

Contents

1 Basic	1
1.1 compile	1
1.2 default code	1
1.3 debug list	1
2 Dark Code	1
2.1 IO optimization	1
3 Geometry	1
3.1 2D point	1
3.2 Convex Hull	1
4 Flow	2
4.1 Dinic	2
4.2 min cost flow	2
5 Mathematics	3
5.1 ax+by=gcd(a,b)	3
5.2 BigInt	3
5.3 GaussElimination	4
5.4 Inverse	4
5.5 LinearPrime	4
5.6 Miller Rabin	4
5.7 Pollard's rho	5
5.8 數論基本工具	5
5.9 Mobius	5
5.10SG	5
5.11Theorem	5
6 Graph	6
6.1 BCC	6
6.2 Prim	6
6.3 Bellman Ford	6
6.4 Kruskal	7
6.5 Dijkstra	7
6.6 Strongly Connected Component(SCC)	7
6.7 Hungarian	8
6.8 KM	8
6.9 最小平均環	8
6.10偵測負環	9
6.11Tarjan	9
6.12Topological Sort	9
7 Data Structure	10
7.1 2D Range Tree	10
7.2 Sparse Table	10
7.3 Segment Tree	10
7.4 Lazy Tag	11
8 String	11
8.1 KMP	11
8.2 smallest rotation	11
8.3 Suffix Array	11
8.4 Z-value	11
9 Others	12
9.1 二分搜尋法	12
9.2 矩陣數定理	12
9.3 1D/1D dp 優化	12
9.4 Theorm - DP optimization	13
9.5 Stable Marriage	13
9.6 python 小抄	14
10 Persistence	14

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++17

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

# Run
./a.out

# Run with fileinput
./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

```
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#include <bits/stdc++.h>
using namespace std;
#define IOS ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define int long long
#define F first
#define S second
typedef pair<int,int> pii;

signed main(){
    IOS;
    int tc; cin >> tc;
    while(tc--){
    }
}
```

1.3 debug list

1. bits/stdc++.h 跟 global variable y1 衝突，不能用
2. 事先將把極端測資加入測試
3. 會不會爆 **long long**?
4. STL 容器要清空
5. 是否讀錯題目，想不到時請字翔請專心看題目不要寫 code
6. 比較容易有問題的地方換人寫
7. 注意公式有沒有推錯，codebook 輪流檢查有沒有抄錯
8. 除非還有題目明顯沒有跟到，否則火力集中寫剩下的題目

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, neg = 0, c = readchar();
    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();
    return (neg? x: -x);
}
```

3 Geometry

3.1 2D point

```
typedef double Double;
struct Point {
    Double x,y;

    bool operator < (const Point &b)const{
        //return tie(x,y) < tie(b.x,b.y);
        return atan2(y,x) < atan2(b.y,b.x);
    }
    Point operator + (const Point &b)const{
        return (Point){x+b.x,y+b.y};
    }
    Point operator - (const Point &b)const{
        return (Point){x-b.x,y-b.y};
    }
    Point operator * (const Double &d)const{
        return Point(d*x,d*y);
    }
    Double operator * (const Point &b)const{
        return x*b.x + y*b.y;
    }
    Double operator % (const Point &b)const{
        return x*b.y - y*b.x;
    }
    friend Double abs2(const Point &p){
        return p.x*p.x + p.y*p.y;
    }
    friend Double abs(const Point &p){
        return sqrt( abs2(p) );
    }
};
typedef Point Vector;

struct Line{
    Point P; Vector v;
    bool operator < (const Line &b)const{
        return atan2(v.y,v.x) < atan2(b.v.y,b.v.x);
    }
};
```

3.2 Convex Hull

```
#include "2Dpoint.cpp"

// return H, The first will occurred TWICE in vector H!
void ConvexHull(vector<Point> &P, vector<Point> &H){
    int n = P.size(), m=0;
    sort(P.begin(),P.end());
    H.clear();

    for (int i=0; i<n; i++){
        while (m>=2 && (P[i]-H[m-2]) % (H[m-1]-H[m-2])
            <0)H.pop_back(), m--;
        H.push_back(P[i]), m++;
    }

    for (int i=n-2; i>=0; i--){
        while (m>=2 && (P[i]-H[m-2]) % (H[m-1]-H[m-2])
            <0)H.pop_back(), m--;
        H.push_back(P[i]), m++;
    }
}
```

4 Flow

4.1 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) + ($\sum 1 * \text{cost for every edge}$)

(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;

```

4.2 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;

```

5 Mathematics

5.1 $ax+by=\gcd(a,b)$

