Reporte: Juan Reyna #10- 10883

Actividad 3: En eta actividad se requiere determinar si hay una secuencia de tamaño **n,** suficientemente pequeño, para el cual alguno de los algoritmos de ordenamientos simples (insertion sort, selection sort, bubble sort,Shell sort) es más rápido que mergesort three. Para determinar este **n**, fue diseñado un programa de comparación llamado: **comparación_on2_nlgn.py,** que contrasta los tiempos entre mergesort three y los algoritmos de ordenamiento simple, comparándolo según el tamaño de la secuencia y mostrando su gráfica. El tipo de secuencia que se usa para determinar el tamaño fueron secuencias de tipo sreal y se hicieron tres intentos para hacer la aplicación de dichos algoritmos. Haciendo diversas pruebas el programa **comparación_on2_nlgn.py,** determina de manera consistente que para **n = 500** shellsort ordena más rápido las secuencias que se les suministra. Por ser **n = 500** un numero bajo, gráficamente es imperceptible determinar dicho resultado cuando se indican diversos tamaños de secuencias muy grandes.

```
     Unnamed: 0     Algoritmos  Num. de elementos  Intento  Tiempo(seg)
96           36  InsertionSort                500        1     0.026983
97           37     BubbleSort                500        1     0.034982
98           38  SelectionSort                500        1     0.007995
99           39      ShellSort                500        1     0.001999
100          40  InsertionSort                500        2     0.023985
101          41     BubbleSort                500        2     0.031980
102          42  SelectionSort                500        2     0.007995
103          43      ShellSort                500        2     0.001999
104          44  InsertionSort                500        3     0.025987
105          45     BubbleSort                500        3     0.033978
106          46  SelectionSort                500        3     0.007995
107          47      ShellSort                500        3     0.001998
    Unnamed: 0  Algoritmos  Num. de elementos  Intento  Tiempo(seg)
38          38  MergeThree                500        1     0.001000
42          42  MergeThree                500        2     0.001999
46          46  MergeThree                500        3     0.000999

ShellSort es mas rápido que: MergeThree
ShellSort 0.0019986629486083
MergeThree 0.0019989013671875

ShellSort es mas rápido que: MergeThree
ShellSort 0.0019984245300292
MergeThree 0.0019989013671875
```

