1 Formula

- Eular Formular: $|V| - |E| + |F| = 2$
- $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$
- $|\bar{A} \cap \bar{B} \cap \bar{C}| = |\Omega| - |A| - |B| - |C| + |A \cap B| + |A \cap C| + |B \cap C| - |A \cap B \cap C|$
- Catalan Number: $C_n = (4n-2)/(n+1)C_{n-1}$

TODO Burnside 引理

- 可以通过 dfs 序列化
- Floyd 算法别忘了设置 dist[i][i] = 0
- 想着比赛的时候可以打表
- 初始化一定不要忘记
- 提交时记得把所有的调试信息都删掉
- 想着可以用二分的方法，把问题转化为判定问题。
- 对于几何问题，没想法就先动手画画图，别上来就解析法。
- 数组一定要开的足够打大，能用 LL 就别用 int
- 数位 dp 一定要写暴力 check
- 最大 (极大) 独立集 + 最小 (极小)(点) 覆盖集 = V
- 最大 (极大) 团 = 补图的最大 (极大) 独立集
- 二分图的最大独立集 = V - 二分图的最大匹配
- 二分图的最大 (点权) 独立集 = SUM - 二分图的最佳匹配
- 二分图的最小 (边权) 覆盖 = 二分图的最佳匹配
- 二分图的最小 (点权) 覆盖 = 最小割 (X-Y 之间的边设为 inf)
- 二分图的最小覆盖 = 二分图的最大匹配
- 注意求递推式的时候可能要用到二项式定理, 如求 $\Sigma_{i-1}^{n} i^k k^i (k \leq 50)$

2 Edit Esp

```
int main() {
    static const int stksz = 10000000;
    static int stk[stksz], espbak;
    __asm__ __volatile__ ( "movl␣%%esp,␣%0\n\tmovl␣%1,␣%%esp\n\t" : "=g"(espbak
        ) : "g"(stk+stksz-1)  );
    solve();
    exit(0);
}
```

3 Java Header

```
import java.io.*;
import java.util.*;
import java.math.*;

class Task {
    void solve( int ri, InputReader in, PrintWriter out ) {
        BigDecimal a = new BigDecimal("23213432.2142143");
        a = a.round( new MathContext(10, RoundingMode.HALF_UP) );
        out.println( a.toPlainString() );
    }
}

public class Main {
    public static void main(String []args) {
        InputStream insm = System.in;
        OutputStream outsm = System.out;
        InputReader in = new InputReader( insm );
        PrintWriter out = new PrintWriter( outsm );
        Task task = new Task();
        task.solve(1, in, out);
        out.close();
    }
}

class InputReader {
    private BufferedReader reader;
```

```
    private StringTokenizer tokenizer;

    public InputReader( InputStream sm ) {
        reader = new BufferedReader( new InputStreamReader(sm) );
        tokenizer = null;
    }

    public String next() {
        while ( tokenizer == null || !tokenizer.hasMoreTokens() ) {
            try {
                tokenizer = new StringTokenizer( reader.readLine() );
            } catch( IOException e ) {
                throw new RuntimeException(e);
            }
        }
        return tokenizer.nextToken();
    }
}
```

4 Dancing Links 精确覆盖 (矩阵处理)

```
const int maxN = 60 * 20, maxM = 60 * 10;
const int max_size = maxN * maxM;
const int inf = 0x3f3f3f3f;
int L[max_size], R[max_size], U[max_size], D[max_size], CH[max_size], RH[
    max_size];
int S[maxM], O[maxM];
int head, size;
int node(int up, int down, int left, int right) {
    U[size] = up, D[size] = down;
    L[size] = left, R[size] = right;
    D[up] = U[down] = R[left] = L[right] = size;
    return size++;
}
bool mat[maxN][maxM];
void init(int N, int M) {
    size = 0;
    head = node(0, 0, 0, 0);
    for (int j = 1; j <= M; ++j) {
        CH[j] = node(size, size, L[head], head), S[j] = 0;
    }
    for (int i = 1; i <= N; ++i) {
        int row = -1, k;
        for (int j = 1; j <= M; ++j) {
            if (!mat[i][j]) continue;
            if (row == -1) {
                row = node(U[CH[j]], CH[j], size, size);
                RH[row] = i, CH[row] = CH[j], ++S[j];
            } else {
                k = node(U[CH[j]], CH[j], L[row], row);
                RH[k] = i, CH[k] = CH[j], ++S[j];
            }
        }
    }
}
void remove(const int &c) {
    L[R[c]] = L[c], R[L[c]] = R[c];
    for (int i = D[c]; i != c; i = D[i]) {
        for (int j = R[i]; j != i; j = R[j]) {
            U[D[j]] = U[j], D[U[j]] = D[j];
            --S[CH[j]];
        }
    }
}
void resume(const int &c) {
    for (int i = U[c]; i != c; i = U[i]) {
        for (int j = L[i]; j != i; j = L[j]) {
            ++S[CH[j]];
            U[D[j]] = D[U[j]] = j;
        }
    }
    L[R[c]] = R[L[c]] = c;
}
int len;
bool DLX(const int &k) {
    if (R[head] == head) {
        len = k - 1;
        return true;
```

```
    }
    int s = inf, c;
    for (int t = R[head]; t != head; t = R[t]) {
        if (S[t] < s) s = S[t], c = t;
    }
    remove(c);
    for (int i = D[c]; i != c; i = D[i]) {
        O[k] = RH[i];
        for (int j = R[i]; j != i; j = R[j]) {
            remove(CH[j]);
        }
        if (DLX(k + 1)) {
            return true;
        }
        for (int j = L[i]; j != i; j = L[j]) {
            resume(CH[j]);
        }
    }
    resume(c);
    return false;
}
```

