

### BFS Code

```
x_size = 20
y_size = 20

def good_neighbors(point, maze):
    neighbor_list = []
    for i in range(-1, 2):
        for j in range(-1, 2):
            neighbor_list.append((point[0]+i, point[1]+j))
    return set(neighbor_list)

def is_valid(point, maze):
    if 0 <= point[0] < x_size:
        if 0 <= point[1] < y_size:
            point_val = maze[int(point[0])][int(point[1])]
            if point_val is not 1:
                return True
    return False

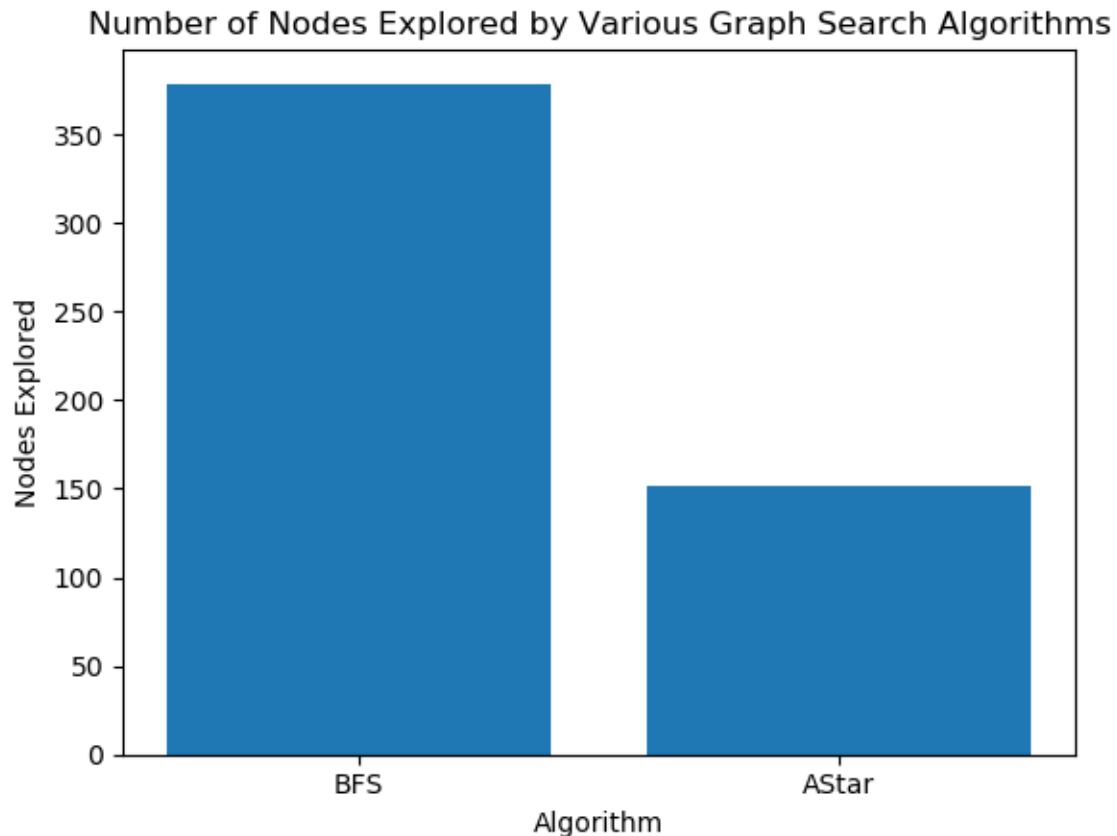
def bfs_path(maze, start, goal):
    visited_nodes = set(start)
    queue = [(start, [start])]
    while queue:
        [vertex, path] = queue.pop()
        for next_node in good_neighbors(vertex, maze) - set(path) - visited_nodes:
            if is_valid(next_node, maze):
                visited_nodes.add(next_node)
                if next_node == goal:
                    return len(visited_nodes), path+[next_node]
                else:
                    queue.append((next_node, path+[next_node]))

if __name__ == "__main__":
    start = (1, 1)
    goal = (19, 19)
    maze = \
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]]
```

```

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
bfs_path(maze, start, goal)

```



BFS did not return the shortest path, because upon running the code the BFS path was 59 nodes long, while the A\* path was only 28 nodes long.

```

(379, [(1, 1), (0, 2), (0, 3), (0, 4), (0, 5), (1, 6), (2, 5), (3, 5), (
59
A* Algorithm
totalNodesvisited = 152 path= [(1, 1), (2, 2), (3, 3), (4, 4), (5, 5),
path length= 28

```