Mark Arias 012416064
Rifat Hasan 025538368
7/23/21

CECS 451 Assignment 2 – Maze Search

Experiment Results and Output

Depth First Search Results

Small Maze

```
-----Project 2-----

Original Maze:
%%%%%%%%%%%%%%%%%%%%%%
% %%        % %      %
%    %%%%%% % %%%%%% %
%%%%%%     P  %      %
%    % %%%%%% %% %%%%%
% %%%% %         %   %
%        %%% %%%   % %
%%%%%%%%%%    %%%%%% %
%.         %%        %
%%%%%%%%%%%%%%%%%%%%%%
Starting coordinates: (Row, Column)  ** zero start index **
(3, 11)
End coordinates: (Row, Column)  ** zero start index **
(8, 1)

Maze destination reached
Number of nodes: 37
Stack:  [(2, 11), (3, 10), (2, 13), (5, 12), (4, 16), (6, 12)]
Path taken:
%%%%%%%%%%%%%%%%%%%%%%
% %%        % %      %
%    %%%%%% % %%%%%% %
%%%%%%     ooo%      %
%    % %%%%%%o%% %%%%%
% %%%% %     oooo%ooo%
%        %%% %%%ooo%o%
%%%%%%%%%%oooo%%%%%%o%
%.ooooooooo%%oooooooo%
%%%%%%%%%%%%%%%%%%%%%%
```

Medium Maze

```
-----Project 2-----

Original Maze:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                 P%
% %%%%%%%%%%%%%%%%%%%%%%% %%%%%%%% %
% %%   %   %      %%%%%%%   %%     %
% %% % % % % %%%% %%%%%%%%% %% %%%%%
% %% % % % %             %% %%     %
% %% % % % % % %%%%  %%%    %%%%%% %
% %  % % %   %    %% %%%%%%%%      %
% %% % % %%%%%%%% %%        %% %%%%%
% %% %   %%       %%%%%%%%% %%     %
%    %%%%%% %%%%%%%      %% %%%%%% %
%%%%%%      %       %%%% %% %      %
%      %%%%%% %%%%% %    %% %% %%%%%
% %%%%%%      %       %%%%% %%     %
%        %%%%%% %%%%%%%%%%% %%  %% %
%%%%%%%%%%                  %%%%%% %
%.         %%%%%%%%%%%%%%%%        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Starting coordinates: (Row, Column)  ** zero start index **
(1, 34)
End coordinates: (Row, Column)  ** zero start index **
(16, 1)

Maze destination reached
Number of nodes: 146
Stack:  [(2, 34), (2, 25), (4, 12), (6, 14), (4, 17), (6, 19), (16, 27), (14, 15)]
Path taken:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%oooooooooooooooooooooooooooooooooo%
%o%%%%%%%%%%%%%%%%%%%%%%%o%%%%%%%% %
%o%%ooo%ooo%      %%%%%%%ooo%%     %
%o%%o%o%o%o% %%%% %%%%%%%%%o%% %%%%%
%o%%o%o%o%o%ooooooooooooo%%o%%     %
%o%%o%o%o%o%o% %%%%oo%%%oooo%%%%%% %
%o%oo%o%o%ooo%    %%o%%%%%%%%      %
%o%%o%o%o%%%%%%%% %%oooooooo%% %%%%%
%o%%o%ooo%%       %%%%%%%%%o%%     %
%oooo%%%%%% %%%%%%%      %%o%%%%%% %
%%%%%%      %       %%%% %%o%      %
%      %%%%%% %%%%% %    %%o%% %%%%%
% %%%%%%      %       %%%%%o%%     %
%        %%%%%% %%%%%%%%%%%o%%  %% %
%%%%%%%%%%oooooooooooooooooo%%%%%% %
%.ooooooooo%%%%%%%%%%%%%%%%        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Large Maze

