COMP 202 – Spring 2022

Assignment 3: Potato Hunter

Assigned 12 May 2022, Due: 22 May 2022 23:59

1 Introduction

The purpose of this assignment is to provide you with experience in working with one of the traversal algorithms, Depth-first search (DFS). Depth-first search is an algorithm for traversing or searching tree or graph data structures.

In this assignment, you will help Mr. Mole to eat as many potatoes as possible in a farm which have lots of potatoes. The farm can be represented as a rectangular field of n × m cells. Each cell is either empty or impassable. Empty cells are marked with '-', impassable cells are marked with '*'. Every two adjacent cells of different types (one empty and one impassable) are divided by a fence containing one potato. At the beginning Mr. Mole is in some empty cell. At every moment he can move to any empty cell that share a side with the current one. For several starting positions you should calculate the maximum number of potatoes that Mr. Mole can eat. He is able to eat the potato only if he is in the cell adjacent to the fence with this potato.

2 Programming Task

2.1 How to start

- Accept the GitHub Classroom assignment using the link: **https://classroom.github.com/a/XPDyRHpR**
- Clone the GitHub repository created for you to a machine in which you plan to do your work

```
$ git clone https://github.com/KocUniversity/hw3-USER.git
(Replace USER with your GitHub username that you accept the assignment).
```

2.2 Task:

You are going to implement a depth-first search (DFS) algorithm to help the Mr. Mole to eat as much potatoes as possible. Below, we described the specifications of the task and input parameters.

2.2.1 Input Parameters

This section describes the input format. You will get the initial state of the grid from one of the provided input text files, which follow a particular format.

In each input file, the first two lines will contain integers, r and c, representing number of rows, number of columns respectively. Each of the next r lines of the file will contain the grid itself, a set of characters of size r∗c with a line break (\n) after each row. Each grid character will be either a "-" for an empty cell, or a "*" for an impassable cell. It is guaranteed that all border cells are impassable, so Mr. Mole can't go out from the farm. The next line will contain a single integer, p, the number of starting positions to process. Each of the last p lines contains two integers x and y ($1 \leq x \leq r$, $1 \leq y \leq c$) — the row and the column index of one of Mr. Mole's starting positions respectively. Rows are numbered from top to bottom, columns - from left to right. It is guaranteed that all starting positions are empty cells.

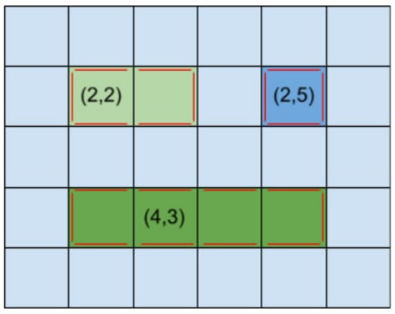
The input files are in inputs folder which is in the same working directory as your program. For example, the following text might be the contents of an example file inp1.txt, a 5x6 grid with 3 starting positions to process:

Input :
```
5
6
******
*--*-*
******
*----*
******
3
2 2
2 5
4 3
```

2.2.2 Output

This section describes the output of the task. Your code should print the k integers — the maximum number of potatoes, that Mr. Mole can eat if he starts in corresponding position. A sample output file out1.txt can be seen below:

Output:
```
6
4
10
```



We can visualize the example input as shown in the figure. Light blue cells are impassable and light green, dark green and dark blue cells are empty. Starting from position (2,2), we can explore the area marked with light green. 6 edges marked with red lines show fences, therefore potatoes which could be collected. Starting from (2,5), we can explore the dark blue region which has 4 fences. Similarly, starting from (4,3), we can explore dark green region which has 10 fences.

2.2.3 Run

Your code should read its input from standard input and write its results to standard output. You are given a starter code which reads the input. You can test your code with input files:

java PotatoHunter < inp1.txt

2.2.4 Time Complexity

Your implementation must have $O(n * m)$ worst case time complexity for calculating the result for one starting position.

3 Evaluation

Your score will be computed out of a maximum of 100 points based on the following distribution:

- **90** Correctness points
- **10** Effective use of version control points

Correctness points: Your code will be evaluated based on a set of given input parameters and should result in correct results.

4 Submission Details

We use GitHub for the submissions.

- Commit all the changes you make:

```
$ git commit -a -m "commit message"
```

- Push your work to GitHub servers:

```
$ git push origin main
```

- To make sure your code is received and avoid any potential problems on Github, submit a copy of your code files on Blackboard as well. Submit all files as a single .zip file, named as: “ID_NAME_SURNAME.zip”

5 Regulations

- You are not allowed to use any other resources than the course textbook.
- You must keep your Github repository private.
- **Questions:** For your further questions about the homework, send an email to: macar20@ku.edu.tr