Using any sorting technique, sort random numbers having your own range specified, and display the top 10 results. [Least time complexity is preferrable]

In [40]:
```
import random

values = random.sample(range(1, 1500), 50)
print(values)
```

```
[1049, 896, 501, 19, 607, 91, 684, 327, 768, 915, 871, 632, 1386, 1269, 470, 9, 261, 737, 300, 1259, 6
11, 754, 771, 73, 1219, 190, 702, 498, 931, 900, 774, 1109, 1461, 891, 465, 1077, 717, 306, 263, 148,
687, 803, 1062, 716, 1268, 1186, 224, 1014, 200, 733]
```

In [41]:
```
# 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(arr)

print("\n")

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

```
The array after sorting in Ascending Order by selection sort is:
[9, 19, 73, 91, 148, 190, 200, 224, 261, 263, 300, 306, 327, 465, 470, 498, 501, 607, 611, 632, 684, 6
87, 702, 716, 717, 733, 737, 754, 768, 771, 774, 803, 871, 891, 896, 900, 915, 931, 1014, 1049, 1062,
1077, 1109, 1186, 1219, 1259, 1268, 1269, 1386, 1461]


The array after sorting having top 10 results in Ascending Order by selection sort is:
[9, 19, 73, 91, 148, 190, 200, 224, 261, 263]
```

In [43]:
```
# Insertion sort in Python
# time complexity O(n)

def insertionSort(arr):

    for i in range(1, len(arr)):

        key = arr[i]
        j = i-1
        while j >=0 and key < arr[j] :
                arr[j+1] = arr[j]
                j -= 1
        arr[j+1] = key



arr = values
insertionSort(arr)
lst = []
print("Sorted array is : ")
for i in range(len(arr)):
    lst.append(arr[i])
print(lst)

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 is :
[9, 19, 73, 91, 148, 190, 200, 224, 261, 263, 300, 306, 327, 465, 470, 498, 501, 607, 611, 632, 684, 6
87, 702, 716, 717, 733, 737, 754, 768, 771, 774, 803, 871, 891, 896, 900, 915, 931, 1014, 1049, 1062,
1077, 1109, 1186, 1219, 1259, 1268, 1269, 1386, 1461]


Sorted array of top 10 results are :
[9, 19, 73, 91, 148, 190, 200, 224, 261, 263]
```

In [50]:
```
# 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: ",array_to_be_sorted)
print("\n")
print("Sorted Array: ",QuickSort(array_to_be_sorted[:10]))
```

```
Original Array:  [9, 19, 73, 91, 148, 190, 200, 224, 261, 263, 300, 306, 327, 465, 470, 498, 501, 607,
611, 632, 684, 687, 702, 716, 717, 733, 737, 754, 768, 771, 774, 803, 871, 891, 896, 900, 915, 931, 10
14, 1049, 1062, 1077, 1109, 1186, 1219, 1259, 1268, 1269, 1386, 1461]


Sorted Array:  [9, 19, 73, 91, 148, 190, 200, 224, 261, 263]
```

Hence, quicksort is considered as fastest sorting algorithm and best performance algorithm.

In [ ]: