u] = 1;
            for (int i = head[u];i;i = nxt[i])
            {
                int v = to[i];
                if (dis[v] > dis[u] + val[i])
                {
                    dis[v] = dis[u] + val[i];
                    q.push ({-dis[v],v});
                }
            }
        }
        return dis;
    }
};
```

2.6.2 SPFA

```
class SPFA
{
    int n,m,cnt;
    vector <int> head,to,nxt,val,vis,times;
    vector <ll> dis;
public:
    SPFA (int n,int m) :
        n (n),m (m),times (n + 1,0),vis (n + 1,0),head (n + 1,0),dis (n + 1,INF),
        to (2 * m + 1,0),nxt (2 * m + 1,0),val (2 * m + 1,0) {cnt = 0;}
    void add (int u,int v,int w)
    {
        to[++cnt] = v;val[cnt] = w;nxt[cnt] = head[u];head[u] = cnt;
        to[++cnt] = u;val[cnt] = w;nxt[cnt] = head[v];head[v] = cnt;
    }
    auto calc (int s)
    {
        queue <int> q;
        for (int i = 1;i <= n;++i) vis[i] = 0,dis[i] = INF;
        dis[s] = 0,vis[s] = 1;q.push (s);
    }
};
```

```

while (!q.empty())
{
    int u = q.front ();
    q.pop (),vis[u] = 0;
    for (int i = head[u];i;i = nxt[i])
    {
        int v = to[i];
        if (dis[v] > dis[u] + val[i])
        {
            dis[v] = dis[u] + val[i];
            times[v] = times[u] + 1;
            if (times[v] >= n) return {-1};//Negative Cycle
            if (!vis[v]) q.push (v),vis[v] = 1;
        }
    }
}
return dis;
}
};

```

2.7 DCC

```

class DCC
{
public:
    struct Edge {int to,nxt;};
    vector <Edge> G;
    vector <int> head,dfn,low,col;
    vector <bool> cut;
    stack <int> stk;
    int cnt,cct,n;
    void init (int nn = 0)
    {
        G.assign (2,{0,0});
        n = nn;head.assign (n + 1,0);
    }
    void dfs (int u,int f = -1)
    {
        dfn[u] = low[u] = ++cnt;
        int ch = 0;
        for (int i = head[u];i;i = G[i].nxt)
            if (i != f)
            {
                int v = G[i].to;
                if (dfn[v] < dfn[u]) stk.push (i >> 1);
                if (!dfn[v])
                {
                    dfs (v,i ^ 1);++ch;
                    low[u] = min (low[u],low[v]);
                    if (low[v] >= dfn[u])
                    {
                        int I = 0;
                        ++cct;cut[u] = true;
                        do
                        {
                            assert (!stk.empty ());
                            I = stk.top ();stk.pop ();
                            col[I] = cct;
                        }
                        while (I != (i >> 1));
                    }
                }
            }
        else if (dfn[v] < low[u]) low[u] = dfn[v];
    }
}

```



```

        if (f == -1 && ch == 1) cut[u] = false;
    }
    void tarjan ()
    {
        cnt = cct = 0;
        col.assign (G.size () >> 1,0);
        dfn.assign (n + 1,0);
        low.assign (n + 1,0);
        cut.assign (n + 1,false);
        while (!stk.empty ()) stk.pop ();
        for (int i = 1;i <= n;++i) if (!dfn[i]) dfs (i);
    }
    void insert (int u,int v)
    {
        G.push_back ({v,head[u]});head[u] = G.size () - 1;
        G.push_back ({u,head[v]});head[v] = G.size () - 1;
    }
    void insert (vector <vector <int>> &G1)
    {
        for (unsigned u = 1;u < G1.size ();++u)
            for (int v : G1[u]) insert (u,v);
    }
    bool operator [] (const int &x) const {return cut[x];}
};

```

2.8 Tarjan

```

class Tarjan
{
    int n,m,cnt,times,scc_cnt;
    vector <int> head,to,nxt,low,scc,dfn;
    stack <int> s;
    void tarjan (int u)
    {
        low[u] = dfn[u] = ++times;
        s.push (u);
        for (int i = head[u];i;i = nxt[i])
        {
            int v = to[i];
            if (!dfn[v])
            {
                tarjan (v);
                low[u] = min (low[u],low[v]);
            }
            else if (!scc[v]) low[u] = min (low[u],dfn[v]);
        }
        if (low[u] == dfn[u])
        {
            ++scc_cnt;
            while (1)
            {
                int x = s.top ();s.pop ();
                scc[x] = scc_cnt;
                if (x == u) break;
            }
        }
    }
}

public:
Tarjan (int n,int m) :
    n (n),m (m),head (n + 1,0),low (n + 1,0),dfn (n + 1,0),scc (n + 1,0),
    to (2 * m + 1,0),nxt (2 * m + 1,0) {cnt = times = scc_cnt = 0;}
void add (int u,int v) // Note that the bidirectional edges
{
    to[++cnt] = v;nxt[cnt] = head[u];head[u] = cnt;
}

```

