# KIET Group of Institutions, Ghaziabad

## Department of Computer Applications

**(An ISO – 9001: 2015 Certified & 'A' Grade accredited Institution by NAAC)**

## Design and Analysis of Algorithm

## RCA 352: Session 2020-21

## DAA Lab

**Objective**: Implement the **Randomized Quick sort** algorithm to sort the given list of N numbers and plot graph.

| Scheduled Date: | Compiled Date: | Submitted Date: |
|---|---|---|
| 01-9-2020 | 01-9-2020 | 06-9-2020 |

### Algorithm:

QUICKSORT(A, p,r)

1. if p < r
2. then q ← PARTITION(A, p,r)
3. QUICKSORT(A, p, q − 1)
4. QUICKSORT(A, q + 1,r)

RANDOMIZED-PARTITION*(A, p, r)*

1. $i \leftarrow$ RANDOM*(p, r)*
2. exchange $A[r] \leftrightarrow A[i]$
3. **return** PARTITION*(A, p, r)*

RANDOMIZED-QUICKSORT*(A, p, r)*

1. **if** $p < r$
2. **then** $q \leftarrow$ RANDOMIZED-PARTITION*(A, p, r)*
3. RANDOMIZED-QUICKSORT*(A, p, q − 1)*
4. RANDOMIZED-QUICKSORT*(A, q + 1, r)*

### Program:

```
#include<time.h>
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<process.h>

int count=0;
int partition(int[10],int,int);
int myrandom(int,int);
```

```
void main()
 {
      void getdata(int[10],int);
      void putdata(int[10],int);
      void quick_sort(int[10],int,int);
      int i,a[100],n;
      clrscr();
      printf("Enter the Size of array=\n");
      scanf("%d",&n);
      getdata(a,n);
      printf("\nBefore soring=\n");
      putdata(a,n);
      quick_sort(a,0,n-1);
      printf("\nAfter sorting=\n");
      putdata(a,n);
      printf("\n For n = %d\n value of count is  %d",n,count);
      getch();

}

void getdata(int a[10],int n)
{
      int k;
      printf("Enter the %d Element for sorting\n",n);
      for(k=0;k<n;k++)
      {
       printf("[%d]=",k);
       scanf("%d",&a[k]);
      }
}
void putdata(int a[10], int n)
{
      int k;
      for(k=0;k<n;k++)
      {
            printf("%d\t",a[k]);
       }
       printf("\n");
}
void quick_sort(int a[],int p,int r)
{
   int q;
      count++;
```

```
      if(p<r)
        {
          count++;
          q=partition(a,p,r);
          count++;
          quick_sort(a,p,q-1);
          count++;
          quick_sort(a,q+1,r);
          count++;


        }
}
int myrandom(int lower,int upper)
{
int num;
count++;
srand(time(0));
count++;
num=(rand() % (upper - lower + 1)) + lower;
count++;
return num;
}


int partition(int a[],int p, int r)
{
  int y,x,i,j,temp;
  y=myrandom(p,r-1);
  temp=a[y];
  a[y]=a[r];
  a[r]=temp;

  x=a[r];
  i=p-1;
  count++;
  for(j=p;j<=r-1;j++)
   {
     count++;
     if(a[j]<x)
     {
          count++;
               i=i+1;
```

```
            count++;
                 temp=a[i];
            count++;
                 a[i]=a[j];
            count++;
                 a[j]=temp;
       }
       count++;
    }
    count++;
       temp=a[i+1];
    count++;
       a[i+1]=a[r];
    count++;
       a[r]=temp;
    count++;

  return(i+1);
}
```

**Output:**

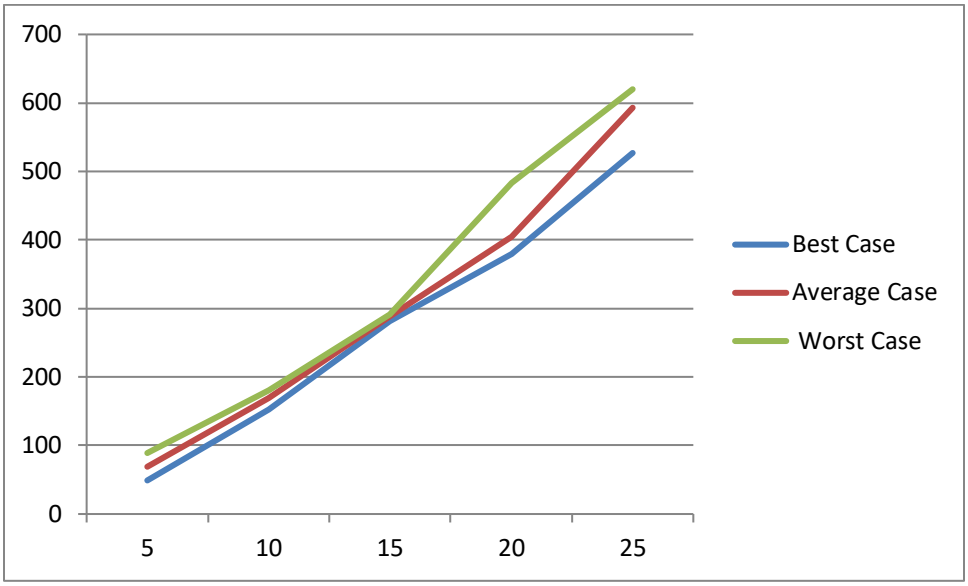
| Inputs | Best Case | Average Case | Worst Case |
|---|---|---|---|
| 5 | 49 | 69 | 69 |
| 10 | 153 | 169 | 181 |
| 15 | 281 | 289 | 291 |
| 20 | 379 | 405 | 483 |
| 25 | 527 | 593 | 595 |

**Design and Analysis of Algorithm**

**RCA 352: Session 2020-21**

**DAA Lab**

**Graph:**




**Conclusion:**

| Case | Running Time : Growth of Function mathematically | Running Time : Growth of Function after observing graph |
|---|---|---|
| Best Case | O(n log n) | O(n log n) |
| Average Case | O(n log n) | O(n log n) |
| Worst Case | O(n log n) | O(n log n) |