Week 2: Fine Finite Computation

The purpose of this assignment is to develop your understanding of finite computation, focusing on the material in Chapter 3 of the textbook and what we covered in Class 5 and Class 6. We have also included some problems to provide more practice with proof techniques including induction, since it is apparent from Problem Set 1 that most students will benefit from these.

Collaboration Policy: You should work on the problems yourself, before discussing with others, and with your cohorts are your cohort meeting. By the Assessed Cohort Meeting, you and all of your cohortmates, should be prepared to present and discuss solutions to all of the assigned problems (including the programming problems). In addition to discussing with your cohortmates, you may discuss the problems with anyone you want, and use any resources you want except for any materials from previous offerings of this course, which are not permitted.

Problem 1 *Cohortommendations*

Reflect back on your experience from last week. Consider, but do not share your answers to, these questions:

1. Did you employ any of the video watching tips? Did you find any of them helpful? Did any have a negative impact on your learning? Are there any you think might help if you employed them in the future?
2. How prepared were you for your first cohort meeting? Should you have done more to prepare in advance?
3. Did you find your interactions with your cohort mates to be fairly natural? How can you personally help to improve your group dynamic for future semesters?

Next, share and discuss these questions with your cohorts:

1. What is working well and what is not working for the course so far?
2. What are things the course staff can do to make things better?
3. What are things you or your cohort can do to improve your ability to interact and learn that you think might be helpful for other cohorts to know about?

Once you've discussed these with your cohort, have one person share at least one of your answers to question 3 on the `#week2` discord channel. You should read the other cohorts answers there also, of course, and if you can, connect your suggestion to other cohorts' answers.

Problem 2 *Infinite Dominoes*

A domino is a tile with an unordered pair of numbers on it (e.g. $0,5$ or $3,3$). Dominoes come in sets containing all pairs of natural numbers less than or equal to some upper bound.

A pack of "double 6" dominoes will contain all unordered pairs of values from the set $\{0,1,2,3,4,5,6\}$ (there will be 28 = 7 + 6 + 5 + 4 + 3 + 2 + 1 total). A pack of "double 3" dominoes will contain all unordered pairs of values from the set $\{0,1,2,3\}$ (there will be 10 total).

A *domino chain* is a sequence of dominoes ordered so that the second value of each domino matches the first value of the next. The domino sequence $(1,2)(2,5)(5,5)(5,0)$ is a valid domino chain, whereas $(1,2)(2,5)(5,5)(0,0)$ is not.

Consider a pack of "double $\mathbb{N}$" dominoes, which contains all of the infinitely-many unordered pairs of natural numbers. Show that there is an uncountable number of infinite-length domino chains that can be constructed from a pack of "double $\mathbb{N}$" dominoes.

Problem 3 *Cantor's Proof*

The *Cantor's Shocking Proof* video presents a proof by contradiction that for any set S, $|\text{pow}(S)| > |S|$. While this may seem obvious for finite sets (see Week 1 Problem 6), Cantor's proof shocked the math world because it also applied to infinite sets, demonstrating for the first time that infinite sets can have different cardinalities.

For this problem, we will be checking that you understand the proof. Be prepared to answer questions like "What statements from the proof are contradictory?", "How is the set A defined?", "Why didn't this proof require the sets to be finite?".

Problem 4 *Straightline Programming*

For this problem, you will be asked to implement several straightline programs in Python. You must adhere to the rules of straightline programs. In particular, make sure:

– Variables contain only bits.

– Function parameters are only the characters `'0'` and `'1'`.

– Each line may only have a single function and a variable assignment (no nested functions, no conditionals, no loops).

– Variable names may not be reused within the same function.

We provide test cases for all functions. If you followed the above rules, and you pass all tests, you can be confident that your implementation is correct.

To complete this problem, download the straightline.py program. The comments in that file guide you through the assignment. Everything you must implement is marked with a comment containing "`TODO`".

Problem 5 *Compare 4 bit numbers* (Exercise 3.1 in TCS book)

Draw a Boolean circuit (using only *AND*, *OR*, and *NOT* gates) that computes the function $CMP_8 : \{0,1\}^8 \to \{0,1\}$ such that $CMP_8(a_0,a_1,a_2,a_3,b_0,b_1,b_2,b_3) = 1$ if and only if the number represented by $a_0a_1a_2a_3$ is larger than the number represented by $b_0b_1b_2b_3$. We will say that a_0, b_0 are the most significant bits and a_3, b_3 are least significant.

Problem 6 *Compare n bit numbers* (Exercise 3.2 in TCS book)

Prove that there exists a constant c such that for every n there is a Boolean circuit (using only *AND, OR*, and *NOT* gates) C of at most $c \cdot n$ gates that computes the function $CMP_{2n} : \{0,1\}^{2n} \to \{0,1\}$ such that $CMP_{2n}(a_0 \cdots a_{n-1}b_0 \cdots b_{n-1}) = 1$ if and only if the number represented by $a_0 \cdots a_{n-1}$ is larger than the number represented by $b_0 \cdots b_{n-1}$.

In other words, generalize the previous problem to describe how to compare n-bit numbers for any specific value n using *AND, OR*, and *NOT*. The total number of gates used should be upper bounded by some constant c times n (i.e. asymptotically linear).

Problem 7 NOR *equals* AON (based on Exercise 3.7 in TCS book)

Let $NOR : \{0,1\}^2 \to \{0,1\}$ defined as $NOR(a,b) = NOT(OR(a,b))$. Prove that $\{NOR\}$ is equivalent to *AND, OR, NOT*. In other words, show that any function that can be computed by *AND,OR,NOT* can also be computed using just *NOR*, and vice-versa.

Problem 8 *XOR does not equal* AON (based on Exercise 3.5 in TCS book)

Prove that the gates *XOR*, 0, 1 is *weaker* than *AND, OR, NOT*. (You can use any strategy you want to prove this; see the book for one hint of a possible strategy, but we think you may be able to find easier ways to prove this, and it is not necessary to follow the strategy given in the book.

