

**Assignment-3**

**Name: - Mann Kothari**

**Reg. No.: - 22BCB7064**

**Course Title: - Design and Analysis of Algorithm (Embedded Lab)**

**Course Code: - CSE3023**

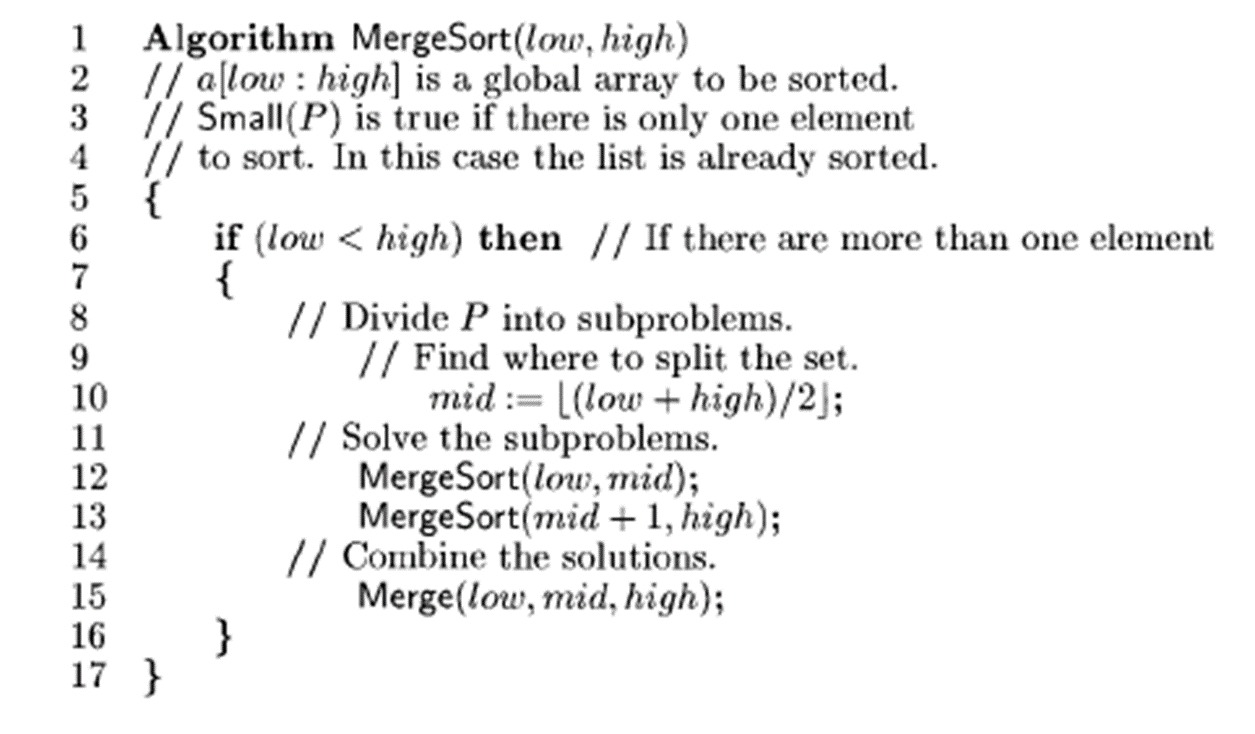
**Slot: - L21+L22**

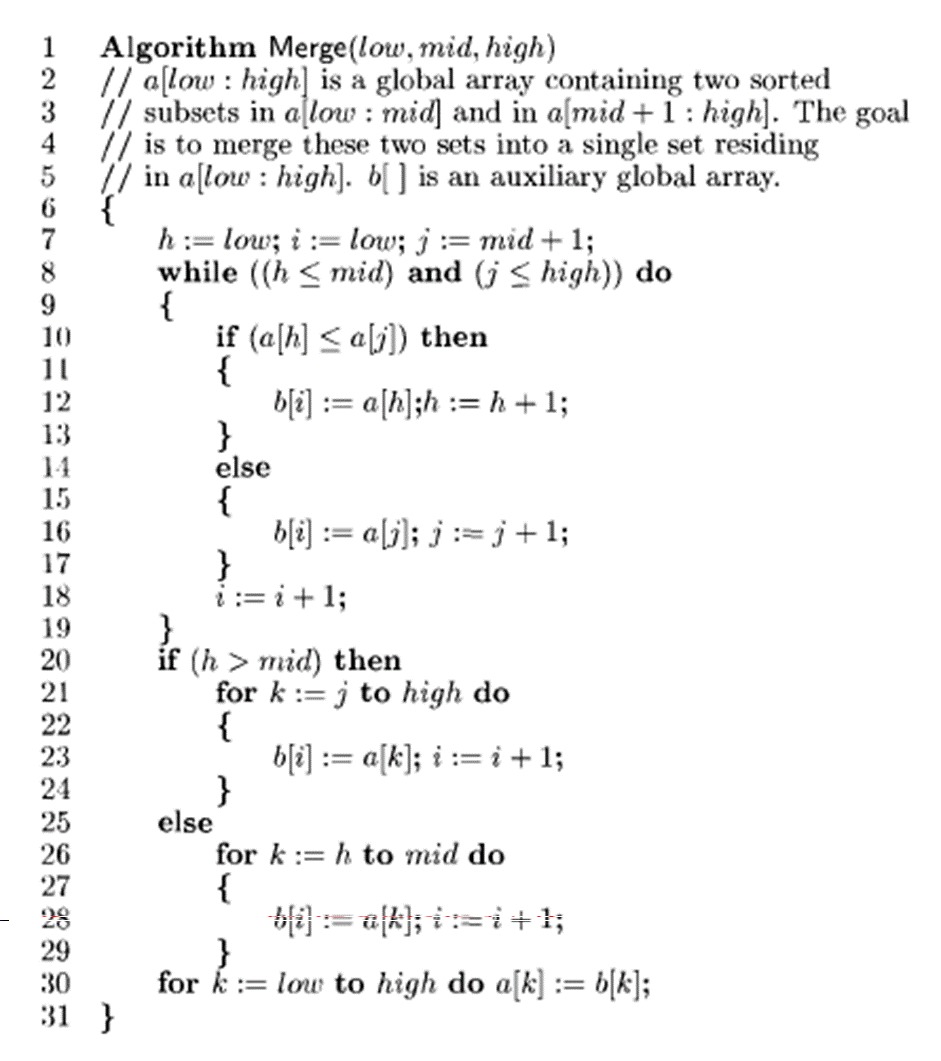
**Submitted to: - Prof. Tanikella Divya Naga Pavani**

* **INDEX**
* Merge Sort.
* Algorithm
* Code
* Output
* Complexity
* Quick Sort.
* Algorithm
* Code
* Output
* Complexity
* Complexity Chart

1. Write a C/C++ program to Implement Merge Sort

Algorithm: -





Time Complexity: -

O(nlogn)

Code: -

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 100

int a[MAX\_SIZE], b[MAX\_SIZE];

void Merge(int low, int mid, int high)

{

    int h = low, i = low, j = mid + 1;

    while ((h <= mid) && (j <= high))

    {

        if (a[h] <= a[j])

        {

            b[i] = a[h];

            h += 1;

        }

        else

        {

            b[i] = b[j];

            j += 1;

        }

        i += 1;

    }

    if (h > mid)

    {

        for (int k = j; k <= high; k++)

        {

            b[i] = a[k];

            i += 1;

        }

    }

    else

    {

        for (int k = h; k <= mid; k++)

        {

            b[i] = a[k];

            i += 1;

        }

    }

    for (int k = low; k <= high; k++)

    {

        a[k] = b[k];

    }

}

void MergeSort(int low, int high)

{

    int mid = 0;

    if (low < high)

