 <b>Marwadi University</b>	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach	
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

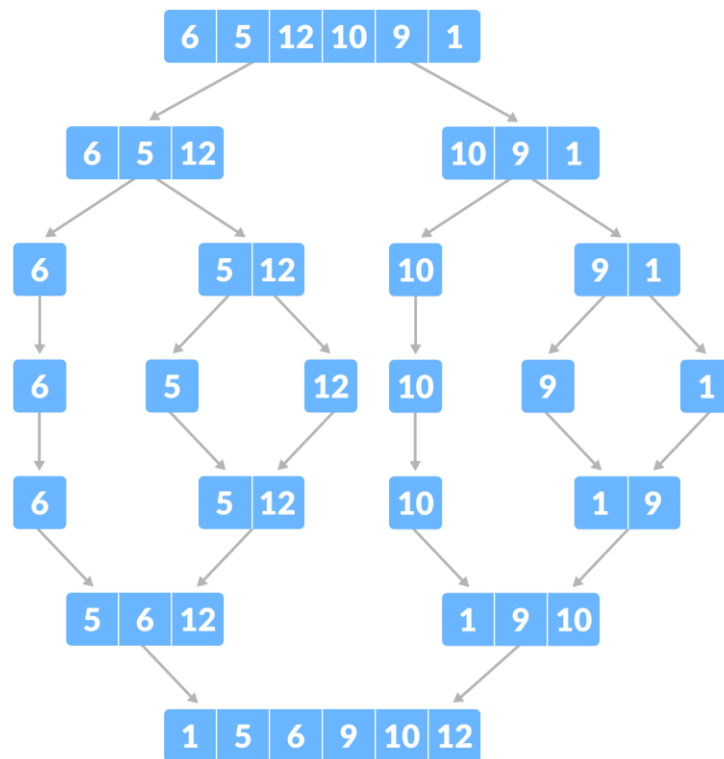
**Aim:** Implementing the Sorting Algorithms Using Divide and Conquer Approach

**IDE:** Visual Studio Code

## **Merge Sort**


### **Theory: -**

- Merge Sort is one of the most popular sorting algorithms that is based on the principle of Divide and Conquer Algorithm.
- Here, a problem is divided into multiple sub-problems. Each sub-problem is solved individually. Finally, sub-problems are combined to form the final solution.



### **Divide and Conquer Strategy**

- Using the Divide and Conquer technique, we divide a problem into subproblems. When the solution to each subproblem is ready, we 'combine' the results from the subproblems to solve the main problem.
- Suppose we had to sort an array A. A subproblem would be to sort a sub-section of this array starting at index p and ending at index r, denoted as A[p..r].

 <b>Marwadi University</b>	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach	
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

### Divide

- If  $q$  is the half-way point between  $p$  and  $r$ , then we can split the subarray  $A[p..r]$  into two arrays  $A[p..q]$  and  $A[q+1, r]$ .

### Conquer

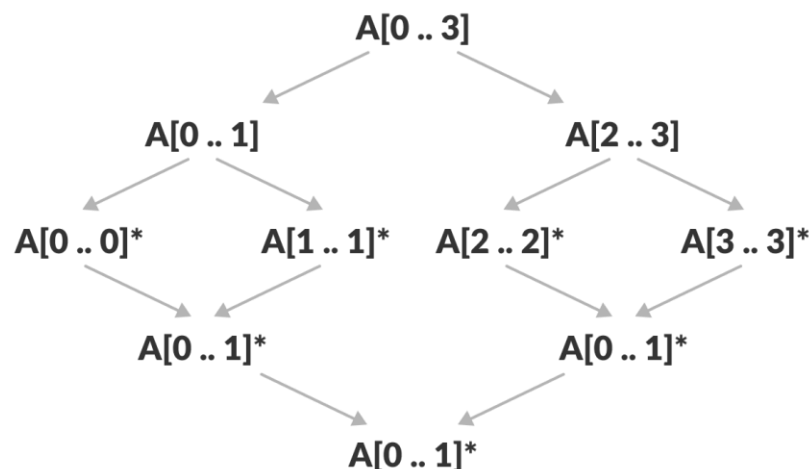
- In the conquer step, we try to sort both the subarrays  $A[p..q]$  and  $A[q+1, r]$ . If we haven't yet reached the base case, we again divide both these subarrays and try to sort them.

