Programming Assignment 1 COSC76 (21F), 2021, Kunal Jha

Setup

To run this program, open the file named *foxes.py* in your preferred Python editor. The 3 test case problems described in the report are preloaded, so running the program as is should produce the results depicted in the report. To add your own test cases, you may create a new **FoxProblem** object. A FoxProblem object takes in the starting state as a parameter, and this state is represented as a triple labeled *(x, y, z)*. The variable *x* represents how many chickens you would like to initially be present. The variable *y* represents how many foxes you would like to initially be present. The variable *z* should be 1 to ensure a solution can be found (it represents the boat being on the left side of the river initially). If *z* is not 1, the program should not return any solution.

Once your FoxProblem object has been created, you can test its performance on different algorithms by creating either a *bfs_search*, *dfs_search*, or *ids_search* object, and inputting the problem as a parameter. The search object won't actually display anything on its own, so you must print the search object after you've created it to visualize whether a successful path was found. A code template can be found below:

testProblem = FoxProblem((numChicken, numFox, 1))
print(searchMethod(testProblem))

To see the full implementation of the search algorithms, open the file named *uninformed_search.py*. The bfs_search, backchaining, dfs_search, and ids_search algorithms are all included sequentially. To see the methods designed to assess the validity of states and find successors, open the file named *FoxProblem.py*. The get_successors, validAction, and goalTest methods are all included sequentially.