

# Contents

<b>1 Basic</b>	<b>1</b>
1.1 compile . . . . .	1
1.2 default code . . . . .	1
1.3 debug list . . . . .	1
<b>2 Dark Code</b>	<b>1</b>
2.1 IO optimization . . . . .	1
<b>3 Flow</b>	<b>2</b>
3.1 Dinic . . . . .	2
3.2 min cost flow . . . . .	2
<b>4 Mathematics</b>	<b>3</b>
4.1 ax+by=gcd(a,b) . . . . .	3
4.2 BigInt . . . . .	3
4.3 FFT . . . . .	4
4.4 FWHT . . . . .	4
4.5 GaussElimination . . . . .	4
4.6 Inverse . . . . .	5
4.7 LinearPrime . . . . .	5
4.8 Miller Rabin . . . . .	5
4.9 Pollard's rho . . . . .	5
4.10 數論基本工具 . . . . .	5
4.11 Mobius . . . . .	6
4.12 Simplex . . . . .	6
4.13 SG . . . . .	7
4.14 Theorem . . . . .	7
<b>5 Graph</b>	<b>7</b>
5.1 BCC . . . . .	7
5.2 Prim . . . . .	8
5.3 Kruskal . . . . .	8
5.4 Dijkstra . . . . .	8
5.5 Strongly Connected Component(SCC) . . . . .	8
5.6 Hungarian . . . . .	9
5.7 KM . . . . .	9
5.8 最小平均環 . . . . .	10
5.9 偵測負環 . . . . .	10
5.10 Tarjan . . . . .	10
5.11 Topological Sort . . . . .	11
<b>6 Data Structure</b>	<b>11</b>
6.1 2D Range Tree . . . . .	11
6.2 Sparse Table . . . . .	12
6.3 Segment Tree . . . . .	12
6.4 Lazy Tag . . . . .	12
<b>7 String</b>	<b>12</b>
7.1 AC 自動機 . . . . .	12
7.2 KMP . . . . .	13
7.3 迴文自動機 . . . . .	13
7.4 Suffix Automaton . . . . .	13
7.5 smallest rotation . . . . .	14
7.6 Suffix Array . . . . .	14
7.7 Z-value . . . . .	14
<b>8 Others</b>	<b>14</b>
8.1 矩陣數定理 . . . . .	14
8.2 CYK . . . . .	15
8.3 數位統計 . . . . .	16
8.4 1D/1D dp 優化 . . . . .	16
8.5 Theorem - DP optimization . . . . .	16
8.6 Stable Marriage . . . . .	16
8.7 parser . . . . .	17
8.8 python 小抄 . . . . .	18
<b>9 Persistence</b>	<b>18</b>

## 1 Basic

### 1.1 compile

```
# preset before coding
echo "cd ~/Desktop" >> ~/.bashrc
gedit -> preference -> tab width: 4

# Editor
gedit a.cpp

# Compile
g++ a.cpp -std=c++11
```

```
**All file will be compiled to a.out unless you use -o(
    not recommended, just use a.out)**
# Run
./a.out

# Run with file input
./a.out < input.txt

# Run with file input and output
./a.out < input.txt > output.txt

# Python Run
python3 a.py < input.txt > output.txt

# Copy Paste In Ubuntu
* copy: ctrl+insert
* paste: shift+insert
```

### 1.2 default code

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> pii;
#ifdef ONLINE_JUDGE
#define cerr if(false) cerr
#endif

int main(){
#ifdef ONLINE_JUDGE
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);
    freopen("debug.txt","w",stderr);
#else
    ios_base::sync_with_stdio(0);
    cin.tie(false);
#endif
}
```

### 1.3 debug list

模板要記得 init  
priority\_queue 要清空  
把邊界條件都加入測資  
邊界條件 (過程溢位, 題目數據範圍), 會不會爆 long long  
是否讀錯題目, 想不到時可以自己讀一次題目  
比較容易有問題的地方換人寫  
注意公式有沒有推錯或抄錯  
精度誤差 sqrt(大大的東西) + EPS  
測試 %lld or %I64d  
喇分 random\_shuffle 隨機演算法

## 2 Dark Code

### 2.1 IO optimization

```
*if output to much, consider put all output in array
    first, then output the array.
getchar() -> getchar_unlocked()
fread() -> fread_unlocked()
-----
inline char readchar() {
    const int S = 1<<20; // buffer size
    static char buf[S], *p = buf, *q = buf;
    if(p == q && (q = (p=buf)+fread(buf,1,S,stdin)) ==
        buf) return EOF;
    return *p++;
}
```

```

inline int nxtint() {
    // if readchar can't use, change readchar() to
    // getchar()
    int x = 0;
    int c = readchar(), neg = false;
    if (c == EOF) return -1;
    while (('0' > c || c > '9') && c != '-' && c != EOF)
        c = readchar();
    if (c == '-') neg = true, c = readchar();
    while ('0' <= c && c <= '9') x = x * 10 + (c ^ '0'),
        c = readchar();
    if (neg) x = -x;
    return x;
}

```

## 3 Flow

### 3.1 Dinic

(a) Bounded Maxflow Construction:

1. add two node ss, tt
2. add\_edge(ss, tt, INF)
3. for each edge u -> v with capacity [l, r]:
  - add\_edge(u, tt, l)
  - add\_edge(ss, v, l)
  - add\_edge(u, v, r-l)
4. see (b), check if it is possible.
5. answer is maxflow(ss, tt) + maxflow(s, t)

(b) Bounded Possible Flow:

1. same construction method as (a)
2. run maxflow(ss, tt)
3. for every edge connected with ss or tt:
  - rule: check if their rest flow is exactly 0
4. answer is possible if every edge do satisfy the rule
5. otherwise, it is NOT possible.

(c) Bounded Minimum Flow:

1. same construction method as (a)
2. answer is maxflow(ss, tt)

(d) Bounded Minimum Cost Flow:

- \* the concept is somewhat like bounded possible flow.
1. same construction method as (a)
  2. answer is maxflow(ss, tt) + ( $\sum$  1 \* cost for every edge)

(e) Minimum Cut:

1. run maxflow(s, t)
2. run cut(s)
3. ss[i] = 1: node i is at the same side with s.

```

const long long INF = 1LL<<60;
struct Dinic { //O(VVE), with minimum cut
    static const int MAXN = 5003;
    struct Edge{
        int u, v;
        long long cap, rest;
    };

    int n, m, s, t, d[MAXN], cur[MAXN];
    vector<Edge> edges;
    vector<int> G[MAXN];

    void init(){
        edges.clear();
        for (int i = 0 ; i < MAXN ; i++) G[i].clear();
    }

    // min cut start
    bool side[MAXN];

```

```

void cut(int u) {
    side[u] = 1;
    for (int i : G[u]) {
        if ( !side[ edges[i].v ] && edges[i].rest )
            cut(edges[i].v);
    }
}
// min cut end

```

```

void add_edge(int u, int v, long long cap){
    edges.push_back( {u, v, cap, cap} );
    edges.push_back( {v, u, 0, 0LL} );
    m = edges.size();
    G[u].push_back(m-2);
    G[v].push_back(m-1);
}

```

```

bool bfs(){
    memset(d, -1, sizeof(d));
    queue<int> que;
    que.push(s); d[s]=0;
    while (!que.empty()){
        int u = que.front(); que.pop();
        for (int ei : G[u]){
            Edge &e = edges[ei];
            if (d[e.v] < 0 && e.rest > 0){
                d[e.v] = d[u] + 1;
                que.push(e.v);
            }
        }
    }
    return d[t] >= 0;
}

```

```

long long dfs(int u, long long a){
    if ( u == t || a == 0 ) return a;
    long long flow = 0, f;
    for (int &i=cur[u]; i < (int)G[u].size() ; i++)
    {
        Edge &e = edges[ G[u][i] ];
        if ( d[u] + 1 != d[e.v] ) continue;
        f = dfs(e.v, min(a, e.rest) );
        if ( f > 0 ) {
            e.rest -= f;
            edges[ G[u][i]^1 ].rest += f;
            flow += f;
            a -= f;
            if ( a == 0 ) break;
        }
    }
    return flow;
}

```

```

long long maxflow(int s, int t){
    this->s = s, this->t = t;
    long long flow = 0, mf;
    while ( bfs() ){
        memset(cur, 0, sizeof(cur));
        while ( (mf = dfs(s, INF)) ) flow += mf;
    }
    return flow;
}
} dinic;

```

### 3.2 min cost flow

