COMP 8700 - Introduction to AI
Assignment 2

Yameen Ajani (110096721)

October 21, 2022

Question 1

1. Problem Formulation

- **Problem:** The problem is to help 3 missionaries and 3 cannibals cross a river. The boat can only carry 2 people at a time. If the cannibals ever outnumber the missionaries on either side of the river, the cannibals will eat the missionaries. The goal is to get all 6 people across the river.
- **State & State Space:** A particular state is represented by a tuple of 3 elements. The first element is the number of missionaries on the left side of the river, the second element is the number of cannibals on the left side of the river, and the third element is the position of the boat (1 if the boat is on the left side and 0 if its on the right). The state space is the set of all possible states. An example state is (3, 1, 1) which means there are 3 missionaries and 1 cannibal on the left side of the river and the boat is also on the left side of the river.
- **Initial State:** The initial state is $(3,3,1)$ which represents 3 missionaries, 3 cannibals and the boat on the left side of the river.
- **Goal State:** The goal state is $(0,0,0)$ which represents 0 missionaries, 0 cannibals on the left side and the boat on the right side of the river.
- **Actions:** The actions can be moving either 1 or 2 people from one side to another. The actions are represented as a tuple of the number of missionaries and cannibals moving from one side to another. An example action would be $(1,0)$ which represents moving 1 missionary from one side of the river to the other. For a given state, the possible actions are the set of all possible actions that can be taken from that state. For instance, in the initial state, the possible actions are $(1,0)$, $(2,0)$, $(0,1)$, $(0,2)$ and $(1,1)$.
- **Transition Model:** The transition model is a function that takes a state and an action and returns the resulting state. For instance, in the initial state, if the action $(1,0)$ is taken, the resulting state would be $(2,3,0)$ which represents 2 missionaries, 3 cannibals on the left side and the boat on the right side of the river.
- **Cost Function:** The cost function is a function that takes a state and an action and returns the cost of taking that action in that state. For this problem, the cost function is a constant function that returns 1 for all states and actions.

- **State Space Graph:** The state space graph is a graph where each node represents a state and each edge represents an action. The graph is directed and weighted. The weight of each edge is the cost of taking that action in that state. The graph is shown below (**NOTE:** Since the state space is huge, I have only drawn the permissible states).

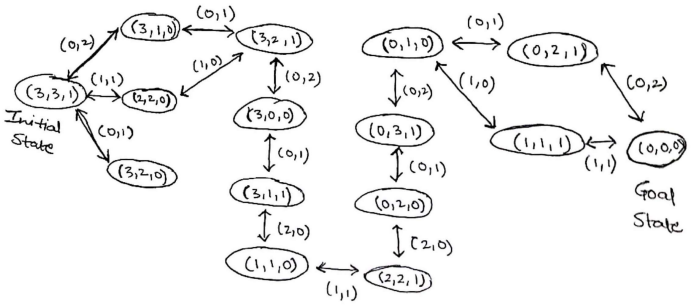

Figure 1: State Space Graph

2. Search Algorithms

Please refer to **this link** for the video of the implementation of the search algorithms. You can view the source code **here**.

3. Heuristic Function

Heuristic Function: The heuristic function is a function that takes a state and returns an estimate of the cost of the cheapest path from that state to the goal state. For this problem, the heuristic function is a function that takes a state and returns the number of missionaries and cannibals on the left side of the river. For instance, in the initial state, the heuristic function would return 6. The heuristic function is admissible because it never overestimates the cost of the cheapest path from a state to the goal state. The heuristic function is consistent because the cost of taking an action is always 1. Therefore, the cost of the cheapest path from a state to the goal state is always 1 more than the cost of the cheapest path from the resulting state to the goal state. Therefore, the heuristic function is consistent.

Please refer to **this link** for the video of the implementation of the heuristic search algorithms. You can view the source code **here**.

4. Repeated States

The search algorithms do not handle repeated states. This means that if a state is reached multiple times, the search algorithms will treat it as a new state each time.

Question 2

Which of the following are true and which are false?

1. Depth-first search always expands at least as many nodes as A* search with an admissible heuristic.
   False. In general, it is true that depth-first search will expand more nodes than A* search with an admissible heuristic. However, this is not always the case. It is possible that the depth-first search gets lucky and finds the goal state before the A* search with an admissible heuristic. In this case, the depth-first search will expand fewer nodes than the A* search.

