Programming and Data Structures Lab (CS2710)
Assignment-03: *Manipulating Arrays – Searching and Sorting*

Lab-work:

1. Implement Binary Serach in the following three ways, where the array is partitioned as –
(i) 1/2 and 1/2; (ii) 1/3 and 2/3; (iii) 1/4 and 2/4
Inputs:
(a) An array of n integers; and
(b) A serach key
Outputs:
(a) The position (array index) of the key in array, if the key is present; or
(b) Print -1 to indicate the key is absent in the array
Experiment the following things:
i. Print the number of comparisons to be made for each of these three cases in a given array of size n.
ii. Plot a graph for these three cases with increasing array size (n) in the x-axis and the number of comparisons in the y-axis.

2. Merge two Sorted Arrays without using Additional Space
Inputs: Two sorted integer arrays, A1[] (of m elements) and A2[] (of n elements)
Outputs: Merged two arrays such that the initial numbers (after complete sorting) are in the first array and the remaining numbers are in the second array [Extra space allowed in O(1)]
Example: Input = A1[] is {1, 5, 9, 10, 15, 20} and A2[] is {2, 3, 8, 13}
=> Output = A1[] is {1, 2, 3, 5, 8, 9} and A2[] is {10, 13, 15, 20}
Hint: This task is simple and O(m+n) if we are allowed to use extra space. But it becomes really complicated when extra space is not allowed and does not look possible in less than O(m.n) worst case time. The idea is to begin from last element of A2[] and search it in A1[]. If there is a greater element in A1[], then you move last element of A1[] to A2[]. To keep A1[] and A2[] sorted, you need to place last element of A2[] at correct place in A1[]. [You may use Insertion Sort type of insertion]

- -

Home-work:

1. Implement Counting Sort
Inputs: An array of n integers
Outputs:
(a) The sorted array of n integers; and
(b) The overall time required to sort (given by system in which the program is running)
Experiment the following thing:
Plot a graph showing the sorting time required in y-axis vs. the increasing size of array (n) in x-axis.

2. Implement the following four sorting algorithms:
(a) Bubble Sort, (b) Insertion Sort, (c) Merge Sort, (d) Quick Sort
Inputs: An array of n integers
Outputs:
(a) The sorted array of n integers; and
(b) The overall time required to sort (given by system in which the program is running)
Experiment the following things:
Plot four separate graphs (for each of the Sorting procedures) with increasing array size (n) in the x-axis and the time taken to sort in the y-axis.
In each of these graphs, plot for the following three cases, where the elements of the array are sorted in – (i) Ascending order; (ii) Descending order; and (iii) Arbitrary / Random order