Neel Khiroya

100160178

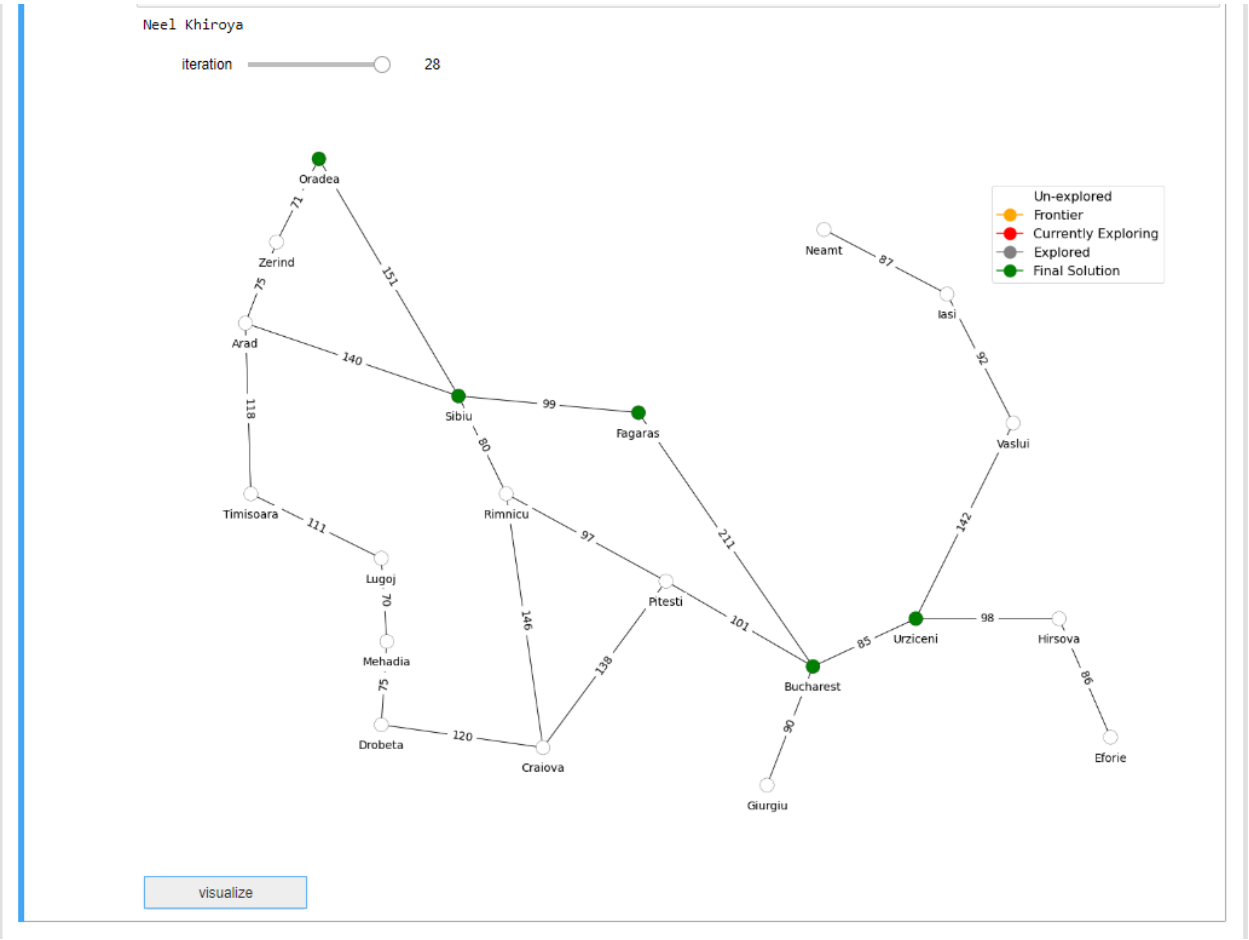
COMP 3613

Nov 10, 2023

Search Algorithms

by Neel Khiroya

1) 3. BREADTH-FIRST GRAPH SEARCH



This code is working properly.

