Assignment #1: Search Strategies COP4630

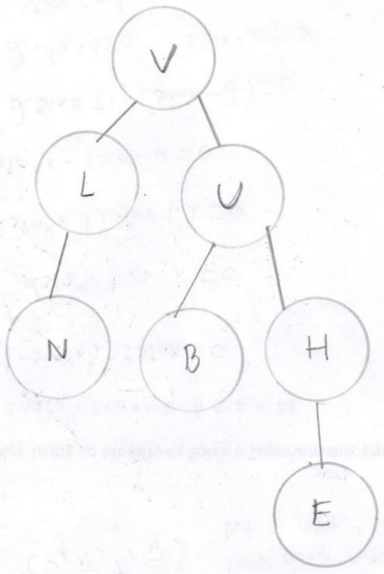
Jonathan Masino, Ava Perrone, Steven Fernandez

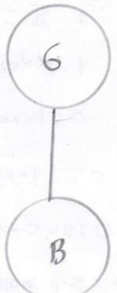
Overview

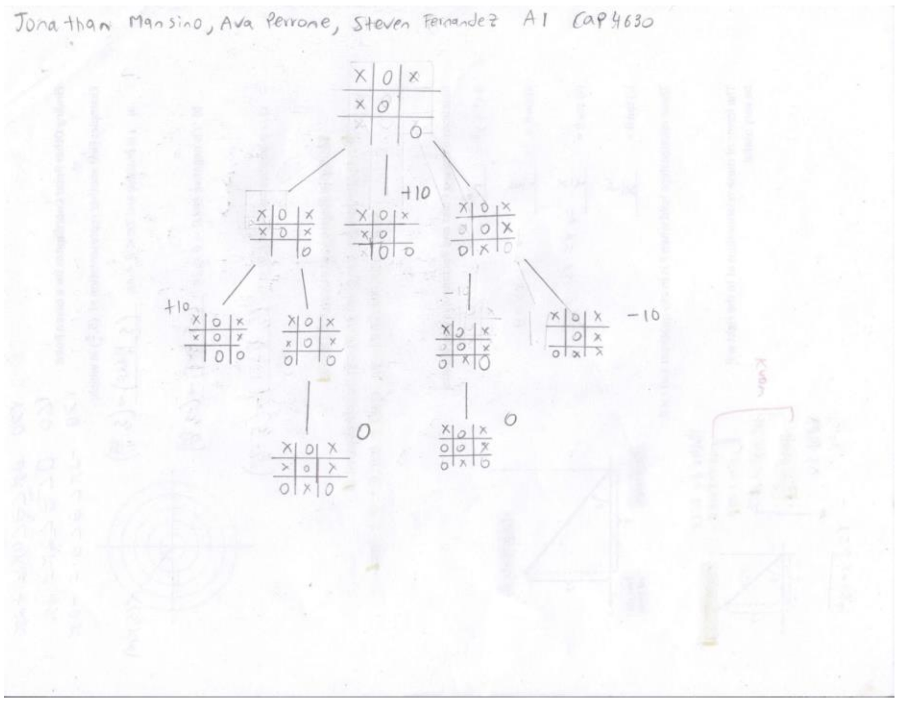
We formed our group days after A1 was released, we have taken some classes in the past, so it was easy for us to assign roles based on that. We read the book and watch the different videos professor Marques has posted on YouTube where he explains each chapter and the code related to them. Our professor videos showing different hints for A1 were very helpful and gave us an idea on how to start this assignment.

We start as suggested on the videos with some graphs with different trials and errors on the way. Here are some of the graphs we did to understand this assignment (they might not be correct). We all educated ourselves on the three different algorithms that we needed prior to taking on this project.

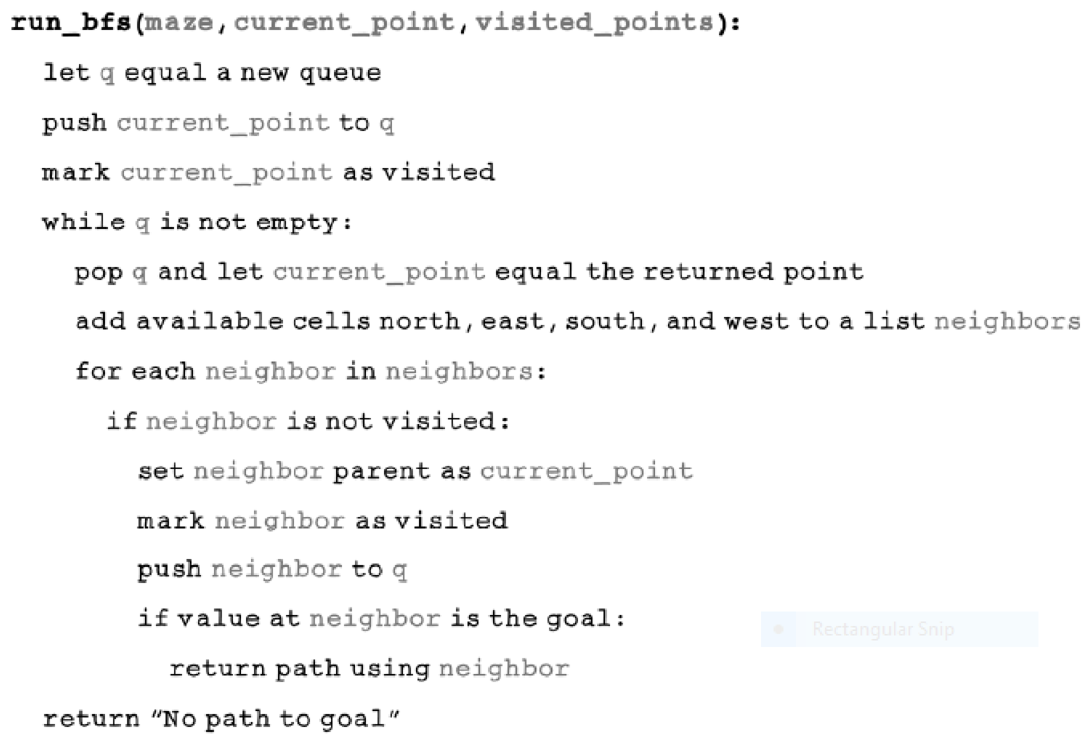
Jonathan Masino, Ava Perrone, Steven Fernandez AI
CAP4630







