CSE 2012- Design and Analysis of Algorithms
Practice Problem Sheet 1

Practice makes you Perfect

Inputs required for PPS1:

- Chapter 1,2,3 of CLRS book
- Design of the Algorithm for sorting (in an increasing order) by insertion and the analysis of the insertion-sort algorithm

Design of Algorithm

- Understand the problem
- Develop the logic to solve the problem
- Validate the logic(developed by you) with simple illustration.
- Write the pseudocode

Analysis of Algorithm

- Proof of Correctness (PoC) of the algorithm
- Computation of running time $T(n)$
- Best-case, average-case running and worst-case
- Time-complexity of the algorithm- Expression of asymptotic growth of $T(n)$ with $\theta, O, \Omega, o, \omega$ notations

Note: Every question should answer the 'design' and 'analysis' component of the algorithm. 'Design' component should involve all the steps described as above. 'Analysis' component should involve all the steps described as above.

1. Consider arranging n numbers stored in an array A, by first finding the smallest element of A and exchanging it with the element in $A[1]$. Then find the second smallest element of A and exchange it with $A[2]$ continue in this manner for the first $(n-1)$

elements. Based on the approach described above, write an algorithm for arranging the given n numbers in an increasing order of the numbers. Compute the best-case running time, worst-case running time and the average-case running time of the algorithm. Compare this algorithm with that of the insertion-sort algortim, based on the respective $T(n)$. Based on your analysis, conclude which algorithm performs better for which type of inputs etc.

2. Consider sorting (arranging in an increasing order) n numbers stored in an array A, by repeatedly by swapping the adjacent elements that are not in an increasing order. If $A[i]$ and $A[i+1]$ are such that $A[i] > A[i+1]$, then $A[i]$ and $A[i+1]$ shall be swapped. Based on the approach described above, write an algorithm for arranging the given n numbers in an increasing order of the numbers. Compute the best-case running time, worst-case running time and the average-case running time of the algorithm. Compare this algorithm with that of the insertion-sort algorithm and conclude which algorithm performs better for which type of inputs etc.
3. Consider an array of $A[1, 2, 3, ..., n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then we call the the pair (i, j) as an inversion of A. For example, the five inversions in the array $A :< 2, 3, 8, 6, 1 >$ are $(1, 5), (2, 5), (3, 4), (3, 5), (4, 5)$. Given an array of numbers, design the pseudocode for computing the number of inversions in the array ('inversion algorithm). Analyse the pseudocode of the inversion algorithm. Compare the running-time of insertion-sort and the running-time of inversion algorithm. Is there anything similar between the 'insertion-sort algorithm' and the 'inversion algorithm'.
4. A nonnegative integer n is given. Design an algorithm to count all the solutions of the inequality $x^2 + y^2 < n$, where x and y are non-negative integers. The pseudocode should not use any operations with real numbers (square roots, etc.). Analyse the pseudocode designed by you.
5. Given the coefficients $a_0, a_1, ..., a_n$, x and k, design two different pseudocode to evaluate the polynomial

$$P(x) = \sum_{k=0}^{n} a_k x^k$$

. Both the pseudocodes should follow different logic. You have to choose the logic in such a way that, the running time of one of the pseudocode should be of linear time- complexity Analyse both the pseudoocdes, with all the components.

6. Given the values of the english letters (lower case) $a, b, c, d, ..., x, y, z$ as $1, 2, 3, 4, 24, 25, 26$ respectively. Given a word w with finite number of english letters, we define the value of a word ($V(w)$) as the sum of the value of the english letters that occur in the word. If $w = good$, $V(w) = 7 + 15 + 15 + 4 = 41$. Given n words, design an algorithm to arrange the given words in a decreasing order of their respective values. Analyse your algorithm with all the required steps.
7. Given two numbers a and b, we usually say that a is less than or equal to b (denoted by $a \leq b$) if the value of a is smaller than or equal to b. Here, we define a new relation $a \leq b$ in a different way, as follows. a is said to be less than or equal to b (denoted by $a \leq b$) if the last digit of a is less than or equal to the last digit of b. For example, 27 is less than 19, since 7 is less than 9. 29 and 39 are said to be equal. 38 is greater than 102. Given n numbers, design a pseudocode to arrange the given numbers in a decreasing order, as per the order defined above. For example: Given the numbers 27,39,1, 55, 124. Numbers in the decreasing order are : 39, 27,55,124,1
   Also, analyze your algorithm with all the steps required.
8. Propose a problem and design an algorithm for the problem proposed by you. Compute the time-complexity of your algorithm. Now, modify the algorithm (by following a different logic for the same problem) so that the modified algorithm is more efficient than the original algorithm. List out the modifications (done in the original algorithm) that has improved the efficiency of the algorithm
9. In the insertion-sort algorithm A (discussed in the class), in lines 5-7, we apply a linear-search method to find the appropriate place for $A[i]$ to be inserted. Instead of a linear-search method, modify the algorithm A by using the 'binary-search method' (instead of the linear-search method) to find the appropriate place for $A[i]$ to be inserted. Let the modified algorithm be called as A'. Analyse both the algorithms A and A' and conclude which is

more efficient. In case, you are not aware of the 'binary-search method' please learn yourself from CLRS book.

10. Express the function

$$n^3/1000 - 100n^2 - 100n - 3$$

in terms of all the five notations: $\theta, O, \Omega, o, \omega$