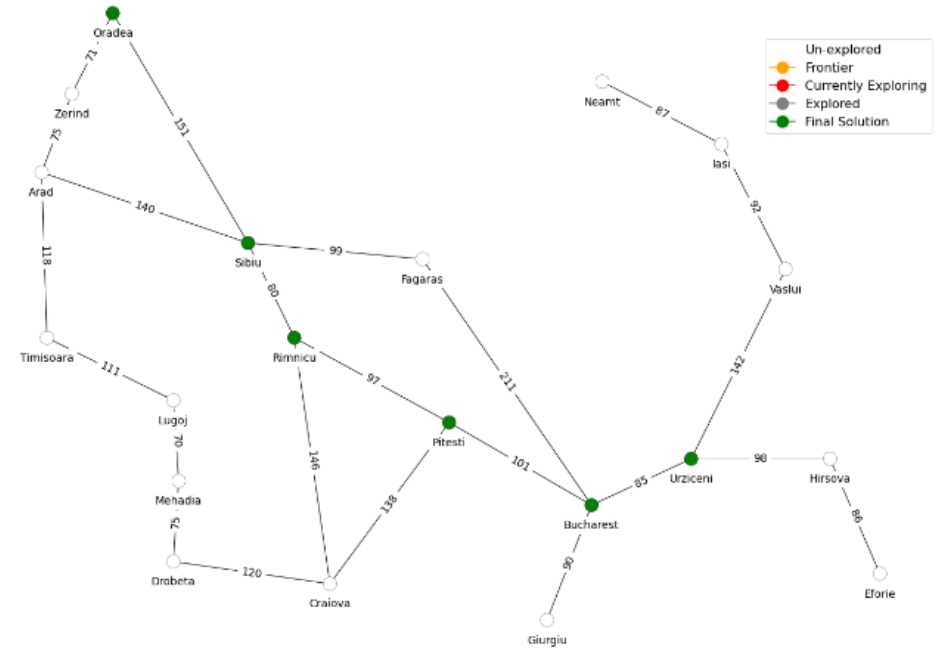
For a Breadth First Search, the Root node is visited first; then it will branch off into all the children's nodes. From there it will then branch off into the children's nodes and then keep track of all the reach nodes. In this search all nodes have the same cost resulting in the cost not being taken into consideration.

The solution is not optimal.

2) 4. DEPTH-FIRST GRAPH SEARCH

```
In [65]: all_node_colors = []
         romania_problem = GraphProblem('Oradea', 'Urziceni', romania_map)
         print("Neel Khiroya")
         display_visual(romania_graph_data, user_input=False,
                        algorithm=depth_first_graph_search,
                        problem=romania_problem)
```

```
Neel Khiroya
iteration  42
```