```

// Long Long version
typedef pair<long long, long long> pll;
struct CostFlow {
    static const int MAXN = 350;
    static const long long INF = 1LL<<60;
    struct Edge {
        int to, r;
        long long rest, c;
    };

    int n, pre[MAXN], preL[MAXN]; bool inq[MAXN];

```

```

long long dis[MAXN], fl, cost;
vector<Edge> G[MAXN];
void init() {
    for (int i = 0; i < MAXN; i++) G[i].clear();
}
void add_edge(int u, int v, long long rest, long long c) {
    G[u].push_back({v, (int)G[v].size(), rest, c});
    G[v].push_back({u, (int)G[u].size()-1, 0, -c});
}
pll flow(int s, int t) {
    fl = cost = 0;
    while (true) {
        fill(dis, dis+MAXN, INF);
        fill(inq, inq+MAXN, 0);
        dis[s] = 0;
        queue<int> que;
        que.push(s);
        while (!que.empty()) {
            int u = que.front(); que.pop();
            inq[u] = 0;
            for (int i = 0; i < (int)G[u].size(); i++) {
                int v = G[u][i].to;
                long long w = G[u][i].c;
                if (G[u][i].rest > 0 && dis[v] > dis[u] + w) {
                    pre[v] = u; preL[v] = i;
                    dis[v] = dis[u] + w;
                    if (!inq[v]) {
                        inq[v] = 1;
                        que.push(v);
                    }
                }
            }
        }
        if (dis[t] == INF) break;
        long long tf = INF;
        for (int v = t, u, l; v != s; v = u) {
            u = pre[v]; l = preL[v];
            tf = min(tf, G[u][l].rest);
        }
        for (int v = t, u, l; v != s; v = u) {
            u = pre[v]; l = preL[v];
            G[u][l].rest -= tf;
            G[v][G[u][l].r].rest += tf;
        }
        cost += tf * dis[t];
        fl += tf;
    }
    return {fl, cost};
}
} flow;

```

## 4 Mathematics

### 4.1 $ax+by=\gcd(a,b)$

```

typedef pair<int, int> pii;
pii extgcd(int a, int b) {
    if (b == 0) return make_pair(1, 0);
    else {
        int p = a / b;
        pii q = extgcd(b, a % b);
        return make_pair(q.second, q.first - q.second * p);
    }
}

```

### 4.2 BigInt

```

struct BigInt {
    static const int LEN = 60;
    static const int BIGMOD = 10000;
    int s;
    int vl, v[LEN];
    // vector<int> v;
    BigInt() : s(1) { vl = 0; }
    BigInt(long long a) {
        s = 1; vl = 0;
        if (a < 0) { s = -1; a = -a; }
        while (a) {
            push_back(a % BIGMOD);
            a /= BIGMOD;
        }
    }
    BigInt(string str) {
        s = 1; vl = 0;
        int stPos = 0, num = 0;
        if (!str.empty() && str[0] == '-') {
            stPos = 1;
            s = -1;
        }
        for (int i = SZ(str)-1, q=1; i >= stPos; i--) {
            num += (str[i] - '0') * q;
            if ((q *= 10) >= BIGMOD) {
                push_back(num);
                num = 0; q = 1;
            }
        }
        if (num) push_back(num);
    }
    int len() const { return vl; /* return SZ(v); */ }
    bool empty() const { return len() == 0; }
    void push_back(int x) { v[vl++] = x; /* v.PB(x); */ }
    void pop_back() { vl--; /* v.pop_back(); */ }
    int back() const { return v[vl-1]; /* return v.back(); */ }
    void n() { while (!empty() && !back()) pop_back(); }
    void resize(int nl) {
        vl = nl; fill(v, v+vl, 0);
        // v.resize(nl); // fill(ALL(v), 0);
    }
    void print() const {
        if (empty()) { putchar('0'); return; }
        if (s == -1) putchar('-');
        printf("%d", back());
        for (int i = len()-2; i >= 0; i--) printf("%.4d", v[i]);
    }
    friend ostream& operator << (ostream& out, const BigInt &a) {
        if (a.empty()) { out << "0"; return out; }
        if (a.s == -1) out << "-";
        out << a.back();
        for (int i = a.len()-2; i >= 0; i--) {
            char str[10];
            snprintf(str, 5, "%.4d", a.v[i]);
            out << str;
        }
        return out;
    }
    int cp3(const BigInt &b) const {
        if (s != b.s) return s > b.s ? 1 : -1;
        if (s == -1) return -(*this).cp3(-b);
        if (len() != b.len()) return len() > b.len() ? 1 : -1;
        for (int i = len()-1; i >= 0; i--)
            if (v[i] != b.v[i]) return v[i] > b.v[i] ? 1 : -1;
        return 0;
    }
    bool operator < (const BigInt &b) const { return cp3(b) == -1; }
    bool operator <= (const BigInt &b) const { return cp3(b) <= 0; }
    bool operator >= (const BigInt &b) const { return cp3(b) >= 0; }
    bool operator == (const BigInt &b) const { return cp3(b) == 0; }
}

```

```

bool operator != (const Bigint &b) const { return cp3(b)
    !=0; }
bool operator > (const Bigint &b) const { return cp3(b)
    ==1; }
Bigint operator - () const {
    Bigint r = (*this);
    r.s = -r.s;
    return r;
}
Bigint operator + (const Bigint &b) const {
    if (s == -1) return -(*this)+(-b);
    if (b.s == -1) return (*this)-(-b);
    Bigint r;
    int nl = max(len(), b.len());
    r.resize(nl + 1);
    for (int i=0; i<nl; i++) {
        if (i < len()) r.v[i] += v[i];
        if (i < b.len()) r.v[i] += b.v[i];
        if (r.v[i] >= BIGMOD) {
            r.v[i+1] += r.v[i] / BIGMOD;
            r.v[i] %= BIGMOD;
        }
    }
    r.n();
    return r;
}
Bigint operator - (const Bigint &b) const {
    if (s == -1) return -(*this)-(-b);
    if (b.s == -1) return (*this)+(-b);
    if ((*this) < b) return -(b-(*this));
    Bigint r;
    r.resize(len());
    for (int i=0; i<len(); i++) {
        r.v[i] += v[i];
        if (i < b.len()) r.v[i] -= b.v[i];
        if (r.v[i] < 0) {
            r.v[i] += BIGMOD;
            r.v[i+1]--;
        }
    }
    r.n();
    return r;
}
Bigint operator * (const Bigint &b) {
    Bigint r;
    r.resize(len() + b.len() + 1);
    r.s = s * b.s;
    for (int i=0; i<len(); i++) {
        for (int j=0; j<b.len(); j++) {
            r.v[i+j] += v[i] * b.v[j];
            if (r.v[i+j] >= BIGMOD) {
                r.v[i+j+1] += r.v[i+j] / BIGMOD;
                r.v[i+j] %= BIGMOD;
            }
        }
    }
    r.n();
    return r;
}
Bigint operator / (const Bigint &b) {
    Bigint r;
    r.resize(max(1, len()-b.len()+1));
    int oriS = s;
    Bigint b2 = b; // b2 = abs(b)
    s = b2.s = r.s = 1;
    for (int i=r.len()-1; i>=0; i--) {
        int d=0, u=BIGMOD-1;
        while(d<u) {
            int m = (d+u+1)>>1;
            r.v[i] = m;
            if((r*b2) > (*this)) u = m-1;
            else d = m;
        }
        r.v[i] = d;
    }
    s = oriS;
    r.s = s * b.s;

```

```

    r.n();
    return r;
}
Bigint operator % (const Bigint &b) {
    return (*this)-(*this)/b*b;
}
};

```

### 4.3 FFT

```

const double pi = atan(1.0)*4;
struct Complex {
    double x,y;
    Complex(double _x=0,double _y=0)
        :x(_x),y(_y) {}
    Complex operator + (Complex &tt) { return Complex(x
        +tt.x,y+tt.y); }
    Complex operator - (Complex &tt) { return Complex(x
        -tt.x,y-tt.y); }
    Complex operator * (Complex &tt) { return Complex(x
        *tt.x-y*tt.y,x*tt.y+y*tt.x); }
};
void fft(Complex *a, int n, int rev) {
    // n是大于等于相乘的两个数组长度的2的幂次
    // 从0开始表示长度，对a进行操作
    // rev==1进行DFT，== -1进行IDFT
    for (int i = 1, j = 0; i < n; ++ i) {
        for (int k = n>>1; k > (j^=k); k >>= 1);
        if (i<j) std::swap(a[i],a[j]);
    }
    for (int m = 2; m <= n; m <= 1) {
        Complex wm(cos(2*pi*rev/m),sin(2*pi*rev/m));
        for (int i = 0; i < n; i += m) {
            Complex w(1.0,0.0);
            for (int j = i; j < i+m/2; ++ j) {
                Complex t = w*a[j+m/2];
                a[j+m/2] = a[j] - t;
                a[j] = a[j] + t;
                w = w * wm;
            }
        }
    }
    if (rev== -1) {
        for (int i = 0; i < n; ++ i) a[i].x /= n,a[i].y
            /= n;
    }
}

```

### 4.4 FWHT

```

// FWHT template
const int MAXN = 1<<20;
void FWHT(int a[], int l=0, int r=MAXN-1){
    if (l==r)return;

    int mid = (l+r)>>1+1, n = r-l+1;
    FWHT(a,l,mid-1);
    FWHT(a,mid,r);

    for (int i=0; i<(n>>1); i++){
        int a1=a[l+i], a2=a[mid+i];
        a[l+i] = a1+a2;
        a[mid+i] = a1-a2;
    }
}

```

### 4.5 GaussElimination

