Contents

1 Basic

1.1 Inversion

```cpp
#define L 500010
int arr[L], buf[L];

long long sol(int left, int right){
    if (right - left <= 1){
        return 0;
    }
    int middle = (right + left) / 2;
    long long ans = sol(left, middle) + sol(middle,
        right);
    int i = left, j = middle, k = left;
    while (i < middle || j < right){
        if (i >= middle){
            buf[k] = arr[j++];
        }
        else if (j >= right){
            buf[k] = arr[i++];
        }
        else{
            if (arr[i] <= arr[j]){
                buf[k] = arr[i++];
            }
            else{
                buf[k] = arr[j++];
                ans += middle - i;
            }
        }
        k++;
    }
    for (int k = left; k < right; k++){
        arr[k] = buf[k];
    }
    return ans;
}
```

2 Graph Theory

2.1 Adjacency List

```cpp
vector<int> list[5];

void Adjacency_List(){

    // initial
    for (int i = 0; i < 5; i++)
        list[i].clear();

    int a, b;   // start & end of an edge

    while (cin >> a >> b)
        list[a].push_back(b);
        // list[b].push_back(a);
}
```

2.2 DFS

```cpp
vector<int> G[N];
bitset<N> vis;
void dfs(int s) {
    vis[s] = 1;
    for (int t : G[s]) {
        if (!vis[i])
            dfs(i);
    }
}
```

2.3 BFS

```cpp
vector<int> G[N];
bitset<N> vis;
void bfs(int s) {
    queue<int> q;
    q.push(s);
    vis[s] = 1;
    while (!q.empty()) {
        int v = q.front();
        q.pop();
        for (int t : G[v]) {
            if (!vis[t]) {
                q.push(t);
                vis[t] = 1;
            }
        }
    }
}
```

2.4 Disjoint Set and Kruskal

```cpp
struct Edge{
    int u, v, w;
    // bool operator < (const Edge &rhs) const {
        return w < rhs.w; }
};

vector<int> parent;
vector<Edge> E;

bool cmp(Edge edge1, Edge edge2){
    return edge2.w > edge1.w;
}

int find(int x){
    if(parent[x] < 0){
        return x;
    }
    return parent[x] = find(parent[x]);
}

bool Uni(int a, int b){
    a = find(a);
    b = find(b);
    if(a == b){
        return false;
    }
    if(parent[a] > parent[b]){
        swap(a, b);
    }
    parent[a] = parent[a] + parent[b];
    parent[b] = a;
    return true;
}

void Kruskal() {

    int cost = 0;

    sort(E.begin(), E.end()); // sort by w
    // sort(E.begin(), E.end(), cmp);

    // two edge in the same tree or not
    for (auto it: E){
        it.s = Find(it.s);
        it.t = Find(it.t);
        if (Uni(it.s, it.t)){
            cost = cost + it.w;;
        }
    }
}

int main(){

    // create N space and initial -1
    parent = vector<int> (N, -1);

    for(i = 0; i < M; i++){
        cin >> u >> v >> w;
        E.push_back({u, v, w});
    }

    Kruskal();

    return 0;
}
```

2.5 Floyd-Warshall

```cpp
for (k = 0; k < n; k++){
    for (i = 0; i < n; i++){
        for (j = 0; j < n; j++){
            w[i][j] = w[j][i] = min(w[i][j],
                max(w[i][k], w[k][j]));
```

2.6 Dijkstra

```cpp
struct edge {
  int s, t;
  LL d;
  edge(){};
  edge(int s, int t, LL d) : s(s), t(t), d(d) {}
};

struct heap {
  LL d;
  int p; // point
  heap(){};
  heap(LL d, int p) : d(d), p(p) {}
  bool operator<(const heap &b) const { return d >
      b.d; }
};

int d[N], p[N];
vector<edge> edges;
vector<int> G[N];
bitset<N> vis;

void Dijkstra(int ss){

    priority_queue<heap> Q;

    for (int i = 0; i < V; i++){
        d[i] = INF;
    }

    d[ss] = 0;
    p[ss] = -1;
    vis.reset() : Q.push(heap(0, ss));
    heap x;

    while (!Q.empty()){

        x = Q.top();
        Q.pop();
        int p = x.p;

        if (vis[p])
            continue;
        vis[p] = 1;

        for (int i = 0; i < G[p].size(); i++){
            edge &e = edges[G[p][i]];
            if (d[e.t] > d[p] + e.d){
                d[e.t] = d[p] + e.d;
                p[e.t] = G[p][i];
                Q.push(heap(d[e.t], e.t));
            }
        }
    }
}
```

