Enes YARDIM – 13213245090 CMPE242 – Section 02

27/03/2022

Assignment/Homework -1

Problem Statement and Code Design

In this assignment, I aimed to find paths to reach treasures in a maze. I get the map of maze from given file. After reading the file, I used 2-dimensional array to search in maze. I used queue and stack to store visited ways and ways which are reach to the treasure. Code includes some sub-parts to make it modular. Sub-modules are reachable from *Figure-1*.

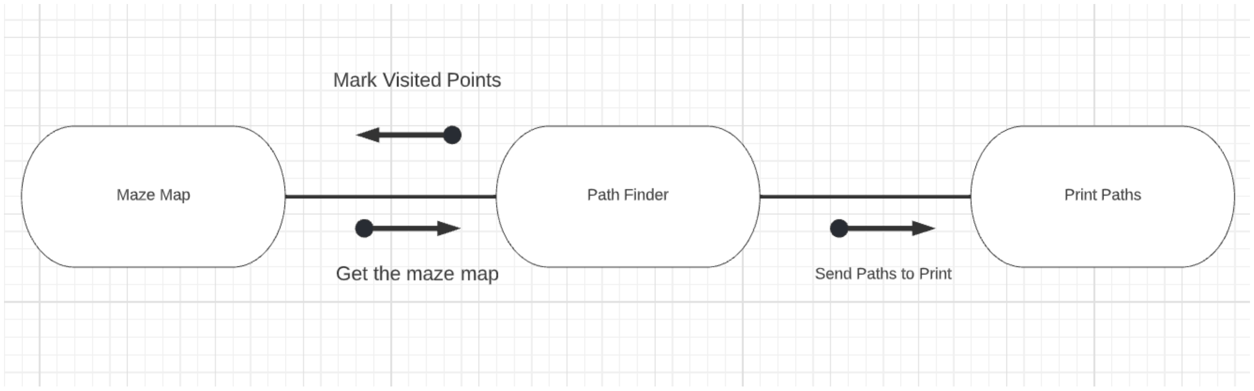

Figure 1

Implementation and Functionality

1) Get the Maze Map

This submodule read map from file and makes path finder use it by storing in two-dimensional. By this way map is more readable and functional.

2) Search the Map

This submodule searches the array on given roads. It follows the lower-case letters which are visitable points in the map. While searching, if there is an "E" letter, it stores the path and "E" to print to the user.

3) Print Paths

This submodule prints if there is a path includes a treasure which is "E". Also let user knows how many findable treasures exist in the map.

```java
public static void main(String[] args) throws
FileNotFoundException {
                        Scanner sc2 = new Scanner(System.in);
                        String filename = sc2.nextLine();
                        File myObj = new File(filename);
                Scanner sc = new Scanner(myObj);

            int rows = 11;
              int columns =16;
              String [][] maze_t = new String[rows][columns];
                          while(sc.hasNextLine()) {
                            for (int i=0; i<maze_t.length; i++) {
                              String[] line =
sc.nextLine().trim().split("");
                              for (int j=0; j<line.length; j++) {
                                     maze_t[i][j] = (line[j]);
                              }
                            }
                          }
```

```java
static void find_path(String[][] maze_t, int m, int n, int x, int y,
String prevC, String newC)
            {
            ...
static boolean isValid(String[][] maze_t, int m, int n, int x, int
y, String prevC, String newC)
            {
                        if(x < 0 || x >= m || y < 0 || y >= n ||
isLowerCase(maze_t[x][y])==false
                        || maze_t[x][y].equals(newC))
                                    return false;
                        return true;
            }
...
if(isValid(maze_t, m, n, posX-1, posY, prevC, newC))
                                    {
                                                s.push(maze_t[posX-
1][posY]);
                                                maze_t[posX-1][posY]=
newC;
                                                queue.add(new
Point(posX-1, posY));

                                    }
...
```

```java
void print(){//printing stack
              for(int i = 0;i<=top;i++){
                System.out.print(a[i]);

              }
```

Figure-2

Testing

After implemented tests, algorithm finds the treasure if it is connected to path from right, up or down. If treasure is left side of the road algorithm can not find the Treasure. Algorithm is able to detect if there is no treasure by visiting all the available roads in the maze. After visiting a road, visited points are marked. As a marked sign I use “7”. So, after visit a point it still stays as “7” so paths include “7” instead of correct letter.

Final Assessments

- Difficulties I had challenged are using a data structure first time and searching algorithms. These difficulties cost me a lot of time to learn, understand and implement.
- As an outcome of this assignment, I learned to think how to solve an algorithm problem wisely and fast.