We also looked at the pseudocode that is in the book to plan and format the code for this assignment.



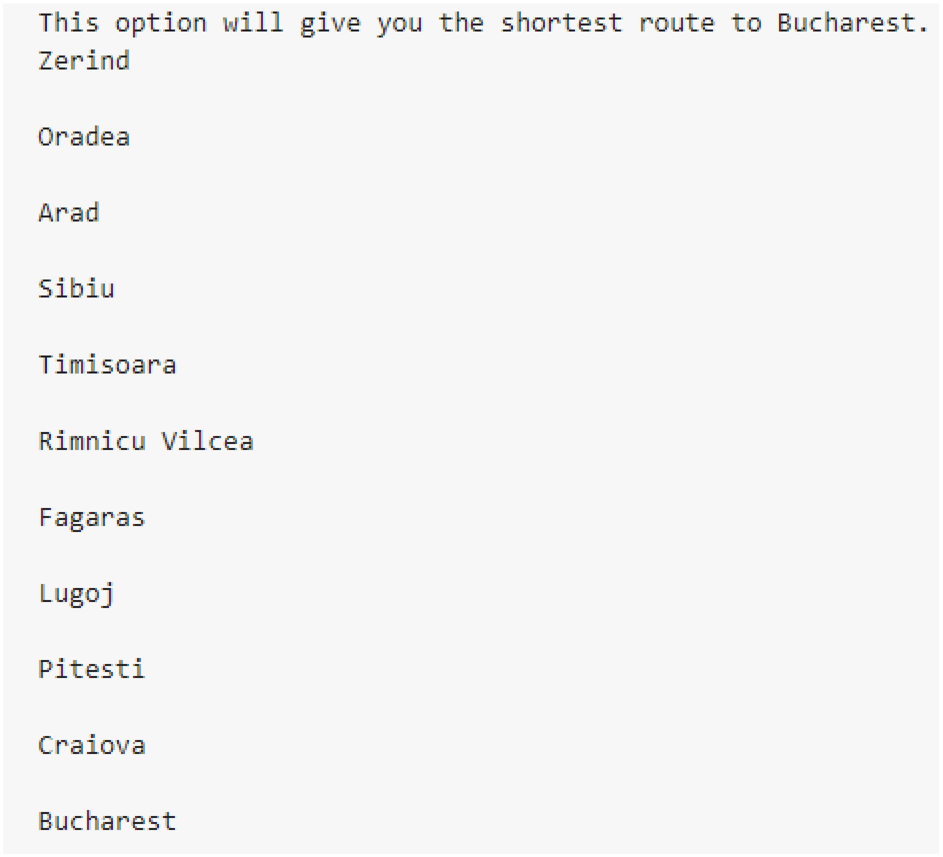
```
run_bfs(maze, current_point, visited_points):
  let q equal a new queue
  push current_point to q
  mark current_point as visited
  while q is not empty:
    pop q and let current_point equal the returned point
    add available cells north, east, south, and west to a list neighbors
    for each neighbor in neighbors:
      if neighbor is not visited:
        set neighbor parent as current_point
        mark neighbor as visited
        push neighbor to q
        if value at neighbor is the goal:
          return path using neighbor
  return "No path to goal"
```

