BE/CS 196a Homework Assignment 1

1 Design a winner-take-all neural network (10 pts)

> You will present the design in class, all students will vote for their favorite design, and the winning design will gain 10 bonus points and be experimentally implemented in later labs.

Design a winner-take-all neural network that has two distinct memories. Use what you learned about winner-take-all neural networks from Lecture 02, with help from the online software tool WTA Compiler. Note: In your design, the "number of WTA groups" should be set to 1 in the WTA Compiler. The concept of grouping in winner-take-all neural networks will be introduced later in the class.

- Design memories: Design two distinct binary patterns that contain the same number of ones. Each pattern should have up to 100 bits with up to 20 ones. The overlap between the two patterns (i.e. shared number of ones divided by the total number of ones) should be no more than 40%.
  - Describe the intended meaning of the two patterns and show a diagram for each pattern, using pixels with distinct colors to indicate zeros and ones.
  - Specify the total number of bits, the number of ones, and the overlap of the patterns.
- Define a winner-take-all neural network mathematically.
  - Specify a binary weight matrix for a winner-take-all neural network that remembers the two patterns that you designed.
  - Give variable names to the inputs and outputs of the neural network, and specify a mathematical formula for computing the outputs from arbitrary inputs.
- Design tests: Design eight test patterns that are representative in terms of both how easily a pattern can be recognized by the neural network and how corrupted a pattern is compared to the memories.
  - Show the test patterns in their weighted sum space, and put them in order of how easily they can be recognized.
  - Specify the number of corrupted bits in each pattern, counting a one flipping to a zero and vice versa.
- Understand the difference between binary and analog weights.

- Given the target patterns that you designed, would a neural network perform better if analog weights are allowed?
- If so, provide an example analog weight matrix for recognizing the target patterns, and explain how the performance improves with the test patterns. If not, explain why.

2 Design a reconfigurable tile array (10 pts)

> You will present the design in class, all students will vote for their favorite design, and the winning design will gain 10 bonus points and be experimentally implemented in later labs.

Design a tile array that can be assembled from just a few types of tiles and reconfigured through the mechanism of tile displacement. Use what you learned about reconfigurable tile arrays from Lecture 03, with help from the example Mathematica notebook provided with this homework assignment.

- Design a finite-sized tile array composed of up to seven types of square tiles that have no more than four unique pairs of edge interactions. For example, the 11-tile heart shape shown in Fig. 1d can be assembled from four types of tiles that have three unique pairs of edge interactions illustrated in Fig. 1b. Each tile edge can be assigned as "non-interactive", "giving", or "receiving". Due to the DNA implementation that we will explore later in the class, the identity of a receiving edge (e.g. $1^\star$ in tile2, $2^\star$ in tile3, and $3^\star$ in tile4) must match the edge numbers shown in Fig 1a, while the identity of a giving edge can be arbitrary.
  - Show a diagram of your designed tile set and array, and specify the total number of tiles, tile types, and unique pairs of edge interactions.
  - List the receiving edges in all tile types and verify that their identities agree with the edge numbers shown in Fig 1a.
  - Evaluate if the designed array is the only shape that can be created allowing all possible tile-tile interactions.
  - If not, show a diagram of all alternative structures that can be assembled from the same tile set.
- Design a pattern on each tile that continues into neighboring tiles throughout the array. You may take inspiration from Truchet tiling and the game Tsuro (here's a video on how to play the game).
  - Show a diagram of your tile set and array with designed pattern.

