

# Contents

<b>1 Basic</b>	
1.1 Default code	
1.2 Linux 對拍	
1.3 Windows 對拍	
1.4 builtin 函數	
1.5 輸入輸出	
1.6 Python 輸入輸出	
<b>2 Data Structure</b>	
2.1 Link-Cut Tree	
2.2 持久化線段樹	
2.3 Treap	
2.4 線段樹	
<b>3 Flow</b>	
3.1 Dinic	
3.2 匈牙利	
3.3 KM	
3.4 MCMF	
<b>4 幾何</b>	
4.1 點宣告	
4.2 矩形面積	
4.3 最近點對	
4.4 凸包	
4.5 兩直線交點	
4.6 兩線段交點	
4.7 李超線段樹	
4.8 最小包圍圓	
4.9 最小包圍球	
4.10 旋轉卡尺	
4.11 Circle Cover	
4.12 Convex Hull Trick	
4.13 Half Plane Intersection	
<b>5 圖論</b>	
5.1 BCC	
5.2 重心剖分	
5.3 輕重鍊剖分	
5.4 歐拉路徑	
5.5 極大團	
5.6 最大團	
5.7 SCC	
5.8 SPFA	
5.9 差分約束	
<b>6 數論</b>	
6.1 離散根號	
6.2 ex-crt	
6.3 ex-gcd	
6.4 FFT	
6.5 高斯消去法	
6.6 喬瑟夫問題	
6.7 定理	
6.8 Miller Rabin	
6.9 NTT	
6.10 Pollard's Rho	
6.11 質數	
6.12 phi	
6.13 矩陣快速冪	
<b>7 字串</b>	
7.1 KMP	
7.2 馬拉車	
7.3 回文樹	
7.4 SA	
7.5 SAM	
7.6 樹哈希	
7.7 trie	
7.8 Z-value	
7.9 minRotation	
<b>8 DP</b>	
8.1 數位 dp	
8.2 SOS dp	
8.3 p-median	
<b>9 Other</b>	
9.1 黑魔法、名次樹	
9.2 Hiber Curve	

## 1 Basic

### 1.1 Default code

```

1 #include<bits/stdc++.h>
1 #define int long long
1 #define mod 1000000007
1 #define endl '\n'
1 #define pii pair<int,int>
1 using namespace std;
2
2 signed main(){
2     ios::sync_with_stdio(0),cin.tie(0);
2 }

```

### 1.2 Linux 對拍

```

3
3 set -e
4 for ((i=0;i<300;i++))
4 do
5     echo "$i"
5     python3 gen.py > input
5     ./ac < input > ac.out
5     ./wa < input > wa.out
6     diff ac.out wa.out || break
6 done

```

### 1.3 Windows 對拍

```

7 @echo off
7 :loop
7     echo %%x
7     python gen.py > input
7     ./ac.exe < input > ac.out
8     ./wa.exe < input > wa.out
9     fc ac.out wa.out
9 if not errorlevel 1 goto loop

```

### 1.4 builtin 函數

```

10 // 右邊第一個 1 的位置
10 int __builtin_ffs(unsigned int);
10 int __builtin_ffsl(unsigned long);
11 int __builtin_ffsll(unsigned long long);
11 // 左邊第一個 1 之前 0 的數量
11 int __builtin_clz(unsigned int);
11 int __builtin_clzl(unsigned long);
11 int __builtin_clzll(unsigned long long);
12 // 右邊第一個 1 之後 0 的數量
12 int __builtin_ctz(unsigned int);
12 int __builtin_ctzl(unsigned long);
12 int __builtin_ctzll(unsigned long long);
13 // 1 的數量
13 int __builtin_popcount(unsigned int);
13 int __builtin_popcountl(unsigned long);
13 int __builtin_popcountll(unsigned long long);
14 // 1 的數量 mod 2
14 int __builtin_parity(unsigned int);
14 int __builtin_parityl(unsigned long);
14 int __builtin_parityll(unsigned long long);
14 // 二進制表示數字
14 int a = 0b101101;

```

### 1.5 輸入輸出

```

15 // 開讀檔
15 fopen("input_file_name","r",stdin);
16 fopen("output_file_name","w",stdout);

```

### 1.6 Python 輸入輸出

```

16 a = list(map(int,input().split()))
16
16 # 開讀檔
16 import sys, os.path
16 if(os.path.exists('input_file.txt')):
16     sys.stdin = open("input_file.txt","r")
16     sys.stdout = open("output_file.txt","w")

```

## 2 Data Structure

### 2.1 Link-Cut Tree

```

struct Splay {
    static Splay nil, mem[MEM], *pmem;
    Splay *ch[2], *f;
    int val, rev, size;
    Splay (int _val=-1) : val(_val), rev(0), size(1)
    { f = ch[0] = ch[1] = &nil; }
    bool isr()
    { return f->ch[0] != this && f->ch[1] != this; }
    int dir()
    { return f->ch[0] == this ? 0 : 1; }
    void setCh(Splay *c, int d){
        ch[d] = c;
        if (c != &nil) c->f = this;
        pull();
    }
    void push(){
        if( !rev ) return;
        swap(ch[0], ch[1]);
        if (ch[0] != &nil) ch[0]->rev ^= 1;
        if (ch[1] != &nil) ch[1]->rev ^= 1;
        rev=0;
    }
    void pull(){
        size = ch[0]->size + ch[1]->size + 1;
        if (ch[0] != &nil) ch[0]->f = this;
        if (ch[1] != &nil) ch[1]->f = this;
    }
} Splay::nil, Splay::mem[MEM], *Splay::pmem = Splay::
mem;
Splay *nil = &Splay::nil;
void rotate(Splay *x){
    Splay *p = x->f;
    int d = x->dir();
    if (!p->isr()) p->f->setCh(x, p->dir());
    else x->f = p->f;
    p->setCh(x->ch[!d], d);
    x->setCh(p, !d);
    p->pull(); x->pull();
}
vector<Splay*> splayVec;
void splay(Splay *x){
    splayVec.clear();
    for (Splay *q=x;; q=q->f){
        splayVec.push_back(q);
        if (q->isr()) break;
    }
    reverse(begin(splayVec), end(splayVec));
    for (auto it : splayVec) it->push();
    while (!x->isr()) {
        if (x->f->isr()) rotate(x);
        else if (x->dir()==x->f->dir())
            rotate(x->f), rotate(x);
        else rotate(x), rotate(x);
    }
}
int id(Splay *x) { return x - Splay::mem + 1; }
Splay* access(Splay *x){
    Splay *q = nil;
    for (;x!=nil;x=x->f){
        splay(x);
        x->setCh(q, 1);
        q = x;
    }
    return q;
}
void chroot(Splay *x){
    access(x);
    splay(x);
    x->rev ^= 1;
    x->push(); x->pull();
}
void link(Splay *x, Splay *y){
    access(x);
    splay(x);
    chroot(y);
    x->setCh(y, 1);
}
void cut_p(Splay *y) {

```

```

    access(y);
    splay(y);
    y->push();
    y->ch[0] = y->ch[0]->f = nil;
}
void cut(Splay *x, Splay *y){
    chroot(x);
    cut_p(y);
}
Splay* get_root(Splay *x) {
    access(x);
    splay(x);
    for(; x->ch[0] != nil; x = x->ch[0])
        x->push();
    splay(x);
    return x;
}
bool conn(Splay *x, Splay *y) {
    x = get_root(x);
    y = get_root(y);
    return x == y;
}
Splay* lca(Splay *x, Splay *y) {
    access(x);
    access(y);
    splay(x);
    if (x->f == nil) return x;
    else return x->f;
}
}

```

### 2.2 持久化線段樹

```

struct Seg{
    struct Node{
        int v;
        Node* l,*r;
    };
    vector<Node*> version;
    Node* build(int l,int r){
        Node* node=new Node;
        if(l==r){
            node->v=l;
            return node;
        }
        int mid=(l+r)/2;
        node->l=build(l,mid);
        node->r=build(mid+1,r);
        return node;
    }
    int query(Node* cur,int l,int r,int x){
        if(l==r){
            return cur->v;
        }
        int mid=(l+r)/2;
        if(x<=mid) return query(cur->l,l,mid,x);
        else return query(cur->r,mid+1,r,x);
    }
    Node* update(Node* cur,int l,int r,int x,int y){
        Node* node=new Node;
        if(l==r){
            node->v=y;
            return node;
        }
        int mid=(l+r)/2;
        if(x<=mid){
            node->l=update(cur->l,l,mid,x,y);
            node->r=cur->r;
        }
        else{
            node->l=cur->l;
            node->r=update(cur->r,mid+1,r,x,y);
        }
        return node;
    }
}
};

```

### 2.3 Treap

```

mt19937 gen(chrono::steady_clock::now().
    time_since_epoch().count()); // C++ randomizer
struct Node {
    int k, p, sz = 1;

```

```

Node *l = 0, *r = 0;
bool tag = 0;
Node(int kk) {
    k = kk;
    p = gen();
}
};
Node *root = 0;
int size(Node *x) {return x ? x->sz : 0;}
void push(Node *x) {
    if(x->tag) {
        if(x->l) x->l->tag ^= true;
        if(x->r) x->r->tag ^= true;
        x->tag = false;
    }
}
void pull(Node* x) {
    x->sz = size(x->l) + size(x->r) + 1;
}
Node* merge(Node *a, Node *b) {
    if(!a || !b) return a ? b;
    if(a->p > b->p) {
        push(a);
        a->r = merge(a->r, b);
        pull(a);
        return a;
    }
    else{
        push(b);
        b->l = merge(a, b->l);
        pull(b);
        return b;
    }
}
void splitKey(Node* x, int k, Node *&a, Node *&b) {
    if(!x) {a = b = 0; return;}
    push(x);
    if(x->k <= k) {
        a = x;
        splitKey(a->r, k, a->r, b);
        pull(a);
    }
    else{
        b = x;
        splitKey(b->l, k, a, b->l);
        pull(b);
    }
}
void splitKth(Node *x, int k, Node *&a, Node *&b) {
    if(!x) {a = b = 0; return;}
    push(x);
    if(size(x->l) < k) {
        a = x;
        splitKth(a->r, k - size(x->l) - 1, a->r, b);
        pull(a);
    }
    else{
        b = x;
        splitKth(b->l, k, a, b->l);
        pull(b);
    }
}
void insert(int id) {
    Node *l, *r;
    splitKey(root, id, l, r);
    Node *m = new Node(id);
    root = merge(l, merge(m, r));
}
void erase(int x) {
    Node *a, *b, *c;
    splitKey(root, x, b, c);
    splitKey(b, x - 1, a, b);
    root = merge(a, c);
}
}

```

## 2.4 線段樹

```

struct Seg{
    vector<int> seg,tag;
    #define cl (i<<1)+1
    #define cr (i<<1)+2
    void push(int i,int l,int r){