Image taken from the book *Artificial Intelligence Algorithms**

Coding of the solution

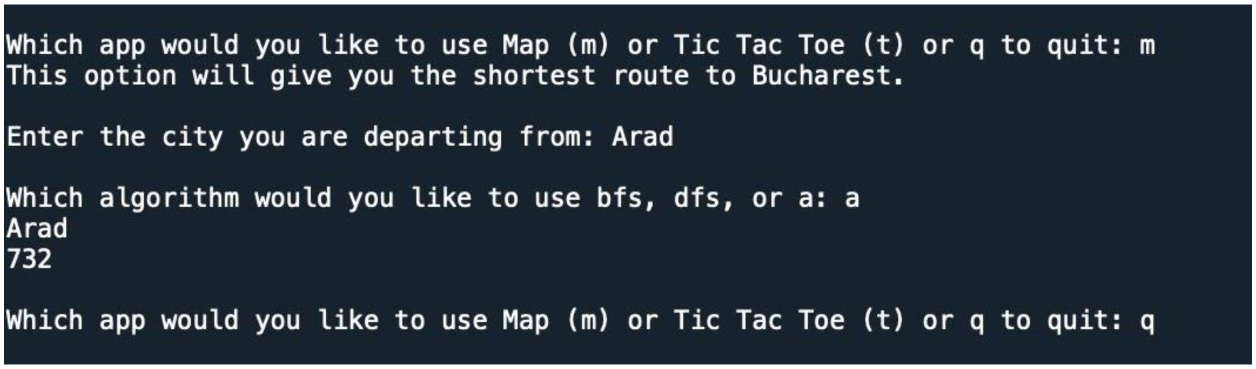
Shortest route code

We started off by learning and fully understanding the algorithms that we are using for the assignment. We started by making two graphs, one containing the cities and their neighbors, and the other containing the cities and with neighbors, the distances between the cities, as well as the straight-line distance. The Breadth First Search was the easiest to understand and implement. We had some small issues with the Depth First Search but were eventually able to debug and get it working. The A* algorithm was the hardest. Understanding this algorithm was easy but implementing it into our program was a challenge. However, through trial and error, we were able to get it working.



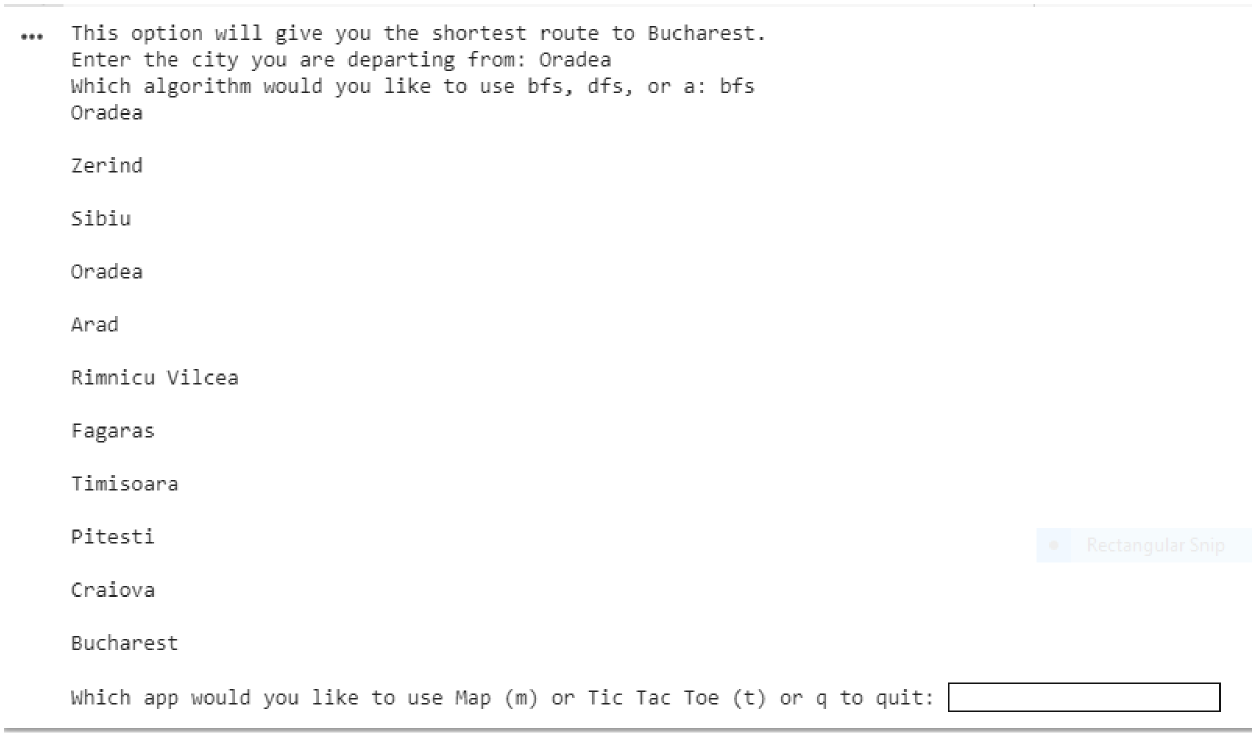
```
This option will give you the shortest route to Bucharest.
Zerind

Oradea

Arad

Sibiu

Timisoara

Rimnicu Vilcea

Fagaras

Lugoj

Pitesti

Craiova

Bucharest
```

One of the first outputs obtained by the code in earlier stages.



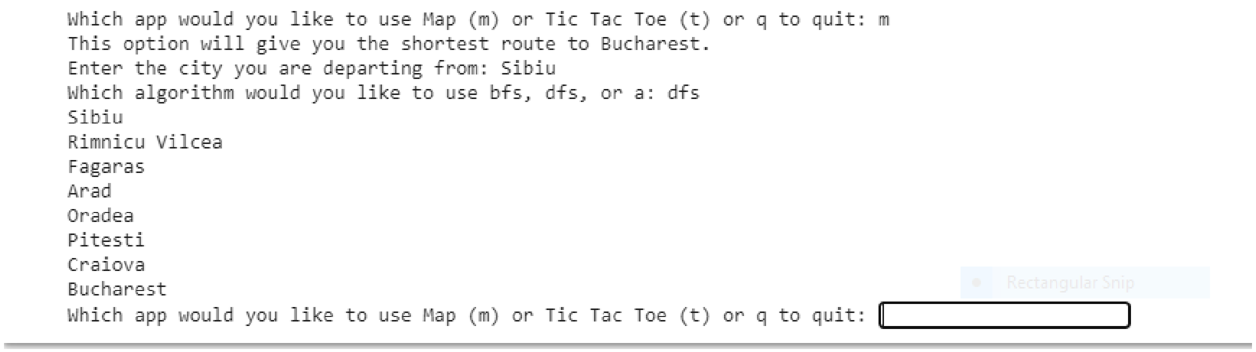
```
Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit: m
This option will give you the shortest route to Bucharest.

Enter the city you are departing from: Arad

Which algorithm would you like to use bfs, dfs, or a: a
Arad
732

Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit: q
```