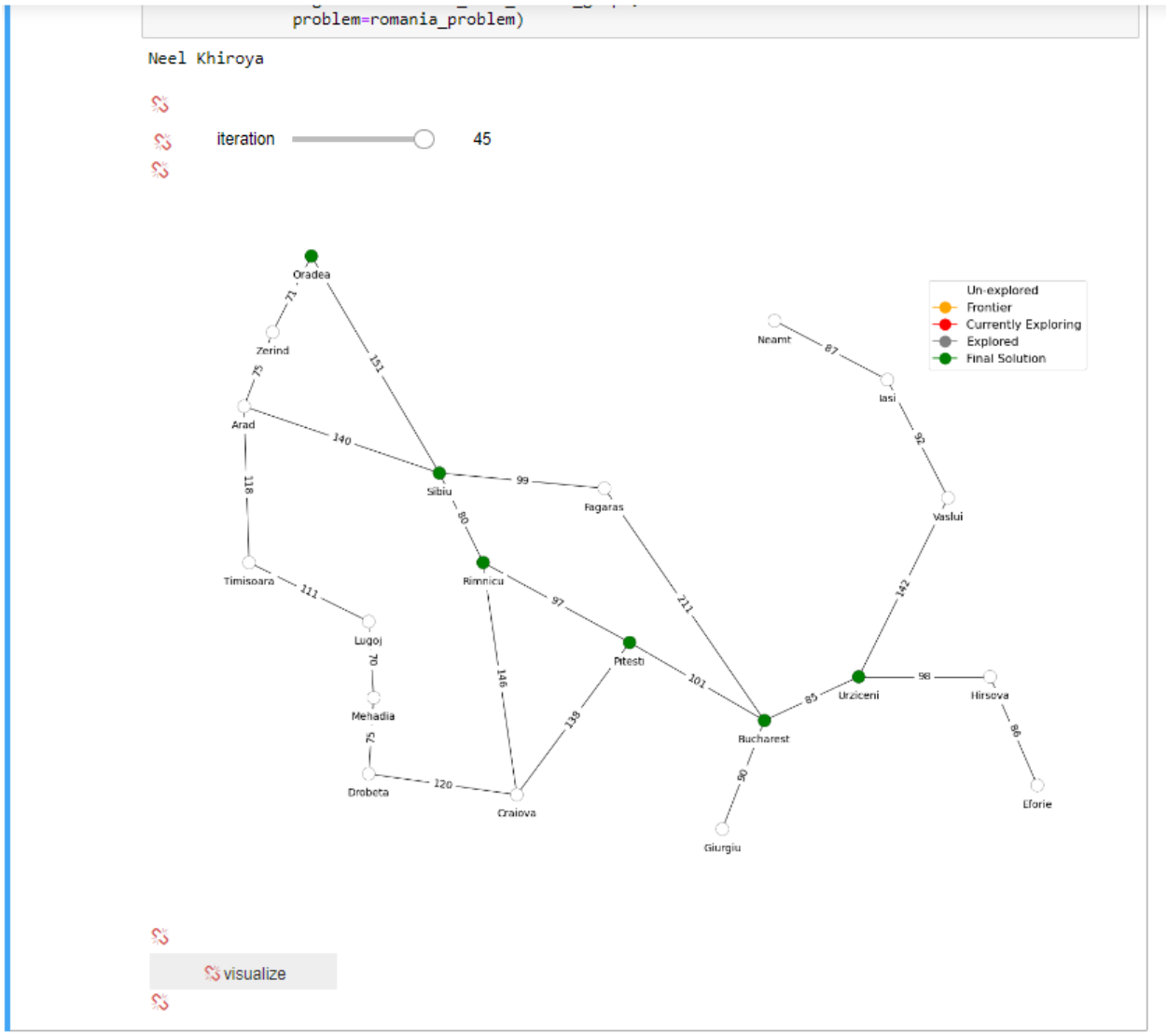
visualize

For this code to work properly we expect it to expand into the deepest node currently accessible by the node that Is being worked on. There is no need to remember all states reached as once it reaches the goal it will stop and will not need to go back to other nodes children.

This code is working properly.

This solution is optimal.