```

        to[++cnt] = u;nxt[cnt] = head[v];head[v] = cnt;
    }
    auto calc ()
    {
        for (int i = 1;i <= n;++i)
            if (!dfn[i]) tarjan (i);
        return scc;
    }
};

```

2.9 Bipartite Graph Matchings

```

class Matching
{
    int l,r;//the number of left/right side points
    vector <vector <int>> ve;
    vector <int> vis,op;
    bool dfs (int u)
    {
        for (auto v : ve[u])
        {
            if (vis[v]) continue;
            vis[v] = 1;
            if (!op[v] || dfs (op[v])) {op[v] = u;return true;}
        }
        return false;
    }
public:
    Matching (int l,int r) : l (l),r (r),vis (r + 1,0),op (r + 1,0),ve (l + 1) {}
    void add (int u,int v) {ve[u].push_back (v);}
    int calc ()
    {
        int ans = 0;
        for (int i = 1;i <= l;++i)
        {
            vis.assign (r + 1,0);
            if (dfs (i)) ++ans;
        }
        return ans;
    }
};

```

2.10 Flow

2.10.1 Edmonds-Karp

```

template <typename T>
class EK
{
    int n,m,s,t,cnt;
    vector <int> head,to,nxt,vis,pre,edge;
    vector <T> val;
    bool bfs ()
    {
        queue <T> q;
        for (int i = 1;i <= n;++i) vis[i] = 0,pre[i] = edge[i] = -1;
        vis[s] = 1;q.push (s);
        while (!q.empty ())
        {
            int u = q.front ();q.pop ();
            for (int i = head[u];i;i = nxt[i])
            {
                int v = to[i];
                if (!vis[v] && val[i])

```

```

        {
            pre[v] = u; edge[v] = i; vis[v] = 1;
            q.push (v);
            if (v == t) return 1;
        }
    }
    return 0;
}

public :
EK (int n, int m, int s, int t) :
    n (n), m (m), s (s), t (t),
    vis (n + 1, 0), head (n + 1, 0), pre (n + 1, -1), edge (n + 1, -1),
    to (m + 1, 0), nxt (m + 1, 0), val (m + 1, 0) {cnt = 1;}
void add (int u, int v, T w)
{
    to[++cnt] = v; val[cnt] = w; nxt[cnt] = head[u]; head[u] = cnt;
    to[++cnt] = u; val[cnt] = 0; nxt[cnt] = head[v]; head[v] = cnt;
}
T calc ()
{
    T ans = 0;
    while (bfs ())
    {
        T mn = INF;
        for (int i = t; i != s; i = pre[i]) mn = min (mn, val[edge[i]]);
        for (int i = t; i != s; i = pre[i]) val[edge[i]] -= mn, val[edge[i] ^ 1]
            += mn;
        ans += mn;
    }
    return ans;
}
};

```

2.10.2 Dinic

```

template <typename T>
class Dinic
{
    int n, m, s, t, cnt;
    vector <int> head, to, nxt, cur, dep;
    vector <T> val;
    int bfs ()
    {
        for (int i = 0; i <= n; ++i) dep[i] = 0, cur[i] = head[i];
        queue <int> q;
        q.push (s), dep[s] = 1;
        while (!q.empty ())
        {
            int u = q.front (); q.pop ();
            for (int i = head[u]; i; i = nxt[i])
            {
                int v = to[i];
                if (val[i] && !dep[v]) q.push (v), dep[v] = dep[u] + 1;
            }
        }
        return dep[t];
    }
    T dfs (int u, int t, T flow)
    {
        if (u == t) return flow;
        T ans = 0;
        for (int &i = cur[u]; i && ans < flow; i = nxt[i])
        {

```

