Name - Raveet Kumar
Roll No. - 1804310039
Branch - Computer Science and Engineering
College - BIET, Jhansi

# Experiment - 04

## Aim-

We have to implement 2 different sorting algorithms and study the time complexities for  each of them and draw graphs of their best, worst and average case.

- *Counting sort*
- *Radix sort*

## Tools & Language Used-

- ❏ Java - for coding the algorithm and calculating time
- ❏ Python - for plotting graphs using matplotlib module.

## Code & Analysis-

- **Counting Sort**
    - ❖ Code

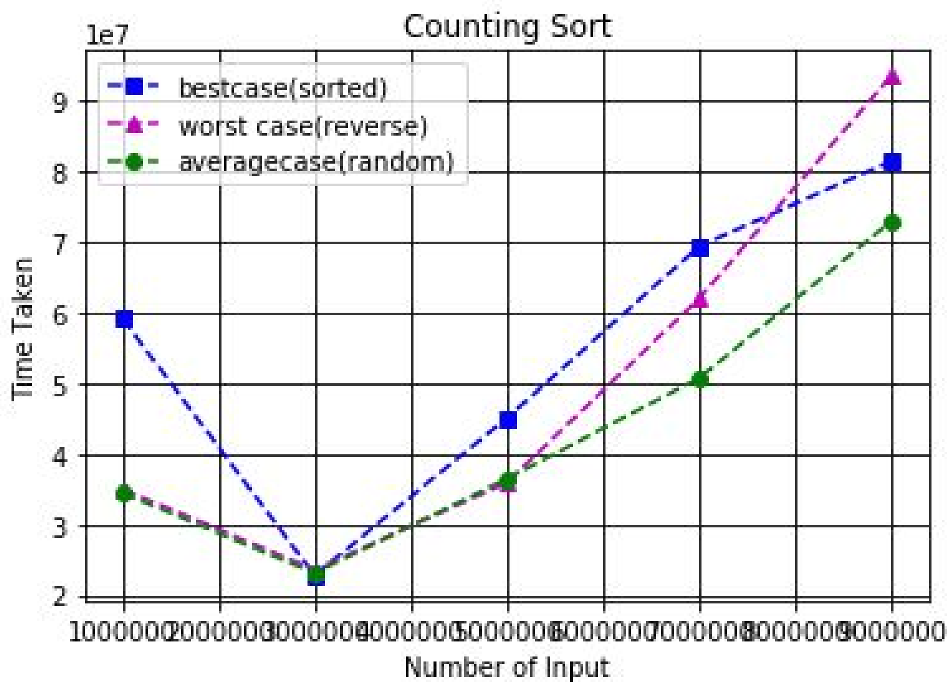
```java
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class CountingSort {

    public static void main(String[] args) throws IOException {

        File f=new File("D:\\Algorithm time Complexity
analysis\\counting_sort_analysis.txt");
```

```
        BufferedWriter bw=new BufferedWriter(new FileWriter(f,false),2);

        //1000000, 3000000, 5000000, 7000000, 9000000
        List<Integer> TestCase=Arrays.asList(1000000, 3000000, 5000000, 7000000,
9000000);
        int[] best;
        int[] worst;
        int[] avg;


        int k=0;
        bw.write(" \tNumber_of_Input\tTime_Taken\n");

        while(k < TestCase.size()) {

                int arrSize=TestCase.get(k);
                best=new int[arrSize];
                worst=new int[arrSize];
                avg=new int[arrSize];

                Random rand=new Random();// To Generate Random Numbers...

                for(int i=0;i<arrSize;i++) {
                        avg[i]=rand.nextInt(1001);              //Filling Numbers in
the range of (0, ........, 1000) in array of size arrSize
                }

                for(int i=0;i<arrSize;i++) {
                        best[i]=avg[i];
                }
                Arrays.sort(best);                              // To make a sorted
array... which we will use for best case..

                for(int i=0;i<arrSize;i++) {
                        worst[i]=best[arrSize-1-i];      // To make reverse order of
the Best case ... to Check the worst case..
                }

                // For Best Case
                long initialTime=System.nanoTime();

                best=counting_sort(best,arrSize);

                long TimeTaken=System.nanoTime()-initialTime;
                bw.write(String.format("Best_Case\t%d\t%d\n",arrSize,TimeTaken));

                // For Worst case
```

```
              initialTime=System.nanoTime();

              worst=counting_sort(worst,arrSize);

              TimeTaken=System.nanoTime()-initialTime;
              bw.write(String.format("Worst_Case\t%d\t%d\n",arrSize,TimeTaken));

              // For Average case
              initialTime=System.nanoTime();

              avg=counting_sort(avg,arrSize);

              TimeTaken=System.nanoTime()-initialTime;
              bw.write(String.format("Avg_Case\t%d\t%d\n",arrSize,TimeTaken));


//            //For TESTING only...
//
//            if(k==0) {
//                   for(int i=0;i<arrSize;i++) {
//                          System.out.print(best[i] +" ");
//                   }
//                   System.out.println();
//
//                   for(int i=0;i<arrSize;i++) {
//                          System.out.print(worst[i] +" ");
//                   }
//                   System.out.println();
//
//                   for(int i=0;i<arrSize;i++) {
//                          System.out.print(avg[i] +" ");
//                   }
//                   System.out.println();
//            }


              k++;
              System.out.println("Success");
        }
        bw.close();

    }
```

```
# Main Code

    private static int[] counting_sort(int[] arr, int n) {
        int max=Integer.MIN_VALUE;
        for(int i=0;i<n;i++) {
                if(arr[i] > max) {
                        max=arr[i];
                }
        }

        int[] count=new int[max+1];
        Arrays.fill(count,0);

        for(int i=0;i<n;i++) {
                count[arr[i]]++;
        }


        for(int i=1;i<count.length;i++) {
                count[i]=count[i]+count[i-1];
        }


        int[] res=new int[n];

        for(int i=n-1;i>=0;i--) {
                res[count[arr[i]]-1]=arr[i];
                count[arr[i]]--;
        }

        return res;
    }

}
```