Figura 1. Comparación

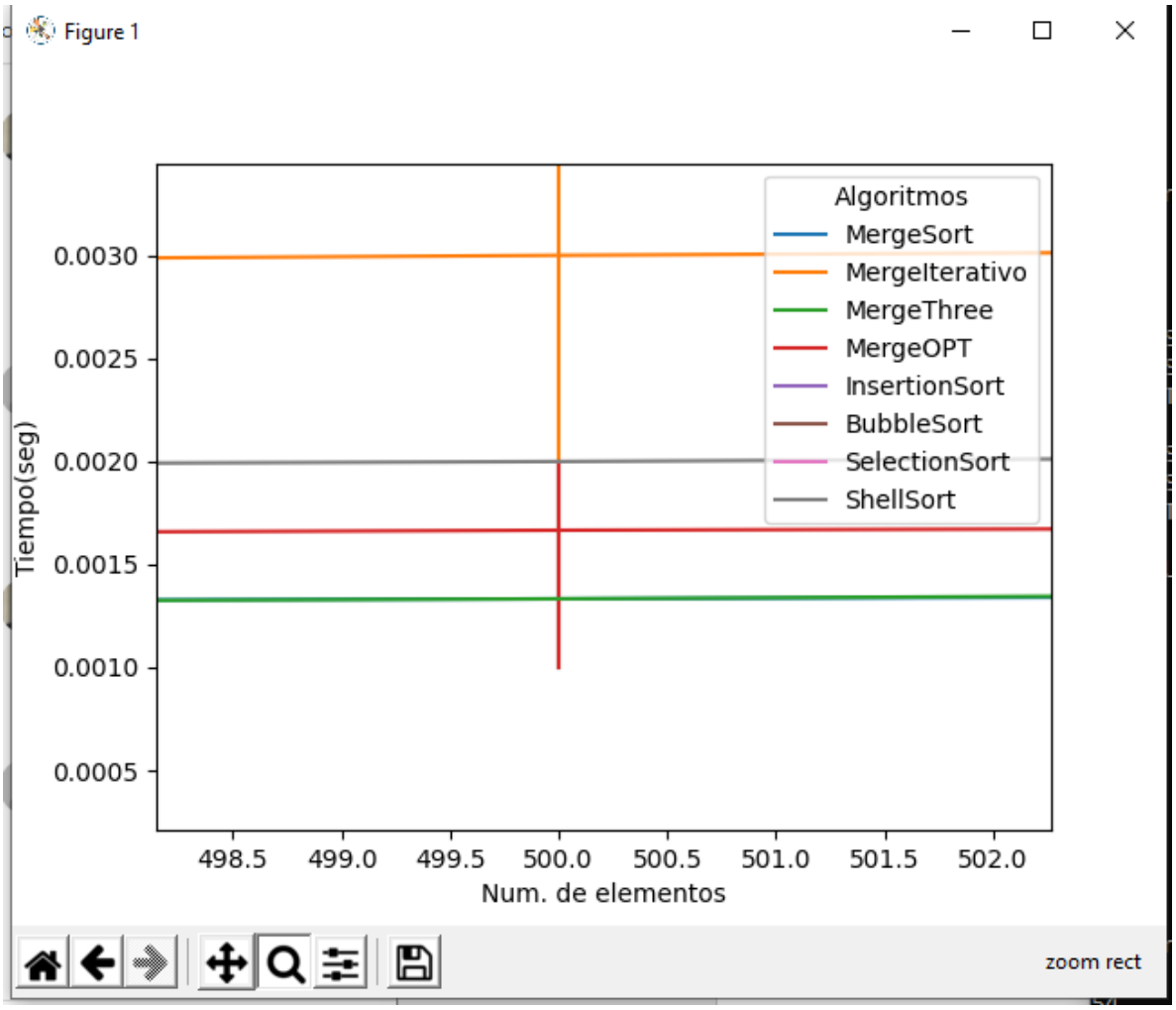


Figura 2. Gráfica de comparación

Actividad 7: Modificando el programa **comparación_on2_nlgn.py**, para analizar el desempeño de mergesortopt nos fijamos que este es mucho más rápido que los todos los algoritmos de ordenamiento para los parámetros exigidos en la **actividad 6.**

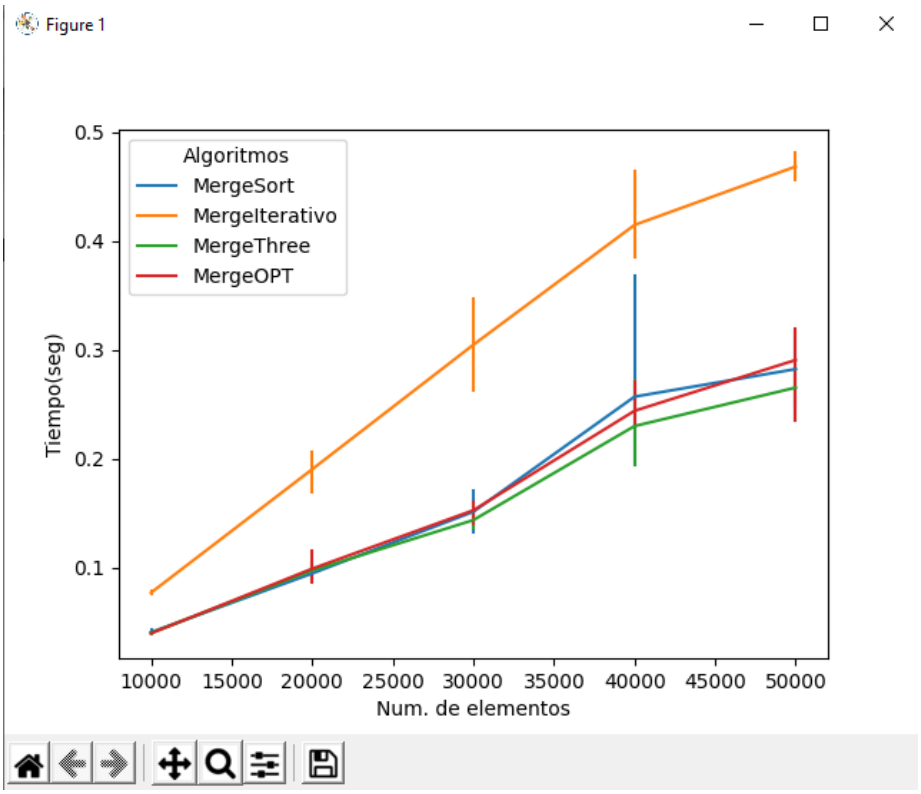


Figure 1
Algoritmos
MergeSort
MergeIterativo
MergeThree
MergeOPT
Tiempo(seg)
0.5
0.4
0.3
0.2
0.1
10000
15000
20000
25000
30000
35000
40000
45000
50000
Num. de elementos