```

        int v = to[i];
        if (val[i] && dep[v] == dep[u] + 1)
        {
            int x = dfs (v,t,min (val[i], flow - ans));
            if (x) val[i] -= x, val[i ^ 1] += x, ans += x;
        }
        if (ans < flow) dep[u] = -1;
        return ans;
    }
public :
Dinic (int n,int m,int s,int t) :
    n (n),m (m),s (s),t (t),
    head (n + 1,0),cur (n + 1,0),dep (n + 1,0),
    to (m + 1,0),nxt (m + 1,0),val (m + 1,0) {cnt = 1;}
void add (int u,int v,T w)
{
    to[++cnt] = v;val[cnt] = w;nxt[cnt] = head[u];head[u] = cnt;
    to[++cnt] = u;val[cnt] = 0;nxt[cnt] = head[v];head[v] = cnt;
}
T calc ()
{
    T ans = 0;
    while (bfs ())
    {
        T x;
        while ((x = dfs (s,t,INF))) ans += x;
    }
    return ans;
}
};

```

3 Data Structure

3.1 Segment Tree

3.2 Fenwick Tree

3.3 Heavy-Light Decomposition

3.4 Splay Tree

3.5 Sparse Table

3.6 Persistent Data Structure

3.7 Linear-Basis

```
template <typename T>
class Basis
{
    int n,lg;
    vector <T> p,ex,nw;
public :
    Basis (int n,int lg) : n (n),lg (lg),p (lg + 1,0),nw (lg + 1,0) {}
    void modify (T x)
    {
        for (int i = lg;~i;--i)
        {
            if (!(1ll << i) & x) continue;
            if (!p[i]) {p[i] = x;break;}
            else x ^= p[i];
        }
    }
    T get_max ()
    {
        T ans = 0;
        for (int i = lg;~i;--i)
            if ((ans ^ p[i]) > ans) ans ^= p[i];
        return ans;
    }
    T get_min ()
    {
        T ans = 0;int cnt = 0;
        for (int i = 0;i <= lg;++i) cnt += p[i] > 0;
        if (cnt < n) return 0;
        for (int i = 0;i <= lg;++i)
            if (p[i] > 0) return p[i];
    }
    void change ()
    {
        for (int i = 0;i <= lg;++i) nw[i] = p[i];
        for (int i = 0;i <= lg;++i)
            for (int j = i - 1;~j;--j)
                if (nw[i] >> j & 1) nw[i] ^= nw[j];
        for (int i = 0;i <= lg;++i)
            if (nw[i]) ex.push_back (i);
    }
    T get_min_k (T k)
    {
        int sz = (int)ex.size ();
        if (sz < n) --k; // element 0
        if (k > (1ll << sz) - 1) return -1;
        T ans = 0;
        for (int j = 0;j < sz;++j)
```

```
        if (k >> j & 1) ans ^= nw[ex[j]];
    return ans;
}
};
```

4 Math

4.1 Elementary Arithmetic Bucket

```
template <typename T,int MOD = 1000000007>
class Z
{
    T x;
    Z <T,MOD> qpow (Z x,T y)
    {
        Z <T,MOD> res (1);
        while (y)
        {
            if (y & 1) res *= x;
            x *= x;
            y >>= 1;
        }
        return res;
    }
public :
    Z () : x(0) {}
    Z (T x) : x (x) {}
    Z & operator = (T o) {x = o;return *this;}
    Z & operator += (Z o) {x = (x + o.x + MOD) % MOD;return *this;}
    Z & operator -= (Z o) {x = (x - o.x + MOD) % MOD;return *this;}
    Z & operator *= (Z o) {x = 1ll * o.x * x % MOD;return *this;}
    Z & operator ^= (Z o) {x = qpow (x,o.x).x;return *this;}
    Z & operator /= (Z o) {*this *= o ^ (MOD - 2);return *this;}
    friend Z operator + (Z x,Z y) {return x += y;}
    friend Z operator - (Z x,Z y) {return x -= y;}
    friend Z operator * (Z x,Z y) {return x *= y;}
    friend Z operator / (Z x,Z y) {return x /= y;}
    friend Z operator ^ (Z x,Z y) {return x ^= y;}
    friend bool operator == (Z x,Z y) {return x.x == y.x;}
    friend bool operator != (Z x,Z y) {return x.x != y.x;}
    friend istream& operator >> (istream& is, Z& o) {T val;is >> val;o = Z(val);
        return is;}
    friend ostream& operator << (ostream &os, const Z &z) {return os << z.x;}
};
```

4.2 Combination

```
template <typename T,int MOD = 1000000007>
class COM
{
    int n;
    vector <Z <T,MOD>> fac,inv;
public :
    COM (int n) : n (n),fac (2 * n + 1),inv (2 * n + 1)
    {
        fac[0] = inv[0] = 1;
        for (int i = 1;i <= 2 * n;++i) fac[i] = fac[i - 1] * i;
        inv[2 * n] = fac[2 * n] ^ (MOD - 2);
        for (int i = 2 * n - 1;i-->0) inv[i] = inv[i + 1] * (i + 1);
    }
    Z <T,MOD> f (int x) {return fac[x];}
    Z <T,MOD> inv_f (int x) {return inv[x];}
    Z <T,MOD> comb (int x,int y)
    {
        if (y < 0 || y > x) return 0;
        else return fac[x] * inv[x - y] * inv[y];
    }
    Z <T,MOD> arr (int x,int y)
```

