In [41]:
```
import numpy as np
import random
import time


# ------------ CONST ---------------

ARRAY_LENGTH = 100000
ARRAY_MAX_ITEM = 1000000000

FILES_TOTAL = 50
MAX_N = 90

# ------------ FUNCTIONS ---------------

def writeInFiles():
    for file in range(FILES_TOTAL):
        f=open("arrays-{}.txt".format(file), "w+")

        for i in range(ARRAY_LENGTH):
            f.write("{}".format((int)(random.random() * (ARRAY_MAX_
ITEM+1))) + "\n")
        f.close()

def readFile(index):
    f=open("arrays-{}.txt".format(index), "r")
    arr = []

    for line in f:
        arr.append(int(line))

    return arr

def insertionSort(A):
    for j in range(1,len(A)):
        key = A[j]

        i = j-1
        while (i > -1) and key < A[i]:
            A[i+1]=A[i]
            i=i-1

        A[i+1] = key
    return A

def findIndex(arr, p, r):
    pivot = arr[(p + r)//2] # Округление
    i = p
    j = r

    while i <= j:
```

```
        while arr[i] < pivot:
            i = i + 1

        while arr[j] > pivot:
            j = j - 1

        if i <= j:
            arr[i], arr[j] = arr[j], arr[i]
            i = i + 1
            j = j - 1

    return i

def quickSort(arr, p, r):
    if p < r:
        index = findIndex(arr, p, r)

        if p < index - 1:
            quickSort(arr, p, index-1)

        if r > index:
            quickSort(arr, index, r)

    return arr

def hybridSort(arr, p, r, n):
    if p < r:
        index = findIndex(arr, p, r)

        if p < index - 1:
            if n > 0 and index - 1 - p <= n:
                arr[p:(index-1)] = insertionSort(arr[p:(index-1)])
            else:
                hybridSort(arr, p, index-1, n)
        if r > index:
            if n > 0 and r - index <= n:
                arr[index:r] = insertionSort(arr[index:r])
            else:
                hybridSort(arr, index, r, n)

    return arr
```

In [42]:

```
# ------------ MAKE FILES ---------------
# writeInFiles()
```

In [43]:
```
# ------------ QUICKSORT ---------------

timeArray = []

for i in range(FILES_TOTAL):
    arr = readFile(i)

    start = time.time()
    arr = quickSort(arr, 0, len(arr)-1)

    end = time.time()
    timeArray.append(end-start)

avgQuickSort = np.mean(timeArray)
print("Среднее время сортировки QuickSort - {}".format(avgQuickSort
)) # 0.33631832122802735
```

```
Среднее время сортировки QuickSort - 0.33631832122802735
```

In [ ]:
```
# ------------ HYBRID -----------------
for n in range(MAX_N, 1, -1):
    timeArray = []

    for i in range(FILES_TOTAL):
        arr = readFile(i)

        start = time.time()
        arr = hybridSort(arr, 0, len(arr)-1, n)
        end = time.time()

        timeArray.append(end-start)

    avgHybridSort = np.mean(timeArray)

    print("n = {}, медленнее на {}".format(n, avgHybridSort - avgQu
ickSort))

    if(avgHybridSort < avgQuickSort):
        print("При n = {0} гибридная быстрее на {1}".format(n, avgQ
uickSort-avgHybridSort))
```

```
n = 90, медленнее на 0.07471851348876951
n = 89, медленнее на 0.07127402305603026
n = 88, медленнее на 0.08598408699035642
n = 87, медленнее на 0.0840213108062744
n = 86, медленнее на 0.1174923515319824
n = 85, медленнее на 0.10798619747161864
n = 84, медленнее на 0.11414143562316892
n = 83, медленнее на 0.07369286537170411
n = 82, медленнее на 0.08119960784912111
n = 81, медленнее на 0.083610315322876
n = 80, медленнее на 0.07092599391937254
n = 79, медленнее на 0.06257050037384032
n = 78, медленнее на 0.08285371780395506
n = 77, медленнее на 0.08101520061492917
n = 76, медленнее на 0.05778845787048337
n = 75, медленнее на 0.038054356575012216
n = 74, медленнее на 0.05886912822723389
n = 73, медленнее на 0.0769012784957886
n = 72, медленнее на 0.05830707550048825
n = 71, медленнее на 0.07539163589477538
n = 70, медленнее на 0.06440838813781735
n = 69, медленнее на 0.054685463905334464
n = 68, медленнее на 0.06543516635894775
n = 67, медленнее на 0.028558464050292942
n = 66, медленнее на 0.020099267959594713
n = 65, медленнее на 0.029464230537414537
n = 64, медленнее на 0.07793584346771237
n = 63, медленнее на 0.027925958633422843
n = 62, медленнее на 0.03386062622070313
n = 61, медленнее на 0.03329984188079832
n = 60, медленнее на 0.029093976020812984
n = 59, медленнее на 0.03774515628814695
n = 58, медленнее на 0.04304813385009765
n = 57, медленнее на 0.02953320503234863
n = 56, медленнее на 0.01972925186157226
n = 55, медленнее на 0.030196323394775404
n = 54, медленнее на 0.015111236572265596
n = 53, медленнее на 0.015794620513916002
n = 52, медленнее на -0.005399227142333984
При n = 52 гибридная быстрее на 0.005399227142333984
n = 51, медленнее на 0.008056488037109344
n = 50, медленнее на 0.026635975837707493
n = 49, медленнее на -0.007959113121032702
При n = 49 гибридная быстрее на 0.007959113121032702
n = 48, медленнее на -0.0012114620208740212
При n = 48 гибридная быстрее на 0.0012114620208740212
n = 47, медленнее на -0.004054880142211947
При n = 47 гибридная быстрее на 0.004054880142211947
n = 46, медленнее на -0.01287998199462892
При n = 46 гибридная быстрее на 0.01287998199462892
```