MATH482

SPRING 2017

Programming Project-minimal obstacles in Instant Insanity

Due Thursday April 27 session 26/30 (please note that I have extended the due date)

Design a computer program to search for "obstacles," if they exist, for Instant Insanity puzzles of size up to **32** cubes. **If an obstacle exists find the smallest one.** As mentioned in class, by obstacle I refer to a subset of cubes which cannot be part of any solution. For instance if a given puzzle has two monochromatic cubes (of the same color) then that puzzle has an obstacle of size two. I will provide you with several puzzles two days prior to the due date. In the meantime you can build your own puzzles to test your algorithm and program. You have free reign over what type of algorithm to implement, and what type of programming language to use. Please turn in your source code along with your results. A brute-force algorithm may work fine for puzzles without solutions of size less than 10 cubes. But getting to size 30 the combinatorial explosion of possibilities may require a more nuanced approach. Particularly insightful algorithm design may qualify for extra credit. Note that observing that a particular color only shows up three times is not the same as identifying the smallest obstacle. For this project we are looking for obstacles, not solving puzzles that have solutions. Of course it is possible (although unlikely) that one of the randomly built puzzles below actually *has* a solution. In that case, simply provide the solution.

As promised I am now supplying you with the data for the puzzles that I will grade.

The routine I describe will fill in the colors in sequence: (cube1 opposite pair 1 left, cube1 opposite pair 1 right, cube1 opposite pair 2 left, cube1 opposite pair 2 right, cube1 opposite pair 3 left, cube1 opposite pair 3 right, cube2 opposite pair 1 left, cube2 opposite pair 1 right, cube2 opposite pair 2 left, … , cube32 opposite pair 3 left, cube32 opposite pair 3 right).

The formula is

$1 + ((\text{floor } n\pi) \bmod 32)$, $1 \leq n \leq 192$, for puzzle one,

and

$1 + ((\text{floor } ne) \bmod 32)$, $1 \leq n \leq 192$, for puzzle two.

These formulas specify, exactly, which colors land on which faces, so each group will have the exact same data set.

Before you get to work searching for (smallest) obstacles I want you to analyze the "random" color assignment routine. **Please print out a hard copy of the input data so I can verify that you are solving the correct problem.**

A After how many color assignments (for what n) do you expect to have every color showing up on a face at least once? (Solve this by *either* theory or simulation. If you use theory be sure to provide a citation for any mathematical methods you are applying.)

B After all **192** color assignments how many colors do you expect will occur in the puzzle *less* than four times? The result of this will all but ensure that the puzzle in question has no solution-and perhaps, a relatively small obstacle. (Use theory or simulation.)

C Please provide an English description of your algorithm for determining the smallest obstacle. Please also provide your source code, and output. The key is an obstacle of minimal size. Be sure to include this in your report.

D As I said earlier, extra credit will be awarded for a particularly clever or insightful algorithm (beyond brute-force). Also you might compare your input with other groups, just to make sure you are dealing with the same puzzles. If your input is wrong I cannot give full credit!

E Finally we may not have time left in the semester for another programming project. Then all the credit for the programming component will have to come out of this one project. Please use the full power and resources of your group to nail out this programming assignment.