Lab 3: Implementing (Un)informed Search Algorithms

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

The nodes refer to the different positions the system can look like. The node S is the initial state of the containers in each stack. The next letters represent the possible positions the containers can move on. The goal is the desired state of the containers in each stack.

For example:

- S=(A); (B); (C)
- A= (A, B) ;();(C)
- B=(A);();(C, B)
- C= () ;(B, A); (C)
- D= () ;(B); (C, A)
- …
- Goal= (A, C); (B);()

Since g(n) is the cost of changing from one node to another, in this case is moving one container from one stack to other. For it I used the next formula:

$$g(n) = Picking\ up + Putting\ it\ down + Moving\ from\ one\ stack\ to\ another$$

$$g(n) = 0.5\,s + 0.5\,s + abs(stack_1 - stack_2) = 1\,s + abs(stack_1 - stack_2)$$

For calculating the maximum goal cost I used the first and last stack number. If there were 3 stacks, then the maximum goal would be of 3 s.

The heuristic needed to have a smaller value than that of the maximum goal, so I rested 2, which is the initial heuristic of the first node, to the maximum goal cost. As said before, given a maximum goal coast of 3 the heuristic would look like this:

Nodes	h(n)
S	2
A	1
B	1
C	1
D	1
…	1
Goal	0

The inadmissible heuristic that I would use is that a value equal or above the maximum goal cost. For example, from the last example if the maximum goal cost is 3 then instead of subtracting from it for the heuristic I would keep it equal or sum another value. The table would look like this

Nodes	h(n)
S	3
A	3
B	3

C	3
D	3
...	3
Goal	0

2. 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 several different problems (at least 3) to avoid a statistical bias. Which of the three algorithms (UCS, A with consistent and A with an inconsistent heuristic) searches the least nodes and which one take the most?

Event though I don't have both A algorithms, I investigated about the behaviors that the three algorithms will have.

The one with the least nodes would be for UCS and the one with the most nodes would be the A with the inconsistent heuristic

3. Why does this happen?

In the UCS it won't create many nodes since it is mainly focused on the current cost of each node.

For the case of the A with an inconsistent heuristic, this happens because it re-expands previous nodes that we already checked.

4. Which algorithms are optimal? Why?

UCS and A with a consistent heuristic are the only optimal algorithms, because they manage to find the best path with the lowest cost. While the A with an inconsistent heuristic as explained before it could create more nodes which could not lead to the most optimal solution.

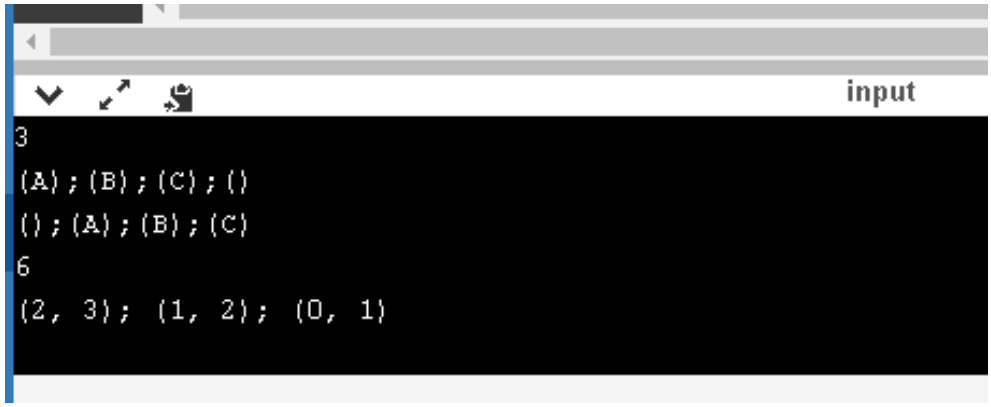
5. In your opinion, what are the benefits of simpler algorithms versus more complex ones?

The benefits of a simpler algorithms are that it is simpler to understand. If by any chance, there is an error or mistake in the algorithm we can correct them faster since we can identify easily where the problem is.

The benefits of a complex one is that it can be developed so it can solve even more difficult problems. It can analyze all possible sceneries of a problem and search for the most optimal solution.

UCS results

The algorithm that I uploaded didn't work even though the logic was fine for at least solving the tests. I add this only as proof or a record of what I did.


input
3
(A);(B);(C);()
();(A);(B);(C)
6
(2, 3); (1, 2); (0, 1)


input
2
(A);(B);()
(A,B);X;X
2
(1, 0)


input
1
(A);(B);()
(A,B);X;X
No solution found