Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Summer, Year:2022), B.Sc. in CSE (Day)

LAB REPORT NO # 03

Course Title: Data Structure Lab

Course Code: CSE 106 Section: PC-213DA

Student Details

Name	ID
Pankaj Mahanto	213902002

Lab Date	**: 22/06/2022**
Submission Date	**: 28/06/2022**
Course Teacher's Name	**: Farhana Akter Sunny**

Farhana Akter Sunny

Senior Professor

Green University of Bangladesh

[For Teachers use only: Don't Write anything inside the box]

Lab Report Status

Marks: . . . . Signature: . . . .

Comments: . . . . Date: . . . .

1. TITLE OF THE LAB EXPERIMENT [1]

- Implement a program of Quick Sort with Recursion.
- Implement a program of merge sort with recursion

2. OBJECTION [1]

In this problem I will discuss recursive function and how it use?

3. PROCEDURE /ANALYSIS/DESIGN/PSEUDOCODE [2]

Quick Sort Algorithm

```
partition(A[], low, high)

1.[Initialize] Set i=low+1, j=high and
PIVOT=low.

2.[Scan from i to j]

(a) Repeat while A[i] <= A[PIVOT] and
i ≠ PIVOT

i= i + 1

[end of loop]

(b) if PIVOT=i, then return.

(c) if A[i]>A PIVOT]> , then

i.[Interchange A[i] AND A[PIVOT]

TEMP= A[PIVOT] , A[PIVOT] = A[i] ,
A[i] =TEMP

ii. Set PIVOT=i

iii. Go to Step 3.

[end of if structure]

3.[Scan from j to i]

(a) Repeat while A[PIVOT] <= A[j] and
PIVOT ≠ j

j = j - 1

[end of loop]

(b) if PIVOT=j, then return.

(c) if A[PIVOT]>A[j], then

i.[Interchange A[PIVOT] AND A[j]]

TEMP= A[PIVOT] , A[PIVOT] = A[j] ,
A[j] =TEMP

ii. Set PIVOT=j
```

iii. Go to Step 2.

[end of if structure]

Merge sort algorithm

```
MERGE_SORT(a, low, high)

    if low < high

        1. set mid = (low + high)/2
        2. MERGE_SORT(a, low, mid)
        3. MERGE_SORT(a, mid + 1, high)
        4. MERGE (a, low, mid, high)

    end of if

    END MERGE_SORT
```

4. IMPLEMENTATION

Quick_Sort:

```
#include <iostream>
using namespace std;

void inputArr(int a[], int n)
{
    cout << "\n store data in array\n"
         << endl;
    for (int i = 0; i < n; i++)
    {
        cout << i + 1 << " element:";
        cin >> a[i];
    }
    cout << endl;
}
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
void PrintArr(int a[], int n)
{
    int i;
    for (int i = 0; i < n; i++)
    {
        cout << a[i] << "  ";
    }
}
int partition(int a[], int low, int high)
{
    int i, j, pivot;
    int n = 6;
    while (low < high)
    {
        i = low + 1;
        j = high;
        pivot = low;
        while (a[i] <= a[pivot] && (pivot != i))
        {
            i++;
        }
        while (a[j] >= a[pivot] && (pivot != j))
        {
            j--;
        }
        if (i < j)
        {
            swap(&a[i], &a[j]);
            // cout<<"sorted "<<i-1<<" pass =";
            PrintArr(a, 6);
            cout << "\n";
            break;
        }
        swap(&a[pivot], &a[j]);
        // cout<<"sorted "<<pivot+4<<" pass =";
        PrintArr(a, 6);
        cout << "\n";
        break;
    }
    return j;
}
void QuickSort(int a[], int low, int high)
{
    if (low < high)
    {
        int index = partition(a, low, high);
        QuickSort(a, low, index - 1);
        QuickSort(a, index + 1, high);
    }
}
int main()
{
    int n = 6;
    int a[n];
    inputArr(a, n);
    cout<<"\nGiven array: ";
    PrintArr(a, n);
    cout << "\n";
    QuickSort(a, 0, 5);
    cout << "\n";
    cout << "\nfinal sorted output here!!\n"
         << endl;
    PrintArr(a, n);
    return 0;
}
```

Merge Sort:

```
#include <iostream>
using namespace std;
void inputArr(int a[], int n)
{
    cout << "\n store data in array\n"
         << endl;
    for (int i = 0; i < n; i++)
    {
        cout << i + 1 << " element:";
        cin >> a[i];
    }
    cout << endl;
}
// void swap(int *a, int *b)
// {
//     int temp = *a;
//     *a = *b;
//     *b = temp;
// }
void PrintArr(int a[], int n)
{
    int i;

    for (int i = 0; i < n; i++)
    {
        cout << a[i] << "  ";
    }
}
```

```
void Merge(int a[], int low, int mid, int high)
{
    int i, j, k, n1, n2;
    n1 = mid - low + 1;
    n2 = high - mid;
    int l[n1], r[n2];
    for (i = 0; i < n1; i++)
    {
        l[i] = a[low + i];
    }
    for (j = 0; j < n2; j++)
    {
        r[j] = a[j + mid + 1];
    }
    i = 0;
    j = 0;
    k = low;
    while (i < n1 && j < n2)
    {
        if (l[i] < r[j])
        {
            a[k] = l[i];
            i++;
        }
        else
        {
            a[k] = r[j];
            j++;
        }
        k++;

    }
    while (i < n1)
    {
        a[k] = l[i];
        i++;
        k++;
        PrintArr(a, 6);
        cout << "\n";
    }
    while (j < n2)
    {
        a[k] = r[j];
        j++;
        k++;
        PrintArr(a, 6);
        cout << "\n";
    }
}

void MergeSort(int a[], int low, int high)
{
    if (low < high)
    {
        int mid = (low + high) / 2;
        MergeSort(a, low, mid);
        MergeSort(a, mid + 1, high);
        Merge(a, low, mid, high);
    }
}
int main()
{
    int n = 6;
    int a[n];
    inputArr(a, n);
    cout << "\n Given array here:" << endl;
    PrintArr(a, n);
    cout << "\n";
    MergeSort(a, 0, 5);
    cout << "\n";
    cout << "\nfinal sorted output here!!\n"
         << endl;
    PrintArr(a, n);
    return 0;
}
```

5. TEST RESULT

Output Quick sort:

```
store data in array

1 element:7
2 element:4
3 element:10
4 element:8
5 element:3
6 element:5


Given array: 7  4  10  8  3  5
7  4  5  8  3  10
7  4  5  3  8  10
3  4  5  7  8  10
3  4  5  7  8  10
3  4  5  7  8  10


final sorted output here!!

3  4  5  7  8  10
```

Output Merge sort:

```
store data in array

1 element:31
2 element:12
3 element:8
4 element:25
5 element:20
6 element:17


 Given array here:
31  12  8  25  20  17
12  31  8  25  20  17
8  12  8  25  20  17
8  12  31  25  20  17
8  12  31  20  25  17
8  12  31  17  20  17
8  12  31  17  20  25
8  12  17  20  25  31


final sorted output here!!

8  12  17  20  25  31
```

6. ANALYSIS AND DISCUSSION

In first problem we get the proper use of recursion and how to use it.In these problem first of all use merge sort and low and high element then use merge then call merge sort and finally solved this problem. In the second question we will try merge sort. First of all face some problem in the particular question but finally solved it.