NAME:	Pranav Sadanand Kore
UID:	2021300065
SUBJECT	Design and Analysis of Algorithm
EXPERIMENT NO :	2 (Additional File)
AIM:	Quick Sort using Random Pivote

ALGORITHM:	**Random Pivote:**
	partition(arr[], lo, hi)
	pivot = arr[hi]
	i = lo // place for swapping
	for j := lo to hi – 1 do
	if arr[j] <= pivot then
	swap arr[i] with arr[j]
	i = i + 1
	swap arr[i] with arr[hi]
	return i
	
	partition_r(arr[], lo, hi)
	r = Random Number from lo to hi
	Swap arr[r] and arr[hi]
	return partition(arr, lo, hi)
	
	quicksort(arr[], lo, hi)
	if lo < hi
	p = partition_r(arr, lo, hi)
	quicksort(arr, lo , p-1)
	quicksort(arr, p+1, hi)

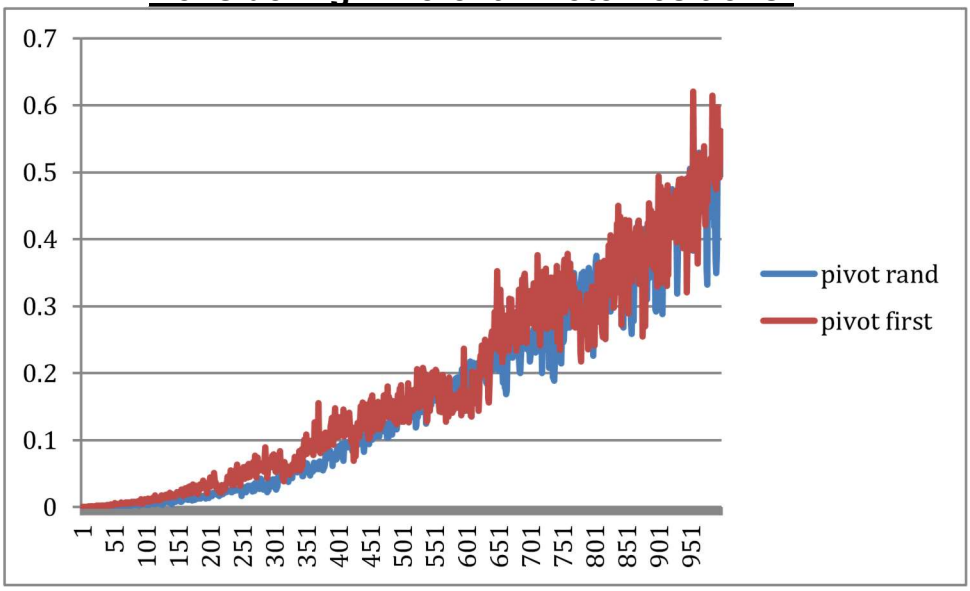
PROGRAM:	**Random Pivote:**

```
import java.util.*;

class RandomizedQsort
{
	static void random(int arr[],int low,int high)
	{

		Random rand= new Random();
		int pivot = rand.nextInt(high-low)+low;

		int temp1=arr[pivot];
		arr[pivot]=arr[high];
		arr[high]=temp1;
	}

	static int partition(int arr[], int low, int high)
	{
		random(arr,low,high);
		int pivot = arr[high];


		int i = (low-1); // index of smaller element
		for (int j = low; j < high; j++)
		{
			if (arr[j] < pivot)
			{
				i++;

				// swap arr[i] and arr[j]
				int temp = arr[i];
				arr[i] = arr[j];
				arr[j] = temp;
			}
		}
		int temp = arr[i+1];
		arr[i+1] = arr[high];
		arr[high] = temp;

		return i+1;
	}

	static void sort(int arr[], int low, int high)
	{
		if (low < high)
```

```
            {
                  int pi = partition(arr, low, high);
                  sort(arr, low, pi-1);
                  sort(arr, pi+1, high);
            }
      }

      static void printArray(int arr[])
      {
            int n = arr.length;
            for (int i = 0; i < n; ++i)
                  System.out.print(arr[i]+" ");
            System.out.println();
      }

      public static void main(String args[])
      {
            int arr[] = {10, 7, 8, 9, 1, 5};
            int n = arr.length;

            sort(arr, 0, n-1);

            System.out.println("Sorted array");
            printArray(arr);
      }
}
```

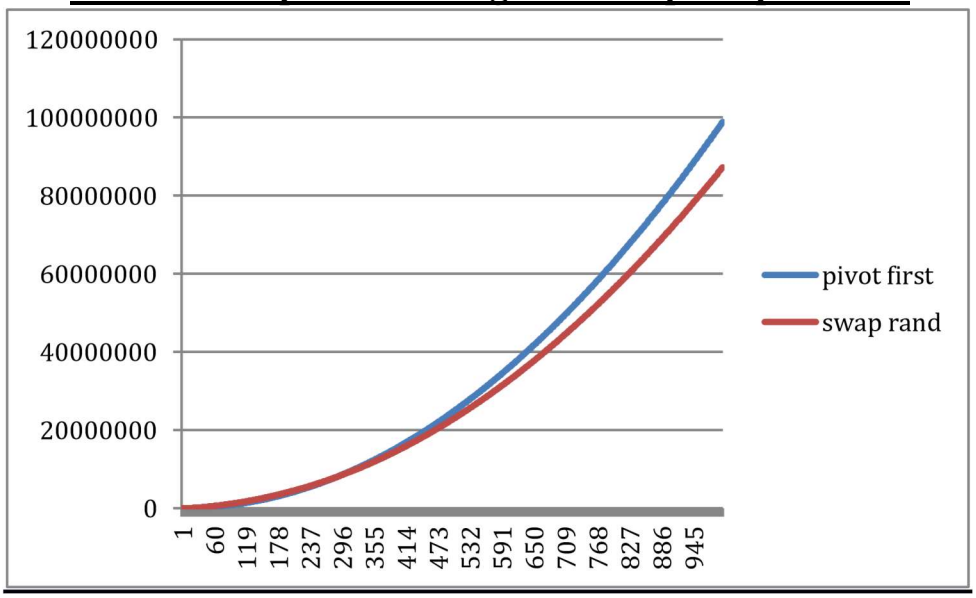
RESULT (SNAPSHOT)

Considering Different Pivote Positions:



We can see here that even when different pivot points are considered, the time complexity of rapid sort is nearly the same

Count of swaps considering different pivot positions:



The number of swaps necessary for quick sort when the pivot is at a random position is fewer than the number of swaps required for quick sort when the pivot is in the end position.

CONCLUSION:	I have studied the Quick Sort using different pivotes and their time complexity. Also, done the comparisons of time complexities.