Fulfill numerical experiments of Example 4 presented in [1]. For this purpose, use any sorting method, determine its time complexity, code ranking/sorting of web search results (pages) and choosing the 10 most important within 34,279 results. Use any programming language. Describe the problem statement, input data, method, code and results obtained. Compare obtained results with ones described in Example 4 of [1]. Upload the file consisting of the code and related report to the UNT Canvas environment.

In [1]:
```
import random
import numpy as np
values = []
for i in range(34279):
    values.append(np.random.randint((100000)))
print(values)
```

```
[92719, 71422, 3093, 54619, 72194, 28996, 93939, 80742, 84794, 38744, 897
09, 79401, 52042, 36568, 11126, 92445, 24907, 98069, 81358, 97991, 25909,
79562, 45150, 86602, 66224, 11623, 31089, 64144, 76524, 4758, 48004, 7024
9, 36896, 12390, 24058, 49070, 85178, 48698, 87890, 31926, 4400, 29465, 9
0655, 75055, 26933, 914, 98822, 48337, 53862, 49884, 94943, 48077, 91657,
71334, 45705, 89214, 48788, 13825, 33302, 7108, 42968, 67482, 38061, 7474
3, 11973, 68835, 98580, 93594, 26527, 2543, 46660, 72228, 10825, 49366, 9
0237, 80105, 58913, 3217, 38567, 34296, 31588, 20291, 28517, 44544, 7133
9, 70549, 46720, 24937, 40832, 42790, 91020, 53809, 16903, 36390, 76809,
71089, 31115, 16515, 70447, 60793, 31235, 11997, 30953, 31848, 8331, 8781
6, 46414, 3099, 67012, 48740, 89463, 25198, 62532, 51566, 53110, 56508, 3
5139, 72920, 15670, 64253, 46671, 51205, 83318, 16962, 65202, 63674, 954
7, 44402, 33860, 55148, 66072, 52529, 93578, 93338, 26498, 47277, 32328,
18789, 48145, 78818, 55888, 5464, 94624, 3921, 75261, 31615, 18756, 6448
6, 42296, 32917, 89872, 92202, 6591, 88803, 42054, 19541, 807, 82919, 523
45, 32259, 41697, 68633, 27036, 33856, 52212, 43461, 93431, 82717, 79416,
99462, 50490, 82992, 75448, 670, 1142, 80667, 71594, 14667, 7203, 87810,
60737, 25312, 44468, 49672, 69911, 40316, 97149, 37472, 88688, 82227, 570
36, 97238, 23764, 32181, 6639, 90246, 12129, 15406, 59442, 3729, 10427, 6
```

In [2]:
```
# Quick sort in Python
# time complexity O(n log n)

def QuickSort(arr):

    elements = len(arr)

    if elements < 2:
        return arr

    current_position = 0

    for i in range(1, elements):
         if arr[i] <= arr[0]:
              current_position += 1
              temp = arr[i]
              arr[i] = arr[current_position]
              arr[current_position] = temp

    temp = arr[0]
    arr[0] = arr[current_position]
    arr[current_position] = temp

    left = QuickSort(arr[0:current_position])
    right = QuickSort(arr[current_position+1:elements])

    arr = left + [arr[current_position]] + right

    return arr

array_to_be_sorted = values

print("Original Array length is: ",len(array_to_be_sorted))
print("\n")
print("Sorted Array of top 10 results: ", QuickSort(array_to_be_sorted)[:10
```

```
Original Array length is:  34279


Sorted Array of top 10 results:  [0, 4, 8, 11, 21, 22, 28, 34, 35, 37]
```

In [3]:
```
# Merge sort in Python
# time complexity O(n log n)

def mergeSort(arr):

    if len(arr) > 1:

        a = len(arr)//2

        l = arr[:a]

        r = arr[a:]

        # Sort the two halves

        mergeSort(l)

        mergeSort(r)

        b = c = d = 0

        while b < len(l) and c < len(r):

            if l[b] < r[c]:

                arr[d] = l[b]

                b += 1

            else:

                arr[d] = r[c]

                c += 1

            d += 1

        while b < len(l):

            arr[d] = l[b]

            b += 1

            d += 1

        while c < len(r):

            arr[d] = r[c]

            c += 1

            d += 1


def printList(arr):
```

```
    for i in range(len(arr)):

        print(arr[i], end=" ")

    print()


# Driver program

if __name__ == '__main__':

    arr = values

    mergeSort(arr)

    lst = []

print("Sorted array length is : ", len(values))

print("\n")

print("Sorted array of top 10 results are : ")
for i in range(len(arr)):
    lst.append(arr[i])
print(lst[:10])
```

```
Sorted array length is :  34279


Sorted array of top 10 results are :
[0, 4, 8, 11, 21, 22, 28, 34, 35, 37]
```

Description:

For sorting a large number that is "34279", I have picked quick sort. It has time complexity O(n log n) which is same as merge sort. Sorting technique might differ in time complexity, but for same input the result has to be same.

Like Merge Sort, QuickSort is a Divide and Conquer algorithm. It picks an element as a pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

The key process in quickSort is a partition(). The target of partitions is, given an array and an element x of an array as the pivot, put x at its correct position in a sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.

Problem Statement: Sort the numbers that are random in range between 1 to 34,279 and print top 10 results.

Input data: I have taken an array of 34279 numbers that are randomly generated within the given range 100000.

Method: As described above, that is Quick sort algorithm.

Code: It is same as above.

Results: Printed the top 10 sorted elements or numbers from the sorted array.

In [11]:
```
# Selection sort in Python
# time complexity O(n*n)


def selectionSort(array, size):

    for ind in range(size):
        min_index = ind

        for j in range(ind + 1, size):

            if array[j] < array[min_index]:
                min_index = j

        (array[ind], array[min_index]) = (array[min_index], array[ind])

arr = values

size = len(arr)
selectionSort(arr, size)

print('The array after sorting in Ascending Order by selection sort is:')
print(len(arr))

print("\n")

print('The array after sorting having top 10 results in Ascending Order by
print(arr[:10])
```

```
The array after sorting in Ascending Order by selection sort is:
34279


The array after sorting having top 10 results in Ascending Order by selec
tion sort is:
[0, 4, 8, 11, 21, 22, 28, 34, 35, 37]
```

In [ ]: