Coins and Scales

Eric McCambridge

Instructions

To run the code, run the main method in the Test.java file. This solves the problem for n=1 to 8. It prints out the number of nanoseconds it took to find a solution to the problem, and also tests the solution found to make sure it will correctly name the odd coin out in any case.

Description

My project searches for a solution to the coin measuring problem from the midterm. It uses A* search through the space of partial plans to find the optimal (i.e. shortest) decision tree that will find the odd coin out. Each state in the search is a partial plan, represented by the most recently added node to the decision tree. This allows us to work on multiple possible subtrees for the same overall tree without representing the common parts of the tree multiple times, and since our heuristic is admissible, the first complete subtree found will be the optimal one.

There are two types of vertices in the search. A measure vertex, which specifies a specific measure to take using the scale, and a belief vertex, which specifies the (coin,weight) pairs that are possible given the measurements so far. A measure vertex is represented by a belief state parent and two sets of coins that will be measured against each other. A belief vertex is represented by a measure vertex parent and two lists, the possible heavy coins and the possible light coins.

When expanded, a measure vertex lists out the belief states that would result from the three possible results from measuring the two sets of coins it specifies, given its belief state parent. If the measurement says the first set is lighter than the second set, either one of the coins on the left is lighter, or one of the coins on the right is heavier, and vice versa if the first set is heavier. If the sets are equal in weight, then one of the coins in neither set is either heavy or light.

A belief vertex, when expanded, enumerates all the possible measurements that could be taken on the coins it believes could be the odd coin. Since this number can grow so rapidly, there are a few assumptions made to make the problem tractable. Obviously, the two sets must be the same size and have no common elements. Also, each pair of sets is only listed once (or twice if there is heavy/light symmetry) which would not be the case if we naïvely chose the subsets. We also assume that we will have equal number of coins we believe are light on each side, so as to evenly distribute the beliefs among the three belief state children. There are many more optimizations that could be done here, but given the time frame of the project, these seem fair and allow for the problem to be solved for small numbers of coins in a reasonable amount of time on an average machine.

Once a node is expanded, it simply waits until its children finish and then hands its solution up to its parent. To prevent searching through space for which a solution has already been found, a node marks itself as complete if either all of its children are complete, or the parent is complete (because a solution was found in a different subtree). This means an overall solution is found when the original root node, a belief state containing all n coins both heavy and light, is complete. Since there are two different types of vertices, we are guaranteed that all children of a certain belief state come from the same measure.

Analysis

My analysis of the system consisted of running the test cases several times on one machine. The results are shown below. Figure 1 is a graph of the run times to find solutions for n=3 to 8 (log scale), and figure 2 is a table of the average run times for n=3 to 8.

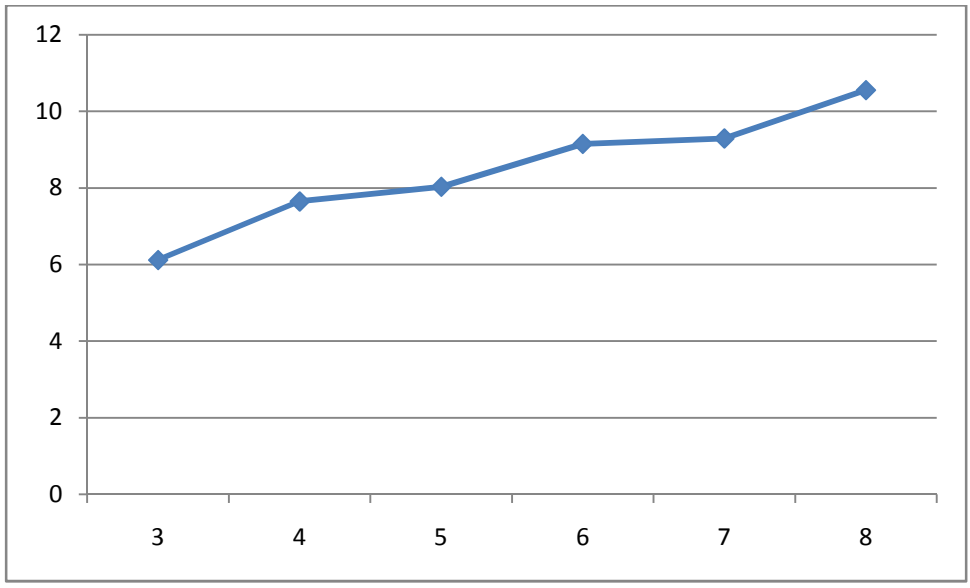

Figure 1 – log(run time) vs. n

N	Time (avg in seconds)
3	0.0013
4	0.0437
5	0.1076
6	1.4012
7	1.9628
8	35.9740

Figure 2 – run time vs. n

As we can see, the growth rate of the running time is exponential, which is expected. Solutions for n=3 to 7 finish quickly (all under 2 seconds), while n=8 is a sharp jump up to over 30 seconds, and a solution for n=9 was not found within several minutes.

There's plenty of work to be done in improving the performance of the system, but it works well on small inputs that would be solvable by humans given a few minutes. I think the first place to attack the problem would be in the enumeration of measures from a given belief state. There is a lot of symmetry there, both between heavy and light mirror images, and in the fact that we treat the coins as labeled, even though treating them as unlabeled would provide all of the same measures. One way to accomplish that would be to change the representation of a belief state from two lists of coins (heavy and light) to three numbers: the number that could only be light, the number that could only be heavy, and the number that could be either.