A* algorithm



```
This option will give you the shortest route to Bucharest.
Enter the city you are departing from: Oradea
Which algorithm would you like to use bfs, dfs, or a: bfs
Oradea

Zerind

Sibiu

Oradea

Arad

Rimnicu Vilcea

Fagaras

Timisoara

Pitesti

Craiova

Bucharest

Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit:
```

BFS algorithm

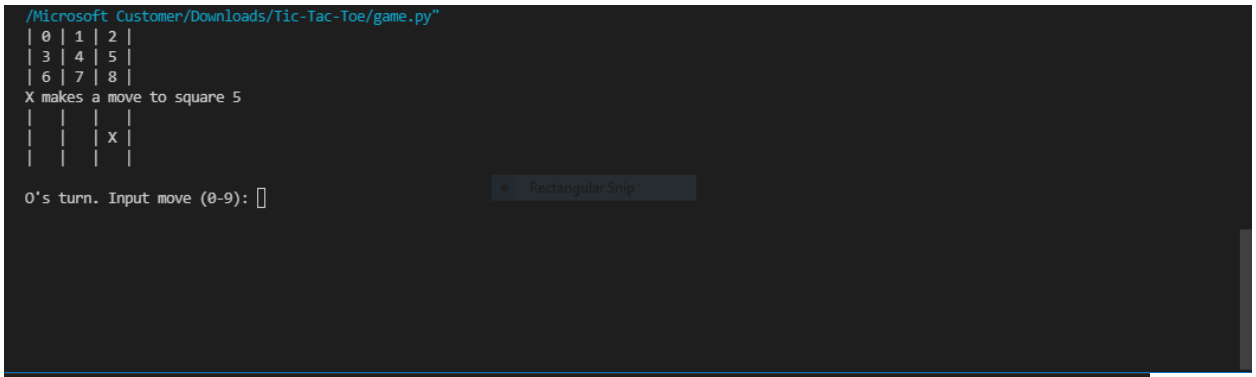


```
Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit: m
This option will give you the shortest route to Bucharest.
Enter the city you are departing from: Sibiu
Which algorithm would you like to use bfs, dfs, or a: dfs
Sibiu
Rimnicu Vilcea
Fagaras
Arad
Oradea
Pitesti
Craiova
Bucharest
Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit:
```