```
**********************************************************
 Maze loaded into memory to solve shown below
**********************************************************
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       % % %           %   %     % %
% %%%%%%% % %%% % %%% %%% %%%%%%% % %
%       %       % %     %     % %   %
%%%%% %%%%% %%% % % % %%% %%%%% % %%%
%   % % % %   % % % %   % %   % %   %
% %%% % % % %%% %%%%% %%% % %%% %%% %
%       %     %   %   %     % % %   %
%%% %%%%%%%%% %%%%%%% %%% %%% % % % %
%             %       % %   %     % %
% % %%%%% % %%% % % %%% % %%% %%% % %
% % %     % % % % %     %   % % % % %
% % % %%%%%%% % %%%%%%%%% %%% % %%% %
% % % %     %   %     %     %   %   %
%%% %%% % %%%%% %%%%% %%% %%% %%%%% %
%     % % %     % %     % %   % % % %
% % % % % %%% %%% %%% %%% % % % % % %
% % % % %                 % % %     %
%%% %%%%%%% % % %%%%% %%% % %%% %%%%%
%       % % % %     %   %     % %   %
%%%%% % % %%%%%%%%% %%%%%%%%%%% % %%%
%   % %           % %     %   % %   %
% %%% %%%%% %%%%%%%%% %%%%% % % %%% %
% %   %      %        %     %       %
% % % %%%%% %%% % % % % %%%%%%%%%%%%%
% % %   %     % % % %       %   % % %
% % %%% %%% % % % %%%%%%%%% %%% % % %
% %   % %   % % %   % %   % % %     %
% %%% %%% %%%%% %%% % % %%%%% % %%%%%
%       %   %     % %     %   % %   %
%%% % %%%%% %%%%% %%% %%% % %%% % %%%
% % % % % % % %     % %   % %   % % %
% % %%% % % % % %%%%%%%%% % % % % % %
%   %   %   %                 %     %
% % % % %%% %%% %%%%%%% %%% %%% %%% %
%.% % %       %   %       %   % %  P%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
**********************************************************
Maze Information
Starting coordinates: (Row, Column)  ** zero start index **
(35, 35)
End coordinates: (Row, Column)  ** zero start index **
(35, 1)
Current Position 35 35
Stack: [(34, 35), (35, 34)]
```

```
Starting coordinates: (Row, Column)  ** zero start index **
(35, 35)
End coordinates: (Row, Column)  ** zero start index **
(35, 1)

