Assignment 1

Team members : Peterling Etienne, Tyler Nguyen, Bryan Perdomo

Part 1: Shortest Route

Project Notes:

Credits:
This implementation to solve the shortest route problem to the assignment specifications was heavily inspired by the provided source code in the official AIMA github repository posted in the Canvas Module 2 resources, and a few Github users who contributed to the Official AIMA repo. All repositories I took inspiration from are linked below:
https://github.com/aimacode/aima-python/blob/master/search.py
https://github.com/kevinrosalesdev/Searching-Algorithms
https://github.com/Addeuz/aima

Each repository provided extremely valuable insight and a robust skeleton from which I could not only implement A1, but also served as an enriching resource to learn about practical implementations of search algorithms. I can't thank these resources enough and all the credit goes to all of the wonderful people who contributed to the AIMA repo.

Process/Learning
I tackled Part 1 by trying to understand the provided source code, as the professor advised it's a good idea to start from there. I quickly realized how complex the source code's implementation was, and was pretty intimidated. The source repo included advanced data structures, cutting-edge python methods/operations, and advanced use of object oriented programming including inheritance.

So I researched and spent several days understanding how: Each searching algorithm was implemented, understanding the utility functions necessary to the implementation, tracing how each algorithm solves the problem step-by-step, figuring out how to keep track of the city's distances, and calling them to display them to the screen.

Processes and Obstacles/Hurdles:
Quite honestly, the biggest obstacle wasn't the conceptual or practical implementation of the search algorithms, but rather having to catch
up on my python knowledge to have the ability to adequately understand the source code and adapt it to what I wanted to solve.
Understanding how the search algorithm implementations were encapsulated using OOP to create general purpose and robust solutions to

several kinds of search problems.
This assignment definitely exposed my python weaknesses and I feel better off for it.

```
This is the first part of Assignment 1, where we implement 3 searching algorithms to solve the shortest route problem using a map of Romania.
2 are uninformed searching algorithms: Breadth-First-Search, and Depth-First-Search.
The last is an informed searching algorithm, namely A* which makes use of a heuristic.
Would you like to run a search algorithm? Answer Y or N:y
Choose which algo to run: B/D/A* B
Please enter starting city:Arad
Running search algorithm for Bucharest...

--- Breadth First Search ---
BFS route:
City and its' distance from root city:  Zerind : 75
City and its' distance from root city:  Sibiu : 140
City and its' distance from root city:  Timisoara : 118
City and its' distance from root city:  Oradea : 146
City and its' distance from root city:  Fagaras : 239
City and its' distance from root city:  Rimnicu : 220
City and its' distance from root city:  Lugoj : 229
Itinerary:  ['Sibiu', 'Fagaras', 'Bucharest']
Total Distance to Bucharest: 450

Would you like to run another algorithm? Enter Y or N:y
Choose which algo to run: B/D/A* B
Please enter starting city:Dobreta
Running search algorithm for Bucharest...

--- Breadth First Search ---
BFS route:
City and its' distance from root city:  Mehadia : 75
City and its' distance from root city:  Craiova : 120
City and its' distance from root city:  Lugoj : 145
City and its' distance from root city:  Rimnicu : 266
City and its' distance from root city:  Pitesti : 258
City and its' distance from root city:  Timisoara : 256
City and its' distance from root city:  Sibiu : 346
Itinerary:  ['Craiova', 'Pitesti', 'Bucharest']
Total Distance to Bucharest: 359
```

```
Would you like to run another algorithm? Enter Y or N:y
Choose which algo to run: B/D/A* D
Please enter starting city:Arad
Running search algorithm for Bucharest...


--- Depth First Search ---
DFS Route:
Current city, and distance traveled so far: Arad 0
Current city, and distance traveled so far: Timisoara 118
Current city, and distance traveled so far: Lugoj 229
Current city, and distance traveled so far: Mehadia 299
Current city, and distance traveled so far: Dobreta 374
Current city, and distance traveled so far: Craiova 494
Current city, and distance traveled so far: Pitesti 632
Itinerary:  ['Timisoara', 'Lugoj', 'Mehadia', 'Dobreta', 'Craiova', 'Pitesti', 'Bucharest']
Total Distance to Bucharest: 733

Would you like to run another algorithm? Enter Y or N:y
Choose which algo to run: B/D/A* D
Please enter starting city:Dobreta
Running search algorithm for Bucharest...


--- Depth First Search ---
DFS Route:
Current city, and distance traveled so far: Dobreta 0
Current city, and distance traveled so far: Craiova 120
Current city, and distance traveled so far: Pitesti 258
Itinerary:  ['Craiova', 'Pitesti', 'Bucharest']
Total Distance to Bucharest: 359

Would you like to run another algorithm? Enter Y or N:y
Choose which algo to run: B/D/A* A*
Please enter starting city:Arad
Running search algorithm for Bucharest...
```

```
Would you like to run another algorithm? Enter Y or N:y
Choose which algo to run: B/D/A* A*
Please enter starting city:Arad
Running search algorithm for Bucharest...


--- A* Search: ---
A* Route:
Current city, and distance traveled so far: Arad 0
Current city, and distance traveled so far: Sibiu 140
Current city, and distance traveled so far: Fagaras 239
Current city, and distance traveled so far: Rimnicu 220
Current city, and distance traveled so far: Pitesti 317
Itinerary:  ['Sibiu', 'Rimnicu', 'Pitesti', 'Bucharest']
Total Distance to Bucharest: 418

Would you like to run another algorithm? Enter Y or N:y
Choose which algo to run: B/D/A* A*
Please enter starting city:Dobreta
Running search algorithm for Bucharest...


--- A* Search: ---
A* Route:
Current city, and distance traveled so far: Dobreta 0
Current city, and distance traveled so far: Craiova 120
Current city, and distance traveled so far: Mehadia 75
Current city, and distance traveled so far: Pitesti 258
Itinerary:  ['Craiova', 'Pitesti', 'Bucharest']
Total Distance to Bucharest: 359

Would you like to run another algorithm? Enter Y or N:n
PS C:\Users\bbper\Intro to AI>
```

