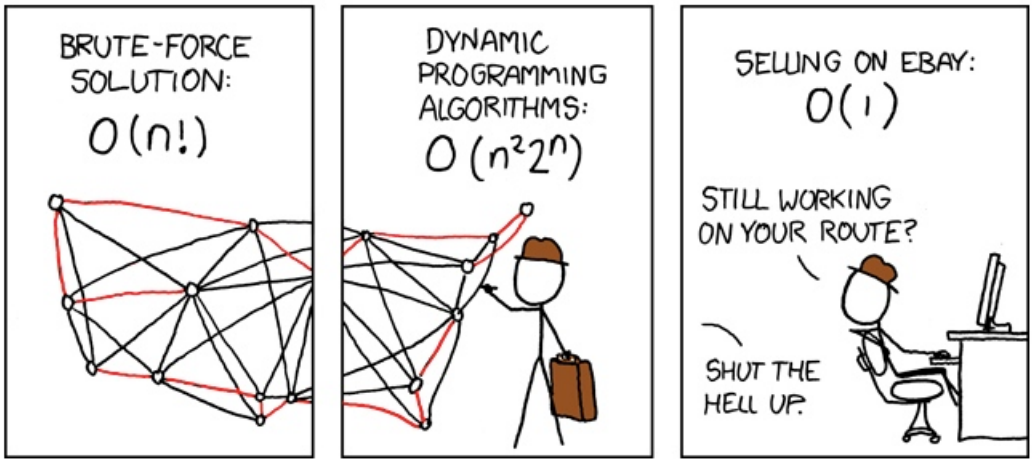
CS140 - Assignment 1

Due: Sunday, 9/8 at 11:59pm



http://xkcd.com/399/

This problem set should be done with a *different* partner than you worked with last week. If you would like help finding a partner, reach out ASAP.

You must use LaTeX to format your solution; one person should upload the pdf to gradescope.

1. **[10 points]** The Geometric Series

   (a) Use induction on n to show that for all integers $n \geq 0$

   $$1 + a + a^2 + a^3 + \ldots + a^n = \frac{a^{n+1} - 1}{a - 1}$$

   where a is some arbitrary real number other than 1. Please make sure to write your proof carefully, with a base case, induction hypothesis, induction step, and conclusion.

   (b) Explain where your induction proof relied on the fact that $a \neq 1$.

   (c) What does the sum evaluate to when $a = 1$?

2. **[40 points]** Asymptotics

   Indicate whether each of the following statements is true or false and then carefully **prove** your answer using the formal definition of Big-O notation and properties of logarithms from the last assignment *(hint, hint)*. Remember that to show that one of these statements is false, you must obtain a formal contradiction; it does not suffice to just say "false".

One useful summation:

$\Sigma_{i=1}^{n} \frac{1}{i} = \ln n + c$ where c is a constant

(a) 2^{n+1} is $O(2^n)$.

(b) 2^{2n} is $O(2^n)$.

(c) $3n^2 \log_2 n + 16n$ is $O(n^3)$.

(d) $25 \log_2 8n^{10}$ is $O(\log_{10} n)$.

(e) $8^{\log_2 n}$ is $O(n^3)$.

3. **[12 points]** Solving Recurrences

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible (in other words, give Θ bounds where possible) and make sure to justify your answers. (Note: you may want to read appendix A.1 in the textbook, which has some summation properties.)

(a) $T(n) = 4T(n/2) + cn$

(b) $T(n) = T(n-1) + n$

(c) $T(n) = T(n-1) + 1/n$

(d) $T(n) = T(9n/10) + n$

4. **[3 points]** An improvement?

You decide to come up with a new sorting algorithm called <*your_name_here*>sort. You notice that *InsertionSort* seems inefficient in how it finds the correct location to insert the next value. You decide that rather than linearly searching one at a time to find the correct place, you use binary search to find the correct place. Is this an improvement? If yes, state the running time of this new algorithm. If no, explain why this is not an improvement. Be specific and clear.

5. **[5 points]** Intersection

Describe an algorithm and *state the worse case running time* (in terms of Θ or O where appropriate) to solve the following problem: given two lists of numbers A and B of lengths m and n respectively, return the intersection of the lists, i.e. all those numbers in A that also occur in B. You can use procedures that we've discussed in class, but no others (e.g. no hashtables). You can assume that in any given list, the numbers are unique. You will be graded on the efficiency of your solutions.