

## Contents

|                                |    |
|--------------------------------|----|
| 1 Basic                        | 1  |
| 1.1 Inversion                  | 1  |
| 2 Graph Theory                 | 1  |
| 2.1 Adjacency List             | 1  |
| 2.2 DFS                        | 1  |
| 2.3 BFS                        | 1  |
| 2.4 Disjoint Set and Kruskal   | 2  |
| 2.5 Floyd-Warshall             | 2  |
| 2.6 Dijkstra                   | 2  |
| 3 Number Theory                | 2  |
| 3.1 Modulo                     | 2  |
| 3.2 Linear Sieve               | 3  |
| 3.3 Prime Factorization        | 3  |
| 3.4 Exponentiating by Squaring | 3  |
| 3.5 Euler                      | 3  |
| 4 Dynamic Programming          | 3  |
| 4.1 Fibonacci                  | 3  |
| 4.2 Pascal Triangle            | 3  |
| 4.3 Robot                      | 3  |
| 4.4 Max Interval Sum           | 4  |
| 4.5 Max Area                   | 4  |
| 4.6 LCS                        | 4  |
| 4.7 0-1 Bag                    | 4  |
| 4.8 Infinite Bag               | 5  |
| 4.9 Tree                       | 5  |
| 5 Depth first Search           | 5  |
| 5.1 Anagram Division           | 5  |
| 5.2 Getting in line            | 5  |
| 5.3 Lotto                      | 6  |
| 6 Breadth first Search         | 6  |
| 6.1 Fire                       | 6  |
| 6.2 Knights                    | 7  |
| 6.3 Oil Deposits               | 8  |
| 6.4 Rat Attack                 | 8  |
| 7 MATH                         | 8  |
| 7.1 Fraction                   | 8  |
| 7.2 Slope                      | 9  |
| 7.3 GCD                        | 9  |
| 8 Others                       | 9  |
| 8.1 Enumerate Twopointers      | 9  |
| 8.2 Physics                    | 9  |
| 8.3 Week                       | 10 |
| 8.4 Carry Change               | 10 |
| 8.5 Recursive                  | 10 |
| 8.6 Prime List                 | 10 |
| 8.7 Probability                | 11 |
| 8.8 distance                   | 11 |
| 8.9 floor                      | 11 |
| 8.10 map                       | 11 |
| 8.11 set intersection          | 11 |
| 8.12 setprecision              | 11 |

## 1 Basic

### 1.1 Inversion

```

1 #define L 500010
2 int arr[L], buf[L];
3
4 long long sol(int left, int right){
5     if (right - left <= 1){
6         return 0;
7     }
8     int middle = (right + left) / 2;
9     long long ans = sol(left, middle) + sol(middle,
10         right);
11     int i = left, j = middle, k = left;
12     while (i < middle || j < right){
13         if (i >= middle){
14             buf[k] = arr[j++];
15         }
16         else if (j >= right){
17             buf[k] = arr[i++];
18         }
19     }
20     for (int k = left; k < right; k++){
21         arr[k] = buf[k];
22     }
23     return ans;
24 }

```

```

18     else{
19         if (arr[i] <= arr[j]){
20             buf[k] = arr[i++];
21         }
22         else{
23             buf[k] = arr[j++];
24             ans += middle - i;
25         }
26     }
27     k++;
28 }
29 for (int k = left; k < right; k++){
30     arr[k] = buf[k];
31 }
32 return ans;
33 }

```

## 2 Graph Theory

### 2.1 Adjacency List

```

1 vector<int> list[5];
2
3 void Adjacency_List(){
4
5     // initial
6     for (int i = 0; i < 5; i++){
7         list[i].clear();
8     }
9
10    int a, b;    // start & end of an edge
11
12    while (cin >> a >> b)
13        list[a].push_back(b);
14    // list[b].push_back(a);
15 }

```

### 2.2 DFS

```

1 vector<int> G[N];
2 bitset<N> vis;
3 void dfs(int s) {
4     vis[s] = 1;
5     for (int t : G[s]) {
6         if (!vis[t])
7             dfs(t);
8     }
9 }

```

### 2.3 BFS

```

1 vector<int> G[N];
2 bitset<N> vis;
3 void bfs(int s) {
4     queue<int> q;
5     q.push(s);
6     vis[s] = 1;
7     while (!q.empty()) {
8         int v = q.front();
9         q.pop();
10        for (int t : G[v]) {
11            if (!vis[t]) {
12                q.push(t);
13                vis[t] = 1;
14            }
15        }
16    }
17 }

```

## 2.4 Disjoint Set and Kruskal

```

1 struct Edge{
2     int u, v, w;
3     // bool operator < (const Edge &rhs) const {
4         // return w < rhs.w; }
5 };
6 vector<int> parent;
7 vector<Edge> E;
8
9 bool cmp(Edge edge1, Edge edge2){
10     return edge2.w > edge1.w;
11 }
12
13 int find(int x){
14     if(parent[x] < 0){
15         return x;
16     }
17     return parent[x] = find(parent[x]);
18 }
19
20 bool Uni(int a, int b){
21     a = find(a);
22     b = find(b);
23     if(a == b){
24         return false;
25     }
26     if(parent[a] > parent[b]){
27         swap(a, b);
28     }
29     parent[a] = parent[a] + parent[b];
30     parent[b] = a;
31     return true;
32 }
33
34 void Kruskal() {
35     int cost = 0;
36
37     sort(E.begin(), E.end()); // sort by w
38     // sort(E.begin(), E.end(), cmp);
39
40     // two edge in the same tree or not
41     for (auto it: E){
42         it.s = Find(it.s);
43         it.t = Find(it.t);
44         if (Uni(it.s, it.t)){
45             cost = cost + it.w;;
46         }
47     }
48 }
49
50 int main(){
51
52     // create N space and initial -1
53     parent = vector<int> (N, -1);
54
55     for(i = 0; i < M; i++){
56         cin >> u >> v >> w;
57         E.push_back({u, v, w});
58     }
59
60     Kruskal();
61
62     return 0;
63 }
64