Maze destination reached
Number of nodes: 387
Stack:  [(32, 35), (32, 33), (30, 31), (32, 27), (34, 27), (32, 25), (34, 23), (23, 14), (
24, 17), (24, 19), (22, 21), (16, 31), (16, 33), (6, 35), (10, 33), (8, 31), (8, 29), (16,
 29), (16, 25), (16, 21), (18, 21), (16, 17), (18, 15), (15, 12), (10, 17), (10, 19), (2,
17), (4, 15), (2, 11), (7, 10), (10, 11), (10, 9), (8, 3), (19, 2), (24, 5), (30, 5), (34,
 3), (32, 1)]
Path taken:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       %o% %ooooooooooo%   %     % %
% %%%%%%%o% %%%o% %%%o%%% %%%%%%% % %
%       %ooooooo% %ooooo%     % %   %
%%%%% %%%%%o%%% % %o%o%%% %%%%% % %%%
%   % % % %ooo% % %o%ooo% %   % %   %
% %%% % % %o%%% %%%%%o%%% % %%% %%% %
%       %  ooo%   %ooo%     % % %ooo%
%%% %%%%%%%%%o%%%%%%%o%%% %%% % %o%o%
%ooooooooooooo%ooooooo% %   %ooooo%o%
%o%o%%%%% % %%%o% % %%% % %%%o%%% %o%
%o%o%     % %o%o% %     %   %o%o% %o%
%o%o% %%%%%%%o%o%%%%%%%%% %%%o%o%%%o%
%o%o% %ooooo%ooo%     %     %ooo%ooo%
%%%o%%%o%o%%%%%o%%%%% %%% %%%o%%%%%o%
%ooooo%o%o%  ooo% %     % %ooo% % %o%
%o%o%o%o%o%%%o%%% %%% %%% %o% % % %o%
%o%o%o%o%ooooooooooooooooo%o% %ooooo%
%%%o%%%%%%%o%o% %%%%% %%%o%o%%%o%%%%%
%  ooooo%o%o%o%     %   %ooooo%o%ooo%
%%%%%o%o%o%%%%%%%%% %%%%%%%%%%%o%o%%%
%ooo%o%oooooooooooo% %     %ooo%o%ooo%
%o%%%o%%%%%o%%%%%%%%% %%%%%o%o%o%%%o%
%o%ooo%oooooo% ooooooo%ooooo%ooooooo%
%o%o% %%%%%o%%%o% % %o%o%%%%%%%%%%%%%
%o%o%   %ooooo%o% % %ooooooo%   % % %
%o%o%%% %%%o%o%o% %%%%%%%%%o%%% % % %
%o%ooo% %ooo%o%o%   % %   %o% %     %
%o%%%o%%%o%%%%%o%%% % % %%%%% % %%%%%
%ooooooo%ooo%ooooo% %     %   % %   %
%%%o% %%%%%o%%%%%o%%% %%% % %%% % %%%
% %o% %o%o%o%o%ooooo% %   % %ooo% % %
% %o%%%o%o%o%o%o%%%%%%%%% % %o%o% % %
%ooo%ooo%ooo%ooooooooooooooooo%ooooo%
%o% %o%o%%%o%%%o%%%%%%% %%% %%%o%%%o%
%.% %o%ooooooo%ooo%       %   %o%ooo%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Breadth First Search Results

Small Maze

```
**********************************************************
 Maze loaded into memory to solve shown below
**********************************************************
%%%%%%%%%%%%%%%%%%%%%%
% %%        % %      %
%    %%%%%% % %%%%%% %
%%%%%%     P  %      %
%    % %%%%%% %% %%%%%
% %%%% %         %   %
%        %%% %%%   % %
%%%%%%%%%%    %%%%%% %
%.         %%        %
%%%%%%%%%%%%%%%%%%%%%%
**********************************************************
Maze Information
Starting coordinates: (Row, Column)  ** zero start index **
(3, 11)
End coordinates: (Row, Column)  ** zero start index **
(8, 1)
Current Position 3 11
Current positions in queue: 3
```

```
%%%%%%%%%%%%%%%%%%%%%%
%+%%++++++++%+%  ++++%
%++++%%%%%%+%+%%%%%%+%
%%%%%%++++++++%++++++%
%++++%+%%%%%%+%%+%%%%%
%+%%%%+%+++++++++%+++%
%++++++++%%%+%%%+++%+%
%%%%%%%%%%++++%%%%%%+%
%.+++++++++%%++++++++%
%%%%%%%%%%%%%%%%%%%%%%
Current Position 8 1
Maze end reached
True
```