### Combine

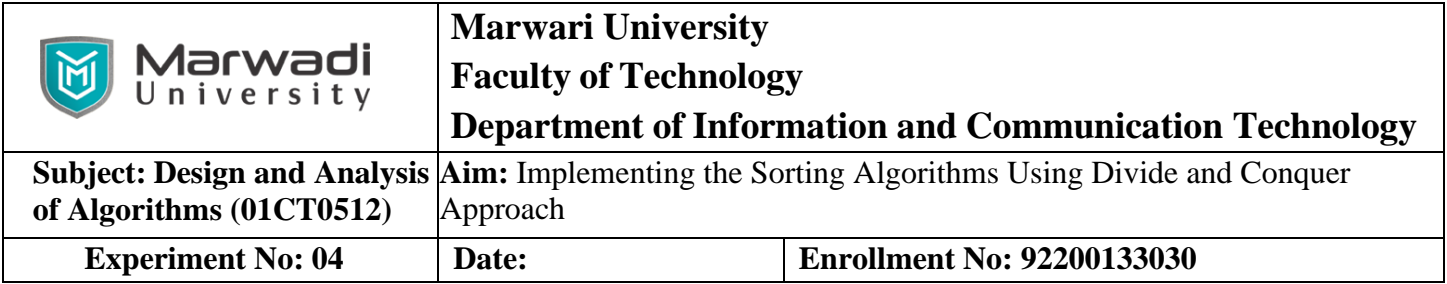
- When the conquer step reaches the base step and we get two sorted subarrays  $A[p..q]$  and  $A[q+1, r]$  for array  $A[p..r]$ , we combine the results by creating a sorted array  $A[p..r]$  from two sorted subarrays  $A[p..q]$  and  $A[q+1, r]$ .


### Working of Merge Sort


- The MergeSort function repeatedly divides the array into two halves until we reach a stage where we try to perform MergeSort on a subarray of size 1 i.e.  $p == r$ .
- After that, the merge function comes into play and combines the sorted arrays into larger arrays until the whole array is merged.
- To sort an entire array, we need to call  $\text{MergeSort}(A, 0, \text{length}(A)-1)$ .
- As shown in the image below, the merge sort algorithm recursively divides the array into halves until we reach the base case of array with 1 element. After that, the merge function picks up the sorted sub-arrays and merges them to gradually sort the entire array.











 <b>Marwadi</b> University	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
	<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

 <b>Marwadi</b> University	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
	<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

 <b>Marwadi</b> University	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
	<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

 <b>Marwadi</b> University	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
	<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

 <b>Marwadi</b> University	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
	<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>


 <b>Marwadi</b> University	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
	<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

**Code :-**

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

void Print_Array(vector<int> Array) {
    for (int i = 0; i < Array.size(); i++) {
        cout << Array[i] << " ";
    }
    cout << endl;
}
```

 <b>Marwadi</b> University	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach	
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

```

void Merge(vector<int>& Array, int low, int mid, int high) {
    int lower_bound = mid - low + 1;
    int upper_bound = high - mid;
    vector<int> Left_Array(lower_bound);
    vector<int> Right_Array(upper_bound);
    for (int i = 0; i < lower_bound; i++) {
        Left_Array[i] = Array[low + i];
    }
    for (int i = 0; i < upper_bound; i++) {
        Right_Array[i] = Array[mid + 1 + i];
    }
    int i = 0;
    int j = 0;
    int k = low;
    while (i < lower_bound && j < upper_bound) {
        if (Left_Array[i] <= Right_Array[j]) {
            Array[k] = Left_Array[i];
            i++;
        }
        else {
            Array[k] = Right_Array[j];
            j++;
        }
        k++;
    }
    while (i < lower_bound) {
        Array[k] = Left_Array[i];
        i++;
        k++;
    }
    while (j < upper_bound) {
        Array[k] = Right_Array[j];
        j++;
        k++;
    }
}


```