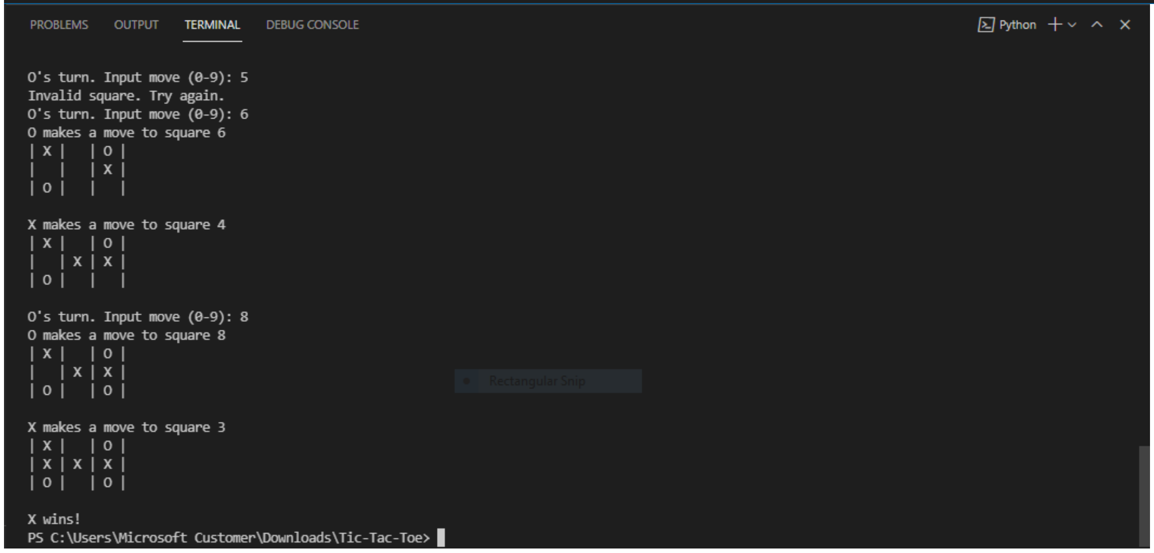
DFS algorithm

Tic-Tac-Toe code

After watching the *video*** suggested by professor Marques for the Tic-Tac-Toe part of the assignment we start working on the code section of it based on the design previously shown. This part was challenging because the MinMax algorithm tries to make as many possible outcomes for the player, reducing the chances of victory for the opponent, in this game X represents the machine and O represents the user's input. This algorithm represent various challenges but with the help of the *video*** linked by professor Marques it was way easier to understand and implement. Other *videos**** helped with the structure and solution of this part of the assignment.



```
/Microsoft Customer/Downloads/Tic-Tac-Toe/game.py"
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |
X makes a move to square 5
|   |   |   |
|   |   | X |
|   |   |   |

O's turn. Input move (0-9): 
```



```
O's turn. Input move (0-9): 5
Invalid square. Try again.
O's turn. Input move (0-9): 6
O makes a move to square 6
| X |   | O |
|   |   | X |
| O |   |   |

X makes a move to square 4
| X |   | O |
|   | X | X |
| O |   |   |

O's turn. Input move (0-9): 8
O makes a move to square 8
| X |   | O |
|   | X | X |
| O |   | O |

X makes a move to square 3
| X |   | O |
| X | X | X |
| O |   | O |

X wins!
PS C:\Users\Microsoft Customer\Downloads\Tic-Tac-Toe>
```

Documentation of the solution

We worked together as a team and individually on this assignment. We broke up the coding parts among us to balance the workload. Ava created graph1 and Jonathan created graph2. Ava worked on the DFS function while Jonathan worked on the DFS function. Both Jonathan and Ava worked together on A* and the main function code. Steven's task was tic-tac-toe. By dividing the work this way, we were able to complete this project in a timely manner. This assignment took us about 4 days to complete.

Project Notes

Part 1

BFS:

- Add starting node to queue and stack
- While in the queue:
- Pop the top node and assign it to ‘current’
- Print ‘current’
- If the current node is the goal, break
- For the neighbor in the graph,
- If the neighbor hasn’t been visited, add it to ‘visited’ and add it to the queue

DFS:

- Add the starting node to the stack
- While in the stack:
- Pop the current node
- If the node hasn’t been visited,
- Print the node
- Add the node to visited
- If that node is the goal, break
- Otherwise, for neighbor in current node:
- Current becomes the neighbor and it is pushed to the stack

A*

- Push starting node to stack
- Create came_from
- Create costSoFar
- Set costSoFar to 0
- Set newCost to 0
- While in the stack:
- Pop current node
- If that node is not yet visited:
- Print it
- Add that node to visited
- If current node is goal, break
- Otherwise visit the neighbors
- Set new cost using a* formula
- If next is not in costSoFar or new cost is less that costSoFar
- costSoFar of next = newCost
- Set priority using formula
- Print priority
- Came_from of next = current
- costSoFar of current added to newCost

Part 2

Tic-tac-toe

- Prompt user for map or tic-tac-toe
- If they choose map:
- Prompt for city name
- Prompt for algorithm
- Run the algorithm function based on the one they choose

- If they choose tic-tac-toe:
- Run the tic-tac-toe function

References

*Rishal Hurbans. (2020). *Grokking artificial intelligence algorithms*. Manning.

***12 Beginner Python Projects - Coding Course*. (n.d.). Www.youtube.com. Retrieved June 11, 2021, from https://www.youtube.com/watch?v=8ext9G7xspg

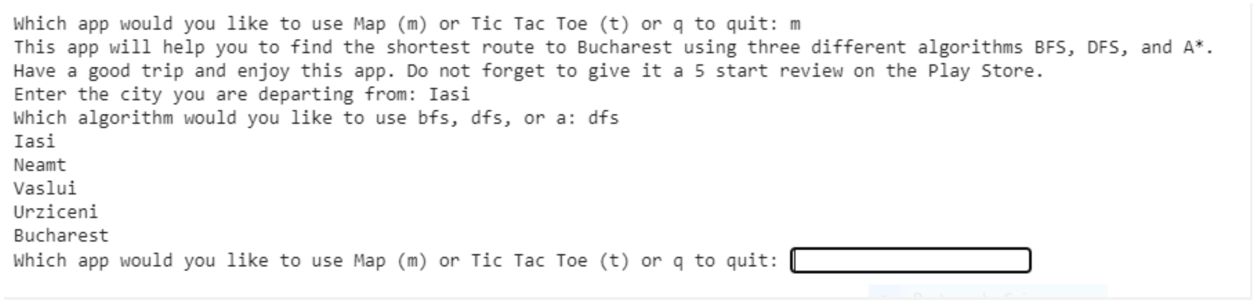
****Coding Challenge 154: Tic Tac Toe AI with Minimax Algorithm - YouTube*. (n.d.). Www.youtube.com. https://www.youtube.com/watch?v=trKjYdBASyQ

****Tic Tac Toe AI with MiniMax using Python | Part 1: Programming Tic Tac Toe*. (n.d.). Www.youtube.com. Retrieved June 11, 2021, from https://www.youtube.com/watch?v=JC1QsLOXp-I

