**Documento Análisis de Algoritmos**

**Nombres:** Nicolás Andrés Tobo Urrutia, Manuel Ricardo Sosa **Códigos:** 201817465, 201815393

1. **SHELLSORT(3x+1):** El mejor caso de ShellSort se presenta cuando el arreglo esta ordenado de forma ascendente y su complejidad es de O () con una aproximación de ~O(). Su peor caso es cuando los elementos están ordenados de forma descendente y su complejidad es de O () con una aproximación de ~O()

**MERGESORT:**  El mejor caso de MergeSort se presenta cuando el arreglo esta ordenado de forma ascendente y su complejidad es de O (). ) donde el algoritmo ahorra tiempo debido a que no realiza intercambios. Su peor caso es cuando los elementos están ordenados de forma descendente y su complejidad es de O (). Cabe resaltar que si se aproximan las dos complejidades en el mejor y el peor caso su complejidad es de ~O (), logrando obtener casi el mismo tiempo de ejecución.

**QUICKSORT:** El mejor caso de QuickSort se presenta cuando el arreglo esta desordenado y su primer pivote tiene una posición en la mitad del arreglo, así cuando se hace la llamada recursiva a sus subarreglos estos realizan la operación más rápidamente, su complejidad en este caso es de O () con una aproximación de ~O (). Su peor caso es cuando el arreglo de encuentra ordenado ascendentemente y el pivote inicial es el primer elemento, debido a que formaría un subarreglo con todos los elementos menos el inicial, haciendo que la recursión de esta parte sea mucha mas lenta, su complejidad en este caso es de O () con una aproximación de ~O ().

**b. SHELLSORT(3x+1):**

* Es un algoritmo in place debido a que no genera una copia auxiliar para poder organizarse.
* Es un algoritmo adaptativo debido a que su complejidad cambia significativamente si está en el mejor o peor caso.
* No es un algoritmo estable debido a que no mantiene un orden relativo entre las llaves del arreglo, quiere decir que si había un orden preexistente entre las llaves del arreglo este algoritmo no lo conserva.

**MERGESORT:**

* No es un Algoritmo in place debido a que genera una copia auxiliar para poder organizarse.
* No es un Algoritmo adaptativo debido a que su complejidad no cambia significativamente si está en el mejor o peor caso.
* Si es un algoritmo estable debido a que mantiene un orden relativo entre las llaves del arreglo, quiere decir que si había un orden preexistente entre las llaves del arreglo este algoritmo lo conserva.

**QUICKSORT:**

* Es un Algoritmo in place debido a que no genera una copia auxiliar para poder organizarse.
* Es un Algoritmo adaptativo debido a que su complejidad cambia significativamente si está en el mejor o peor caso.
* No es un algoritmo estable debido a que no mantiene un orden relativo entre las llaves del arreglo, quiere decir que si había un orden preexistente entre las llaves del arreglo este algoritmo no lo conserva.

**C. Caso general**

|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño muestra | Shellsort (mseg) | Mergesort (mseg) | Quicksort (mseg) |
| 30000 | 130 | 54 | 120 |
| 60000 | 226 | 107 | 235 |
| 90000 | 373 | 178 | 389 |
| 120000 | 2004 | 668 | 512 |
| 150000 | 2207 | 917 | 655 |
| 180000 | 2578 | 1340 | 843 |
| 210000 | 4812 | 1870 | 1311 |
| 240000 | 5137 | 1679 | 1177 |

**Conclusiones:** En un caso general el método de ordenamiento Quicksort es más rápido que los otros dos y que el método de ordenamiento más lento es Shellsort.

**D. Caso ordenados ascendentemente**

|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño muestra | Shellsort (mseg) | Mergesort (mseg) | Quicksort (mseg) |
| 30000 | 164 | 93 | 120 |
| 60000 | 477 | 245 | 214 |
| 90000 | 665 | 354 | 320 |
| 120000 | 1000 | 574 | 491 |
| 150000 | 1391 | 614 | 641 |
| 180000 | 1803 | 737 | 1006 |
| 210000 | 2221 | 883 | 1027 |
| 240000 | 3007 | 1057 | 1073 |

**Conclusiones:** En el caso ordenado ascendentemente se puede ver que el más demorado es el Shellsort,y que el Mergesort es el método de ordenamiento más rápido en este caso, debido a que como los datos ya se encuentran ordenados la partición entre los subarreglos es más rápida de ordenar al no tener que mover algún dato.

**E. Caso descendente**

|  |  |  |  |
| --- | --- | --- | --- |
| Tamaño muestra | Shellsort (mseg) | Mergesort (mseg) | Quicksort (mseg) |
| 30000 | 85 | 52 | 59 |
| 60000 | 216 | 141 | 124 |
| 90000 | 373 | 177 | 188 |
| 120000 | 607 | 370 | 274 |
| 150000 | 915 | 333 | 359 |
| 180000 | 831 | 422 | 419 |
| 210000 | 1132 | 508 | 576 |
| 240000 | 1549 | 672 | 627 |

**Conclusiones:** Se puede observar que el Mergesort y el Quicksort son más rápidos a diferencia del Shellsort. Aunque en este caso el Quicksort es método de ordenamiento más rápido, debido a que sus particiones de los arreglos no son intervalos fijos permitiéndole reacomodar más rápido los elementos.

La conclusión general de todos los casos es que el Mergesort y el Quicksort son los métodos de ordenamiento más rápidos ya que en ocasiones uno le gana al otro o viceversa; el último de los tres es el Shellsort.

# Referencias

Wadler, M. N. (2007). *Java Generics and Collections.* O’Reilly Media, Inc.

Wayne, R. S. ( 2011 ). *Algorithms.* Boston: Pearson Education, Inc. .