State Space Search Problem
Lab Assignment - 0

Aman Kumar Raj 201951019
Balram Choudhary 201951039
Hemant Udeniya 201951070
Mayank Sharma 201951092

***Abstract*—In this experiment we have model a given problem in terms of state space search problem and solve the same using BFS/ DFS.**

I. Introduction

The goal of this project is to understand state space search problems. These types of problems mainly consists of initial state and goal state. And aim is to reach from initial state to goal state with given constraints. In this lab there are two problems.
1) MissionaryCannibal problem and
2) Rabbit Leap problem.
In MissionaryCannibal problem there are total 3 Missionaries and 3 Cannibals on left side of a rive. They all have to reach right side of the river, such that at most 2 people can ride at a time and Cannibal should not be majority in no anywhere.(presumably Cannibals will eat Missionaries if became majority). We have to find an approach which requires minimum no of trip.
In Rabbit Leap problem there are total 7 stones. on left most 3 stones there are rabbits(call it L) who want to go to right and on right most 3 stones there are rabbits(call it R) who want to go to left. The middle stone is empty. A L rabbit can only jump to next stone or next to next stone in right given it is empty. Similarly a R rabbit can only jump to next stone or next to next stone in right given it is empty. (i.e. they can't jump backward). We have to find a approach which requires minimum no of jump. We have simulated these problems and solved it using python in Vs-code. You can find all the links here:- (here)

II. Theory

A. State Space Search problem

State space search problems can be solved using BFS or DFS. Few of the commons steps irrespective of problem be it MissionaryCannibal or Rabbit Leap are as follows:

We first create Node Class which has three variables.
State:- State of the node
Parent:- Parent of the node from where AGENT [1] has come to current state
path cost:- Total cost required to reach current state from initial state
Then We maintain two data structures:-
Frontier:- It stores all the valid neighbours of current node. It is like a queue(priority queue) with methods like pop node, top node, push/add node, is empty .
Visited:- It stores all visited nodes. It can be a simple list or a hashtable.
Transition Matrix:- It stores possible neighbours of current state.

B. Why we use BFS DFS

To solve these problems if we use brute force approach then state space will grow exponentially and it not be possible to solve with present computation power.(Both in terms of storage and time)

III. Procedure

A. Our implementation (BFS and DFS)

In this nodes are expanded based on minimum path cost. It is kind of a greedy approach which always chooses the path with minimum path cost.
In this we started with initial node and started exploring it. We picked on node from initial node(which is root node) and we explored it. So before exploring we moved it to Visited list. In exploration process we searched for all the valid possible neighbours of current state and kept it in Frontier. Next:-
In **Best First Search** we pick the node with minimum path cost and explore it.
In **Depth First Search** we pick node by any order like lexicography order and explore it.
As previous we first moved it to visited. If the state was already present in Visited list the we remove it and search another minimum path cost node in Frontier. Also when we explore a node we look for all of it's neighbour and put them in Frontier. If a state is already present in Frontier then we compare their path cost and node with least path cost in two are kept and other one is dropped.
We keep on exploring nodes using above aproach untill we reach goal state or our Frontier empties in which case goal is not reachable.

B. Outputs:

We have implemented this in python, here is the outputs of both these problems:-
MissionaryCannibal