```
// by bcw_codebook

const int MAXN = 300;
const double EPS = 1e-8;

int n;
double A[MAXN][MAXN];

void Gauss() {
    for(int i = 0; i < n; i++) {
        bool ok = 0;
        for(int j = i; j < n; j++) {
            if(fabs(A[j][i]) > EPS) {
                swap(A[j], A[i]);
                ok = 1;
                break;
            }
        }
        if(!ok) continue;

        double fs = A[i][i];
        for(int j = i+1; j < n; j++) {
            double r = A[j][i] / fs;
            for(int k = i; k < n; k++) {
                A[j][k] -= A[i][k] * r;
            }
        }
    }
}
```

## 4.6 Inverse

```
int inverse[100000];
void invTable(int b, int p) {
    inverse[1] = 1;
    for( int i = 2; i <= b; i++ ) {
        inverse[i] = (long long)inverse[p%i] * (p-p/i) % p;
    }
}

int inv(int b, int p) {
    return b == 1 ? 1 : ((long long)inv(p % b, p) * (p-p/
    b) % p);
}
```

## 4.7 LinearPrime

```
const int MAXP = 100; //max prime
vector<int> P; // primes
void build_prime(){
    static bitset<MAXP> ok;
    int np=0;
    for (int i=2; i<MAXP; i++){
        if (ok[i]==0)P.push_back(i), np++;
        for (int j=0; j<np && i*P[j]<MAXP; j++){
            ok[ i*P[j] ] = 1;
            if ( i%P[j]==0 )break;
        }
    }
}
```

## 4.8 Miller Rabin

```
typedef long long LL;

inline LL bin_mul(LL a, LL n,const LL& MOD){
    LL re=0;
    while (n>0){
        if (n&1) re += a;
        a += a; if (a==MOD) a-=MOD;
        n>>=1;
    }
}
```

```

    }
    return re%MOD;
}

inline LL bin_pow(LL a, LL n,const LL& MOD){
    LL re=1;
    while (n>0){
        if (n&1) re = bin_mul(re,a,MOD);
        a = bin_mul(a,a,MOD);
        n>>=1;
    }
    return re;
}

bool is_prime(LL n){
    //static LL sprp[3] = { 2LL, 7LL, 61LL};
    static LL sprp[7] = { 2LL, 325LL, 9375LL,
        28178LL, 450775LL, 9780504LL,
        1795265022LL };
    if (n==1 || (n&1)==0 ) return n==2;
    int u=n-1, t=0;
    while ( (u&1)==0 ) u>>=1, t++;
    for (int i=0; i<3; i++){
        LL x = bin_pow( sprp[i]%n, u, n);
        if (x==0 || x==1 || x==n-1)continue;

        for (int j=1; j<t; j++){
            x=x*x%n;
            if (x==1 || x==n-1)break;
        }
        if (x==n-1)continue;
        return 0;
    }
    return 1;
}
```

## 4.9 Pollard's rho

```
// from PEC
// does not work when n is prime
Int f(Int x, Int mod){
    return add(mul(x, x, mod), 1, mod);
}

Int pollard_rho(Int n) {
    if ( !(n & 1) ) return 2;
    while (true) {
        Int 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;
    }
}
```

## 4.10 數論基本工具

```
Int POW(Int a, Int n, Int mod){
    Int re=1;
    while (n>0){
        if (n&1LL) re = re*a%mod;
        a = a*a%mod;
        n>>=1;
    }
    return re;
}

Int C(Int n, Int m){
    if (m<0 || m>n)return 0;
    return J[n] * inv(J[m]*J[n-m]%MOD) %MOD;
}
```

## 4.11 Mobius

```
void mobius() {
    fill(isPrime, isPrime + MAXN, 1);
    mu[1] = 1, num = 0;
    for (int i = 2; i < MAXN; ++i) {
        if (isPrime[i]) primes[num++] = i, mu[i] = -1;
        static int d;
        for (int j = 0; j < num && (d = i * primes[j])
              < MAXN; ++j) {
            isPrime[d] = false;
            if (i % primes[j] == 0) {
                mu[d] = 0; break;
            } else mu[d] = -mu[i];
        }
    }
}
```

## 4.12 Simplex

```
// Two-phase simplex algorithm for solving linear
// programs of the form
//
//      maximize      c^T x
//      subject to    Ax <= b
//                  x >= 0
//
// INPUT: A -- an m x n matrix
//        b -- an m-dimensional vector
//        c -- an n-dimensional vector
//        x -- a vector where the optimal solution will
//              be stored
//
// OUTPUT: value of the optimal solution (infinity if
//         unbounded
//         above, nan if infeasible)
//
// To use this code, create an LPSolver object with A,
// b, and c as
// arguments. Then, call Solve(x).
```

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <cmath>
#include <limits>
```

```
using namespace std;
```

```
typedef long double DOUBLE;
typedef vector<DOUBLE> VD;
typedef vector<VD> VVD;
typedef vector<int> VI;
```

```
const DOUBLE EPS = 1e-9;
```

```
struct LPSolver {
    int m, n;
    VI B, N;
    VVD D;

    LPSolver(const VVD &A, const VD &b, const VD &c) :
        m(b.size()), n(c.size()), N(n + 1), B(m), D(m + 2,
            VD(n + 2)) {
        for (int i = 0; i < m; i++) for (int j = 0; j < n;
            j++) D[i][j] = A[i][j];
        for (int i = 0; i < m; i++) { B[i] = n + i; D[i][n]
            = -1; D[i][n + 1] = b[i]; }
        for (int j = 0; j < n; j++) { N[j] = j; D[m][j] = -
            c[j]; }
        N[n] = -1; D[m + 1][n] = 1;
    }
}
```

```
void Pivot(int r, int s) {
```

```
double inv = 1.0 / D[r][s];
for (int i = 0; i < m + 2; i++) if (i != r)
    for (int j = 0; j < n + 2; j++) if (j != s)
        D[i][j] -= D[r][j] * D[i][s] * inv;
for (int j = 0; j < n + 2; j++) if (j != s) D[r][j]
    *= inv;
for (int i = 0; i < m + 2; i++) if (i != r) D[i][s]
    *= -inv;
D[r][s] = inv;
swap(B[r], N[s]);
}
```

```
bool Simplex(int phase) {
    int x = phase == 1 ? m + 1 : m;
    while (true) {
        int s = -1;
        for (int j = 0; j <= n; j++) {
            if (phase == 2 && N[j] == -1) continue;
            if (s == -1 || D[x][j] < D[x][s] || D[x][j] ==
                D[x][s] && N[j] < N[s]) s = j;
        }
        if (D[x][s] > -EPS) return true;
        int r = -1;
        for (int i = 0; i < m; i++) {
            if (D[i][s] < EPS) continue;
            if (r == -1 || D[i][n + 1] / D[i][s] < D[r][n +
                1] / D[r][s] ||
                (D[i][n + 1] / D[i][s]) == (D[r][n + 1] / D[r]
                    [s]) && B[i] < B[r]) r = i;
        }
        if (r == -1) return false;
        Pivot(r, s);
    }
}
```

```
DOUBLE Solve(VD &x) {
    int r = 0;
    for (int i = 1; i < m; i++) if (D[i][n + 1] < D[r][
        n + 1]) r = i;
    if (D[r][n + 1] < -EPS) {
        Pivot(r, n);
        if (!Simplex(1) || D[m + 1][n + 1] < -EPS) return
            numeric_limits<DOUBLE>::infinity();
        for (int i = 0; i < m; i++) if (B[i] == -1) {
            int s = -1;
            for (int j = 0; j <= n; j++)
                if (s == -1 || D[i][j] < D[i][s] || D[i][j]
                    == D[i][s] && N[j] < N[s]) s = j;
            Pivot(i, s);
        }
    }
    if (!Simplex(2)) return numeric_limits<DOUBLE>::
        infinity();
    x = VD(n);
    for (int i = 0; i < m; i++) if (B[i] < n) x[B[i]] =
        D[i][n + 1];
    return D[m][n + 1];
}
```

```
int main() {
```

```
    const int m = 4;
    const int n = 3;
    DOUBLE _A[m][n] = {
        { 6, -1, 0 },
        { -1, -5, 0 },
        { 1, 5, 1 },
        { -1, -5, -1 }
    };
    DOUBLE _b[m] = { 10, -4, 5, -5 };
    DOUBLE _c[n] = { 1, -1, 0 };
```

```
VVD A(m);
VD b(_b, _b + m);
VD c(_c, _c + n);
```

```

for (int i = 0; i < m; i++) A[i] = VD(_A[i], _A[i] +
    n);

LPSolver solver(A, b, c);
VD x;
DOUBLE value = solver.Solve(x);

cerr << "VALUE: " << value << endl; // VALUE: 1.29032
cerr << "SOLUTION:"; // SOLUTION: 1.74194 0.451613 1
for (size_t i = 0; i < x.size(); i++) cerr << " " <<
    x[i];
cerr << endl;
return 0;
}

```

## 4.13 SG

Anti Nim (取走最後一個石子者敗)

先手必勝 **if and only if**

1. 「所有」堆的石子數都為 1 且遊戲的 SG 值為 0。
2. 「有些」堆的石子數大於 1 且遊戲的 SG 值不為 0。

Anti-SG (決策集合為空的遊戲者贏)

定義 SG 值為 0 時，遊戲結束，

則先手必勝 **if and only if**

1. 遊戲中沒有單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數為 0。
2. 遊戲中某個單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數不為 0。

Sprague-Grundy

1. 雙人、回合制
2. 資訊完全公開
3. 無隨機因素
4. 可在有限步內結束
5. 沒有和局
6. 雙方可採取的行動相同

SG(S) 的值為 0：後手(P)必勝

不為 0：先手(N)必勝

```

int mex(set S) {
    // find the min number >= 0 that not in the S
    // e.g. S = {0, 1, 3, 4} mex(S) = 2
}

```

```

state = []
int SG(A) {
    if (A not in state) {
        S = sub_states(A)
        if( len(S) > 1 ) state[A] = reduce(operator.xor, [
            SG(B) for B in S])
        else state[A] = mex(set(SG(B) for B in next_states(
            A)))
    }
    return state[A]
}

```

## 4.14 Theorem

```

/*
Lucas's Theorem
For non-negative integer n,m and prime P,
C(m,n) mod P = C(m/M,n/M) * C(m%M,n%M) mod P
= mult_i ( C(m_i,n_i) )
where m_i is the i-th digit of m in base P.