```

{
    if (y > x) return 0;
    else return fac[x] * inv[x - y];
}
Z <T,MOD> catalan (int x) {return comb (2 * x,x) / (x + 1);}
};

```

4.3 The Sieve of Primes Euler's Totient Function Möbius Inversion

```

class Prime
{
    int mx;
    vector <int> p,phi,mu;
    void pre ()
    {
        vector <int> fl (mx + 1,0);
        phi[1] = mu[1] = 1;
        for (int i = 2;i <= mx;++i)
        {
            if (!fl[i]) p.push_back (i),phi[i] = i - 1,mu[i] = -1;
            for (auto v : p)
            {
                if (i * v > mx) break;
                fl[i * v] = 1;
                if (i % v == 0) {phi[i * v] = phi[i] * v,mu[i * v] = 0;break;}
                phi[i * v] = phi[i] * (v - 1);mu[i * v] = -mu[i];
            }
        }
    }
public :
    Prime (int mx) : mx (mx),phi (mx + 1,0),mu (mx + 1,0) {pre ();}
    auto get_prime () {return p;}
    auto get_phi () {return phi;}
    auto get_mu () {return mu;}
};

```

4.4 Exgcd

```

ll exgcd (ll a,ll b,ll &x,ll &y)
{
    if (!b) {x = 1;y = 0;return a;}
    ll d = exgcd (b,(a % b + b) % b,y,x);
    y -= a / b * x;
    return d;
}

```

4.5 CRT

```

ll CRT ()
{
    //ans % a[i] = b[i] if gcd (a[i],a[j]) = 1
    ll sum = 1,ans = 0;
    for (int i = 1;i <= n;++i) sum *= a[i];
    for (int i = 1;i <= n;++i)
    {
        ll x,y,tmp = sum / a[i];
        exgcd (tmp,a[i],x,y);
        ans = (ans + tmp * x * b[i]) % sum;
    }
    return (ans + sum) % sum;
}

```

4.6 Gaussian Elimination


```

vector <vector <double>> a (n + 1, vector <double> (n + 2)); // n * (n + 1)
for (int i = 1; i <= n; ++i)
{
    int p = cur, ok = 0;
    while (p <= n)
    {
        if (a[p][i] != 0) {ok = 1; break;}
        ++p;
    }
    if (!ok) continue;
    for (int j = i; j <= n + 1; ++j) swap (a[p][j], a[cur][j]);
    for (int j = n + 1; j >= i; --j) a[cur][j] /= a[cur][i];
    for (int j = 1; j <= n; ++j)
    {
        if (j == cur) continue;
        for (int k = n + 1; k >= i; --k) a[j][k] -= a[cur][k] * a[j][i];
    }
    ++cur;
}
if (cur <= n) puts ("No Solution"); //0 solution or infinite solutions
else {for (int i = 1; i <= n; ++i) printf ("%0.6lf\n", a[i][n + 1]);}

```

4.7 FFT

5 Computational Geometry

```

using LD = long double;
const LD pi = acos (-1.0);
const LD eps = 1e-8;
int dcmp (LD x) {return x < -eps ? -1 : (x > eps ? 1 : 0);}
struct Point {LD x, y; Point (LD x = 0, LD y = 0) : x (x), y (y) {};}
struct Circle {Point O; LD r; Circle (Point O = Point (), LD r = 0) : O (O), r (r) {};}
typedef Point Vector;
Vector operator + (Vector A, Vector B) {return Vector (A.x + B.x, A.y + B.y);}
Vector operator - (Vector A, Vector B) {return Vector (A.x - B.x, A.y - B.y);}
Vector operator * (Vector A, LD k) {return Vector (A.x * k, A.y * k);}
Vector operator / (Vector A, LD k) {return Vector (A.x / k, A.y / k);}