    {

        mid = ((low + high) / 2);

        MergeSort(low, mid);

        MergeSort(mid + 1, high);

        Merge(low, mid, high);

    }

}

int main()

{

    int n;

    printf("Enter array size\n");

    scanf("%d", &n);

    int arr[n];

    printf("Enter the array elemnts");

    for (int i = 0; i < n; i++)

    {

        scanf("%d", &arr[i]);

    }

    MergeSort(0, n - 1);

    printf("The sorted array is: \n");

    for (int i = 0; i < n; i++)

    {

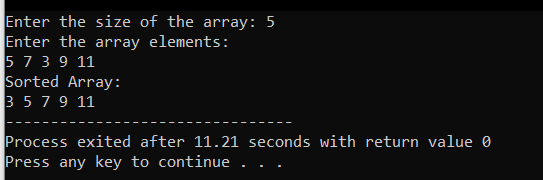
        printf("%d\t", arr[i]);

    }

    return 0;

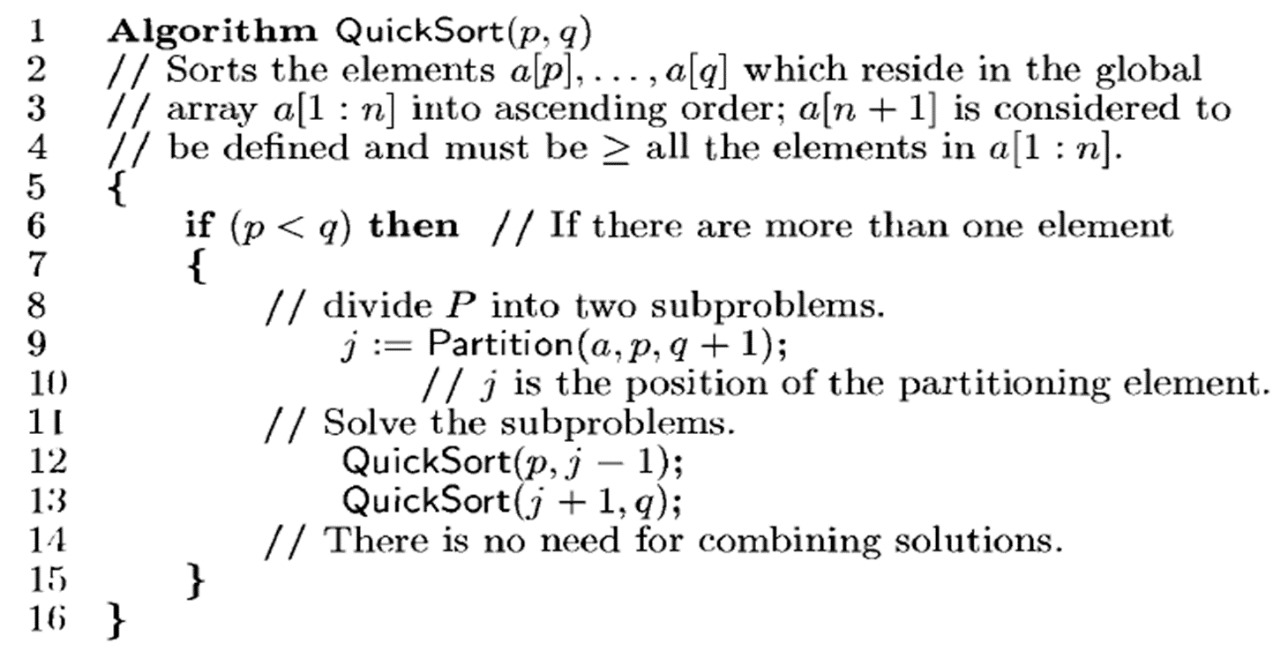
}

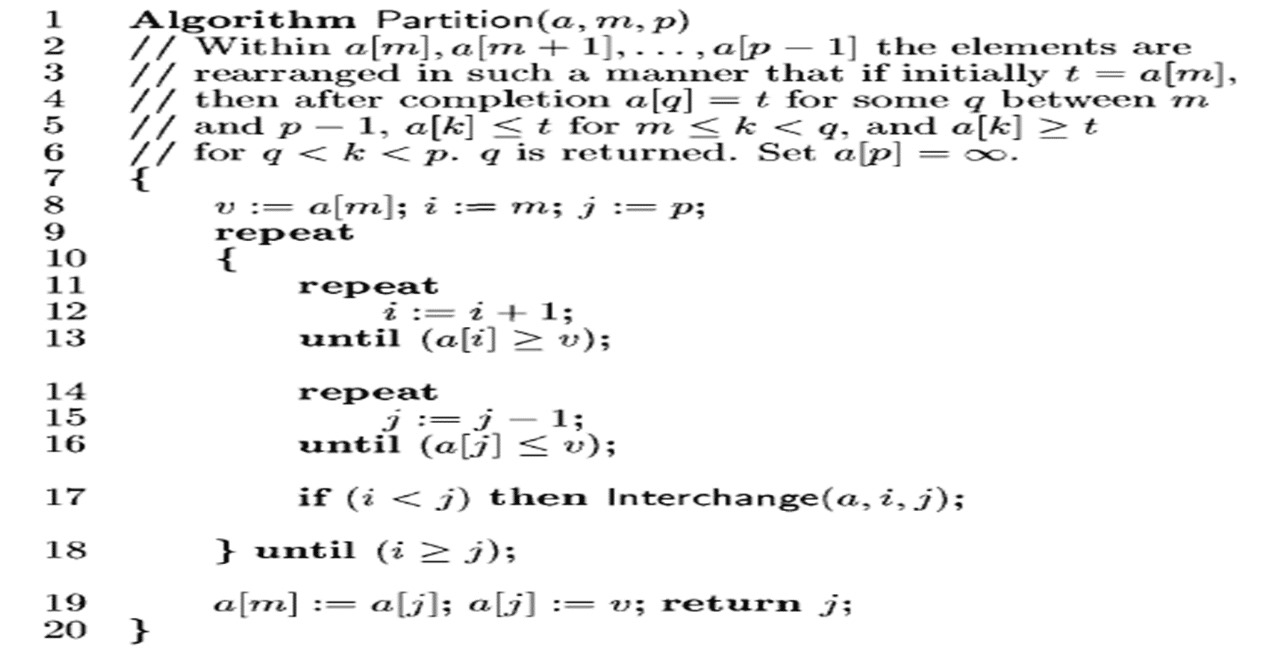
* **Output:**

****

1. Write a C/C++ program to Implement Quick Sort

Algorithm: -





Time Complexity: -

O(nlogn)

Code: -

#include <stdio.h>

void swap(int \*a, int \*b) {

    int temp = \*a;

    \*a = \*b;

    \*b = temp;

}

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++;

            swap(&arr[i], &arr[j]);

        }

    }

    swap(&arr[i + 1], &arr[high]);

    return (i + 1);

}

void quickSort(int arr[], int low, int high) {

    if (low < high) {

        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);

        quickSort(arr, pi + 1, high);

    }

}

int main() {

    int n;

    printf("Enter the size of the array: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter the array 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++) {

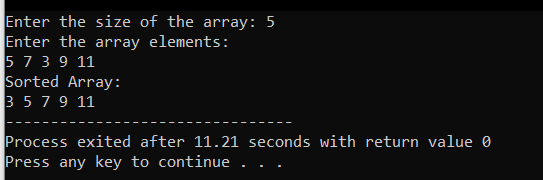
        printf("%d ", arr[i]);

    }

    return 0;

}

* **Output:**



|  |  |  |
| --- | --- | --- |
| Algorithm | Time Complexity | Space Complexity |
| Merge Sort | O(nlogn) | O(n) |
| Binary search | O(nlogn) | O(n) |