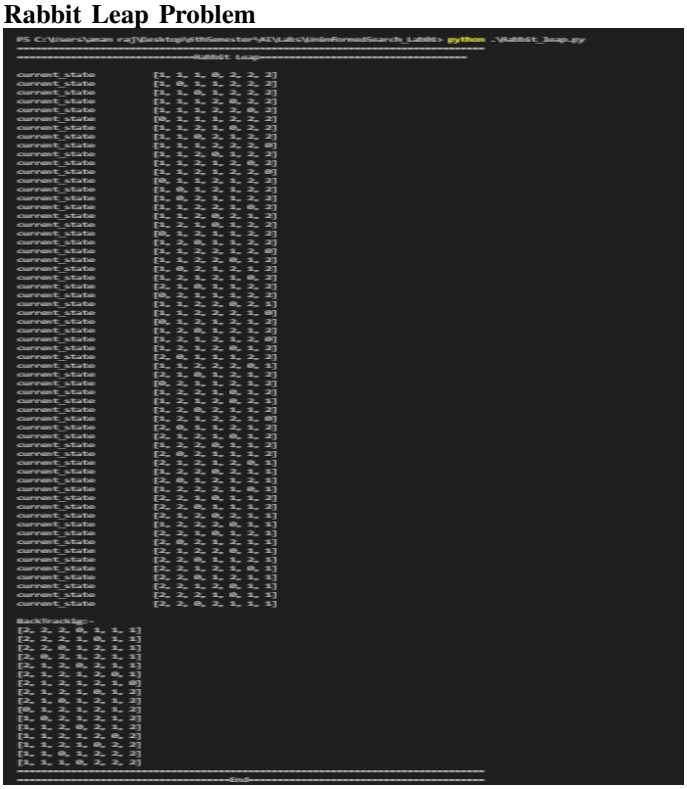
```
PS C:\Users\aman raj\Desktop\6thSemester\AI\Labs\UninformedSearch_Lab01> python .\Missionary_and_Cannibal.py
================================================================================
==============================Missionary&Cannibal===============================

current_state               [3, 3, 0]
current_state               (3, 2, 0)
current_state               (3, 1, 0)
current_state               (2, 2, 0)
current_state               (0, 3, 0)
current_state               (0, 2, 0)
current_state               (0, 0, 1)

BackTracking:
(0, 0, 1)
(0, 2, 0)
(0, 3, 0)
(2, 2, 0)
(3, 1, 0)
(3, 2, 0)
[3, 3, 0]
================================================================================
===================================End==========================================

PS C:\Users\aman raj\Desktop\6thSemester\AI\Labs\UninformedSearch_Lab01>
```

When we have set **Verbose 1(maximum logs):**

```
PS C:\Users\aman raj\Desktop\6thSemester\AI\Labs\UninformedSearch_Lab01> python .\Missionary_and_Cannibal.py
================================================================================
==============================Missionary&Cannibal===============================

visited: []
current_state: [3, 3, 0]

visited: [[3, 3, 0]]
current_state: (3, 2, 0)

visited: [[3, 3, 0], (3, 2, 0)]
current_state: (3, 1, 0)

visited: [[3, 3, 0], (3, 2, 0), (3, 1, 0)]
current_state: (2, 2, 0)

visited: [[3, 3, 0], (3, 2, 0), (3, 1, 0), (2, 2, 0)]
current_state: (0, 3, 0)

visited: [[3, 3, 0], (3, 2, 0), (3, 1, 0), (2, 2, 0), (0, 3, 0)]
current_state: (0, 2, 0)

current_state: (0, 0, 1)

BackTracking:
(0, 0, 1)
(0, 2, 0)
(0, 3, 0)
(2, 2, 0)
(3, 1, 0)
(3, 2, 0)
[3, 3, 0]
================================================================================
-----------------------------------End------------------------------------------

PS C:\Users\aman raj\Desktop\6thSemester\AI\Labs\UninformedSearch_Lab01>
```

When we have set **Verbose 1(maximum logs):**

Rabbit Leap Problem



IV. Conclusion

We see that although search space is quite small in comparison to brute force approach but still, with even such small problems our search space has grown very large. So for bigger problems like Chess or Go these Uninformed Search solutions will become intractable as The Frontier will grow exponentially. So we need to come up with some heuristics[1]. This experiment was very interesting I got to learn how to formulate state space search problems and also understood computation complexities of these problems.

References

[1] Artificial Intelligence: a Modern Approach, Russell and Norvig (Fourth edition)