CS 4033/6033 Artificial Intelligence

Homework Assignment #3

In Equal Contribution: Joe Ferguson, Brad Greene, Brett Marchese

Execution of the Program & Notes:

To run each of the searches:

```
? – breadth_first(startnode, targetnode, X).

%Where start and target node are undercase

? – depth_first(startnode, targetnode, X).

%For the incomplete, bad, but legitimate depth-first search

? – iterative_deepening_search(startnode,targetnode,X).

%For the complete(in this situation) depth first search

? – a_star(startnode,targetnode, X).

? – do_prog().

%For the test runs.
```

Please read the comments in the code for a better explanation.

Correctness & Efficiency:

Breadth-First: As far as correctness is concerned, breadth-first search is proven to be complete as a search algorithm. Meaning, it will always find a path between two nodes if a path exists.

For theoretical time and space complexity, breadth-first search is O(b^d) for both. This means that breadth-first search is relatively good for time, but bad for space as it keeps all nodes in memory.

It would be optimal if every edge had the same weight, which is not the case in this situation, so it is not optimal.

As an uninformed search, it performs very well, and is more reliable than depth first search. However, A* search outperforms it every time in this situation.

For the test cases, it visited a total of **14** cities, and a total of **1,435** cost.

Depth-First Search (The Iterative Deepening Variant): This is not a complete algorithm, both in theory and in this situation. However, by modifying it to never return to an already seen node, and by adding the iterative deepening functionality, it is able to find the path from every start node to Bucharest in this program.

For theoretical time and space complexity, depth-first search has O(b^m) and O(bm) respectively (m being maximum depth). This means that it may not be good for time complexity, at least in relation to breadth-first search, but it is better for space management than breadth-first search.

It is definitely not optimal. No explanation needed for this.

In all, in this situation (where there is not a lot of nodes, where it shines usually), it is not an appropriate algorithm, and either breadth-first or A* are way better algorithms.

For the test cases, it visited a total of **28** cities, for a total of **2,590** cost.

A*: This informed search algorithm is complete, because the heuristic (in this case) is admissible.

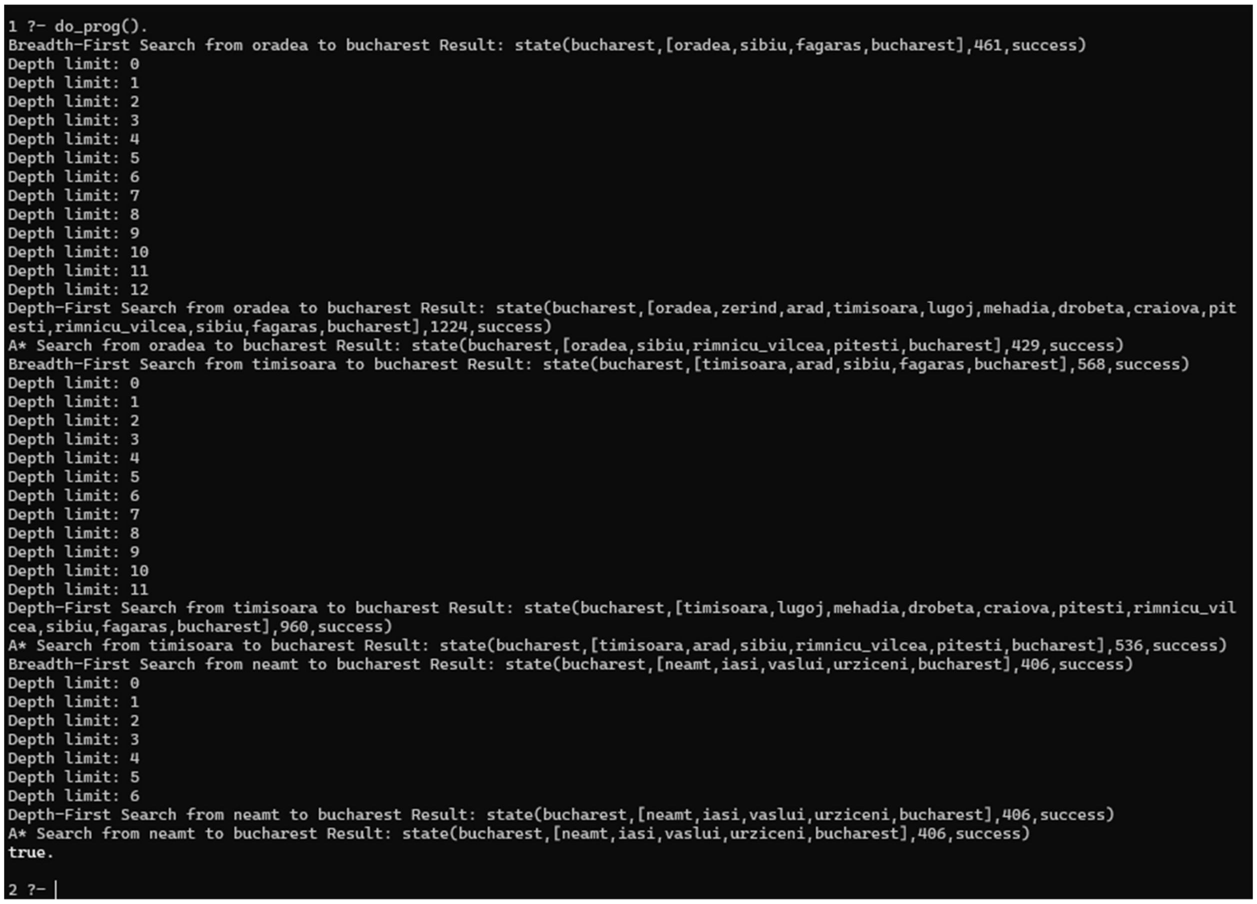
For theoretical time and space complexity, it is the same as breadth-first search. However, it should be noted as they are maximum measurements, and as the quality of the heuristic increases, it will both be faster and take up less space than breadth-first search. Which in this case, is true.