Medium Maze

```
**********************************************************
 Maze loaded into memory to solve shown below
**********************************************************
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                 P%
% %%%%%%%%%%%%%%%%%%%%%%% %%%%%%%% %
% %%   %   %      %%%%%%%   %%     %
% %% % % % % %%%% %%%%%%%%% %% %%%%%
% %% % % % %             %% %%     %
% %% % % % % % %%%%  %%%    %%%%%% %
% %  % % %   %    %% %%%%%%%%      %
% %% % % %%%%%%%% %%        %% %%%%%
% %% %   %%       %%%%%%%%% %%     %
%    %%%%%% %%%%%%%      %% %%%%%% %
%%%%%%      %       %%%% %% %      %
%      %%%%%% %%%%% %    %% %% %%%%%
% %%%%%%      %       %%%%% %%     %
%        %%%%%% %%%%%%%%%%% %%  %% %
%%%%%%%%%%                  %%%%%% %
%.         %%%%%%%%%%%%%%%%        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
**********************************************************
Maze Information
Starting coordinates: (Row, Column)  ** zero start index **
(1, 34)
End coordinates: (Row, Column)  ** zero start index **
(16, 1)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%++++++++++++++++++++++++++++++++++%
%+%%%%%%%%%%%%%%%%%%%%%%%+%%%%%%%%+%
%+%%+++%+++%++++++%%%%%%%+++%%+++++%
%+%%+%+%+%+%+%%%%+%%%%%%%%%+%%+%%%%%
%+%%+%+%+%+%+++++++++++++%%+%%+++++%
%+%%+%+%+%+%+%+%%%%++%%%++++%%%%%%+%
%+%++%+%+%+++%++++%%+%%%%%%%%++++++%
%+%%+%+%+%%%%%%%%+%%++++++++%%+%%%%%
%+%%+%+++%%+++++++%%%%%%%%%+%%+++++%
%++++%%%%%%+%%%%%%%++++++%%+%%%%%%+%
%%%%%%++++++%+++++++%%%%+%%+%++++++%
%++++++%%%%%%+%%%%%+%++++%%+%%+%%%%%
%+%%%%%%++    %+++++++%%%%%+%%+++++%
%++++++++%%%%%%+%%%%%%%%%%%+%%++%%+%
%%%%%%%%%%++++++++++++++++++%%%%%%+%
%.+++++++++%%%%%%%%%%%%%%%%++++++++%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Current Position 16 1
Maze end reached
True
```