```

```

        if(tag[i]!=0){
            seg[i]+=tag[i]; // update by tag
            if(l!=r){
                tag[cl]+=tag[i]; // push
                tag[cr]+=tag[i]; // push
            }
            tag[i]=0;
        }
    }
}
void pull(int i,int l,int r){
    int mid=(l+r)>>1;
    push(cl,l,mid);push(cr,mid+1,r);
    seg[i]=max(seg[cl],seg[cr]); // pull
}
void build(int i,int l,int r,vector<int>&arr){
    if(l==r){
        seg[i]=arr[l]; // set value
        return;
    }
    int mid=(l+r)>>1;
    build(cl,l,mid,arr);
    build(cr,mid+1,r,arr);
    pull(i,l,r);
}
Seg(vector<int>& arr){
    seg.resize(arr.size()*4);
    tag.resize(arr.size()*4);
    build(0,0,arr.size()-1,arr);
}
void update(int i,int l,int r,int nl,int nr,int x){
    push(i,l,r);
    if(nl<=l&&r<=nr){
        tag[i]+=x;
        return;
    }
    int mid=(l+r)>>1;
    if(nl<=mid) update(cl,l,mid,nl,nr,x);
    if(nr>mid) update(cr,mid+1,r,nl,nr,x);
    pull(i,l,r);
}
int query(int i,int l,int r,int nl,int nr){
    push(i,l,r);
    if(nl<=l&&r<=nr){
        return seg[i];
    }
    int mid=(l+r)>>1;
    int ans=0;
    if(nl<=mid) ans=max(ans,query(cl,l,mid,nl,nr));
    if(nr>mid) ans=max(ans,query(cr,mid+1,r,nl,nr));
    return ans;
}
};

```

## 3 Flow

### 3.1 Dinic

```

const int MXN=1000;
struct Dinic
{
    struct Edge
    {
        int v, f, re;
    };
    int n, s, t, level[MXN];
    vector<Edge> E[MXN];
    void init(int _n, int _s, int _t)
    {
        n = _n;
        s = _s;
        t = _t;
        for (int i = 0; i < n; i++)
            E[i].clear();
    }
    void addEdge(int u, int v, int f)
    {
        E[u].push_back({v, f, (int)(E[v].size())});
        E[v].push_back({u, 0, (int)(E[u].size()-1)});
    }
    bool BFS()
    {

```

```

for (int i = 0; i < n; i++)
    level[i] = -1;
queue<int> que;
que.push(s);
level[s] = 0;
while (!que.empty())
{
    int u = que.front();
    que.pop();
    for (auto it : E[u])
    {
        if (it.f > 0 && level[it.v] == -1)
        {
            level[it.v] = level[u] + 1;
            que.push(it.v);
        }
    }
}
return level[t] != -1;
}
int DFS(int u, int nf)
{
    if (u == t)
        return nf;
    int res = 0;
    for (auto &it : E[u])
    {
        if (it.f > 0 && level[it.v] == level[u] + 1)
        {
            int tf = DFS(it.v, min(nf, it.f));
            res += tf;
            nf -= tf;
            it.f -= tf;
            E[it.v][it.re].f += tf;
            if (nf == 0)
                return res;
        }
    }
    if (!res)
        level[u] = -1;
    return res;
}
int flow(int res = 0)
{
    while (BFS())
        res += DFS(s, 2147483647);
    return res;
}
} flow;

```

### 3.2 匈牙利

```

#define NIL -1
#define INF 100000000
int n, matched;
int cost[MAXN][MAXN];
bool sets[MAXN]; // whether x is in set S
bool sett[MAXN]; // whether y is in set T
int xlabel[MAXN], ylabel[MAXN];
int xy[MAXN], yx[MAXN]; // matched with whom
int slack[MAXN]; // given y: min{xlabel[x]+ylabel[y]-cost[x][y]} | x not in S
int prev[MAXN]; // for augmenting matching
inline void relabel() {
    int i, delta = INF;
    for (i = 0; i < n; i++) if (!sett[i]) delta = min(slack[i], delta);
    for (i = 0; i < n; i++) if (sets[i]) xlabel[i] -= delta;
    for (i = 0; i < n; i++) {
        if (sett[i]) ylabel[i] += delta;
        else slack[i] -= delta;
    }
}
inline void add_sets(int x) {
    int i;
    sets[x] = 1;
    for (i = 0; i < n; i++) {
        if (xlabel[x] + ylabel[i] - cost[x][i] < slack[i]) {
            slack[i] = xlabel[x] + ylabel[i] - cost[x][i];
            prev[i] = x;
        }
    }
}

```

```

}
inline void augment(int final) {
    int x = prev[final], y = final, tmp;
    matched++;
    while (1) {
        tmp = xy[x]; xy[x] = y; yx[y] = x; y = tmp;
        if (y == NIL) return;
        x = prev[y];
    }
}
inline void phase() {
    int i, y, root;
    for (i = 0; i < n; i++) { sets[i] = sett[i] = 0; slack[i] = INF; }
    for (root = 0; root < n && xy[root] != NIL; root++)
        add_sets(root);
    while (1) {
        relabel();
        for (y = 0; y < n; y++) if (!sett[y] && slack[y] == 0) break;
        if (yx[y] == NIL) { augment(y); return; }
        else { add_sets(yx[y]); sett[y] = 1; }
    }
}
inline int hungarian() {
    int i, j, c = 0;
    for (i = 0; i < n; i++) {
        xy[i] = yx[i] = NIL;
        xlabel[i] = ylabel[i] = 0;
        for (j = 0; j < n; j++) xlabel[i] = max(cost[i][j], xlabel[i]);
    }
    for (i = 0; i < n; i++) phase();
    for (i = 0; i < n; i++) c += cost[i][xy[i]];
    return c;
}
}

```

### 3.3 KM

```

struct KM { // max weight, for min negate the weights
    int n, mx[MAXN], my[MAXN], pa[MAXN];
    ll g[MAXN][MAXN], lx[MAXN], ly[MAXN], sy[MAXN];
    bool vx[MAXN], vy[MAXN];
    void init(int _n) { // 1-based
        n = _n;
        for (int i = 1; i <= n; i++) fill(g[i], g[i] + n + 1, 0);
    }
    void addEdge(int x, int y, ll w) { g[x][y] = w; }
    void augment(int y) {
        for (int x, z; y; y = z)
            x = pa[y], z = mx[x], my[y] = x, mx[x] = y;
    }
    void bfs(int st) {
        for (int i = 1; i <= n; ++i) sy[i] = INF, vx[i] = vy[i] = 0;
        queue<int> q; q.push(st);
        for (;;) {
            while (q.size()) {
                int x = q.front(); q.pop(); vx[x] = 1;
                for (int y = 1; y <= n; ++y) if (!vy[y]) {
                    ll t = lx[x] + ly[y] - g[x][y];
                    if (t == 0) {
                        pa[y] = x;
                        if (!my[y]) { augment(y); return; }
                        vy[y] = 1, q.push(my[y]);
                    } else if (sy[y] > t) pa[y] = x, sy[y] = t;
                }
            }
            ll cut = INF;
            for (int y = 1; y <= n; ++y)
                if (!vy[y] && cut > sy[y]) cut = sy[y];
            for (int j = 1; j <= n; ++j) {
                if (vx[j]) lx[j] -= cut;
                if (vy[j]) ly[j] += cut;
                else sy[j] -= cut;
            }
            for (int y = 1; y <= n; ++y) if (!vy[y] && sy[y] == 0) {
                if (!my[y]) { augment(y); return; }
                vy[y] = 1, q.push(my[y]);
            }
        }
    }
    ll solve() {
        fill(mx, mx + n + 1, 0); fill(my, my + n + 1, 0);
        fill(ly, ly + n + 1, 0); fill(lx, lx + n + 1, -INF);
        for (int x = 1; x <= n; ++x) for (int y = 1; y <= n; ++y)
            lx[x] = max(lx[x], g[x][y]);
        for (int x = 1; x <= n; ++x) bfs(x);
    }
}

```

```

    ll ans = 0;
    for(int y=1; y<=n; ++y) ans += g[my[y]][y];
    return ans;
} }graph;

```

### 3.4 MCMF

```

struct MCMF {
    #define SZ(x) (int)(x.size())
    struct Edge {
        int v, f, re, c;
    };
    vector<vector<Edge>> E;
    vector<int> dis, x, y;
    int n, s, t;
    MCMF(int nn, int ss, int tt) {
        n = nn; s = ss; t = tt;
        E.resize(n);
        x.resize(n);
        y.resize(n);
    }
    void addEdge(int u, int v, int w, int c) {
        E[u].push_back({v, w, SZ(E[v]), c});
        E[v].push_back({u, 0, SZ(E[u]) - 1, -c});
    }
    bool spfa() {
        dis.assign(n, 0x3f3f3f3f);
        x.assign(n, -1);
        y.assign(n, -1);
        vector<bool> inq(n, false);
        queue<int> q;
        q.push(s);
        inq[s] = true;
        dis[s] = 0;
        while(q.size()) {
            int u = q.front(); q.pop();
            inq[u] = false;
            for(int i = 0; i < E[u].size(); i++) {
                auto& it = E[u][i];
                int v = it.v;
                if(it.f > 0 && dis[v] > dis[u] + it.c) {
                    dis[v] = dis[u] + it.c;
                    x[v] = u;
                    y[v] = i;
                    if(!inq[v]) {
                        q.push(v);
                        inq[v] = true;
                    }
                }
            }
        }
        return x[t] != -1;
    }
    pii solve() {
        int mf = 0, mc = 0;
        while(spfa()) {
            int nf = 0x3f3f3f3f;
            for(int i = t; i != s; i = x[i]) {
                nf = min(nf, E[x[i]][y[i]].f);
            }
            for(int i = t; i != s; i = x[i]) {
                auto& it = E[x[i]][y[i]];
                it.f -= nf;
                E[it.v][it.re].f += nf;
            }
            mf += nf;
            mc += nf * dis[t];
        }
        return {mf, mc};
    }
};

```

## 4 幾何

### 4.1 點宣告

```

typedef long double ld;
const ld eps = 1e-8;
int dcmp(ld x) {
    if(abs(x) < eps) return 0;
    else return x < 0 ? -1 : 1;
}

```

```

}
struct Pt {
    ld x, y;
    Pt(ld _x=0, ld _y=0):x(_x), y(_y) {}
    Pt operator+(const Pt &a) const {
        return Pt(x+a.x, y+a.y);
    }
    Pt operator-(const Pt &a) const {
        return Pt(x-a.x, y-a.y);
    }
    Pt operator*(const ld &a) const {
        return Pt(x*a, y*a);
    }
    Pt operator/(const ld &a) const {
        return Pt(x/a, y/a);
    }
    ld operator*(const Pt &a) const {
        return x*a.x + y*a.y;
    }
    ld operator^(const Pt &a) const {
        return x*a.y - y*a.x;
    }
    bool operator<(const Pt &a) const {
        return x < a.x || (x == a.x && y < a.y);
    }
    //return dcmp(x-a.x) < 0 || (dcmp(x-a.x) == 0 &&
    //    dcmp(y-a.y) < 0);
    bool operator==(const Pt &a) const {
        return dcmp(x-a.x) == 0 && dcmp(y-a.y) == 0;
    }
};
ld norm2(const Pt &a) {
    return a*a;
}
ld norm(const Pt &a) {
    return sqrt(norm2(a));
}
Pt perp(const Pt &a) {
    return Pt(-a.y, a.x);
}
Pt rotate(const Pt &a, ld ang) {
    return Pt(a.x*cos(ang)-a.y*sin(ang), a.x*sin(ang)+a.y*cos(ang));
}
struct Line {
    Pt s, e, v; // start, end, end-start
    ld ang;
    Line(Pt _s=Pt(0, 0), Pt _e=Pt(0, 0)):s(_s), e(_e) { v = e-s; ang = atan2(v.y, v.x); }
    bool operator<(const Line &l) const {
        return ang < l.ang;
    }
};
struct Circle {
    Pt o; ld r;
    Circle(Pt _o=Pt(0, 0), ld _r=0):o(_o), r(_r) {}
};