3 Number Theory

3.1 Modulo

- $(a+b) \bmod p = (a \bmod p + b \bmod p) \bmod p$
- $(a-b) \bmod p = (a \bmod p - b \bmod p + p) \bmod p$
- $(a*b) \bmod p = (a \bmod p \cdot b \bmod p) \bmod p$
- $(a^b) \bmod p = ((a \bmod p)^b) \bmod p$
- $((a+b) \bmod p + c) \bmod p = (a + (b+c)) \bmod p$
- $((a \cdot b) \bmod p \cdot c) \bmod p = (a \cdot (b \cdot c)) \bmod p$
- $(a+b) \bmod p = (b+a) \bmod p$
- $(a \cdot b) \bmod p = (b \cdot a) \bmod p$

- $((a+b) \bmod p \cdot c) = ((a \cdot c) \bmod p + (b \cdot c) \bmod p) \bmod p$
- $a \equiv b \pmod m \Rightarrow c \cdot m = a - b, c \in \mathbb{Z}$
  $\Rightarrow a \equiv b \pmod m \Rightarrow m \mid a - b$
- $a \equiv b \pmod c, b \equiv d \pmod c$
  則 $a \equiv d \pmod c$
- $\begin{cases} a \equiv b \pmod m \\ c \equiv d \pmod m \end{cases} \Rightarrow \begin{cases} a \pm c \equiv b \pm d \pmod m \\ a \cdot c \equiv b \cdot d \pmod m \end{cases}$

3.2 Linear Sieve

```cpp
vector<int> p;
bitset<MAXN> is_notp;
void PrimeTable(int n){

    is_notp.reset();
    is_notp[0] = is_notp[1] = 1;

    for (int i = 2; i <= n; ++i){
        if (!is_notp[i]){
            p.push_back(i);
        }
        for (int j = 0; j < (int)p.size(); ++j){
            if (i * p[j] > n){
                break;
            }

            is_notp[i * p[j]] = 1;

            if (i % p[j] == 0){
                break;
            }
        }
    }
}
```

3.3 Prime Factorization

```cpp
void primeFactorization(int n){
    for(int i = 0; i < (int)p.size(); i++){
        if(p[i] * p[i] > n){
            break;
        }
        if(n % p[i]){
            continue;
        }
        cout << p[i] << ' ';
        while(n % p[i] == 0){
            n /= p[i];
        }
    }
    if(n != 1){
        cout << n << ' ';
    }
    cout << '\n';
}
```

3.4 Exponentiating by Squaring

```cpp
T pow(int a, int b, int c){ // calculate a ^ b % c
    T ans = 1, tmp = a;
    for (; b; b >>= 1) {
        if (b & 1){ // b is odd
            ans = ans * tmp % c;
        }
        tmp = tmp * tmp % c;
    }
  return ans;
}
```

3.5 Euler

```cpp
int Phi(int n){
    int ans = n;
    for (int i: p) {
        if (i * i > n){
            break;
        }
        if (n % i == 0){
            ans /= i;
            ans *= i - 1;
            while (n % i == 0){
                n /= i;
            }
        }
    }
    if (n != 1) {
        ans /= n;
        ans *= n - 1;
    }
    return ans;
}
```

4 Dynamic Programming

4.1 Fibonacci

```cpp
// f(n) = f(n - 1) + f(n - 2)
// f(0) = 0, f(1) = 1
int dp[30];
int f(int n){
    if (dp[n] != -1){
        return dp[n];
    }
    return dp[n] = f(n - 1) + f(n - 2);
}

int main(){
    memset(dp, -1, sizeof(dp));
    dp[0] = 0;
    dp[1] = 1;
    cout << f(25) << '\n';
}
```

4.2 Pascal Triangle

```cpp
// init : f(i, 0) = f(i, i) = 1
// tren : f(i, j) = f(i - 1, j) + f(i - 1, j - 1)
#define N 30
int dp[N][N];
void Pascal_Traingle(void){
    for(int i = 0; i < N; i++){
        dp[i][0] = dp[i][i] = 1;
        for(int j = 1; j < i; j++){
            dp[i][j] = dp[i - 1][j] + dp[i - 1][j -
                1];
        }
    }
}
```

4.3 Robot

```cpp
// f(1, j) = f(i, 1) = 1
// f(i, j) = f(i - 1, j) + f(i, j - 1)
int dp[105][105];
dp[1][1] = 1;
for(int i = 1; i <= 100: ++i){
    for(int j = 1; j <= 100; ++j){
        if(i + 1 <= 100) dp[i + 1][j] += dp[i][j];
```

```
        if(j + 1 <= 100) dp[i][j + 1] += dp[i][j];
    }
}
```