LD dot (Vector A, Vector B) {return A.x * B.x + A.y * B.y;}
LD dis (Point A, Point B) {return sqrt ((A.x - B.x) * (A.x - B.x) + (A.y - B.y) * (A.y - B.y));}
LD cross (Vector A, Vector B) {return A.x * B.y - A.y * B.x;} // A -> B counter-clockwise if cross (A, B) > 0
LD len (Point A) {return sqrt (A.x * A.x + A.y * A.y);}
LD angle (Vector A, Vector B) {return acos (dot (A, B) / (len (A) * len (B)));}
Vector proj (Vector A, Vector B) {return A * (dot (A, B) / dot (A, A));} //project onto A
Point foot (Point P, Point A, Point B) {Vector AP = P - A, AB = B - A; return A + proj (AB, AP);} //foot
Point reflect (Point P, Point A, Point B) {Point F = foot (P, A, B); return F * 2 - P;} //symmetry point
Point rotate (Point P, LD theta) {return (Point){P.x * cos (theta) - P.y * sin (theta), P.x * sin (theta) + P.y * cos (theta)}};
bool on_line (Point P, Point A, Point B) {return dcmp (cross (P - A, B - A)) == 0;}
bool on_seg (Point P, Point A, Point B) {return on_line (P, A, B) && dcmp (dot (P - A, P - B)) <= 0;} //judge whether on segment AB
LD dis_seg (Point P, Point A, Point B)
{
    if (dcmp (dot (B - A, P - A)) < 0) return dis (P, A);
    if (dcmp (dot (A - B, P - B)) < 0) return dis (P, B);
    return fabs (cross (P - A, P - B)) / dis (A, B);
}

```

```

Point inter_line (Point A,Point B,Point C,Point D) {return A + (B - A) * cross (C
- A,D - C) / cross (B - A,D - C);}
bool pd_ll_inter (Point A,Point B,Point C,Point D) {return dcmp (cross (B - A,D -
C)) != 0;} // line - line
bool pd_ls_inter (Point A,Point B,Point C,Point D) {return on_line (inter_line (A
,B,C,D),C,D);} //The intersection of AB(line) and CD (line) is on the CD (seg
).
bool pd_ss_inter (Point A,Point B,Point C,Point D) // seg - seg
{
    LD c1 = cross (B - A,C - A),c2 = cross (B - A,D - A);
    LD d1 = cross (D - C,A - C),d2 = cross (D - C,B - C);
    if (dcmp (c1) * dcmp (c2) < 0 && dcmp (d1) * dcmp (d2) < 0) return true;
    if (dcmp(c1) == 0 && on_seg (C,A,B)) return true;
    if (dcmp(c2) == 0 && on_seg (D,A,B)) return true;
    if (dcmp(d1) == 0 && on_seg (A,C,D)) return true;
    if (dcmp(d2) == 0 && on_seg (B,C,D)) return true;
    return false;
}

LD area (vector <Point> P)
{
    int n = P.size ();
    LD res = 0;
    for (int i = 0;i < n;++i) res += cross (P[i],P[(i + 1) % n]);
    return res / 2.0;
}
bool is_convex (vector <Point> P)
{
    int n = P.size ();
    for(int i = 0;i < n - 1;++i)
        if (dcmp (cross (P[i + 1] - P[i],P[(i + 2) % n] - P[i])) < 0) return
            false;
    return true;
}
int in_Poly (vector <Point> P,Point A)
{
    int cnt = 0,n = P.size ();
    for (int i = 0;i < n;++i)
    {
        int j = (i + 1) % n;
        if (on_seg (A,P[i],P[j])) return 2;// on the edge
        if (A.y >= min (P[i].y,P[j].y) && A.y < max (P[i].y,P[j].y)) // the
            intersection is on the right
            cnt += dcmp (((A.y - P[i].y) * (P[j].x - P[i].x) / (P[j].y - P[i].y)
                + P[i].x) - A.x) > 0;
    }
    return cnt & 1;
}
auto convex_hull (vector <Point> P) // strict convex hull (<= 0)
{
    int n = P.size ();
    sort (P.begin (),P.end (),[] (Point &x,Point &y) {return x.x == y.x ? x.y < y
        .y : x.x < y.x;});
    vector <Point> hull;
    hull.resize (2 * n + 1);
    int k = 0;
    for (int i = 0;i < n;++i)
    {
        while (k >= 2 && dcmp (cross (hull[k - 1] - hull[k - 2],P[i] - hull[k -
            2])) <= 0) --k;
        hull[k++] = P[i];
    }
    for (int i = n - 2,t = k;i >= 0;--i)

```