```

### 4.2 矩形面積

```

struct AreaofRectangles{
    #define cl(x) (x<<1)
    #define cr(x) (x<<1|1)
    ll n, id, sid;
    pair<ll,ll> tree[MXN<<3]; // count, area
    vector<ll> ind;
    tuple<ll,ll,ll,ll> scan[MXN<<1];
    void pull(int l, int r) {
        if(tree[l].first) tree[l].second = ind[r+1] - ind[l];
        else if(l != r) {
            int mid = (l+r)>>1;
            tree[l].second = tree[cl(l)].second + tree[cr(l)].second;
        }
        else tree[l].second = 0;
    }
    void upd(int l, int r, int ql, int qr, int v) {
        if(ql <= l && r <= qr) {
            tree[l].first += v;
            pull(l, r); return;
        }
        int mid = (l+r) >> 1;
        if(ql <= mid) upd(cl(l), l, mid, ql, qr, v);
        if(qr > mid) upd(cr(l), mid+1, r, ql, qr, v);
        pull(l, r);
    }
    void init(int _n) {
        n = _n; id = sid = 0;
        ind.clear(); ind.resize(n<<1);
        fill(tree, tree+(n<<2), make_pair(0, 0));
    }
    void addRectangle(int lx, int ly, int rx, int ry) {
        ind[id++] = lx; ind[id++] = rx;
    }
}

```

```

        scan[sid++] = make_tuple(ly, 1, lx, rx);
        scan[sid++] = make_tuple(ry, -1, lx, rx);
    }
    ll solve(){
        sort(ind.begin(), ind.end());
        ind.resize(unique(ind.begin(), ind.end()) - ind
            .begin());
        sort(scan, scan + sid);
        ll area = 0, pre = get<0>(scan[0]);
        for(int i = 0; i < sid; i++){
            auto [x, v, l, r] = scan[i];
            area += tree[1].second * (x-pre);
            upd(1, 0, ind.size()-1, lower_bound(ind
                .begin(), ind.end(), l)-ind.begin(),
                lower_bound(ind.begin(), ind.end(), r)-
                ind.begin()-1, v);
            pre = x;
        }
        return area;
    }
} }rect;

```

### 4.3 最近點對

```

#include<bits/stdc++.h>
#define int long long
using namespace std;
using ld = long double;
const int mod = 1e9+7;
struct pt{
    int x,y;
    int id;
    ld dis(const pt& rhs){
        return sqrt((x-rhs.x)*(x-rhs.x)+(y-rhs.y)*(y-
            rhs.y));
    }
};
signed main(){
    int n;
    cin>>n;
    vector<pt> a(n);
    for(int i=0;i<n;i++){
        cin>>a[i].x>>a[i].y;
        a[i].id=i;
    }
    ld ans = 1e19;
    sort(a.begin(),a.end(),[](const pt&a,const pt&b){
        if(a.x==b.y)return a.y<b.y;
        return a.x<b.x;
    });
    pt ans2;
    function<void(int,int)> dng = [&](int l,int r){
        if(r-l<4){
            for(int i=l;i<=r;i++){
                for(int j=i+1;j<=r;j++){
                    ld temans = a[i].dis(a[j]);
                    if(temans<ans){
                        ans=temans;
                        ans2 = {a[i].id,a[j].id};
                    }
                }
            }
            sort(a.begin()+l,a.begin()+r+1,[](const pt&
                a,const pt&b){return a.y<b.y;});
            return;
        }
        int mid = (l+r)/2;
        int midx = a[mid].x;
        dng(l,mid);dng(mid+1,r);
        inplace_merge(a.begin()+l,a.begin()+mid+1,a
            .begin()+r+1,[](const pt&a,const pt&b){
                return a.y<b.y;});
        vector<int> c;c.reserve(r-l+1);
        for(int i=l;i<=r;i++){
            if(abs(a[i].x-midx)<ans){
                for(int j=c.size()-1;j>=0&&a[i].y-a[c[j]
                    ]>ans;j--){
                    ld temans = a[i].dis(a[c[j]]);
                    if(temans<ans){
                        ans=temans;
                        ans2 = {a[i].id,a[c[j]].id
                            };
                    }
                }
            }
        }
    };
    dng(0,n-1);
    cout<<min(ans2.x,ans2.y)<<' '<<max(ans2.x,ans2.y)<<
        ' '<<fixed<<setprecision(6)<<ans<<'\n';
}

```

```

        }
    }
    c.push_back(i);
}
};
dng(0,n-1);
cout<<min(ans2.x,ans2.y)<<' '<<max(ans2.x,ans2.y)<<
    ' '<<fixed<<setprecision(6)<<ans<<'\n';
}

```

### 4.4 凸包

```

double cross(Pt o, Pt a, Pt b){
    return (a-o) ^ (b-o);
}
vector<Pt> convex_hull(vector<Pt> pt){
    sort(pt.begin(),pt.end());
    int top=0;
    vector<Pt> stk(2*pt.size());
    for (int i=0; i<(int)pt.size(); i++){
        while (top >= 2 && cross(stk[top-2],stk[top-1],pt[i]
            ]) <= 0)
            top--;
        stk[top++] = pt[i];
    }
    for (int i=pt.size()-2, t=top+1; i>=0; i--){
        while (top >= t && cross(stk[top-2],stk[top-1],pt[i]
            ]) <= 0)
            top--;
        stk[top++] = pt[i];
    }
    stk.resize(top-1);
    return stk;
}

```

### 4.5 兩直線交點

```

Pt LLIntersect(Line a, Line b) {
    Pt p1 = a.s, p2 = a.e, q1 = b.s, q2 = b.e;
    ld f1 = (p2-p1)^(q1-p1),f2 = (p2-p1)^(p1-q2),f;
    if(dcmp(f=f1+f2) == 0)
        return dcmp(f1)?Pt(NAN,NAN):Pt(INFINITY,INFINITY);
    return q1*(f2/f) + q2*(f1/f);
}

```

### 4.6 兩線段交點

```

int ori( const Pt& o , const Pt& a , const Pt& b ){
    LL ret = ( a - o ) ^ ( b - o );
    return (ret > 0) - (ret < 0);
}
// p1 == p2 || q1 == q2 need to be handled
bool banana( const Pt& p1 , const Pt& p2 ,
    const Pt& q1 , const Pt& q2 ){
    if( ( ( p2 - p1 ) ^ ( q2 - q1 ) ) == 0 ) // parallel
        if( ori( p1 , p2 , q1 ) ) return false;
    return ( ( p1 - q1 ) * ( p2 - q1 ) ) <= 0 ||
        ( ( p1 - q2 ) * ( p2 - q2 ) ) <= 0 ||
        ( ( q1 - p1 ) * ( q2 - p1 ) ) <= 0 ||
        ( ( q1 - p2 ) * ( q2 - p2 ) ) <= 0;
}
return (ori( p1, p2, q1 ) * ori( p1, p2, q2 )<=0) &&
    (ori( q1, q2, p1 ) * ori( q1, q2, p2 )<=0);
}

```

### 4.7 李超線段樹

```

struct LiChao_min{
    struct line{
        ll m,c;
        line(ll _m=0,ll _c=0){ m=_m; c=_c; }
        ll eval(ll x){ return m*x+c; } // overflow
    };
    struct node{
        node *l,*r; line f;
        node(line v){ f=v; l=r=NULL; }
    };
    typedef node* pnode;
    pnode root; ll sz,ql,qr;
#define mid ((l+r)>>1)
    void insert(line v,ll l,ll r,pnode &nd){

```



```

/* if(!(ql<=l&&r<=qr)){
    if(!nd) nd=new node(line(0,INF));
    if(ql<=mid) insert(v,l,mid,nd->l);
    if(qr>mid) insert(v,mid+1,r,nd->r);
    return;
} used for adding segment */
if(!nd){ nd=new node(v); return; }
ll trl=nd->f.eval(l),trr=nd->f.eval(r);
ll vl=v.eval(l),vr=v.eval(r);
if(trl<=vl&&trr<=vr) return;
if(trl>vl&&trr>vr) { nd->f=v; return; }
if(trl>vl) swap(nd->f,v);
if(nd->f.eval(mid)<v.eval(mid))
    insert(v,mid+1,r,nd->r);
else swap(nd->f,v),insert(v,l,mid,nd->l);
}
ll query(ll x,ll l,ll r,pnode &nd){
    if(!nd) return INF;
    if(l==r) return nd->f.eval(x);
    if(mid>=x)
        return min(nd->f.eval(x),query(x,l,mid,nd->l));
    return min(nd->f.eval(x),query(x,mid+1,r,nd->r));
}
/* -sz<=ll query_x<=sz */
void init(ll _sz){ sz=_sz+1; root=NULL; }
void add_line(ll m,ll c,ll l=-INF,ll r=INF){
    line v(m,c); ql=l; qr=r; insert(v,-sz,sz,root);
}
ll query(ll x) { return query(x,-sz,sz,root); }
};

```

## 4.8 最小包圍圓

```

/* minimum enclosing circle */
int n;
Pt p[ N ];
const Circle circumcircle(Pt a,Pt b,Pt c){
    Circle cir;
    double fa,fb,fc,fd,fe,ff,dx,dy,dd;
    if( iszero( ( b - a ) ^ ( c - a ) ) ){
        if( ( ( b - a ) * ( c - a ) ) <= 0 )
            return Circle((b+c)/2,norm(b-c)/2);
        if( ( ( c - b ) * ( a - b ) ) <= 0 )
            return Circle((c+a)/2,norm(c-a)/2);
        if( ( ( a - c ) * ( b - c ) ) <= 0 )
            return Circle((a+b)/2,norm(a-b)/2);
    }else{
        fa=2*(a.x-b.x);
        fb=2*(a.y-b.y);
        fc=norm2(a)-norm2(b);
        fd=2*(a.x-c.x);
        fe=2*(a.y-c.y);
        ff=norm2(a)-norm2(c);
        dx=fc*fe-ff*fb;
        dy=fa*ff-fd*fc;
        dd=fa*fe-fd*fb;
        cir.o=Pt(dx/dd,dy/dd);
        cir.r=norm(a-cir.o);
        return cir;
    }
}
inline Circle mec(int fixed,int num){
    int i;
    Circle cir;
    if(fixed==3) return circumcircle(p[0],p[1],p[2]);
    cir=circumcircle(p[0],p[0],p[1]);
    for(i=fixed;i<num;i++) {
        if(cir.inside(p[i])) continue;
        swap(p[i],p[fixed]);
        cir=mec(fixed+1,i+1);
    }
    return cir;
}
inline double min_radius() {
    if(n<=1) return 0.0;
    if(n==2) return norm(p[0]-p[1])/2;
    scramble();
    return mec(0,n).r;
}

