LAB - 3

Fundamentals of Algorithmic Problem Solving-II:

(Analysis of time complexity of algorithms through the concept of asymptotic notations)

PROGRAM EXERCISE

Lab. Exercise (LE)

3. 1 Rewrite the program no-2.3 **(Insertion Sort)** with the following details.

i. Compare the best case, worst case and average case time complexity with the same data except time complexity will count the CPU clock time.

ii. Plot a graph showing the above comparison (n, the input data Vs. CPU times for best, worst & average case)

iii. Compare manually program no-2.1 graph vs program no-3.1 graph and draw your inference.

3. 2 Let A be a list of n (not necessarily distinct) integers. Write a program by using User Defined Function(UDF)s to test whether any item occurs more than □ n/2□ times in A.

a) UDF should take $O(n^2)$ time and use no additional space.

b) UDF should take $O(n)$ time and use $O(1)$ additional space.

3. 3 Write a program by using an user defined function for computing □ √n□ for any positive integer n. Besides assignment and comparison, your algorithm may only use the four basic arithmetical operations.

3. 4 Let A be an array of n integers $a_0,a_1,\dots,a_{n-1}$ (negative integers are allowed), denoted, by A[i... j], the sub-array $a_i, a_{i+1},\dots,a_j$ for $i \le j$. Also let $S_{i\text{-}j}$ denote the sum $a_i + a_{i+1} + \cdots + a_j$. Write a programby using an udf that must run in $O(n^2)$ time to find out the maximum value of $S_{i\text{-}j}$ for all the pair i, j with $0 \le i \le j \le n-1$. Call this maximum value S. Also obtains the maximum of these computed sums. Let $j < i$ in the notation A[i... j] is also allowed. In this case, A[i... j] denotes the empty sub-array (that is, a sub-array that ends before it starts) with sum $S_{i\text{-}j} = 0$. Indeed, if all the elements of A are negative, then one returns 0 as the maximum sub-array sum.

a. For example, for the array A[]={1, 3, 7, 2, 1, 5, 1, 2, 4, 6, 3}.

b. This maximum sum is $S = S_{3\text{-}8} = 2+(1)+5+(1)+(2)+4 = 7$.

Round Exercise (RE)

3. 5 Design a data structure to maintain a set S of n distinct integers that supports the following two operations:

   a) INSERT(x, S): insert integer x into S
   b) REMOVE-BOTTOM-HALF(S): remove the smallest $\lfloor n/2 \rfloor$ integers from S.
   c) Write a program by using UDFs that give the worse-case time complexity of the two operations INSERT(x, S) in O(1) time and REMOVE-BOTTOM-HALF(S) in O(n) time.

3. 6 Consider an n × n matrix A = (a_{ij}), each of whose elements a_{ij} is a nonnegative real number, and suppose that each row and column of A sums to an integer value. We wish to replace each element a_{ij} with either $\lfloor a_{ij} \rfloor$ or $\lceil a_{ij} \rceil$ without disturbing the row and column sums. Here is an example:

$$\begin{pmatrix} 10.9 & 2.5 & 1.3 & 9.3 \\ 3.8 & 9.2 & 2.2 & 11.8 \\ 7.9 & 5.2 & 7.3 & 0.6 \\ 3.4 & 13.1 & 1.2 & 6.3 \end{pmatrix} \to \begin{pmatrix} 11 & 3 & 1 & 9 \\ 4 & 9 & 2 & 12 \\ 7 & 5 & 8 & 1 \\ 4 & 13 & 2 & 6 \end{pmatrix}$$

Write a program by defining an user defined function that is used to produce the rounded matrix as described in the above example. Find out the time complexity of your algorithm/function.