4.4 Max Interval Sum

```
// No Limit
int ans = A[1];
sum[1] = dp[1] = A[1];

for(int i = 2; i <= n; ++i){
    sum[i] = A[i] + sum[i - 1];
    dp[i] = min(dp[i - 1], sum[i]);
    ans = max(ans, sum[i] - dp[i - 1]);
}

// length <= L
int a[15] = {0, 6, -8, 4, -10, 7, 9, -6, 4, 5, -1};
int sum[15];

int main(){
    int L = 3, ans = 0;
    for (int i = 1; i <= 10; ++i)
    {
        sum[i] = a[i] + sum[i - 1];
    }
    deque<int> dq;
    dq.push_back(0);
    for (int i = 1; i <= 10; ++i){
        if (i - dq.front() > L){
            dq.pop_front();
        }
        ans = max(ans, sum[i] - sum[dq.front()]);
        while(!dq.empty() && sum[i] < sum[dq.back()]){
            dq.pop_back();
        }
        dq.push_back(i);
    }
    cout << ans << '\n';
}
```

4.5 Max Area

```
const int N = 25;

int main(){
    int n;
    cin >> n;
    vector<int> H(n + 5), L(n + 5), R(n + 5);
    for (int i = 0; i < n; ++i){
        cin >> H[i];
    }
    stack<int> st;
    // calculate R[]
    for (int i = 0; i < n; ++i){
        while (!st.empty() && H[st.top()] > H[i]){
            R[st.top()] = i - 1;
            st.pop();
        }
        st.push(i);
    }
    while (!st.empty()){
        R[st.top()] = n - 1;
        st.pop();
    }
    // calculate L[]
    for (int i = n - 1; i >= 0; --i){
        while (!st.empty() && H[st.top()] > H[i]){
            L[st.top()] = i + 1;
            st.pop();
        }
        st.push(i);
    }
    while (!st.empty()){
        L[st.top()] = 0;
        st.pop();
    }
    int ans = 0;
    for (int i = 0; i < n; ++i){
        ans = max(ans, H[i] * (R[i] - L[i] + 1));
        cout << i << ' ' << L[i] << ' ' << R[i] <<
            '\n';
    }
    cout << ans << '\n';
}
```

4.6 LCS

```
// init：dp[i][0] = dp[0][i] = 0
// tren：dp[i][j] =
    // if a[i] = b[j]
        // dp[i - 1][j - 1] + 1
    // else
        // max(dp[i - 1][j], dp[i][j - 1])
// LIS
    // init：dp[0] = 0
    // tren：dp[i] = max{dp[j] |  j < i and A[j] <
        A[i]} + 1
// LIS → LCS (嚴格遞增)
    // A 為原序列，B = sort(A)
    // 對 A, B 做 LCS
// LCS → LIS (數字重複、有數字在 B 裡面不在 A 裡面)
    // A, B 為原本的兩序列
    // 對 A 序列作編號轉換，將轉換規則套用在 B
    // 對 B 做 LIS
int dp[a.size() + 1][b.size() + 1];
for(int i = 0; i <= a.size(); i++){
    dp[i][0]= 0;
}
for(int i = 0; i <= b.size(); i++){
    dp[0][i] = 0;
}

for(int i = 1; i <= a.size(); i++){
    for(int j = 1; j <= b.size(); j++){
        if(a[i - 1] == b[j - 1]){
            dp[i][j] = dp[i - 1][j - 1] + 1;
        }
        else{
            dp[i][j] = max(dp[i - 1][j], dp[i][j -
                1]);
        }
    }
}

return 0;
```

4.7 0-1 Bag

```
// 不放：重量和價值不變
    // to f(i, j) = f(i - 1, j)
// 放：重量 + w_i，價值 + v_i
    // to f(i, j) = f(i - 1, j - w_i) + v_i
// tren：f(i, j) = max(f(i - 1, j), f(i - 1, j - w_i)
    + v_i)
int dp[MXN + 1][MXW + 1];
memset(dp, 0, sizeof(dp));
for (int i = 1; i <= MXN; ++i){
    for (int j = 0; j < w[i]; ++j){
        dp[i][j] = dp[i - 1][j];
    }
    for (int j = w[i]; j <= MXW; ++j){
        dp[i][j] = max(dp[i - 1][j - w[i]] + v[i],
            dp[i - 1][j]);
    }
```

```
}
cout << dp[MXN][MXW] << '\n';
```