```

## 4.9 最小包圍球

```

// Pt : { x , y , z }
#define N 202020
int n, nouter; Pt pt[ N ], outer[4], res;
double radius,tmp;
void ball() {
    Pt q[3]; double m[3][3], sol[3], L[3], det;
    int i,j; res.x = res.y = res.z = radius = 0;
    switch ( nouter ) {
        case 1: res=outer[0]; break;
        case 2: res=(outer[0]+outer[1])/2; radius=norm2(res, outer[0]); break;
        case 3:
            for (i=0; i<2; ++i) q[i]=outer[i+1]-outer[0];
            for (i=0; i<2; ++i) for(j=0; j<2; ++j) m[i][j]=(q[i] * q[j])*2;
            for (i=0; i<2; ++i) sol[i]=(q[i] * q[i]);
            if (fabs(det=m[0][0]*m[1][1]-m[0][1]*m[1][0])<eps)
                return;
            L[0]=(sol[0]*m[1][1]-sol[1]*m[0][1])/det;
            L[1]=(sol[1]*m[0][0]-sol[0]*m[1][0])/det;
            res=outer[0]+q[0]*L[0]+q[1]*L[1];
            radius=norm2(res, outer[0]);
            break;
        case 4:
            for (i=0; i<3; ++i) q[i]=outer[i+1]-outer[0], sol[i]=(q[i] * q[i]);
            for (i=0; i<3; ++i) for(j=0; j<3; ++j) m[i][j]=(q[i] * q[j])*2;
            det= m[0][0]*m[1][1]*m[2][2]
                + m[0][1]*m[1][2]*m[2][0]
                + m[0][2]*m[1][0]*m[2][1]
                - m[0][2]*m[1][1]*m[2][0]
                - m[0][1]*m[1][0]*m[2][2]
                - m[0][0]*m[1][2]*m[2][1];
            if ( fabs(det)<eps ) return;
            for (j=0; j<3; ++j) {
                for (i=0; i<3; ++i) m[i][j]=sol[i];
                L[j]=( m[0][0]*m[1][1]*m[2][2]
                    + m[0][1]*m[1][2]*m[2][0]
                    + m[0][2]*m[1][0]*m[2][1]
                    - m[0][2]*m[1][1]*m[2][0]
                    - m[0][1]*m[1][0]*m[2][2]
                    - m[0][0]*m[1][2]*m[2][1]
                    ) / det;
                for (i=0; i<3; ++i) m[i][j]=(q[i] * q[j])*2;
            }
            res=outer[0];
            for (i=0; i<3; ++i) res = res + q[i] * L[i];
            radius=norm2(res, outer[0]);
    }
}
void minball(int n){ ball();
    if( nouter < 4 ) for( int i = 0 ; i < n ; i ++ )
        if( norm2(res, pt[i]) - radius > eps ){
            outer[ nouter ++ ] = pt[ i ]; minball(i); --
            nouter;
            if(i>0){ Pt Tt = pt[i];
                memmove(&pt[1], &pt[0], sizeof(Pt)*i); pt[0]=Tt;
            }
        }
}
double solve(){
    // n points in pt
    random_shuffle(pt, pt+n); radius=-1;
    for(int i=0;i<n;i++) if(norm2(res,pt[i])-radius>eps)
        nouter=1, outer[0]=pt[i], minball(i);
    return sqrt(radius);
}

```

## 4.10 旋轉卡尺

```

int FarthestPair(vector<Pt>& arr){
    int ret=0;
    for(int i = 0, j = i+1; i<arr.size(); i++){
        while(distance(arr[i], arr[j]) < distance(arr[i], arr[(j+1)%arr.size()])) {
            j = (j+1) % arr.size();
        }
        ret = max(ret, distance(arr[i],arr[j]));
    }
    return ret;
}

```

## 4.11 Circle Cover

```

#define N 1021
#define D long double
struct CircleCover{
    int C; Circ c[ N ]; //填入C(圖數量),c(圖陣列)
    bool g[ N ][ N ]; overlap[ N ][ N ];
    // Area[i] : area covered by at least i circles
    D Area[ N ];
    void init( int _C ){ C = _C; }
    bool CCinter( Circ& a , Circ& b , Pt& p1 , Pt& p2 ){
        Pt o1 = a.O , o2 = b.O;
        D r1 = a.R , r2 = b.R;
        if( norm( o1 - o2 ) > r1 + r2 ) return false;
        if( norm( o1 - o2 ) < max(r1, r2) - min(r1, r2) )
            return true;
        D d2 = ( o1 - o2 ) * ( o1 - o2 );
        D d = sqrt(d2);
        if( d > r1 + r2 ) return false;
        Pt u=(o1+o2)*0.5 + (o1-o2)*((r2*r2-r1*r1)/(2*d2));
        D A=sqrt((r1+r2+d)*(r1-r2+d)*(r1+r2-d)*(r1-r2-d));
        Pt v=Pt( o1.Y-o2.Y , -o1.X + o2.X ) * A / (2*d2);
        p1 = u + v; p2 = u - v;
        return true;
    }
    struct Teve {
        Pt p; D ang; int add;
        Teve() {}
        Teve(Pt _a, D _b, int _c):p(_a), ang(_b), add(_c){}
        bool operator<(const Teve &a)const
            {return ang < a.ang;}
    }eve[ N * 2 ];
    // strict: x = 0, otherwise x = -1
    bool disjuct( Circ& a, Circ &b, int x )
    {return sign( norm( a.O - b.O ) - a.R - b.R ) > x;}
    bool contain( Circ& a, Circ &b, int x )
    {return sign( a.R - b.R - norm( a.O - b.O ) ) > x;}
    bool contain(int i, int j){
        /* c[j] is non-strictly in c[i]. */
        return (sign(c[i].R - c[j].R) > 0 ||
            (sign(c[i].R - c[j].R) == 0 && i < j) ) &&
            contain(c[i], c[j], -1);
    }
    void solve(){
        for( int i = 0 ; i <= C + 1 ; i ++ )
            Area[ i ] = 0;
        for( int i = 0 ; i < C ; i ++ )
            for( int j = 0 ; j < C ; j ++ )
                overlap[i][j] = contain(i, j);
        for( int i = 0 ; i < C ; i ++ )
            for( int j = 0 ; j < C ; j ++ )
                g[i][j] = !(overlap[i][j] || overlap[j][i] ||
                    disjuct(c[i], c[j], -1));
        for( int i = 0 ; i < C ; i ++ ){
            int E = 0, cnt = 1;
            for( int j = 0 ; j < C ; j ++ )
                if( j != i && overlap[j][i] )
                    cnt ++;
            for( int j = 0 ; j < C ; j ++ )
                if( i != j && g[i][j] ){
                    Pt aa, bb;
                    CCinter(c[i], c[j], aa, bb);
                    D A=atan2(aa.Y - c[i].O.Y, aa.X - c[i].O.X);
                    D B=atan2(bb.Y - c[i].O.Y, bb.X - c[i].O.X);
                    eve[E ++] = Teve(bb, B, 1);
                    eve[E ++] = Teve(aa, A, -1);
                    if(B > A) cnt ++;
                }
            if( E == 0 ) Area[ cnt ] += pi * c[i].R * c[i].R;
            else{
                sort( eve , eve + E );
                eve[E] = eve[0];
                for( int j = 0 ; j < E ; j ++ ){
                    cnt += eve[j].add;
                    Area[cnt] += (eve[j].p ^ eve[j + 1].p) * 0.5;
                    D theta = eve[j + 1].ang - eve[j].ang;
                    if (theta < 0) theta += 2.0 * pi;
                    Area[cnt] +=
                        (theta - sin(theta)) * c[i].R*c[i].R * 0.5;
                }
            }
        }
    }
};

```

#### 4.12 Convex Hull Trick

/\* Given a convexhull, answer queries in  $O(\lg N)$

```

CH should not contain identical points, the area should
be > 0, min pair(x, y) should be listed first */
double det( const Pt& p1 , const Pt& p2 )
{ return p1.X * p2.Y - p1.Y * p2.X; }
struct Conv{
    int n;
    vector<Pt> a;
    vector<Pt> upper, lower;
    Conv(vector<Pt> _a) : a(_a){
        n = a.size();
        int ptr = 0;
        for(int i=1; i<n; ++i) if (a[ptr] < a[i]) ptr = i;
        for(int i=0; i<=ptr; ++i) lower.push_back(a[i]);
        for(int i=ptr; i<n; ++i) upper.push_back(a[i]);
        upper.push_back(a[0]);
    }
    int sign( LL x ){ // fixed when changed to double
        return x < 0 ? -1 : x > 0; }
    pair<LL, int> get_tang(vector<Pt> &conv, Pt vec){
        int l = 0, r = (int)conv.size() - 2;
        for( ; l + 1 < r; ){
            int mid = (l + r) / 2;
            if(sign(det(conv[mid+1]-conv[mid],vec))>0)r=mid;
            else l = mid;
        }
        return max(make_pair(det(vec, conv[r]), r),
            make_pair(det(vec, conv[0]), 0));
    }
    void upd_tang(const Pt &p, int id, int &i0, int &i1){
        if(det(a[i0] - p, a[id] - p) > 0) i0 = id;
        if(det(a[i1] - p, a[id] - p) < 0) i1 = id;
    }
    void bi_search(int l, int r, Pt p, int &i0, int &i1){
        if(l == r) return;
        upd_tang(p, l % n, i0, i1);
        int sl=sign(det(a[l % n] - p, a[(l + 1) % n] - p));
        for( ; l + 1 < r; ){
            int mid = (l + r) / 2;
            int smid=sign(det(a[mid%n]-p, a[(mid+1)%n]-p));
            if (smid == sl) l = mid;
            else r = mid;
        }
        upd_tang(p, r % n, i0, i1);
    }
    int bi_search(Pt u, Pt v, int l, int r) {
        int sl = sign(det(v - u, a[l % n] - u));
        for( ; l + 1 < r; ){
            int mid = (l + r) / 2;
            int smid = sign(det(v - u, a[mid % n] - u));
            if (smid == sl) l = mid;
            else r = mid;
        }
        return l % n;
    }
    // 1. whether a given point is inside the CH
    bool contain(Pt p) {
        if (p.X < lower[0].X || p.X > lower.back().X)
            return 0;
        int id = lower_bound(lower.begin(), lower.end(), Pt
            (p.X, -INF)) - lower.begin();
        if (lower[id].X == p.X) {
            if (lower[id].Y > p.Y) return 0;
        }else if(det(lower[id-1]-p,lower[id]-p)<0)return 0;
        id = lower_bound(upper.begin(), upper.end(), Pt(p.X
            , INF), greater<Pt>()) - upper.begin();
        if (upper[id].X == p.X) {
            if (upper[id].Y < p.Y) return 0;
        }else if(det(upper[id-1]-p,upper[id]-p)<0)return 0;
        return 1;
    }
    // 2. Find 2 tang pts on CH of a given outside point
    // return true with i0, i1 as index of tangent points
    // return false if inside CH
    bool get_tang(Pt p, int &i0, int &i1) {
        if (contain(p)) return false;
        i0 = i1 = 0;
        int id = lower_bound(lower.begin(), lower.end(), p)
            - lower.begin();
        bi_search(0, id, p, i0, i1);
        bi_search(id, (int)lower.size(), p, i0, i1);
        id = lower_bound(upper.begin(), upper.end(), p,
            greater<Pt>()) - upper.begin();
    }

```



```

    bi_search((int)lower.size() - 1, (int)lower.size()
        - 1 + id, p, i0, i1);
    bi_search((int)lower.size() - 1 + id, (int)lower.
        size() - 1 + (int)upper.size(), p, i0, i1);
    return true;
}
// 3. Find tangent points of a given vector
// ret the idx of vertex has max cross value with vec
int get_tang(Pt vec){
    pair<LL, int> ret = get_tang(upper, vec);
    ret.second = (ret.second + (int)lower.size() - 1) % n;
    ret = max(ret, get_tang(lower, vec));
    return ret.second;
}
// 4. Find intersection point of a given line
// return 1 and intersection is on edge (i, next(i))
// return 0 if no strictly intersection
bool get_intersection(Pt u, Pt v, int &i0, int &i1){
    int p0 = get_tang(u - v), p1 = get_tang(v - u);
    if(sign(det(v-u, a[p0]-u)) * sign(det(v-u, a[p1]-u)) < 0){
        if(p0 > p1) swap(p0, p1);
        i0 = bi_search(u, v, p0, p1);
        i1 = bi_search(u, v, p1, p0 + n);
        return 1;
    }
    return 0;
}
}
};

