Missionaries and Cannibals Solution

Your name here

January 12, 2014

1 Introduction

States are either legal, or not legal. First, give an upper bound on the number of states, without considering legality of states. (Hint – 331 is one state. 231 another, although illegal. Keep counting.) Describe how you got this number.

Use a drawing program such as inkscape to draw part of the graph of states, including at least the first state, all actions from that state, and all actions from those first-reached states. Show which of these states are legal and which aren't (for example, by using the color of the nodes). Include and describe this figure in your report.

Latex note: You should save your figure as a pdf, which is a vector graphic format, not as a pixel-based format such as png, jpeg, gif, tiff, or bmp. Vector graphics can be scaled and will still look good in your pdf report, while bitmaps (pixel-based) will not scale well.

2 Implementation of the model

Now you are ready to write and test the code that implements the model. (You'll write the search algorithms in the next section.) CannibalProblem.java is your starting point. Ultimately, this will extend 'UUSearch-Problem', but for now, UUSearchProblem is unfinished and thus broken, so you probably want to delete the 'extends' keyword. A CannibalProblem instance should keep track of how many cannibals and missionaries there are in the problem (three of each is standard, but your solution should be general).

The CannibalProblem class is also a good place to put an inner class that represents nodes for the search. A CannibalNode will represent the "current" state of the problem at some point in the exploration. For example, at some point in the search, we might be exploring the (2, 2, 1) state.

A 'CannibalNode' should provide a method that creates a list of nodes that represent legal states reachable using legal actions from the current node. I called this getSuccessors, but you'll have ot implement the body of the method. Also implement any other methods that have a comment like "you write this" in the provided code.

Test your code thoroughly before you go on; write some test code (perhaps in a main method of CannibalProblem) that creates some nodes, finds their successors, and prints out the results. The results should agree with those that you drew for the intro. This testing step is not only required, but important. Errors in your model will lead to very-hard to diagnose errors that will make it virtually impossible to debug your search algorithms.

Present the work in your report. Include the key parts of your code using the listings environment in LaTeX, and discuss how the code works. Also describe your testing and convince the reader that your testing process demonstrated correctness of your code.

The model is implemented in `CannibalProblem.java`. Here's my code for `getSuccessors`:

```
public ArrayList<UUSearchNode> getSuccessors() {
  // your code here
}
```

The basic idea of `getSuccessors` is...
I used a method `isSafeState` that returns `true` if...

```
private boolean isSafeState(int m, int c) {

}
```

3 Breadth-first search

4 Memoizing depth-first search

5 Path-checking depth-first search

6 Iterative deepening search

7 Lossy missionaries and cannibals