3) 6. UNIFORM COST SEARCH

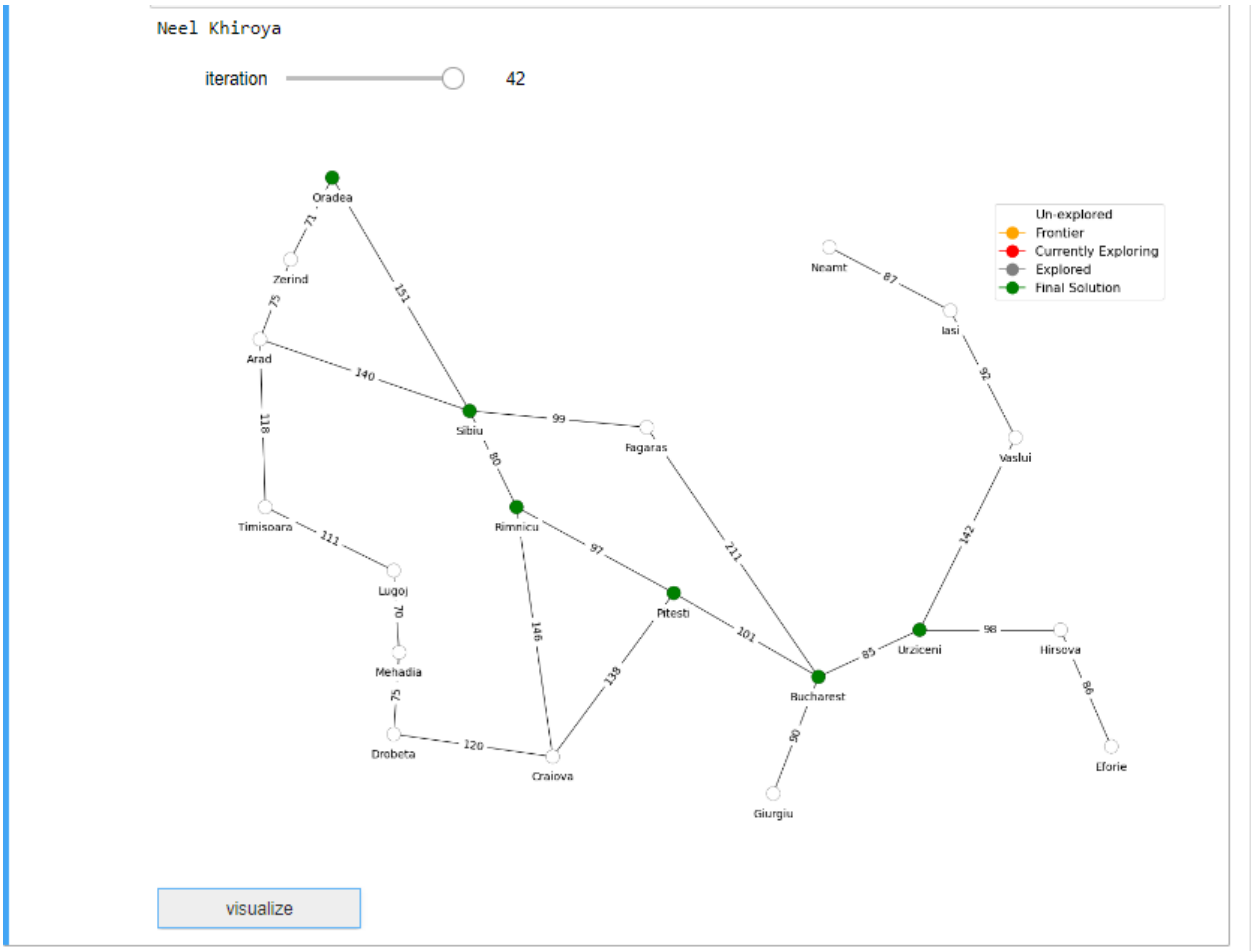


This algorithm is designed to find the least cost path to our goal in our data. This algorithm will expand to the cheapest node first then expanding the cheapest route from there until it reaches our goal.

Yes this code is working properly. (kernel wasn't connect at the time a took the screenshot, see the orange breaks.)

Yes, this is optimal.