```

Pick's Theorem

$$A = i + b/2 - 1$$

Kirchhoff's theorem

$$A_{\{ii\}} = \deg(i), A_{\{ij\}} = (i,j) \setminus \text{in } E ? -1 : 0$$

Deleting any one row, one column, and cal the det(A)

Nth Catalan recursive function:

$$C_0 = 1, C_{n+1} = C_n * 2(2n + 1)/(n+2)$$

Mobius Formula

$$u(n) = \begin{cases} 1, & \text{if } n = 1 \\ (-1)^m, & \text{若 } n \text{ 無平方數因數, 且 } n = p_1 * p_2 * p_3 * \dots * p_k \\ 0, & \text{若 } n \text{ 有大於 } 1 \text{ 的平方數因數} \end{cases}$$

- Property

1. (積性函數)  $u(a)u(b) = u(ab)$
2.  $\sum_{d|n} u(d) = [n == 1]$

Mobius Inversion Formula

$$\begin{aligned} \text{if } f(n) &= \sum_{d|n} g(d) \\ \text{then } g(n) &= \sum_{d|n} u(n/d)f(d) \\ &= \sum_{d|n} u(d)f(n/d) \end{aligned}$$

- Application

the number/power of gcd(i, j) = k

- Trick

分塊,  $O(\sqrt{n})$

Chinese Remainder Theorem ( $m_i$  兩兩互質)

$$\begin{aligned} x &= a_1 \pmod{m_1} \\ x &= a_2 \pmod{m_2} \\ &\dots \\ x &= a_i \pmod{m_i} \end{aligned}$$

construct a solution:

$$\begin{aligned} \text{Let } M &= m_1 * m_2 * m_3 * \dots * m_n \\ \text{Let } M_i &= M / m_i \end{aligned}$$

$$\begin{aligned} t_i &= 1 / M_i \\ t_i * M_i &= 1 \pmod{m_i} \end{aligned}$$

$$\begin{aligned} \text{solution } x &= a_1 * t_1 * M_1 + a_2 * t_2 * M_2 + \dots \\ &\quad + a_n * t_n * M_n + k * M \\ &= k * M + \sum a_i * t_i * M_i, k \text{ is positive integer.} \end{aligned}$$

$$\text{under mod } M, \text{ there is one solution } x = \sum a_i * t_i * M_i$$

Burnside's Lemma

$$|G| * |X/G| = \sum (|X^g|) \text{ where } g \text{ in } G$$

總方法數：每一種旋轉下不動點的個數總和 除以 旋轉的方法數

\*/

## 5 Graph

### 5.1 BCC

邊雙連通

任意兩點間至少有兩條不重疊的路徑連接，找法：

1. 標記出所有的橋
2. 對全圖進行 DFS，不走橋，每一次 DFS 就是一個新的邊雙連通

// from BCW

```

struct BccEdge {
    static const int MXN = 100005;
    struct Edge { int v,eid; };

```



```

int n,m,step,par[MXN],dfn[MXN],low[MXN];
vector<Edge> E[MXN];
DisjointSet djs;
void init(int _n) {
    n = _n; m = 0;
    for (int i=0; i<n; i++) E[i].clear();
    djs.init(n);
}
void add_edge(int u, int v) {
    E[u].PB({v, m});
    E[v].PB({u, m});
    m++;
}
void DFS(int u, int f, int f_eid) {
    par[u] = f;
    dfn[u] = low[u] = step++;
    for (auto it:E[u]) {
        if (it.eid == f_eid) continue;
        int v = it.v;
        if (dfn[v] == -1) {
            DFS(v, u, it.eid);
            low[u] = min(low[u], low[v]);
        } else {
            low[u] = min(low[u], dfn[v]);
        }
    }
}
void solve() {
    step = 0;
    memset(dfn, -1, sizeof(int)*n);
    for (int i=0; i<n; i++) {
        if (dfn[i] == -1) DFS(i, i, -1);
    }
    djs.init(n);
    for (int i=0; i<n; i++) {
        if (low[i] < dfn[i]) djs.uni(i, par[i]);
    }
}
}graph;

```

## 5.2 Prim

```

// edge strucute
struct edge{
    int a, b;
    double data;
    bool operator <(const edge b)const{
        return data > b.data;
    }
};

// main prim algorithm
int n, m, root, aa, bb, cc;
while (cin >> n >> m){
    priority_queue<edge>yee;
    int visit[500] = {}, p[500] = {};
    double a[500][500] = {};
    //undirectional edge aa to bb is weighted cc
    for (int i = 0; i < m; i++){
        cin >> aa >> bb >> cc;
        a[aa][bb] = a[bb][aa] = cc;
    }
    cin >> root;
    yee.push({ 0, root, 0 });
    edge tmp;
    double total = 0;
    while (!yee.empty()){
        tmp = yee.top(); yee.pop();
        if (visit[tmp.b])continue;
        total += tmp.data; p[tmp.b] = tmp.a; visit[tmp.b] = 1;
        for (int i = 1; i <= n; i++){
            if (a[tmp.b][i] != 0 && (!visit[i])){
                yee.push({tmp.b, i, a[tmp.b][i]});
            }
        }
    }
}

```

```

}
cout << total << endl;
}

```

## 5.3 Kruskal

```

struct v {
    int a, b, c;
};

int p[200001];v a[200001];

bool sor(v a, v b) {
    return a.c < b.c;
}

int find(int x) {
    return(x != p[x] ? (p[x] = find(p[x])) : x);
}

int main() {
    int n, m, i, j, sum;
    while (cin >> n >> m) {
        sum = 0;
        for (i = 0; i < 200001; i++)p[i] = i;
        for (i = 0; i < m; i++)cin >> a[i].a >> a[i].b >> a[i].c;
        sort(a, a + m, sor);
        for (i = 0, j = 0; j < m; j++) {
            if (find(a[j].a) != find(a[j].b)){
                i++;
                p[find(a[j].a)] = find(a[j].b);
                sum += a[j].c;
            }
        }
        cout << ((i==n-1)?sum:-1) << endl;
    }
}

```

## 5.4 Dijkstra

```

typedef struct Edge{
    int v; long long len;
    bool operator > (const Edge &b)const { return len>b.len; }
} State;

const long long INF = 1LL<<60;

void Dijkstra(int n, vector<Edge> G[], long long d[],
    int s, int t=-1){
    static priority_queue<State, vector<State>, greater<State> > pq;
    while ( pq.size() )pq.pop();
    for (int i=1; i<=n; i++)d[i]=INF;
    d[s]=0; pq.push( (State){s,d[s]} );
    while ( pq.size() ){
        auto x = pq.top(); pq.pop();
        int u = x.v;
        if (d[u]<x.len)continue;
        if (u==t)return;
        for (auto &e:G[u]){
            if (d[e.v] > d[u]+e.len){
                d[e.v] = d[u]+e.len;
                pq.push( (State) {e.v,d[e.v]} );
            }
        }
    }
}

```

## 5.5 Strongly Connected Component(SCC)



```

#define MXN 100005
#define PB push_back
#define FZ(s) memset(s,0,sizeof(s))

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 add_edge(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++;
        }
    }
}
};

```

## 5.6 Hungarian

```

// Maximum Cardinality Bipartite Matching

struct Graph {
    static const int MAXN = 5005;
    vector<int> G[MAXN];
    int n;
    int match[MAXN]; // Matching Result
    int vis[MAXN];

    void init(int _n) {
        n = _n;
        for (int i = 0 ; i < n ; i++ ) G[i].clear();
    }

    bool dfs(int u) {
        for ( auto v:G[u] ) {
            if (!vis[v]) {
                vis[v] = true;
                if (match[v] == -1 || dfs(match[v])) {
                    match[v] = u;
                    match[u] = v;
                    return true;
                }
            }
        }
        return false;
    }
};

```

```

int solve() {
    int res = 0;
    memset(match, -1, sizeof(match));
    for (int i = 0; i < n; i++) {
        if (match[i] == -1) {
            memset(vis, 0, sizeof(vis));
            if (dfs(i)) res += 1;
        }
    }
    return res;
}
} graph;

```

## 5.7 KM

Detect non-perfect-matching:

1. set all edge[i][j] as INF
2. if solve() >= INF, it is **not** perfectmatching.

