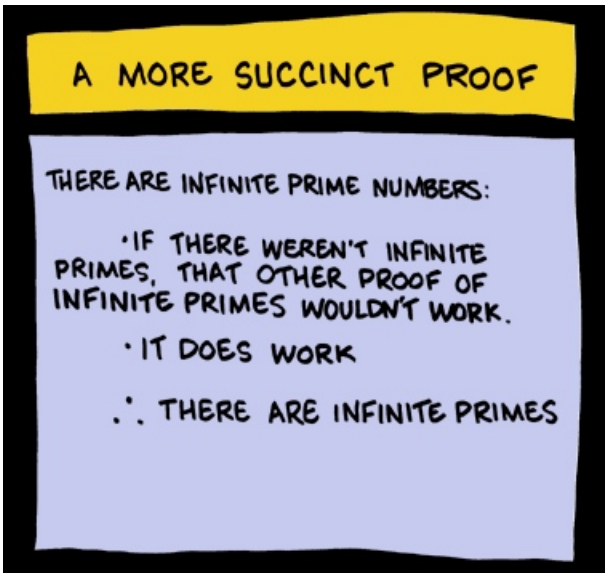
CS140 - Assignment 3

Due: Sunday, Sept. 18 at 11:59pm



http://www.smbc-comics.com/index.php?db=comics&id=1099

- You must work on this assignment with a partner, though it does not have to be within your learning group. There are an odd number of people in the class right now, so I will allow one group of three, but you need to get permission from me first.
- You must use LaTeX to format your solution.

1. **[8 points]** Give the asymptotic bounds for each of the recurrences below. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible. If you use the master method, you must specify Θ bounds, but only need to specify O if you use another approach.

   (a) $T(n) = 9T(n/3) + n^2$

   (b) $T(n) = 2T(n/2) + n^3$

   (c) $T(n) = 3T(n/2) + n \log n$

   (d) $T(n) = T(n-2) + n$

2. **[15 points]** Sorting Partially Sorted Data

   (a) You are given an array of n elements to sort. The good news is that the array is already partitioned into n/k blocks of k elements each. The elements in the first block (elements at array indices 1 through k) are unsorted, but they are *all* less than the elements in the second block (elements at array indices $k+1$ through $2k$), and so forth. In other words, each of the n/k blocks is unsorted, but the elements in each block are strictly smaller than the elements in the next block.

   Prove that any comparison-based sorting algorithm that receives this kind of "partially" sorted data has a lower bound of $\Omega(n \log k)$ on its worst-case running time.

   Note: It is not at all rigorous nor correct to simply combine the $\Omega(k \log k)$ lower bounds for sorting each of the n/k blocks! To see why, observe that a skeptic could rightfully ask if there might not be a special clever algorithm that exploits the information about the blocks to do better than we would without knowing that information. A rigorous proof will need to follow the paradigm that we used in class to get the lower bound for sorting in general.

   (b) Briefly describe how such an array (an array of length n with n/k subsequences such that the elements in each subsequence are all smaller than the elements in the next subsequence) can be sorted in time $O(n \log k)$. From the first part of this problem, you can now conclude that your algorithm is asymptotically optimal!

3. **[25 points]** Stock Market Problem

   You're given an array of numbers representing a stock's prices over n days. You goal is to identify the longest consecutive number of days during which the stock's value does not decrease. For example, consider the stock values below:

Day:	1	2	3	4	5	6	7	8
Value:	42	40	45	45	44	43	50	49

   In this example, the length of the longest consecutive non-decreasing run is 3. This run goes from day 2 to day 4.

   (a) Briefly describe a very simple "naive" algorithm for this problem and explain why the worst-case running time is $\Theta(n^2)$.

   (b) Describe a divide-and-conquer algorithm whose running time is asymptotically better than $\Theta(n^2)$. Provide pseudocode and/or a clear English description of your algorithm. (Note that your algorithm must be a divide-and-conquer algorithm. And yes there are non-divide-and-conquer algorithms that are very, very good!)

   Hint: Like writing recursive functions, when trying to come up with a divide-and-conquer solution, assume that your algorithm works correctly on the divided parts. Then, how do you construct your answer to the overall solution?

   (c) Analyze the running time of your algorithm: what are tight bounds on the best and worst case behavior?

   (Note that next week's assignment will ask you to implement your divide-and-conquer algorithm.)