```

## 2.5 Floyd-Warshall

```

1 for (k = 0; k < n; k++){
2     for (i = 0; i < n; i++){
3         for (j = 0; j < n; j++){
4             w[i][j] = w[j][i] = min(w[i][j],
5                                     max(w[i][k], w[k][j]));
6         }
7     }
8 }

```

## 2.6 Dijkstra

```

1 struct edge {
2     int s, t;
3     LL d;
4     edge(){};
5     edge(int s, int t, LL d) : s(s), t(t), d(d) {}
6 };
7
8 struct heap {
9     LL d;
10    int p; // point
11    heap(){};
12    heap(LL d, int p) : d(d), p(p) {}
13    bool operator<(const heap &b) const { return d >
14        b.d; }
15 };
16 int d[N], p[N];
17 vector<edge> edges;
18 vector<int> G[N];
19 bitset<N> vis;
20
21 void Dijkstra(int ss){
22
23     priority_queue<heap> Q;
24
25     for (int i = 0; i < V; i++){
26         d[i] = INF;
27     }
28
29     d[ss] = 0;
30     p[ss] = -1;
31     vis.reset() : Q.push(heap(0, ss));
32     heap x;
33
34     while (!Q.empty()){
35
36         x = Q.top();
37         Q.pop();
38         int p = x.p;
39
40         if (vis[p])
41             continue;
42         vis[p] = 1;
43
44         for (int i = 0; i < G[p].size(); i++){
45             edge &e = edges[G[p][i]];
46             if (d[e.t] > d[p] + e.d){
47                 d[e.t] = d[p] + e.d;
48                 p[e.t] = G[p][i];
49                 Q.push(heap(d[e.t], e.t));
50             }
51         }
52     }
53 }

```

## 3 Number Theory

### 3.1 Modulo

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

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

```

1 vector<int> p;
2 bitset<MAXN> is_notp;
3 void PrimeTable(int n){
4
5     is_notp.reset();
6     is_notp[0] = is_notp[1] = 1;
7
8     for (int i = 2; i <= n; ++i){
9         if (!is_notp[i]){
10             p.push_back(i);
11         }
12         for (int j = 0; j < (int)p.size(); ++j){
13             if (i * p[j] > n){
14                 break;
15             }
16             is_notp[i * p[j]] = 1;
17
18             if (i % p[j] == 0){
19                 break;
20             }
21         }
22     }
23 }
24 }

```

### 3.3 Prime Factorization

```

1 void primeFactorization(int n){
2     for(int i = 0; i < (int)p.size(); i++){
3         if(p[i] * p[i] > n){
4             break;
5         }
6         if(n % p[i]){
7             continue;
8         }
9         cout << p[i] << ' ';
10        while(n % p[i] == 0){
11            n /= p[i];
12        }
13    }
14    if(n != 1){
15        cout << n << ' ';
16    }
17    cout << '\n';
18 }

```

### 3.4 Exponentiating by Squaring

```

1 T pow(int a, int b, int c){ // calculate a ^ b % c
2     T ans = 1, tmp = a;
3     for (; b; b >>= 1) {
4         if (b & 1){ // b is odd
5             ans = ans * tmp % c;
6         }
7         tmp = tmp * tmp % c;
8     }
9     return ans;
10 }

```

### 3.5 Euler

```

1 int Phi(int n){
2     int ans = n;
3     for (int i: p) {
4         if (i * i > n){
5             break;
6         }
7         if (n % i == 0){
8             ans /= i;
9             ans *= i - 1;
10            while (n % i == 0){
11                n /= i;
12            }
13        }
14    }
15    if (n != 1) {
16        ans /= n;
17        ans *= n - 1;
18    }
19    return ans;
20 }

```

## 4 Dynamic Programming

### 4.1 Fibonacci

```

1 // f(n) = f(n - 1) + f(n - 2)
2 // f(0) = 0, f(1) = 1
3 int dp[30];
4 int f(int n){
5     if (dp[n] != -1){
6         return dp[n];
7     }
8     return dp[n] = f(n - 1) + f(n - 2);
9 }
10
11 int main(){
12     memset(dp, -1, sizeof(dp));
13     dp[0] = 0;
14     dp[1] = 1;
15     cout << f(25) << '\n';
16 }

```

### 4.2 Pascal Triangle

```

1 // init: f(i, 0) = f(i, i) = 1
2 // tren: f(i, j) = f(i - 1, j) + f(i - 1, j - 1)
3 #define N 30
4 int dp[N][N];
5 void Pascal_Traingle(void){
6     for(int i = 0; i < N; i++){
7         dp[i][0] = dp[i][i] = 1;
8         for(int j = 1; j < i; j++){
9             dp[i][j] = dp[i - 1][j] + dp[i - 1][j - 1];
10        }
11    }
12 }