4.8 Infinite Bag

```
// f(i, j) = max(f(i - 1, j), f(i - 1, j - wi) + vi,
    f(i, j - wi) + vi)
    // coin chage
        // 最少幾枚能湊成 M 元
            //
                f(i,j)=min(f(i-1,j),f(i-1,j-ci)+1,f(i,j-ci)+1)
        // 多少種能湊成 M 元
            // f(i, j) = f(i - 1, j) + f(i, j - ci)
int dp[MXW];
memset(dp, -INF, sizeof(dp));
dp[0] = 0;
for (int i = 0; i < N; ++i){
    for (int j = w[i]; j <= MXW; ++j){
        dp[j] = max(dp[j - w[i]] + v[i], dp[j]);
    }
}
```

4.9 Tree

```
#include <bits/stdc++.h>
using namespace std;
const int MXV = 15;
vector<int> G[MXV];
int high[MXV][2];
int ans[MXV], height[MXV];

void dfs(int u){
    height[u] = 1;
    for (int v : G[u]){
        dfs(v);
        height[u] = max(height[u], height[v] + 1);
        if (high[u][0] == 0 || height[high[u][0]] <
            height[v]){
            high[u][1] = high[u][0];
            high[u][0] = v;
        }
        else if (high[u][1] == 0 ||
            height[high[u][1]] < height[v]){
            high[u][1] = v;
        }
    }
}

void dfs2(int u, int legnth){
    ans[u] = height[high[u][0]] +
        max(height[high[u][1]], legnth) + 1;
    for (int v : G[u]){
        if (v == high[u][0]){
            dfs2(v, max(height[high[u][1]], legnth) +
                1);
        }
        else{
            dfs2(v, max(height[high[u][0]], legnth) +
                1);
        }
    }
}

int main(){
    int n;
    cin >> n;
    for (int i = 1; i < n; ++i){
        int x, y;
        cin >> x >> y;
        G[x].emplace_back(y);
    }
    dfs(1);
    dfs2(1, 0);
    for (int i = 1; i <= n; ++i){
        cout << ans[i] << '\n';
    }
}
```

5 Depth first Search

5.1 Anagram Division

```
// 給一個字串 s 和一個正整數 d，計算 s
    有幾種排列可以被 d 整除
void dfs( int depth, string now ){

    memset( used, true, sizeof(used) );

    // 算此種排列組合是否可以被整除
    if( depth == n ){

        digit = 0;
        for( int i = n - 1; i >= 0; i-- ){
            digit *= 10;
            digit += ( now[ i ] - '0' );
        }
        if( digit % d == 0 ){
            quantity++;
        }
        return;
    }

    // 排列組合
    // 記得用 true/false 確定排過與否
    for( int i = 0; i < n; i++ ){

        if( flag[i] && used[ str[i] - '0' ] ){

            flag[i] = false;
            used[ str[i] - '0' ] = false;

            dfs( depth + 1 , now + str[i] );
            flag[i] = true;

        }
    }
    return;
}



int main(){

    int t;
    cin >> t;

    while( t-- ){

        memset( flag, true, sizeof(flag) );

        cin >> str >> d;
        n = str.size();
        quantity = 0;

        dfs( 0, "" );

        cout << quantity << endl;
        str.clear();
    }
}
```

5.2 Getting in line

```
double calculate( int x1, int y1, int x2, int y2 ){

```

```
    // 計算兩點之間的距離
    // pow 次方 -> pow( 底數, 指數 )
    // sqrt 開根號 -> sqrt( 數 )
    return sqrt( pow( ( x1 - x2 ) , 2 ) + pow( ( y1 -
        y2 ) , 2 ) );

}

void dfs( int depth, double path ){

    if( depth == n ){
        if( path < shortest ){

            shortest = path;
            final_edge.clear();

            for( int i = 0; i < n; i++ ){
                final_edge.push_back( x_now[ i ] );
                final_edge.push_back( y_now[ i ] );
            }
        }
        return;
    }

    // 這次的 dfs 要對每個點做開關 ( true or false )
    // 在做完一趟後 直接更改 depth - 1 的點後 去對
        depth 的點 ( 改變末兩點 )
    // 第二趟時 跟改 depth - 2 的點後
        先依輸入順序填入後面其他點
        而後下幾輪再繼續排列
    for( int i = 0; i < n; i++ ){

        if( flag[i] ){

            flag[i] = false;

            x_now[ depth ] = x[i];
            y_now[ depth ] = y[i];

            if( depth == 0 ){
                dfs( depth + 1, 0 );
            }
            else{
                dfs( depth + 1, path + 16 +
                    calculate( x_now[ depth ], y_now[
                    depth ], x_now[ depth - 1 ],
                    y_now[ depth - 1 ] ) );
            }
            flag[i] = true;
        }
    }
}

int main(){

    int num = 1;

    while( cin >> n && n ){

        int edge;
        // 先隨便設個最小值
        shortest = 2147483647;
        for( int i = 0; i < n; i++ ){

            cin >> edge;
            x.push_back(edge);

            cin >> edge;
            y.push_back(edge);

            flag.insert( pair<int, bool>( i, true ) );
        }

        dfs( 0, 0 );

            ...
    }
}
```

