Zinn Morton

Josh Bell

CS331 Programming Assignment 1 Writeup

1. How to Run the Program:

- To compile: g++ -std=c++11 assignment1.cpp -g -o assignment1.exe
- To run: ./assignment1.exe [input file] [goal file] [bfs/dfs/iddfs/astar] [output file]

2. Methodology:

a. Implementation Methodology:

I implemented my searches with a vector of visited states, a frontier, and a HashMap that maps states to their depth. If an explored state is seen again, it is skipped *unless* its depth is lower than the saved depth on the HashMap. The frontier is a stack for DFS, a queue for BFS, a stack for IDDFS, and a min priority queue (extracts minimum depth + heuristic) for A*. I used no depth limit on DFS, BFS, or A*.

b. How the Algorithm Works:

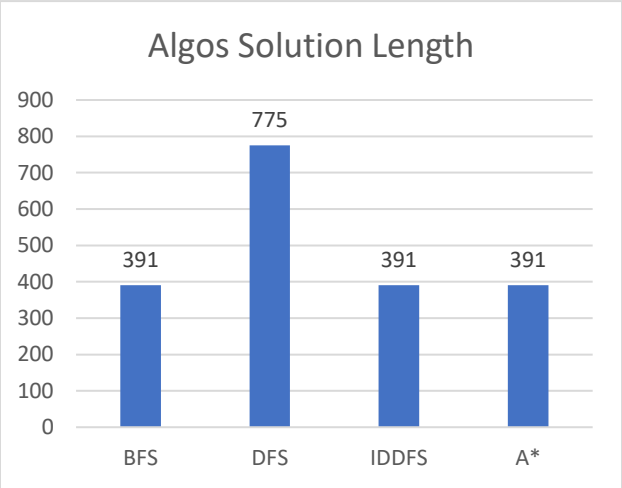
- Initialize seen states, frontier (push initial state onto frontier), and explored HashMap.
- While(true)
  - If frontier is empty, fail and return.
  - Take state from frontier.
  - If explored and the depth <= the depth stored in HashMap, continue to next state in frontier.
  - If state is goal state, print solution and return.
  - Mark depth for state in HashMap
  - Expand current state and add to frontier.
- IDDFS simply repeats DFS with an incrementing depth limit.
- My A* heuristic (where g is goal and s is current state):

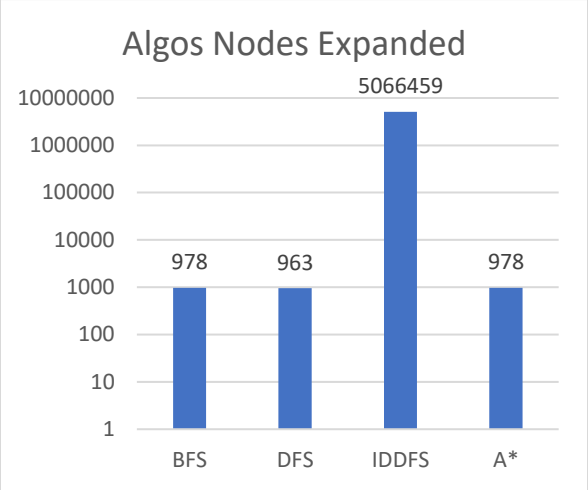
$$\frac{|g.RightWolves - s.RightWolves| + |g.RightChickens - s.RightChickens| + |g.LeftWolves - s.LeftWolves| + |g.LeftChickens - s.LeftChickens|}{4}$$

I chose this heuristic because it gets lower the closer the state is to the goal state. I divided by 4 so it was admissible.

3. Results:

a. Using *start3.txt* and *goal3.txt*:





4. Discussion:

a. Expected Outcomes:

- BFS, IDDFS, and A* were optimal.
- DFS was not optimal.
- IDDFS had the most nodes expanded.
- IDDFS took the longest to run.

b. Unexpected Outcomes:

- A* didn't decrease the nodes expanded. Perhaps the heuristic I used was too optimistic?
-

5. Conclusion:

a. What can I conclude from these results?

- For this particular problem, BFS or A* (especially with a better heuristic) is best for an optimal solution.
- For any solution, DFS will likely take the least amount of time.

b. Which search algorithm performs the best?

DFS, although it is not optimal.

c. Was this expected?

Yes. There are many solutions so it makes sense that DFS would find one while expanding less than BFS or A*.