COSC 5P01 – Coding Theory
Winter 2020
Assignment #2

Due Date: March 26th, 9:30am (at the start of class)
This assignment accounts for 20% of your final grade and is worth a total of 100 marks.

In this assignment you are given the challenge of attempting to establish bounds on the size of optimal error-correcting codes. You can choose to apply this to codes defined using Hamming distance, edit distance or insertion-deletion distance. The problem definition is as follows.

A $(n, M, d)_q$ code is a set of M q-ary codewords of length n such that every pair of codewords are distance at least d from each other. The value M is called the *size* of the code and the value d is called the *minimum distance* of the code for the chosen distance measure.

Given n, d and q, we wish to determine the largest possible value of M for which there exists a $(n, M, d)_q$ code. A $(n, M, d)_q$ code is *optimal* if M attains this largest possible value. For Hamming distance codes, this largest value is denoted as $A_q(n, d)$. Equivalently, for edit distance codes we have $E_q(n, d)$ and for insertion-deletion correcting codes we have $D_q(n, d)$.

Tables of best-known values for $A_q(n, d)$ may be found here:

- Binary: http://www.win.tue.nl/~aeb/codes/binary-1.html
- Ternary: http://www.win.tue.nl/~aeb/codes/ternary-1.html
- Quaternary: http://www.win.tue.nl/~aeb/codes/quaternary-1.html
- 5ary: http://www.win.tue.nl/~aeb/codes/5ary-1.html

Equivalent tables for $E_q(n, d)$ may be found at http://www.cosc.brocku.ca/~houghten/edit.html and for $D_q(n, d)$ at http://www.cosc.brocku.ca/~houghten/insdel.html.

Note that while there are exact values for $A_q(n, d)$, $E_q(n, d)$ and $D_q(n, d)$ for some parameter sets, many others have upper and lower bounds only. You can choose to work on either upper or lower bounds. For edit distance codes and insertion-deletion correcting codes, you can also choose to work on fixed-length or variable-length codes. You are free to choose any method that you think is appropriate including those discussed in class. Generally speaking, if you wish to lower an upper bound then you will require an exhaustive search with mathematical arguments, while if you wish to raise a lower bound then you may consider artificial intelligence techniques (for example evolutionary algorithms).

Important note: Naturally, you are not expected to establish **new** bounds (because this is very hard!) but rather to develop a working program that will either:

1. Show that a $(n, d)_q$ code with more than M codewords *cannot* exist, for some chosen values n, M, d and q (by an exhaustive search for such a code), or
2. Show that a $(n, d)_q$ code with at least M codewords *does* exist, for some chosen values n, M, d and q (by producing such a code).

You can choose any value of q that you wish. While there are fewer words to consider for binary codes, and they have an easier representation, they are also *much* more studied than ternary, quaternary (etc) so it will be harder to approach the values in the tables.

Recommendations:

If you wish to use a particular approach for this assignment then you are more than welcome to discuss this with your instructor.

Regardless of the approach chosen, given a large enough value of *n* your program will take an extremely long time to run! It will also require huge amounts of memory. I strongly recommend you start with codes of small lengths to get comfortable with the problem first. After this, try your program on incrementally larger lengths.

You should also think carefully about representation of the words. You will save a lot of space and also time if you store words as integers (bit-vectors) and use bit manipulations for the calculations if possible.

Save plenty of time for proper analysis of your results, and to write the description of your approach and your summary.

Submission Requirements:

All of the following must be handed to the instructor directly:

1. A cover sheet, available from `http://www.cosc.brocku.ca/forms/cover`, completely filled out. Your assignment will not be marked unless one is submitted with the assignment.
2. A 1-2 page description of the approach used to solve the problem.
3. A 2-3 page summary of results/conclusions that you can draw from using the chosen approach. This should describe the most difficult parameter set(s) that your program was able to solve. It should also include a description of possible future improvements.
4. Commented and properly documented printouts for all of your program(s).
5. You must also submit your assignment electronically. To do this, create a directory on Sandcastle containing all files for this assignment, and run the script `submit5p01` from this directory.