```

### 4.3 Robot

```

1 // f(1, j) = f(i, 1) = 1
2 // f(i, j) = f(i - 1, j) + f(i, j - 1)
3 int dp[105][105];
4 dp[1][1] = 1;
5 for(int i = 1; i <= 100; ++i){
6     for(int j = 1; j <= 100; ++j){
7         if(i + 1 <= 100) dp[i + 1][j] += dp[i][j];

```

```

8         if(j + 1 <= 100) dp[i][j + 1] += dp[i][j];
9     }
10 }

```

#### 4.4 Max Interval Sum

```

1 // No Limit
2 int ans = A[1];
3 sum[1] = dp[1] = A[1];
4
5 for(int i = 2; i <= n; ++i){
6     sum[i] = A[i] + sum[i - 1];
7     dp[i] = min(dp[i - 1], sum[i]);
8     ans = max(ans, sum[i] - dp[i - 1]);
9 }
10
11 // length <= L
12 int a[15] = {0, 6, -8, 4, -10, 7, 9, -6, 4, 5, -1};
13 int sum[15];
14
15 int main(){
16     int L = 3, ans = 0;
17     for (int i = 1; i <= 10; ++i)
18     {
19         sum[i] = a[i] + sum[i - 1];
20     }
21     deque<int> dq;
22     dq.push_back(0);
23     for (int i = 1; i <= 10; ++i){
24         if (i - dq.front() > L){
25             dq.pop_front();
26         }
27         ans = max(ans, sum[i] - sum[dq.front()]);
28         while(!dq.empty() && sum[i] < sum[dq.back()]){
29             dq.pop_back();
30         }
31         dq.push_back(i);
32     }
33     cout << ans << '\n';
34 }

```

#### 4.5 Max Area

```

1 const int N = 25;
2
3 int main(){
4     int n;
5     cin >> n;
6     vector<int> H(n + 5), L(n + 5), R(n + 5);
7     for (int i = 0; i < n; ++i){
8         cin >> H[i];
9     }
10     stack<int> st;
11     // calculate R[]
12     for (int i = 0; i < n; ++i){
13         while (!st.empty() && H[st.top()] > H[i]){
14             R[st.top()] = i - 1;
15             st.pop();
16         }
17         st.push(i);
18     }
19     while (!st.empty()){
20         R[st.top()] = n - 1;
21         st.pop();
22     }
23     // calculate L[]
24     for (int i = n - 1; i >= 0; --i){
25         while (!st.empty() && H[st.top()] > H[i]){
26             L[st.top()] = i + 1;
27             st.pop();
28         }
29         st.push(i);
30     }

```

```

31 while (!st.empty()){
32     L[st.top()] = 0;
33     st.pop();
34 }
35 int ans = 0;
36 for (int i = 0; i < n; ++i){
37     ans = max(ans, H[i] * (R[i] - L[i] + 1));
38     cout << i << ' ' << L[i] << ' ' << R[i] << '\n';
39 }
40 cout << ans << '\n';
41 }

```

#### 4.6 LCS

```

1 // init: dp[i][0] = dp[0][i] = 0
2 // tren: dp[i][j] =
3 // if a[i] == b[j]
4 // dp[i - 1][j - 1] + 1
5 // else
6 // max(dp[i - 1][j], dp[i][j - 1])
7 // LIS
8 // init: dp[0] = 0
9 // tren: dp[i] = max{dp[j] | j < i and A[j] < A[i]} + 1
10 // LIS → LCS (嚴格遞增)
11 // A 為原序列, B = sort(A)
12 // 對 A, B 做 LCS
13 // LCS → LIS (數字重複、有數字在 B 裡面不在 A 裡面)
14 // A, B 為原本的兩序列
15 // 對 A 序列作編號轉換, 將轉換規則套用在 B
16 // 對 B 做 LIS
17 int dp[a.size() + 1][b.size() + 1];
18 for(int i = 0; i <= a.size(); i++){
19     dp[i][0] = 0;
20 }
21 for(int i = 0; i <= b.size(); i++){
22     dp[0][i] = 0;
23 }
24
25 for(int i = 1; i <= a.size(); i++){
26     for(int j = 1; j <= b.size(); j++){
27         if(a[i - 1] == b[j - 1]){
28             dp[i][j] = dp[i - 1][j - 1] + 1;
29         }
30         else{
31             dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
32         }
33     }
34 }
35
36 return 0;

```

#### 4.7 0-1 Bag

```

1 // 不放：重量和價值不變
2 // to f(i, j) = f(i - 1, j)
3 // 放：重量 + w_i, 價值 + v_i
4 // to f(i, j) = f(i - 1, j - w_i) + v_i
5 // tren: f(i, j) = max(f(i - 1, j), f(i - 1, j - w_i) + v_i)
6 int dp[MXN + 1][MXW + 1];
7 memset(dp, 0, sizeof(dp));
8 for (int i = 1; i <= MXN; ++i){
9     for (int j = 0; j < w[i]; ++j){
10         dp[i][j] = dp[i - 1][j];
11     }
12     for (int j = w[i]; j <= MXW; ++j){
13         dp[i][j] = max(dp[i - 1][j - w[i]] + v[i], dp[i - 1][j]);
14     }

```

```

15 }
16 cout << dp[MXN][MXW] << '\n';

