PacMan - DFS

AI has a lot of problems that involve searches. In this track you will learn most of the search techniques used in AI.

In this game, PacMan is positioned in a grid. PacMan has to find the food using Depth First Search (DFS). Assume the grid is completely observable, perform a DFS on the grid and then print the path obtained by DFS from the PacMan to the food.

Input Format

The first line contains 2 space separated integers which is the position of the PacMan.
The second line contains 2 space separated integers which is the position of the food.
The third line of the input contains 2 space separated integers indicating the size of the rows and columns respectively. The largest grid size is 30x30.

This is followed by *row* (r) lines each containing *column* (c) characters. A wall is represented by the character '%' (ascii value 37), PacMan is represented by UpperCase alphabet 'P' (ascii value 80), empty spaces which can be used by PacMan for movement is represented by the character '-' (ascii value 45) and food is represented by the character '.' (ascii value 46)

You have to mark the nodes explored while populating it into the stack and not when its expanded.

Note
+ The grid is indexed as per matrix convention
+ The evaluation process follows iterative-DFS and not recursive-DFS.

Populating Stack

In order to maintain uniformity across submissions, please follow the below mentioned order in pushing nodes to stack. If a node has all the 4 adjacent neighbors. Then,

UP is inserted first into the stack, followed by **LEFT**, followed by **RIGHT** and then by **DOWN**.

so, if (1,1) has all its neighbors not visited, (0,1), (1,0), (1,2), (2,1) then,

1. (0,1) - UP is inserted first
2. (1,0) - LEFT is inserted second
3. (1,2) - RIGHT is inserted third
4. (2,1) - DOWN is inserted fourth (on top)

So, (2,1) is the first to be popped from the stack.

Constraints

1 <= r,c <= 40

Output Format

Each cell in the grid is represented by its position in the grid (r,c). PacMan can move only UP, DOWN, LEFT or RIGHT. Your task is to print all the nodes that you encounter while printing DFS tree. While populating the stack, the following convention must be followed.

```
 %
%--
-
```

In the above cell, LEFT and UP are invalid moves. You can either go RIGHT or DOWN. RIGHT is pushed first

followed by DOWN.

Print the number of nodes explored (a node is marked explored only when the node is popped out of the stack) (E) in the first line. Starting from the source node, 'P' (including it), print all the nodes (r,c) expanded using DFS each node in a new line (r,c) until the food node is explored.

```
E
r c
r1 c1
....
```

Then, print the distance 'D' between the source 'P' and the destination '.' calculated using DFS. D+1 lines follow, each line having a node encountered between 'P' and '.' both included. D+1 lines essentially representing the path between source and the destination.

Sample Input

```
3 9
5 1
7 20
%%%%%%%%%%%%%%%%%%%%
%--------------%---%
%-%%-%%-%%-%%-%%-%-%
%--------P-------%-%
%%%%%%%%%%%%%%%%%%-%
%.----------------%
%%%%%%%%%%%%%%%%%%%%
```

Sample Output

sample output

In this example, PacMan is at the position (3,9) and the food is at the position (5,1). DFS tree is printed starting from (3,9) until the food node is expanded. The DFS path length between (3,9) and (5,1) is 32. All the nodes encountered between (3,9) and (5,1) both included is printed in the next 33 lines.

Task

Your task is to complete the function dfs that takes in r,c as the grid size, pacman_r and pacman_c which is the position of Pacman and food_r and food_c as the position of food and the grid as input and print the output in the required format.