Algorithm – 05
– Linear-Time-Selection

A. Problem Description

Select(i) is the i'th element in the sorted order of elements using 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: Computer that which subarray the element we want is in and search the element by recursive calls to Random-select.

Combine: When q is equal to r, A[q](or A[r]) is exactly the element that we want.

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


RandomizedSelect(array, p, r, k):
    if      p == r:
```

```
    return array[p]
i = RandomizedPartition(array, p, r)
j = i - p + 1
if      k <= j:
    return RandomizedSelect(array, p, i, k)
else:
    return RandomizedSelect(array, i + 1, r, k - j);
```

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

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

C. Code.[Python]

```python
#!/usr/bin/python
# Filename: Randomized-Select.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 RandomizedSelect(array, p, r, k):
    if    p == r:
        return array[p]
    i = RandomizedPartition(array, p, r)
    j = i - p + 1
    if    k <= j:
        return RandomizedSelect(array, p, i, k)
```

```
else:
      return RandomizedSelect(array, i + 1, r, k - j)
```