CS375 Problem Set 1

Due Oct 4th 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.

1. First, a problem that routinely appears in interviews, and is quite related to the webcomic that appears on our course webpage. A group of four out on a hike needs to cross a bridge that appears to be very old: observing the ropes, some quick mental maths reveal that only two people can cross at a time safely. To complicate matters, the sun has set and it is very dark, and due to a bunch of missing planks in the bridge the family decides that anyone crossing the bridge must have a flashlight with them, meaning that someone will need to keep ferrying the flashlight back and forth in order to help people cross. Looking at the battery life left on their flashlight, they see the flashlight will only last for precisely another 17 minutes. Unfortunately, one member of the group has injured themselves and will take 10 minutes to cross the bridge. The others will take 1, 2, and 5 minutes to cross. Under the constraint that when two people cross together they must move at the speed of the slower person (to shine the flashlight adequately), will the group be able to cross the bridge in 17 minutes? Why or why not?

   Obviously this problem doesn't 'feel' very algorithms-y. However, I assert that it exercises the modality of thinking that is extremely important in this field, particularly with respect to constraint satisfaction.

   For this exercise, explain the full thought process by which you arrived at your answer! Or, if you arent able to find a full answer, explain the thought process as far as you get with your reasoning. For example, one might say First, we thought about sending [blah blah blah], but we then realized [blah blah blah]. Then, to address that, we thought [blah blah blah], but that didnt work because of [blah blah blah]. Because of that, we . . . . (Please try to avoid using the word blah in your answer!)

2. List the following functions of n according to their order of growththat is, how fast each function grows as n gets bigfrom lowest to highest:

$$(n-2)!, \quad 5(\log(n+100))^{10}, \quad 2^{2n}, \quad 0.001n^4+3n^3+1, \quad (\log(n))^2, \quad \sqrt[3]{n}, \quad 3^n$$

Although you dont need to explain every part of the ordering for this exercise, please give short explanations (12 sentences) for the following:

(a) how you know the second-smallest comes before the third-smallest; and

(b) how you know the second-largest comes after the third-largest.

3. Consider the following pseudocode:

```
// Input: An n×n matrix of integers A
mystery(A):
    for i = 0, ..., n-2:
        for j = i+1, ..., n-1:
            if A[i, j] ≠ A[j, i]:
                return False
    return True
```

(a) What does this algorithm do? Give an English description of what inputs lead to it returning True and what inputs lead to it returning False. (You do not need to give examples as part of your answer, but you are welcome to include example 2D arrays along with the English description, if it would make your answer clearer.)

(b) Using summation notation formulas (i.e., with Σs to represent summations), give a summation expression for the number of $\neq$ comparisons that are made *in the worst case* by this algorithm. Then, solve that summation that is, find an equivalent simple formula for that expression, showing the number of $\neq$ operations as a function of n.

(c) Using your answer to the above, what is the best Θ bound for this?

(d) Demonstrate the best case number of $\neq$ operations.

4. Design an iterative (i.e., without using recursion) algorithm to find all the common elements in two sorted lists of numbers. For example, for input lists [2, 5, 5, 5] and [2, 2, 3, 5, 5, 7], the output should be the list [2, 5, 5]. Specifically, your algorithm should look like:

```
computeSortedIntersection(A[0:m], B[0:n]):
    ...
    ...
```

(a) Use a loop invariant argument to demonstrate its correctness.

(b) Demonstrate its worst case runtime (I'm looking for a Θ bound here)

(c) Demonstrate its best case runtime (I'm looking for a Θ bound here)