Bharatiya Vidya Bhavan's

SARDAR PATEL INSTITUTE OF TECHNOLOGY

(Autonomous Institute Affiliated to University of Mumbai) Munshi Nagar, Andheri (W), Mumbai – 400 058.

Experiment No. 2

- **Aim** – Experiment on finding the running time of a divide and conquer algorithm.

Details – The understanding of running time of algorithms is explored by implementing two basic sorting algorithms namely Insertion and Selection sorts. These algorithms work as follows.

Merge sort– Merge sort is a sorting algorithm that works by dividing an array into smaller subarrays, sorting each subarray, and then merging the sorted subarrays back together to form the final sorted array.In simple terms, we can say that the process of merge sort is to divide the array into two halves, sort each half, and then merge the sorted halves back together. This process is repeated until the entire array is sorted.

Quicksort– Like Merge Sort, QuickSort is a Divide and Conquer algorithm. It picks an element as a pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.The key process in quickSort is a partition(). The target of partitions is, given an array and an element x of an array as the pivot, put x at its correct position in a sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.

Problem Definition & Assumptions – For this experiment, you need to implement two sorting algorithms namely merge sort and Quick sort methods. Compare these algorithms based on time and space complexity. Time required to sorting algorithms can be performed using high_resolution_clock::now() under namespace std::chrono.

You have togenerate1,00,000 integer numbers using C/C++ Rand function and save them in a text file. Both the sorting algorithms uses these 1,00,000 integer numbers as input as follows. Each sorting algorithm sorts a block of 100 integers numbers with array indexes numbers A[0..99], A[0..199], A[0..299],…, A[0..99999]. You need to use high_resolution_clock::now() function to find the time required for 100, 200, 300…. 100000 integer numbers. Finally, compare two algorithms namely Insertion and Selection by plotting the time required to sort 100000 integers using LibreOffice Calc/MS Excel. The x-axis of 2-D plot represents the block no. of 1000 blocks. The y-axis of 2-D plot representsthe tunning time to sort 1000 blocks of 100,200,300,...,100000 integer numbers.

Note – You have to use C/C++ file processing functions for reading and writing randomly generated 100000 integer numbers. Important Links:

1. C/C++ Rand function Online libraryhttps://cplusplus.com/reference/cstdlib/rand/
2. Time required calculation Online libraryhttps://en.cppreference.com/w/cpp/chrono/high_resolution_clock/now
3. Draw 2-D plot using OpenLibre/MS Excelhttps://support.microsoft.com/en-us/topic/present-your-data-in-a-scatter-chart-or-a-line-chart-4570a80f 599a-4d6b-a155-104a9018b86e

----

Input –

1) Each student have to generate random 100000 numbers using rand() function and use this input as 1000 blocks of 100 integer numbers to Insertion and Selection sorting algorithms.

Output –

1) Store the randomly generated 100000 integer numbers to a text file.
2) Draw two 2D plot of all functions such that the x-axis of 2-D plot represents the block no. of 1000 blocks. The y-axis of 2-D plot represents the running time to sort 1000 blocks of 100,200,300,...,100000 integer numbers. 3) Comment on Space complexity for two sorting algorithms.