Algorithm – 04
– Random-Quick-Sort

A. Problem Description

Quicksort applies the divide-and-conquer paradigm. Here is the three-step divide-and-conquer process for sorting a typical subarray A[p..r]:

Divide: Partition (rearrange) the array A[p..r] into two (possibly empty) subarrays A[p..q – 1] and A[q + 1..r] such that each element of A[p..q – 1] is less than or equal to A[q], which is, in turn, less than or equal to each element of A[q + 1..r]. Compute the index q as part of this partitioning procedure.

Conquer: Sort the two subarrays A[p..q – 1] and A[q + 1..r] by recursive calls to quicksort.

Combine: Because the subarrays are already sorted, no work is needed to combine them: the entire array A[p..r] is now sorted.

B. Description of Algorithm

```
RandomizedPartition(array, p, r)
    index = Random(p, r)
    base = array[index]
    create array 'a[]'
    create array 'b[]'

    for i = p to r + 1
        if i == index
            continue
        else if array[i] <= base:
            copy array[i] to a[]
        else
            copy array[i] to b[]

    x = p
    for i = 1 to a.length
        array[x] = a[i]
        x += 1
    array[x] = base
    q = x
    x += 1
    for i = 1 to b.length
        array[x] = b[i]
        x += 1

    return q


RandomizedQuickSort(array, p, r):
    if p < r
        q = RandomizedPartition(array, p, r)
        RandomizedQuickSort(array, p, r)
        RandomizedQuickSort(array, p, r)
```

$$T(n) = \begin{cases} O(1) & n <= 1 \\ 2T(n/2) + O(n) & n > 1 \end{cases}$$

$$=> T(n) = O(n \lg n)$$

C. Code.[Python]

```
#!/usr/bin/python
# Filename: Randomized-Quick-Sort.py

import random

def RandomizedPartition(array, p, r):
index = random.randint(p, r + 1)
base = array[index]
a = [0]
b = [0]

for i in range(p, r + 1):
    if i == index:
        continue
    elif array[i] <= base:
        a.append(array[i])
    elif array[i] > base:
        b.append(array[i])
    else:
        pass

x = p

for i in range(1, len(a)):
    array[x] = a[i]
    x += 1

array[x] = base
q = x
x += 1

for i in range(1, len(b)):
    array[x] = b[i]
    x += 1

return q

def RandomizedQucikSort(array, p, r):
if      p < r:
    q = RandomizedPartition(array, p, r)
    RandomizedQucikSort(array, p, q - 1)
    RandomizedQucikSort(array, q + 1, r)
```