```

typedef pair<int, int> pii;

pii exgcd(int a, int b){
    if(b == 0) return make_pair(1, 0);
    else{
        int p = a / b;
        pii q = exgcd(b, a % b);
        int aa = q.second, bb = q.first - q.second * p;
        if(aa < 0) aa += b, bb -= a;
        return make_pair(aa, bb);
    }
}

```

5.2 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 std::ostream& operator << (std::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;
}
};

```

5.3 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;
            }
        }
    }
}

```

5.4 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);
}

```

5.5 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;
        }
    }
}

```

5.6 Miller Rabin

```

typedef long long LL;

inline LL modMul(LL a, LL b, LL m){
    return __int128{a} * b % m;
}

inline LL pow(LL a, LL b, LL m){LL ret = 1;
    for (; b; a = modMul(a, a, m), b >>= 1)
        if (b % 2) ret = modMul(ret, a, m);
    return ret;
}

bool is_prime(LL n){
    //LL sprp[3] = { 2LL, 7LL, 61LL};
    LL sprp[7] = {2, 325, 9375, 28178, 450775, 9780504,
        1795265022};
    if(n == 1 || (n & 1) == 0) return n == 2;
    LL u = n - 1, t = 0; for(; u % 2 == 0; t++) u >>= 1;
    //for(int i = 0; i < "sprp.size()"; i++)
    for(int i = 0; i < 7; i++){ LL a = sprp[i] % n;
        if(a == 0 || a == 1 || a == n - 1) continue;
        LL x = pow(a, u, n); if(x == 1 || x == n-1)
            continue;
        for(int j = 1; j < t; j++){ x = modMul(x, x, n);
            if (x == 1) return 0; if (x == n - 1) break;
        }
        if(x == n - 1)continue; return 0;
    }
    return 1;
}

```

5.7 Pollard's rho

```

// does not work when n is prime
LL pollard_rho(LL n){
    //pre-define f = (x * x + 1) % mod
    if(!(n&1)) return 2;
    while(1){
        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;
    }
}

```

```

}
}

```

5.8 數論基本工具

```

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

void factorize(LL n, vector<LL> &ans){
    if(is_prime(n)){
        ans.push_back(n);
    }else{
        LL p = pollard_rho(n);
        factorize(p, ans);
        factorize(n / p, ans);
    }
}

```

5.9 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];
        }
    }
}

```

5.10 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]
}

```

5.11 Theorem

```

/*
Lucas's Theorem
For non-negative integer n,m and prime P,
 $C(m,n) \bmod P = C(m/M,n/M) * C(m \% M,n \% M) \bmod P$ 
= 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
 $A_{ii} = \deg(i), A_{ij} = (i,j) \in E ? -1 : 0$ 
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
 $u(n) = 1$ , if  $n = 1$ 
 $(-1)^m$ , 若  $n$  無平方數因數, 且  $n = p_1 * p_2 * p_3 * \dots * p_k$ 
 $0$ , 若  $n$  有大於 1 的平方數因數
- Property
1. (積性函數)  $u(a)u(b) = u(ab)$ 
2.  $\sum_{d|n} u(d) = [n == 1]$ 
-----
Mobius Inversion Formula
if  $f(n) = \sum_{d|n} g(d)$ 
then  $g(n) = \sum_{d|n} u(n/d)f(d)$ 
 $= \sum_{d|n} u(d)f(n/d)$ 
- Application
the number/power of gcd(i, j) = k
- Trick
分塊,  $O(\sqrt{n})$ 
-----
Chinese Remainder Theorem ( $m_i$  兩兩互質)

 $x = a_1 \pmod{m_1}$ 
 $x = a_2 \pmod{m_2}$ 
....
 $x = a_i \pmod{m_i}$ 

construct a solution:

Let  $M = m_1 * m_2 * m_3 * \dots * m_n$ 
Let  $M_i = M / m_i$ 

 $t_i = 1 / M_i$ 
 $t_i * M_i = 1 \pmod{m_i}$ 

solution  $x = a_1 * t_1 * M_1 + a_2 * t_2 * M_2 + \dots$ 
 $+ a_n * t_n * M_n + k * M$ 
 $= k * M + \sum a_i * t_i * M_i$ , k is positive integer.

under mod M, there is one solution  $x = \sum a_i * t_i * M_i$ 
-----
Burnside's Lemma
 $|G| * |X/G| = \sum (|X^g|)$  where g 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
*/

```

6 Graph

6.1 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;

```

6.2 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;
}

```

6.3 Bellman Ford