```

### 4.13 Half Plane Intersection

```

// for point or line solution, change > to >=
bool onleft(Line L, Pt p) {
    return dcmp(L.v ^ (p - L.s)) > 0;
} // segment should add Counterclockwise
// assume that Lines intersect
vector<Pt> HPI(vector<Line> &L) {
    sort(L.begin(), L.end()); // sort by angle
    int n = L.size(), fir, las;
    Pt *p = new Pt[n];
    Line *q = new Line[n];
    q[fir=las=0] = L[0];
    for(int i = 1; i < n; i++) {
        while(fir < las && !onleft(L[i], p[las-1])) las--;
        while(fir < las && !onleft(L[i], p[fir])) fir++;
        q[++las] = L[i];
        if(dcmp(q[las].v ^ q[las-1].v) == 0) {
            las--;
            if(onleft(q[las], L[i].s)) q[las] = L[i];
        }
        if(fir < las) p[las-1] = LLIntersect(q[las-1], q[las]);
    }
    while(fir < las && !onleft(q[fir], p[las-1])) las--;
    if(las - fir <= 1) return {};
    p[las] = LLIntersect(q[las], q[fir]);
    int m = 0;
    vector<Pt> ans(las - fir + 1);
    for(int i = fir; i <= las; i++) ans[m++] = p[i];
    return ans;
}

```

## 5 圖論

### 5.1 BCC

```

struct BccVertex {
    int n, nScc, step, dfn[MAXN], low[MAXN];
    vector<int> E[MAXN], sccv[MAXN];
    int top, stk[MAXN];
    void init(int _n) {
        n = _n; nScc = step = 0;
        for (int i=0; i<n; i++) E[i].clear();
    }
    void addEdge(int u, int v)
    { E[u].PB(v); E[v].PB(u); }
    void DFS(int u, int f) {
        dfn[u] = low[u] = step++;
        stk[top++] = u;
        for (auto v:E[u]) {
            if (v == f) continue;
            if (dfn[v] == -1) {
                DFS(v, u);

```

```

                low[u] = min(low[u], low[v]);
            if (low[v] >= dfn[u]) {
                int z;
                sccv[nScc].clear();
                do {
                    z = stk[--top];
                    sccv[nScc].PB(z);
                } while (z != v);
                sccv[nScc++].PB(u);
            }
        } else {
            low[u] = min(low[u], dfn[v]);
        }
    }
    vector<vector<int>> solve() {
        vector<vector<int>> res;
        for (int i=0; i<n; i++)
            dfn[i] = low[i] = -1;
        for (int i=0; i<n; i++)
            if (dfn[i] == -1) {
                top = 0;
                DFS(i, i);
            }
        REP(i, nScc) res.PB(sccv[i]);
        return res;
    }
} graph;

```

### 5.2 重心剖分

```

struct CentroidDecomposition {
    int n;
    vector<vector<int>> G, out;
    vector<int> sz, v;
    CentroidDecomposition(int _n) : n(_n), G(_n), out(
        _n), sz(_n), v(_n) {}
    int dfs(int x, int par){
        sz[x] = 1;
        for (auto &i : G[x]) {
            if(i == par || v[i]) continue;
            sz[x] += dfs(i, x);
        }
        return sz[x];
    }
    int search_centroid(int x, int p, const int mid){
        for (auto &i : G[x]) {
            if(i == p || v[i]) continue;
            if(sz[i] > mid) return search_centroid(i, x, mid);
        }
        return x;
    }
    void add_edge(int l, int r){
        G[l].PB(r); G[r].PB(l);
    }
    int get(int x){
        int centroid = search_centroid(x, -1, dfs(x, -1)/2);
        v[centroid] = true;
        for (auto &i : G[centroid]) {
            if(!v[i]) out[centroid].PB(get(i));
        }
        v[centroid] = false;
        return centroid;
    }
};

```

### 5.3 輕重鍊剖分

```

#define REP(i, s, e) for(int i = (s); i <= (e); i++)
#define REPD(i, s, e) for(int i = (s); i >= (e); i--)
const int MAXN = 100010;
const int LOG = 19;
struct HLD{
    int n;
    vector<int> g[MAXN];
    int sz[MAXN], dep[MAXN];
    int ts, tid[MAXN], tdi[MAXN], tl[MAXN], tr[MAXN];
    // ts : timestamp , useless after yutruli
    // tid[ u ] : pos. of node u in the seq.
    // tdi[ i ] : node at pos i of the seq.
    // tl , tr[ u ] : subtree interval in the seq. of node u
    int prt[MAXN][LOG], head[MAXN];

```

```
// head[ u ] : head of the chain contains u
void dfsz(int u, int p){
    dep[u] = dep[p] + 1;
    prt[u][0] = p; sz[u] = 1; head[u] = u;
    for(int& v:g[u]) if(v != p){
        dep[v] = dep[u] + 1;
        dfsz(v, u);
        sz[u] += sz[v];
    }
}
void dfshl(int u){
    ts++;
    tid[u] = tl[u] = tr[u] = ts;
    tdi[tid[u]] = u;
    sort(ALL(g[u]),
        [&](int a, int b){return sz[a] > sz[b];});
    bool flag = 1;
    for(int& v:g[u]) if(v != prt[u][0]){
        if(flag) head[v] = head[u], flag = 0;
        dfshl(v);
        tr[u] = tr[v];
    }
}
inline int lca(int a, int b){
    if(dep[a] > dep[b]) swap(a, b);
    int diff = dep[b] - dep[a];
    REPD(k, LOG-1, 0) if(diff & (1<<k)){
        b = prt[b][k];
    }
    if(a == b) return a;
    REPD(k, LOG-1, 0) if(prt[a][k] != prt[b][k]){
        a = prt[a][k]; b = prt[b][k];
    }
    return prt[a][0];
}
void init(int _n){
    n = _n; REP(i, 1, n) g[i].clear();
}
void addEdge(int u, int v){
    g[u].push_back(v);
    g[v].push_back(u);
}
void yutruli(){ //build function
    dfsz(1, 0);
    ts = 0;
    dfshl(1);
    REP(k, 1, LOG-1) REP(i, 1, n)
        prt[i][k] = prt[prt[i][k-1]][k-1];
}
vector<PII> getPath(int u, int v){
    vector<PII> res;
    while(tid[u] < tid[head[v]]){
        res.push_back(PII(tid[head[v]], tid[v]));
        v = prt[head[v]][0];
    }
    res.push_back(PII(tid[u], tid[v]));
    reverse(ALL(res));
    return res;
}
/* res : list of intervals from u to v
 * u must be ancestor of v
 * usage :
 * vector<PII> &path = tree.getPath(u, v)
 * for(PII tp : path){
 *     int l, r; tie(l, r) = tp;
 *     upd(l, r);
 *     uu = tree.tdi[l], vv = tree.tdi[r];
 *     uu ~> vv is a heavy path on tree
 * }
 */
}
tree;
```

## 5.4 歐拉路徑

```
#define FOR(i,a,b) for(int i=a;i<=b;i++)
int dfs_st[10000500], dfn=0;
int ans[10000500], cnt=0, num=0;
vector<int> G[10000500];
int cur[10000500];
int ind[10000500], out[10000500];
void dfs(int x){
```

```
    FOR(i,1,n) sort(G[i].begin(), G[i].end());
    dfs_st[++dfn]=x;
    memset(cur, -1, sizeof(cur));
    while(dfn>0){
        int u=dfs_st[dfn];
        int complete=1;
        for(int i=cur[u]+1; i<G[u].size(); i++){
            int v=G[u][i];
            num++;
            dfs_st[++dfn]=v;
            cur[u]=i;
            complete=0;
            break;
        }
        if(complete) ans[++cnt]=u, dfn--;
    }
}
bool check(int &start){
    int l=0, r=0, mid=0;
    FOR(i,1,n){
        if(ind[i]==out[i]+1) l++;
        if(out[i]==ind[i]+1) r++, start=i;
        if(ind[i]==out[i]) mid++;
    }
    if(l==1&&r==1&&mid==n-2) return true;
    l=1;
    FOR(i,1,n) if(ind[i]!=out[i]) l=0;
    if(l){
        FOR(i,1,n) if(out[i]>0){
            start=i;
            break;
        }
        return true;
    }
    return false;
}
int main(){
    cin>>n>>m;
    FOR(i,1,m){
        int x,y; scanf("%d%d",&x,&y);
        G[x].push_back(y);
        ind[y]++, out[x]++;
    }
    int start=-1, ok=true;
    if(check(start)){
        dfs(start);
        if(num!=m){
            puts("What a shame!");
            return 0;
        }
        for(int i=cnt; i>=1; i--){
            printf("%d ", ans[i]);
            puts("");
        }
        else puts("What a shame!");
    }
}
```

## 5.5 極大團

```
#define N 80
struct MaxClique{ // 0-base
    typedef bitset<N> Int;
    Int lnk[N], v[N];
    int n;
    void init(int _n){
        n = _n;
        for(int i = 0; i < n; i++){
            lnk[i].reset(); v[i].reset();
        }
    }
    void addEdge(int a, int b)
    { v[a][b] = v[b][a] = 1; }
    int ans, stk[N], id[N], di[N], deg[N];
    Int cans;
    void dfs(int elem_num, Int candi, Int ex){
        if(candi.none() && ex.none()){
            cans.reset();
            for(int i = 0; i < elem_num; i++){
                cans[id[stk[i]]] = 1;
                ans = elem_num; // cans is a maximal clique
            }
            return;
        }
        int pivot = (candi ^ ex)._Find_first();
```

```

Int smaller_candi = candi & (~lnk[pivot]);
while(smaller_candi.count()){
    int nxt = smaller_candi._Find_first();
    candi[nxt] = smaller_candi[nxt] = 0;
    ex[nxt] = 1;
    stk[elem_num] = nxt;
    dfs(elem_num+1, candi&lnk[nxt], ex&lnk[nxt]);
} }
int solve(){
    for(int i = 0 ; i < n ; i ++){
        id[i] = i; deg[i] = v[i].count();
    }
    sort(id , id + n , [&](int id1, int id2){
        return deg[id1] > deg[id2]; });
    for(int i = 0 ; i < n ; i ++){ di[id[i]] = i;
    for(int i = 0 ; i < n ; i ++){
        for(int j = 0 ; j < n ; j ++){
            if(v[i][j]) lnk[di[i]][di[j]] = 1;
        }
        ans = 1; cans.reset(); cans[0] = 1;
        dfs(0, Int(string(n, '1')), 0);
        return ans;
    } } solver;

```

## 5.6 最大團

```

#define N 111
struct MaxClique{ // 0-base
    typedef bitset<N> Int;
    Int linkto[N] , v[N];
    int n;
    void init(int _n){
        n = _n;
        for(int i = 0 ; i < n ; i ++){
            linkto[i].reset(); v[i].reset();
        }
    }
    void addEdge(int a , int b)
    { v[a][b] = v[b][a] = 1; }
    int popcount(const Int& val)
    { return val.count(); }
    int lowbit(const Int& val)
    { return val._Find_first(); }
    int ans , stk[N];
    int id[N] , di[N] , deg[N];
    Int cans;
    void maxclique(int elem_num, Int candi){
        if(elem_num > ans){
            ans = elem_num; cans.reset();
            for(int i = 0 ; i < elem_num ; i ++){
                cans[id[stk[i]]] = 1;
            }
            int potential = elem_num + popcount(candi);
            if(potential <= ans) return;
            int pivot = lowbit(candi);
            Int smaller_candi = candi & (~linkto[pivot]);
            while(smaller_candi.count() && potential > ans){
                int next = lowbit(smaller_candi);
                candi[next] = !candi[next];
                smaller_candi[next] = !smaller_candi[next];
                potential --;
                if(next == pivot || (smaller_candi & linkto[next
                ]).count()){
                    stk[elem_num] = next;
                    maxclique(elem_num + 1, candi & linkto[next]);
                }
            }
        }
    }
    int solve(){
        for(int i = 0 ; i < n ; i ++){
            id[i] = i; deg[i] = v[i].count();
        }
        sort(id , id + n , [&](int id1, int id2){
            return deg[id1] > deg[id2]; });
        for(int i = 0 ; i < n ; i ++){ di[id[i]] = i;
        for(int i = 0 ; i < n ; i ++){
            for(int j = 0 ; j < n ; j ++){
                if(v[i][j]) linkto[di[i]][di[j]] = 1;
            }
        }
        Int cand; cand.reset();
        for(int i = 0 ; i < n ; i ++){ cand[i] = 1;
        ans = 1;
        cans.reset(); cans[0] = 1;
        maxclique(0, cand);
        return ans;
    } } solver;