5 Dancing Links 重复覆盖 (矩阵处理)

```
const int head = 0;
const int INF=10000000;
const int maxn = 1700;
const int maxd = 1000000;
int N, M, K, n, m, cnt, res;
int mat[maxn][maxn], s[maxn], l[maxd], r[maxd], u[maxd], d[maxd], c[maxd], o[
    maxn], row[maxd];
bool use[maxn];
void makegragh(int &n, int &m) {
    memset(mat, 0, sizeof(mat));
    //init
}
void initial(int n, int m) {
    memset(use, false, sizeof(use));
    res = n + 1;
    int i, j, rowh;
    memset(s, 0, sizeof(s));
    for(i=head; i<=m; i++) {
        r[i]=(i+1)%(m+1);
        l[i]=(i-1+m+1)%(m+1);
        u[i]=d[i]=i;
    }
    cnt=m+1;
    for(i=1; i<=n; i++) {
        rowh=-1;
        for(j=1; j<=m; j++) {
            if(mat[i][j])
            {
                s[j]++; u[cnt]=u[j]; d[u[j]]=cnt;
                u[j]=cnt; d[cnt]=j; row[cnt]=i; c[cnt]=j;
                if(rowh==-1) {
                    l[cnt]=r[cnt]=cnt; rowh=cnt;
                }
                else {
                    l[cnt] = l[rowh]; r[l[rowh]] = cnt;
                    r[cnt] = rowh; l[rowh] = cnt;
                }
                cnt++;
            }
        }
    }
}
void remove(int c) {
    for(int i=d[c]; i!=c; i=d[i]) {
        r[l[i]]=r[i]; l[r[i]]=l[i];
    }
}
void resume(int c) {
    for(int i=d[c]; i!=c; i=d[i])
        r[l[i]]=l[r[i]]=i;
}
int h() {
    bool has[maxn];
    memset(has, false, sizeof(has));
```

```
    int ans=0;
    for(int i=r[head]; i!=head; i=r[i])
        if(!has[i]) {
            ans++;
            for(int j=d[i]; j!=i; j=d[j])
                for(int k=r[j]; k!=j; k=r[k])
                    has[c[k]]=true;
        }
    return ans;
}
bool dfs(int k) {
    if(k+h()>=res)return false;//A* cut
    if(r[head]==head) {
        if(k<res) res=k;
        return true;
    }
    int ms=INF, cur=0;
    for(int t=r[head]; t!=head; t=r[t])
        if(s[t]<ms) {
            ms=s[t]; cur=t;
        }
    for(int i=d[cur]; i!=cur; i=d[i]) {
        remove(i);
        for(int j=r[i]; j!=i; j=r[j]) {
            remove(j); s[c[j]]--;
        }
        dfs(k+1);
        for(int j=l[i]; j!=i; j=l[j]) {
            resume(j); s[c[j]]++;
        }
        resume(i);
    }
    return false;
}
```