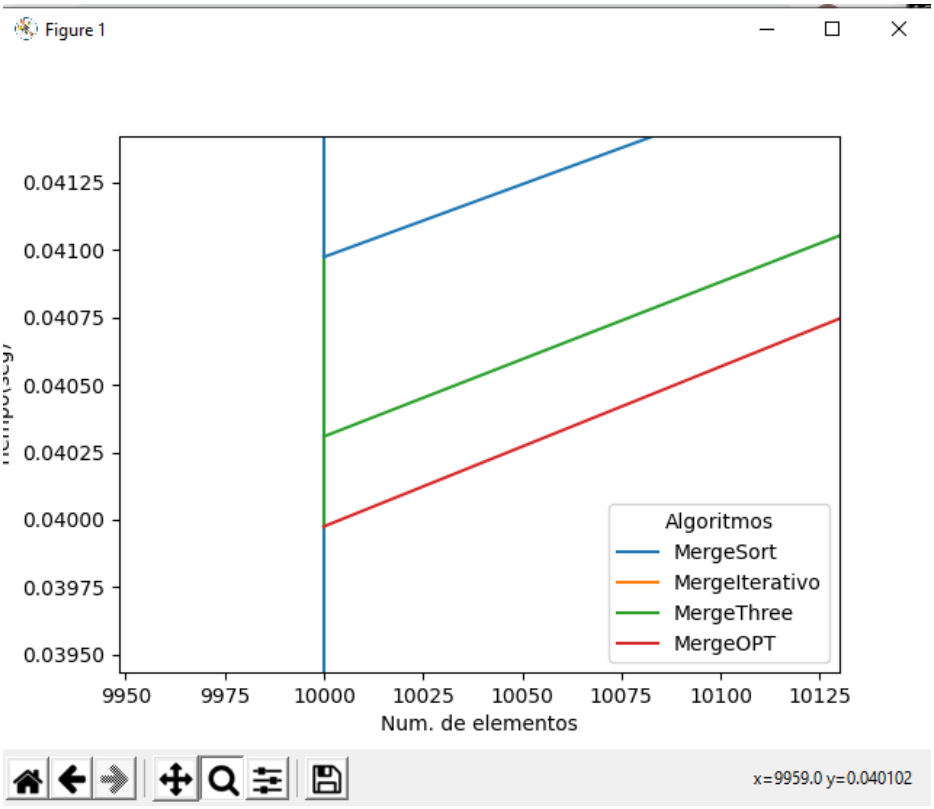


Figure 1
0.04125
0.04100
0.04075
0.04050
0.04025
0.04000
0.03975
0.03950
9950
9975
10000
10025
10050
10075
10100
10125
Num. de elementos
Algoritmos
MergeSort
MergeIterativo
MergeThree
MergeOPT
x=9959.0 y=0.040102

