WEEK 03

1. Preparation for Assignment

If, and *only if* you can truthfully assert the truthfulness of each statement below are you ready to start the assignment.

1.1. **Reading Comprehension Self-Check.**

- I know that *brute force* is a straightforward approach to solving a problem, usually directly based on the problem statement and definitions of the concepts involved.
- I understand that a first application of the brute-force approach often results in an algorithm that can be improved with a modest amount of effort.
- I know *why* it is **false** to say that a strength of the brute-force approach is subpar algorithmic efficiency.
- I understand that *exhaustive search* is a brute-force approach to combinatorial problems that suggests generating each and every combinatorial object of the problem, selecting those of them that satisfy all the constraints, and then finding a desired object.
- I know why it is **false** to say that exhaustive search is practical for all but very small instances of problems it can be applied to.
- I have reviewed Appendix A specifically focusing on understanding the formulas and rules for summations.
- I have studied and pondered these common algorithms and understand why they have the time efficiencies they do:
  - Finding the maximum element in an array.
  - Determining the uniqueness of all elements of an array.
  - Multiplying two n-by-n matrices.
  - Converting a base 10 number to binary.

1.2. **Memory Self-Check.**

1.2.1. *Algorithm Efficiency Calculation.* Create brute force algorithms for each of these three situations:

(1) Computing a^n, where a is positive and n is a nonnegative integer.
(2) Computing $n!$.
(3) Sequential search.

. Determine $\mathcal{O}$, Ω, Θ, input size, and time efficiency for each algorithm.

Date: September 28, 2018.

2. Week 02 Exercises

2.1. **Exercise 1 on page 102.**

2.2. **Exercise 8 on page 103.**

2.3. **Exercise 6 on page 114.**

2.4. **Exercise 4 on page 128.**

2.5. **Exercise 8 on page 129.**

2.6. **Find the Door.** You are facing a wall that stretches infinitely in both directions. There is a door in the wall, but you know neither how far away nor in which direction. You can see the door only when you are right next to it. Design and write code for an algorithm that enables you to reach the door by walking at most $\mathcal{O}(n)$ steps where n is the (unknown to you) number of steps between your initial position and the door. (Hint: walk alternately right and left going each time exponentially farther from your initial position.)

3. Week 02 Problems

3.1. **Exercise 14 on page 103.**

3.2. **Exercise 5 on page 129.** Make sure you do a rigorous mathematical proof.

3.3. **Exercise 6 on page 121.** Write code for this algorithm.