❖ Graph



❖ Analysis

➢ **Best case Complexity= O(n+k)**

➢ **Worst case Complexity= O(n+k)**

➢ **Average case Complexity= O(n+k)**

Where, **n**=number of Element in Array
**k**=Base of Element (For Example for Number =10 and For Alphabets=26)

*-------------*------------*-----------------*-------------------*----------------------*--------*

## ● Radix Sort

❖ Code

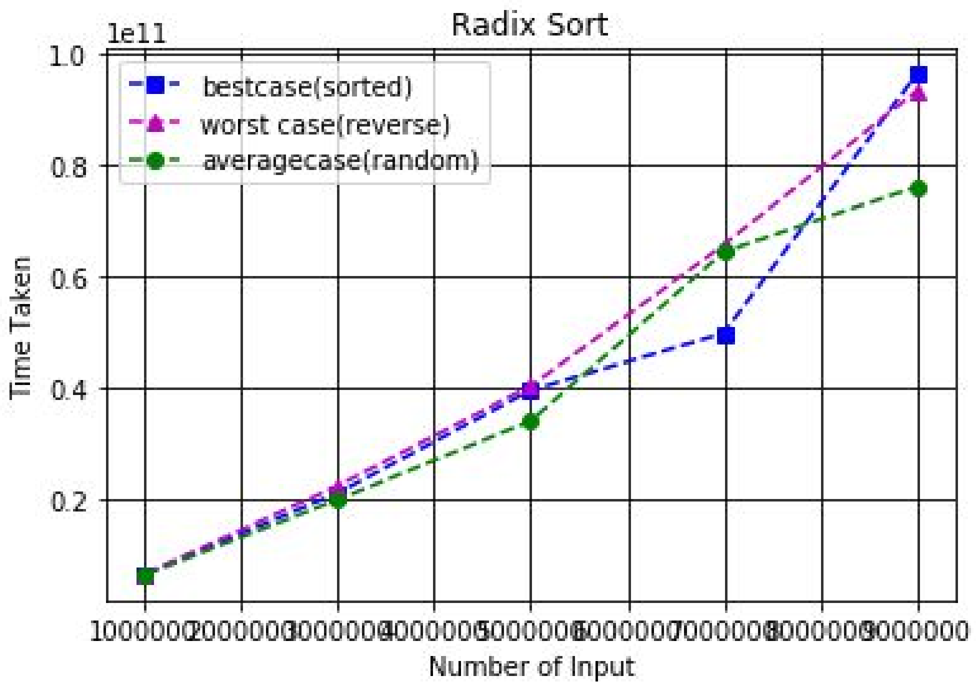
```java
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class RadixSort {
    public static void main(String[] args) throws IOException {

        File f=new File("D:\\Algorithm time Complexity
analysis\\Radix_sort_analysis.txt");
        BufferedWriter bw=new BufferedWriter(new FileWriter(f,false),2);

        //1000000, 3000000, 5000000, 7000000, 9000000
        List<Integer> TestCase=Arrays.asList(1000000, 3000000, 5000000, 7000000,
9000000);
        String[] best;
        String[] worst;
        String[] avg;


        int k=0;
        bw.write(" \tNumber_of_Input\tTime_Taken\n");

        while(k < TestCase.size()) {

               int arrSize=TestCase.get(k);
               best=new String[arrSize];
               worst=new String[arrSize];
               avg=new String[arrSize];

               Random rand=new Random();// To Generate Random Numbers...

               char[] chars = "abcdefghijklmnopqrstuvwxyz".toCharArray();

               for(int i=0;i<arrSize;i++) {
                      StringBuilder sb = new StringBuilder(20);

                      for (int j = 0; j < 20; j++) {
                           char c = chars[rand.nextInt(chars.length)];
                           sb.append(c);
                      }
                      String output = sb.toString();
                      avg[i]=output;            //Filling Numbers in the range of
(0, arrSize*10-1) in array of size arrSize
               }

               for(int i=0;i<arrSize;i++) {
                      best[i]=avg[i];
               }
               Arrays.sort(best);                             // To make a sorted
array... which we will use for best case..

               for(int i=0;i<arrSize;i++) {
                     worst[i]=best[arrSize-1-i];      // To make reverse order of
the Best case ... to Check worst case..
              }

              // For Best Case
              long initialTime=System.nanoTime();

              radix_sort(best,arrSize);

              long TimeTaken=System.nanoTime()-initialTime;
              bw.write(String.format("Best_Case\t%d\t%d\n",arrSize,TimeTaken));