```

    {
        while (k > t && dcmp (cross (hull[k - 1] - hull[k - 2], P[i] - hull[k - 2])) <= 0) --k;
        hull[k++] = P[i];
    }
    hull.resize (k - 1);
    return hull;
}
LD diameter (vector <Point> P)
{
    int n = P.size ();
    if (n <= 1) return 0;
    if (n == 2) return len (P[1] - P[0]);
    LD res = 0;
    for (int i = 0, j = 2; i < n; ++i)
    {
        while (dcmp (cross (P[(i + 1) % n] - P[i], P[j] - P[i]) - cross (P[(i + 1) % n] - P[i], P[(j + 1) % n] - P[i])) <= 0) j = (j + 1) % n;
        res = max (res, max (len (P[j] - P[i]), len (P[j] - P[(i + 1) % n])));
    }
    return res;
}

bool in_cir (Circle C, Point P) {return dcmp (len (P - C.O) - C.r) <= 0;}
Point get_cir_p (Circle C, LD theta) {return {C.O.x + C.r * cos (theta), C.O.y + C.r * sin (theta)}};
int pd_lc_inter (Point A, Point B, Circle C)
{
    double d = dis_seg (C.O, A, B);
    if (dcmp (d - C.r) == 0) return 0; // tangent
    if (dcmp (d - C.r) > 0) return -1; // separation
    return 1; // intersection
}
int pd_cc_inter (Circle A, Circle B) // the number of tangent lines
{
    LD d = len (A.O - B.O);
    if (dcmp (A.r + B.r - d) < 0) return 4; // externally separate
    if (dcmp (A.r + B.r - d) == 0) return 3; // externally tangent
    if (dcmp (fabs (A.r - B.r) - d) == 0) return 1; // internally tangent
    if (dcmp (fabs (A.r - B.r) - d) > 0) return 0; // one circle inside the other
    return 2; // intersection
}
auto lc_inter (Point A, Point B, Circle C)
{
    Point F = foot (C.O, A, B); LD d = dis (C.O, F);
    Vector E = (B - A) / dis (A, B);
    Point P1 = F - E * sqrt (C.r * C.r - d * d);
    Point P2 = F + E * sqrt (C.r * C.r - d * d);
    return {P1, P2};
}
auto cc_inter (Circle A, Circle B)
{
    Vector k = B.O - A.O;
    LD d = len (k);
    LD alpha = atan2 (k.y, k.x), beta = acos ((A.r * A.r + d * d - B.r * B.r) / (2 * A.r * d));
    Point P1 = get_cir_p (A, alpha - beta), P2 = get_cir_p (A, alpha + beta);
    return {P1, P2};
}
auto tan_cir (Point P, Circle C)
{
    LD d = len (C.O - P), theta = asin (C.r / d);
    Vector E = (C.O - P) / d;

```

```

    Vector P1 = P + (rotate (E,theta) * sqrt (d * d - C.r * C.r));
    Vector P2 = P + (rotate (E,-theta) * sqrt (d * d - C.r * C.r));
    return {P1,P2};
}
Circle triangle_incir (Point A,Point B,Point C)
{
    LD a = dis (B,C),b = dis (A,C),c = dis (A,B);
    Point O = (A * a + B * b + C * c) / (a + b + c);
    return {O,dis_seg (O,A,B)};
}
Circle triangle_circum (Point A,Point B,Point C)
{
    LD Bx = B.x - A.x,By = B.y - A.y,Cx = C.x - A.x,Cy = C.y - A.y;
    LD D = 2 * (Bx * Cy - By * Cx);
    LD x = (Cy * (Bx * Bx + By * By) - By * (Cx * Cx + Cy * Cy)) / D + A.x;
    LD y = (Bx * (Cx * Cx + Cy * Cy) - Cx * (Bx * Bx + By * By)) / D + A.y;
    Point P (x,y);
    return Circle (P,dis (A,P));
}
auto get_tangents (Circle A,Circle B)
{
    vector <pair <Point,Point>> tangents;
    LD d = len (A.O - B.O),dif = A.r - B.r,sum = A.r + B.r;
    if (dcmp (d - fabs (dif)) < 0) return tangents;
    LD base = atan2 (B.O.y - A.O.y,B.O.x - A.O.x);
    if (dcmp (d - fabs (dif)) == 0)
    {
        tangents.push_back ({get_cir_p (A,base + (A.r < B.r ? pi : 0)),get_cir_p
            (A,base + (A.r < B.r ? pi : 0))});
        return tangents;
    }
    LD theta = acos (dif / d);
    tangents.push_back ({get_cir_p (A,base + theta),get_cir_p (B,base + theta)});
    tangents.push_back ({get_cir_p (A,base - theta),get_cir_p (B,base - theta)});
    if (dcmp (d - sum) == 0) tangents.push_back ({get_cir_p (A,base),get_cir_p (A
        ,base)});
    if (dcmp (d - sum) > 0)
    {
        theta = acos (sum / d);
        tangents.push_back ({get_cir_p (A,base + theta),get_cir_p (B,base + theta
            + pi)});
        tangents.push_back ({get_cir_p (A,base - theta),get_cir_p (B,base - theta
            + pi)});
    }
    return tangents;
}
LD tri_ploy_area (Point A,Point B,Circle C)
{
    Vector OA = A - C.O,OB = B - C.O;
    LD S = cross (OA,OB),sign = dcmp (cross (OA,OB)) > 0 ? 1 : -1;
    bool da = dcmp (len (OA) - C.r) < 0,db = dcmp (len (OB) - C.r) < 0;
    if (dcmp (S) == 0) return 0;
    if (da && db) return S * 0.5; // triangle
    if (!da && !db)
    {
        if (pd_lc_inter (A,B,C) == 1)// arc + triangle + arc
        {
            auto [P1,P2] = lc_inter (A,B,C);
            Vector OP1 = P1 - C.O,OP2 = P2 - C.O;
            if (dis (A,P1) > dis (A,P2)) swap (P1,P2);
            return cross (OP1,OP2) * 0.5 + sign * 0.5 * C.r * C.r * (angle (OA,
                OP1) + angle (OB,OP2));
        }
    }
}

```

