CS4300Spring 2018

Project #1: Missionaries and Cannibals
Due Date: 2/20/2018

Submission: Please submit all files through Canvas. Please do not email final solutions to me, but feel free to ask me questions and code by email.

In this project you will be attempting to solve the Cannibals and Missionaries problem:
Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place.

Your task is to implement and solve this problem by generating an appropriate tree structure and then using an appropriate search algorithm to search this tree.

First you will want to formulate the problem precisely and ensure that you understand what a valid solution is. In order to simplify this for you, I suggest you use the following tips, though you can use your own formulation if you want:

Represent the current state using a simple vector <a,b,c>. This would represent the number of missionaries on the wrong side, cannibals on the wrong side and number of boats on the wrong side. So starting out, all the cannibals, boats(we only have one) and missionaries are on the wrong side, for <3,3,1>

Represent actions by using vector subtraction/addition to change the state. For example, suppose a boat carrying one cannibal crosses the river, you would subtract <0,1,1> from our <3,3,1> to get <3,2,0>. Meaning that we now have 3 missionaries on the wrong side, 2 cannibals on the wrong side, and no boats on the right side.

For each state, there are 5 possible actions, <1,0,1>, <2,0,1>, <0,1,1> <0,2,1>, <1,1,1>. You start at <3,3,1> and start generating a search tree, getting the successor nodes by *subtracting* these actions from it. This generates the child states. From these children, you get to the grandchildren by *adding* these actions to it. So at each depth you reverse the addition or subtraction. Continue this until you reach a node of <0,0,0>, which is the solution.

Make sure to check for invalid states, that is, if any node has more cannibals than missionaries on either bank. For example, from the initial state of <3,3,1>, only 3 valid children will be generated, <3,2,0>, <3,1,0> and <2,2,0>.

You must use iterative deepening for this search. I suggest you write a function that, given a maximum depth, searches for a solution and either returns it or says it did not find a solution. Then repeat this until a solution is found. I expect your program to present the entire solution(ie: Which actions were taken) after it finds a solution.

Sample output should be:
Solution found at depth 11!
S0: <3,3,1>
S1: <2,2,0>
…
S11: <0,0,0>

Hint: As with most projects, try and do this incrementally. Test your function that generates possible transitions from a node before even coding the overall search. Then fix a depth of 1 (or 2) and output all the transitions, making sure all of them are legal. In a tree of that small depth, it should be possible to walk through it by hand and make sure things are valid. Do not code this entire project and then only test it at the end! You will have no idea where your error is at that point.

Turnin:
Submit your project on canvas as a zip file or an attached source file if it is only one file. Include in the zip file or as another attachment a README file explaining your environment that you used and how to compile your project.