2. $h(n) = 0$ is an admissible heuristic for the 8-puzzle.
   True. $h(n) = 0$ is an admissible heuristic for the 8-puzzle. An admissible heuristic is a heuristic that never overestimates the cost of the cheapest path from a state to the goal state. Therefore, since choosing $h(n) = 0$ never overestimates the cost, it is an admissible heuristic.

3. A* is of no use in robotics because percepts, states, and actions are continuous.
   False. A* can be used to find the optimal path from a state to a goal state even if the state space is continuous. This can be done by discretizing the state space. Hence, A* algorithm can be used in any problem where the state space is continuous including robotics.

4. Breadth-first search is complete even if zero step costs are allowed.
   True. This is because technically the step cost is irrelevant as far as "completeness" is concerned. A search algorithm is "complete" if it always finds a solution if one exists. It does not matter if the solution is optimal or not. Thus, if a goal exists, breadth-first search will always find it.

5. Every admissible heuristic is also consistent.
   False. The AI textbook mentions that even though it is difficult to find an admissible heuristic that is not consistent, it is not impossible. However, [1] paper by Felner et al. shows that it is possible to find an admissible heuristic that is not consistent. It gives the example of the 8-puzzle problem. The heuristic function chooses at random either the set of tiles (1, 2, 3) or the set of tiles (4, 5, 6) and computes their Manhattan distance from their goal positions. This heuristic function is admissible because it never overestimates the cost of the cheapest path from a state to the goal state. However, it is not consistent because there isn't a clear relationship between the heuristic estimates at each node.

