Content of Lab. Manual Prepared by Prof. Anil Kumar Swain

CONTENTS

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 $a_0,a_1,\ldots,a_{n-1}$ (negative integers are allowed), denoted, by A[i... j], the sub-array $a_i, a_{i+1},\ldots,a_j$ for i≤j. Also let S_{i-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-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 $S_{i-j} = 0$. 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 $S = S_{3-8} = 2+(1)+5+(1)+(2)+4 = 7$.

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:

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

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.

LAB - 4
Divide and Conquer Method

(Binary Search, Merge Sort, Quick Sort, Randomized Quick Sort)

PROGRAM EXERCISE

Lab. Exercise (LE)

4.1 Write a program to search an element x in an array of n integers using **binary search** algorithm that uses divide and conquer technique. Find out the best case, worst case and average case time complexities for different values of n and plot a graph of the time taken versus n. The n integers can be generated randomly and x can be choosen randomly, or any element of the array or middle or last element of the array depending on type of time complexity analysis.

4.2 Write a program to **sort a list** of n elements using the **merge sort** method and determine the time required to sort the elements. Repeat the experiment for different values of n and different nature of data (random data, sorted data, reversely sorted data) in the list. n is the user input and n integers can be generated randomly. Finally plot a graph of the time taken versus n.

Home Exercise (HE)

4.3 Write a program to use divide and conquer method to determine the time required to find the maximum and minimum element in a list of n elements. The data for the list can be generated randomly. Compare this time with the time taken by straight forward algorithm or brute force algorithm for finding the maximum and minimum element for the same list of n elements. Show the comparison by plotting a required graph for this problem.

4.4 Write a program that uses a divide-and-conquer algorithm/user defined function for the exponentiation problem of computing a^n where $a > 0$ and n is a positive integer. How does this algorithm compare with the brute-force algorithm in terms of number of multiplications made by both algorithms.

Round Exercise (RE)

4.5 A and B are playing a guessing game where B first thinks up an integer X (positive, negative or zero, and could be of arbitrarily large magnitude) and A tries to guess it. In response to A's guess, B gives exactly one of the following three replies:

a) Try a bigger number
b) Try a smaller number or
c) You got it.

Write a program by designing an efficient algorithm to minimize the number of guesses A has to make. An example (not necessarily an efficient one) below:

Let B thinks up the number 35

A's guess	B's response
10	Try a bigger number
20	Try a bigger number
30	Try a bigger number
40	Try a smaller number
35	You got it

4.6 Write a program by using an user defined function to implement **merge sort** without a recursion (bottom-up version of mergesort) by starting with merging adjacent elements of a given array, then merging sorted pairs, and so on.

4.7 The **quick sort** algorithm is an efficient and popular sorting technique that sorts a list of keys recursively by choosing a pivot key. A pivot may be chosen as the first or last or mean or median or any random element of the list. Write a program to implement this sorting algorithm and execute the sorting programs for the following sets of data.

i. Ordered List
ii. Reverse order List
iii. A list containing the same value through out
iv. Random List
v. 50% of the List sorted

Also measure CPU time, number of partitions and number of comparisons for data sizes 1K, 50K, 1L, 1.5L, 2L, 2.5L, 3L, 3.5L, 4L, 4.5L and 1M. Present your results using tables and graphs and write a 1-page report that summarize the behavior of sorting algorithms tested and their suitability in each case as mentioned above.

4.8 In a social gathering, there are b boys and g girls (b > g) of different ages. You have two unsorted arrays giving their ages (one for the boys, the other for the girls). Write a program by devising an efficient O(b log g) algorithm to find out the ages that are common between both the boys and girls.

Example:

If Arrayboy = {10, 20, 11, 89, 23, 21} and Arraygirl = {12, 30, 11, 20},
Then Arraycommon = {11, 20}

4.9 Refer the following **new sort algorithm** for sorting an array A of n numbers. The algorithm is described below:

(i) If there is only one number, return.
(ii) If there are two numbers, perform a single comparison to determine the order.
(iii) If there are more than two numbers, then first sort the top two-thirds of the elements recursively. Follow this by sorting the bottom two-thirds of the elements recursively and then sorting the top two-thirds of the elements again.

a) Write a program that uses a recursive algorithm to implement the above strategy.
b) What is the comparison complexity of this new-sort algorithm? Formulate a recurrence relation and solve the same to justify your answer.