4) 7. DEPTH LIMITED SEARCH

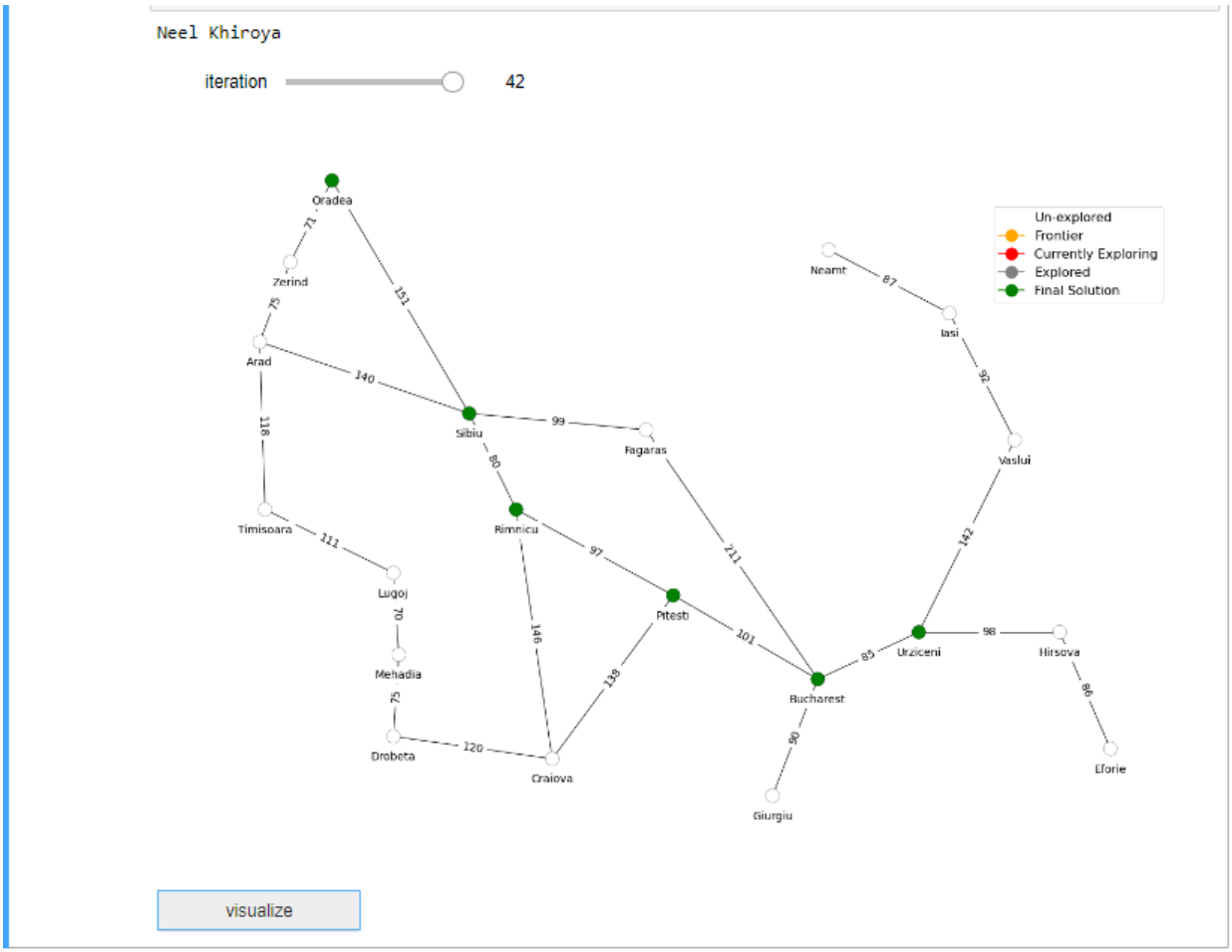


Depth limited search will create a limit as to how far down the tree will be searched. This isn't the number of iterations but instead the depth of the tree that we are limited to search.

I do not believe this code is working properly, the code doesn't expand all directions but rather tends to expand towards the goal, giving the impression that its not exploring based on depth limits.

Yes, this optimal.

