Cách di chuyển:

|  |  |  |
| --- | --- | --- |
|  | **↑** |  |
| **←** |  | **→** |
|  | **↓** |  |

Giả sử từ ô [1, 1]:

→ [1, 2]

→ [1, 0]

→ [0, 1]

→ [2, 1]

Giả sử từ ô [r, c]:

→ [r, c+1]

→ [r, c-1]

→ [r-1, c]

→ [r+1, c]

**dr = [0, 0, -1, 1]**

**dc = [1, -1, 0, 0]**

Điều kiện để valid:

- Có đúng 2 điểm mở, điểm mở là các ô mang giá trị ‘.’ ở rìa của ma trận

- Có đường đi giữa 2 điểm mở đó

Nếu coi mỗi ô trong ma trận là một đỉnh của đồ thị, thì chúng ta chỉ cần BFS từ điểm bắt đầu để tìm xem có đường đi tới đỉnh kết thúc hay không.

**Ví dụ**

6

4 4

####

#...

#.##

#.##

Điểm mở: [1, 3], [3, 1]

Có thể đi theo đường: [1, 3] → [1, 2] → [1, 1] → [2, 1] → [3, 1]

→ **valid**

5 5

#.###

#..##

##..#

#.#.#

###.#

Điểm mở: [0, 1], [4, 3]

Có thể đi: [0, 1] → [1, 1] → [1, 2] → [2, 2] → [2, 3] → [3, 3] → [4, 3]

→ **valid**

1 1

.

Điểm mở: [0, 0]

→ **invalid**

5 1

#

#

.

.

#

Điểm mở: [2, 0], [3, 0]

Đường đi: [2, 0] → [3, 0] hoặc là [3, 0] → [2, 0]

→ **valid**

2 2

#.

.#

Điểm mở: [0, 1], [1, 0]

Đường đi: Không có đường đi giữa 2 điểm mở này

→ **invalid**

3 4

#..#

#.##

#.##

Điểm mở: [0, 1], [0, 2], [2, 1]

→ **invalid**

**Giải thuật:**

**Bước 1:** Nhập dữ liệu.

**Bước 2:** Tìm lối vào và lối ra cho mê cung.

- Nếu có đúng 1 lối vào và 1 lối ra, có đúng 2 điểm mở trên ma trận, tiếp tục bước số 3.

- Ngược lại, trả về “**invalid**”.

**Bước 3:** BFS từ lối vào.

**Bước 4:** Nếu có đường đi từ lối vào tới lối ra, trả về “**valid**”, ngược lại, trả về “**invalid**”.

**Mã giả:**

**dr = [0, 0, -1, 1]**

**dc = [1, -1, 0, 0]**

**function BFS(s, f):**

**queue = [];**

**queue.push(s);**

**while queue.size() > 0:**

**ur, uc = queue.front();**

**queue.pop();**

**if (ur, uc) == f:**

**return true;**

**for i = 0 to 3:**

**r = ur + dr[i];**

**c = uc + dc[i];**

**if r < 0 or r >= m or c < 0 or c >= n:**

**continue;**

**if maze[r][c] == ‘#’ or visited[r][c] == true:**

**continue;**

**visited[r][c] = true;**

**queue.push((r, c));**

**return false;**

**read(T);**

**for testcase = 1 to T:**

**read(m, n);**

**for i = 0 to m-1:**

**read(maze[i]);**

**entrance = [];**

**for i = 0 to m-1:**

**for j = 0 to n-1:**

**visited[i][j] = false;**

**if maze[i][j] == ‘.’**

**and (i == 0 or i == m-1 or j == 0 or j == n-1):**

**entrance.push((i, j));**

**if entrance.size() != 2:**

**print(‘invalid’);**

**else:**

**if BFS(entrance[0], entrance[1]) == true:**

**print(‘valid’);**

**else:**

**print(‘invalid’);**

**Độ phức tạp: O(T \* (V + E)) = O(T \* M \* N)**

**T** là số lượng testcase

**M** là số lượng hàng trong ma trận

**N** là số lượng cột trong ma trận