5.3 Lotto

```
void dfs( int depth, int now ){

    // 題目要求每 6 個元素做排列組合
    if( depth == 6 ){

        for( int i = 0; i < 6; i++ ){
            if( i ){
                cout << " ";
            }
            cout << ans[i];
        }
        cout << endl;

        // 這個 return 很重要!! 沒有他會 RE
        return;
    }

    for( int i = now; i < k; i++ ){

        ans[ depth ] = input[ i ];
        dfs( depth + 1, i + 1 );

        // 當 depth = 6 後 會回來做這個 for 迴圈
        // 此時 depth = 5 回到上一次 call dfs 前的深度
        // 此時 i = i ，但因此時 for 迴圈走向下一迴
            i++ 於是 i = i + 1
        // 然後將 input[i] 的值 覆蓋過 ans[5] 接著
            call dfs 去輸出 再 return 回來
        // 依此類推 當 depth = 5 做完後 會到 depth =
            4 ...
    }
}

int main(){

    bool flag = false;
    while( cin >> k && k ){

        if( flag ){
            cout << endl;
        }

        int n;
        for( int i = 0; i < k; i++ ){
            cin >> n;
            input.push_back(n);
        }

        // 從深度為 0 開始往下
        dfs( 0, 0 );

        flag = true;
        input.clear();
    }
}
```

6 Breadth first Search

6.1 Fire

```
int step[4][2] = { { 0, -1 }, { 0, 1 }, { -1, 0 }, {
    1, 0 } };

deque< pair<int,int> > fn;
deque< pair<int,int> > joen;

```

```
void bfs_fire( int n ){

    for( int i = 0; i < 4 ; i++ ){

        int xx = fn[n].first + step[i][0];
        int yx = fn[n].second + step[i][1];

        if( xx > 0 && xx <= r && yx > 0 && yx <= c ){

            if( mp[ xx ][ yx ] == '.' ){
                mp[ xx ][ yx ] = 'F';
                fn.push_back( make_pair( xx, yx ) );
            }
        }
    }
    vis_f++;
}

void bfs_joe( int n ){

    for( int i = 0; i < 4; i++ ){

        int xx = joen[n].first + step[i][0];
        int yx = joen[n].second + step[i][1];

        if( mp[ xx ][ yx ] == '.' ){

            mp[ xx ][ yx ] = 'J';
            escape = true;
            joen.push_back( make_pair( xx, yx ) );
        }
        if( mp[ xx ][ yx ] == ' ' ){

            fin = true;
            break;
        }
    }
    vis_j++;
}

int main(){

    cin >> t;
    while( t-- ){

        cin >> r >> c;
        memset( mp, ' ', sizeof(mp) );

        while( !fn.empty() ){
            fn.pop_front();
        }
        while( !joen.empty() ){
            joen.pop_front();
        }

        for( int i = 1; i <= r; i++ ){
            for( int j = 1; j <= c; j++ ){

                cin >> mp[i][j];
                if( mp[i][j] == 'F' ){
                    fn.push_back( make_pair( i, j ) );
                }
                if( mp[i][j] == 'J' ){
                    joen.push_back( make_pair( i, j )
                        );
                }
            }
        }

        times = 0;
        escape = true;
        fin = false;

        vis_f = 0;
        vis_j = 0;

        while( escape ){

            escape = false;
            times++;
            max_f = fn.size();
            max_j = joen.size();

            for( int i = vis_f; i < max_f; i++ ){
                bfs_fire( i );
            }
            for( int i = vis_j; i < max_j; i++ ){
                bfs_joe( i );
            }

            if( fin ){
                cout << times << endl;
                break;
            }
        }

        if( !fin ){
            cout << "IMPOSSIBLE" << endl;
        }
        check++;
    }
}
```