Part 2: Tic Tac Toe game

Credits: Kylie Ying YouTube https://youtu.be/8ext9G7xspg

This implementation uses MinMax algorithm to play tic tac toe game versus human player. By calculating the best scores before playing a move, it will try to max out the chance to win, and minimize the loss.

Learning Process: I tried to follow the video and typing out the code to understand. Kylie King is really good at explaining the code and I learnt a lot from her. Sometimes, when I feel she went too fast, I would pause and try to re-read the code or replay to get a better understanding of what she meant. She shared her code on github, but I did try my best to type it out to follow instead.

Output 1:

```
This TicTacToe game AI robot vs player will use MinMax algorithm to perform an adversarial search
which will find the best stragery to play 3x3 tictactoe game,
it will try to find the best way to win or tie a game
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |
X's turn. Input move (0-8): 4
X makes a move to square 4
|   |   |   |
|   | X |   |
|   |   |   |

O makes a move to square 0
| O |   |   |
|   | X |   |
|   |   |   |

X's turn. Input move (0-8): 1
X makes a move to square 1
| O | X |   |
|   | X |   |
|   |   |   |

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

X's turn. Input move (0-8): 3
X makes a move to square 3
| O | X |   |
| X | X |   |
|   | O |   |

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

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

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

X's turn. Input move (0-8): 6
```

```
X makes a move to square 6
| O | X | O |
| X | X | O |
| X | O | X |

It is a tie!
Do you want to play again? (y/n): n
PS C:\Users\Tyler\Desktop\AI_Hw\part2>
```

Output 2:

```
This TicTacToe game AI robot vs player will use MinMax algorithm to perform an adversarial search
which will find the best stragery to play 3x3 tictactoe game,
it will try to find the best way to win or tie a game
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |
X's turn. Input move (0-8): 4
X makes a move to square 4
|   |   |   |
|   | X |   |
|   |   |   |

O makes a move to square 0
| O |   |   |
|   | X |   |
|   |   |   |

X's turn. Input move (0-8): 1
X makes a move to square 1
| O | X |   |
|   | X |   |
|   |   |   |

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

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

O makes a move to square 6
| O | X | X |
|   | X |   |
| O | O |   |

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

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

O wins!
```