Assignment 6

2020CS10341

10 February 2021

1 Bonus: Maze Traversal

`slmtrx` represents the solution matrix and it gets constantly updated as we move through the grid looking for the exit `end`. Initially, the solution matrix is the same as the initial matrix.
`pth` represents the path list(stack) that we are going to output.
`lcn` represents the current location in the Maze/solution matrix.

1.1 Main While Loop

```
while lcn != end:
    if check(slmtrx,lcn[0],lcn[1]+1):
        slmtrx[lcn[0]][lcn[1]] = "Y"
        pth.append("R")
        lcn = (lcn[0],lcn[1]+1)
    elif check(slmtrx,lcn[0]+1,lcn[1]):
        slmtrx[lcn[0]][lcn[1]] = "Y"
        pth.append("D")
        lcn = (lcn[0]+1,lcn[1])
    elif check(slmtrx,lcn[0],lcn[1]-1):
        slmtrx[lcn[0]][lcn[1]] = "Y"
        pth.append("L")
        lcn = (lcn[0],lcn[1]-1)
    elif check(slmtrx,lcn[0]-1,lcn[1]):
        slmtrx[lcn[0]][lcn[1]] = "Y"
        pth.append("U")
        lcn = (lcn[0]-1,lcn[1])
    else:
        if pth:
            z = pth.pop()
            slmtrx[lcn[0]][lcn[1]] = "X"
            if z == "D":
                lcn = (lcn[0]-1,lcn[1])
            elif z == "R":
```

```
            lcn = (lcn[0],lcn[1]-1)
        elif z == "U":
            lcn = (lcn[0]+1,lcn[1])
        elif z == "L":
            lcn = (lcn[0],lcn[1]+1)
    else:
        return "No␣path"
```

Pre-Condition: *At every point, we are removing one location that we cannot traverse in the maze.*
Post-Condition: *Since there are only finite locations in the maze, either we will find the exit or there will be no path out of the maze.*
Now we check whether we can go towards right, down, left and up in that order and then take the first possible step. Then we mark the current cell as `Y` in the solution Matrix so that we can check later that the current cell has been visited and do not visit it again.
This way, we are decreasing the number of traversable locations in the Matrix by 1.
Now, if we reach a dead end (no place to go), we must go back to the location from where we got to this location in the maze, and we also mark the current place in the maze as `X` as we cannot go anywhere from there. This way as well, we decrease the number of traversable locations in the maze by 1 (although we do go back to a previously non-traversable location `Y`, now there is one less possible step we can take from there.)
So this way, we are eliminating possible paths until we reach the exit. After every step, we append the step taken to the path stack `pth` and then when we reach the end location `E`, we return `pth` as output.