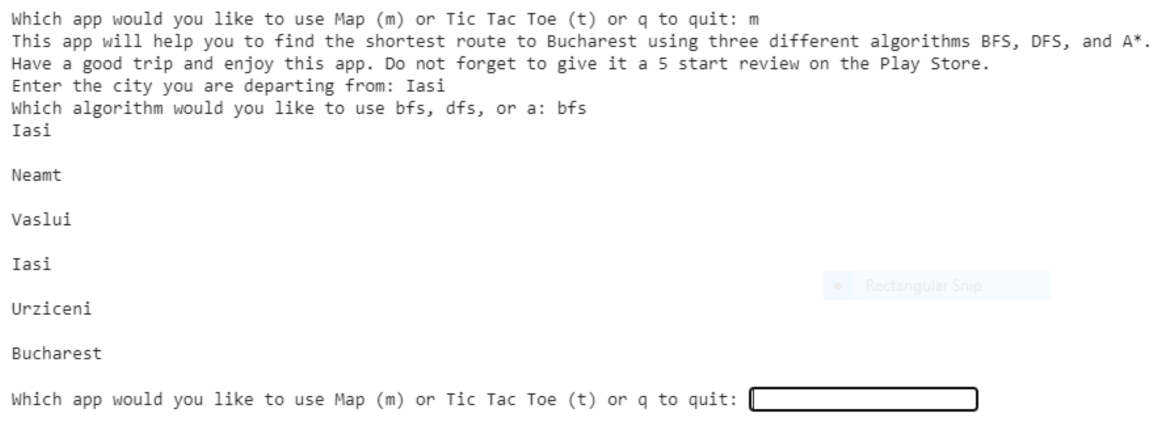
Kylie. (2021, June 7). *kying18/tic-tac-toe*. GitHub. https://github.com/kying18/tic-tac-toe

Screenshots of representative results

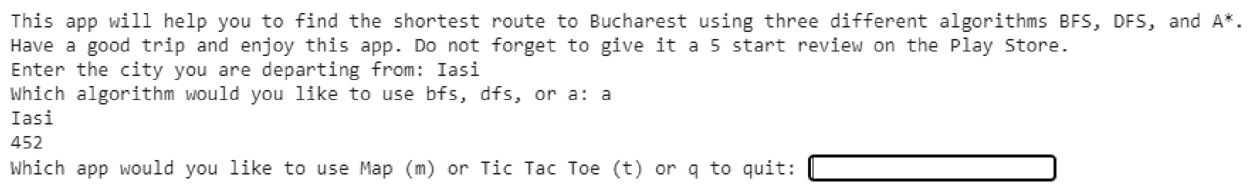
Iasi



```
Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit: m
This app will help you to find the shortest route to Bucharest using three different algorithms BFS, DFS, and A*.
Have a good trip and enjoy this app. Do not forget to give it a 5 start review on the Play Store.
Enter the city you are departing from: Iasi
Which algorithm would you like to use bfs, dfs, or a: dfs
Iasi
Neamt
Vaslui
Urziceni
Bucharest
Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit:
```

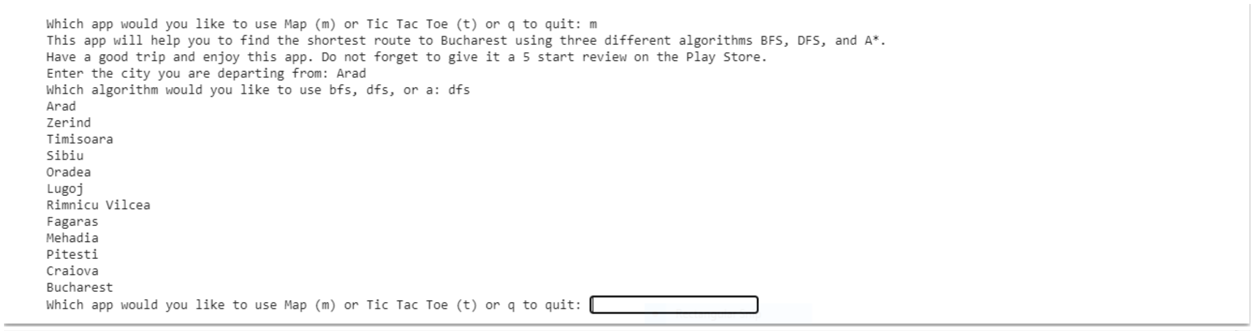


```
Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit: m
This app will help you to find the shortest route to Bucharest using three different algorithms BFS, DFS, and A*.
Have a good trip and enjoy this app. Do not forget to give it a 5 start review on the Play Store.
Enter the city you are departing from: Iasi
Which algorithm would you like to use bfs, dfs, or a: bfs
Iasi

Neamt

Vaslui

Iasi

Urziceni

Bucharest

Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit:
```

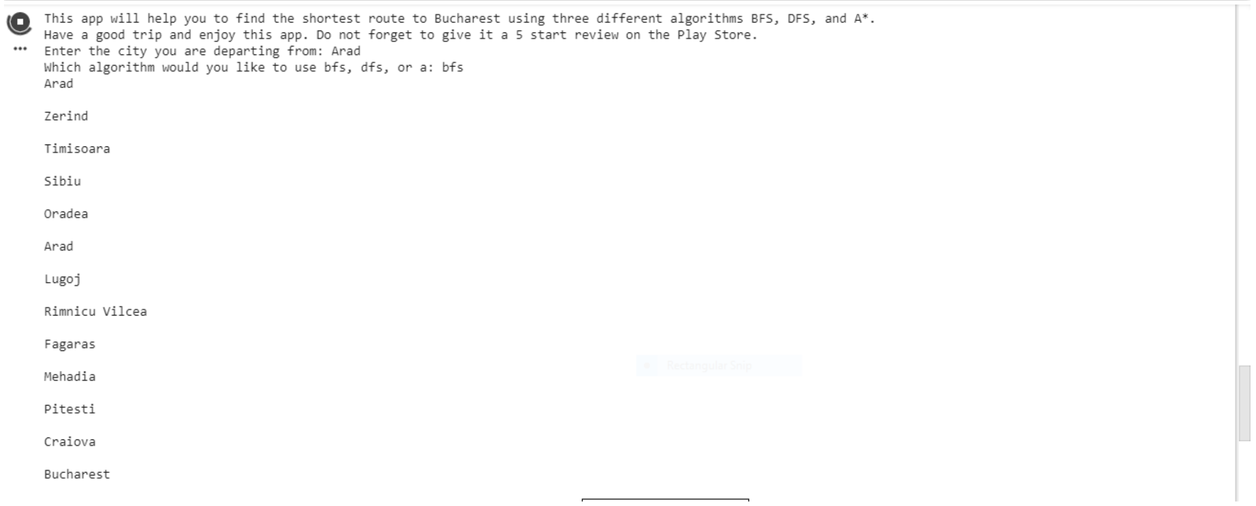


```
This app will help you to find the shortest route to Bucharest using three different algorithms BFS, DFS, and A*.
Have a good trip and enjoy this app. Do not forget to give it a 5 start review on the Play Store.
Enter the city you are departing from: Iasi
Which algorithm would you like to use bfs, dfs, or a: a
Iasi
452
Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit:
```

