# Exercise 1

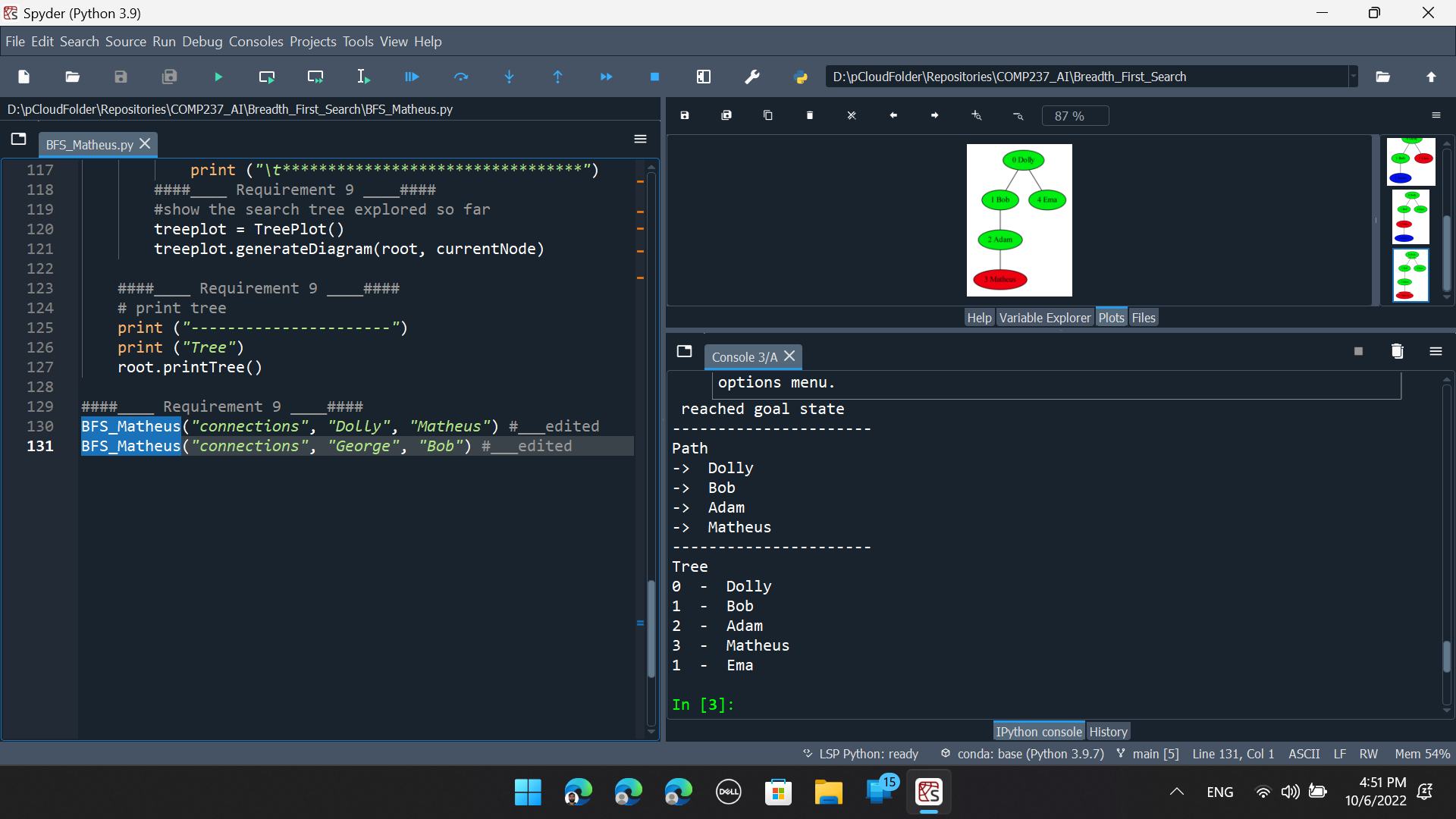


Figure 1: From Dolly to Matheus

A screenshot of a computer

Description automatically generated with medium confidence

Figure 2: From George to Bob

# Exercise 2

## Conditions

1. On the images:
   1. Green nodes: were visited and expanded
   2. Blue nodes are on the frontier
   3. The red node is the current node
2. The Greed Search and the A\* algorithms were executed as they were without changing any line from the given code.
3. I have used the A\* code as the base for the UCS code, and I’ve made changes to ignore the heuristic.
4. **I have made improvements to the code to make the plot cleaner.**
   1. **Nodes that have already been visited and expanded will not be put on the frontier again. For example, the Maths Building should have 2 children. However, the Car Park is not printed as a Maths Building child because it was already printed as the Maths Building parent. Also, Canteen is not printed as a Maths Building child because it was visited before through the Store’s path, which is the shortest path, so there is no need to include a new path to the priority queue again.**
   2. I have rounded the costs, so the numbers fit better as labels in the plot.