```

## 5.7 SCC

```

struct Scc{
    int n, nScc, vst[MXN], bln[MXN];
    vector<int> E[MXN], rE[MXN], vec;
    void init(int _n){
        n = _n;
        for (int i=0; i<MXN; i++)
            E[i].clear(), rE[i].clear();
    }
    void addEdge(int u, int v){
        E[u].PB(v); rE[v].PB(u);
    }
    void DFS(int u){
        vst[u]=1;
        for (auto v : E[u]) if (!vst[v]) DFS(v);
        vec.PB(u);
    }
    void rDFS(int u){
        vst[u] = 1; bln[u] = nScc;
        for (auto v : rE[u]) if (!vst[v]) rDFS(v);
    }
    void solve(){
        nScc = 0;
        vec.clear();
        FZ(vst);
        for (int i=0; i<n; i++)
            if (!vst[i]) DFS(i);
        reverse(vec.begin(), vec.end());
        FZ(vst);
        for (auto v : vec)
            if (!vst[v]){
                rDFS(v); nScc++;
            }
    }
};

```

## 5.8 SPFA

```

#define MXN 200005
struct SPFA{
    int n;
    LL inq[MXN], len[MXN];
    vector<LL> dis;
    vector<pair<int, LL>> edge[MXN];
    void init(int _n){
        n = _n;
        dis.clear(); dis.resize(n, 1e18);
        for(int i = 0; i < n; i++){
            edge[i].clear();
            inq[i] = len[i] = 0;
        }
    }
    void addEdge(int u, int v, LL w){
        edge[u].push_back({v, w});
    }
    vector<LL> solve(int st = 0){
        deque<int> dq; //return {-1} if has negative cycle
        dq.push_back(st); //otherwise return dis from st
        inq[st] = 1; dis[st] = 0;
        while(!dq.empty()){
            int u = dq.front(); dq.pop_front();
            inq[u] = 0;
            for(auto [to, d] : edge[u]){
                if(dis[to] > d+dis[u]){
                    dis[to] = d+dis[u];
                    len[to] = len[u]+1;
                    if(len[to] > n) return {-1};
                    if(inq[to]) continue;
                    (!dq.empty()&&dis[dq.front()] > dis[to]?
                    dq.push_front(to) : dq.push_back(to));
                    inq[to] = 1;
                }
            }
        }
        return dis;
    }
};

```

## 5.9 差分約束

約束條件  $V_j - V_i \leq W$  addEdge( $V_i, V_j, W$ ) and run bellman-ford or spfa

## 6 數論

### 6.1 離散根號

```

void calcH(LL &t, LL &h, const LL p) {
    LL tmp=p-1; for(t=0;(tmp&1)==0;tmp/=2) t++; h=tmp;
}
// solve equation x^2 mod p = a
bool solve(LL a, LL p, LL &x, LL &y) {
    if(p == 2) { x = y = 1; return true; }
    int p2 = p / 2, tmp = mypow(a, p2, p);
    if (tmp == p - 1) return false;
    if ((p + 1) % 4 == 0) {
        x=mypow(a,(p+1)/4,p); y=p-x; return true;
    } else {
        LL t, h, b, pb; calcH(t, h, p);
        if (t >= 2) {
            do {b = rand() % (p - 2) + 2;
                } while (mypow(b, p / 2, p) != p - 1);
            pb = mypow(b, h, p);
        } int s = mypow(a, h / 2, p);
        for (int step = 2; step <= t; step++) {
            int ss = (((LL)(s * s) % p) * a) % p;
            for(int i=0;i<t-step;i++) ss=mul(ss,ss,p);
            if (ss + 1 == p) s = (s * pb) % p;
            pb = ((LL)pb * pb) % p;
        } x = ((LL)s * a) % p; y = p - x;
    } return true;
}

```

## 6.2 ex-crt

```

typedef __int128 ll;
void exgcd(ll a,ll b,ll &g,ll &x,ll &y) {
    if (b == 0) {
        g = a;
        x = 1;
        y = 0;
        return;
    }
    exgcd(b,a%b,g,y,x);
    y-=(a/b)*x;
}
bool flag = false;
ll a1,a2,n1,n2;
ll abs(ll x) {
    return x>0?x:-x;
}
void china() {
    ll d = a2 - a1;
    ll g,x,y;
    exgcd(n1,n2,g,x,y);
    if (d % g == 0) {
        x = ((x*d/g)%(n2/g)+(n2/g))%(n2/g);
        a1 = x*n1 + a1;
        n1 = (n1*n2)/g;
    }
    else
        flag = true;
}
int n;
long long as[100001]; //算式答案 x
long long ns[100001]; //模数 MOD
ll realchina() {
    a1 = as[0];
    n1 = ns[0];
    for (ll i = 1;i<n;i++) {
        a2 = as[i];
        n2 = ns[i];
        china();
        if (flag)
            return -1;
    }
    return a1;
}
int main() {
    cin>>n;
    flag = false;
    for (ll i = 0;i<n;i++)
        cin>>ns[i]>>as[i];
    cout<<(long long)realchina()<<endl;
}

```

## 6.3 ex-gcd

```

int exgcd(int a,int b,int&x,int&y){

```

```

    if(b==0)return x=1,y=0,a;
    int d = exgcd(b,a%b,y,x);
    y-=a/b*x;
    return d;
}

```

## 6.4 FFT

```

// const int MAXN = 262144;
// (must be 2^k)
// before any usage, run pre_fft() first
typedef long double ld;
typedef complex<ld> cplx; //real() ,imag()
const ld PI = acos(-1);
const cplx I(0, 1);
cplx omega[MAXN+1];
void pre_fft(){
    for(int i=0; i<=MAXN; i++)
        omega[i] = exp(i * 2 * PI / MAXN * I);
}
// n must be 2^k
void fft(int n, cplx a[], bool inv=false){
    int basic = MAXN / n;
    int theta = basic;
    for (int m = n; m >= 2; m >= 1) {
        int mh = m >> 1;
        for (int i = 0; i < mh; i++) {
            cplx w = omega[inv ? MAXN-(i*theta%MAXN)
                                : i*theta%MAXN];
            for (int j = i; j < n; j += m) {
                int k = j + mh;
                cplx x = a[j] - a[k];
                a[j] += a[k];
                a[k] = w * x;
            }
            theta = (theta * 2) % MAXN;
        }
        int i = 0;
        for (int j = 1; j < n - 1; j++) {
            for (int k = n >> 1; k > (i ^= k); k >= 1);
            if (j < i) swap(a[i], a[j]);
        }
        if(inv) for (i = 0; i < n; i++) a[i] /= n;
    }
    cplx arr[MAXN+1];
    inline void mul(int _n,ll a[],int _m,ll b[],ll ans[]){
        int n=1,sum=_n+_m-1;
        while(n<sum)
            n<=1;
        for(int i=0;i<n;i++) {
            double x=(i<_n?a[i]:0),y=(i<_m?b[i]:0);
            arr[i]=complex<double>(x+y,x-y);
        }
        fft(n,arr);
        for(int i=0;i<n;i++)
            arr[i]=arr[i]*arr[i];
        fft(n,arr,true);
        for(int i=0;i<sum;i++)
            ans[i]=(long long int)(arr[i].real()/4+0.5);
    }
}

```

## 6.5 高斯消去法

```

const int GAUSS_MOD = 100000007LL;
struct GAUSS{
    int n;
    vector<vector<int>> v;
    int ppow(int a , int k){
        if(k == 0) return 1;
        if(k % 2 == 0) return ppow(a * a % GAUSS_MOD ,
            k >> 1);
        if(k % 2 == 1) return ppow(a * a % GAUSS_MOD ,
            k >> 1) * a % GAUSS_MOD;
    }
    vector<int> solve(){
        vector<int> ans(n);
        REP(now , 0 , n){
            REP(i , now , n) if(v[now][now] == 0 && v[i
                ][now] != 0)
                swap(v[i] , v[now]); // det = -det;
            if(v[now][now] == 0) return ans;

```

```

    int inv = ppow(v[now][now] , GAUSS_MOD - 2)
    REP(i , 0 , n) if(i != now){
        int tmp = v[i][now] * inv % GAUSS_MOD;
        REP(j , now , n + 1) (v[i][j] +=
            GAUSS_MOD - tmp * v[now][j] %
            GAUSS_MOD) %= GAUSS_MOD;
    }
    REP(i , 0 , n) ans[i] = v[i][n + 1] * ppow(v[i][i] , GAUSS_MOD - 2) % GAUSS_MOD;
    return ans;
}
// gs.v.clear() , gs.v.resize(n , vector<int>(n + 1 , 0));
} gs;

```

## 6.6 喬瑟夫問題

```

int josephus(int n, int m){ //n人每m次
    int ans = 0;
    for (int i=1; i<=n; ++i)
        ans = (ans + m) % i;
    return ans;
}

```

## 6.7 定理

- Lucas's Theorem :  
For  $n, m \in \mathbb{Z}^*$  and prime  $P$ ,  $C(m, n) \bmod P = \prod(C(m_i, n_i))$  where  $m_i$  is the  $i$ -th digit of  $m$  in base  $P$ .
- Stirling approximation :  
$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}$$
- Stirling Numbers(permutation  $|P| = n$  with  $k$  cycles):  
 $S(n, k) = \text{coefficient of } x^k \text{ in } \prod_{i=0}^{n-1} (x + i)$
- Stirling Numbers(Partition  $n$  elements into  $k$  non-empty set):  
$$S(n, k) = \frac{1}{k!} \sum_{j=0}^k (-1)^{k-j} \binom{k}{j} j^n$$
- Pick's Theorem :  $A = i + b/2 - 1$   
A: Area,  $i$ : grid number in the inner,  $b$ : grid number on the side
- Catalan number :  $C_n = \binom{2n}{n} / (n + 1)$   
$$C_n^{n+m} - C_{n+1}^{n+m} = (m + n)! \frac{n-m+1}{n+1} \text{ for } n \geq m$$
  
$$C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$$
  
$$C_0 = 1 \text{ and } C_{n+1} = 2 \binom{2n+1}{n+2} C_n$$
  
$$C_0 = 1 \text{ and } C_{n+1} = \sum_{i=0}^n C_i C_{n-i} \text{ for } n \geq 0$$
- Euler Characteristic:  
planar graph:  $V - E + F - C = 1$   
convex polyhedron:  $V - E + F = 2$   
 $V, E, F, C$ : number of vertices, edges, faces(regions), and components
- Kirchhoff's theorem :  
 $A_{ii} = \deg(i), A_{ij} = (i, j) \in E ? -1 : 0$ , Deleting any one row, one column, and cal the  $\det(A)$
- Polya' theorem ( $c$  is number of color,  $m$  is the number of cycle size):  
$$(\sum_{i=1}^m c^{\gcd(i, m)}) / m$$
- Burnside lemma:  
 $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$
- 錯排公式: ( $n$  個人中, 每個人皆不再原來位置的組合數):  
$$dp[0] = 1; dp[1] = 0;$$
  
$$dp[i] = (i - 1) * (dp[i - 1] + dp[i - 2]);$$
- Bell 數 (有  $n$  個人, 把他們拆組的方法總數) :  
$$B_0 = 1$$
  
$$B_n = \sum_{k=0}^n s(n, k) \text{ (second - stirling)}$$
  
$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k$$
- Wilson's theorem :  
$$(p - 1)! \equiv -1 \pmod p$$
- Fermat's little theorem :  
$$a^p \equiv a \pmod p$$
- Euler's totient function:  
$$A^{B^C} \bmod p = \text{pow}(A, \text{pow}(B, C, p - 1)) \bmod p$$
- 歐拉函數降冪公式:  
$$A^B \bmod C = A^{B \bmod \phi(C) + \phi(C)} \bmod C$$
- 6 的倍數:  
$$(a - 1)^3 + (a + 1)^3 + (-a)^3 + (-a)^3 = 6a$$