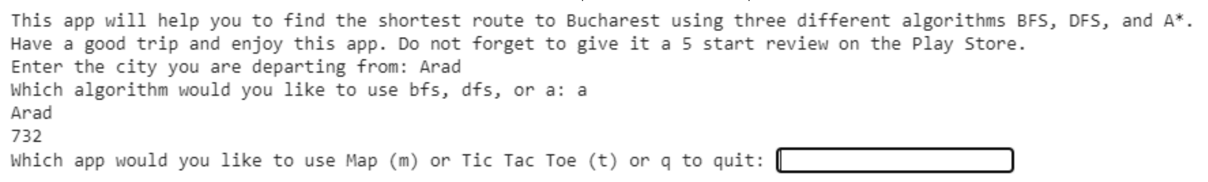
Arad



```
Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit: m
This app will help you to find the shortest route to Bucharest using three different algorithms BFS, DFS, and A*.
Have a good trip and enjoy this app. Do not forget to give it a 5 start review on the Play Store.
Enter the city you are departing from: Arad
Which algorithm would you like to use bfs, dfs, or a: dfs
Arad
Zerind
Timisoara
Sibiu
Oradea
Lugoj
Rimnicu Vilcea
Fagaras
Mehadia
Pitesti
Craiova
Bucharest
Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit:
```

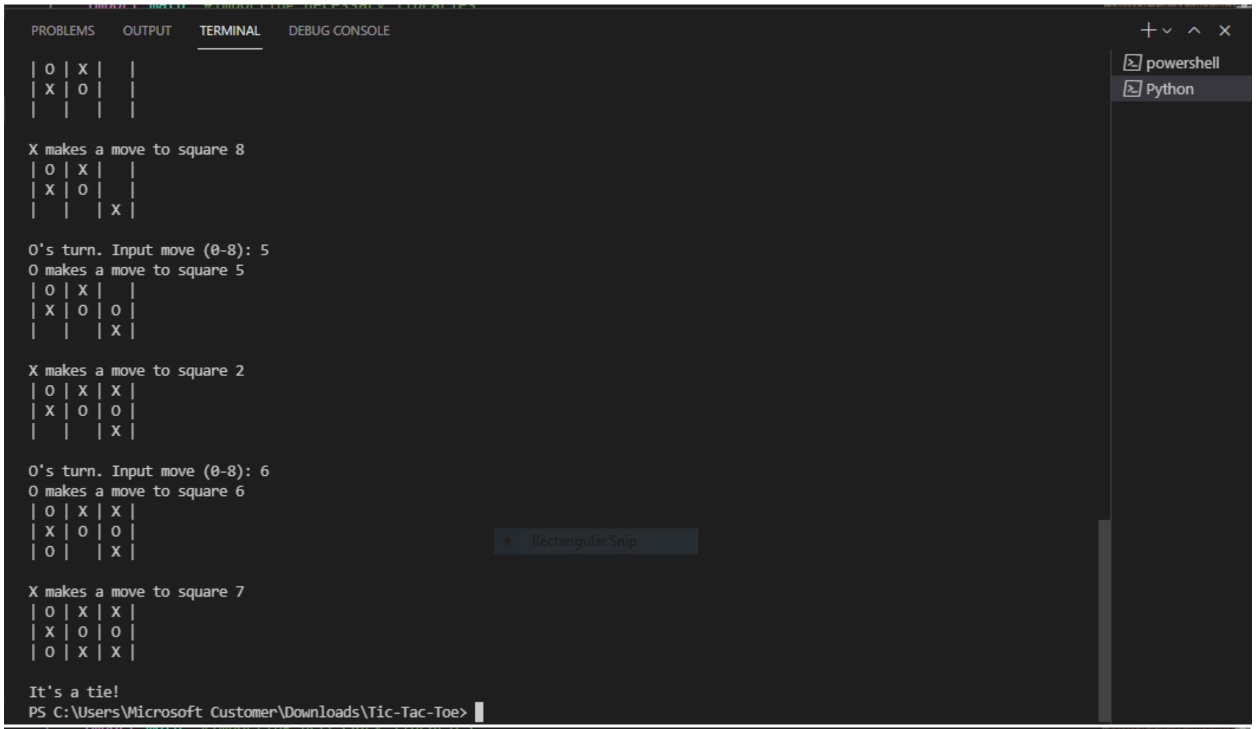


```
This app will help you to find the shortest route to Bucharest using three different algorithms BFS, DFS, and A*.
Have a good trip and enjoy this app. Do not forget to give it a 5 start review on the Play Store.
Enter the city you are departing from: Arad
Which algorithm would you like to use bfs, dfs, or a: bfs
Arad

Zerind

Timisoara

Sibiu

Oradea

Arad

Lugoj

Rimnicu Vilcea

Fagaras

Mehadia

Pitesti

Craiova

Bucharest
```