## Analysis

As we can see from Figure 4, the UCS algorithm does find the optimum path (minimum distance = 12), but to do that, it needs to expand 10 nodes (9 in green, 1 in red). From Figure 5 we notice that the Greed Search algorithm consumes much fewer resources (time and memory) because it only needs to expand 6 nodes (5 in green, 1 in red). Still, it could not find the optimum path, the total distance traveled using this algorithm was 14.2. Finally, Figure 6 shows us that the A\* algorithm can find the optimum path (as UCS) and it only expands 6 nodes (as the Greed Search). For this particular case, all of them require the same number of levels.

Chart

Description automatically generated

Figure 3: Graphical visualization of the given problem.

Diagram

Description automatically generated

Figure 4: UCS output.

Diagram

Description automatically generated

Figure 5:Greed Search output.

Diagram

Description automatically generated

Figure 6: A\* output.

# Exercise 3

## a) Complete the tree search showing all possible states that the agent will investigate to reach to the final state, indicating the values of h(n), g(n) and f(n). Below is the initial state and the first level expansion:

Figure 7: Complete tree for the given problem.

## b) Is this a good heuristic for the problem, explain why.

Yes, it is! To be a good heuristic it must be admissible and consistent. Not every admissible heuristic is consistent, but every consistent heuristic is admissible. So, all we need to do is prove that this heuristic is consistent, which I will do.

Mathematically, a heuristic is consistent if:

(1)

Where:

h(n) is the heuristic function;

n is any given node;

is the cost from node n to node (n+1)

This inequation derives from the triangle inequality; which that states no side of any triangle can be greater than the sum of the other 2 sides. Look at Figure 8 for reference.

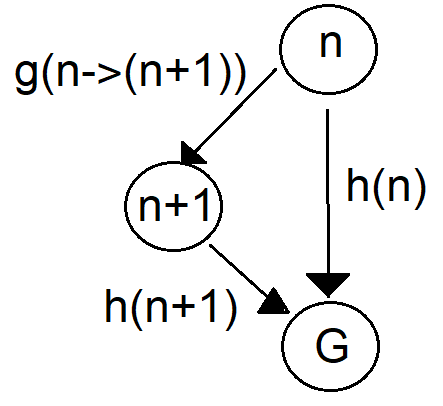


Figure 8: Heuristic from node n cannot be greater than heuristic from node (n+1) plus cost to get to node (n+1).

If we take the heuristic function, h(n), to be the number of misplaced tiles, and the cost, g(n) to be equal to the number of movements. We must consider 2 cases.

1. In the first, we make a move, and the moved tile will go to the right place. In this case:
   1. g(n->(n+1)) = 1 because we made a move
   2. h(n+1) = h(n) – 1 because we assumed that we have put a title in the right place, so the heuristic will reduce its value by 1 unit.

Now we can plug these values in equation 1, and evaluate it:

Therefore, in this case, the inequation holds still.

1. In the second case, we make a move, and the moved tile will still be in the wrong place. In this case:
   1. g(n->(n+1)) = 1 because we made a move
   2. h(n+1) = h(n) because we didn’t change the number of tiles in the wrong place, so the heuristic value doesn’t change.

Now we can plug these values in equation 1, and evaluate it:

Therefore, in this case, the inequation also holds still.

In conclusion, we have just proved that this heuristic is consistent; hence it is also admissible. As this heuristic is both admissible and consistent, it is a good heuristic for this problem.

## c) Suggest another heuristic, explain why you think it would be a good heuristic.

Using the sum of the distances (either Manhattan distance or Euclidian distance) of each tile to their respective right places is also a good heuristic. This can be proved in a very similar way to what I’ve just demonstrated in the previous question.