```

## 4.8 Infinite Bag

```

1 // f(i, j) = max(f(i - 1, j), f(i - 1, j - wi) + vi,
2 //             f(i, j - wi) + vi)
3 // coin change
4 // 最少幾枚能湊成 M 元
5 // 多少種能湊成 M 元
6 // f(i, j) = f(i - 1, j) + f(i, j - ci)
7 int dp[MXW];
8 memset(dp, -INF, sizeof(dp));
9 dp[0] = 0;
10 for (int i = 0; i < N; ++i){
11     for (int j = w[i]; j <= MXW; ++j){
12         dp[j] = max(dp[j - w[i]] + v[i], dp[j]);
13     }
14 }

```

## 4.9 Tree

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int MXV = 15;
4 vector<int> G[MXV];
5 int high[MXV][2];
6 int ans[MXV], height[MXV];
7
8 void dfs(int u){
9     height[u] = 1;
10    for (int v : G[u]){
11        dfs(v);
12        height[u] = max(height[u], height[v] + 1);
13        if (high[u][0] == 0 || height[high[u][0]] <
14            height[v]){
15            high[u][1] = high[u][0];
16            high[u][0] = v;
17        }
18        else if (high[u][1] == 0 ||
19            height[high[u][1]] < height[v]){
20            high[u][1] = v;
21        }
22    }
23 }
24
25 void dfs2(int u, int legnth){
26     ans[u] = height[high[u][0]] +
27         max(height[high[u][1]], legnth) + 1;
28     for (int v : G[u]){
29         if (v == high[u][0]){
30             dfs2(v, max(height[high[u][1]], legnth) +
31                 1);
32         }
33         else{
34             dfs2(v, max(height[high[u][0]], legnth) +
35                 1);
36         }
37     }
38 }
39
40 int main(){
41     int n;
42     cin >> n;
43     for (int i = 1; i < n; ++i){
44         int x, y;
45         cin >> x >> y;
46         G[x].emplace_back(y);
47     }
48     dfs(1);
49     dfs2(1, 0);

```

```

45 for (int i = 1; i <= n; ++i){
46     cout << ans[i] << '\n';
47 }
48 }

```

## 5 Depth first Search

### 5.1 Anagram Division

```

1 // 給一個字串 s 和一個正整數 d，計算 s
2 // 有幾種排列可以被 d 整除
3 void dfs( int depth, string now ){
4     memset( used, true, sizeof(used) );
5
6     // 算此種排列組合是否可以被整除
7     if( depth == n ){
8
9         digit = 0;
10        for( int i = n - 1; i >= 0; i-- ){
11            digit *= 10;
12            digit += ( now[ i ] - '0' );
13        }
14        if( digit % d == 0 ){
15            quantity++;
16        }
17        return;
18    }
19
20    // 排列組合
21    // 記得用 true/false 確定排過與否
22    for( int i = 0; i < n; i++ ){
23
24        if( flag[i] && used[ str[i] - '0' ] ){
25
26            flag[i] = false;
27            used[ str[i] - '0' ] = false;
28
29            dfs( depth + 1 , now + str[i] );
30            flag[i] = true;
31        }
32    }
33    return;
34 }
35
36 int main(){
37
38     int t;
39     cin >> t;
40
41     while( t-- ){
42
43         memset( flag, true, sizeof(flag) );
44
45         cin >> str >> d;
46         n = str.size();
47         quantity = 0;
48
49         dfs( 0, "" );
50
51         cout << quantity << endl;
52         str.clear();
53     }
54 }

```

### 5.2 Getting in line

```

1 double calculate( int x1, int y1, int x2, int y2 ){
2

```

```

3 // 計算兩點之間的距離
4 // pow 次方 -> pow( 底數, 指數 )
5 // sqrt 開根號 -> sqrt( 數 )
6 return sqrt( pow( ( x1 - x2 ), 2 ) + pow( ( y1 -
    y2 ), 2 ) );
7
8 }
9
10 void dfs( int depth, double path ){
11
12     if( depth == n ){
13         if( path < shortest ){
14
15             shortest = path;
16             final_edge.clear();
17
18             for( int i = 0; i < n; i++ ){
19                 final_edge.push_back( x_now[ i ] );
20                 final_edge.push_back( y_now[ i ] );
21             }
22         }
23         return;
24     }
25
26     // 這次的 dfs 要對每個點做開關 ( true or false )
27     // 在做完一趟後 直接更改 depth - 1 的點後 去對
28     // depth 的點 ( 改變末兩點 )
29     // 第二趟時 跟改 depth - 2 的點後
30     // 先依輸入順序填入後面其他點
31     // 而後下幾輪再繼續排列
32     for( int i = 0; i < n; i++ ){
33
34         if( flag[i] ){
35
36             flag[i] = false;
37
38             x_now[ depth ] = x[i];
39             y_now[ depth ] = y[i];
40
41             if( depth == 0 ){
42                 dfs( depth + 1, 0 );
43             }
44             else{
45                 dfs( depth + 1, path + 16 +
46                     calculate( x_now[ depth ], y_now[
47                         depth ], x_now[ depth - 1 ],
48                         y_now[ depth - 1 ] ) );
49             }
50             flag[i] = true;
51         }
52     }
53 }
54
55 int main(){
56
57     int num = 1;
58
59     while( cin >> n && n ){
60
61         int edge;
62         // 先隨便設個最小值
63         shortest = 2147483647;
64         for( int i = 0; i < n; i++ ){
65
66             cin >> edge;
67             x.push_back(edge);
68
69             cin >> edge;
70             y.push_back(edge);
71
72             flag.insert( pair<int, bool>( i, true ) );
73         }
74
75         dfs( 0, 0 );
76
77         ...

```

