UNSA

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

ANALISIS Y DISEÑO DE ALGORITMOS

INSERT VS QUICK VS MERGESORT

NOMBRES:

HUANCA PARQUI ELIZABETH YASMIN

20190748

PROFESOR:

CARLOS EDUARDO ATENCIO TORRES

2021

ÍNDICE

1. INTRODUCCION

En computación y matemáticas un algoritmo de ordenamiento es un algoritmo que pone elementos de una lista o un vector en una secuencia dada por una relación de orden, es decir, el resultado de salida ha de ser una permutación o reordenamiento de la entrada que satisfaga la relación de orden dada. Las relaciones de orden más usadas son el orden numérico y el orden lexicográfico. Ordenamientos eficientes son importantes para optimizar el uso de otros algoritmos (como los de búsqueda y fusión) que requieren listas ordenadas para una ejecución rápida. También es útil para poner datos en forma canónica y para generar resultados legibles por humanos.

2. DESARROLLO

	INSERT	QUICK	MERGESORT
Conceptos	Al igual que el método de la burbuja va comparando los elementos pero se diferencia que no es necesario recorrer el vector completamente si el elemento es menor se compara inmediatamente con su antecesor.	Consiste simplemente en dividir el vector tomando un pivote como un índice con lo cual se van tomando los elementos comparando con respecto al pivote los cuales los menores lo va dejando a su izquierda así ordenando el vector.	Este método consiste en dividir el vector en varias partes las cuales los compara y coloca los menores a la izquierda y luego todas esas partes los mezcla lo cual compara los elementos colocando siempre el menor a la izquierda
Ventajas	Este algoritmo exhibe un buen rendimiento cuando se trabaja con una lista pequeña de datos.	Este algoritmo es capaz de tratar con una enorme lista de elementos.	Este algoritmo es efectivo para conjuntos de datos que se puedan acceder secuencialmente como arreglos, vectores y listas.
Desventajas	No funciona bien con una lista grande.	Consume muchos recursos para su ejecución.	Requiere un espacio adicional de memoria.

1.En caso de arreglos semiordenados, ¿quién ganará? Haga pruebas para Insert vs Quick vs Mergesort

INSERTSORT

```
Consola de depuración de Microsoft Visual Studio
SEMIORDENADO
 1 2 3 4 5 6 7 8 9 10
Execution Time-> 0.004
```

QUICKSORT

```
Consola de depuración de Microsoft Visual Studio
SEMIORDENADO
 1 2 3 4 5 6 7 8 9 10
Execution Time-> 0.006
```

MERGESORT

```
Consola de depuración de Microsoft Visual Studio
SEMIORDENADO
 1 2 3 4 5 6 7 8 9 10
Execution Time-> 0.006
```