6.2 Knights

```
int row[8] = { 1, 2, 2, 1, -1, -2, -2, -1 };
int column[8] = { 2, 1, -1, -2, -2, -1, 1, 2 };

int bfs(){

    chess[0][0] = letter_start;
    chess[0][1] = digit_start;
    visited[ letter_start ][ digit_start ] = true;

    for( int i = 0, knights = 1; i < knights; i++ ){

        letter_now = chess[i][0];
        digit_now = chess[i][1];

        if( letter_now == letter_end && digit_now ==
            digit_end ){
            return step[ letter_now ][ digit_now ];
        }

        for( int j = 0; j < 8; j++ ){

            letter_next = letter_now + column[j];
            digit_next = digit_now + row[j];

            if( letter_next < 1 || digit_next < 1 ||
                letter_next > 8 || digit_next > 8 ||
                visited[ letter_next ][ digit_next ]
                ){
                continue;
            }
            else{
                visited[ letter_next ][ digit_next ]
                    = true;
                step[ letter_next ][ digit_next ] =
                    step[ letter_now ][ digit_now ] +
                    1;

                chess[ knights ][0] = letter_next;
                chess[ knights ][1] = digit_next;

                knights++;
            }
        }
    }
    return -1;
}

int main(){
```

```
    while( cin >> letter1 >> digit_start >> letter2 >> digit_end ){

        letter_start = letter1 - 'a' + 1;
        letter_end = letter2 - 'a' + 1;

        for( int i = 0; i < 10; i++ ){
            for( int j = 0; j < 10; j++ ){
                step[i][j] = 0;
                visited[i][j] = false;
            }
        }
        cout << bfs() << " knight moves." << endl;
    }
}
```

6.3 Oil Deposits

```
int row[] = { 1, 1, 1, 0, -1, -1, -1, 0 };
int column[] = { -1, 0, 1, 1, 1, 0, -1, -1 };

void bfs( int x_now, int y_now ){

    for( int j = 0; j < 8; j++ ){

        x_next = x_now + row[j];
        y_next = y_now + column[j];

        if( x_next < m && y_next < n && x_next >= 0 && y_next >= 0 && oil[ x_next ][ y_next ] == '@' ){

            // 此點已找過 就把他改成普通地板
            oil[ x_next ][ y_next ] = '*';
            bfs( x_next, y_next );
        }
    }
    return;
}

int main(){

    while( cin >> m >> n && m && n ){

        memset( oil, '0', sizeof(oil) );
        ans = 0;

        for( int i = 0; i < m; i++ ){
            for( int j = 0; j < n; j++ ){
                cin >> oil[i][j];
            }
        }

        for( int i = 0; i < m; i++ ){
            for( int j = 0; j < n; j++ ){

                if( oil[i][j] == '@' ){
                   ans++;
                   bfs( i, j );
                }

            }
        }
        cout << ans << endl;
    }
}
```

6.4 Rat Attack

```
void bfs( int xn, int yn ){

    int xx, yx;

    // 從 -d 到 d 之間的所有格子
    // 因為起始點是中心，所以要用 -d ~ d 的方式算點
    for( int i = 0 - d; i <= d; i++ ){
        for( int k = 0 - d; k <= d; k++ ){

            xx = xn + i;
            yx = yn + k;

            if( xx >= 0 && xx < 1025 && yx >= 0 && yx < 1025 ){

                maxi[ xx ][ yx ] += rat[ xn ][ yn ];

                if( maxi[ xx ][ yx ] > maxm[2] ){
                    maxm[0] = xx;
                    maxm[1] = yx;
                    maxm[2] = maxi[ xx ][ yx ];
                }
            }
        }
    }
}

int main(){

    int t;
    cin >> t;

    while( t-- ){

        memset( rat, 0, sizeof(rat) );
        memset( maxi, 0, sizeof(maxi) );
        memset( check, 0, sizeof(check) );
        memset( maxm, 0, sizeof(maxm) );

        cin >> d >> n;
        int num = 0;

        for( int i = 0; i < n; i++ ){

            int a, b;
            cin >> a >> b;
            cin >> rat[a][b];

            check[num][0] = a;
            check[num][1] = b;
            num++;

        }

        for( int i = 0; i < num; i++ ){
            bfs( check[i][0], check[i][1] );
        }

       for( int i = 0; i < 3; i++ ){

            if( i != 2 ){
                cout << maxm[i] << " ";
            }
            else{
                cout << maxm[i] << endl;
            }
        }
    }
}
```

7 MATH

7.1 Fraction

```
#include<iostream>
using namespace std;

```

```cpp
// 1/k = 1/x + 1/y
// 給你 k，請你寫一個程式找出所有的 x 和 y
int main(){

    int n;
    while(cin>>n){

        int i;
        int N[10000+5][2]={0};
        int flag=0;

        for(i=n+1; i<= 2*n; i++){

            int r = i-n;

            if((n*i)% r ==0){

                N[flag][0]= (n*i)/r;
                N[flag][1]= i;
                flag++;

            }
        }

        cout<< flag << endl;
        for(i=0; i<flag; i++){
            cout<< "1/" << n << " = 1/" << N[i][0] <<
                " + 1/" << N[i][1] << endl;
        }
    }
    return 0;
}
```