-----  
*// Maximum Weight Perfect Bipartite Matching*  
*// allow negative weight!*

```

typedef long long Int;
struct KM {
    static const int MAXN = 1050;
    static const int INF = 1LL<<60;
    int n, match[MAXN], vx[MAXN], vy[MAXN];
    Int edge[MAXN][MAXN], lx[MAXN], ly[MAXN], slack[
        MAXN];
    void init(int _n){
        n = _n;
        for ( int i = 0 ; i < n ; i++ )
            for ( int j = 0 ; j < n ; j++ )
                edge[i][j] = 0;
    }
    void add_edge(int x, int y, Int w){
        edge[x][y] = w;
    }
    bool DFS(int x){
        vx[x] = 1;
        for ( int y = 0 ; y < n ; y++ ) {
            if ( vy[y] ) continue;
            if ( lx[x] + ly[y] > edge[x][y] ) {
                slack[y] = min(slack[y], lx[x] + ly[y]
                    - edge[x][y]);
            } else {
                vy[y] = 1;
                if ( match[y] == -1 || DFS(match[y]) ){
                    match[y] = x;
                    return true;
                }
            }
        }
        return false;
    }
    Int solve() {
        fill(match, match + n, -1);
        fill(lx, lx + n, -INF);
        fill(ly, ly + n, 0);
        for ( int i = 0; i < n; i++ )
            for ( int j = 0; j < n; j++ )
                lx[i] = max(lx[i], edge[i][j]);
        for ( int i = 0 ; i < n; i++ ) {
            fill(slack, slack + n, INF);
            while (true){
                fill(vx, vx + n, 0);
                fill(vy, vy + n, 0);
                if ( DFS(i) ) break;
                Int d = INF;
                for ( int j = 0 ; j < n ; j++ )
                    if ( !vy[j] ) d = min(d, slack[j]);
                for ( int j = 0 ; j < n ; j++ ) {
                    if (vx[j]) lx[j] -= d;
                    if (vy[j]) ly[j] += d;
                    else slack[j] -= d;
                }
            }
        }
    }
};

```

```

    }
    }
    Int res = 0;
    for ( int i = 0 ; i < n ; i++ ) {
        res += edge[ match[i] ][i];
    }
    return res;
}
} graph;

```

## 5.8 最小平均環

```

// from BCW
/* minimum mean cycle */
const int MAXE = 1805;
const int MAXN = 35;
const double inf = 1029384756;
const double eps = 1e-6;
struct Edge {
    int v,u;
    double c;
};
int n,m,prv[MAXN][MAXN], prve[MAXN][MAXN], vst[MAXN];
Edge e[MAXE];
vector<int> edgeID, cycle, rho;
double d[MAXN][MAXN];
inline void bellman_ford() {
    for(int i=0; i<n; i++) d[0][i]=0;
    for(int i=0; i<n; i++) {
        fill(d[i+1], d[i+1]+n, inf);
        for(int j=0; j<m; j++) {
            int v = e[j].v, u = e[j].u;
            if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j].c) {
                d[i+1][u] = d[i][v]+e[j].c;
                prv[i+1][u] = v;
                prve[i+1][u] = j;
            }
        }
    }
}
double karp_mmc() {
    // returns inf if no cycle, mmc otherwise
    double mmc=inf;
    int st = -1;
    bellman_ford();
    for(int i=0; i<n; i++) {
        double avg=-inf;
        for(int k=0; k<n; k++) {
            if(d[n][i]<inf-eps) avg=max(avg,(d[n][i]-d[k][i])/(n-k));
            else avg=max(avg,inf);
        }
        if (avg < mmc) tie(mmc, st) = tie(avg, i);
    }
    for(int i=0; i<n; i++) vst[i] = 0;
    edgeID.clear(); cycle.clear(); rho.clear();
    for (int i=n; !vst[st]; st=prv[i-1][st]) {
        vst[st]++;
        edgeID.PB(prve[i][st]);
        rho.PB(st);
    }
    while (vst[st] != 2) {
        int v = rho.back(); rho.pop_back();
        cycle.PB(v);
        vst[v]++;
    }
    reverse(ALL(edgeID));
    edgeID.resize(SZ(cycle));
    return mmc;
}

```

## 5.9 偵測負環

```

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

const int INF = 1000000;
const int MAXN = 200;
int n, m, q;
int d[MAXN][MAXN];

int main () {
    while ( cin >> n >> m >> q && n ) {

        for ( int i = 0 ; i <= n ; i++ ) {
            for ( int j = 0 ; j <= n ; j++ ) d[i][j] =
                (i==j ? 0 : INF);
        }

        for ( int i = 0 ; i < m ; i++ ) {
            int a, b, c;
            cin >> a >> b >> c;
            d[a][b] = min(d[a][b], c);
        }

        for ( int k = 0 ; k < n ; k++ ) {
            for ( int i = 0 ; i < n ; i++ ) {
                for ( int j = 0 ; j < n ; j++ ) {
                    if ( d[i][j] > d[i][k] + d[k][j] &&
                        d[i][k] < INF && d[k][j] < INF ) {
                        //printf("%d > %d + %d\n", d[i][j], d[i][k], d[k][j]);
                        //if ( d[i][k] >= INF || d[k][j] >= INF ) cout << "NO : "
                        << i << " " << j << " " << k << "--";
                        d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
                    }
                }
            }
        }

        for ( int i = 0 ; i < n ; i++ ) {
            for ( int j = 0 ; j < n ; j++ ) {
                for ( int k = 0 ; k < n && d[i][j] != -
                    INF ; k++ ) {
                    if ( d[k][k] < 0 && d[i][k] != INF
                        && d[k][j] != INF )
                        d[i][j] = -INF;
                }
            }
        }

        int u, v;
        for (int i=0; i<q; i++){
            scanf("%d%d",&u,&v);

            if (d[u][v] == INF) printf("Impossible\n");
            else if (d[u][v] == -INF) printf("-Infinity\n");
            else printf("%d\n",d[u][v]);
        }
        puts("");
    }
    return 0;
}

```

## 5.10 Tarjan

割點

點  $u$  為割點 **if and only if** 滿足 1. **or** 2.

1.  $u$  為樹根，且  $u$  有多於一個子樹。
2.  $u$  不為樹根，且滿足存在  $(u,v)$  為樹枝邊（或稱父子邊，即  $u$  為  $v$  在搜索樹中的父親），使得  $DFN(u) \leq Low(v)$

```

    °
-----
橋
一條無向邊 (u,v) 是橋 if and only if (u,v) 為樹枝邊，且
滿足 DFN(u) < Low(v) °

// 0 base
struct TarjanSCC{
    static const int MAXN = 1000006;
    int n, dfn[MAXN], low[MAXN], scc[MAXN], scn, count;
    vector<int> G[MAXN];
    stack<int> stk;
    bool ins[MAXN];

    void tarjan(int u){
        dfn[u] = low[u] = ++count;
        stk.push(u);
        ins[u] = true;

        for(auto v:G[u]){
            if(!dfn[v]){
                tarjan(v);
                low[u] = min(low[u], low[v]);
            }else if(ins[v]){
                low[u] = min(low[u], dfn[v]);
            }
        }

        if(dfn[u] == low[u]){
            int v;
            do {
                v = stk.top();
                stk.pop();
                scc[v] = scn;
                ins[v] = false;
            } while(v != u);
            scn++;
        }
    }

    void getSCC(){
        memset(dfn,0,sizeof(dfn));
        memset(low,0,sizeof(low));
        memset(ins,0,sizeof(ins));
        memset(scc,0,sizeof(scc));
        count = scn = 0;
        for(int i = 0 ; i < n ; i++ ){
            if(!dfn[i]) tarjan(i);
        }
    }
}SCC;

```

## 5.11 Topological Sort

```

bool adj[9][9];        // adjacency matrix
int visit[9];          // 0:DFS^1M%L^2^3I
int order[9], n;       // x's@OX^2z^2±RC^9^5

bool cycle;            // 0:DFS^2Lμ{H-H-O$-°»'°""

void DFS(int s)
{
    // back edge;A^1^3^2C
    if (visit[s] == 1) {cycle = true; return;}
    // forward edge;Bcross edge;C
    if (visit[s] == 2) return;

    visit[s] = 1;
    for (int t=0; t<9; ++t)
        if (adj[s][t])
            DFS(t);
    visit[s] = 2;
}

```

```

    order[n--] = s;      // 0:DFS^1M%L^2^3I
}

void topological_ordering()
{
    // 0:DFS^1M%L^2^3I
    for (int i=0; i<9; i++) visit[i] = 0;
    cycle = false;
    n = 9-1;

    // 0:DFS^1M%L^2^3I
    for (int s=0; s<9; ++s)
        if (!v[s])
            DFS(s);

    // 0:DFS^1M%L^2^3I
    if (cycle)
        cout << "1qW^1^3^2";
    else
        // 0:DFS^1M%L^2^3I
        for (int i=0; i<9; ++i)
            cout << order[i];
}

