CS 413: Analysis of Algorithms
Homework 2

Due on Sunday, 03/01/2020, 11:59 PM

This is an **individual** homework assignment. All solutions have to be **typed**. No credit will be received for handwritten solutions (except for figures or long equations that might be handwritten). You need to submit your assignment on BB in the related digital repository by the deadline.

As it appears in the course syllabus, for the homework assignments, students are encouraged to discuss the problems with others, but you are expected to turn in the results of your own effort (not the results of a friend's efforts)". Even when not explicitly asked, you are supposed to justify your answers concisely.

15 points

1. (a) **Coding**:
   Write a **a** (C++/Java) code that checks if a number is Palindrome (use **recursive** function as you code). A palindrome number is a number that reads the same from beginning to end and from end to beginning, in other words, a palindrome number remains the same when its digits are reversed. For example, 13431 is a palindrome number. 2332 is another one. (**Note**: Your program should receive an integer number as the input and always work with integer types, i.e., you cannot use string or char types throughout your code.)

   (b) **test cases**:
   You need to run your code for the following test cases and show the result obtained by the program. A screenshot of each run should be added to your solution.

1) 0	(output: yes)
2) 1234554321	(output: yes)
3) 123454321	(output: yes)
4) 1221	(output: yes)
5) 1234	(output: no)
6) 7676	(output: no)
7) -121	(output: yes)
8) -456	(output: no)

5 points

2. What is the time complexity of your algorithm in question 1? Cleary justify your answer. Explain enough.

10 points, 2 points for T/F. 8 points for the justification.

3. Assume you have functions T and S such that T(n) = O(S(n)). For the following statement, decide whether you think it is True or False. If you think the answer is True, prove it, otherwise, provide a counter example.

   T(n)+2S(n)=O(S(n))

10 points

4. One can obtain an asymptotically tight bound directly by computing a limit as n goes to infinity. Essentially, if the ratio of functions f(n) and g(n) converges to a positive constant as n goes to infinity, then f(n) = θ (g(n)). Let f and g be two functions that

$$\lim_{n\to\infty} f(n)/g(n)$$

exists and is equal to some number c > 0. Then f(n) = θ(g(n)).

Prove the above statement.

20 points

5. (a) Consider the (recursive) factorial function in the slides. Write a recurrence relation for the time complexity of the function. Justify your answer.

   (b) Analyze the time complexity of the relation obtained in part (a) in the worst-case in Big-O expression. Justify your answer again.

   (c) Write the non-recursive version of the factorial function. Analyze its time complexity in the worst-case in Big-O notation. Justify your answer.