6 dinic

```
#include <cstring>
#include <algorithm>
using namespace std;
typedef long long LL;

const int maxn = 5001;
const int maxe = 60200;
const int inf = 2000000000; // >= maxc * 2;

struct node {
    int b, c; node *next, *anti;
} *ge[maxn], pool[maxe], *pooltp;

void init( int n ) {
    pooltp = pool; for ( int i = 0; i <= n; ++i ) ge[i] = 0;
}

node *_ins( int a, int b, int c ) {
    node *p = pooltp++; p->b = b; p->c = c; p->next = ge[a]; return ge[a] = p;
}

void ins1( int a, int b, int c ) {
    node *p = _ins( a, b, c ), *q = _ins(b, a, 0);
    p->anti = q; q->anti = p;
}

bool bfs( int n, int s, int t, int dist[] ) {
    static int q[maxn], *qt, *qb;
    qb = qt = q;
    memset( dist, -1, sizeof(dist[0]) * (n+1) );
    dist[s] = 0; *qt++ = s;
    for(; qt != qb; qb++) {
        for ( node *p = ge[*qb]; p; p = p->next ) {
            if ( p->c && dist[p->b] == -1 ) {
                dist[p->b] = dist[*qb] + 1;
                *qt++ = p->b;
                if ( p->b == t ) return true;
            }
        }
    }
    return false;
```

```
}

LL maxflow( int n, int s, int t ) {
    static int dist[maxn], pre[maxn];
    static node *cur[maxn], *path[maxn];

    LL tot = 0;
    while ( bfs(n, s, t, dist) ) {
        memcpy( cur, ge, sizeof(ge[0]) * (n+1) );
        for ( int i = s; dist[s] != -1; ) {
            if ( i == t ) {
                int flow = inf;
                for (; i != s; i = pre[i]) flow = min( flow, path[i]->c );
                tot += flow;
                for ( int i = t; i != s; i = pre[i] ) {
                    path[i]->c -= flow; path[i]->anti->c += flow;
                }
            }
            for ( node *&p = cur[i]; p; p = p->next ) {
                int v = p->b;
                if ( p->c && dist[v] == dist[i] + 1 ) {
                    pre[v] = i; path[v] = 0; i = v; break;
                }
            }
            if ( cur[i] == 0 ) {
                dist[i] = -1; i = pre[i];
            }
        }
    }
    return tot;
}

int main() {
}
```

7 costflow

```
const int maxn = 5001 * 2, maxe = 60200 * 5;
const int inf = 20000000000; // >= maxc * 2

struct node {
    int b, c, w; node *next, *anti;
} *ge[maxn], pool[maxe], *pooltp;

void init(int n) {
    pooltp = pool; for ( int i = 0; i <= n; ++i ) ge[i] = 0;
}

inline node* _ins(int a, int b, int c, int w) {
    node *p = pooltp++; p->b = b; p->c = c; p->w = w; p->next = ge[a]; ge[a] =
        p; return p;
}

inline void ins1(int a, int b, int c, int w) {
    node *p = _ins(a, b, c, w), *q = _ins(b, a, 0, -w);
    p->anti = q; q->anti = p;
}

complex<LL> aug(int n, int s, int t, int lim) {
    static int q[maxn], *qt, *qb, inq[maxn], dis[maxn], pre[maxn];
    static node *path[maxn];
#define enq(x) { *qt++ = x; if (qt == q + maxn) qt = q; inq[x] = 1; }
#define deq(x) { x = *qb++; if (qb == q + maxn) qb = q; inq[x] = 0; }
    qb = qt = q; enq(s);
    rep(i, n+1) dist[i] = 0; dist[s] = 0;
    while (qb != qt) {
        int u; deq(u);
        for (node *p = ge[u]; p; p = p->next) {
            if (p->c > 0 && dist[p->b] > dist[u] + p->w) {
                dist[p->b] = dist[u] + p->w;
                pre[p->b] = u; path[p->b] = p;
                if (!inq[p->b]) enq(p->b);
            }
        }
    }
    LL flow = lim, cost = 0;
    if ( dist[t] == inf ) return complex<LL>(0, 0);
    for(int i = t; i != s; i = pre[i])
```

```
        flow = min<LL>(flow, path[i]->c);
    for(int i = t; i != s; i = pre[i]) {
        cost += flow * path[i]->w;
        path[i]->c -= flow; path[i]->anti->c += flow;
    }
    return complex<LL>(flow, cost);
}

complex<LL> mincostmaxflow(int n, int s, int t, int lim = inf) {
    complex<LL> ret = 0, del;
    while ( (del = aug(n, s, t, lim)).real() > 0 ) {
        ret += del; lim -= del.real();
    }
    return ret;
}
```