```

## 6 Data Structure

### 6.1 2D Range Tree

```

// remember sort x !!!!!
typedef int T;
const int LGN = 20;
const int MAXN = 100005;

struct Point{
    T x, y;
    friend bool operator < (Point a, Point b){
        return tie(a.x,a.y) < tie(b.x,b.y);
    }
};

struct TREE{
    Point pt;
    int toleft;
}tree[LGN][MAXN];

struct SEG{
    T mx, Mx;
    int sz;
    TREE *st;
}seg[MAXN*4];

vector<Point> P;

void build(int l, int r, int o, int deep){
    seg[o].mx = P[l].x;
    seg[o].Mx = P[r].x;
    seg[o].sz = r-l+1;

    if(l == r){
        tree[deep][r].pt = P[r];
        tree[deep][r].toleft = 0;
        seg[o].st = &tree[deep][r];
        return;
    }

    int mid = (l+r)>>1;
    build(l,mid,o+o,deep+1);
    build(mid+1,r,o+o+1,deep+1);

    TREE *ptr = &tree[deep][l];
    TREE *pl = &tree[deep+1][l], *nl = &tree[deep+1][
        mid+1];
    TREE *pr = &tree[deep+1][mid+1], *nr = &tree[deep
        +1][r+1];

    int cnt = 0;
    while(pl != nl && pr != nr) {

```



```

    while (o->next[c]==NULL) o = o->fail;
    return o->next[c];
}

void make_fail(){
    static queue<Node*> que;

    for (int i=0; i<sigma; i++){
        ori->next[i] = root;
        root->fail = ori;

        que.push(root);
        while ( que.size() ){
            Node *u = que.front(); que.pop();
            for (int i=0; i<sigma; i++){
                if (u->next[i]==NULL) continue;
                u->next[i]->fail = trans(u->fail,i);
                que.push(u->next[i]);
            }
            u->cnt += u->fail->cnt;
        }
    }
} ac;

```

## 7.2 KMP

```

template<typename T>
void build_KMP(int n, T *s, int *f){ // 1 base
    f[0]=-1, f[1]=0;
    for (int i=2; i<=n; i++){
        int w = f[i-1];
        while (w>0 && s[w+1]!=s[i]) w = f[w];
        f[i]=w+1;
    }
}

template<typename T>
int KMP(int n, T *a, int m, T *b){
    build_KMP(m,b,f);
    int ans=0;

    for (int i=1, w=0; i<=n; i++){
        while ( w>0 && b[w+1]!=a[i] ) w = f[w];
        w++;
        if (w==m){
            ans++;
            w=f[w];
        }
    }
    return ans;
}

```

## 7.3 迴文字動機

```

// remember init()      !!!
// remember make_fail() !!!
// insert s need 1 base !!!
// notice MLE
const int sigma = 62;
const int MAXC = 1000006;
inline int idx(char c){
    if ('a'<= c && c <= 'z') return c-'a';
    if ('A'<= c && c <= 'Z') return c-'A'+26;
    if ('0'<= c && c <= '9') return c-'0'+52;
}
struct PalindromicTree{
    struct Node{
        Node *next[sigma], *fail;
        int len, cnt; // for dp
        Node(){
            memset(next,0,sizeof(next));
            fail=0;
            len = cnt = 0;
        }
    };
};

```

```

    }
    buf[MAXC], *bufp, *even, *odd;

    void init(){
        bufp = buf;
        even = new (bufp++) Node();
        odd = new (bufp++) Node();
        even->fail = odd;
        odd->len = -1;
    }

    void insert(char *s){
        Node* ptr = even;
        for (int i=1; s[i]; i++){
            ptr = extend(ptr,s[i]);
        }
    }

    Node* extend(Node *o, char *ptr){
        int c = idx(*ptr);
        while ( *ptr != *(ptr-1-o->len) ) o=o->fail;
        Node *&np = o->next[c];
        if (!np){
            np = new (bufp++) Node();
            np->len = o->len+2;
            Node *f = o->fail;
            if (f){
                while ( *ptr != *(ptr-1-f->len) ) f=f->fail;
                np->fail = f->next[c];
            }
            else {
                np->fail = even;
            }
            np->cnt = np->fail->cnt;
        }
        np->cnt++;
        return np;
    }
} PAM;

```

## 7.4 Suffix Automaton

```

// par : fail link
// val : a topological order ( useful for DP )
// go[x] : automata edge ( x is integer in [0,26) )

struct SAM{
    struct State{
        int par, go[26], val;
        State () : par(0), val(0){ FZ(go); }
        State (int _val) : par(0), val(_val){ FZ(go); }
    };
    vector<State> vec;
    int root, tail;

    void init(int arr[], int len){
        vec.resize(2);
        vec[0] = vec[1] = State(0);
        root = tail = 1;
        for (int i=0; i<len; i++){
            extend(arr[i]);
        }
    }

    void extend(int w){
        int p = tail, np = vec.size();
        vec.PB(State(vec[p].val+1));
        for ( ; p && vec[p].go[w]==0; p=vec[p].par)
            vec[p].go[w] = np;
        if (p == 0){
            vec[np].par = root;
        }
        else {
            if (vec[vec[p].go[w]].val == vec[p].val+1){
                vec[np].par = vec[p].go[w];
            }
            else {
                int q = vec[p].go[w], r = vec.size();
                vec.PB(vec[q]);
            }
        }
    }
};

```

```

        vec[r].val = vec[p].val+1;
        vec[q].par = vec[np].par = r;
        for ( ; p && vec[p].go[w] == q; p=vec[p].par)
            vec[p].go[w] = r;
    }
}
tail = np;
}
};

```

## 7.5 smallest rotation

```

string mcp(string s){
    int n = s.length();
    s += s;
    int i=0, j=1;
    while (i<n && j<n){
        int k = 0;
        while (k < n && s[i+k] == s[j+k]) k++;
        if (s[i+k] <= s[j+k]) j += k+1;
        else i += k+1;
        if (i == j) j++;
    }
    int ans = i < n ? i : j;
    return s.substr(ans, n);
}

```

Contact GitHub API Training Shop Blog About

## 7.6 Suffix Array

*/\*he[i]保存了在後綴數組中相鄰兩個後綴的最長公共前綴長度  
 \*sa[i]表示的是字典序排名為i的後綴是誰（字典序越小的排名越靠前）  
 \*rk[i]表示的是後綴我所對應的排名是多少 \*/*

```

const int MAX = 1020304;
int ct[MAX], he[MAX], rk[MAX];
int sa[MAX], tsa[MAX], tp[MAX][2];
void suffix_array(char *ip){
    int len = strlen(ip);
    int alp = 256;
    memset(ct, 0, sizeof(ct));
    for(int i=0;i<len;i++) ct[ip[i]+1]++;
    for(int i=1;i<alp;i++) ct[i]+=ct[i-1];
    for(int i=0;i<len;i++) rk[i]=ct[ip[i]];
    for(int i=1;i<len;i*=2){
        for(int j=0;j<len;j++){
            if(j+i>len) tp[j][1]=0;
            else tp[j][1]=rk[j+i]+1;
            tp[j][0]=rk[j];
        }
        memset(ct, 0, sizeof(ct));
        for(int j=0;j<len;j++) ct[tp[j][1]+1]++;
        for(int j=1;j<len+2;j++) ct[j]+=ct[j-1];
        for(int j=0;j<len;j++) tsa[ct[tp[j][1]]+j]=j;
        memset(ct, 0, sizeof(ct));
        for(int j=0;j<len;j++) ct[tp[j][0]+1]++;
        for(int j=1;j<len+1;j++) ct[j]+=ct[j-1];
        for(int j=0;j<len;j++)
            sa[ct[tp[j][0]]+j]=j;
        rk[sa[0]]=0;
        for(int j=1;j<len;j++){
            if( tp[sa[j]][0] == tp[sa[j-1]][0] &&
               tp[sa[j]][1] == tp[sa[j-1]][1] )
                rk[sa[j]] = rk[sa[j-1]];
            else
                rk[sa[j]] = j;
        }
    }
    for(int i=0,h=0;i<len;i++){
        if(rk[i]==0) h=0;
        else{

```

```

            int j=sa[rk[i]-1];
            h=max(0,h-1);
            for(;ip[i+h]==ip[j+h];h++);
        }
        he[rk[i]]=h;
    }
}

```

## 7.7 Z-value

```

z[0] = 0;
for ( int bst = 0, i = 1; i < len ; i++ ) {
    if ( z[bst] + bst <= i ) z[i] = 0;
    else z[i] = min(z[i - bst], z[bst] + bst - i);
    while ( str[i + z[i]] == str[z[i]] ) z[i]++;
    if ( i + z[i] > bst + z[bst] ) bst = i;
}

// 回文版

void Zpal(const char *s, int len, int *z) {
    // Only odd palindrome len is considered
    // z[i] means that the longest odd palindrom
    // centered at
    // i is [i-z[i] .. i+z[i]]
    z[0] = 0;
    for (int b=0, i=1; i<len; i++) {
        if (z[b] + b >= i) z[i] = min(z[2*b-i], b+z[b]-i);
        else z[i] = 0;
        while (i+z[i]+1 < len and i-z[i]-1 >= 0 and
               s[i+z[i]+1] == s[i-z[i]-1]) z[i] ++;
        if (z[i] + i > z[b] + b) b = i;
    }
}

