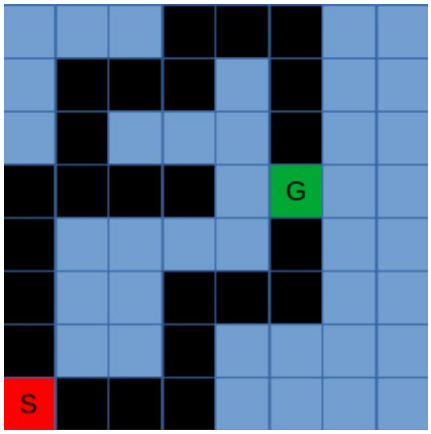
Artificial Intelligence and Computer Vision
Lab 6
Marek Antoszewski

Breadth First Search implementation and calculation on first example




```
solved first maze:
[['#' '#' '#' ' ' ' ' ' ' ' ' '#' '#']
 ['#' ' ' ' ' ' ' ' ' '#' ' ' '#' '#']
 ['#' ' ' '#' '#' '#' ' ' '#' '#']
 [' ' ' ' ' ' ' ' ' ' '#' 'x' '#' '#']
 [' ' '#' '#' '#' '#' 'x' '#' '#']
 [' ' '#' '#' 'x' 'x' 'x' '#' '#']
 [' ' '#' '#' 'x' '#' '#' '#' '#']
 ['x' 'x' 'x' 'x' '#' '#' '#' '#']]
```

```python
# calculation BFS
end = np.where(maze1 == "G")
start = np.where(maze1 == "S")
walls = []
cords_bfs = np.where(maze1 == "#")
for index in range(len(cords_bfs[0])):
    walls.append((cords_bfs[0][index], cords_bfs[1][index]))
height, width = maze1.shape
child, parent = bfs(height, width, start, end, walls)
solution = {}
for idx in range(len(child)):
    solution[child[idx]] = parent[idx]

x_s, y_s = start[0][0], start[1][0]
x, y = end[0][0], end[1][0]
path_bfs = [(x, y)]
while (x, y) != (x_s, y_s):
    (x, y) = solution[x, y]
    path_bfs.append((x, y))
del walls
```

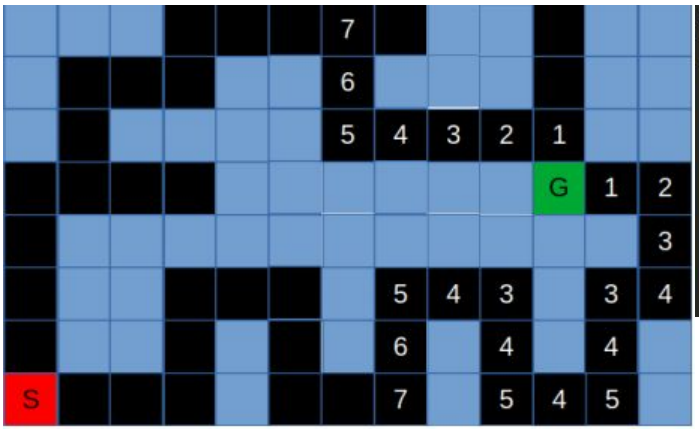
I am finding end start point, and walls coordinates, my bfs function requires to get height and width of maze too, so i get the shape of maze1.

Then in bfs I loop over length of front array containing “moves” across the grid.

```
def bfs(wid, hei, start_pos, end_pos, obstacle):
    front = [(start_pos[0][0], start_pos[1][0])]
    visit = []
    prev = [(start_pos[0][0], start_pos[1][0])]
    while len(front) > 0:
        (x_c, y_c) = front[0]

        if (x_c, y_c) == (end_pos[0][0], end_pos[1][0]):
            visit.append((x_c, y_c))
            del front
            break

        if (x_c, y_c) not in visit:

            if x_c + 1 < hei and (x_c + 1, y_c) not in obstacle and (x_c + 1, y_c) not in visit:
                front.append((x_c + 1, y_c))
                prev.append((x_c, y_c))
            if x_c - 1 >= 0 and (x_c-1, y_c) not in obstacle and (x_c - 1, y_c) not in visit:
                front.append((x_c - 1, y_c))
                prev.append((x_c, y_c))
            if y_c + 1 < wid and (x_c, y_c + 1) not in obstacle and (x_c, y_c + 1) not in visit:
                front.append((x_c, y_c + 1))
                prev.append((x_c, y_c))
            if y_c - 1 >= 0 and (x_c, y_c-1) not in obstacle and (x_c, y_c - 1) not in visit:
                front.append((x_c, y_c - 1))
                prev.append((x_c, y_c))
            visit.append((x_c, y_c))

        del front[0]
    del prev[len(prev) - 1]
    return visit, prev
```

A star algorithm, implementation and solution on given mazes

In this algorithm we explore neighbours but also check their manhattan distance(h score) from goal and the also the node to node distance(g score) from goal.
We consider only that neighbour that has the sum of both the distances minimum because we are interested in exploring the shortest path.




```
solved second maze:
[['#' '#' '#' 'x' 'x' 'x' 'x' ' ' '#' '#' ' ' '#' '#']
 ['#' 'x' 'x' 'x' '#' '#' 'x' '#' '#' '#' ' ' '#' '#']
 ['#' 'x' '#' '#' '#' '#' 'x' 'x' 'x' 'x' 'x' '#' '#']
 ['x' 'x' ' ' ' ' '#' '#' '#' '#' '#' '#' 'x' ' ' ' ']
 ['x' '#' '#' '#' '#' '#' '#' '#' '#' '#' '#' '#' ' ']
 ['x' '#' '#' ' ' ' ' ' ' '#' ' ' ' ' ' ' '#' ' ' ' ']
 ['x' '#' '#' ' ' '#' ' ' '#' ' ' '#' ' ' '#' ' ' '#']
 ['x' ' ' ' ' ' ' '#' ' ' ' ' ' ' '#' ' ' ' ' ' ' '#']]
```

The algorithm works perfectly fine for second maze. The method of its operation is not that hard

```
# A-star calculations
a = alg.AStar()
cords_astar = np.where(maze2 == '#')
walls = []
for index in range(len(cords_astar[0])):
    walls.append((cords_astar[0][index], cords_astar[1][index]))
a.init_grid(8, 13, walls, (7, 0), (3, 10))
path = a.solve()
```

implementation of A star search class in source code

We can divide it into 3 main steps

1. Generate a list of all possible next steps
   towards goal from current position

2. Store children in priority queue
   based on distance to goal, closest first

3. Select closest child and repeat until goal reached or no more children

Then in case of both tasks I just replace correct path with Xs in my mazes matrices and plot it.