8 planarmincut

```
#include <vector>
#include <iostream>
#include <set>
#include <cmath>
using namespace std;

typedef long long LL;
#define mp make_pair
#define pb push_back
#define rep(i, n) for(int i = 0; i < (n); ++i)

template<class T> inline void chkmin(T& a, const T& b) { if (a > b) a = b; }
template<class T> inline void chkmax(T& a, const T& b) { if (a < b) a = b; }

typedef pair<int, int> Point;
#define x first
#define y second

const int inf = 1000000000;
const int maxn = 100000 * 2 + 5;
const int maxe = maxn * 2 + 5;

struct Graph {
    vector< pair<int, int> > ge[maxn]; int n;
    void init(int nn) {
        n = nn; rep(i, n) ge[i].clear();
    }
    void ins2(int a, int b, int c) {
        ge[a].pb( mp(b, c) ); ge[b].pb( mp(a, c) );
    }
    LL sssp(int s, int t) {
        set< pair<LL, int> > h;
        static LL dis[maxn];
        for ( int i = 0; i < n; ++i ) {
            dis[i] = i == s ? 0 : inf;
            h.insert( mp(dis[i], i) );
        }
        while ( !h.empty() ) {
            int u = h.begin()->second; h.erase(h.begin());
            for (int k = 0; k < ge[u].size(); ++k) {
                int v = ge[u][k].first, d = ge[u][k].second;
                if ( dis[v] > dis[u] + d ) {
                    h.erase( mp(dis[v], v) );
                    dis[v] = dis[u] + d;
                    h.insert( mp(dis[v], v) );
                }
            }
        }
        return dis[t];
    }
} graph;

struct MaxflowPlanar {
    Point p[maxn];
    int n, ecnt, fcnt;

    struct edge {
        int a, b, c, vis, find;
        edge *prev, *anti;
        double ang;
```

```
    edge *next() {
        return anti->prev;
    }
    void init(int aa, int bb, int cc, double aang, edge *aanti) {
        a = aa; b = bb; c = cc; ang = aang, anti = aanti;
        vis = 0;
    }
} e[maxe], *ptr[maxe];

struct Cmp {
    bool operator()(const edge *x, const edge* y) const {
        if ( x->a != y->a ) return x->a < y->a;
        return x->ang < y->ang;
    }
};

void init(Point q[], int nn) {
    n = nn; ecnt = fcnt = 0; copy(q, q + n, p);
}

void ins2(int a, int b, int c) {
    int dy = p[b].y - p[a].y, dx = p[b].x - p[a].x;
    e[ecnt].init(a, b, c, atan2l(dy, dx), &e[ecnt^1]); ++ecnt;
    e[ecnt].init(b, a, c, atan2l(-dy, -dx), &e[ecnt^1]); ++ecnt;
}
LL maxflow() {
    for (int i = 0; i < ecnt; ++i) ptr[i] = e + i;
    sort( ptr, ptr + ecnt, Cmp() );
    for (int i = 0, j; i < ecnt; i = j) {
        for (j = i + 1; j < ecnt && ptr[i]->a == ptr[j]->a; ++j);
        for (int k = i; k < j; ++k) ptr[k]->prev = ptr[k-1];
        ptr[i]->prev = ptr[j-1];
    }
    for (int i = 0; i < ecnt; ++i) {
        if (ptr[i]->vis) continue;
        ptr[i]->find = fcnt; ptr[i]->vis = 1;
        for (edge* p = ptr[i]->next(); p != ptr[i]; p = p->next() )
            p->find = fcnt, p->vis = 1;
        ++fcnt;
    }
    graph.init(fcnt);
    int s = -1, t = -1;
    for (int i = 0; i < ecnt; ++i) {
        if (ptr[i]->c != inf) {
            graph.ins2(ptr[i]->find, ptr[i]->anti->find, ptr[i]->c);
        } else if (s == -1) {
            s = ptr[i]->find, t = ptr[i]->anti->find;
        }
    }
    return graph.sssp(s, t);
}
} flow;

void solve() {
    int n, m; cin >> n >> m;
    static Point p[maxn];
    int maxY = -inf, minY = inf;
    for (int i = 0; i < n; ++i) {
        scanf("%d%d", &p[i].x, &p[i].y);
        chkmin(minY, p[i].y); chkmax(maxY, p[i].y);
    }
    int s = min_element(p, p + n) - p, t = max_element(p, p + n) - p;
    p[n] = mp(p[s].x-1, maxY+1); p[n+1] = mp(p[t].x+1, maxY+1); //??

    flow.init(p, n + 2);
    flow.ins2(s, n, inf); flow.ins2(n, n + 1, inf); flow.ins2(n+1, t, inf);
    for (int i = 0; i < m; ++i) {
        int a, b, c; scanf("%d%d%d", &a, &b, &c);
        flow.ins2(a-1, b-1, c);
    }
    cout << flow.maxflow() << endl;
}

int main() { int re; cin >> re; while (re--) solve(); }
```

