* **MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY**

**Date:** 17/02/2021

ADA: LAB-ASSIGNMENT 4 (17-02-2021)

1) Finding the maximum and minimum value in an array using the divide and conquer approach.

#include <bits/stdc++.h>

using **namespace** std;

**struct** Pair

{

**int** min;

**int** max;

};

**struct** Pair getMinMax(**int** arr[], **int** low, **int** high)

{

**struct** Pair minmax, mml, mmr;

**int** mid;

    if (low == high)

    {

        minmax.max = arr[low];

        minmax.min = arr[low];

        return minmax;

    }

    if (high == low + 1)

    {

        if (arr[low] > arr[high])

        {

            minmax.max = arr[low];

            minmax.min = arr[high];

        }

        else

        {

            minmax.max = arr[high];

            minmax.min = arr[low];

        }

        return minmax;

    }

    mid = (low + high) / 2;

    mml = getMinMax(arr, low, mid);

    mmr = getMinMax(arr, mid + 1, high);

    if (mml.min < mmr.min)

        minmax.min = mml.min;

    else

        minmax.min = mmr.min;

    if (mml.max > mmr.max)

        minmax.max = mml.max;

    else

        minmax.max = mmr.max;

    return minmax;

}

**int** main()

{

**int** n;

    cout << "Enter number of elements : ";

    cin >> n;

**int** arr[n];

    cout << "Enter the " << n << " elements:\n";

    for (**int** i = 0; i < n; i++)

    {

        cin >> arr[i];

    }

**struct** Pair minmax = getMinMax(arr, 0, n - 1);

    cout << "Minimum element is "

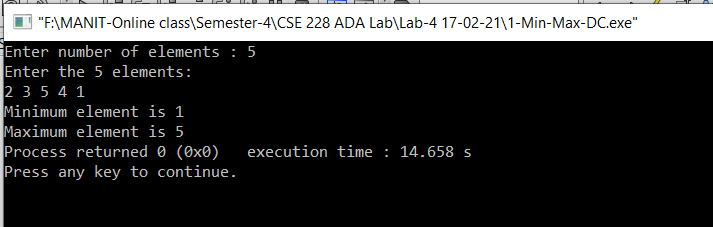
         << minmax.min << endl;

    cout << "Maximum element is "

         << minmax.max;

    return 0;

}



2) Computes the sum of an array of integers using the divide and conquer approach.

#include <bits/stdc++.h>

using **namespace** std;

**int** sum(**int** arr[], **int** l, **int** r)

{

*//Base case*

    if (l == r)

        return arr[l];

*//Divide & Conquer*

**int** mid = (l + r) / 2;

**int** lsum = sum(arr, l, mid);

**int** rsum = sum(arr, mid+1, r);

    return lsum + rsum;

}

**int** main()

{

**int** n;

    cout << "Enter number of elements : ";

    cin >> n;

**int** arr[n];

    cout << "Enter the " << n << " elements:\n";

    for (**int** i = 0; i < n; i++)

    {

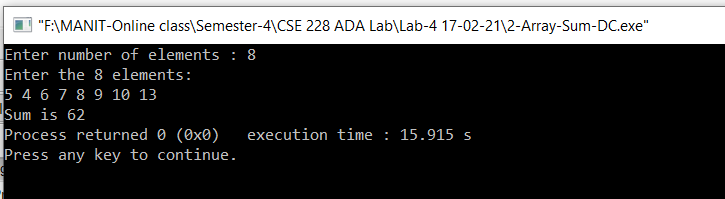
        cin >> arr[i];

    }

    cout << "Sum is " << sum(arr, 0, n - 1);

    return 0;

}



3) What is a recursive Quick Sort? Write an algorithm and analyze the Time complexity of the Algorithm.

#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;

    do

    {

        do

        {

            i++;

        } while (A[i] <= pivot);

        do

        {

            j--;

        } while (A[j] > pivot);

        if (i < j)

        {

            swap(A[i], A[j]);

        }

    } while (i < j);

    swap(A[low], A[j]);

    return j;

}

**void** quickSort(**int** A[], **int** low, **int** high)

{

    if (low < high)

    {

**int** q = partition(A, low, high);

        quickSort(A, low, q);

        quickSort(A, q + 1, high);

    }

}

**int** main()

{

**int** num;

    cout << "Enter number of elements to be sorted:";

    cin >> num;

**int** A[num];

    cout << "Enter the " << num << " elements:\n";

    for (**int** i = 0; i < num; i++)

    {

        cin >> A[i];

    }

    quickSort(A, 0, num);

    cout << endl;

    cout << "Sorted array\n";

    for (**int** i = 0; i < num; i++)

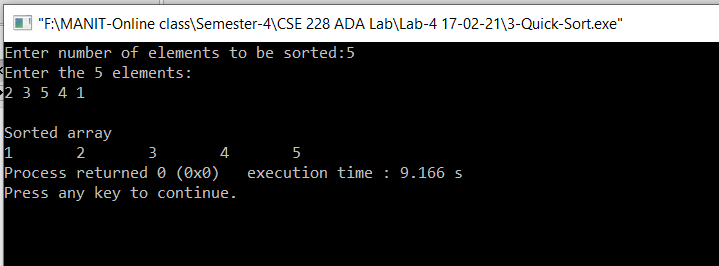
    {

        cout << A[i] << "\t";

    }

    return 0;

}



Time Complexity:

**Average case** O(nlogn) When the array is balanced partitioned.

**Worst case** O(n^2), worst-case behaviour for quicksort occurs when the partition method produces one subproblem with n-1 elements and one with 0 elements.