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

```
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

```
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

```
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

```
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

```
// 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

```
// init: f(i, 0) = f(i, i) = 1
// tren: f(i, j) = f(i - 1, j) + f(i - 1, j - 1)
int main(){
    int dp[30][30];
    memset(dp, 0, sizeof(dp));
    for (int i = 0; i < 30; ++i){
        dp[i][0] = dp[i][i] = 1;
    }
    for (int i = 1; i < 30; ++i){
        for (int j = 1; j < 30; ++j){
            dp[i][j] = dp[i - 1][j] + dp[i - 1][j -
                1];
        }
    }
}
```

4.3 Robot

```
// 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){
```

```
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,
         f(i, j - wi) + vi)
2      // coin chage
3          // 最少幾枚能湊成 M 元
4              //
                   f(i,j)=min(f(i-1,j),f(i-1,j-ci)+1,f(i,j-ci)+1)
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]] <
               height[v]){
14             high[u][1] = high[u][0];
15             high[u][0] = v;
16         }
17         else if (high[u][1] == 0 ||
               height[high[u][1]] < height[v]){
18             high[u][1] = v;
19         }
20     }
21 }
22
23 void dfs2(int u, int legnth){
24     ans[u] = height[high[u][0]] +
               max(height[high[u][1]], legnth) + 1;
25     for (int v : G[u]){
26         if (v == high[u][0]){
27             dfs2(v, max(height[high[u][1]], legnth) +
                   1);
28         }
29         else{
30             dfs2(v, max(height[high[u][0]], legnth) +
                   1);
31         }
32     }
33 }
34
35 int main(){
36     int n;
37     cin >> n;
38     for (int i = 1; i < n; ++i){
39         int x, y;
40         cin >> x >> y;
41         G[x].emplace_back(y);
42     }
43     dfs(1);
```

```
44     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
       有幾種排列可以被 d 整除
2  void dfs( int depth, string now ){
3
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     }
34     return;
35 }
36
37
38
39 int main(){
40
41     int t;
42     cin >> t;
43
44     while( t-- ){
45
46         memset( flag, true, sizeof(flag) );
47
48         cin >> str >> d;
49         n = str.size();
50         quantity = 0;
51
52         dfs( 0, "" );
53
54         cout << quantity << endl;
55         str.clear();
56     }
57 }
```

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
          depth 的點 ( 改變末兩點 )
28     // 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
29     for( int i = 0; i < n; i++ ){
30
31         if( flag[i] ){
32
33             flag[i] = false;
34
35             x_now[ depth ] = x[i];
36             y_now[ depth ] = y[i];
37
38             if( depth == 0 ){
39                 dfs( depth + 1, 0 );
40             }
41             else{
42                 dfs( depth + 1, path + 16 +
                      calculate( x_now[ depth ], y_now[
                      depth ], x_now[ depth - 1 ],
                      y_now[ depth - 1 ] ) );
43             }
44             flag[i] = true;
45         }
46     }
47 }
48
49 int main(){
50
51     int num = 1;
52
53     while( cin >> n && n ){
54
55         int edge;
56         // 先隨便設個最小值
57         shortest = 2147483647;
58         for( int i = 0; i < n; i++ ){
59
60             cin >> edge;
61             x.push_back(edge);
62
63             cin >> edge;
64             y.push_back(edge);
65
66             flag.insert( pair<int, bool>( i, true ) );
67         }
68
69         dfs( 0, 0 );
```

```
70
71              ...
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
              i++ 於是 i = i + 1
26         // 然後將 input[i] 的值 覆蓋過 ans[5] 接著
              call dfs 去輸出 再 return 回來
27         // 依此類推 當 depth = 5 做完後 會到 depth =
              4 ...
28     }
29 }
30
31 int main(){
32
33     bool flag = false;
34     while( cin >> k && k ){
35
36         if( flag ){
37             cout << endl;
38         }
39
40         int n;
41         for( int i = 0; i < k; i++ ){
42             cin >> n;
43             input.push_back(n);
44         }
45
46         // 從深度為 0 開始往下
47         dfs( 0, 0 );
48
49         flag = true;
50         input.clear();
51     }
52 }
```

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;
```

```cpp
 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
75         times = 0;
76         escape = true;
77         fin = false;
78
79         vis_f = 0;
80         vis_j = 0;
```

```cpp
80
81         while( escape ){
82
83             escape = false;
84             times++;
85             max_f = fn.size();
86             max_j = joen.size();
87
88             for( int i = vis_f; i < max_f; i++ ){
89                 bfs_fire( i );
90             }
91             for( int i = vis_j; i < max_j; i++ ){
92                 bfs_joe( i );
93             }
94
95             if( fin ){
96                 cout << times << endl;
97                 break;
98             }
99         }
100
101         if( !fin ){
102             cout << "IMPOSSIBLE" << endl;
103         }
104         check++;
105     }
106 }
```

6.2 Knights

```cpp
 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 ==
                digit_end ){
16             return step[ letter_now ][ digit_now ];
17         }
18
19         for( int j = 0; j < 8; j++ ){
20
21             letter_next = letter_now + column[j];
22             digit_next = digit_now + row[j];
23
24             if( letter_next < 1 || digit_next < 1 ||
                    letter_next > 8 || digit_next > 8 ||
                    visited[ letter_next ][ digit_next ]
                    ){
25                 continue;
26             }
27             else{
28                 visited[ letter_next ][ digit_next ]
                        = true;
29                 step[ letter_next ][ digit_next ] =
                        step[ letter_now ][ digit_now ] +
                        1;
30
31                 chess[ knights ][0] = letter_next;
32                 chess[ knights ][1] = digit_next;
33
34                 knights++;
35             }
36         }
37     }
38     return -1;
39 }
```

