CS-463G

Program 1

Report

Jacob Blankenship, Daniel Krutsick

**A description in English of your data structure:**

**Code for the data structures, along with instructions on how to run it:**

**An example of the GUI output:**

**A description of the randomizer:**

Our randomizer is simple but effective. We determined that without center moves, there are 12 unique moves that a cube can have. 8 on one side, and 4 on an adjacent side of that side. Before ending the movement function, we store the current move being made, if the next move chosen is that same previous move, we would just pick a new move and continue this process until a new move is selected.

**Code for the randomizer, along with instructions on how to run it:**

Our randomizer is part of our original function.

**Your heuristic, clearly described and justified, including an argument that it is admissible:**

Our heuristics will be very simple. It will consider two factors, but we will only implement 1 for this assignment:

* How many unique values are on each side of each face.
* How far non-present values are from that face.

Take the following cube side:

[ 1 6 8 ]

[ 9 8 9 ]

[ 2 2 1 ]

The example side has only 5 unique values, meaning that 4 are missing (the 3,4,5, and 7). The nearest available 3 and 4 are one side away. By available, I mean that it there is at least 2 3’s, 2 4’s, and 2 7’s on one side adjacent to our example face, so “giving up” a 2, 3, or 7 is ok. If there was only 1 two or 1 three, we would be essentially stealing it from the other face. There is also 3 5’s that are on the backside of the cube, and as a result would require 2 movement to bring a “free” 5 to our example side.

This would evaluate this sides “score” to be: +3 for the nearby available 3,4, and 7, +2 for the available 5 that is two movements away. We would then find all movements for the sides, let’s say they all add up to 40, and then we would divide by 12, since 12 sides are changed when a movement is made, and in a perfect case, 1 movement would decrease the score by 12. We would then round down (we could probably round up, but just to ensure the heuristic is admissible, we will round down), to get a score of 3. This would evaluate the cube to be three “movements” from being solved.

The only issue this would cause would be when the heuristic has a score that is under 12, which would evaluate the cube to being solved. In cases like this, we would round up (this would not be an overestimate since a cube can only be solved when the heuristic evaluates to 0 before the dividing, which can only occur when all sides have unique values).

While a full proof for our cube’s admissibleness is out of myself and Daniels scope, we can simulate a few examples our code ran to determines the solvability of the cube.

**A statement of what you learned from this assignment:**

**The who-did-what for you and your programming buddy -- in the comments on canvas; do not put your or others' names in the main submission, because FERPA.**