```

void Merge_Sort(vector<int>& Array, int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        Merge_Sort(Array, left, mid);

```

 <b>Marwadi University</b>	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach	
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

```

        Merge_Sort(Array, mid + 1, right);
        Merge(Array, left, mid, right);
    }
}

int main() {
    vector<int> Array = { 12, 45, 57, 78, 89, 62, 7, 49, 21, 23 };
    int size = Array.size();
    cout << "Array Before Sorting :- " << endl;
    Print_Array(Array);
    Merge_Sort(Array, 0, size - 1);
    cout << "Array After Sorting :- " << endl;
    Print_Array(Array);
    return 0;
}

```

### Output :-

```

PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 4> cd "
d:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 4\" ; if (
$?) { g++ Merge_Sort.cpp -o Merge_Sort } ; if ($?) { .\Merge_Sort }
Array Before Sorting :-
12 45 57 78 89 62 7 49 21 23
Array After Sorting :-
7 12 21 23 45 49 57 62 78 89
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 4>

```

**Space Complexity:-** \_\_\_\_\_

**Justification: -**

---



---



---



---



---

**Time Complexity:**

**Best Case Time Complexity:** \_\_\_\_\_

**Justification: -**

---



---




---



---



---

 <b>Marwadi</b> University	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach	
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

**Worst Case Time Complexity:-** \_\_\_\_\_

**Justification: -**

---



---



---



---



---

## **Quick Sort**

### **Theory: -**

- Quicksort is a sorting algorithm based on the divide and conquer approach where
  - 1) An array is divided into subarrays by selecting a pivot element (element selected from the array).
  - 2) While dividing the array, the pivot element should be positioned in such a way that elements less than pivot are kept on the left side and elements greater than pivot are on the right side of the pivot.
  - 3) The left and right subarrays are also divided using the same approach. This process continues until each subarray contains a single element.
  - 4) At this point, elements are already sorted. Finally, elements are combined to form a sorted array.

### **Working of Quick Sort**

#### **1) Select the Pivot Element**

- There are different variations of quicksort where the pivot element is selected from different positions. Here, we will be selecting the rightmost element of the array as the pivot element.