## 6.8 Miller Rabin

```

// n < 4,759,123,141      3 : 2, 7, 61
// n < 1,122,004,669,633  4 : 2, 13, 23, 1662803
// n < 3,474,749,660,383  6 : pirmes <= 13
// n < 2^64              7 :
// 2, 325, 9375, 28178, 450775, 9780504, 1795265022
// Make sure testing integer is in range [2, n-2] if
// you want to use magic.
LL magic[]={}
bool witness(LL a, LL n, LL u, int t){
    if(!a) return 0;
    LL x=mpow(a, u, n);
    for(int i=0; i<t; i++){
        LL nx=mul(x, x, n);
        if(nx==1&&x!=1&&x!=n-1) return 1;
        x=nx;
    }
    return x!=1;
}
bool miller_rabin(LL n) {
    int s=(magic number size)
    // iterate s times of witness on n
    if(n<2) return 0;
    if(!(n&1)) return n == 2;
    ll u=n-1; int t=0;
    // n-1 = u*2^t
    while(!(u&1)) u>>=1, t++;
    while(s--){
        LL a=magic[s]%n;
        if(witness(a, n, u, t)) return 0;
    }
    return 1;
}

```

## 6.9 NTT

```

// Remember coefficient are mod P
/* p=a*2^n+1
n    2^n    p    a    root
16   65536   65537   1    3
20   1048576 7340033 7    3 */
// (must be 2^k)
template<LL P, LL root, int MAXN>
struct NTT{
    static LL bigmod(LL a, LL b) {
        LL res = 1;
        for (LL bs = a; b; b >= 1, bs = (bs * bs) % P)
            if(b&1) res=(res*bs)%P;
        return res;
    }
    static LL inv(LL a, LL b) {
        if(a==1) return 1;
        return (((LL)(a-inv(b*a,a))*b+1)/a)%b;
    }
    LL omega[MAXN+1];
    NTT() {
        omega[0] = 1;
        LL r = bigmod(root, (P-1)/MAXN);
        for (int i=1; i<=MAXN; i++)
            omega[i] = (omega[i-1]*r)%P;
    }
    // n must be 2^k
    void tran(int n, LL a[], bool inv_ntt=false){
        int basic = MAXN / n, theta = basic;
        for (int m = n; m >= 2; m >= 1) {
            int mh = m >> 1;
            for (int i = 0; i < mh; i++) {
                LL w = omega[i*theta%MAXN];
                for (int j = i; j < n; j += m) {
                    int k = j + mh;
                    LL x = a[j] - a[k];
                    if (x < 0) x += P;
                    a[j] += a[k];
                    if (a[j] > P) a[j] -= P;
                    a[k] = (w * x) % P;
                }
            }
            theta = (theta * 2) % MAXN;
        }
        int i = 0;
        for (int j = 1; j < n - 1; j++) {

```



```

    for (int k = n >> 1; k > (i ^ k); k >= 1);
    if (j < i) swap(a[i], a[j]);
}
if (inv_ntt) {
    LL ni = inv(n,P);
    reverse(a+1, a+n);
    for (i = 0; i < n; i++)
        a[i] = (a[i] * ni) % P;
} } }
const LL P=2013265921, root=31;
const int MAXN=4194304;
NTT<P, root, MAXN> ntt;

```

## 6.10 Pollard's Rho

```

// does not work when n is prime O(n^(1/4))
LL f(LL x, LL mod){ return add(mul(x,x,mod),1,mod); }
LL pollard_rho(LL n) {
    if(!(n&1)) return 2;
    while(true){
        LL y=2, x=rand()%(n-1)+1, res=1;
        for(int sz=2; res==1; sz*=2) {
            for(int i=0; i<sz && res<=1; i++) {
                x = f(x, n);
                res = __gcd(abs(x-y), n);
            }
            y = x;
        }
        if (res!=0 && res!=n) return res;
    }
}

```

## 6.11 質數

```

/* 12721, 13331, 14341, 75577, 123457, 222557, 556679
 * 999983, 1097774749, 1076767633, 100102021, 99997771
 * 1001010013, 1000512343, 987654361, 999991231
 * 999888733, 98789101, 987777733, 999991921, 1010101333
 * 1010102101, 1000000000039, 100000000000037
 * 2305843009213693951, 4611686018427387847
 * 9223372036854775783, 18446744073709551557 */
int mu[ N ], p_tbl[ N ];
vector<int> primes;
void sieve() {
    mu[ 1 ] = p_tbl[ 1 ] = 1;
    for( int i = 2 ; i < N ; i ++ ){
        if( !p_tbl[ i ] ){
            p_tbl[ i ] = i;
            primes.push_back( i );
            mu[ i ] = -1;
        }
        for( int p : primes ){
            int x = i * p;
            if( x >= M ) break;
            p_tbl[ x ] = p;
            mu[ x ] = -mu[ i ];
            if( i % p == 0 ){
                mu[ x ] = 0;
                break;
            }
        }
    }
}
vector<int> factor( int x ){
    vector<int> fac{ 1 };
    while( x > 1 ){
        int fn = SZ(fac), p = p_tbl[ x ], pos = 0;
        while( x % p == 0 ){
            x /= p;
            for( int i = 0 ; i < fn ; i ++ )
                fac.PB( fac[ pos ++ ] * p );
        }
    }
    return fac;
}

```

## 6.12 phi

```

ll phi(ll n){ // 計算小於n的數中與n互質的有幾個
    ll res = n, a=n; // O(sqrtN)
    for(ll i=2;i*i<=a;i++){
        if(a%i==0){
            res = res/i*(i-1);
            while(a%i==0) a/=i;
        }
    }
    if(a>1) res = res/a*(a-1);
    return res;
}

```

## 6.13 矩陣快速冪

```

LL len,mod;
vector<vector<LL>> operator*(vector<vector<LL>> x,
vector<vector<LL>> y){
    vector<vector<LL>> ret(len,vector<LL>(len,0));
    for(int i=0;i<len;i++){
        for(int j=0;j<len;j++){
            for(int k=0;k<len;k++){
                ret[i][j]=(ret[i][j]+x[i][k]*y[k][j])%
                    mod;
            }
        }
    }
    return ret;
}
struct Martix_fast_pow{ //O(len^3 lg k)
    LL init(int _len,LL m=9223372036854775783LL){
        len=_len, mod=m;
    }
    // mfp.solve(k,{0, 1}, {1, 1}) k'th fib {值,係數} // 0-base
    LL solve(LL n,vector<vector<LL>> poly){
        if(n<len) return poly[n][0];
        vector<vector<LL>> mar(len,vector<LL>(len,0)),x
            (len,vector<LL>(len,0));
        for(int i=0;i<len;i++) mar[i][i]=1;
        for(int i=0;i+1<len;i++) x[i][i+1]=1;
        for(int i=0;i<len;i++) x[len-1][i]=poly[i][1];
        while(n){
            if(n&1) mar=mar*x;
            n>>=1, x=x*x;
        }
        LL ans=0;
        for(int i=0;i<len;i++) ans=(ans+mar[len-1][i]
            *poly[i][0]%mod)%mod;
        return ans;
    }
}mfp;

```

## 7 字串

### 7.1 KMP

/\* len-failure[k]:  
在k結尾的情況下，這個子字串可以由開頭  
長度為(len-failure[k])的部分重複出現來表達

failure[k]為次長相同前綴後綴  
如果我們不只想求最多，而且以0-base做為考量  
，那可能的長度由大到小會是  
failuer[k]、failure[failuer[k]-1]  
、failure[failure[failuer[k]-1]-1]..  
直到有值為0為止 \*/

```

int failure[MAXN];
vector<int>ret;
void KMP(string& t, string& p){
    if (p.size() > t.size()) return;
    for (int i=1, j=failure[0]=-1; i<p.size(); ++i){
        while (j >= 0 && p[j+1] != p[i])
            j = failure[j];
        if (p[j+1] == p[i]) j++;
        failure[i] = j;
    }
    for (int i=0, j=-1; i<t.size(); ++i){
        while (j >= 0 && p[j+1] != t[i])
            j = failure[j];
        if (p[j+1] == t[i]) j++;
        if (j == p.size()-1){
            ret.push_back( i - p.size() + 1 );
            j = failure[j];
        }
    }
    return ;
}

```

### 7.2 馬拉車

```

void manacher(char *s,int len,int *z){
    len=(len<<1)+1;
    for(int i=len-1;i>=0;i--){
        s[i]=i&1?s[i>>1]:'@';
        z[0]=1;
        for(int i=1,l=0,r=0;i<len;i++){
            z[i]=i<r?min(z[l+l-i],r-i):1;
            while(i-z[i]>=0&&i+z[i]<len&&s[i-z[i]]==s[i+z[i]])
                ++z[i];
        }
    }
}

```



```
    if(i+z[i]>r) l=i,r=i+z[i];
} }
```

### 7.3 回文樹

```
// len[s]是對應的回文長度
// num[s]是有幾個回文後綴
// cnt[s]是這個回文子字串在整個字串中的出現次數
// fail[s]是他長度次長的回文後綴 · aba的fail是a
const int MXN = 1000010;
struct PalT{
    int nxt[MXN][26],fail[MXN],len[MXN];
    int tot,lst,n,state[MXN],cnt[MXN],num[MXN];
    int diff[MXN],sfail[MXN],fac[MXN],dp[MXN];
    char s[MXN]={-1};
    int newNode(int l,int f){
        len[tot]=l,fail[tot]=f,cnt[tot]=num[tot]=0;
        memset(nxt[tot],0,sizeof(nxt[tot]));
        diff[tot]=(l>0?l-len[f]:0);
        sfail[tot]=(l>0&&diff[tot]==diff[f]?sfail[f]:f);
        return tot++;
    }
    int getfail(int x){
        while(s[n-len[x]-1]!=s[n]) x=fail[x];
        return x;
    }
    int getmin(int v){
        dp[v]=fac[n-len[sfail[v]]-diff[v]];
        if(diff[v]==diff[fail[v]])
            dp[v]=min(dp[v],dp[fail[v]]);
        return dp[v]+1;
    }
    int push(){
        int c=s[n]-'a',np=getfail(lst);
        if(!(lst=nxt[np][c])){
            lst=newNode(len[np]+2,nxt[getfail(fail[np])][c]);
            nxt[np][c]=lst; num[lst]=num[fail[lst]]+1;
        }
        fac[n]=n;
        for(int v=lst;len[v]>0;v=sfail[v])
            fac[n]=min(fac[n],getmin(v));
        return ++cnt[lst],lst;
    }
    void init(const char *_s){
        tot=lst=n=0;
        newNode(0,1),newNode(-1,1);
        for(;_s[n];) s[n+1]=_s[n],++n,state[n-1]=push();
        for(int i=tot-1;i>1;i--) cnt[fail[i]]+=cnt[i];
    }
}palt;
```