```

72 }
73 }

```

### 5.3 Lotto

```

1 void dfs( int depth, int now ){
2
3     // 題目要求每 6 個元素做排列組合
4     if( depth == 6 ){
5
6         for( int i = 0; i < 6; i++ ){
7             if( i ){
8                 cout << " ";
9             }
10            cout << ans[i];
11        }
12        cout << endl;
13
14        // 這個 return 很重要!! 沒有他會 RE
15        return;
16    }
17
18    for( int i = now; i < k; i++ ){
19
20        ans[ depth ] = input[ i ];
21        dfs( depth + 1, i + 1 );
22
23        // 當 depth = 6 後 會回來做這個 for 迴圈
24        // 此時 depth = 5 回到上一次 call dfs 前的深度
25        // 此時 i = i , 但因此時 for 迴圈走向下一迴
26        // i++ 於是 i = i + 1
27        // 然後將 input[i] 的值 覆蓋過 ans[5] 接著
28        // call dfs 去輸出 再 return 回來
29        // 依此類推 當 depth = 5 做完後 會到 depth =
30        // 4 ...
31    }
32 }
33
34 int main(){
35
36     bool flag = false;
37     while( cin >> k && k ){
38
39         if( flag ){
40             cout << endl;
41         }
42
43         int n;
44         for( int i = 0; i < k; i++ ){
45             cin >> n;
46             input.push_back(n);
47         }
48
49         // 從深度為 0 開始往下
50         dfs( 0, 0 );
51
52         flag = true;
53         input.clear();
54     }
55 }

```

## 6 Breadth first Search

### 6.1 Fire

```

1 int step[4][2] = { { 0, -1 }, { 0, 1 }, { -1, 0 }, {
    1, 0 } };
2
3 deque< pair<int,int> > fn;
4 deque< pair<int,int> > joen;
5

```

```

6 void bfs_fire( int n ){
7
8     for( int i = 0; i < 4 ; i++ ){
9
10         int xx = fn[n].first + step[i][0];
11         int yx = fn[n].second + step[i][1];
12
13         if( xx > 0 && xx <= r && yx > 0 && yx <= c ){
14
15             if( mp[ xx ][ yx ] == '.' ){
16                 mp[ xx ][ yx ] = 'F';
17                 fn.push_back( make_pair( xx, yx ) );
18             }
19         }
20     }
21     vis_f++;
22 }
23
24 void bfs_joe( int n ){
25
26     for( int i = 0; i < 4; i++ ){
27
28         int xx = joen[n].first + step[i][0];
29         int yx = joen[n].second + step[i][1];
30
31         if( mp[ xx ][ yx ] == '.' ){
32
33             mp[ xx ][ yx ] = 'J';
34             escape = true;
35             joen.push_back( make_pair( xx, yx ) );
36         }
37         if( mp[ xx ][ yx ] == ' ' ){
38
39             fin = true;
40             break;
41         }
42     }
43     vis_j++;
44 }
45
46 int main(){
47
48     cin >> t;
49     while( t-- ){
50
51         cin >> r >> c;
52         memset( mp, ' ', sizeof(mp) );
53
54         while( !fn.empty() ){
55             fn.pop_front();
56         }
57         while( !joen.empty() ){
58             joen.pop_front();
59         }
60
61         for( int i = 1; i <= r; i++ ){
62             for( int j = 1; j <= c; j++ ){
63
64                 cin >> mp[i][j];
65                 if( mp[i][j] == 'F' ){
66                     fn.push_back( make_pair( i, j ) );
67                 }
68                 if( mp[i][j] == 'J' ){
69                     joen.push_back( make_pair( i, j ) );
70                 }
71             }
72         }
73
74         times = 0;
75         escape = true;
76         fin = false;
77
78         vis_f = 0;
79         vis_j = 0;
80
81         while( escape ){

```

```

82             escape = false;
83             times++;
84             max_f = fn.size();
85             max_j = joen.size();
86
87             for( int i = vis_f; i < max_f; i++ ){
88                 bfs_fire( i );
89             }
90             for( int i = vis_j; i < max_j; i++ ){
91                 bfs_joe( i );
92             }
93
94             if( fin ){
95                 cout << times << endl;
96                 break;
97             }
98         }
99
100         if( !fin ){
101             cout << "IMPOSSIBLE" << endl;
102         }
103         check++;
104     }
105 }
106 }

```

## 6.2 Knights

