Carla Estrada
Hugo De la Rosa

Report Lab 3: Implement Uninformed Search Algorithms

a. Which heuristics did you use for the A* algorithm?

Consistent heuristic and inconsistent heuristic for telling the number of misplaced containers in the stacks. As we saw in class a consistent heuristic is an admissible heuristic with the property that if there is a path from node x to node y then h(x) ≤ d(x, y) + h(y), where d(x, y) is the distance from x to y

b. Test your program with a couple of different problems. Increase the size of the problem to test the limits of your program. Make a table comparing how many nodes are searched to find the answer for each problem. For this table, you should compare a number of different problems (at least 3) to avoid a statistical bias.

Test 1:

2
(A); (B); (C)
(A, C); X; X

Test 2:

2

(A); (B); ()

(A, B); X; X

Test 3:

3

(A); (B); (C); ()

(); (A); (B); (C)

Test 4:

1

(A); (B); ()

(A, B); (); ()

Algorithm	2 (A); (B); (C) (A, C); X; X	2 (A); (B); () (A, B); X; X	3 (A); (B); (C); () (); (A); (B); (C
UCS	68	32	24
A* consistent	11	13	38
A* inconsistent	21	18	40

Carla Estrada
Hugo De la Rosa

c. Which of the four algorithms searches the least nodes and which one take the most?

After this the A* is the most efficient one in terms of larger problems but also is important to consider that in terms of how long the final path produced by the algorithm is, then A* is equivalent to any other optimal search algorithm. A* consistent searches the less nodes this is because it takes the less cost node, this depends having a good heuristic, if the heuristic overestimates if should not work correctly. And the one that visits the most nodes is inconsistent because it opens almost all the nodes since the heuristic value is not consistent to the value of the carried cost in the three, it get out of the bound limit with the consistent rule.

d. Why does this happen?

A* is special because can be morphed into other path-finding algorithms by playing with how it evaluates nodes and the heuristics it uses, it visits only the less cost nodes so it closer to the final path. The use of the priority structure helps this algorithm for finding the solution faster and with a lower cost. The tree does not grow through an expensive direction, as it always search for the cheapest path.

e. Which algorithms are optimal? Why?

A* algorithm guides an optimal path to a goal if the heuristic function h(n) is admissible, meaning it never overestimates actual cost. By definition A* algorithm is a best first search algorithm in which the cost associated with a node is f(n) = g(n) + h(n), where g(n) is the cost of the path from the initial state to node n and h(n) is the heuristic estimate or the cost or a path from node n to a goal. Thus, f(n) estimates the lowest total cost of any solution path going through node n.

f. In your opinion, what are the benefits of simpler algorithms versus more complex ones?

Simpler algorithms can be easier implement and getting the solution but more complex ones can also find the solution but with a lower cost and time. IN the other hand there are other characteristics that are needed to take in count for talking about optimal such the complexity of the tree and the size or the memory needed. So depending in the problem a more complex or simpler algorithm will be prefered