#### **2) Rearrange the Array**

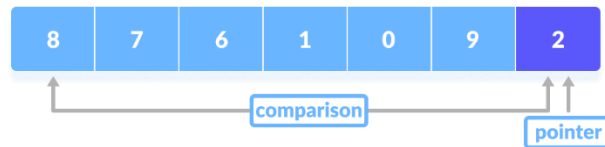
- Now the elements of the array are rearranged so that elements that are smaller than the pivot are put on the left and the elements greater than the pivot are put on the right.



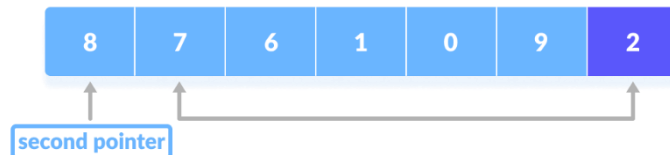
- Here's how we rearrange the array:**

 <b>Marwadi University</b>	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach	
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

- 1) A pointer is fixed at the pivot element. The pivot element is compared with the elements beginning from the first index.



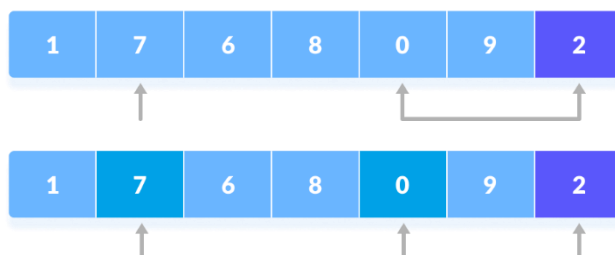
- 2) If the element is greater than the pivot element, a second pointer is set for that element.




- 3) Now, pivot is compared with other elements. If an element smaller than the pivot element is reached, the smaller element is swapped with the greater element found earlier.



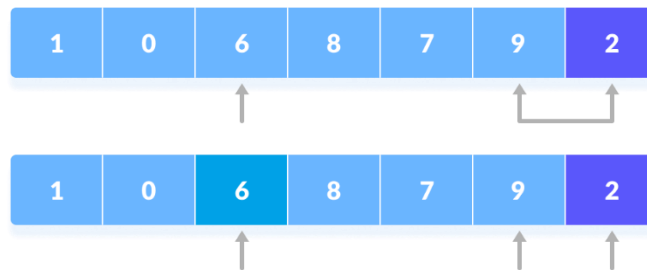
- 4) Again, the process is repeated to set the next greater element as the second pointer. And, swap it with another smaller element.



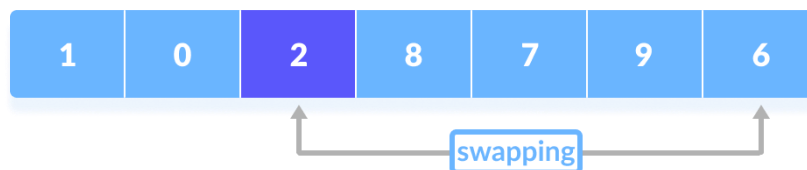


 <b>Marwadi University</b>	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim: Implementing the Sorting Algorithms Using Divide and Conquer Approach</b>	
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

5) The process goes on until the second last element is reached.



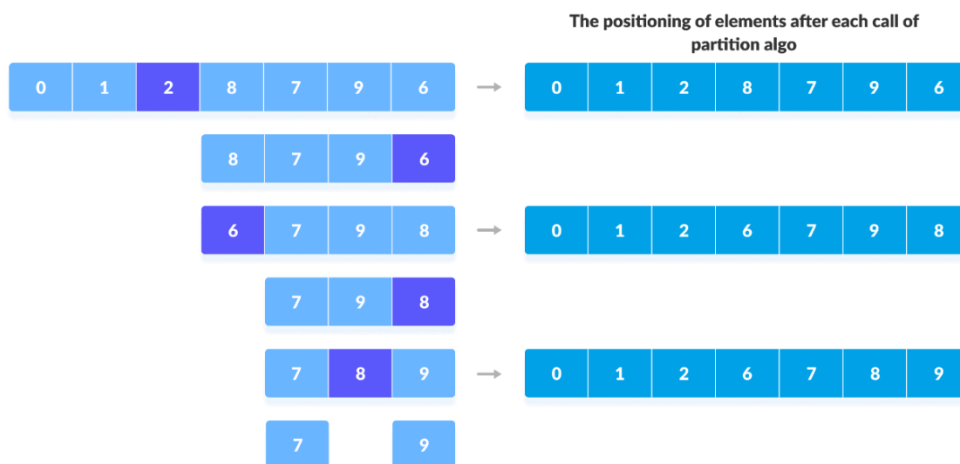
6) Finally, the pivot element is swapped with the second pointer.

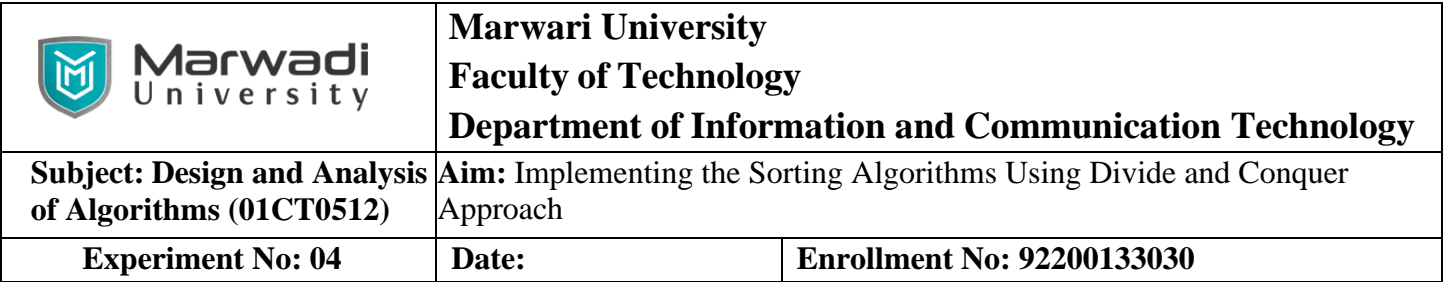


### 3) Divide Subarrays

- Pivot elements are again chosen for the left and the right sub-parts separately. And, step 2 is repeated.

quicksort(arr, pi, high)





**Marwari University**  
**Faculty of Technology**  
**Department of Information and Communication Technology**

### **Aim:** Implementing the Sorting Algorithms Using Divide and Conquer Approach

Date:

**Enrollment No: 92200133030**


This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

**Code :-**

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

void Swap(int& x, int& y) {
    int temp = x;
    x = y;
    y = temp;
}

void Print_Array(vector<int> Array) {
    for (int i = 0; i < Array.size(); i++) {
        cout << Array[i] << " ";
    }
}
```

