State-Space Search Using BFS and DFS: Missionaries & Cannibals and Rabbit Leap

Gajipara Nikunj, Parmar Divyraj, Saikat Jana
M.Tech CSE (AI), Batch 2025–27
Indian Institute of Information Technology, Vadodara
Email: {student1, student2, student3}@iiitvadodara.ac.in

***Abstract*—This experiment focuses on solving classical state-space search problems using uninformed search algorithms—Breadth-First Search (BFS) and Depth-First Search (DFS). Two well-known AI problems, Missionaries and Cannibals and the Rabbit Leap Puzzle, are modeled as state-space search problems. Both algorithms are implemented in Python to generate optimal or feasible solutions, and their performance in terms of optimality, completeness, and complexity is compared.**

***Index Terms*—Breadth-First Search, Depth-First Search, State-Space Search, Missionaries and Cannibals, Rabbit Leap**

I. Objective

- To model AI problems as state-space search problems.
- To apply and compare Breadth-First Search (BFS) and Depth-First Search (DFS).
- To analyze and compare their solution paths, optimality, and complexity.
- To implement the algorithms in Python and visualize solution sequences.

II. Problem Definition

A. Problem 1: Missionaries and Cannibals

Three missionaries and three cannibals are on one side of a river. The boat can hold at most two people. The goal is to move all to the opposite side without ever leaving more cannibals than missionaries on either bank.

B. Problem 2: Rabbit Leap Puzzle

Three rabbits facing east and three rabbits facing west are separated by an empty space on a line of stones. Rabbits can move forward one step or jump over one rabbit. The goal is to swap the positions of all rabbits.

III. Methodology

Both problems were formulated as state-space search tasks, where each valid configuration of the environment represents a node in the search graph. The algorithms BFS and DFS were implemented to explore this graph or tree.

GitHub Repository: https://github.com/NikunjGajipara27/Lab-Assignment

A. State-Space Representation

1) Missionaries & Cannibals:

- State: $(M_{\text{left}}, C_{\text{left}}, \text{BoatSide})$
- Initial State: $(3, 3, \text{‘L’})$
- Goal State: $(0, 0, \text{‘R’})$
- Valid transitions ensure missionaries are never outnumbered.

2) Rabbit Leap:

- State: A list representing stone positions, e.g., [E, E, E, _, W, W, W].
- Initial: East rabbits on left, West rabbits on right.
- Goal: West rabbits on left, East rabbits on right.
- Legal moves: move one step forward or jump over one opponent.

IV. Results and Analysis

Table I
Comparison of BFS and DFS for Both Problems

Problem	Algorithm	Optimal	Steps	Time	Space
Missionaries & Cannibals	BFS	Yes	11	$O(b^d)$	$O(b^d)$
Missionaries & Cannibals	DFS	Not guaranteed	14	$O(b^m)$	$O(m)$
Rabbit Leap	BFS	Yes	15	$O(b^d)$	$O(b^d)$
Rabbit Leap	DFS	Not guaranteed	17	$O(b^m)$	$O(m)$

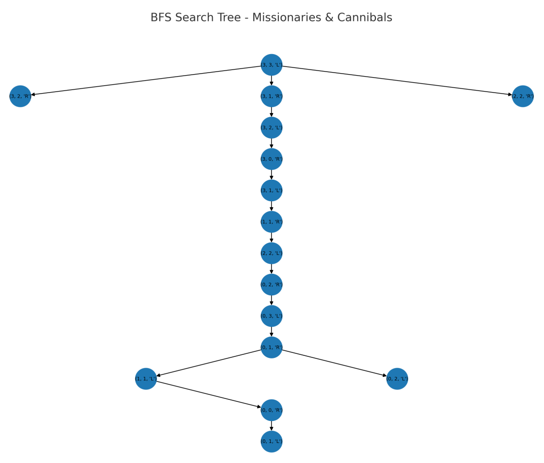

Figure 1. BFS search tree for the Missionaries & Cannibals problem.