```

## 8 Others

### 8.1 矩陣數定理

新的方法介绍

下面我们介绍一种新的方法——Matrix-Tree定理(Kirchhoff矩阵-树定理)。

Matrix-Tree定理是解决生成树计数问题最有力的武器之一。它首先于1847年被Kirchhoff证明。在介绍定理之前，我们首先明确几个概念：

1、G的度数矩阵D[G]是一个n\*n的矩阵，并且满足：当i≠j时，dij=0；当i=j时，dij等于vi的度数。

2、G的邻接矩阵A[G]也是一个n\*n的矩阵，并且满足：如果vi、vj之间有边直接相连，则aij=1，否则为0。

我们定义G的Kirchhoff矩阵(也称为拉普拉斯算子)C[G]为C[G]=D[G]-A[G]，

则Matrix-Tree定理可以描述为：G的所有不同的生成树的个数等于其Kirchhoff矩阵C[G]任何一个n-1阶主子式的行列式的绝对值。

所谓n-1阶主子式，就是对于r(1≤r≤n)，将C[G]的第r行、第r列同时去掉后得到的新矩阵，用Cr[G]表示。

生成树计数

算法步骤：

1、构建拉普拉斯矩阵

Matrix[i][j] = degree(i), i==j

-1, i-j有边

0, 其他情况

2、去掉第r行，第r列 (r任意)

3、计算矩阵的行列式

## 8.2 CYK

```

/* *****
MYID   : Chen Fan
LANG   : G++
PROG   : Count_Spaning_Tree_From_Kuangbin
***** */
#include <stdio.h>
#include <string.h>
#include <algorithm>
#include <iostream>
#include <math.h>
using namespace std;
const double eps = 1e-8;
const int MAXN = 110;
int sgn(double x)
{
    if(fabs(x) < eps) return 0;
    if(x < 0) return -1;
    else return 1;
}
double b[MAXN][MAXN];
double det(double a[][MAXN], int n)
{
    int i, j, k, sign = 0;
    double ret = 1;
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++) b[i][j] = a[i][j];
    for(i = 0; i < n; i++)
    {
        if(sgn(b[i][i]) == 0)
        {
            for(j = i + 1; j < n; j++)
                if(sgn(b[j][i]) != 0) break;
            if(j == n) return 0;
            for(k = i; k < n; k++) swap(b[i][k], b[j][k]);
            sign++;
        }
        ret *= b[i][i];
        for(k = i + 1; k < n; k++) b[i][k] /= b[i][i];
        for(j = i + 1; j < n; j++)
            for(k = i + 1; k < n; k++) b[j][k] -= b[j][i] * b[i][k];
    }
    if(sign & 1) ret = -ret;
    return ret;
}
double a[MAXN][MAXN];
int g[MAXN][MAXN];
int main()
{
    int T;
    int n, m;
    int u, v;
    scanf("%d", &T);
    while(T--)
    {
        scanf("%d%d", &n, &m);
        memset(g, 0, sizeof(g));
        while(m--)
        {
            scanf("%d%d", &u, &v);
            u--; v--;
            g[u][v] = g[v][u] = 1;
        }
        memset(a, 0, sizeof(a));
        for(int i = 0; i < n; i++)
            for(int j = 0; j < n; j++)
                if(i != j && g[i][j])
                {
                    a[i][i]++;
                    a[i][j] = -1;
                }
        double ans = det(a, n-1);
        printf("%.0Lf\n", ans);
    }
    return 0;
}

```

```

// 2016 NCPC from sunmoon
// 轉換
#define MAXN 55
struct CNF{
    int s, x, y; // s->xy | s->x, if y==--1
    int cost;
    CNF(){}
    CNF(int s, int x, int y, int c): s(s), x(x), y(y), cost(c){}
};
int state; // 規則數量
map<char, int> rule; // 每個字元對應到的規則，小寫字母為終端字符
vector<CNF> cnf;
inline void init(){
    state = 0;
    rule.clear();
    cnf.clear();
}
inline void add_to_cnf(char s, const string &p, int cost)
{
    if(rule.find(s) == rule.end()) rule[s] = state++;
    for(auto c: p) if(rule.find(c) == rule.end()) rule[c] = state++;
    if(p.size() == 1){
        cnf.push_back(CNF(rule[s], rule[p[0]], -1, cost));
    } else {
        int left = rule[s];
        int sz = p.size();
        for(int i = 0; i < sz - 2; ++i){
            cnf.push_back(CNF(left, rule[p[i]], state, 0));
            left = state++;
        }
        cnf.push_back(CNF(left, rule[p[sz-2]], rule[p[sz-1]], cost));
    }
}
// 計算
vector<long long> dp[MAXN][MAXN];
vector<bool> neg_INF[MAXN][MAXN]; // 如果花費是負的可能會有無限小的情形
inline void relax(int l, int r, const CNF &c, long long cost, bool neg_c = 0){
    if(!neg_INF[l][r][c.s] && (neg_INF[l][r][c.x] || cost < dp[l][r][c.s])){
        if(neg_c || neg_INF[l][r][c.x]){
            dp[l][r][c.s] = 0;
            neg_INF[l][r][c.s] = true;
        } else dp[l][r][c.s] = cost;
    }
}
inline void bellman(int l, int r, int n){
    for(int k = 1; k <= state; ++k)
        for(auto c: cnf)
            if(c.y == -1) relax(l, r, c, dp[l][r][c.x] + c.cost, k == n);
}
inline void cyk(const vector<int> &tok){
    for(int i = 0; i < (int)tok.size(); ++i){
        for(int j = 0; j < (int)tok.size(); ++j){
            dp[i][j] = vector<long long>(state + 1, INT_MAX);
            neg_INF[i][j] = vector<bool>(state + 1, false);
        }
        dp[i][i][tok[i]] = 0;
        bellman(i, i, tok.size());
    }
    for(int r = 1; r < (int)tok.size(); ++r){
        for(int l = r - 1; l >= 0; --l){
            for(int k = 1; k < r; ++k)
                for(auto c: cnf)

```



```

        if (~c.y) relax(1, r, c, dp[1][k][c.x] + dp[k+1][r][
            c.y] + c.cost);
        bellman(1, r, tok.size());
    }
}
}

```

### 8.3 數位統計

```

int dfs(int pos, int state1, int state2 ....., bool
    limit, bool zero) {
    if ( pos == -1 ) return 是否符合條件;
    int &ret = dp[pos][state1][state2][....];
    if ( ret != -1 && !limit ) return ret;
    int ans = 0;
    int upper = limit ? digit[pos] : 9;
    for ( int i = 0 ; i <= upper ; i++ ) {
        ans += dfs(pos - 1, new_state1, new_state2,
            limit & ( i == upper ), ( i == 0 ) && zero);
    }
    if ( !limit ) ret = ans;
    return ans;
}

int solve(int n) {
    int it = 0;
    for ( ; n ; n /= 10 ) digit[it++] = n % 10;
    return dfs(it - 1, 0, 0, 1, 1);
}

```

### 8.4 1D/1D dp 優化

```

#include <bits/stdc++.h>

int t, n, L;
int p;
char s[MAXN][35];
ll sum[MAXN] = {0};
long double dp[MAXN] = {0};
int prevd[MAXN] = {0};

long double pw(long double a, int n) {
    if ( n == 1 ) return a;
    long double b = pw(a, n/2);
    if ( n & 1 ) return b*b*a;
    else return b*b;
}

long double f(int i, int j) {
    // cout << (sum[i] - sum[j] + i - j - 1 - L) << endl;
    return pw(abs(sum[i] - sum[j] + i - j - 1 - L), p) + dp[j];
}

struct INV {
    int L, R, pos;
};
INV stk[MAXN*10];
int top = 1, bot = 1;
void update(int i) {
    while ( top > bot && i < stk[top].L && f(stk[top].L,
        i) < f(stk[top].L, stk[top].pos) ) {
        stk[top - 1].R = stk[top].R;
        top--;
    }
    int lo = stk[top].L, hi = stk[top].R, mid, pos =
        stk[top].pos;
    //if ( i >= lo ) lo = i + 1;
    while ( lo != hi ) {
        mid = lo + (hi - lo) / 2;
        if ( f(mid, i) < f(mid, pos) ) hi = mid;
        else lo = mid + 1;
    }
    if ( hi < stk[top].R ) {
        stk[top + 1] = (INV) { hi, stk[top].R, i };
        stk[top++].R = hi;
    }
}