```
40
41  int main(){
42
43      while( cin >> letter1 >> digit_start >> letter2
            >> digit_end ){
44
45          letter_start = letter1 - 'a' + 1;
46          letter_end = letter2 - 'a' + 1;
47
48          for( int i = 0; i < 10; i++ ){
49              for( int j = 0; j < 10; j++ ){
50                  step[i][j] = 0;
51                  visited[i][j] = false;
52              }
53          }
54          cout << bfs() << " knight moves." << endl;
55      }
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
              && y_next >= 0 && oil[ x_next ][ y_next ]
              == '@' ){
12
13              // 此點已找過 就把他改成普通地板
14              oil[ x_next ][ y_next ] = '*';
15              bfs( x_next, y_next );
16          }
17      }
18      return;
19  }
20
21  int main(){
22
23      while( cin >> m >> n && m && n ){
24
25          memset( oil, '0', sizeof(oil) );
26          ans = 0;
27
28          for( int i = 0; i < m; i++ ){
29              for( int j = 0; j < n; j++ ){
30                  cin >> oil[i][j];
31              }
32          }
33
34          for( int i = 0; i < m; i++ ){
35              for( int j = 0; j < n; j++ ){
36
37                  if( oil[i][j] == '@' ){
38                      ans++;
39                      bfs( i, j );
40                  }
41
42              }
43          }
44          cout << ans << endl;
45      }
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
                  < 1025 ){
14
15                  maxi[ xx ][ yx ] += rat[ xn ][ yn ];
16
17                  if( maxi[ xx ][ yx ] > maxm[2] ){
18                      maxm[0] = xx;
19                      maxm[1] = yx;
20                      maxm[2] = maxi[ xx ][ yx ];
21                  }
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>
```

```cpp
using namespace std;

// 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;
    }
```

```
17 | }
```

8.3 Week

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

8.4 Carry Change

```
 1 | // 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
 2 | // 求符合提議的最小 N
 3 | // 當 N = 62 時，用來表示62進制的字符為 0..9, A..Z,
   |     a..z。
 4 |
 5 | int main(){
 6 |
 7 |     // R = 265
 8 |     // = 2*N*N + 6*N + 5
 9 |     // = 2*N*(N-1+1) + 6*N + 5
10 |     // = 2*N*(N-1) + 2*(N-1+1) + 6*(N-1+1) + 5
11 |     // = 2*N*(N-1) + 2*(N-1) + 2 + 6*(N-1) + 6 + 5
12 |     // = (2*N + 2 + 6)*(N-1) + (2 + 6 + 5)
13 |     // because R % N-1 == 0
14 |     // so (2+6+5) == N-1
15 |
16 |     string str;
17 |     while( getline( cin, str ) ){
18 |
19 |         int tmp;
20 |         int max = 1, sum = 0;
21 |         bool flag = true;
22 |
23 |         for( int i = 0; i < str.size(); i++ ){
24 |
25 |             if( str[i] >= '0' && str[i] <= '9' ){
26 |                 tmp = str[i] - '0';
27 |             }
28 |             else if( str[i] >= 'A' && str[i] <= 'Z' ){
29 |                 tmp = str[i] - 'A' + 10;
30 |             }
31 |             else if( str[i] >= 'a' && str[i] <= 'z' ){
32 |                 tmp = str[i] - 'a' + 10 + 26;
33 |             }
34 |             else{
35 |                 continue;
36 |             }
37 |
38 |             if( tmp > max){
39 |                 max = tmp;
40 |             }
41 |             sum += tmp;
42 |         }
```

```
43 |         for( int i = max; i < 62; i++ ){
44 |             if( !( sum % i ) ){
45 |                 cout << i + 1 << endl;
46 |                 flag = false;
47 |                 break;
48 |             }
49 |         }
50 |         if( flag ){
51 |             cout << "such number is impossible!" <<
   |                 endl;
52 |         }
53 |         str.clear();
54 |     }
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 |
 8 | int g( int n ){
 9 |
10 |     // 已使用過此數字 直接從陣列中呼叫
11 |     if( known[n] ){
12 |         return known[n];
13 |     }
14 |
15 |     else{
16 |
17 |         // 利用 g( n - 1)
   |             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
   |     常用於初始化字串、陣列..
```

```
4  // memset( 陣列名稱，初始化成甚麼，範圍 )
5      memset( prime, false, sizeof(prime) );
6      memset( prime, true, 2);
7
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 /
               全部的人有機會成功的機率(1-全部人都失敗)
20         ans = p * pow( 1.0 - p , i-1 ) / ( 1.0 - pow(
               1.0 - p , n ) );
21         cout << fixed << setprecision(4) << ans
               <<endl;
22     }
23 }
```

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 -
               y2 ) , 2 ) );
7
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) ) +
               check;
14
15         cout << (int)count << endl;
16     }
17 }
```

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     for( auto i = m.begin(); i != m.end(); i++ ){
24         cout << i->first << " " << i->second <<endl;
25     }
26 }
```

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 =
               {5,7,9,10}
13         // output = 5 7
14         // 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 )
15         set_intersection(str1.begin(), str1.end(),
               str2.begin(), str2.end(),
               insert_iterator<deque<char>>(dq,dq.begin()));
16
17         for( int i=0; i<dq.size(); i++){
18
19             cout<<dq[i];
20
21         }
22         cout << endl;
23
24     }
25 }
```

8.12 setprecision

```cpp
double x = 10.19395;

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

    // 輸出小數點後三位數
    cout << fixed << setprecision(3) << x;

// output:
// 10.2
// 10.194
// 都會自動四捨五入進位
```