```

1 int row[8] = { 1, 2, 2, 1, -1, -2, -2, -1 };
2 int column[8] = { 2, 1, -1, -2, -2, -1, 1, 2 };
3
4 int bfs(){
5
6     chess[0][0] = letter_start;
7     chess[0][1] = digit_start;
8     visited[ letter_start ][ digit_start ] = true;
9
10     for( int i = 0, knights = 1; i < knights; i++ ){
11
12         letter_now = chess[i][0];
13         digit_now = chess[i][1];
14
15         if( letter_now == letter_end && digit_now ==
16             digit_end ){
17             return step[ letter_now ][ digit_now ];
18         }
19
20         for( int j = 0; j < 8; j++ ){
21
22             letter_next = letter_now + column[j];
23             digit_next = digit_now + row[j];
24
25             if( letter_next < 1 || digit_next < 1 ||
26                 letter_next > 8 || digit_next > 8 ||
27                 visited[ letter_next ][ digit_next ]
28             ){
29                 continue;
30             }
31             else{
32                 visited[ letter_next ][ digit_next ]
33                     = true;
34                 step[ letter_next ][ digit_next ] =
35                     step[ letter_now ][ digit_now ] +
36                     1;
37
38                 chess[ knights ][0] = letter_next;
39                 chess[ knights ][1] = digit_next;
40
41                 knights++;
42             }
43         }
44     }
45     return -1;
46 }
47
48 int main(){

```

```

42 |
43 | while( cin >> letter1 >> digit_start >> letter2
44 |       >> digit_end ){
45 |
46 |     letter_start = letter1 - 'a' + 1;
47 |     letter_end = letter2 - 'a' + 1;
48 |
49 |     for( int i = 0; i < 10; i++ ){
50 |         for( int j = 0; j < 10; j++ ){
51 |             step[i][j] = 0;
52 |             visited[i][j] = false;
53 |         }
54 |     }
55 |     cout << bfs() << " knight moves." << endl;
56 | }

```

### 6.3 Oil Deposits

```

1 | int row[] = { 1, 1, 1, 0, -1, -1, -1, 0 };
2 | int column[] = { -1, 0, 1, 1, 1, 0, -1, -1 };
3 |
4 | void bfs( int x_now, int y_now ){
5 |
6 |     for( int j = 0; j < 8; j++ ){
7 |
8 |         x_next = x_now + row[j];
9 |         y_next = y_now + column[j];
10 |
11 |         if( x_next < m && y_next < n && x_next >= 0
12 |             && y_next >= 0 && oil[ x_next ][ y_next ]
13 |             == '@' ){
14 |
15 |             // 此點已找過 就把他改成普通地板
16 |             oil[ x_next ][ y_next ] = '*';
17 |             bfs( x_next, y_next );
18 |         }
19 |     }
20 |     return;
21 | }
22 |
23 | int main(){
24 |
25 |     while( cin >> m >> n && m && n ){
26 |
27 |         memset( oil, '0', sizeof(oil) );
28 |         ans = 0;
29 |
30 |         for( int i = 0; i < m; i++ ){
31 |             for( int j = 0; j < n; j++ ){
32 |                 cin >> oil[i][j];
33 |             }
34 |         }
35 |
36 |         for( int i = 0; i < m; i++ ){
37 |             for( int j = 0; j < n; j++ ){
38 |
39 |                 if( oil[i][j] == '@' ){
40 |                     ans++;
41 |                     bfs( i, j );
42 |                 }
43 |             }
44 |         }
45 |         cout << ans << endl;
46 |     }

```

### 6.4 Rat Attack

```

1 | void bfs( int xn, int yn ){
2 |
3 |     int xx, yx;

```

```

4 |
5 | // 從 -d 到 d 之間的所有格子
6 | // 因為起始點是中心，所以要用 -d ~ d 的方式算點
7 | for( int i = 0 - d; i <= d; i++ ){
8 |     for( int k = 0 - d; k <= d; k++ ){
9 |
10 |         xx = xn + i;
11 |         yx = yn + k;
12 |
13 |         if( xx >= 0 && xx < 1025 && yx >= 0 && yx
14 |             < 1025 ){
15 |
16 |             maxi[ xx ][ yx ] += rat[ xn ][ yn ];
17 |
18 |             if( maxi[ xx ][ yx ] > maxm[2] ){
19 |                 maxm[0] = xx;
20 |                 maxm[1] = yx;
21 |                 maxm[2] = maxi[ xx ][ yx ];
22 |             }
23 |         }
24 |     }
25 | }
26 |
27 | int main(){
28 |
29 |     int t;
30 |     cin >> t;
31 |
32 |     while( t-- ){
33 |
34 |         memset( rat, 0, sizeof(rat) );
35 |         memset( maxi, 0, sizeof(maxi) );
36 |         memset( check, 0, sizeof(check) );
37 |         memset( maxm, 0, sizeof(maxm) );
38 |
39 |         cin >> d >> n;
40 |         int num = 0;
41 |
42 |         for( int i = 0; i < n; i++ ){
43 |
44 |             int a, b;
45 |             cin >> a >> b;
46 |             cin >> rat[a][b];
47 |
48 |             check[num][0] = a;
49 |             check[num][1] = b;
50 |             num++;
51 |
52 |         }
53 |
54 |         for( int i = 0; i < num; i++ ){
55 |             bfs( check[i][0], check[i][1] );
56 |         }
57 |
58 |         for( int i = 0; i < 3; i++ ){
59 |
60 |             if( i != 2 ){
61 |                 cout << maxm[i] << " ";
62 |             }
63 |             else{
64 |                 cout << maxm[i] << endl;
65 |             }
66 |         }
67 |     }
68 | }

```

## 7 MATH

### 7.1 Fraction

