DAA Lab 2: Merge and Quick sort

### Meet Jain, 23BCP093

**Theory**

**Merge sort**is a sorting algorithm that follows the [**divide-and-conquer**](https://www.geeksforgeeks.org/introduction-to-divide-and-conquer-algorithm-data-structure-and-algorithm-tutorials/)approach. It works by recursively dividing the input array into smaller subarrays and sorting those subarrays then merging them back together to obtain the 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.

**Code**

#include <bits/stdc++.h>

using namespace std;

void merge(int a[],int l,int mid,int h){

    int temp[h-l+1];

    int left = l;

    int right = mid+1;

    int i = 0;

    while(left<=mid && right <=h){

        if(a[left] < a[right]){

            temp[i] = a[left];

            i++;

            left++;

        }

        else {

            temp[i] = a[right];

            i++;

            right++;

        }

    }

    while(left <= mid){

        temp[i] = a[left];

        i++;

        left++;

    }

    while(right<=h){

        temp[i] = a[right];

        i++;

        right++;

    }

    for(int j=l;j<=h;j++){

        a[j] = temp[j-l];

    }

}

void mergesort(int a[], int l,int h){

    if(l == h) return;

    int mid = (l+h)/2;

    mergesort(a,l,mid);

    mergesort(a,mid+1,h);

    merge(a,l,mid,h);

}

int main(){

    int array[] = {5,4,3,2,1};

    int n = sizeof(array)/sizeof(array[0]);

    mergesort(array,0,n-1);

    for(int i=0;i<n;i++){

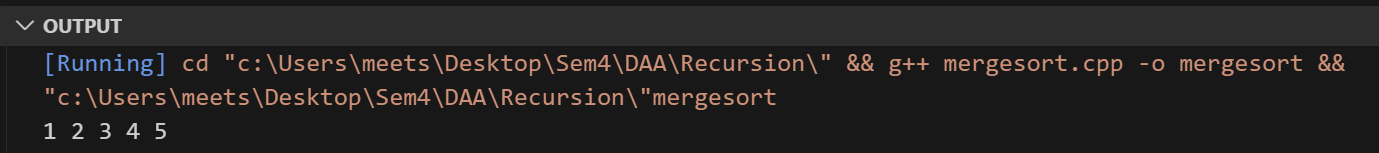
        cout << array[i] << " ";

    }

    return 0;

}

**Output**

****

**Theory**

**Quick Sort** is a divide-and-conquer algorithm like [the Merge Sort](https://takeuforward.org/data-structure/merge-sort-algorithm/). But unlike Merge sort, this algorithm does not use any extra array for sorting(though it uses an auxiliary stack space). So, from that perspective, Quick sort is slightly better than Merge sort.

This algorithm is basically a repetition of two simple steps that are the following:

* Pick a pivot and place it in its correct place in the sorted array.
* Shift smaller elements(i.e. Smaller than the pivot) on the left of the pivot and larger ones to the right.

**Code**

#include <bits/stdc++.h>

using namespace std;

int partition(int a[],int low,int high){

    int pivot = a[low];

    int i = low;

    int j = high;

    while(i < j){

        while (a[i] <= pivot && i<=high-1)

        {

            i++;

        }

        while(a[j] > pivot && j>low){

            j--;

        }

        if(i<j) swap(a[i],a[j]);

    }

    swap(a[low],a[j]);

    return j;

}

void quicksort(int a[],int low, int high){

    if(low<high){

        int p = partition(a,low,high);

        quicksort(a,low,p-1);

        quicksort(a,p+1,high);

    }

}

int main(){

    int array[] = {15,41,13,12,1};

    int n = sizeof(array)/sizeof(array[0]);

    quicksort(array,0,n-1);

    for(int i=0;i<n;i++){

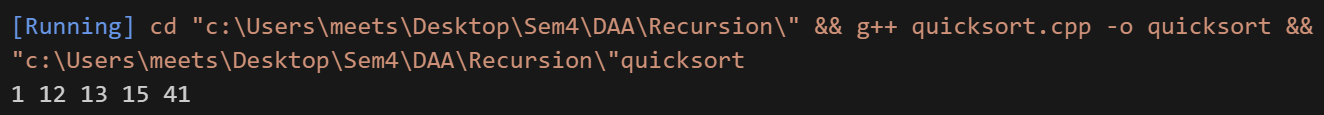
        cout << array[i] << " ";

    }

    return 0;

}

**Output**

****