Large Maze

```
***********************************************************
 Maze loaded into memory to solve shown below
***********************************************************
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       % % %           %   %     % %
% %%%%%%% % %%% % %%% %%% %%%%%%% % %
%       %       % %     %     % %   %
%%%%% %%%%% %%% % % % %%% %%%%% % %%%
%   % % % %   % % % %   % %   % %   %
% %%% % % % %%% %%%%% %%% % %%% %%% %
%       %     %   %   %     % % %   %
%%% %%%%%%%%% %%%%%%% %%% %%% % % % %
%             %       % %   %     % %
% % %%%%% % %%% % % %%% % %%% %%% % %
% % %     % % % % %     %   % % % % %
% % % %%%%%%% % %%%%%%%%% %%% % %%% %
% % % %     %   %     %     %   %   %
%%% %%% % %%%%% %%%%% %%% %%% %%%%% %
%     % % %     % %     % %   % % % %
% % % % % %%% %%% %%% %%% % % % % % %
% % % % %                 % % %     %
%%% %%%%%%% % % %%%%% %%% % %%% %%%%%
%       % % % %     %   %     % %   %
%%%%% % % %%%%%%%%% %%%%%%%%%%% % %%%
%   % %           % %     %   % %   %
% %%% %%%%% %%%%%%%%% %%%%% % % %%% %
% %   %      %        %     %       %
% % % %%%%% %%% % % % % %%%%%%%%%%%%%
% % %   %     % % % %       %   % % %
% % %%% %%% % % % %%%%%%%%% %%% % % %
% %   % %   % % %   % %   % % %     %
% %%% %%% %%%%% %%% % % %%%%% % %%%%%
%       %   %     % %     %   % %   %
%%% % %%%%% %%%%% %%% %%% % %%% % %%%
% % % % % % % %     % %   % %   % % %
% % %%% % % % % %%%%%%%%% % % % % % %
%   %   %   %                 %     %
% % % % %%% %%% %%%%%%% %%% %%% %%% %
%.% % %       %   %       %   % %  P%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
***********************************************************
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%+++++++%+%+%+++++++++++%+++%+++++%+%
%+%%%%%%%+%+%%%+%+%%%+%%%+%%%%%%%+%+%
%+++++++%+++++++%+%+++++%+++++%+%+++%
%%%%%+%%%%%+%%%+%+%+%+%%%+%%%%%+%+%%%
%+++%+%+%+%+++%+%+%+%+++%+%+++%+%+++%
%+%%%+%+%+%+%%%+%%%%%+%%%+%+%%%+%%%+%
%+++++++%+++++%+++%+++%+++++%+%+%+++%
%%%+%%%%%%%%%+%%%%%%%+%%%+%%%+%+%+%+%
%+++++++++++++%+++++++%+%+++%+++++%+%
%+%+%%%%%+%+%%%+%+%+%%%+%+%%%+%%%+%+%
%+%+%+++++%+%+%+%+%+++++%+++%+%+%+%+%
%+%+%+%%%%%%%+%+%%%%%%%%%+%%%+%+%%%+%
%+%+%+%+++++%+++%+++++%+++++%+++%+++%
%%%+%%%+%+%%%%%+%%%%%+%%%+%%%+%%%%%+%
%+++++%+%+%+++++%+%+++++%+%+++%+%+%+%
%+%+%+%+%+%%%+%%%+%%%+%%%+%+%+%+%+%+%
%+%+%+%+%+++++++++++++++++%+%+%+++++%
%%%+%%%%%%%+%+%+%%%%%+%%%+%+%%%+%%%%%
%+++++++%+%+%+%+++++%+++%+++++%+%+++%
%%%%%+%+%+%%%%%%%%%+%%%%%%%%%%%+%+%%%
%   %+%+++++++++++%+%+++++%+++%+%+++%
% %%%+%%%%%+%%%%%%%%%+%%%%%+%+%+%%%+%
% %+++%++++++%++++++++%+++++%+++++++%
%+%+%+%%%%%+%%%+%+%+%+%+%%%%%%%%%%%%%
%+%+%+++%+++++%+%+%+%+++++++%+++%+%+%
%+%+%%%+%%%+%+%+%+%%%%%%%%%+%%%+%+%+%
%+%+++%+%+++%+%+%+++%+%+++%+%+%+++++%
%+%%%+%%%+%%%%%+%%%+%+%+%%%%%+%+%%%%%
%+++++++%+++%+++++%+%+++++%+++%+%+++%
%%%+%+%%%%%+%%%%%+%%%+%%%+%+%%%+%+%%%
%+%+%+% % %+%+%+++++%+%+++%+%+++%+%+%
%+%+%%% % %+%+%+%%%%%%%%%+%+%+%+%+%+%
%+++%   %   %+++++++++++++++++%+++++%
%+%+% % %%% %%%+%%%%%%%+%%%+%%%+%%%+%
%.%+% %       %+++%+++++++%+++%+%+++%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Current Position 35 1
Maze end reached
True
```