7.2 Slope

```cpp
// 八皇后 上下左右斜行皆不重複
int check( int x, int y ){

    for( int i = 0; i < x; i++ ){

        if( dq[1][i] == y ){
            return 0;
        }

        // 如果兩皇后在同一斜線上 其斜率為 1
        // 如果 x2 - x1 == y2 - y1 -> y2 - y1 / x2 -
            x1 == 1
        if( abs( x - i ) == abs( dq[1][i] - y ) ){
            return 0;
        }

    }
    return 1;
}
```

7.3 GCD

```cpp
// 如果兩數互質 最終結果其中一方為0時 另一方必為1
// 若兩數有公因數 最終結果其中一方為0時 另一方必不為1
    while ( ( num1 %= num2 ) != 0 && ( num2 %= num1 )
        != 0 );
```

8 Others

8.1 Enumerate Twopointers

```cpp
#include<bits/stdc++.h>
using namespace std;
//2021.09.14

int main(){

    long long int t;
    cin >> t;
    while( t-- ){

        long long int n;
        deque<int> snowflakes;
        set<long long int> difference;

        cin >> n;
        for( int i = 0; i < n; i++ ){

            int m;
            cin >> m;
            snowflakes.push_back(m);

        }
        int longest = 0;

        // 利用 L左指標 和 R右指標
        // 每次迭代右指標先往前一個位置
        for( int L = 0, R = 0; R < n; R++ ){

            // 利用 set.count 先確認 set
                內是否有重複元素
            while( difference.count( snowflakes[R] )
                ){

                // 如果有 利用 set.erase 和左指標
                // 將與 右指標重複的元素
                    以前的所有元素刪除
                difference.erase( snowflakes[ L++ ] );

            }

            difference.insert( snowflakes[R] );

            // std::max 可比較兩者之間誰大
            // max( 比較方1, 比較方2 )
                比較方可以不一定是 int
                但一定要相同型態
            longest = max( longest,
                (int)difference.size() );
        }

        cout << longest << endl;
        difference.clear();
        snowflakes.clear();
    }
}
```

8.2 Physics

```cpp
int main(){

    // s = vot + 1/2 at^2
    // v = vo + at
    // a = ( v - vo ) / t

    // vo = 0
    // a = v / t
    // s = 0 + 1/2 v/t 2t^2 = 1/2 v 4t = 2vt

    int v,t;
    while( cin >> v >> t ){
        int s;
        s =  2 * v * t ;
        cout << s << endl;
    }
}
```

8.3 Week

```cpp
int main(){

    int month[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    string week[] = { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday" };

    int n;
    cin >> n;
    while( n-- ){

        int m, d;
        cin >> m >> d;
        int w = 4;

        for( int i = 0; i<m-1 ; i++ )
            w += month[i];

        cout << week[ ( w + d )% 7 ] << endl;
    }
}
```

8.4 Carry Change

```cpp
// 題目給定一個 N 進制 (2 <= N <= 62) 的數字 R，R 保證可以被 (N-1) 整除
// 求符合提議的最小 N
// 當 N = 62 時，用來表示62進制的字符為 0..9, A..Z, a..z。

int main(){

    // R = 265
    // = 2*N*N + 6*N + 5
    // = 2*N*(N-1+1) + 6*N + 5
    // = 2*N*(N-1) + 2*(N-1+1) + 6*(N-1+1) + 5
    // = 2*N*(N-1) + 2*(N-1) + 2 + 6*(N-1) + 6 + 5
    // = (2*N + 2 + 6)*(N-1) + (2 + 6 + 5)
    // because R % N-1 == 0
    // so (2+6+5) == N-1

    string str;
    while( getline( cin, str ) ){

        int tmp;
        int max = 1, sum = 0;
        bool flag = true;

        for( int i = 0; i < str.size(); i++ ){

            if( str[i] >= '0' && str[i] <= '9' ){
                tmp = str[i] - '0';
            }
            else if( str[i] >= 'A' && str[i] <= 'Z' ){
                tmp = str[i] - 'A' + 10;
            }
            else if( str[i] >= 'a' && str[i] <= 'z' ){
                tmp = str[i] - 'a' + 10 + 26;
            }
            else{
                continue;
            }

            if( tmp > max){
                max = tmp;
            }
            sum += tmp;
        }
        for( int i = max; i < 62; i++ ){
            if( !( sum % i ) ){
                cout << i + 1 << endl;
                flag = false;
                break;
            }
        }
        if( flag ){
            cout << "such number is impossible!" << endl;
        }
        str.clear();
    }
}
```

