Programming Utilizing Uninformed and Informed Search Algorithms

K. Boring, H. Nguyen

Oregon State University

Corvallis, OR 97333

17 April 2015

1. Methodology

All four algorithms are taken in by the same parameters including the starting node and a goal node. An *unordered_map* is used as a *hashTable* to store places that have already been visited. Parent nodes are remembered for each node and can traverse upwards. A *deque* is used for the fringe. For all algorithms, the initial check for whether or not the program has the hit the goal node is passed first, then the expansion of a node follows afterwards. For every expansion, the algorithm runs through a switch statement that every possible case of movement by the missionaries and cannibals.

For Breadth-first search, if the state has never been visited before, then it is added to the *hashTable* and the fringe is pushed back. This process is continued until the fringe is empty.

For Depth-first search, the node is pushed front to the fringe if the state has not been visited before. If the new depth is less than the existing node, then it is erased from the hashTable and then it is pushed to the front of the fringe.

For Iterative Deepening Depth-first search, the maximum depth is initialized at 1. Nested in a while loop, the algorithm runs through the fringe and checks if the node's depth is past the maximum depth. The node is then expanded. It is added to the *hashTable* if it has not been visited before and replaced only if the new depth is less than the existing node of that state. After the fringe has been empty, the maximum depth is then incremented by 1, and the algorithm reruns through the fringe.

For A* search, a priority queue is used to categorize the nodes by its $f\,value$, calculated by its Heuristic function. The heuristic function for the A* search is calculated by the minimum amount of times the boat would have to travel without restrictions to carry all missionaries and cannibals across the river. This function can be expressed by:

$$H(n) = ((NumberOfCannibals + NumberOfMissionaries)/2)$$

2. Results

Results from the three test cases can be seen in Figure 1.0.

	BFS	DFS	IDDFS	ASTAR
Test 1: Depth	11	11	11	11
Test 1: Expanded Nodes	14	12	94	13
Test 2: Depth	33	33	33	33
Test 2: Expanded Nodes	72	34	1615	47
Test 3: Depth	377	377	377	387
Test 3: Expanded Nodes	2184	378	13632445	593

Figure 1.0: Table of solution paths and nodes expanded for search algorithms

3. Discussion

All four algorithms get to the same depth of the solution path, which means that they all were able to retrieve a solution to the problem. Many of the programs too significantly longer than others to compute. IDDFS for example, took to the length of 7-10 minutes to compute for the third test and resulted in an exceedingly large amount of nodes expanded. This is what was to be expected as instructed by the professor.

DFS is also noticeable for finding a solution path very quickly without expanding many nodes. We can justify its solution path if it takes the same nodal expansion every time successfully and it looks just for the first solution possible. A* and BFS have very similar numbers with their node expansions for each test as well as the same expanded nodes for the first test, which confirms similar behavior within its node expansion algorithm itself. We found H(n) to be admissible, but it is dominated by another expression:

$$H(n) = (2\ x\ (NumberOfCannibals\ + NumberOfMissionaries) - 1)$$

This heuristic function however could not be implemented correctly, portraying higher depth than anticipated for the last test. Which is comparable to our last test 3. It is possible that the implementation for a large amount of missionaries and cannibals could be misinterpreted with our A* algorithm

4. Conclusion

We can conclude that while all algorithms work in their own way, others are more optimal, preferred, and performs much better. With the right heuristic function, we find that the A* search algorithm works the best that finds the optimal path for the solution of any depth, while maintaining its expansion of nodes low. A* search is both complete and optimal as long as the heuristic function is consistent and that the depth is finite. This becomes especially into scaling when more missionaries and cannibals are inputted onto the program. DFS reveals a small path, but it is not the optimal solution. IDDFS shows the complete and optimal solution, but the time complexity becomes significantly higher. The only competitive search becomes BFS, which is the secondarily ideal algorithm, but it could also vary depending on where the solution path would exist. Compared to A* search algorithm, as long as there is a good heuristic function, then it is less likely to have a time complexity of $O(b^d)$.