9 kosaraju

```
const int maxn = 500000 + 5;
```

```
const int inf = 2000000000;
int vis[maxn], order[maxn], group[maxn], cnt;

struct node { int b; node *next;
} *ge[maxn], *gr[maxn], *gg[maxn], pool[maxn * 10], *pooltp = pool;

void dfs(int u) {
    vis[u] = 1;
    for (node *p = ge[u]; p; p = p->next) {
        int v = p->b; if (!vis[v]) dfs(v);
    }
    order[cnt++] = u;
}

void rfs(int u) {
    vis[u] = 1; group[u] = cnt;
    for (node *p = gr[u]; p; p = p->next) {
        int v = p->b; if (!vis[v]) rfs(v);
    }
}

int scc(int n) {
    cnt = 0; clr(vis, 0, n+1);
    rep(i, n) if (!vis[i]) dfs(i); //may be changed to 1..n
    cnt = 0; clr(vis, 0, n+1);
    for (int i = n-1; i >= 0; --i) {
        int u = order[i]; if (!vis[u]) {
            rfs(u); ++cnt;
        }
    }
    return cnt;
}

#define ins(ge, a, b) {\
    node *_p = pooltp++; _p->b = b; _p->next = ge[a]; ge[a] = _p; }

int val[maxn], dest[maxn];
int group_val[maxn], group_dest[maxn];
int dp[maxn];

int main() {
    int n, m, a, b, s;
    while (cin >> n >> m) {
        clr(ge, 0, n); clr(gr, 0, n); clr(dest, 0, n);
        pooltp = pool;
        rep(i, m) {
            scanf("%d%d", &a, &b); --a; --b;
            ins(ge, a, b); ins(gr, b, a);
        }
        rep(i, n) {
            scanf("%d", &val[i]); dest[--b] = 1;
        }
        scanf("%d%d", &s, &a); --s;
        rep(i, a) {
            scanf("%d", &b); dest[--b] = true;
        }
        scc(n);
        rep(i, cnt) {
            group_val[i] = group_dest[i] = 0;
            dp[i] = -inf; gg[i] = 0;
        }
        s = group[s];
        rep(i, n) {
            group_dest[ group[i] ] |= dest[i];
            group_val[ group[i] ] += val[i];
            for (node *p = ge[i]; p; p = p->next) {
                if ( group[i] == group[p->b] ) continue;
                ins(gg, group[i], group[p->b]);
            }
        }
        static int q[maxn], *qt, *qb, inq[maxn];
#define enq(x) { *qt++ = x; if (qt == q + maxn) qt = q; inq[x] = 1; }
#define deq(x) { x = *qb++; if (qb == q + maxn) qb = q; inq[x] = 0; }
        clr(inq, 0, cnt); qb = qt = q;
        enq(s); dp[s] = group_val[s];
        while (qb != qt) {
            int u; deq(u);
            for (node *p = gg[u]; p; p = p->next) {
```

```
                if (dp[p->b] < dp[s] + group_val[p->adj]) {
                    dp[p->b] = dp[s] + group_val[p->b];
                    if (!inq[p->b]) enq(p->b);
                }
            }
        }
        int maxval = 0;
        rep(i, cnt) if ( group_dest[i] && dp[i] > maxval )
            maxval = dp[i];
        cout << maxval << endl;
    }
}
```

10 kmp AND exkmp

```
void preMP(const char x[], int m, int next[]) {
    int i, j;
    i = next[0] = -1; j = 0;
    while (j < m) {
        while (i > -1 && x[i] != x[j]) i = next[i];
        next[++j] = ++i;
    }
}
int kmp(const char x[], int m, const char y[], int n) {
    int i = 0, j = 0, ret;
    preMP(x, m, next);
    while(j < n) {
        while (i > -1 && x[i] != y[j]) i = next[i];
        ++i; ++j;
        if (i >= m) {
            //OUTPUT(j - i)
            ret++; i = next[i];
        }
    }
    return ret;
}
void prez(const char x[], int m, int next[]) {
    int j, k = 1, r = 0; next[0] = m;
    for (int i = 1; i < m; ++i) {
        if ( i + next[i-k] < r ) {
            next[i] = next[i-k];
        } else {
            for (j = max(r-i, 0); i + j < m && x[i+j] == x[j]; ++j);
            next[i] = j; k = i; r = i + j;
        }
    }
}

//next[i]: lcp of x[i..m-1] and x[0..m-1]
//ext[i]:  lcp of y[i..n-1] and x[0..m-1]
void z(const char x[], int m, const char y[], int n, int next[], int ext[]) {
    int k = 0, r = 0, j;
    prez(x, m, next); next[0] = 0;
    for (int i = 0; i < n; ++i) {
        if ( i + next[i-k] < r ) {
            ext[i] = next[i-k];
        } else {
            for (j = max(r-i, 0); j < m && i + j < n && x[j] == y[i+j]; ++j);
            ext[i] = j; k = i; r = i + j;
        }
    }
}
```