```

int a[100][100], d[100], p[100];

void bellman_ford(int root, int n){
    for (int i = 1; i <= n; i++)d[i] = 1e9;
    d[root] = 0, p[root] = 0;
    for (int i = 0; i < n - 1; i++){
        for (int j = 1; j <= n; j++){
            for (int k = 1; k <= n; k++){
                if (d[j] != 1e9 && a[j][k] != 1e9){
                    if (d[j] + a[j][k] < d[k]){
                        d[k] = d[j] + a[j][k], p[k] = j;
                    }
                }
            }
        }
    }
}

bool nega_cyc(int n){
    for (int i = 1; i <= n; i++){
        for (int j = 1; j <= n; j++){
            if (d[i] != 1e9 && a[i][j] != 1e9)
                if (d[i] + a[i][j] < d[j]){
                    return 0;
                }
        }
    }
    return 1;
}

int main(){
    int n, m, aa, bb, dd;
    while (cin >> n >> m){
        for (int i = 0; i <= n; i++)for (int j = 0; j <= n; j++){
            a[i][j] = 1e9;
        }
        memset(p, 0, sizeof(p));
        for (int i = 0; i < m; i++){
            cin >> aa >> bb >> dd;
            a[aa][bb] = min(a[aa][bb], dd);
        }
        cin >> aa;
        bellman_ford(aa, n);
        int t = nega_cyc(n);
        if(t){
            for (int i = 1; i <= n; i++)cout << d[i] << " \n" [i==n];
            for (int i = 1; i <= n; i++)cout << p[i] << " \n" [i==n];
        }
        else cout << "There is a negative weight cycle in the graph\n";
    }
}

```

6.4 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;
    }
}

```

6.5 Dijkstra

```

int e[300][300], d[300], p[300];

struct node {
    int b, w;
    bool operator < (const node& bb)const {
        return w > bb.w;
    }
};

void dijkstra(int root, int n) {
    for (int i = 0; i <= n; i++)d[i] = (INT_MAX >> 1);
    memset(p, 0, sizeof(p));
    priority_queue<node>yee;
    d[root] = p[root] = 0;
    yee.push({ root, d[root] });

    while (!yee.empty()) {
        node tmp = yee.top(); yee.pop();
        for (int i = 1; i <= n; i++) {
            if (d[i] > d[tmp.b] + e[tmp.b][i]) {
                d[i] = d[tmp.b] + e[tmp.b][i];
                p[i] = tmp.b;
                yee.push( { i, d[tmp.b] } );
            }
        }
    }
}

int main() {
    int n, m, aa, bb, root, cc;
    while (cin >> n >> m) {
        memset(e, 0, sizeof(e));
        for (int i = 0; i <= n; i++)for (int j = 0; j <= n; j++)e[i][j] = (INT_MAX >> 1);
        for (int i = 0; i < m; i++) {
            cin >> aa >> bb >> cc;
            e[aa][bb] = cc;
        }
        cin >> root;
        dijkstra(root, n);
        for (int i = 1; i <= n; i++)cout << d[i] << " \n" [i==n];
        for (int i = 1; i <= n; i++)cout << p[i] << " \n" [i==n];
    }
}

```



```
|}
```

6.6 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++;
        }
    }
}
};
```

6.7 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;
```

6.8 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;

```

6.9 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--][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;
}

```

6.10 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;
}

```

6.11 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;

```

6.12 Topological Sort

```

#define N 87

bool adj[N][N];    // adjacency matrix
int visit[N];      // record visited coordinations in DFS
int order[N], n;   // save the order

bool cycle;        // detect the cycle

void DFS(int s)
{
    // back edge occured, detected the cycle
    if (visit[s] == 1) {cycle = true; return;}
    // forward edge and cross edge
    if (visit[s] == 2) return;

    visit[s] = 1;
    for (int t=0; t<N; ++t){
        if (adj[s][t]) DFS(t);
    }
    visit[s] = 2;
    order[n--] = s;    // record the order
}

void topological_ordering()
{
    memset(visit, 0, sizeof(visit));
    cycle = false;
    n = N - 1;

    for (int s=0; s<9; ++s)
        if (!v[s])
            DFS(s);

    if (cycle) cout << "The graph has the cycle!";
}

```

```

else{
    for (int i=0; i<N; ++i)
        cout << order[i];
    }
}
}

```

7 Data Structure

7.1 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) {
        *(ptr) = pl->pt.y <= pr->pt.y ? cnt++, *(pl++):
            *(pr++);
        ptr -> toleft = cnt; ptr++;
    }
    while(pl != nl) *(ptr) = *(pl++), ptr -> toleft = ++cnt, ptr++;
    while(pr != nr) *(ptr) = *(pr++), ptr -> toleft = cnt, ptr++;
}

int main(){
    int n; cin >> n;
    for(int i = 0 ; i < n; i++){
        T x,y; cin >> x >> y;
        P.push_back((Point){x,y});
    }
    sort(P.begin(),P.end());
    build(0,n-1,1,0);
}

```

7.2 Sparse Table