```

```

}
}

int main() {
    cin >> t;
    while ( t-- ) {
        cin >> n >> L >> p;
        dp[0] = sum[0] = 0;
        for ( int i = 1 ; i <= n ; i++ ) {
            cin >> s[i];
            sum[i] = sum[i-1] + strlen(s[i]);
            dp[i] = numeric_limits<long double>::max();
        }
        stk[top] = (INV) {1, n + 1, 0};
        for ( int i = 1 ; i <= n ; i++ ) {
            if ( i >= stk[bot].R ) bot++;
            dp[i] = f(i, stk[bot].pos);
            update(i);
            // cout << (LL) f(i, stk[bot].pos) << endl;
        }
        if ( dp[n] > 1e18 ) {
            cout << "Too hard to arrange" << endl;
        } else {
            vector<PI> as;
            cout << (ll)dp[n] << endl;
        }
    }
    return 0;
}

```

### 8.5 Theorm - DP optimization

Monotonicity & 1D/1D DP & 2D/1D DP

Definition xD/yD

1D/1D  $DP[j] = \min(0 \leq i < j) \{ DP[i] + w(i, j) \}$ ;  $DP[0] = k$   
 2D/1D  $DP[i][j] = \min(i < k \leq j) \{ DP[i][k - 1] + DP[k][j] \}$   
 $+ w(i, j)$ ;  $DP[i][i] = 0$

Monotonicity

$$\begin{array}{cc} & c & d \\ a & | & w(a, c) & w(a, d) \\ b & | & w(b, c) & w(b, d) \end{array}$$

Monge Condition

Concave (凹四邊形不等式):  $w(a, c) + w(b, d) \geq w(a, d) + w(b, c)$

Convex (凸四邊形不等式):  $w(a, c) + w(b, d) \leq w(a, d) + w(b, c)$

Totally Monotone

Concave (凹單調):  $w(a, c) \leq w(b, d) \rightarrow w(a, d) \leq w(b, c)$

Convex (凸單調):  $w(a, c) \geq w(b, d) \rightarrow w(a, d) \geq w(b, c)$

1D/1D DP  $O(n^2) \rightarrow O(n \lg n)$

\*\*CONSIDER THE TRANSITION POINT\*\*

Solve 1D/1D Concave by Stack

Solve 1D/1D Convex by Deque

2D/1D Convex DP (Totally Monotone)  $O(n^3) \rightarrow O(n^2)$

$h(i, j - 1) \leq h(i, j) \leq h(i + 1, j)$

### 8.6 Stable Marriage

```

// normal stable marriage problem
// input:
//3
//Albert Laura Nancy Marcy
//Brad Marcy Nancy Laura
//Chuck Laura Marcy Nancy

```

```

//Laura Chuck Albert Brad
//Marcy Albert Chuck Brad
//Nancy Brad Albert Chuck

#include <bits/stdc++.h>
using namespace std;
const int MAXN = 505;

int n;
int favor[MAXN][MAXN]; // favor[boy_id][rank] = girl_id
;
int order[MAXN][MAXN]; // order[girl_id][boy_id] = rank
;
int current[MAXN]; // current[boy_id] = rank; boy_id
will pursue current[boy_id] girl.
int girl_current[MAXN]; // girl[girl_id] = boy_id;

void initialize() {
    for ( int i = 0 ; i < n ; i++ ) {
        current[i] = 0;
        girl_current[i] = n;
        order[i][n] = n;
    }
}

map<string, int> male, female;
string bname[MAXN], gname[MAXN];
int fit = 0;

void stable_marriage() {

    queue<int> que;
    for ( int i = 0 ; i < n ; i++ ) que.push(i);
    while ( !que.empty() ) {
        int boy_id = que.front();
        que.pop();

        int girl_id = favor[boy_id][current[boy_id]];
        current[boy_id] ++;

        if ( order[girl_id][boy_id] < order[girl_id][
            girl_current[girl_id]] ) {
            if ( girl_current[girl_id] < n ) que.push(
                girl_current[girl_id]); // if not the first
                time
            girl_current[girl_id] = boy_id;
        } else {
            que.push(boy_id);
        }
    }
}

int main() {
    cin >> n;

    for ( int i = 0 ; i < n ; i++ ) {
        string p, t;
        cin >> p;
        male[p] = i;
        bname[i] = p;
        for ( int j = 0 ; j < n ; j++ ) {
            cin >> t;
            if ( !female.count(t) ) {
                gname[fit] = t;
                female[t] = fit++;
            }
            favor[i][j] = female[t];
        }
    }

    for ( int i = 0 ; i < n ; i++ ) {
        string p, t;
        cin >> p;
        for ( int j = 0 ; j < n ; j++ ) {
            cin >> t;
            order[female[p]][male[t]] = j;
        }
    }
}

```

```

    }
}

initialize();
stable_marriage();

for ( int i = 0 ; i < n ; i++ ) {
    cout << bname[i] << " " << gname[favor[i][current[i]
        ] - 1] << endl;
}
}

```

## 8.7 parser

```

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

typedef long long T;
bool GG;

T Eval2(char *&end) {
    T Eval0(char *&);
    T res=0;
    if ( *end=='(' ){
        res = Eval0(++end);
        if (*(end++)==')') return res;
        else { GG = true; return -1; }
    }
    else if( isdigit(*end) ){
        return strtol(end, &end, 10);
    } // 可改成 {strtol ,strtoll strtod}
    else { GG = true; return -1; }
}

T Evalx(char *&end){
    if(GG) return -1;
    T res = Eval2(end); if(GG) return -1;
    while (*end == '%'){
        end++;
        res = ( res % Eval2(end) );
        if(GG) return -1;
    }
    return res;
}

T Eval1(char *&end) {
    if(GG) return -1;
    T res = Evalx(end); if(GG) return -1;
    while (*end=='*' || *end=='/'){
        end++;
        if(*(end-1) == '*')res = ( res * Evalx(end) );
        else if(*(end-1) == '/')res = ( res / Evalx(end)
            );
        if(GG) return -1;
    }
    return res;
}

T Eval12(char *&end){
    if(GG) return -1;
    T res=1;
    if(*end == '-'){
        end++;
        res = -1;
    }
    res *= Evalx(end);
    while (*end=='*' || *end=='/'){
        end++;
        if(*(end-1) == '*')res = ( res * Evalx(end) );
        else if(*(end-1) == '/')res = ( res / Evalx(end)
            );
        if(GG) return -1;
    }
    return res;
}

```

```

}
T Eval0(char *&end) {
    if(GG) return -1;
    T res;
    res = Eval12(end); if(GG) return -1;
    while (*end=='+' || *end=='-'){
        end++;
        if(*(end-1) == '+')res = ( res + Eval1(end) );
        else res = ( res - Eval1(end) );
        if(GG) return -1;
    }
    return res;
}

T parse(char *s){
    GG = false;
    T res = Eval0(s);
    while(*s != '\0'){
        if(*s != ' ')GG = true;
        s++;
    }
    return res;
}

int main() {
    char expr[3003];
    string str;
    int cnt = 0;
    while (getline (cin,str)){
        printf("case %d:\n",++cnt);
        strcpy(expr,str.c_str());
        T ans = parse(expr);
        if(GG) puts("syntactically incorrect\n");
        else printf("%lld\n\n", ans);
    }
}

/*
E0 = E1' (+-E1)*
E1 = Ex (*Ex)*
Ex = E2 (%E2)*
E2 = (E0) or R+
E1' = Ex (* Ex)* or -Ex (* Ex)*
*/

```

## 8.8 python 小抄

```

#!/usr/bin/env python3

# 帕斯卡三角形
n = 10
dp = [ [1 for j in range(n)] for i in range(n) ]
for i in range(1,n):
    for j in range(1,n):
        dp[i][j] = dp[i][j-1] + dp[i-1][j]

for i in range(n):
    print( ' '.join( '{:5d}'.format(x) for x in dp[i] )
        )

# EOF
while True:
    try:
        n, m = map(int, input().split())
    except:
        break
    print( min(n,m), max(n,m) )

# input a sequence of number
a = [ int(x) for x in input().split() ]
a.sort()
print( ''.join( str(x)+' ' for x in a ) )

# LCS

```

```

ncase = int( input() )
for _ in range(ncase):
    n, m = [int(x) for x in input().split()]
    a, b = "$"+input(), "$"+input()

    dp = [ [int(0) for j in range(m+1)] for i in range(
        n+1) ]

    for i in range(1,n+1):
        for j in range(1,m+1):
            dp[i][j] = max(dp[i-1][j],dp[i][j-1])
            if a[i]==b[j]:
                dp[i][j] = max(dp[i][j],dp[i-1][j-1]+1)

    for i in range(1,n+1):
        print(dp[i][1:])

    print('a={:s}, b={:s}, /LCS(a,b)/={:d}'.format(a
        [1:],b[1:],dp[n][m]))

# Basic operator
a, b = 10, 20
a/b # 0.5
a//b # 0
a%b # 10
a**b # 10^20

# if, else if, else
if a==0:
    print('zero')
elif a>0:
    print('positive')
else:
    print('negative')

# stack # C++
stack = [3,4,5]
stack.append(6) # push()
stack.pop() # pop()
stack[-1] # top()
len(stack) # size() 0(1)

# queue # C++
from collections import deque
queue = deque([3,4,5])
queue.append(6) # push()
queue.popleft() # pop()
queue[0] # front()
len(queue) # size() 0(1)

```

## 9 Persistence