

Class: CS76

Term: Fall

Year: 2021

Assignment: PA1

Name: Okorie Kenechukwu

https://github.com/rehoboth23/cs70_pa1.git

Implementation

BFS

My breath first search solution explores the nodes in the order they are discovered but always at uniform depth. I use the python deque data structure as the fringe to implement a FIFO operation in constant time. I also maintain a set of seen (a state is added to seen when it first comes in as a successor) states to prevent exploring the same state multiple times. The search terminates when it attempts to explore from the goal state. The solution path (if one exists is found through back tracking). The search node has a parent field which is also a SearchNode. The back track ends when it finds a Node which has no parent node (i.e. parent = None)

DFS

My depth first search explores each successor to a depth limit(or a pseudo limit determined as redundant through path checking). Each recursion call has the solution with the current successor added to the top of the path.

The if on the collapse of the recursive stack the last item in the solution path is the goal node then a solution was found. If after searching all possible successors of a node no solution was found, the last item on the path (which by applying recursive logic is the current node) is removed and the path is returned.

IDS

My iterative deepening search simply leverages my depth first search solution. The depth first search solution I designed has depth checking and a depth limit built into it. So the iterative deepening search simply call my dfs solution with increasing depth limits.

Search Problem

The search problem was abstracted into a generic class (like a java interface) that has a start_state, a goal_state, a method to generate successors of a state and a method to check if a state is the goal for the problem. The **FoxProblem** inherits from this class and adds a boat capacity (this is something I added. I edited the problem to solve for an arbitrary number of seats on the boat).

Visual solution

I added a visualizer for the solution of the fox problem path.int **visualize_solution.py**.