3.8 (a)

Answer: because if there are arbitrary negative path costs, any unvisited node, which might hold a value negative enough, can bring advantage to a search path to be the optimal path by appending the negative-valued node itself to the search path.

39 (a)

Initial state – three missionaries and three cannibals on one bank of a river along with a boat

Actions – ship one or two of the missionaries and cannibals from one to the other side of the river

Transition model – returns the resulting state as per the given state and actions

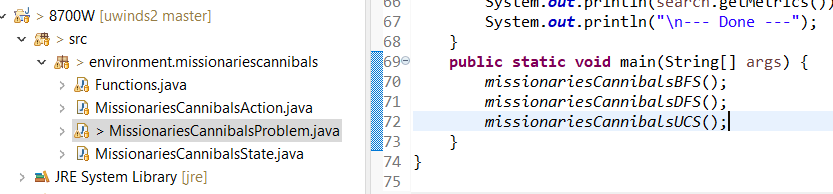
Goal test – checks whether all the missionaries and cannibals are on the other bank of the river.

Path cost – the cost is the number of shipments where each shipment is of cost 1

39 (b) Implement and solve the problem optimally using an appropriate search algorithm. Is it a good idea to check for repeated states?

Firstly, yes it is a good idea to check repeated states to avoid repeated node visit and expanding, the advantage is obvious, with repeated states check, it usually finds the optimal solution with dozens of node expanding, while if not check, it will conduct thousands of expanding.

Source code of my implementation in 4 algorithms:



Run the MissionariesCannibalsProblem.java file to see the results:

BreadthFirstSearch - {maxQueueSize=3, nodesExpanded=13, pathCost=11.0, queueSize=2}

DepthFirstSearch - {maxQueueSize=5, nodesExpanded=11, pathCost=11.0, queueSize=4}

UniformCostSearch - {maxQueueSize=3, nodesExpanded=14, pathCost=11.0, queueSize=1}

UniformCostSearch NotCheckRepeatedStates

* {maxQueueSize=14520, nodesExpanded=11445, pathCost=11.0, queueSize=14519}

Performance comparison:

As shown above, with graph search checking repeated states, BFS, DFS, and UnifomCostSearch performs well to find the optimal path, but with tree search during UniformCostSearch which does not check repeated states, it expands more than ten thousands of node expanding owning to repeated states visits.

3.13

3.21

Create a small example of your own and follow the A\* algorithm on non-admissible heuristic to see what the output is. Is the output optimal?

Note question 3.9 is in fact a programming question, although the original question asks you to implement a solution using one algorithm, you should try at 3 different search algorithms and compare their performance. In your submission, you should report the algorithms implemented and show the performance comparison.