Source Code Section

```
# Project 2
# Mark Arias 012416064
# Rifat Hasan 025538368
# CECS 451
#---------------------------------------------------------------------------
import pandas as pd
import numpy as np


# class and method definitions
# Definition of a graph object
class Graph(object):
    """ Constructs a graph data structure. """

    def __init__(self, data_file_path=""):

        ld = LoadData(data_file_path)
        self.data = ld.LoadData_Pandas()
        self.columns_count = ld.ColumnsCount()

        self.vert_dict = {}
        self.num_vertices = 0

    def __iter__(self):
        return iter(self.vert_dict.values())

    def graph_length(self):
        return self.columns_count

    # this is professors method for constructing the graph
    def graph_build(self):
        """ This method construct graph from a data file """
```

```
        for row in range(self.columns_count):
            self.add_vertex(row)
            for column in range(self.columns_count):
                if self.data.loc[row, column] != 0:
                    self.add_edge(row, column, self.data.loc[row, column])

    def add_vertex(self, node):

        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(node)
        self.vert_dict[node] = new_vertex
        return new_vertex

    def get_vertex(self, n):

        if n in self.vert_dict:
            return self.vert_dict[n]
        else:
            return None

    def add_edge(self, frm, to, cost=0):

        if frm not in self.vert_dict:
            self.add_vertex(frm)
        if to not in self.vert_dict:
            self.add_vertex(to)

        self.vert_dict[frm].add_neighbor(self.vert_dict[to], cost)

    # The line below will be enabled if the graph is not directed
    # self.vert_dict[to].add_neighbor(self.vert_dict[frm], cost)

    def get_vertices(self):
        return self.vert_dict.keys()

    def graph_summary(self):

        # print("The number of nodes in the graph is: ",
self.columns_count)
```

```python
        for v in self.vert_dict.values():
            for w in v.get_connections():
                vid = v.get_id()
                wid = w.get_id()
                print("( %s , %s, %3d)" % (vid, wid, v.get_weight(w)))

        for v in self.vert_dict.values():
            print("g.vert_dict[%s]=%s" % (v.get_id(),
self.vert_dict[v.get_id()]))


# -------------------[End of class]-------------------------


class Vertex:
    """ keeps a node information """

    def __init__(self, node):
        self.id = node
        self.adjacent = {}

    def __str__(self):
        return str(self.id) + " adjacent: " + str([x.id for x in
self.adjacent])

    def add_neighbor(self, neighbor, weight=0):
        self.adjacent[neighbor] = weight

    def get_connections(self):
        return self.adjacent.keys()

    def get_id(self):
        return self.id

    def get_weight(self, neighbor):
        return self.adjacent[neighbor]


# -------------------[End of class]--------------------------------
```

```python
# maze loader from input file
def load_maze(file_name):
    f = open(file_name)
    maze = f.read()
    f.close()
    return maze


# convert maze into readable array
def convert_maze(maze):
    converted_maze = []
    lines = maze.splitlines()
    for line in lines:
        converted_maze.append(list(line))
    return converted_maze


# make the maze print in a readable way
def print_maze(maze):
    for row in maze:
        for item in row:
            print(item, end="")
        print()


# find the starting point of the maze
def find_start(maze):
    for row in range(len(maze)):
        for col in range(len(maze[0])):
            if maze[row][col] == "P":
                return row, col


# find the end point of the maze
def find_endpoint(maze):
    for row in range(len(maze)):
        for col in range(len(maze[0])):
            if maze[row][col] == ".":
                return row, col
```

```
# helper method to check for valid position
def is_valid_position(maze, pos_r, pos_c):
    if pos_r < 0 or pos_c < 0:
        return False
    if pos_r >= len(maze) or pos_c >= len(maze[0]):
        return False
    if maze[pos_r][pos_c] in " .":  # adding a space before the . got this
to work
        return True
    return False

# method to solve an input maze via a stack ( dfs search)
def solve_maze_dfs(maze, start):
    # use python list as stack
    stack = []
    numNodes = 0
    fringe = 0
    # add entry point (as tuple)
    stack.append(start)

    # go through stack as long as there are elements
    while len(stack) > 0:
        nodesExpanded = 0
        nodesExpanded = len(stack) + nodesExpanded
        pos_r, pos_c = stack.pop()  # pop out coordinates from the stack
of tuples