```
const int MAXN = 200005;
const int lgN = 20;

struct SP{ //sparse table
    int Sp[MAXN][lgN];
    function<int(int,int)> opt;
    void build(int n, int *a){ // 0 base
        for (int i=0 ;i<n; i++){
            Sp[i][0]=a[i];

            for (int h=1; h<lgN; h++){
                int len = 1<<(h-1), i=0;
                for (; i+len<n; i++){
                    Sp[i][h] = opt( Sp[i][h-1] , Sp[i+len][h-1] );
                }
                for (; i<n; i++){
                    Sp[i][h] = Sp[i][h-1];
                }
            }
        }
    }
    int query(int l, int r){
        int h = __lg(r-l+1);
        int len = 1<<h;
        return opt( Sp[l][h] , Sp[r-len+1][h] );
    }
};
```

7.3 Segment Tree

```
// might have some problem
struct node{
    int val;
    node *l, *r;
    node(int v): val(v), l(0), r(0){}
    void pull(){val = min(l->val, r->val);}
};

int arr[N];
node* build(int l, int r, node *p){
    if(l == r) return new node(arr[l]);
    int m = l + r >> 1;
    p = new node(0);
    p->l = build(l, m, p->l), p->r = build(m+1, r, p->r);
    p->pull();
}

int query(int ql, int qr, int l, int r, node *p){
    if(ql <= l && r <= qr) return p->val;
    int m = l + r >> 1;
    if(qr <= m) return query(ql, qr, l, m, p->l);
    if(ql > m) return query(ql, qr, m+1, r, p->r);
    return min(query(ql, qr, l, m, p->l), query(ql, qr, m+1, r, p->r));
}

void modify(int x, int l, int r, node *p, int v){
    if(l == r)
        return p->val = v;
    int m = l + r >> 1;
    if(x <= m) modify(x, l, m, p->l, v);
    else modify(x, m+1, r, p->r, v);
    p->pull();
}
```

7.4 Lazy Tag

```
void modify(type value, int l, int r, int L, int R,
            vertex v){
    if(l == L && r == R){
        //打懶標在v上;
        return;
    }
    int M = (L + R) / 2;
    if(r <= M) modify(value, l, r, L, M, //v的左子節點);
    else if(l > M) modify(value, l, r, M + 1, R, //v的右子節點);
    else{
        modify(value, l, M, L, M, v的左子節點);
        modify(value, M + 1, r, M + 1, R, //v的右子節點);
    }
}
```

```
}
//用兩個子節點的答案更新v的答案;
}
```

8 String

8.1 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;
}
```

8.2 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

8.3 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;
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[tsa[j]][0]]++]=tsa[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;
}
}
}

```

8.4 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;
    }
}

```

9 Others

9.1 二分搜尋法

```

int BinarySearch(vector<int>& nums, int target) {
    int l = 0, r = nums.size() - 1, m;
    while(l <= r){
        m = l + (r - l >> 1);
        if(nums[m] == target) return m;
        else if(target < nums[m]) r = m - 1;
        else l = m + 1;
    }
    if(target < nums[m]) return m;
    return m + 1;
}

```

9.2 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 $n \times n$ 的矩阵，并且满足：当 $i \neq j$ 时， $d_{ij}=0$ ；当 $i=j$ 时， d_{ij} 等于 v_i 的度数。
 - 2、G的邻接矩阵 $A[G]$ 也是一个 $n \times n$ 的矩阵，并且满足：如果 v_i 、 v_j 之间有边直接相连，则 $a_{ij}=1$ ，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 $r(1 \leq r \leq n)$ ，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
 $Matrix[i][j] = \begin{cases} degree(i), & i=j \\ -1, & i-j \text{ 有边} \\ 0, & \text{其他情况} \end{cases}$
- 2、去掉第 r 行，第 r 列 (r 任意)
- 3、计算矩阵的行列式

```

#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;
}

```

9.3 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;
}

```

9.4 Theorem - DP optimization

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

Definition xD/yD

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

Monotonicity

	c	d
a	w(a, c)	w(a, d)
b	w(b, c)	w(b, d)

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) $O(n^3) \rightarrow O(n^2)$

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

9.5 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;
    }
}

```

```

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] )
        )

# EOF1
while True:
    try:
        n, m = map(int, input().split())
    except:
        break

# EOF2
import sys
for s in sys.stdin:
    print(eval(s.replace("/", "//")))

# 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]) )

# list, dict, string
a = [1, 3, 4, 65, 65]
b = list.copy() # b = [1, 3, 4, 65], list a 跟 list b
互相獨立
cnt = list.count(65) # cnt == 2
loc = list.index(65) # loc == 3, find the leftmost
element, if not found then return ERROR
list.sort(reverse = True|False, key = None|lambda x:x
[1]) # list.sort has side effect but no return
value

# 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)

```

10 Persistence

9.6 python 小抄

```
#!/usr/bin/env python3
```

```
# 帕斯卡三角形
n = 10
```