CS375 Problem Set 2

Due Oct 18th 10PM ET

A general note for CS375: When writing up your homework, please write neatly and explain your answers clearly, giving all details needed to make your answers easy to understand. Graders may not award full credit to incomplete or illegible solutions.

As a reminder, your solutions to these must be typed. You are encouraged to discuss the problems with your classmates, **but your submission should be your own.** When writing your solutions up, you should not be looking at any other student's submission.

Clear communication **is the point**, on every assignment. In general in CS375, unless explicitly specified otherwise, answers should be accompanied by explanations. Answers without explanations may not receive full credit. Please feel free to ask me any questions about explanations that might come up!

Necessary acknowledgements: many of these questions are reproductions or adaptations of questions of Eric Aaron's devising. Others come from CLRS, Erickson's text, or Kleinberg and Tardös' *Algorithm Design* (which is another fantastic algorithms text that I highly recommend perusing).

1. Consider the following 9 functions:

$$A(n) = 2A(\tfrac{n}{4}) + \Theta(\sqrt{n}) \quad B(n) = 2B(\tfrac{n}{4}) + \Theta(n) \quad C(n) = 2C(\tfrac{n}{4}) + \Theta(n^2)$$

$$D(n) = 3D(\tfrac{n}{3}) + \Theta(\sqrt{n}) \quad E(n) = 3E(\tfrac{n}{3}) + \Theta(n) \quad F(n) = 3F(\tfrac{n}{3}) + \Theta(n^2)$$

$$G(n) = 4G(\tfrac{n}{2}) + \Theta(\sqrt{n}) \quad H(n) = 4H(\tfrac{n}{2}) + \Theta(n) \quad I(n) = 4I(\tfrac{n}{2}) + \Theta(n^2)$$

   (a) For each of the above functions, determine the runtime in terms of Θ via the recursion tree method.

   (b) Use the master theorem to confirm your answers to $C(n)$, $E(n)$, and $G(n)$.

2. Consider the following pseudocode:

```
def Q(A[0, ..., n - 1]):
    // A is an array of real numbers, indexed 0 to n - 1
    if n == 1:
        return A[0]
    else:
        temp = Q(A[0, ..., n - 2])
        if temp ≤ A[n - 1]:
            return temp
        else:
            return A[n - 1]
```

   (a) What does this function compute? Explain your answer.

   (b) What is the runtime complexity of this function? Demonstrate your answer via a recursion tree.

3. Suppose you have a Java .class file (so not the .java file, you can't see the code) that can compute the median of an unsorted array in linear time ($\Theta(n)$). You can assume it was compiled by the same version of Java that you use, so you can run it freely. How can you use it to get a linear runtime algorithm for finding the i^{th} smallest item in a list in linear time? Write down pseudocode to solve this problem and show that the runtime is linear.

4. You've just made a **stack** of fresh pancakes and your inner computer scientist *demands* that you fix your stack so that it is sorted by size, with your biggest pancakes at the bottom and smallest at the top. However, with your spatula the only operation you can perform is inserting the spatula into the stack and flipping over the entire stack above where you inserted.

   (a) Describe an algorithm that will correctly sort any stack of n pancakes in $O(n)$ flips. *Exactly* how many flips does your algorithm perform in the worst case?

   (b) For every positive integer n, describe a stack of n pancakes that requires $\Omega(n)$ flips for any spatula-based sorting algorithm. (*hint: suppose two adjacent pancakes in the stack won't be adjacent after being sorted. What does each pair of such pancakes force?*)

5. Recall that we developed an algorithm for multiplying two n digit numbers in $n^{\log_2(3)}$ time.

   (a) Develop an algorithm for multiplying two n digit numbers by instead multiplying 6 numbers of $\frac{n}{3}$ digits: show that it has a runtime of $\Theta(n^{\log_3(6)})$. Observe that this is actually worse than $n^{\log_2(3)}$.

   (b) Improve your algorithm from a so that it performs one less multiplication. As a hint: if $X = x_2 10^{\frac{2n}{3}} + x_1 10^{\frac{n}{3}} + x_0$ and $Y = y_2 10^{\frac{2n}{3}} + y_1 10^{\frac{n}{3}} + y_0$, then you should compute

   - $(x_2 + x_1)(y_2 + y_1)$
   - $(x_1 + x_0)(y_1 + y_0)$
   - $(x_2 - x_0)(y_2 - y_0)$
   - $(x_2 + x_1 + x_0)(y_2 + y_1 + y_0)$
   - $(x_2 - x_1 + x_0)(y_2 - y_1 + y_0)$

   (I know this is kinda ugly, but it was the simplest way I could shrink it down, if you make a better with prettier coefficients then hats off to you)