```

        else return sign * 0.5 * C.r * C.r * angle (OA,OB); // arc
    }
    else // triangle + arc
    {
        auto [P1,P2] = lc_inter (A,B,C);
        if (on_seg (P2,A,B)) swap (P1,P2);
        Vector OP1 = P1 - C.O;
        if (dcmp (len (OA) - C.r) < 0) return cross (OA,OP1) * 0.5 + sign * 0.5 *
            C.r * C.r * angle (OP1,OB);
        else return cross (OP1,OB) * 0.5 + sign * 0.5 * C.r * C.r * angle (OP1,OA
    );
    }
}
LD cc_area (Circle C1,Circle C2)
{
    int op = pd_cc_inter (C1,C2);
    if (op <= 1) return pi * min (C1.r,C2.r) * min (C1.r,C2.r);
    else if (op == 4) return 0;
    else
    {
        LD d = dis (C1.O,C2.O);
        LD alpha = 2 * acos ((C1.r * C1.r - C2.r * C2.r + d * d) / (2 * C1.r * d)
        );
        LD beta = 2 * acos ((C2.r * C2.r - C1.r * C1.r + d * d) / (2 * C2.r * d))
        ;
        return 0.5 * (C1.r * C1.r * (alpha - sin (alpha)) + C2.r * C2.r * (beta -
            sin (beta)));
    }
}
}

```

6 String

6.1 Hash

```
template <unsigned long long base = 13331>
class Hash
{
    using u64 = unsigned long long;
    vector <u64> pw,hsh;
    Hash (char *s)
    {
        int n = strlen (s);
        pw.assign (n,0),hsh.assign (n,0);
        pw[0] = 1;hsh[0] = 0;
        for (int i = 1; i <= n;++i) pw[i] = pw[i - 1] * base,hsh[i] = hsh[i - 1]
            * base + s[i - 1];
        return hsh;
    }
public:
    u64 get (int l,int r) {return hsh[r] - hsh[l - 1] * pw[r - l + 1];}
    u64 link (int l1,int r1,int l2,int r2) {return get (l1,r1) * pw[r2 - l2 + 1]
        + get (l2,r2);}
    bool same (int l1,int r1,int l2,int r2) {return get (l1,r1) == get (l2,r2);}
};
```

6.2 KMP

```
class KMP
{
    vector <int> fail,ans;
    void getfail (char *s) // Match a with b
    {
        int len = strlen (s);
        fail.assign (len + 1,0);
        fail[0] = -1;
        for (int i = 1;i < len;++i)
        {
            int cnt = fail[i - 1];
            while (cnt >= 0 && s[cnt + 1] != s[i]) cnt = fail[cnt];
            if (s[cnt + 1] == s[i]) ++cnt;
            fail[i] = cnt;
        }
    }
public:
    auto get_pos (char *s,char *t)
    {
        getfail (t);
        int lens = strlen (s),lent = strlen (t),cnt = -1;
        for (int i = 0;i < lens;++i)
        {
            while (cnt >= 0 && t[cnt + 1] != s[i]) cnt = fail[cnt];
            if (t[cnt + 1] == s[i])
            {
                ++cnt;
                if (cnt + 1 == lent) ans.push_back (i - lent + 2),cnt = fail[cnt];
            }
        }
        return ans;
    }
};
```

6.3 Manacher