It is optimal. This once again is proven by the heuristic being admissible. It is also consistent.

A* is undoubtedly the best search algorithm for this situation. However, in situations where there are a large number of nodes, it's space management will become a problem.

For the test cases, it visited a total of **16** cities, and a total of **1,371** cost.

Paths of The Tests:



```
1 ?- do_prog().
Breadth-First Search from oradea to bucharest Result: state(bucharest,[oradea,sibiu,fagaras,bucharest],461,success)
Depth limit: 0
Depth limit: 1
Depth limit: 2
Depth limit: 3
Depth limit: 4
Depth limit: 5
Depth limit: 6
Depth limit: 7
Depth limit: 8
Depth limit: 9
Depth limit: 10
Depth limit: 11
Depth limit: 12
Depth-First Search from oradea to bucharest Result: state(bucharest,[oradea,zerind,arad,timisoara,lugoj,mehadia,drobeta,craiova,pit
esti,rimnicu_vilcea,sibiu,fagaras,bucharest],1224,success)
A* Search from oradea to bucharest Result: state(bucharest,[oradea,sibiu,rimnicu_vilcea,pitesti,bucharest],429,success)
Breadth-First Search from timisoara to bucharest Result: state(bucharest,[timisoara,arad,sibiu,fagaras,bucharest],568,success)
Depth limit: 0
Depth limit: 1
Depth limit: 2
Depth limit: 3
Depth limit: 4
Depth limit: 5
Depth limit: 6
Depth limit: 7
Depth limit: 8
Depth limit: 9
Depth limit: 10
Depth limit: 11
Depth-First Search from timisoara to bucharest Result: state(bucharest,[timisoara,lugoj,mehadia,drobeta,craiova,pitesti,rimnicu_vil
cea,sibiu,fagaras,bucharest],960,success)
A* Search from timisoara to bucharest Result: state(bucharest,[timisoara,arad,sibiu,rimnicu_vilcea,pitesti,bucharest],536,success)
Breadth-First Search from neamt to bucharest Result: state(bucharest,[neamt,iasi,vaslui,urziceni,bucharest],406,success)
Depth limit: 0
Depth limit: 1
Depth limit: 2
Depth limit: 3
Depth limit: 4
Depth limit: 5
Depth limit: 6
Depth-First Search from neamt to bucharest Result: state(bucharest,[neamt,iasi,vaslui,urziceni,bucharest],406,success)
A* Search from neamt to bucharest Result: state(bucharest,[neamt,iasi,vaslui,urziceni,bucharest],406,success)
true.

2 ?- |
```

Breadth-First Search from **oradea** to bucharest Result: state(bucharest,[oradea,sibiu,fagaras,bucharest],461,success)

Depth-First Search from **oradea** to bucharest Result: state(bucharest,[oradea,zerind,arad,timisoara,lugoj,mehadia,drobeta,craiova,pitesti,rimnicu_vilcea,sibiu,fagaras,bucharest],1224,success)

A* Search from **oradea** to bucharest Result: state(bucharest,[oradea,sibiu,rimnicu_vilcea,pitesti,bucharest],429,success)

Breadth-First Search from **timisoara** to bucharest Result: state(bucharest,[timisoara,arad,sibiu,fagaras,bucharest],568,success)

Depth-First Search from **timisoara** to bucharest Result: state(bucharest,[timisoara,lugoj,mehadia,drobeta,craiova,pitesti,rimnicu_vilcea,sibiu,fagaras,bucharest],960,success)

A* Search from **timisoara** to bucharest Result: state(bucharest,[timisoara,arad,sibiu,rimnicu_vilcea,pitesti,bucharest],536,success)

Breadth-First Search from **neamt** to bucharest Result: state(bucharest,[neamt,iasi,vaslui,urziceni,bucharest],406,success)

Depth-First Search from **neamt** to bucharest Result:

```
state(bucharest,[neamt,iasi,vaslui,urziceni,bucharest],406,success)
```

A* Search from **neamt** to bucharest Result:

```
state(bucharest,[neamt,iasi,vaslui,urziceni,bucharest],406,success)
```

As we can see, the algorithms performed as we thought they would. Depth-first, while able to find all the paths, takes the longest routes and has the most cost compared to the others. Breadth-first finds short paths, but not as optimally as A*. A* finds the shortest paths for this test cases, as well as any search for this situation (compared to the others).