5) 8. ITERATIVE DEEPENING SEARCH

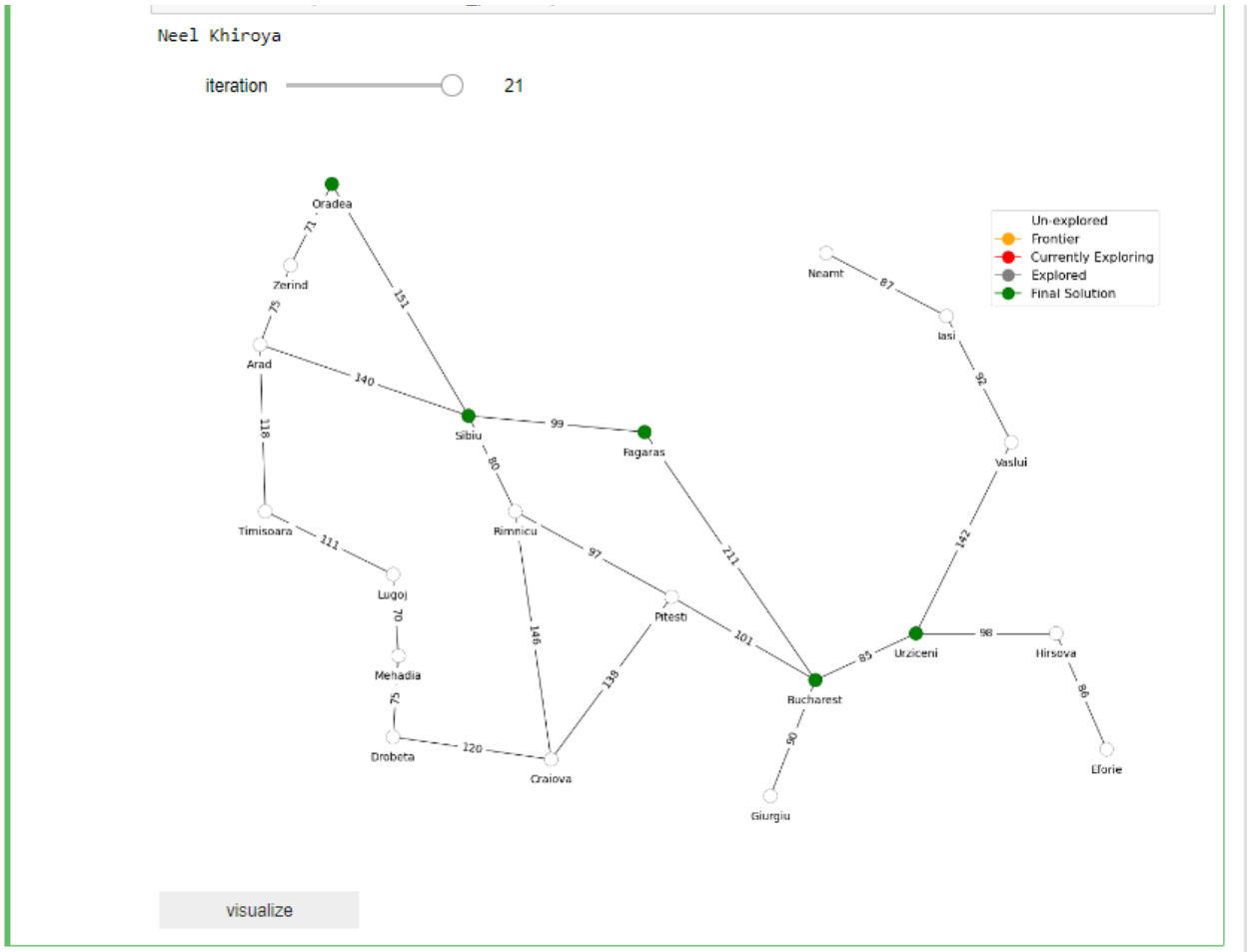


This algorithm combines DFS and BFS such that you use DFS with a depth limit that increments every complete search if no goal was found.

This code is not working properly since it doesn't expand to all nodes in the depth limit but rather tends to expand towards the goal.

Yes, this is optimal.