 <b>Marwadi</b> University	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach	
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

```

    }
    cout << endl;
}

int Partion(vector<int>& Array, int left, int right) {
    int pivot = Array[left];
    while (true) {
        while (left < right && Array[left] < pivot) {
            left++;
        }
        while (left < right && Array[right] > pivot) {
            right--;
        }
        if (left >= right) {
            break;
        }


        Swap(Array[left], Array[right]);
    }

    Swap(Array[right], pivot);
    return right;
}

void Quick_Sort(vector<int>& Array, int left, int right) {
    if (left < right) {
        int Pivot_Index = Partion(Array, left, right);
        Quick_Sort(Array, left, Pivot_Index - 1);
        Quick_Sort(Array, Pivot_Index + 1, right);
    }
}

int main() {
    vector<int> Array = { 12, 45, 57, 78, 89, 62, 7, 49, 21, 23 };
    int size = Array.size();
    cout << "Array Before Sorting :- " << endl;
    Print_Array(Array);
    Quick_Sort(Array, 0, size - 1);
    cout << "Array After Sorting :- " << endl;
    Print_Array(Array);
    return 0;
}

```

 <b>Marwadi University</b>	<b>Marwari University</b> <b>Faculty of Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Design and Analysis of Algorithms (01CT0512)</b>	<b>Aim:</b> Implementing the Sorting Algorithms Using Divide and Conquer Approach	
<b>Experiment No: 04</b>	<b>Date:</b>	<b>Enrollment No: 92200133030</b>

**Output :-**

```
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 4> cd "d:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 4\" ; if ($?) { g++ Quick_sort.cpp -o Quick_sort } ; if ($?) { .\Quick_sort }
Array Before Sorting :-
12 45 57 78 89 62 7 49 21 23
Array After Sorting :-
7 12 21 23 45 49 57 62 78 89
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 4>
```

**Space Complexity:-** \_\_\_\_\_

**Justification: -**

---

---

---

---

---

**Time Complexity:**

**Best Case Time Complexity:** \_\_\_\_\_

**Justification: -**

---

---

---

---

---

**Worst Case Time Complexity:-** \_\_\_\_\_

**Justification: -**

---

---

---

---

---

---

**Conclusion:-**

---

---

---

---

---