```

int Manacher (char *s)
{
    int n = strlen (s), cnt = 0, r = 0, mid = 0, ans = 0;
    vector <char> a (2 * n + 5); vector <int> p (2 * n + 5);
    a[++cnt] = '!'; a[++cnt] = '#';
    for (int i = 0; i < n; ++i) a[++cnt] = s[i], a[++cnt] = '#';
    a[++cnt] = '~';
    for (int i = 2; i < cnt; ++i)
    {
        if (i <= r) p[i] = min (r - i + 1, p[mid * 2 - i]);
        else p[i] = 1;
        while (a[i - p[i]] == a[i + p[i]]) ++p[i];
        if (i + p[i] > r) r = i + p[i] - 1, mid = i;
        ans = max (ans, p[i]);
    }
    return ans - 1;
}

```

6.4 Trie

6.4.1 Trie

```

struct Trie
{
    int n, m, cnt; // m total len
    vector <vector <int>> ch;
    vector <int> vis;
public :
    Trie (int n, int m) : n (n), m (m), ch (m, vector <int> (26, 0)), vis (n) {cnt = 0;}
    void insert (char *s)
    {
        int u = 1, len = strlen (s + 1);
        for (int i = 1; i <= len; ++i)
        {
            int c = s[i] - 'a';
            if (!ch[u][c]) ch[u][c] = ++cnt;
            u = ch[u][c];
        }
        ++vis[u];
    }
    int query (char *s)
    {
        int u = 1, len = strlen (s + 1);
        for (int i = 1; i <= len; ++i)
        {
            int c = s[i] - 'a';
            if (!ch[u][c]) return 0;
            u = ch[u][c];
        }
        return vis[u];
    }
};

```

6.4.2 01-Trie

```

class Trie
{
    int n, cnt;
    vector <vector <int>> ch;
    vector <int> val, w;
public :
    Trie (int n, int lg) : n (n), val (2 * lg * n + 1, 0), w (2 * lg * n + 1, 0), ch (2 * lg * n + 1, vector <int> (2, 0)) {cnt = 1;}
}

```

```

void pushup (int u)
{
    w[u] = val[u] = 0;
    //w[u] Number of values (weights) on the edge between node u and its
    //parent node
    //val[u] XOR sum maintained by the subtree rooted at u
    if (ch[u][0]) w[u] ^= w[ch[u][0]], val[u] ^= val[ch[u][0]] << 1;
    if (ch[u][1]) w[u] ^= w[ch[u][1]], val[u] ^= (val[ch[u][1]] << 1) | w[ch[u]
        ][1]];
}
void modify (int &u,int v,int dep)
{
    if (!u) u = ++cnt;
    w[u] ^= 1;
    if (dep < 0) return ;
    modify (ch[u][v & 1],v >> 1,dep - 1);
    pushup (u);
}
void erase (int u,int v,int dep)
{
    if (!u) return ;
    w[u] ^= 1;
    if (dep < 0) return ;
    erase (ch[u][v & 1],v >> 1,dep - 1);
    pushup (u);
}
void add (int u) // add 1 in [1,n]
{
    swap (ch[u][0],ch[u][1]);
    if (ch[u][0]) add (ch[u][0]);
    pushup (u);
}
};

```

6.5 Aho-Corasick Automaton

6.6 SA/SAM

```

class SAM
{
    class node
    {
        public:int ch[26],len,fa;
        node (const int &L = 0) {memset (ch,0,sizeof (ch));fa = 0;len = L;}
    };
    public:
    vector <node> t;int lst;
    void GetParentTree (vector <vector <int>> &G)
    {
        G.resize (t.size ());
        for (unsigned i = 1;i < t.size ();++i) G[t[i].fa].push_back (i);
    }
    void extend (const int &c)
    {
        int p = lst,np = lst = t.size ();
        t.push_back (node (t[p].len + 1));
        for (;p&&!t[p].ch[c];p = t[p].fa) t[p].ch[c] = np;
        if (!p) t[np].fa = 1;
        else
        {
            int v = t[p].ch[c];
            if (t[v].len == t[p].len + 1) t[np].fa = v;
            else
            {

```



```

        int nv = t.size ();t.push_back (t[v]);
        t[nv].len = t[p].len + 1;
        for (;p && t[p].ch[c] == v;p = t[p].fa) t[p].ch[c] = nv;
        t[np].fa = t[v].fa = nv;
    }
}
SAM () {t.assign (2,node ());lst = 1;}
inline void clear () {t.assign (2,node ());lst = 1;}
inline int next (int p,int c) {return t[p].ch[c];}
inline int Len (int p) {return t[p].len;}
};

```

6.7 Border/Fail Tree