Homework 3 — Squeak — due Monday 25 February

Total number of points available on this homework is 200. Full credit is equivalent to 100 points.

3.1 Tic-tac-toe (70pts)

Write a Squeak program that allows a human user to play tic-tac-toe against the computer. Either the computer or the human can move first. The computer may choose its moves using any policy whatsoever, including random.

3.2 Smarter tic-tac-toe (30pts)

Same as above, but the computer should use a policy that guarantees it will not lose.

- precisely formulate a policy for the computer (10pts)
- prove that the policy guarantees the computer will not lose (10pts)
- implement the policy in Squeak (10pts)

3.3 Extra credit: The 16-puzzle (30pts)

Write a Squeak program that allows a human user to play the 16-puzzle, starting from randomly reshuffled board configurations.

3.4 Extra credit: 16-puzzle again (40pts)

Implement an additional mode in the 16-puzzle problem above, in which the computer solves the puzzle itself.

- Use a 1-second delay between successive moves, so that the moves can be observed. (5pts)
- Without any delay between moves, report the number of moves the computer took, and the elapsed time. Do a series of tests with at least 30 random initial board configurations, and report the number of moves and elapsed time. Your goal is to devise a solution strategy that minimizes either the number of moves or the elapsed time. (35pts)

3.5 Extra credit: Safe transmission codes and molecules (30pts)

This builds upon exercises 1.4 and 2.4. There we had the following constraint, called the *direct matching constraint*: the Hamming distance between any two strings in the chosen subset must be greater than D. In molecular computation systems, there is often another constraint, called the *reverse complement constraint*: for any two strings x and y, $H(x^R, y^C) \geq D$. Here x^R denotes the reverse of the string x, and

y^C denotes the complement. For strings over the alphabet A, G, C, T, the complement is defined position-wise, and the complements of the individual symbols are defined as follows: $A^C = T, G^C = C, C^C = G, T^C = A$.

(We could also define the Watson-Crick complement of a string as $x^W = (x^C)^R$, and then the constraint is $H(x^W, y) \geq D$. The *reverse complement constraint* expresses our desire that strands of oligonucleotides should not unintentionally bond with strands that are not exactly their Watson-Crick complements but are very similar to them. A molecular computation should use a set of non-interfering strands. On the other hand, the *direct matching constraint* simply expresses the idea that separate reactions that occur simultaneously in solution should have sufficiently distinguishable products.)

Construct sets of oligonucleotides as large as possible, given word length N, the fixed alphabet A, G, C, T with $K = 4$, and minimum distance D. Try three different cases: impose the direct matching constraint only, impose the reverse complement constraint, and impose both constraints.

You may find useful the following link:
`http://www.cs.unm.edu/~darko/classes/2002s-451/amit99.ps`.

Write your code in Scheme, Smalltalk, Prolog, ML, or C.

How to turn in

Turn in your code by running

~dmykola/handin your-file

on a regular UNM CS machine.

You should use whatever filename is appropriate in place of your-file. You can put multiple files on the command line, or even directories. Directories will have their entire contents handed in, so please be sure to clean out any cruft.

Special considerations for Squeak:

For each exercise, write a class method testIt that does all the work necessary. In the header of the method write a comment such as "TicTacToe testIt.".

Please do not turn in an entire directory with Squeak images—it would be too big. Instead, group all the classes you have created into a single class category, called CS451HW3, and file out. Then submit the file `CS451HW3.st`.

Thus, the grader will start a fresh Squeak, file in the file `CS451HW3.st`, open a system browser, find the category CS451HW3, and for each of the exercise classes, find its class method testIt, highlight the text in the comment, such as "TicTacToe testIt.", and then doIt.