6) 9. GREEDY BEST FIRST SEARCH

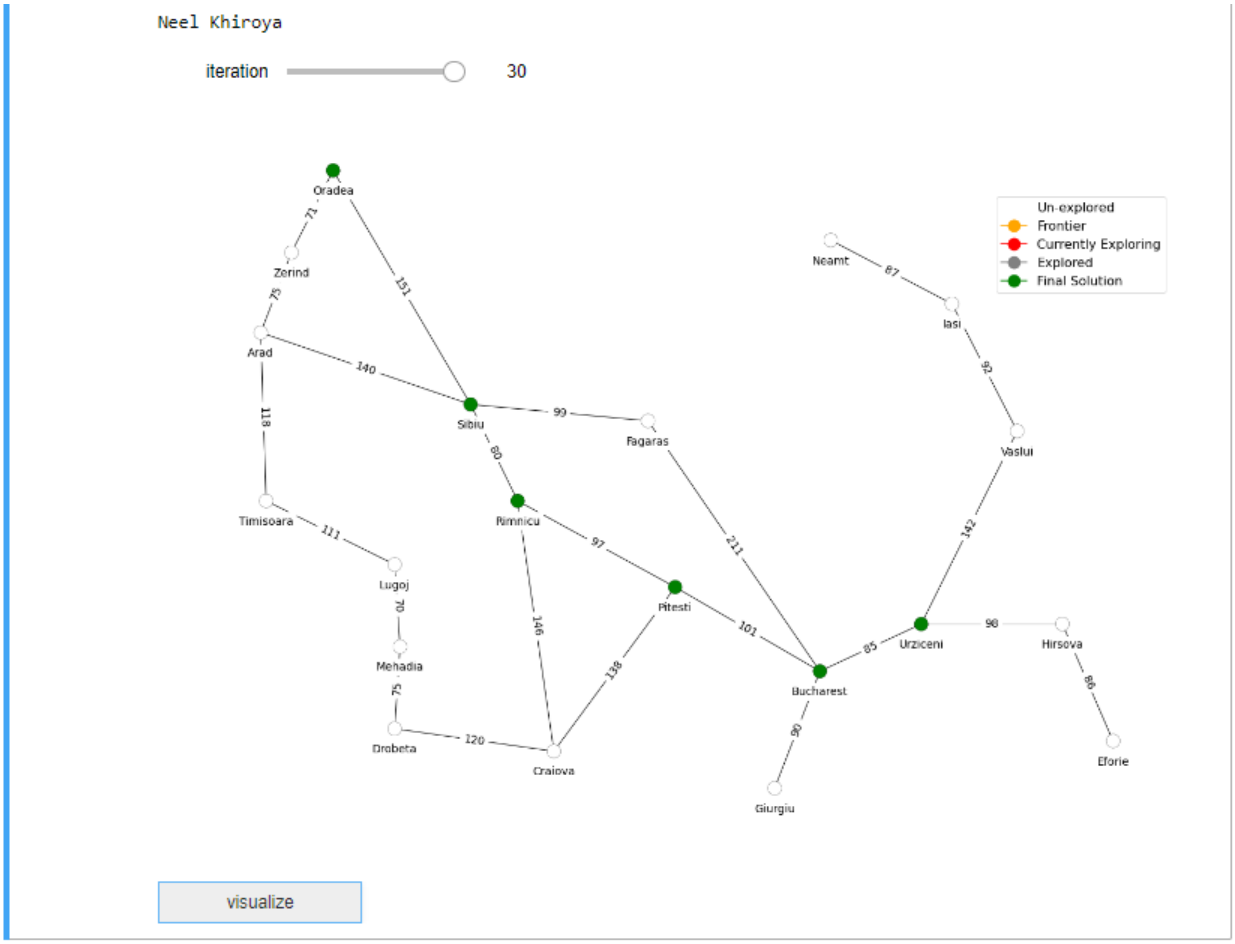


This algorithm will extend to whatever node it thinks it closest to the goal based off Heuristic.

Yes, this code is working properly.

No, this solution is not optimal.