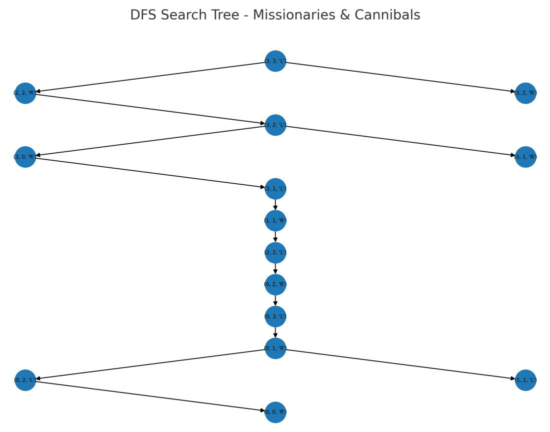

Figure 2. DFS search tree for the Missionaries & Cannibals problem.

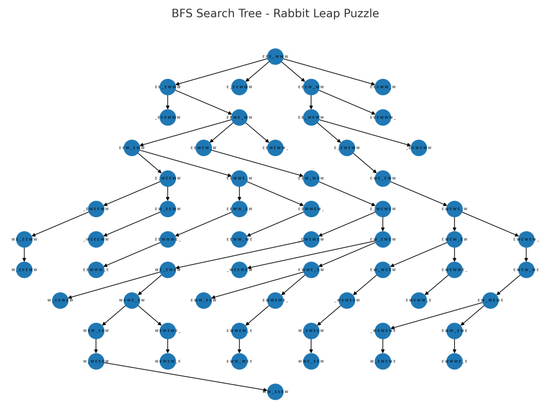

Figure 3. Search tree for the Rabbit Leap puzzle.

Observations

- BFS finds the shortest solution due to its level-order traversal.
- DFS may find a longer or looping path if cycles are not avoided.
- BFS consumes more memory but guarantees optimality.

V. Discussion

This experiment demonstrates how uninformed search algorithms explore problem spaces:

- BFS ensures optimality by exploring all nodes at a given depth.
- DFS may get stuck in deep branches but uses less memory.
- Proper state modeling and successor generation are critical.
- In both problems, the branching factor is moderate, but BFS still grows exponentially with depth.

VI. Conclusion

The assignment successfully models two classical AI problems using state-space search. Both BFS and DFS were

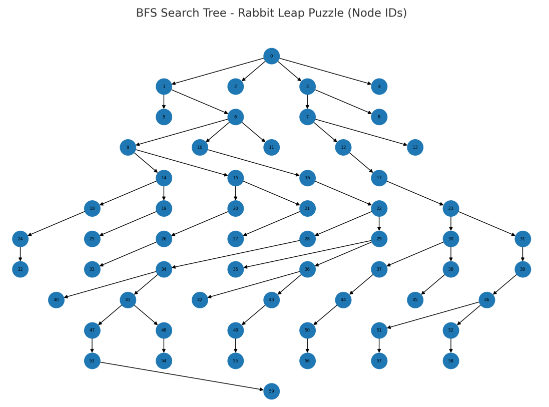

Figure 4. BFS search tree for the Rabbit Leap puzzle (node IDs).

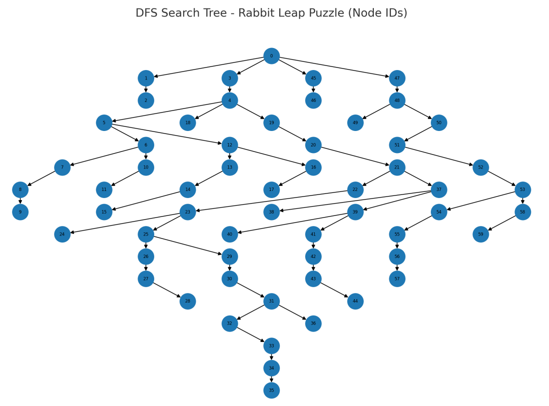

Figure 5. DFS search tree for the Rabbit Leap puzzle (node IDs).

implemented to find valid solutions. BFS consistently found optimal paths, while DFS provided feasible (but not always shortest) paths with less memory overhead. This experiment reinforces fundamental AI search principles and introduces practical problem modeling in Python.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.

[2] D. Khemani, *A First Course in Artificial Intelligence*. McGraw-Hill, 2013.

[3] CS659 Lecture notes and lab manual, IIIT Vadodara, Autumn 2025–26.