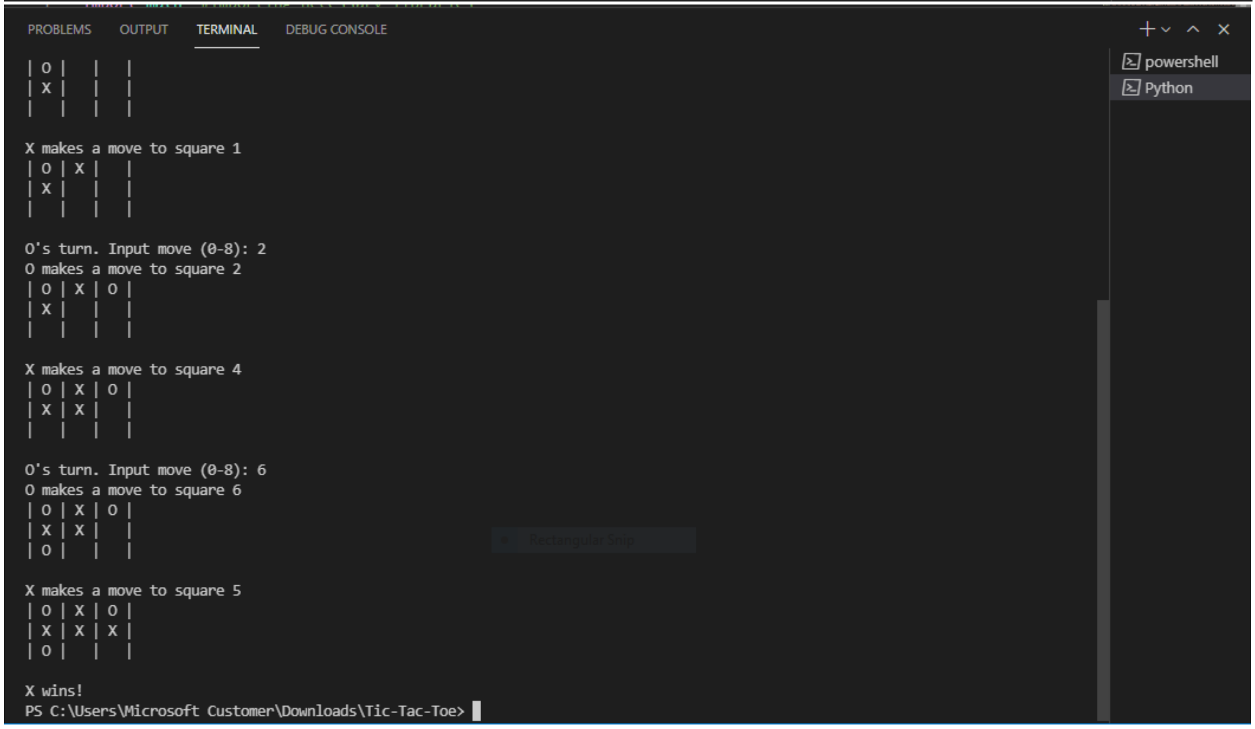


```
This app will help you to find the shortest route to Bucharest using three different algorithms BFS, DFS, and A*.
Have a good trip and enjoy this app. Do not forget to give it a 5 start review on the Play Store.
Enter the city you are departing from: Arad
Which algorithm would you like to use bfs, dfs, or a: a
Arad
732
Which app would you like to use Map (m) or Tic Tac Toe (t) or q to quit:
```

Tic-Tac-Toe



```
| O | X |   |
| X | O |   |
|   |   |   |

X makes a move to square 8
| O | X |   |
| X | O |   |
|   |   | X |

O's turn. Input move (0-8): 5
O makes a move to square 5
| O | X |   |
| X | O | O |
|   |   | X |

X makes a move to square 2
| O | X | X |
| X | O | O |
|   |   | X |

O's turn. Input move (0-8): 6
O makes a move to square 6
| O | X | X |
| X | O | O |
| O |   | X |

X makes a move to square 7
| O | X | X |
| X | O | O |
| O | X | X |

It's a tie!
PS C:\Users\Microsoft Customer\Downloads\Tic-Tac-Toe>
```



```
| O |   |   |
| X |   |   |
|   |   |   |

X makes a move to square 1
| O | X |   |
| X |   |   |
|   |   |   |

O's turn. Input move (0-8): 2
O makes a move to square 2
| O | X | O |
| X |   |   |
|   |   |   |

X makes a move to square 4
| O | X | O |
| X | X |   |
|   |   |   |

O's turn. Input move (0-8): 6
O makes a move to square 6
| O | X | O |
| X | X |   |
| O |   |   |

X makes a move to square 5
| O | X | O |
| X | X | X |
| O |   |   |

X wins!
PS C:\Users\Microsoft Customer\Downloads\Tic-Tac-Toe>
```