Lakes In Berland

Link submit: http://codeforces.com/problemset/problem/723/D

Solution:

C++	https://ideone.com/kZNwna
Java	https://ideone.com/lmuTVo
Python	https://ideone.com/hJkFvC

Tóm tắt đề: Có một bản đồ kích thước n * m với n là số lượng hàng ngang còn m là số lượng hàng dọc. Ô trên dòng i , cột j được gọi là ô (i j). Mỗi ô (i j) sẽ được quy định bằng dấu '*' hoặc '.', trong đó '*' được quy định là đất, còn '.' là nước. Bao bọc xung quanh bản đồ này là biển. Một vùng được gọi là hồ nếu như thỏa mãn 2 điều kiện sau:

- Vùng này toàn là nước, 2 ô là nước sẽ được tính là kề nhau nếu như chúng chung cạnh.
- Vùng này không giáp biển. (Lưu ý biển có thể từ biên đi vào trong theo đường nước '.')

Bạn được yêu cầu lấp lại các hồ này sao cho tổng số lượng ô nước cần lấp là ít nhất, và phải còn lại đúng k hồ.

Input

- Dòng đầu gồm 3 số nguyên dương n, m, k cách nhau bởi dấu cách, đại diện cho số lượng hàng ngang, số lượng hàng dọc và số lượng hồ cần thiết còn lại sau khi lấp ($1 \le n, m \le 50$, $0 \le k \le 50$)
- n dòng sau, mỗi dòng gồm m ký tự. Ký tự thứ j trên dòng thứ i thể hiện ô (i j) là nước hay đất. Nếu ô (i, j) là '*' thì ô (i j) là đất, ngược lại là nước nếu như là '.'

Output

- Dòng đầu tiên là một số nguyên là tổng số lượng ô nước nhỏ nhất cần được lấp đầy.
- n dòng sau, mỗi dòng gồm m ký tự là trạng thái của bản đồ sau khi lấp đầy nước một cách tối ưu.

Ví dụ

5 4 1 **** *..* **** **.*	1 **** *..* **** ****

. . * *	. . * *

Giải thích: Ta sẽ lấp một ô nước ở vị trí (4, 3) và bản đồ sẽ ra như hình bên.

Hướng dẫn giải:

Trước tiên, ta cần chuẩn bị thuật DFS mà có cải tiến là ta sẽ tiến hành lấp nước trực tiếp trên hàm DFS luôn. Như vậy, khi ta đang xét tại một đỉnh (x y), ta sẽ tiến hành gán trực tiếp ô (x y) thành dấu '*'. Cách viết hàm DFS như thế nào thì bạn coi trong solution.

Ý tưởng: Vì ta chỉ quan tâm đến các hồ nước, nên trước hết, ta sẽ lấp các vũng nước kề với biển trước cho đỡ "phiền phức". Tức là ta sẽ duyệt những ô trên cạnh của bảng, nếu như ô đó mà là nước, ta sẽ tiến hành DFS từ ô đó.

Sau đó, ta duyệt lại toàn bộ bảng, nếu như gặp ô (i j) là nước thì ta sẽ tiến hành lấp nước bắt đầu từ ô đó, đồng thời sẽ lưu lại số ô nước mà có thể đi được từ ô (i j). Ta sẽ đưa vào một vector tọa độ của ô nước đó và số lượng ô nước có thể lấp được nếu bắt đầu đi từ ô đó.

Ta tiến hành sắp xếp lại vector đó theo chiều tăng dần của số lượng ô nước.

Vì đề yêu cầu ta phải lấp sao cho còn lại đúng k hồ. Nên nếu gọi d là số lượng hồ nước trong bảng, ta sẽ cần phải lấp tổng cộng là d – k hồ đầu tiên mà mỗi lần lấp ta sẽ đi từ những đỉnh đầu tiên của vector.

Do đó, ta cần phải khôi phục lại bản đồ của trạng thái ban đầu sau đó tiến hành lấp d – k ô nước ban đầu được lưu trong vector.

Độ phức tạp trong trường hợp xấu nhất: **O(n * m)** với n và m lần lượt là kích thước của bảng.