Question 3

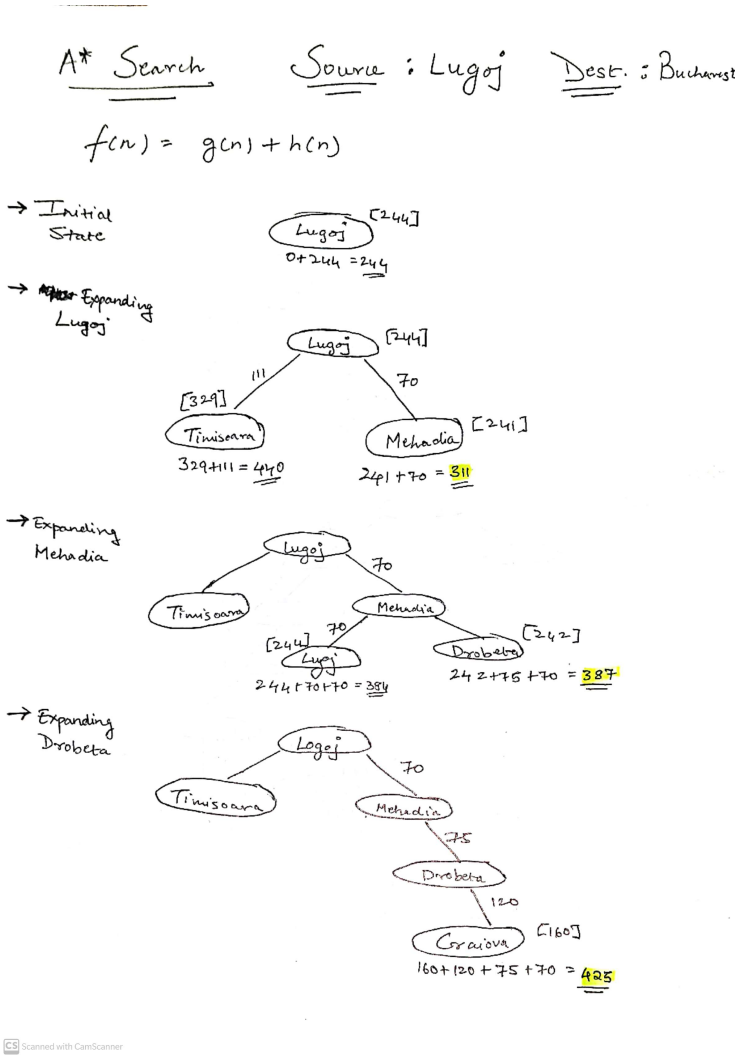

Figure 2: A* Search - 1

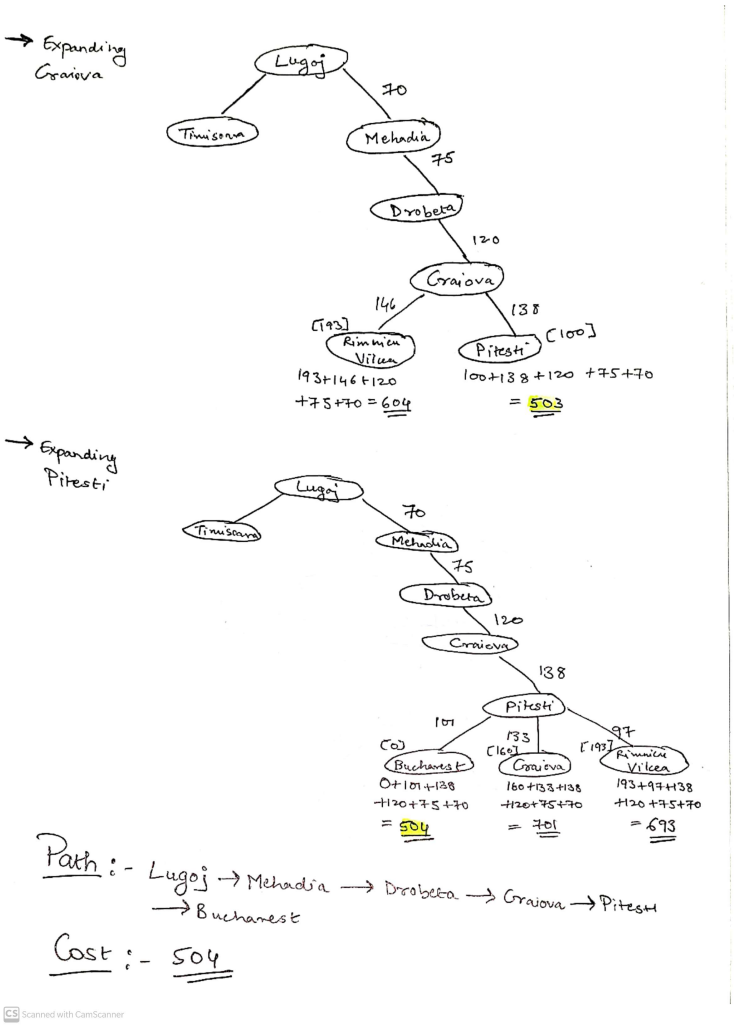

Figure 3: A* Search - 2

Question 4

An algorithm is said to be **complete** if it always finds a solution if one exists. It should also report failure if no solution exists. **Cost optimality** means that the algorithm finds a solution with the minimum path cost.

The A* algorithm will **always be complete** as long as all the costs are non-negative and the search space is finite. Considering the example of *Fig 3.1* in the textbook, if we eliminate loops while also keeping track of the paths then all the nodes are reachable in some way.

We will check if A* is cost optimal in the cases described below -

1. **The heuristic function used is admissible.**
   A heuristic function is admissible if it never overestimates the cost of the cheapest

path from a state to the goal state. If the heuristic function is admissible, then the A* algorithm will always find the optimal path i.e. it is cost-optimal. For instance
-

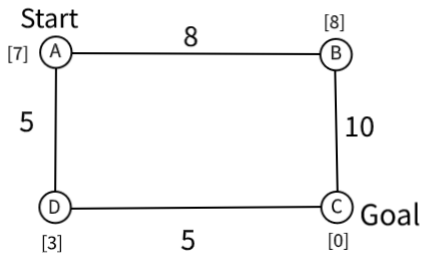

Figure 4: Admissible Heuristic

If we calculate the heuristic function for nodes B and D, we get $B = 16$, $D = 8$. According to these values, the algorithm will expand D which is the optimal path.

2. **The heuristic function used is always overestimate for each state.**
   If the heuristic is an overestimation at every node then the solution obtained may or may not be optimal. It will be optimal only in a special case where the heuristic is overestimated only by the difference between the second-most optimal solution and the most optimal solution. Consider the following example -

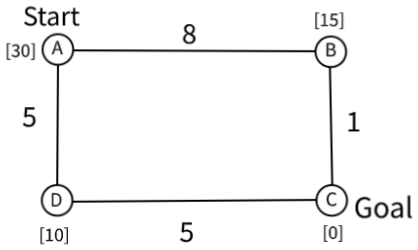

Figure 5: Overestimate Heuristic

We know that $f(n) = g(n) + h(n)$. The value of $f(n)$ at B is 23 while the value of $f(n)$ at D is 15. However, going through D is not the optimal path. The optimal path is through B which is ignored due to the overestimated heuristic value.

3. **The heuristic function used sometimes overestimates for some states and sometimes underestimates for some states.**
   In such a case, the assuredness of the algorithm to find the optimal path is not guaranteed. It will vary case to case and depends on the heuristic function in that particular case. Hence, cost-optimality is possible but not guaranteed.

