**Final Project**

*E2: modeling, algorithmic solution, program implementation, and scalability analysis*

Eternity 2 (E2) is an edge-matching puzzle (EMP) made of 16 x 16 tiles.



**Description**

The E2 puzzle has a solution where all tiles are used, and none of them have an edge that does not match an adjacent edge. I would like to generalize this problem to n x n (square) EMPs of size n = 2 to 16. This allow you to study how any solver that you devise scales-up in time (computations) and space (memory) as n increases.

Your challenge is easy to describe, hard to meet.

1. **Problem**. Describe the problem in detail in *your own language*. Put the problem in a form that (you think) renders it amenable to an algorithmic solution of your own making.

2. **Modeling**. Provide a *precise description*, using a) data structures (perhaps with graphic illustrations) and b) algorithms (perhaps via precise pseudo-codes or flow charts) of a model that represents a framework of a first attempt at a working solution, then iterate, to make sure that you always have a precise description of your latest implemented solution: see below.

3. **Solution**. Code your solution using computer programs (in Java or C++). Also, employ methods allowing you to a) measure not just the perfection of your solution (e.g., no. of mismatching edges) but more importantly, b) the *time* (or rather, the no. of basic calculations) it takes to find a solution, and the *memory* (in bytes) required for such a solution-- always as a function of some scaling variable (typically, n). Present your best solutions(for each n attempted), and provide graphs that demonstrate how computation and memory scale-up with n; you will have to explain/attempt to explain those in analysis (below).

4. **Analysis**. *Critically* assess the results achieved; provide explanations or hypotheses for your ability/inability to find some optimal solutions (for different n values) and for the particular manner in which computational time and space scaled-up with n. Were those results a fluke of hardware/programming or are they a property of an underlying mathematical truth, or what?

**Marking Criteria**

½ the Project Mark is for Program (evaluated by Marker)

* 10% Program **Correctness**: the program does what it’s supposed to do and does it without crashing, given valid inputs.
* 20% Algorithm **Originality**: though ideas for the program can be borrowed from a multiplicity of external sources, the source code itself (but for external libraries with functions implementing *basic* data structures and algorithms) must be written and understood by the student him/herself.
* 35% Implementation **Quality**: especially, the appropriateness and quality of representation, evaluation functions (for effectiveness and efficiency), add/delete/modify mutations (and how they’re used), selection criteria (for the new individual).
* 35% **Output** (during the demo): this includes both output *correctness* and *speed* of recall, which should not exceed seconds, not minutes.

½ the Project Marks is for Report (evaluated by Instructor)

* 10% **Problem Description**
* 30% **Modeling**
* 30% **Solution**
* 20% **Analysis**
* 10% **Overall** **Quality of Writing + Formatting**

**Material**

I have provided attachments that allow you to tackle E2-like problems of different sizes: these are text files with numerically-coded collections of tiles. However, you are ultimately responsible for using the same tiles as those of E2 itself.