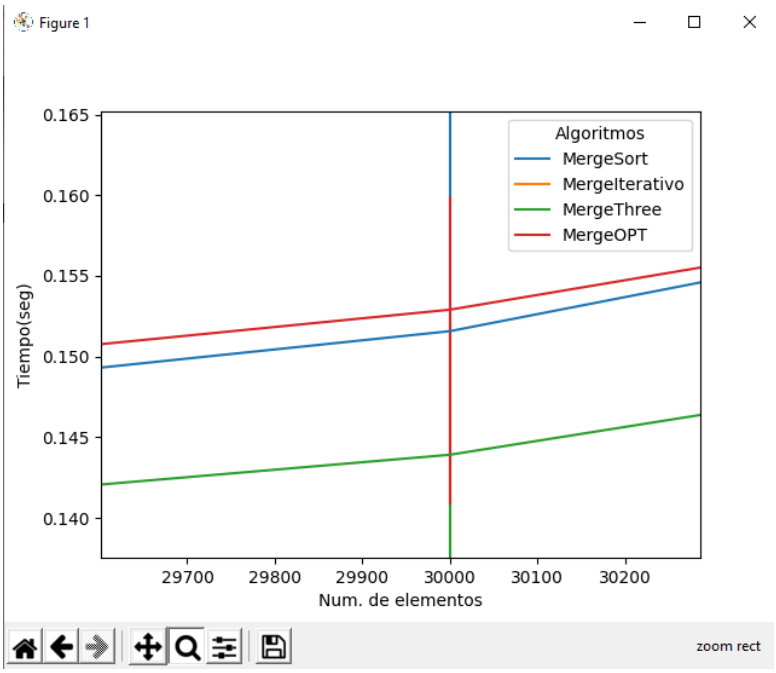


Figure 1
Algoritmos
MergeSort
MergeIterativo
MergeThree
MergeOPT
Tiempo(seg)
0.165
0.160
0.155
0.150
0.145
0.140
29700
29800
29900
30000
30100
30200
Num. de elementos
zoom rect

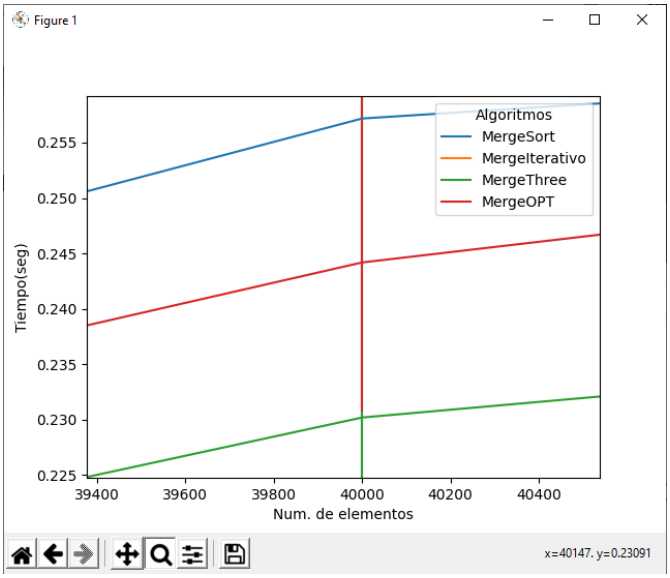
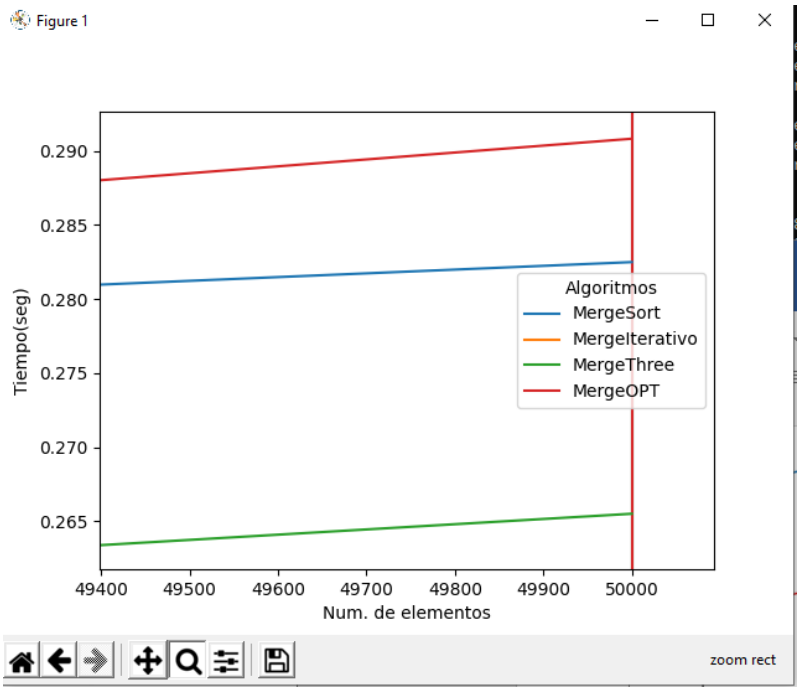


Figure 1
Algoritmos
MergeSort
MergeIterativo
MergeThree
MergeOPT
Tiempo(seg)
0.255
0.250
0.245
0.240
0.235
0.230
0.225
39400
39600
39800
40000
40200
40400
Num. de elementos
x=40147, y=0.23091



Una ventaja del Shellsort es su eficiencia para secuencias de longitud suficientemente pequeña. Shellsort es el algoritmo más rápido de todos aquellos de complejidad O(n^2), como es el caso del Bubble sort y el Insertion sort. Combinado con merge sort (mergesortopt) es un algoritmo más rápido que los otros involucrados en esta actividad.