Homework assignment 07

Please read these instructions carefully for each assignment, though they are generally do not vary between the assignments

1. You need to follow carefully the specific instructions for the assignment as written below.

2. For the first stage, each student must submit his/her solution set as a *single* pdf file to NYUClasses. You can submit a scanned pdf, or pdf converted from jpg photos. But multiple pdfs must be joined together. (Translation: have kindness for the graders, who have lives to live, and will have a lot of solutions to process.) Alternatively, you can produce a text-based pdf from a processing system such as MS word or (much better yet:) LaTeX. *Caveat:* If you submit handwritten solutions, the readability is your responsibility.

3. Leave 1.5 inches (or 4.0 centimeters) of space between each part of each problem for the second stage.

4. The submissions for the first stage are due on Tuesday, October 24 by 11:55pm.

5. Solutions will be posted or handed after the lecture on Wednesday, October 25.

6. Once you get the solutions you are required to self-criticize your original answers as described below. Based on the handout solutions, use space that follows each homework answer, as described in item 3 to write your own self-criticisms of your answers. Do *not* re-write your solution, add material to you original solution as written.

   A good self-criticism *briefly*

   - points explicitly at the mistakes of your solution,
   - summarizes the idea, step, or technique that you had missed.

   It is not enough to just mention some ideas from the correct solution, you have to explain how these ideas are related to your initial solution. If you think that your solution is correct, write this explicitly.

   Additional requirement: mark correct submissions with a green check, and write your self-criticisms in red. If you are uncertain if an answer is correct, say so in blue.

7. The submissions for the second stage are due to Friday, October 27 by 11:55pm.

8. For every problem below, the number of points is specified as "$a + b$ points". This means that you get from 0 to a points for your solution submitted on the first stage and form 0 to b points for your self-criticism. In total, you get at most $a + b$ points for the problem.

9. Be sure to follow the academic integrity rules listed on the course webpage. The department and the university treat academic integrity very seriously and I am required to report all possible violations.

In all problems below you have to explicity do the following things

- explain why your algorithm is correct;
- analyse the complexity.

Problems

Problem 1: (3 + 1 points) Let a be the number from your NetID (for example, my NetID is gp72, so $a = 72$ for me). You are not allowed to pick any other number. Let $n = 2000 + a$ and $m = 10000 - a$. Consider a Karatsuba multiplication algorithm (p. 735 in the textbook) with base case $n = 1$ instead of $n = 64$ and all operations with digits performed base 10 instead of base 2. Draw the tree of all recursive calls made by such an algorithm for multiplication n by m.

Problem 2: (4 + 2 points) The complexity of some algorithm is defined by the following recurrence (n is a power of 2)

$$T(n) = \begin{cases} 3, \text{ if } n = 1, \\ 3T\left(\frac{n}{2}\right) + 5n^2, \text{ if } n > 1. \end{cases}$$

Find an expression for $T(n)$ in Θ-notation.

Problem 3: (4 + 2 points) Given is an array of $n-1$ numbers containing an increasing subsequence of the sequence $1, 2, \ldots, n$. Design an algorithm for finding the missing number with the complexity $O(\log_2 n)$. For example, if the input array is $[1, 2, 3, 5, 6]$ ($n = 6$), then the output should be 4.

Problem 4: (5 + 2 points) Design an algorithm that takes $n = 2^k - 1$ as an input and produces a schedule of a round-robin tournament for teams with numbers $1, \ldots, 2^k - 1$ in $2^k - 1$ days such that team with number i does not play a game on the day number i for every i. The complexity of the algorithm has to be $O(n^2)$.

Problem 5: (5 + 2 points) As an input you get positive integer C, the capacity of the knapsack, and an array of n pairs (s, v) of positive integers, where s and v the size and the value of the corresponding item. Design an algorithm that finds a set consisting of even number of items that fit into the knapsack such that the sum of the values is the largest possible. The complexity of the algorithms has to be $O(nC)$.

Reading: pages 734–738 in the textbook and the notes on the webpage.