        fringe += 1
        print("Current Position", pos_r, pos_c)

        if maze[pos_r][pos_c] == ".":  # end of maze reached
            print("Maze end reached")
            print("Path cost: " + str(numNodes))
            print("Nodes Expanded: " + str(nodesExpanded))
            print("Maximum Size of Fringe: " + str(fringe))
            return True

        if maze[pos_r][pos_c] == "+":
            # already visited location
```

```
            continue
        fringe += 1
        # mark position as visited
        maze[pos_r][pos_c] = "+"
        numNodes += 1
        # check for all possible positions to move to, and add to stack if
possible
        if is_valid_position(maze, pos_r - 1, pos_c):  # check left
            stack.append((pos_r - 1, pos_c))
            fringe += 1
        if is_valid_position(maze, pos_r + 1, pos_c):  # check right
            stack.append((pos_r + 1, pos_c))
            fringe += 1
        if is_valid_position(maze, pos_r, pos_c - 1):  # check down
            stack.append((pos_r, pos_c - 1))
            fringe += 1
        if is_valid_position(maze, pos_r, pos_c + 1):  # check up
            stack.append((pos_r, pos_c + 1))
            fringe += 1

        # to follow the maze

        print("Stack:", stack)
        print_maze(maze)
        fringe += 1
    # path not found
    return False


# BFS search method for maze
def solve_maze_bfs(maze, start):

    numNodes = 0
    fringe = 0
    # create a queue for use in bfs
    queuebfs = queue.Queue()
    nodesExpanded = 0
    nodesExpanded = len(queuebfs) + nodesExpanded
    queuebfs.put(start)  # put the starting position in the queue
```

```
    while not (queuebfs.empty()):

        pos_r, pos_c = queuebfs.get()
        fringe += 1
        print("Current Position", pos_r, pos_c)

        if maze[pos_r][pos_c] == ".":  # end of maze reached
            print("Maze end reached!")
            print("Path cost: " + str(numNodes))
            print("Nodes Expanded: " + str(nodesExpanded))
            print("Maximum Size of Fringe: " + str(fringe))
            return True

        if maze[pos_r][pos_c] == "+":
            # already visited location
            continue

        # mark current position as visited
        maze[pos_r][pos_c] = "+"
        numNodes += 1
        # check for all possible positions to move to, and add to queue if
possible
        if is_valid_position(maze, pos_r - 1, pos_c):  # check left
            queuebfs.put((pos_r - 1, pos_c))
            fringe += 1
        if is_valid_position(maze, pos_r + 1, pos_c):  # check right
            queuebfs.put((pos_r + 1, pos_c))
            fringe += 1
        if is_valid_position(maze, pos_r, pos_c - 1):  # check up
            queuebfs.put((pos_r, pos_c - 1))
            fringe += 1
        if is_valid_position(maze, pos_r, pos_c + 1):  # check down
            queuebfs.put((pos_r, pos_c + 1))
            fringe += 1

        print("Current positions in queue:", queuebfs.qsize())
        print_maze(maze)

        # path not found
    return False
```

```python
# Main code
if __name__ == "__main__":
    print("Project 2")

    # load in the maze
    maze = load_maze("smallMaze.lay")
    maze = convert_maze(maze)
    print_maze(maze)

    # find starting point of the maze
    start = find_start(maze)
    print("Starting coordinates: (Row, Column)  ** zero start index **")
    print(start)

    # find end point of the maze
    endPoint = find_endpoint(maze)
    print("End coordinates: (Row, Column)  ** zero start index **")
    print(endPoint)

    # solve the maze
    print(solve_maze(maze, start))
```

Implementation

BFS:
For breadth first search, the path cost was 19, about half of the cost of depth first search. The program loops through a queue and checks each side to see if there are valid positions for the player to move to. Each node that the player moves to is marked with “+” and an if statement was used within the while loop to return true, ending the loop with a message to the user. If the goal was not met, all positions would continue to be exhausted until no other positions are valid to move to.

DFS:
For depth first search, the path cost was 37, which was about twice as much as breadth first search. A stack queue was used as opposed to the graph class provided. Through

this stack, the program would run in one direction and keep following down this path until is_valid_position returns false, which means that the program needs to backtrack to its parent nodes until it finds a different child node to follow.

***Maximum nodes expanded is off, we tried to add the length of the queue or stack as an integer and add that value everytime a new stack was added to the old one but weren't able to.