St. Francis Institute of Technology, Mumbai-400 103
Department of Information Technology

A.Y. 2023-24
Class: SE-ITA/B, Semester: III
Subject: DATA STRUCTURE LAB

Experiment – 6 Selection sort and Merge sort algorithms

1. Aim: Write a menu driven program to implement Selection sort and Merge sort algorithm

2. Objectives: After study of this experiment, the student will be able to

• To use basic principles of programming as applied to sorting

• To learn fundamentals of sorting techniques

3. Outcomes: After study of this experiment, the student will be able to

• Implement sorting algorithms on given set of data and understand its operations

• Understand the concepts and apply the techniques of searching, hashing and sorting

4. Prerequisite: Sorting techniques.

5. Requirements: PC and Codeblock 20.03

6. Pre-Experiment Exercise:

Brief Theory:

A. Sorting

A sorting algorithm is defined as an algorithm that puts elements of a list in a certain order (that can either be numerical order, lexicographical order or any user-defined order). Efficient sorting algorithms are widely used to optimize the use of other algorithms like search and merge algorithms which require sorted lists to work correctly. There are two types of sorting:

• Internal sorting which deals with sorting the data stored in computer's memory • External sorting which deals with sorting the data stored in files. External sorting is applied when there is voluminous data that cannot be stored in computer's memory.

B. Sorting Techniques

1. Selection Sort

It selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list.

Algorithm:

1.Set the first element as' minimum'

2.Compare' minimum' with the second element. If the second element is smaller than minimum, assign the second element as 'minimum'. Compare 'minimum' with the third element. Again, if the third element is smaller, then assign' minimum' to the third element otherwise do nothing. The process goes on until the last element.

3.After each iteration, 'minimum' is placed in the front of the unsorted list.

For each iteration, indexing starts from the first unsorted element. Step 1 to 3 are repeated until all the elements are placed at their correct positions.s

2. Merge sort

It uses the divide, conquer, and combine algorithmic paradigm.

1.Divide means partitioning the n-element array to be sorted into two sub-arrays of n/2 elements. If A is an array containing zero or one element, then it is already sorted. However, if there are more elements in the array, divide A into two sub-arrays, A1 and A2 , each containing about half of the elements of A.

2.Conquer means sorting the two sub-arrays recursively using merge sort.

3.Combine means merging the two sorted sub-arrays of size n/2 to produce the sorted array of n elements.

Algorithm:

```
MERGE_SORT(ARR, BEG, END)
Step 1: IF BEG < END
              SET MID = (BEG + END)/2
              CALL MERGE_SORT (ARR, BEG, MID)
              CALL MERGE_SORT (ARR, MID + 1, END)
              MERGE (ARR, BEG, MID, END)
        [END OF IF]
Step 2: END
```

```
MERGE (ARR, BEG, MID, END)

Step 1: [INITIALIZE] SET I = BEG, J = MID + 1, INDEX = 0
Step 2: Repeat while (I <= MID) AND (J<=END)
            IF ARR[I] < ARR[J]
                  SET TEMP[INDEX] = ARR[I]
                  SET I = I + 1
            ELSE
                  SET TEMP[INDEX] = ARR[J]
                  SET J = J + 1
            [END OF IF]
            SET INDEX = INDEX + 1
      [END OF LOOP]
Step 3: [Copy the remaining elements of right sub-array, if any]
            IF I > MID
               Repeat while J <= END
                  SET TEMP[INDEX] = ARR[J]
                  SET INDEX = INDEX + 1, SET J = J + 1
               [END OF LOOP]
      [Copy the remaining elements of left sub-array, if any]
            ELSE
               Repeat while I <= MID
                  SET TEMP[INDEX] = ARR[I]
                  SET INDEX = INDEX + 1, SET I = I + 1
               [END OF LOOP]
            [END OF IF]
Step 4: [Copy the contents of TEMP back to ARR] SET K=0
Step 5: Repeat while K < INDEX
               SET ARR[K] = TEMP[K]
               SET K = K + 1
      [END OF LOOP]
Step 6: END
```

7. Laboratory Exercise

A. Procedure

Write a C program to implement Selection sort, Merge sort show all the following operations in switch case,

i) Enter values in array
ii) Sort using selection sort
iii) Sort using Merge sort
iv) Display array

B. Result/Observation/Program code:

Observe the output for the above code and print it.

8. Post-Experiments Exercise

A. Questions:

1. Sort the following numbers using Merge sort.
   12,56,97,2,4,86,15,94,23,48
2. Sort the following numbers using Quick sort.
   56,88,74,64,35,12,95,37,24
3. List and compare the running time various sorting techniques.
4. Write a program for insertion sort and show output after every pass

B. Conclusion:

1. Summary of Experiment
2. Importance of Experiment

C. References:

1. S. K Srivastava, Deepali Srivastava; Data Structures through C in Depth; BPB Publications; 2011.
2. Reema Thareja; Data Structures using C; Oxford.
3. Data Structures A Pseudocode Approach with C, Richard F. Gilberg & Behrouz A. Forouzan, second edition, CENGAGE Learning.