              // For Worst case
              initialTime=System.nanoTime();

              radix_sort(worst,arrSize);

              TimeTaken=System.nanoTime()-initialTime;
              bw.write(String.format("Worst_Case\t%d\t%d\n",arrSize,TimeTaken));

              // For Average case
              initialTime=System.nanoTime();

              radix_sort(avg,arrSize);

              TimeTaken=System.nanoTime()-initialTime;
              bw.write(String.format("Avg_Case\t%d\t%d\n",arrSize,TimeTaken));


//            //For TESTING only...
//
//            if(k==0) {
//                   for(int i=0;i<arrSize;i++) {
//                          System.out.print(best[i] +" ");
//                   }
//                   System.out.println();
//
//                   for(int i=0;i<arrSize;i++) {
//                          System.out.print(worst[i] +" ");
//                   }
//                   System.out.println();
//
//                   for(int i=0;i<arrSize;i++) {
//                          System.out.print(avg[i] +" ");
//                   }
//                   System.out.println();
//            }
```

```
            k++;
            System.out.println("Success");
        }
        bw.close();

    }
```

## #Main Code

```
    private static void radix_sort(String[] arr, int n) {
        int maxLen=getMaxLen(arr);

        for(int pos=maxLen;pos >= 1;pos--) {
            countingSort(arr,pos);
        }
    }



    private static void countingSort(String[] arr, int pos) {
        String[] res=new String[arr.length];
        int n=arr.length;
        int[] count=new int[26];

        for(int i=0;i<n;i++) {
            count[(int)(arr[i].charAt(pos-1)-97)]++;
        }

        for(int i=1;i<26;i++) {
            count[i]=count[i]+count[i-1];
        }

        for(int i=n-1;i>=0;i--) {
            res[--count[(int)(arr[i].charAt(pos-1)-97)]]=arr[i];
        }

        for(int i=0;i<n;i++) {
            arr[i]=res[i];
        }
    }

    private static int getMaxLen(String[] arr) {
        int max=0;
        for(int i=0;i<arr.length;i++) {
            if(arr[i].length() > max) {
                    max=arr[i].length();
                }
            }
        return max;
    }

}
```

❖ <u>Graph</u>



❖ <u>Analysis</u>

➢ **Worst case Complexity= O((n+b)*d)**

➢ **Best case Complexity= O((n+b)*d)**

➢ **Average case Complexity= O((n+b)*d)**

Where, **n**=number of Element in Array
**b**=Base of Element (For Example for Number =10 and For Alphabets=26)
**d**=log(max(element)) base=b