7) 10. A* SEARCH

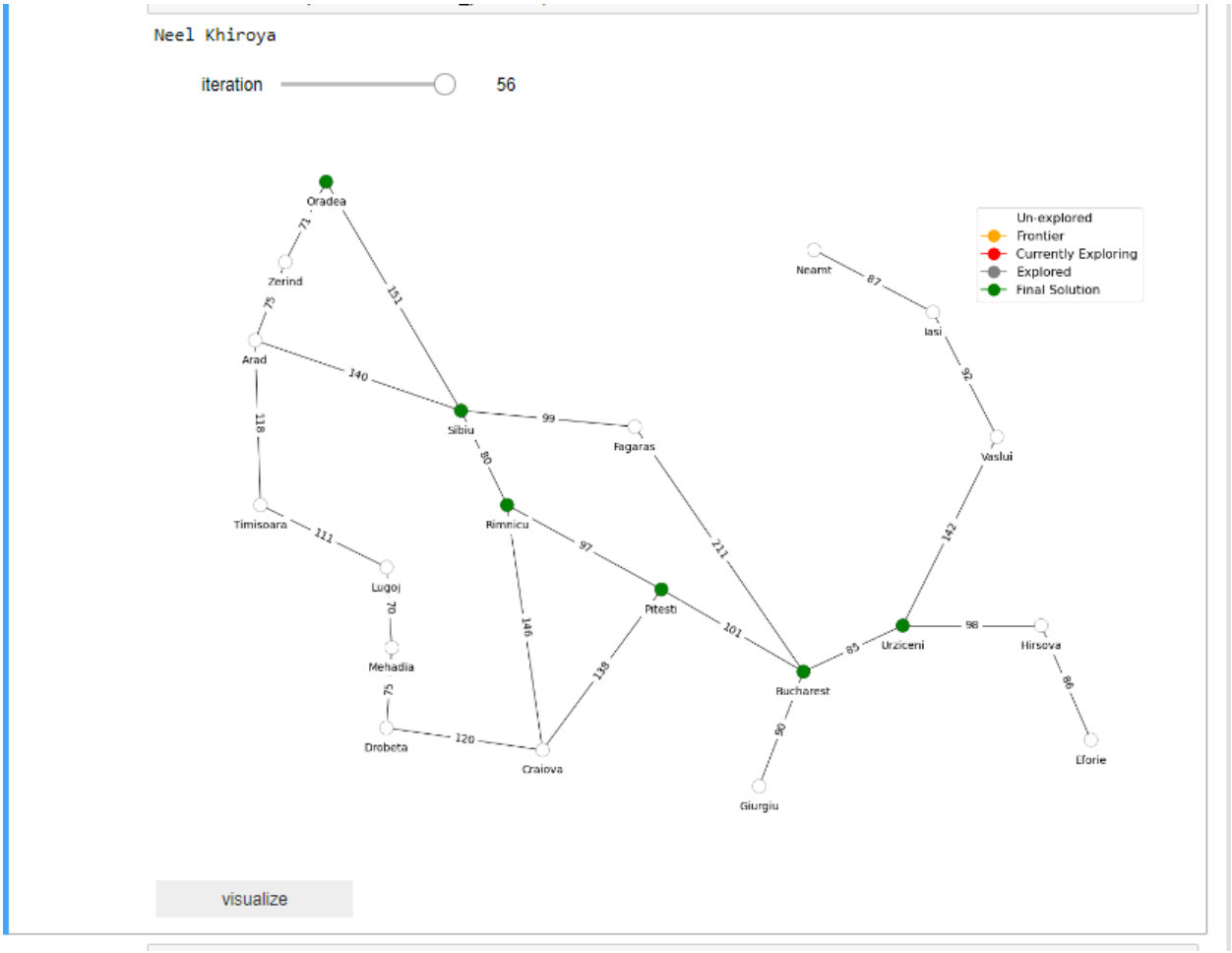


A* Search combines the algorithms of UCS and Greedy search, by calculating the sum of a node's proximity to the goal and the path cost. Once calculated then it will advance on the lowest sum.

Yes, this code is working properly.

Yes, this solution is optimal.

8) 11. RECURSIVE BEST FIRST SEARCH



This algorithm also calculates values as to what node it should advance to but will also store a limit value such that if the nodes current estimate is above that limit, it will revert to the parent node.

Yes, this code is working properly.

Yes, this solution is optimal.