Symbiosis Institute of Technology

Faculty of Engineering

CSE- Academic Year 2024-25

Data Structures – Lab   Batch 2023-27

Lab Assignment No:- 3	
Name of Student	Pankhuri Varshney
PRN No.	23070122160
Batch	2023-27
Class	CS-B2
Academic Year & Semester	2024-25 Semester 3
Date of Performance	9th August, 2024
Title of Assignment:	Implement following sorting techniques find the time complexity: Merge
Theory Questions:	1. Apply merge Sort on 9 input items and show the partial pass-wise sorting done. Analyze its Time Complexity (Best, Worst, and Average Case) & Space Complexity 2. Discuss time complexity of merge sort and quick sort in detail.

Source Code/Algorithm/Flow Chart:

Implement following sorting techniques find the time complexity: Merge

Source Code:

```
#include <stdio.h>
void mergeSort(int arr[], int leftIndex, int rightIndex);
void merge(int arr[], int leftIndex, int mid, int rightIndex);

void mergeSort(int arr[], int leftIndex, int rightIndex){
    if(leftIndex>=rightIndex)
        return;
    int mid=(leftIndex+rightIndex)/2;
    mergeSort(arr, leftIndex, mid);
    mergeSort(arr, mid+1, rightIndex);

    merge(arr, leftIndex, mid, rightIndex);
}
void merge(int arr[], int leftIndex, int mid, int rightIndex){
    int left=mid-leftIndex+1;
    int right=rightIndex-mid;
    int L[left], R[right];
    for(int i=0; i<left; i++){
        L[i]=arr[leftIndex+i];
    }
    for(int j=0; j<right; j++){
        R[j]=arr[mid+1+j];
    }
    int i,j,k;
    for(i=0, j=0, k=leftIndex; i<left&&j<right; k++){
        if(L[i]<R[j]){
            arr[k]=L[i];
            i++;
        }
        else{
            arr[k]=R[j];
            j++;
        }
    }
    while (i<left) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j<right) {
        arr[k] = R[j];
        j++;
        k++;
    }
}


int main(){
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the elements: \n");
    for(int i=0; i<n; i++)
        scanf("%d", &arr[i]);
    mergeSort(arr, 0, n-1);
    printf("SORTED ARRAY: \n");
    for(int i=0; i<n; i++)
        printf("%d\t", arr[i]);
    printf("\n");
}
```

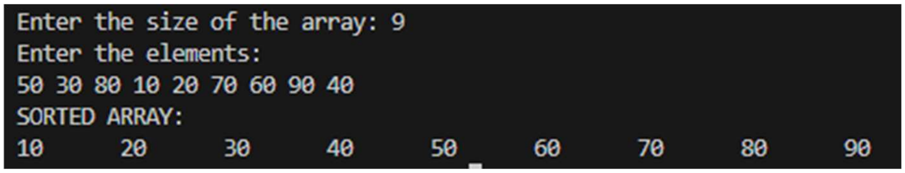
Practice Problem: Implement Quick Sort

Source Code:

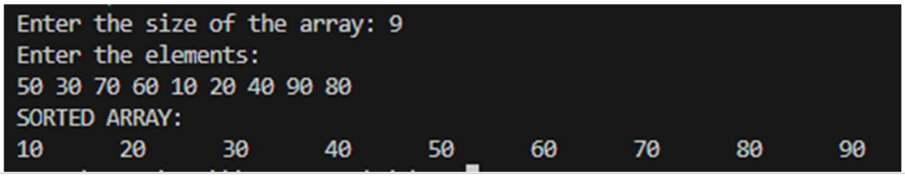
```
#include <stdio.h>
void quickSort(int arr[], int low, int high);
int partition(int arr[], int low, int high);

void quickSort(int arr[], int low, int high){
    if(low>=high)
        return;
    int pivot=partition(arr, low, high);
    quickSort(arr, low, pivot-1);
    quickSort(arr, pivot+1, high);
}
int partition(int arr[], int low, int high){
    int pivot=arr[high];
    int i=low-1;
    for(int j=low; j<high; j++){
        if(arr[j]<pivot){
            i++;
            int temp=arr[i];
            arr[i]=arr[j];
            arr[j]=temp;
        }
    }
    int temp=arr[high];
    arr[high]=arr[i+1];
    arr[i+1]=temp;
    return i+1;
}
int main(){
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the elements: \n");
    for(int i=0; i<n; i++)
        scanf("%d", &arr[i]);
    quickSort(arr, 0, n-1);
    printf("SORTED ARRAY: \n");
    for(int i=0; i<n; i++)
        printf("%d\t", arr[i]);
    printf("\n");
}
```

Output Screenshots

MERGE SORT:



```
Enter the size of the array: 9
Enter the elements:
50 30 80 10 20 70 60 90 40
SORTED ARRAY:
10      20      30      40      50      60      70      80      90
```

	QUICK SORT:  Enter the size of the array: 9 Enter the elements: 50 30 70 60 10 20 40 90 80 SORTED ARRAY: 10 20 30 40 50 60 70 80 90
Practice questions	1. Implement Quick sort 2. o/p screenshot
Conclusion	Thus, we have studied different sorting algorithms and their time complexities.