11 FFT

```
#include <algorithm>
#include <cassert>
using namespace std;
typedef long long LL;

#define rep(i, n) for (int i = 0; i < (n); ++i)
const int maxn = 10000;

struct Zp {
    const LL mod; const int pri;
    Zp(LL mod, int pri) : mod(mod), pri(pri) {
    }
    Zp(LL mod) : mod(mod), pri( primitive() ) {
```

```
}
LL add(LL a, LL b) {
    a += b; return a >= mod ? a - mod : a;
}
LL sub(LL a, LL b) {
    a -= b; return a < 0 ? a + mod : a;
}
LL mul(LL a, LL b) {
    if ( mod <= 1000000000 ) return a * b % mod;
    LL t = (LL)( (double)a * b / mod + 0.5 );
    LL r = ( a * b - t * mod ) % mod;
    return r >= 0 ? r : r + mod;
}
LL pow(LL a, LL b) {
    LL r = 1;
    for (;b;) {
        if ( b & 1 ) r = mul(r, a);
        if ( b >>= 1 ) a = mul(a, a);
    }
    return r;
}
LL inv(LL a) {
    return pow(a, mod - 2);
}
void fft(int n, LL root, LL a[]) {
    for (int m = n; m >= 2; m >>= 1) {
        int mh = m >> 1; LL w = 1;
        for (int i = 0; i < mh; ++i) {
            for (int j = i; j < n; j += m) {
                int k = j + mh;
                LL t = sub(a[j], a[k]);
                a[j] = add(a[j], a[k]);
                a[k] = mul(w, t);
            }
            w = mul(w, root);
        }
        root = mul(root, root);
    }
    for (int j = 1, i = 0; j < n - 1; ++j) {
        for (int k = n >> 1; k > (i ^= k); k >>= 1);
        if (j < i) swap(a[i], a[j]);
    }
}
void dft(const LL a[], int an, LL b[], int n) {
    LL root = pow(pri, mod / n);
    copy(a, a + an, b); fill(b + an, b + n, 0);
    fft(n, root, b);
}
void nft(const LL a[], LL b[], int n) {
    LL root = pow(pri, mod / n); root = inv(root);
    copy(a, a + n, b);
    fft(n, root, b);
    LL invn = inv(n);
    rep(i, n) b[i] = mul(b[i], invn);
}
int primitive() {
    int n = mod - 1;
    LL p[25], pcnt = 0;
    for (LL i = 2; i * i <= n; ++i) {
        if ( n % i == 0 ) {
            do n /= i; while (n % i == 0);
            p[pcnt++] = i;
        }
    }
    if (n > 1) p[pcnt++] = n;
    for (int g = 2;;++g) {
        int ok = 1; //assert( pow(g, mod-1) == 1 );
        rep(i, pcnt) if ( pow(g, (mod-1)/p[i]) == 1 ) {
            ok = 0; break;
        }
        if (ok) return g;
    }
}
} zp(0xb1a2bc2edc0001LL, 3);

struct poly {
    const static int maxn = ::maxn * 4 + 5;
    LL a[maxn]; int n;
```

```
    template<class T> void init(const T a[], int n) {
        this->n = n; copy(a, a + n, this->a);
    }
    LL eval(LL x) const {
        LL ans = 0;
        for (int i = n - 1; i >= 0; --i)
            ans = zp.add( zp.mul(ans, x), a[i] );
        return ans;
    }
    friend void mul(poly& r, const poly& x, const poly& y) {
        static LL xb[maxn], yb[maxn];
        int n = 1; while ( n < x.n + y.n ) n *= 2;
        LL root = zp.pow( zp.pri, zp.mod / n );
        zp.dft( x.a, x.n, xb, n );
        rep(i, n) assert( x.eval(zp.pow(root, i)) == xb[i] );
        zp.dft( y.a, y.n, yb, n );
        rep(i, n) assert( y.eval(zp.pow(root, i)) == yb[i] );
