Nicholas Biggerstaff
Thomas Tonini
CS 331
April 18, 2021

# Programming Assignment #1 Report

- Methodology:

For this assignment, we coded breadth-first search, depth-first search, iterative deepening depth-first search, and A-Star search. We chose to code this in python. For the game, we created it using the dictionary data type to represent the node of the tree. This suited us well because it has the efficiency of O(1) if the key is known. This limits the amount of time that our search algorithms take to execute the best path.

Breadth-first search and depth-first search are very similar in their execution except BFS is done using a first in first out queue. Depth-first search is done using a last in the first out queue. This difference in queues is how we differentiate between the two searches. A FIFO queue will search all of the nodes that it visits first before going to another leaf. A LIFO will search down the entire branch until the frontier is empty or a solution is found.

Iterative Deeping Depth-first Search takes a blend of breadth and depth-first search. IDDFS has the space efficiency of a DFS graph - O(depth of the tree), and the speed of BFS as long as it is not a large tree.

A-star is very efficient because it utilizes heuristics so that it can have “brains” and complete the best path possible. For our heuristic, we first returned h(n) = 0 if n was the goal. Then we assumed that animals start on the right bank and move to the left bank, and found the number of animals left on the right bank. Since up to 2 animals can move at once, we subtracted 1 from this value to get the underestimated number of moves. We also didn’t want our heuristic to return 0 for non-goal states, so we made it return the max of 1 and the sum on the right bank - 1.

- Results
-

| | | | |
|---|---|---|---|
| - | - Start1.txt & Goal1.txt | - Start2.txt & Goal2.txt | - Start3.txt & Goal3.txt |

| | | | |
|---|---|---|---|
| - BFS | - Steps: 12<br>- Nodes: 45 | - DNF | - DNF |
| - DFS | - Steps: 11<br>- Nodes: 11 | - Steps: 93<br>- Nodes: 93 | - Steps: 750<br>- Nodes: 750 |
| - IDDFS | - Steps: 12<br>- Nodes: 0 | - DNF | - DNF |
| - A-star | - Steps: 12<br>- Nodes: 27 | - Steps: 44<br>- Nodes: 228 | - Steps: 378<br>- Nodes: 1428 |

-

- Discussion

  All of these tests were run on the engineering server that I was connected to with a VPN. I also had a twenty-second cut-off for these algorithms, that's why some did not finish. The breadth-first search was very slow. It was only able to complete the first test file within the allotted time. DFS was able to complete all the tests quickly, but as discussed in the text, it does not give optimal results. IDDFS was extremely slow and, just like BFS was not able to complete the last two tests. A* completed all its tests, much quicker than DFS did.

  Breadth-first, Iterative Deeping Depth-first, and A* Search all come to an optimal solution, depth-first search does not necessarily come to an optimal solution

- Conclusion

  Overall A-star seems to be the best choice for a search algorithm. It was able to give an optimal solution but also be quick. In my opinion, breadth-first search and Iterative Deeping Depth-first search should not be used for larger sample sizes since they will take much longer.