Assignment 5

Due date: Friday, Dec 1, 2019 at 11:59pm
Individual assignment. No group work allowed.
Weight: 10% of the final grade.

Finding a shortest path in a maze

Implement a function called `solve_maze()` that takes a single argument - a filename. The function then:

- reads a maze from the file,
- determines the starting point and exit point in the maze,
- calculates the shortest path from the starting point to the exit point,
- if a path exists:
    - prints out the maze with the shortest path marked with "+",
    - prints out the length of the shortest path,
- if there is no path:
    - the function just prints the maze unmodified,
    - and a message "There is no path."

To find the shortest path, you need to implement Dijkstra's algorithm. It will probably help if you convert your maze to a graph data-structure, but this is not strictly necessary.

Submission

Submit your assignment as a Jupyter notebook or .py file to D2L.

Examples:

Input file test1.txt:

```
..#..E
..#...
S.####
..#...
..#...
..#...
```

Running solve_maze():

```
solve_maze("test1.txt")
..#..E
..#...
S.####
..#...
..#...
```

```
..#...
There is no path.
```

Input file test2.txt:

```
......
......
S.....
......
......
.....E
```

Running solve_maze():

```
solve_maze("test2.txt")
......
......
S+++++
.....+
.....+
.....E
Path length: 7
```

Helpful functions

You can use any of the functions below in your implementation.

```
def read_maze(fname):
    rows = []
    with open(fname) as f:
        for line in f:
            line = line.rstrip()
            if len(line) > 0:
                rows.append(list(line))
    return rows

m = read_maze("m4.txt")
m
```

```
[['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#']
 ['#', 'E', '#', '+', '+', '+', '+', '+', '#', '+', '+', '+', '#']
 ['#', '+', '#', '+', '.', '#', '#', '+', '+', '+', '#', '+', '#']
 ['#', '+', '#', '+', '.', '#', '#', '.', '#', '#', '#', '+', '#']
 ['#', '+', '+', '+', '.', '.', '.', '.', '#', '.', '.', '+', '#']
 ['#', '#', '#', '#', '#', '#', '#', '#', '#', '.', '#', '+', '#']
 ['#', '.', '.', '.', '.', '.', '.', '.', '.', '.', '#', 'S', '#']
```

```
 ['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#']
```

```
def print_maze(maze):
    for row in maze:
        print("".join(row))

print_maze(m)
```

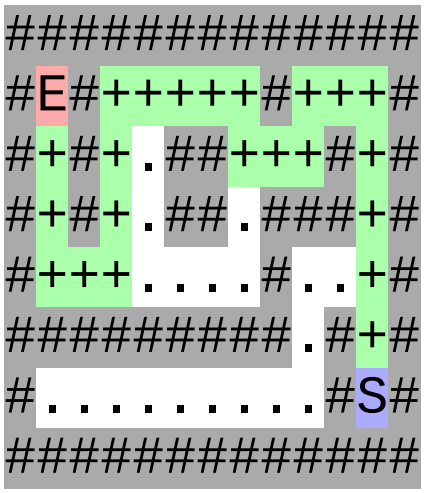
```
#############
#E#++++++++##
#++#.#######
#++#.##.####
#+++....#..+
#########.#+#
#.........#S#
#############
```

```
from PIL import Image, ImageDraw
from IPython.display import display # to display images

def plot_maze(maze):
    scale = 15
    trans = { 's': (0,0,255), '.': (255,255,255), 'e': (255,0,0),
'#': (0,0,0) }
    nrows = len(maze)
    ncols = len(maze[0])
    im = Image.new( "RGB", (ncols, nrows))
    pixels = im.load()
    for row in range(nrows):
        for col in range(ncols):
            pixels[col,row] = trans[maze[row][col].lower()]
    im = im.resize((ncols*scale, nrows*scale))
    draw = ImageDraw.Draw(im)
    dx = im.size[0] / ncols
    dy = im.size[1] / nrows
    for ix in range(1,ncols+1):
        line = ((ix * dx, 0), (ix * dx, im.size[1]))
        draw.line(line, fill=(127, 127, 127))
    for iy in range(1,nrows+1):
        line = ((0, iy * dy), (im.size[0], iy * dy))
        draw.line(line, fill=(127, 127, 127))
```

```
    display(im)

m = read_maze("m4.txt")
plot_maze(m)
```

```
# print a colored maze
def print_cmaze(maze):
    import IPython.display as ipd
    html = """
    <style>
        .mazeWall { background-color: #aaa; }
        .mazeEmpty { background-color: #fff; }
        .mazeStart { background-color: #aaf; }
        .mazeEnd { background-color: #faa; }
        .mazePath { background-color: #afa; }
        .colPrint {
            font-family: monospace;
            font-size: 20pt;
            line-height: 1em;
            }
        .colPrint span {
            display: inline-block;
            width: 25px;
            height: 25px;
            text-align: center;
            border: solid 1px #8880;
            line-height: 25px; }
    </style>
    """;
    trans = { 's': "mazeStart", '.': "mazeEmpty", 'e': "mazeEnd",
'#': "mazeWall" }
    html += "<div class=colPrint>";
    for row in maze:
        for c in row:
            html += "<span class={}>{}</span>".format(trans[c.lowe
        html += "<br>"
    html += '</div>'
    display(ipd.HTML(html))
```

```
m = read_maze("m4.txt")
print_cmaze(m)
```




```
def matrix_to_table(m):
    from IPython.display import display, HTML
    html = "<table style='border: 1px solid blue'>"
    for row in m:
        html += "<tr>"
        for c in row:
            html += "<td>" + str(c) + "</td>"
        html += "</tr>"
    html += "</table>"
    display(HTML(html))
```