CS 306	Algorithms and Complexity

Vastness

Capstone *Exploration*

Goal

The purpose of this exploration is for you to apply what you've learned to synthesize and analyze an algorithm for solving a deceptively simple problem, vast in its extent.

Requirements

Create an elisp program for solving the following problem:

Given a positive integer n, is there a power of 2 that has a specified n-digit number as a prefix?

While simple to state, there are numerous questions this problem raises, not the least of which is, is it tractable? With certain 1-digit numbers (1 through 6, and 8) the answer is trivially yes, by mere inspection of a few small powers of 2:

$$\begin{aligned} 1 &= 2^0 \\ 2 &= 2^1 \\ 32 &= 2^5 \\ 4 &= 2^2 \\ 512 &= 2^9 \\ 64 &= 2^6 \\ 8 &= 2^3 \end{aligned}$$

A program is certainly not needed to find those! And for the single digits 7 and 9, a determined search will eventually find $70368744177664 = 2^{46}$ and $9007199254740992 = 2^{53}$, which are the smallest numbers that fit the bill. But what about 2-digit numbers? 3? 4? 27?!

One of the goals of this class was for you to become conversant with the topics and issues surrounding algorithms and complexity. Recall that these included:

- Basic algorithms analysis: Asymptotic analysis of upper and average complexity bounds.
- Best, average, and worst case behaviors.
- Big-Oh, little-Oh, Big-Omega, and Big-Theta notation.
- Standard complexity classes.
- Empirical measurements of performance.
- Time and space tradeoffs in algorithms.

Together with the new patterns of thinking and mind tools you have acquired throughout the semester, you should call upon for this "capstone" exploration all of your prior intuition and creativity and *curiosity* — so treat this as a superlative gersy principle application opportunity!

The following is a partial list of questions you should address—thinking of others is part of the creative challenge:

1. What data structures are appropriate?
2. What algorithm design technique(s) is/are best.
3. What form should your program's input and output take?
4. What is/are the efficiency/complexity class(es) of your algorithm(s)?
5. What is the actual running time for some test runs?
6. Is this decision problem NP-Complete?
7. What problem sizes are feasible?
8. Is there hope of ever finding your "personal" power of 2, if it even exists? For example, the smallest power of 2 prefixed by your 9-digit ID number or 10-digit phone number? What about a 27-digit number consisting of the 9 digits of your Social Security number, followed by the 8 digits of your birth date (2 for month, 2 for day, 4 for year), followed by the 10 digits of your cell phone number?

Grading Criteria

Grading for this exploration is based on application and engagement (whether or not or to what extent you figured out the hows and the whys), correctness and completeness, and creativity and efficiency. Details, as always, will be found in the self-assessment.