### 7.4 SA

```
const int N = 300010;
struct SA{
    #define REP(i,n) for ( int i=0; i<(int)(n); i++ )
    #define REP1(a,b) for ( int i=(a); i<=(int)(b); i++ )
    bool _t[N*2];
    int _s[N*2], _sa[N*2], _c[N*2], x[N], _p[N], _q[N*2],
        hei[N], r[N];
    int operator [] (int i){ return _sa[i]; }
    void build(int *s, int n, int m){
        memcpy(_s, s, sizeof(int) * n);
        sais(_s, _sa, _p, _q, _t, _c, n, m);
        mkhei(n);
    }
    void mkhei(int n){
        REP(i,n) r[_sa[i]] = i;
        hei[0] = 0;
        REP(i,n) if(r[i]) {
            int ans = i>0 ? max(hei[r[i-1]] - 1, 0) : 0;
            while(_s[i+ans] == _s[_sa[r[i]-1]+ans]) ans++;
            hei[r[i]] = ans;
        }
    }
    void sais(int *s, int *sa, int *p, int *q, bool *t,
        int *c, int n, int z){
        bool uniq = t[n-1] = true, neq;
        int nn = 0, nmzx = -1, *nsa = sa + n, *ns = s + n,
            lst = -1;
        #define MS0(x,n) memset((x),0,n*sizeof(*(x)))
        #define MAGIC(XD) MS0(sa, n); \
```

```
memcpy(x, c, sizeof(int) * z); \
XD; \
memcpy(x + 1, c, sizeof(int) * (z - 1)); \
REP(i,n) if(sa[i] && !t[sa[i]-1]) sa[x[sa[i]
    ]-1]++ = sa[i]-1; \
memcpy(x, c, sizeof(int) * z); \
for(int i = n - 1; i >= 0; i--) if(sa[i] && t[sa[i]
    ]-1) sa[-x[sa[i]-1]] = sa[i]-1;
MS0(c, z);
REP(i,n) uniq &= ++c[s[i]] < 2;
REP(i,z-1) c[i+1] += c[i];
if (uniq) { REP(i,n) sa[-c[s[i]]] = i; return; }
for(int i = n - 2; i >= 0; i--) t[i] = (s[i]==s[i
    +1] ? t[i+1] : s[i]<s[i+1]);
MAGIC(REP1(i,1,n-1) if(t[i] && !t[i-1]) sa[-x[s[i]
    ]]=p[q[i]=nn++]=i);
REP(i, n) if (sa[i] && t[sa[i]] && !t[sa[i]-1]) {
    neq=lst<0||memcmp(s+sa[i],s+lst,(p[q[sa[i]]+1]-sa
        [i])*sizeof(int));
    ns[q[lst=sa[i]]]=nmzx+=neq;
}
sais(ns, nsa, p + nn, q + n, t + n, c + z, nn, nmzx
    + 1);
MAGIC(for(int i = nn - 1; i >= 0; i--) sa[-x[p[
    nsa[i]]]] = p[nsa[i]]);
}
}sa;
int H[ N ], SA[ N ];
void suffix_array(int* ip, int len) {
    // should padding a zero in the back
    // ip is int array, len is array length
    // ip[0..n-1] != 0, and ip[len] = 0
    ip[len++] = 0;
    sa.build(ip, len, 128);
    for (int i=0; i<len; i++) {
        H[i] = sa.hei[i + 1];
        SA[i] = sa._sa[i + 1];
    }
    // resulting height, sa array \in [0,len)
}
```

### 7.5 SAM

```
// any path start from root forms a substring of S
// occurrence of P : iff SAM can run on input word P
// number of different substring : ds[1]-1
// total length of all different substring : dsl[1]
// max/min length of state i : mx[i]/mx[mom[i]]+1
// assume a run on input word P end at state i:
// number of occurrences of P : cnt[i]
// first occurrence position of P : fp[i]-lpl+1
// all position of P : fp of "dfs from i through rmom"
const int MXM = 1000010;
struct SAM{
    int tot, root, lst, mom[MXM], mx[MXM]; //ind[MXM]
    int nxt[MXM][33]; //cnt[MXM],ds[MXM],dsl[MXM],fp[MXM]
    // bool v[MXM]
    int newNode(){
        int res = ++tot;
        fill(nxt[res], nxt[res]+33, 0);
        mom[res] = mx[res] = 0; //cnt=ds=dsl=fp=v=0
        return res;
    }
    void init(){
        tot = 0;
        root = newNode();
        lst = root;
    }
    void push(int c){
        int p = lst;
        int np = newNode(); //cnt[np]=1
        mx[np] = mx[p]+1; //fp[np]=mx[np]-1
        for(; p && nxt[p][c] == 0; p = mom[p])
            nxt[p][c] = np;
        if(p == 0) mom[np] = root;
        else{
            int q = nxt[p][c];
            if(mx[p]+1 == mx[q]) mom[np] = q;
            else{
                int nq = newNode(); //fp[nq]=fp[q]
                mx[nq] = mx[p]+1;
                for(int i = 0; i < 33; i++)
```

```

    nxt[nq][i] = nxt[q][i];
    mom[nq] = mom[q];
    mom[q] = nq;
    mom[np] = nq;
    for(; p && nxt[p][c] == q; p = mom[p])
        nxt[p][c] = nq;
} }
lst = np;
}
void calc(){
    calc(root);
    iota(ind, ind+tot, 1);
    sort(ind, ind+tot, [&](int i, int j){return mx[i]<mx[j];});
    for(int i=tot-1; i>=0; i--){
        cnt[mom[ind[i]]] += cnt[ind[i]];
    }
}
void calc(int x){
    v[x]=ds[x]=1; dsl[x]=0; //rmom[mom[x]].push_back(x);
    for(int i=1; i<=26; i++){
        if(nxt[x][i]){
            if(!v[nxt[x][i]]) calc(nxt[x][i]);
            ds[x] += ds[nxt[x][i]];
            dsl[x] += ds[nxt[x][i]] + dsl[nxt[x][i]];
        }
    }
}
void push(const string& str){
    for(int i = 0; i < str.size(); i++){
        push(str[i] - 'a' + 1);
    }
}
} sam;

```

## 7.6 樹哈希

```

map<vector<int>, int> id;
ll dfs(int u){
    vector<ll> h;
    for(ll child : edge[u]){
        h.push_back(dfs(child));
    }
    sort(h.begin(), h.end());
    if(id.count(h)) return id[h];
    else return id[h]=id.size();
}

```

## 7.7 trie

```

//01 bitwise trie
struct trie{
    trie *nxt[2]; // 差別
    int cnt; //紀錄有多少個數字以此節點結尾
    int sz; //有多少數字的前綴包括此節點
    trie():cnt(0),sz(0){
        memset(nxt, 0, sizeof(nxt));
    }
};
//創建新的字典樹
trie *root;
void insert(int x){
    trie *now = root; // 每次從根節點開始
    for(int i=22; i>=0; i--){ // 從最高位元開始往低位元走
        now->sz++;
        //cout<<(x>>i&1)<<endl;
        if(now->nxt[x>>i&1] == NULL){ //判斷當前第 i 個位元是 0 還是 1
            now->nxt[x>>i&1] = new trie();
        }
        now = now->nxt[x>>i&1]; //走到下一個位元
    }
    now->cnt++;
    now->sz++;
}

```

## 7.8 Z-value

```

int z[MAXN];
void Z_value(const string& s) { //z[i] = lcp(s[1...], s[i...])
    int i, j, left, right, len = s.size();
    left=right=0; z[0]=len;
    for(i=1; i<len; i++){
        j=max(min(z[i-left], right-i), 0);
        for(; i+j<len&&s[i+j]==s[j]; j++);
    }
}

```

```

z[i]=j;
if(i+z[i]>right) {
    right=i+z[i];
    left=i;
} } }

```

## 7.9 minRotation

```

//rotate(begin(s), begin(s)+minRotation(s), end(s))
int minRotation(string s) {
    int a = 0, N = s.size(); s += s;
    for(int b=0; b<N; b++){
        for(int k=0; k<N; k++){
            if(a+k == b || s[a+k] < s[b+k])
                {b += max(0, k-1); break;}
            if(s[a+k] > s[b+k]) {a = b; break;}
        } return a;
    }
}

```

## 8 DP

### 8.1 數位 dp

```

ll dp[MXN_BIT][PRE_NUM][LIMIT][F0]; //字串位置，根據題目的值，是否上界，前導0
ll dfs(int i, int pre, bool lim, bool f0, const string& str){
    if(v[i][pre][f0][lim]) return dp[i][pre][f0][lim];
    v[i][pre][f0][lim] = true;

    if(i == str.size())
        return dp[i][pre][f0][lim] = 1;

    ll ret = 0, h = lim ? str[i] : '9';

    for(int j='0'; j<=h; j++){
        if(abs(j-pre)>=2 || f0){
            ret += dfs(i+1, j, j==h && lim, f0 && j=='0', str);
        }
    }
    return dp[i][pre][f0][lim] = ret;
}

```

### 8.2 SOS dp

```

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)];
}

```

### 8.3 p-median

```

void p_Median(){
    for (int i=1; i<=N; ++i)
        for (int j=i; j<=N; ++j){
            m = (i+j)/2, d[i][j] = 0; // m是中位數 · d[i][j]為距離的總和
            for (int k=i; k<=j; ++k) d[i][j] += abs(arr[k] - arr[m]);
        }
    for (int p=1; p<=P; ++p)
        for (int n=1; n<=N; ++n){
            dp[p][n] = 1e9;
            for (int k=p; k<=n; ++k)
                if (dp[p-1][k-1] + d[k][n] < dp[p][n]){
                    dp[p][n] = dp[p-1][k-1] + d[k][n];
                    r[p][n] = k; // 從第k個位置往右到第j個位置
                }
        }
}

```

## 9 Other

### 9.1 黑魔法、名次樹

```

#include <bits/extc++.h>
using namespace __gnu_pbds;

```

```

typedef tree<int,null_type,less<int>,rb_tree_tag,
    tree_order_statistics_node_update> set_t;
#include <ext/pb_ds/assoc_container.hpp>
typedef cc_hash_table<int,int> umap_t;
typedef priority_queue<int> heap;
#include<ext/rope>
using namespace __gnu_cxx;
int main(){
    // Insert some entries into s.
    set_t s; s.insert(12); s.insert(505);
    // The order of the keys should be: 12, 505.
    assert(*s.find_by_order(0) == 12);
    assert(*s.find_by_order(3) == 505);
    // The order of the keys should be: 12, 505.
    assert(s.order_of_key(12) == 0);
    assert(s.order_of_key(505) == 1);
    // Erase an entry.
    s.erase(12);
    // The order of the keys should be: 505.
    assert(*s.find_by_order(0) == 505);
    // The order of the keys should be: 505.
    assert(s.order_of_key(505) == 0);

    heap h1 , h2; h1.join( h2 );
    rope<char> r[ 2 ];
    r[ 1 ] = r[ 0 ]; // persistenet
    string t = "abc";
    r[ 1 ].insert( 0 , t.c_str() );
    r[ 1 ].erase( 1 , 1 );
    cout << r[ 1 ].substr( 0 , 2 );
}

```

## 9.2 Hiber Curve

```

long long hilbert(int n,int x,int y){
    long long res=0;
    for(int s=n/2;s;s>>=1){
        int rx=(x&s)>0,ry=(y&s)>0; res+=s*111*s*((3*rx)^ry)
        ;
        if(ry==0){ if(rx==1) x=s-1-x,y=s-1-y; swap(x,y); }
    }
    return res;
}

```





