



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment - 5

**Student Name:** Hemant Narain Jha

**UID:** 23BCS10022

**Branch:** BE-CSE

**Section/Group:** KRG-2B

**Semester:** 5<sup>th</sup>

**Date of Performance:** 9/10/25

**Subject Name:** Design and Analysis of Algorithms

**Subject Code:** 23CSH-301

**1. Aim:** Sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted. The elements can be read from a file or can be generated using the random number generator.

**2. Objective :** To understand quick sort.

**3. Input/Apparatus Used :** Quick sort concept is used.

**4. Procedure/Algorithm:**

Quick Sort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quick Sort that pick pivot in different ways.

- Always pick first element as pivot.
- Always pick last element as pivot (implemented below)
- Pick a random element as pivot.
- Pick median as pivot.

The key process in quick Sort is partition (). Target of partition() is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x.

### QUICKSORT

Quicksort is a sorting algorithm whose worst-case running time is ( $n^2$ ) on an input array of  $n$  numbers. In spite of this slow worst-case running time, quicksort is often the best practical choice for sorting because it is remarkably efficient on the average: its expected



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

running time is  $(n \lg n)$ , and the constant factors hidden in the  $(n \lg n)$  notation are quite small. It also has the advantage of sorting in place (see page 3), and it works well even in virtual memory environments.

The following procedure implements quicksort.

## 5. Code and Output—

```
import java.util.*;  
  
public class  
QuickSortExample {  
  
    // Partition function  
  
    public static 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] and  
                // arr[j]  
            }  
        }  
        return i;  
    }  
}
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

arr[j]

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

// Swap arr[i + 1] and

arr[high] (pivot)

int temp = arr[i + 1];

arr[i + 1] = arr[high];

arr[high] = temp;

return i + 1;

}

// QuickSort function

public static void

quickSort(int[] arr, int

low, int high) {



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
if (low < high) {  
  
    int pi =  
  
        partition(arr, low,  
  
        high);  
  
        // Recursively sort  
  
        subarrays  
  
        quickSort(arr, low,  
  
        pi - 1);  
  
        quickSort(arr, pi +  
  
        1, high);  
  
}  
  
}  
  
// Main method  
  
public static void  
main(String[] args) {  
  
    int[] arr = {10, 7, 8, 9,  
  
    1, 5};
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
quickSort(arr, 0,  
         arr.length - 1);  
  
System.out.println("S  
orted array:");  
  
for (int x : arr) {  
  
    System.out.print(x  
+ " ");  
  
}  
  
System.out.println();  
  
}  
  
}
```

## Your Output

```
Sorted array using quick sort: 1 5 7 8 9 10
```

## 6. Time Complexity:

Best case	Average case	Worst case
$O(n\log n)$	$O(n\log n)$	$O(n^2)$