        rep(i, n) xb[i] = zp.mul(xb[i], yb[i]);
        zp.nft(xb, r.a, n);
        r.n = n; while (r.n > 0 && r.a[r.n-1] == 0) --r.n;
    }
};

struct mp { //BigUnsignedInteger
    static const int digit = 4;
    static const int base = 10000;
    static const int cap = 50000 * 2 + 5; // 10 ^ 500
    static const int maxn = cap / digit + 1;
    int dat[maxn], n;

    mp(const mp& o) : n(o.n) {
        copy(o.dat, o.dat + n, dat);
    }
    mp(LL v = 0) {
        for (n = 0; v; v /= base) dat[n++] = v % base;
    }
    void parse(const char *s) {
        n = 0;
        for (int i = strlen(s) - 1, v = 0, m = 1; i >= 0; --i) {
            v = v + (s[i] - '0') * m; m *= 10;
            if (m == base || i == 0) {
                dat[n++] = v; v = 0; m = 1;
            }
        }
    }
    char *toString(char *s) const {
        if (n == 0) {
            sprintf(s, "0");
        } else {
            char *p = s;
            p += sprintf(p, "%d", dat[n-1]);
            for (int i = n - 2; i >= 0; --i)
                p += sprintf(p, "%0*d", digit, dat[i]);
        }
        return s;
    }
    char *toString() const {
        static char buf[cap + 5]; return toString(buf);
    }
    friend void add(mp& r, const mp& x, const mp& y) {
        int i = 0;
        for (int t = 0; i < x.n || i < y.n || t; ++i, t /= base) {
            if (i < x.n) t += x.dat[i];
            if (i < y.n) t += y.dat[i];
            r.dat[i] = t % base;
        }
        r.n = i;
    }
    friend void sub(mp& r, const mp& x, const mp& y) {
        r.n = x.n;
        for (int i = 0, t = 0; i < r.n; ++i) {
            r.dat[i] = x.dat[i] - t;
            if ( i < y.n ) r.dat[i] -= y.dat[i];
            if ( r.dat[i] < 0 ) {
                t = 1; r.dat[i] += base;
            } else {
                t = 0;
            }
```

```
        }
        while (r.n && r.dat[r.n - 1] == 0) --r.n;
    }
    friend void mul(mp& r, const mp& x, int y) {
        int i = 0;
        for (LL t = 0; i < x.n || t; ++i, t /= base) {
            if (i < x.n) t += (LL)(x.dat[i]) * y;
            r.dat[i] = t % base;
        }
        r.n = i;
    }
    friend void mulfft(mp& r, const mp& x, const mp& y) {
        static poly px, py, pr;
        px.init(x.dat, x.n);
        py.init(y.dat, y.n);
        mul(pr, px, py);
        int i = 0;
        for (LL t = 0; i < pr.n || t; ++i, t /= base) {
            if (i < pr.n) t += pr.a[i];
            r.dat[i] = t % base;
        }
        r.n = i;
    }
    friend void div(mp& q, int &r, const mp& x, int y) {
        q.n = x.n; r = 0;
        for (int i = x.n - 1; i >= 0; --i, r %= y) {
            r = r * base + x.dat[i];
            q.dat[i] = r / y;
        }
        while (q.n && q.dat[q.n-1] == 0) --q.n;

    }
};

int main() {
    static mp x, y, z;
    static char buf[1000000];
    while ( gets(buf) ) {
        x.parse(buf);
        gets(buf); y.parse(buf);
        mulfft(z, x, y);
        puts( z.toString() );
    }
}
```

