Guilty Prince

Link submit: http://lightoj.com/volume_showproblem.php?problem=1012

Solution:

C++	http://ideone.com/xiNVED
Java	https://ideone.com/jn1TbC
Python	https://ideone.com/iSewBz

Tóm tắt đề:

Cho một bảng hình chữ nhật kích thước W x H, các ký tự trên bảng gồm ba loại ký tự thể hiện như sau:

- Loại '.' : Đất
- Loại '#': Nước
- Loại '@': Vị trí hiện tại đang đứng.

Yêu cầu: đếm xem có bao nhiêu ô '.' (bao gồm cả ô có chứa ký tự '@') có đường đi tới ô '@'. Hai ô có thể đi được với nhau nếu chúng có chung cạnh và hai ô đều là đất (ô '@' ban đầu cũng là một ô đất).

Input:

Dòng đầu tiên chứa một số nguyên T thể hiện số bộ test của đề bài.

T bộ dữ liệu đầu vào sau, mỗi bộ dữ liệu được tổ chức như sau:

- Dòng đầu chứa hai số nguyên dương W H là chiều rộng và chiều dài của bảng.
- W dòng sau, mỗi dòng chứa H ký tự chỉ bao gồm 3 loại '.' , '#' và '@'.

Output:

Gồm T dòng, mỗi dòng in theo dạng "Case i: ans", trong đó:

- i là thứ tự của bộ test, bắt đầu từ 1.
- ans là kết quả của bộ dữ liệu tương ứng.

Ví dụ:

4 6 9 ....#. .....# .... #@...# .#..#. 11 9 .#......... .#.#######. .#.#.....#. .#.#.###.#. .#.#..@#.#. .#.#####.#. .#.......#. .#########. .... 11 6 ..#..#..#.. ..#..#..#.. ..#..#..### ..#..#..#@. ..#..#..#.. ..#..#..#.. 7 7 ..#.#.. ..#.#.. ###.### ...@... ###.### ..#.#.. ..#.#..	Case 1: 45 Case 2: 59 Case 3: 6 Case 4: 13

Hướng dẫn giải:

Bạn có thể thấy rằng hai ô (x, y) và ô (i, j) sẽ đi được với nhau nếu chúng có chung cạnh. Do đó, nếu bạn có thể xem mỗi ô trên bảng là một đỉnh của một đồ thị, thì hai đỉnh trên đồ thị sẽ có một cạnh nối với nhau nếu chúng thỏa mãn ba tính chất sau đây:

- Hai ô được biểu diễn bởi hai đỉnh của đồ thị phải nằm trong bảng đầu vào.
- Hai ô, không ô nào được phép biểu diễn bởi ký tự '#'.
- Hai ô phải chung cạnh với nhau.

Như vậy, sau khi ta xây dựng được một đồ thị vô hướng, cách đơn giản để giải quyết bài này là:

- Ta đi xác định vị trí của ô '@', giả sử là ô (sx, sy).
- Sau đó, với mỗi ô (i, j) trong bảng được biểu diễn bằng dấu '.', ta sẽ sử dụng kỹ thuật duyệt DFS từ ô (i, j) để xác định xem có thể đến được ô (sx, sy) hay không. Nếu có, ta tăng biến kết quả ans lên 1.
- Kết quả là ans + 1 (Tính cả ô (sx, sy)).

Phải xét mỗi ô (i, j) có đến được ô (sx, sy) hay không như vậy khá tốn thời gian, ta có một cách khác để cải tiến thuật toán ở trên:

Dựa vào một nhận xét tự nhiên, ta thấy mọi ô giả sử có đường đi đến ô (sx, sy) thì chúng đều sẽ tập trung lại tại ô (sx , sy). Do đó, nếu như ta nhìn nhận bài toán ở một khía cạnh khác, chúng ta sẽ phát hiện rằng: thay vì với mỗi ô (i , j), ta kiểm tra xem có đường đi tới (sx, sy), thì bây giờ ta sẽ xuất phát từ ô (sx , sy), đi loang ra các đỉnh khác, mỗi lần ta tiếp cận được với một đỉnh mà chưa được xét thì ta sẽ tăng biến ans lên 1 đơn vị. Kết quả là ans.

Độ phức tạp:

Cách chưa cải tiến

- ***Time Complexity:*** O(T * W * W * H * H)
- ***Space Complexity:*** O(W * H)
- W và H không quá 20 nên độ phức tạp O(T * W * W * H * H) vẫn có thể chấp nhận được.

Cách cải tiến

- ***Time Complexity:*** **O(T * W * H)** với W và H lần lượt là chiều rộng và chiều cao của bảng, còn T là số lượng bộ test đầu vào.
- ***Space Complexity:*** **O(W * H)**