Question 5

Steepest-ascent hill climbing algorithm

Please refer to **this link** for the video of the implementation of the steepest-ascent hill climbing algorithm for the 8 queens problem. You can view the source code **here**.

Steepest-ascent hill climbing algorithm with sideways moves

Please refer to **this link** for the video of the implementation of the steepest-ascent hill climbing algorithm with sideways moves for the 8 queens problem. You can view the source code **here**.

For implementing the steepest-ascent hill climbing algorithm in python, I used this article as a reference.

Question 6

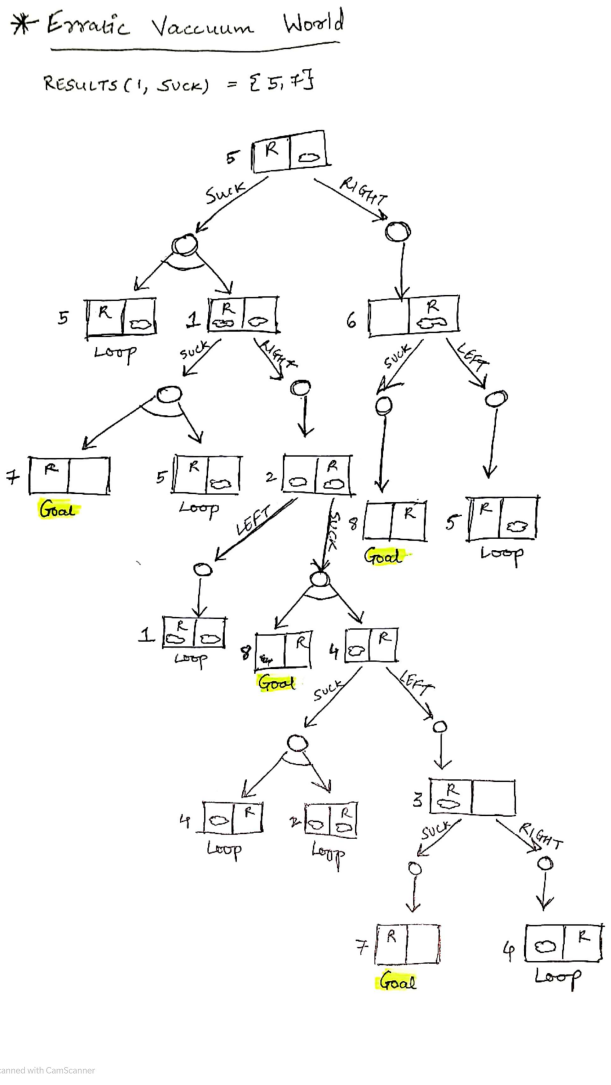

Figure 6: AND-OR search tree for Erratic Vaccuum World

Question 7

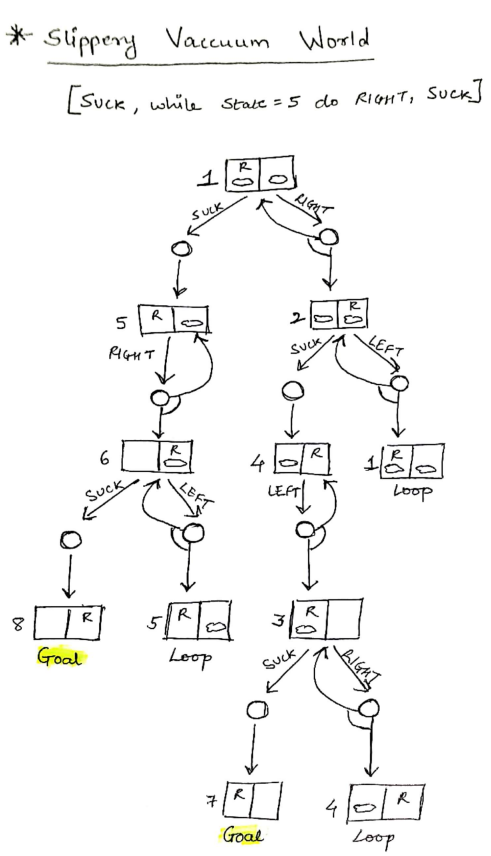

Figure 7: AND-OR search tree for Slippery Vaccuum World

References

[1] A. Felner, U. Zahavi, R. Holte, J. Schaeffer, N. Sturtevant, and Z. Zhang, "Inconsistent heuristics in theory and practice," *Artificial Intelligence*, vol. 175, no. 9, pp. 1570–1603, 2011.