Laura Daniela Vargas Herrera A01023961
María Fernanda Vargas Herrera A01023957

(Un)informed Search Lab

- **Which heuristics did you use for the A* algorithm?**
    - The admissible heuristic we chose is how many letter are in the wrong block
    - The non-admissible heuristic we used is how many letter are in the right block

  We tried 2 other heuristics: Manhattan distance and Manhattan distance too, adding the difference of the actual level of the crane. We tried with this two different heuristic looking for improving time execution, but with this two new heuristic the cost was bigger.

  Alphagrader does not accept two dummy problem because of time execution, we tried to improve but the error still. Just to mention in our computer time execution for :

```
2
(A); (B); (C)
(A, C); X; X
3
(2, 0)

real    0m9.578s
user    0m0.024s
sys     0m0.017s
Air-de-Fernanda:UniformedSearchLab fernandavargas$ time python heuristic.py
3
(A); (B); (C); ()
(); (A); (B); (C)
6
(2, 3), (1, 2), (0, 1)

real    0m10.938s
user    0m0.071s
sys     0m0.015s
```

- 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.

Test1: function(3, "(A); (B); (C); ()", "(); (A); (B); (C)")

Test2: function(2, "(A); (B); (C)", "(A, C); X; X")

Test3: function(2, "(A); (B); ()", "(A, B); X; X")

Algorithm / Test	Test 1	Test 2	Test 3
BFS	1357	5	3
DFS	48	11	1
A* admissible	293	6	2

A* non-admissible	312	6	2

- **Which of the four algorithms searches the least nodes and which one take the most?**

  Based on the table the algorithm that visits less nodes is DFS, this happens because we added a limit, the tree is not going to extend the nodes that are in the n level or more (n means the max length of the containers), it will also get stuck in an infinite branch if it does not register the node who were already visited. After this the A* admissible is the one that 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 finally the algorithm that visits the most nodes is BFS because it opens all the nodes of each level and searches for the goal one.

- **Why does this happen?**

  Because A* is considered as an AI algorithm for searching the best path to a given problem, it visits only the less cost nodes so it closer to the final path. In the other hand BFS visits a specific level until it expands each node and if its doesn't find it in that level it will expand the next one until it reaches the goal node, this will take longer but it will find a solution.

- **Which algorithms are optimal? Why?**

  A* is optimal, the heuristic helps you to find the path as we are underestimating instead of overestimating the cost . 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. Breadth First Search is optimal , it visits more nodes than other algorithms but it the answer is on the tree it will find even if it takes longer that other algorithms.

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

  In our opinion simpler algorithms can be easier to understand and implement and getting the solution but more complex ones can also find the solution but with a lower cost and time. A* is a good algorithm for AI problems, the heuristic function helps to get a real cost of each node so with this the path can be reached with less nodes visited.