12 isprime

```
bool witness(LL a, LL n) {
    int k = 0;
    LL m = n - 1;
    do {m /= 2; k++; } while (m % 2 == 0);
    LL x = pow_mod(a, m, n);
    if (x == 1) return true;
    for (int i = 0; i < k; x = mul_mod(x, x, n); ++i)
        if (x == n - 1) return true;
    return false;
}

bool miller_rabin(LL n, int time = 50) {
    if (2 == n || 3 == n || 5 == n || 7 == n) return true;
    if (1 == n || n % 2 == 0 || n % 3 == n || n % 5 == 0 || n % 7 == 0) return
        false;
    while (time--) {
        LL r = rand() % (n-2) + 2;
        if ( gcd(r, n) != 1 || !witness(r % n, n) ) return false;
    }
    return true;
}
```

13 rho

```
LL rho(LL n) {
    LL x, y, d, c;
    for (int k, i;;) {
        c = rand() % (n - 1) + 1;
        x = y = rand() % n;
        k = 2; i = 1;
```

```
        do {
            d = gcd( ABS(x - y), n );
            if ( d > 1 && d < n ) return d;
            if ( ++i == k ) y = x, k *= 2;
            x = mul_mod(x, x, n); x = (x + c) % n;
        } while ( x != y );
    }
}
```

14 crt

```
bool crt2(T &dd, T& rr, T d1, T r1, T d2, T r2) {
    T q1, q2, g = exgcd(d1, d2, q1, q2);
    T c = r1 - r2; if (c < 0) c += d1;
    dd = d1 / g * d2;
    if (c % g) { rr = -1; return false; }
    T t = d1 / g;
    q2 *= c / g; q2 %= t; if (q2 <= 0) q2 += t;
    rr = q2 * d2 + r2; if (rr >= dd) rr -= dd;
    return true;
}

bool crt(T& dd, T& rr, T d[], T r[], int n) {
    dd = 1, rr = 0;
    rep(i, n) if (!crt2(dd, rr, dd, rr, d[i], r[i])) return false;
    return true;
}
```

15 log

```
struct Zn {
    const LL n;
    Zn(LL nn) : n(nn) {
    }
    LL eval(LL a) {
        a %= n; return a >= 0 ? a : a + n;
    }
    LL mul(LL a, LL b) {
        if (n <= 1000000000) return a * b % n;
        assert(0);
    }
    LL pow(LL a, LL b) {
        LL r = 1 % n;
        for (;b;) {
            if (b & 1) r = mul(r, a);
            if (b >>= 1) a = mul(a, a);
        }
        return r;
    }
    LL inv(LL a) {
        LL x, y, d = exgcd(a, n, x, y);
        assert(d == 1);
        return eval(x);
    }
    LL log(LL a, LL b);
};

struct Zp : Zn {
    Zp(LL n) : Zn(n) {
    }
    const static int maxsqrtn = 100000 + 5;
    static int id[]; static LL mexp[];

    struct logcmp {
        bool operator()(int a, int b) { return mexp[a] < mexp[b]; }
    };
    LL log(LL a, LL b) { //a ^ x = b
        int m = (int)( ceil( sqrt(n) ) );
        LL v=  inv( pow(a, m) );
        id[0] = 0; mexp[0] = 1;
        for (int i = 1; i <= m; ++i) {
            id[i] = i; mexp[i] = mul(mexp[i-1], a);
        }
        stable_sort(id + 1, id + m + 1, logcmp());
        sort(mexp + 1, mexp + m + 1);
        for (int i = 0; i < m; ++i) {
            int j = lower_bound(mexp, mexp + m + 1, b) - mexp;
```

```
            if (j <= m && mexp[j] == b) return i * m + id[j];
            b = mul(b, v);
        }
        return -1;
    }
}

LL Zn::log(LL a, LL b) { //a ^ x = b
    for (int i = 0; i <= 50; ++i) if ( pow(a, i) == b ) return i;
    LL g, d = 1, n = this->n, x = 0;
    while ( (g = gcd(a, n)) > 1 ) {
        if ( b % g ) return -1;
        b /= g; n /= g; d = mul(d, a/g); ++x;
    }
    Zp zp(n); LL ans = zp.log( a, zp.mul(b, zp.inv(d)) );
    return ans == -1 ? -1 : ans + x;
}

int Zp::id[Zp::maxsqrtn]; LL Zp::maxp[ Zp::maxsqrtn ];
```

16 romberg

```
real f(real x) {
    return exp(-x * x);
}
//O(2 ^ maxitr) function evaluations
real Romberg(real a, real b, real(*f)(real), real eps, int maxitr = 20) {
    real T[maxitr][maxitr];
    for (int i = 0; i < maxitr; ++i) {
        real h = (b - a) / (1 << i), x = a + h, pow = 4;
        T[i][0] = (f(a) + f(b)) / 2;
        for (int j = (1 << i) - 1; j >= 1; x += h, --j) T[i][0] += f(x); T[i
            ][0] *= h;
        for (int j = 1; j <= i; pow *= 4, ++j)
            T[i][j] = T[i][j - 1] + (T[i][j - 1] - T[i - 1][j - 1]) / (pow - 1)
                ; if (i > 0 && fabs(T[i][i] - T[i - 1][i - 1]) <= eps) return T
                [i][i];
    }
    return T[maxitr - 1][maxitr - 1];
}
```

leftist

```
//TODO
```

treap by HL 迪卡尔树 SA AC 自动机 Manacher