```

1 | #include<iostream>
2 | using namespace std;
3 |

```



```

4 // 1/k = 1/x + 1/y
5 // 給你 k，請你寫一個程式找出所有的 x 和 y
6 int main(){
7
8     int n;
9     while(cin>>n){
10
11         int i;
12         int N[10000+5][2]={0};
13         int flag=0;
14
15         for(i=n+1; i<= 2*n; i++){
16
17             int r = i-n;
18
19             if((n*i)% r ==0){
20
21                 N[flag][0]= (n*i)/r;
22                 N[flag][1]= i;
23                 flag++;
24
25             }
26         }
27
28         cout<< flag << endl;
29         for(i=0; i<flag; i++){
30             cout<< "1/" << n << " = 1/" << N[i][0] <<
31                 " + 1/" << N[i][1] << endl;
32         }
33         return 0;
34 }

```

## 7.2 Slope

```

1 // 八皇后 上下左右斜行皆不重複
2 int check( int x, int y ){
3
4     for( int i = 0; i < x; i++ ){
5
6         if( dq[1][i] == y ){
7             return 0;
8         }
9
10        // 如果兩皇后在同一斜線上 其斜率為 1
11        // 如果 x2 - x1 == y2 - y1 -> y2 - y1 / x2 -
12        // x1 == 1
13        if( abs( x - i ) == abs( dq[1][i] - y ) ){
14            return 0;
15        }
16    }
17    return 1;
18 }

```

## 7.3 GCD

```

1 // 如果兩數互質 最終結果其中一方為0時 另一方必為1
2 // 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
3 while ( ( num1 % num2 ) != 0 && ( num2 % num1 )
4         != 0 );

```

## 8 Others

### 8.1 Enumerate Twopointers

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 //2021.09.14
4

```

```

5 int main(){
6
7     long long int t;
8     cin >> t;
9     while( t-- ){
10
11         long long int n;
12         deque<int> snowflakes;
13         set<long long int> difference;
14
15         cin >> n;
16         for( int i = 0; i < n; i++ ){
17
18             int m;
19             cin >> m;
20             snowflakes.push_back(m);
21
22         }
23         int longest = 0;
24
25         // 利用 L左指標 和 R右指標
26         // 每次迭代右指標先往前一個位置
27         for( int L = 0, R = 0; R < n; R++ ){
28
29             // 利用 set.count 先確認 set
30             // 內是否有重複元素
31             while( difference.count( snowflakes[R] )
32                 ){
33
34                 // 如果有 利用 set.erase 和左指標
35                 // 將與 右指標重複的元素
36                 // 以前的所有元素刪除
37                 difference.erase( snowflakes[ L++ ] );
38
39             }
40
41             difference.insert( snowflakes[R] );
42
43             // std::max 可比較兩者之間誰大
44             // max( 比較方1, 比較方2 )
45             // 比較方可以不一定是 int
46             // 但一定要相同型態
47             longest = max( longest,
48                 (int)difference.size() );
49         }
50
51         cout << longest << endl;
52         difference.clear();
53         snowflakes.clear();
54     }
55 }

```

## 8.2 Physics

```

1 int main(){
2
3     // s = vot + 1/2 at^2
4     // v = vo + at
5     // a = ( v - vo ) / t
6
7     // vo = 0
8     // a = v / t
9     // s = 0 + 1/2 v/t 2t^2 = 1/2 v 4t = 2vt
10
11     int v,t;
12     while( cin >> v >> t ){
13         int s;
14         s = 2 * v * t ;
15         cout << s << endl;
16     }
17 }

```

### 8.3 Week

```

1 int main(){
2
3     int month[] = { 31, 28, 31, 30, 31, 30, 31, 31,
4                     30, 31, 30, 31 };
5     string week[] = { "Monday", "Tuesday",
6                       "Wednesday", "Thursday", "Friday",
7                       "Saturday", "Sunday" };
8
9     int n;
10    cin >> n;
11    while( n-- ){
12
13        int m, d;
14        cin >> m >> d;
15        int w = 4;
16
17        for( int i = 0; i < m - 1; i++ )
18            w += month[i];
19
20        cout << week[ ( w + d ) % 7 ] << endl;
21    }
22 }
```

### 8.4 Carry Change

```

1 // 題目給定一個 N 進制 ( 2 <= N <= 62 ) 的數字 R, R
2 // 保證可以被 (N-1) 整除
3 // 求符合提議的最小 N
4 // 當 N = 62 時, 用來表示 62 進制的字符為 0..9, A..Z,
5 // a..z。
6
7 int main(){
8     // R = 265
9     // = 2*N*N + 6*N + 5
10    // = 2*N*(N-1+1) + 6*N + 5
11    // = 2*N*(N-1) + 2*(N-1+1) + 6*(N-1+1) + 5
12    // = 2*N*(N-1) + 2*(N-1) + 2 + 6*(N-1) + 6 + 5
13    // = (2*N + 2 + 6)*(N-1) + (2 + 6 + 5)
14    // because R % N-1 == 0
15    // so (2+6+5) == N-1
16
17    string str;
18    while( getline( cin, str ) ){
19
20        int tmp;
21        int max = 1, sum = 0;
22        bool flag = true;
23
24        for( int i = 0; i < str.size(); i++ ){
25
26            if( str[i] >= '0' && str[i] <= '9' ){
27                tmp = str[i] - '0';
28            }
29            else if( str[i] >= 'A' && str[i] <= 'Z' ){
30                tmp = str[i] - 'A' + 10;
31            }
32            else if( str[i] >= 'a' && str[i] <= 'z' ){
33                tmp = str[i] - 'a' + 10 + 26;
34            }
35            else{
36                continue;
37            }
38
39            if( tmp > max ){
40                max = tmp;
41            }
42
43            sum += tmp;
44        }
45
46        for( int i = max; i < 62; i++ ){
47            if( !( sum % i ) ){
48                cout << i + 1 << endl;
49                flag = false;
50            }
51        }
52    }
53 }
```

```

47         break;
48     }
49 }
50 if( flag ){
51     cout << "such number is impossible!" <<
52     endl;
53 }
54 str.clear();
55 }
```