8.5 Recursive

```cpp
int gcd( int i, int j ){

    while( ( j %= i ) != 0  && ( i %= j ) != 0 );
    return j + i;

}

int g( int n ){

    // 已使用過此數字 直接從陣列中呼叫
    if( known[n] ){
        return known[n];
    }

    else{

        // 利用 g( n - 1) 去確認此次輪迴為尚不知道結果的最大數字
        known[n] += g( n - 1 );

        // 計算本次結果 同時將本次結果儲存於陣列中
        for( int i = 1; i < n; i++ ){
            known[n] += gcd( i, n );
        }
        return known[n];
    }
}

int main(){

    known[2] = 1;
    int n;

    while( cin >> n ){

        if( n == 0 ){
            break;
        }

        cout << g(n) << endl;

        // 題目方法
        // for( int i = 1; i < n; i++ ){
        //     for( int j = i + 1; j <= n; j++ ){
        //         g += gcd( i, j );
        //     }
        // }
    }
}
```

8.6 Prime List

```cpp
bool prime[1000000];

// memset: 對一段內存空間全部設置為某個字符 常用於初始化字串、陣列..
// memset( 陣列名稱, 初始化成甚麼, 範圍 )
    memset( prime, false, sizeof(prime) );
    memset( prime, true, 2);

```

```
    for( int i = 2; i < 1000000; i++ ){

        if( !prime[i] ){

            for( int j = i + i; j < 1000000; j += i ){

                prime[j] = true;

            }
        }
    }
```

8.7 Probability

```
int main(){

    int s, n, i;
    double p, p2, ans;
    cin >> s;

    while( s-- ){

        cin >> n >> p >> i;
        p2 = pow( 1.0 - p , n );

        if( p2 == 1){

            cout << "0.0000" << endl;
            continue;

        }

        //第i個人成功的機率 /
            全部的人有機會成功的機率(1-全部人都失敗)
        ans = p * pow( 1.0 - p , i-1 ) / ( 1.0 - pow(
            1.0 - p , n ) );
        cout << fixed << setprecision(4) << ans
            <<endl;
    }
}
```

8.8 distance

```
double calculate( int x1, int y1, int x2, int y2 ){

    // 計算兩點之間的距離
    // pow 次方 -> pow( 底數, 指數 )
    // sqrt 開根號 -> sqrt( 數 )
    return sqrt( pow( ( x1 - x2 ) , 2 ) + pow( ( y1 -
        y2 ) , 2 ) );

}
```

8.9 floor

```
int main(){

    int a, b;
    while( cin >> a >> b  && ( a || b ) ){

        int aa, bb, check = 0;
        if( floor( sqrt(a) ) == sqrt(a) ){
            check = 1;
        }
        double count = 0;

        //floor -> 不大於 x 的最大整數 ( 浮點型 )
        count = floor( sqrt(b) ) - floor( sqrt(a) ) +
            check;

        cout << (int)count << endl;
    }
}
```

8.10 map

```
int main(){

    int n;
    string country, name;

    map<string, int> m;
    map<string, int>::iterator it;

    cin>>n;
    while( n-- ){

        cin >> country ;
        getline(cin, name);
        it = m.find( country );

        if( it != m.end() ){
            m[ country ]++;
        }
        else{
            m.insert( pair<string, int>( country, 1)
                );
        }
    }
    for( auto i = m.begin(); i != m.end(); i++ ){
        cout << i->first << " " << i->second <<endl;
    }
}
```

8.11 set intersection

```
int main(){

    while(getline(cin,str1) && getline(cin, str2)){

        sort(str1.begin(), str1.end());
        sort(str2.begin(), str2.end());

        deque<char> dq;
        dq.clear();

        // set_intersection 在 C++ 中查詢集合交集
        // ex str1 = {1,2,3,4,5,6,7,8}, str2 =
            {5,7,9,10}
        // output = 5 7
        // set_intersection( 字串1頭, 字串1尾,
            字串2頭, 字串2尾, 比較完要放進的地方 )
        set_intersection(str1.begin(), str1.end(),
            str2.begin(), str2.end(),
            insert_iterator<deque<char>>(dq,dq.begin()));

        for( int i=0; i<dq.size(); i++){

            cout<<dq[i];

        }
        cout << endl;

    }
}
```

8.12 setprecision

```
double x = 10.19395;

    // 總共輸出三位數
    cout << setprecision(3) << x << endl;
```

```
5
6       // 輸出小數點後三位數
7       cout << fixed << setprecision(3) << x;
8
9   // output:
10  // 10.2
11  // 10.194
12  // 都會自動四捨五入進位
```