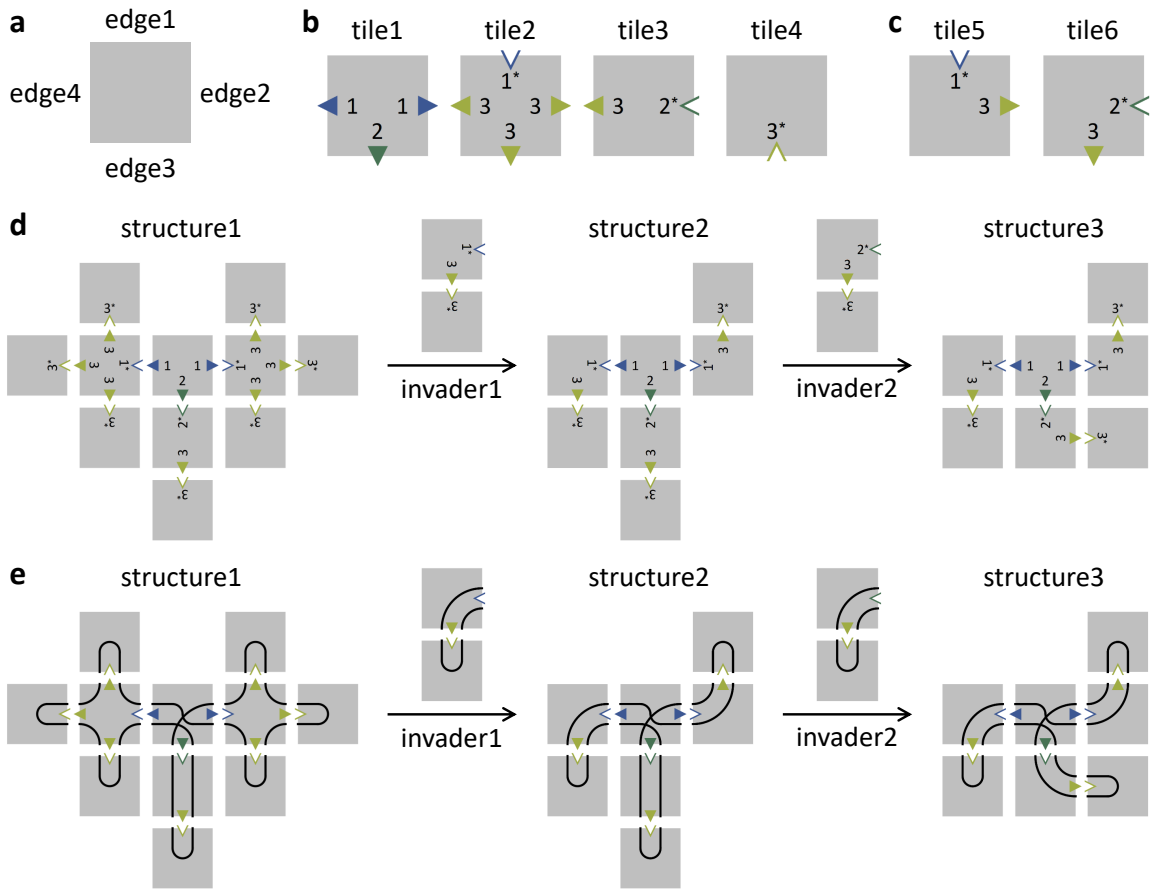

Figure 1: **An example design of a reconfigurable tile array. a,** A square tile with four programmable edges. **b,** Four types of tiles designed for the assembly of a heart shape. **c,** Two additional types of tiles designed for creating invaders that reconfigure the heart into a chair. **d,** A tile-displacement process with two invaders that are present either sequentially or simultaneously. Invader1 is composed of tile5 and tile4, and invader2 composed of tile6 and tile4. **e,** An example pattern design.

  - Explain if there are any desired properties of the overall pattern on the entire array that you have designed for, and how these properties arise from the patterns on individual tiles. For example, a desired property of the patterns shown in Fig 1e is a single loop with two crossings.
  - If your design allows multiple choices of patterns at specific locations in the array, for example through rotatable tiles with edge symmetry and pattern asymmetry, show a diagram of the array with all possible patterns and explain if there are any desired differences across the patterns that you have designed for.

- Design a set of invader tiles that reconfigure the shape and pattern of the array through tile displacement. The total number of tile types in the original and reconfigured structures should not exceed ten.

- Show a diagram of your invader tile set, the structures of multi-tile invaders if there are any, intermediate products of reconfigured structures if there are any, and the final structure.
- If your design allows multiple orders of reconfiguration steps, show a diagram of the tile displacement processes with all possible orders and explain if there are any desired differences across the processes that you have designed for.

3 Verify the function of a dual-rail circuit (5 pts)

Use Mathematica to verify that the dual-rail circuit generated by Seesaw Compiler in Lab 02 correctly computes the floor of the square root of a four-bit binary number, similar to the verification of the original Boolean logic circuit that was practiced in Lab 02. Turn in your Mathematica notebook that contains the verification.

4 Analyze a DNA-based square root circuit (5 pts)

Turn in the Mathematica package from Lab 02 that contains your modification for simulating the molecular kinetics of the DNA-based square root circuit with varying inputs. You may use Grid to display the plots for all possible inputs with valid dual-rail values. Add text to explain how the kinetic trajectories are interpreted as output values and whether they correctly compute the desired function of the input.

5 Simulate a DNA-strand-displacement reaction (5 pts)

Turn in your Mathematica notebook for modeling and simulation of a DNA-strand-displacement reaction from Lab 03.

6 Automated protocol design (5 pts)

Write Mathematica functions for automatically generating annealing, dilution, and mixing protocols from Lab 03, with variable species names, concentrations, and volumes. Turn in your Mathematica notebook that contains the functions.