### 8.5 Recursive

```

1 int gcd( int i, int j ){
2
3     while( ( j %= i ) != 0 && ( i %= j ) != 0 );
4     return j + i;
5 }
6
7 int g( int n ){
8
9     // 已使用過此數字 直接從陣列中呼叫
10    if( known[n] ){
11        return known[n];
12    }
13
14    else{
15
16        // 利用 g( n - 1 )
17        // 去確認此次輪迴為尚不知道結果的最大數字
18        known[n] += g( n - 1 );
19
20        // 計算本次結果 同時將本次結果儲存於陣列中
21        for( int i = 1; i < n; i++ ){
22            known[n] += gcd( i, n );
23        }
24        return known[n];
25    }
26 }
27
28 int main(){
29
30     known[2] = 1;
31     int n;
32
33     while( cin >> n ){
34
35         if( n == 0 ){
36             break;
37         }
38
39         cout << g(n) << endl;
40
41         // 題目方法
42         // for( int i = 1; i < n; i++ ){
43         //     for( int j = i + 1; j <= n; j++ ){
44         //         g += gcd( i, j );
45         //     }
46         // }
47     }
48 }
```

### 8.6 Prime List

```

1 bool prime[1000000];
2
3 // memset: 對一段內存空間全部設置為某個字符
4 // 常用於初始化字串、陣列..
5 // memset( 陣列名稱, 初始化成甚麼, 範圍 )
6 memset( prime, false, sizeof(prime) );
7 memset( prime, true, 2);
```

```

8   for( int i = 2; i < 1000000; i++ ){
9
10      if( !prime[i] ){
11
12         for( int j = i + i; j < 1000000; j += i ){
13
14            prime[j] = true;
15
16         }
17      }
18 }

```

## 8.7 Probability

```

1 int main(){
2
3     int s, n, i;
4     double p, p2, ans;
5     cin >> s;
6
7     while( s-- ){
8
9         cin >> n >> p >> i;
10        p2 = pow( 1.0 - p , n );
11
12        if( p2 == 1){
13
14            cout << "0.0000" << endl;
15            continue;
16
17        }
18
19        //第i個人成功的機率 /
20        //全部的人有機會成功的機率(1-全部都失敗)
21        ans = p * pow( 1.0 - p , i-1 ) / ( 1.0 - pow(
22            1.0 - p , n ) );
23        cout << fixed << setprecision(4) << ans
24            <<endl;
25    }
26 }

```

## 8.8 distance

```

1 double calculate( int x1, int y1, int x2, int y2 ){
2
3     // 計算兩點之間的距離
4     // pow 次方 -> pow( 底數, 指數 )
5     // sqrt 開根號 -> sqrt( 數 )
6     return sqrt( pow( ( x1 - x2 ) , 2 ) + pow( ( y1 -
7         y2 ) , 2 ) );
8 }

```

## 8.9 floor

```

1 int main(){
2
3     int a, b;
4     while( cin >> a >> b && ( a || b ) ){
5
6         int aa, bb, check = 0;
7         if( floor( sqrt(a) ) == sqrt(a) ){
8             check = 1;
9         }
10        double count = 0;
11
12        //floor -> 不大於 x 的最大整數 ( 浮點型 )
13        count = floor( sqrt(b) ) - floor( sqrt(a) ) +
14            check;
15        cout << (int)count << endl;

```

## 8.10 map

```

1 int main(){
2
3     int n;
4     string country, name;
5
6     map<string, int> m;
7     map<string, int>::iterator it;
8
9     cin>>n;
10    while( n-- ){
11
12        cin >> country ;
13        getline(cin, name);
14        it = m.find( country );
15
16        if( it != m.end() ){
17            m[ country ]++;
18        }
19        else{
20            m.insert( pair<string, int>( country, 1)
21                );
22        }
23    }
24    for( auto i = m.begin(); i != m.end(); i++ ){
25        cout << i->first << " " << i->second <<endl;
26    }
27 }

```

## 8.11 set intersection

```

1 int main(){
2
3     while(getline(cin, str1) && getline(cin, str2)){
4
5         sort(str1.begin(), str1.end());
6         sort(str2.begin(), str2.end());
7
8         deque<char> dq;
9         dq.clear();
10
11        // set_intersection 在 C++ 中查詢集合交集
12        // ex str1 = {1,2,3,4,5,6,7,8}, str2 =
13        // {5,7,9,10}
14        // output = 5 7
15        // set_intersection( 字串1頭, 字串1尾,
16        // 字串2頭, 字串2尾, 比較完要放進的地方 )
17        set_intersection(str1.begin(), str1.end(),
18            str2.begin(), str2.end(),
19            insert_iterator<deque<char>>(dq, dq.begin()));
20
21        for( int i=0; i<dq.size(); i++){
22            cout<<dq[i];
23        }
24        cout << endl;
25    }
26 }

```

## 8.12 setprecision

```

1 double x = 10.19395;
2
3 // 總共輸出三位數
4 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
```