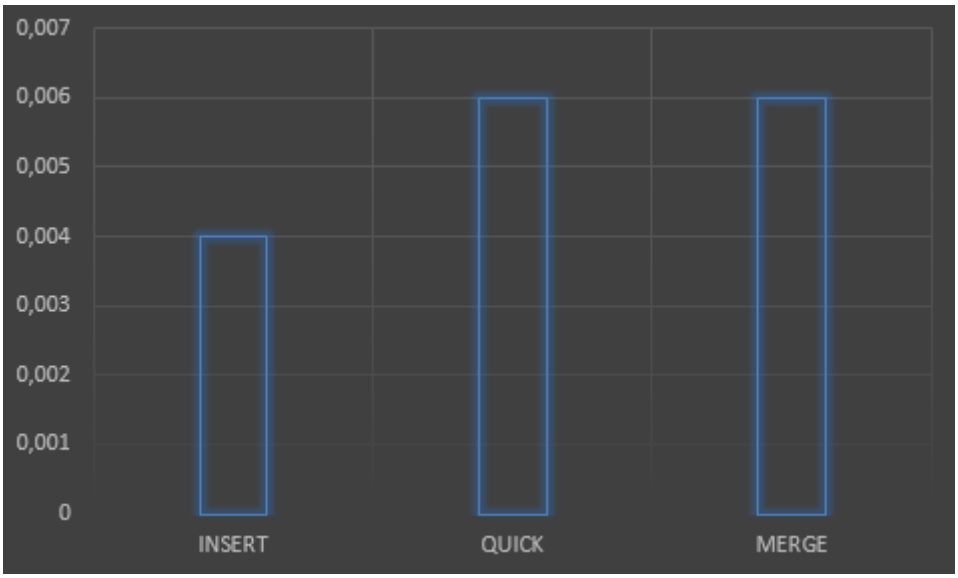

Figura 1.1-Prueba con arreglos semiordenados entre Insert, Quick y Merge

Para un arreglo de 10 elementos de prueba vemos en la figura 1.1 que el algoritmo de ordenamiento por inserción (InsertSort) le toma menos tiempo en ordenar nuestro arreglo, por ende seria el ganador.

2. Compare Quicksort vs Mergesort. Ejecute una sola prueba con arreglos crecientes y haga una prueba más exhaustiva con arreglos aleatorios. ¿Por qué uno siempre gana en un cierto tipo de prueba que otro?

QUICKSORT

```
Consola de depuración de Microsoft Visual Studio
CRECIENTE
 1 2 3 4 5 6 7 8 9 10
Execution Time-> 0.008
```

MERGESORT

```
Consola de depuración de Microsoft Visual Studio
CRECIENTE
 1 2 3 4 5 6 7 8 9 10
Execution Time-> 0.005
```

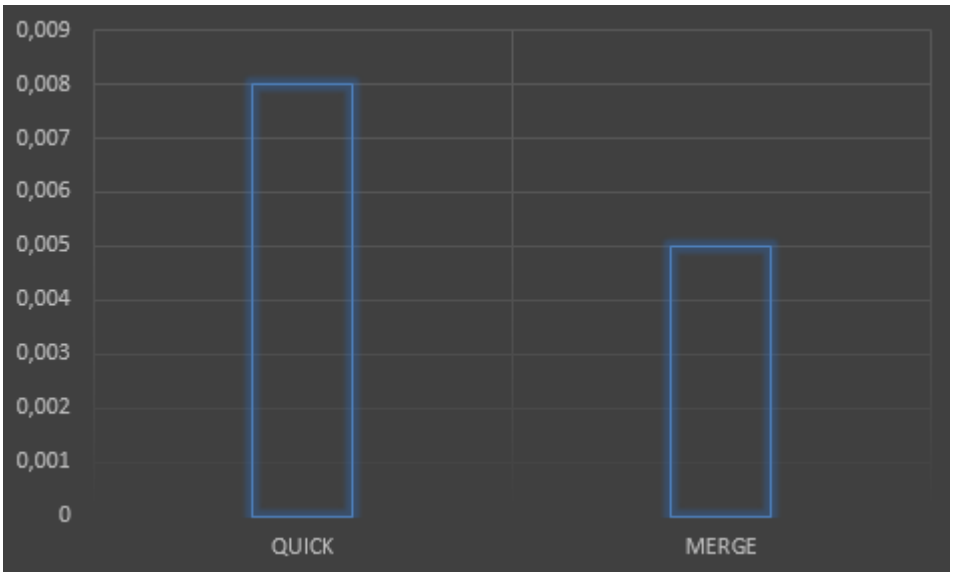

Figura 2.1-Prueba con arreglos crecientes entre Quick y Merge

En esta prueba de 10 Arreglos ingresados de forma creciente vemos en las figura 2.1 que Merge Sort gana a QuickSort .

Elementos	QUICK	MERGE
10	0,001	0,005
20	0,006	0,005
30	0,006	0,007
40	0,013	0,005
50	0,01	0,017
60	0,016	0,006
70	0,012	0,01
80	0,018	0,01
90	0,027	0,014
100	0,035	0,022
500	0,164	0,103
1000	0,21	0,16

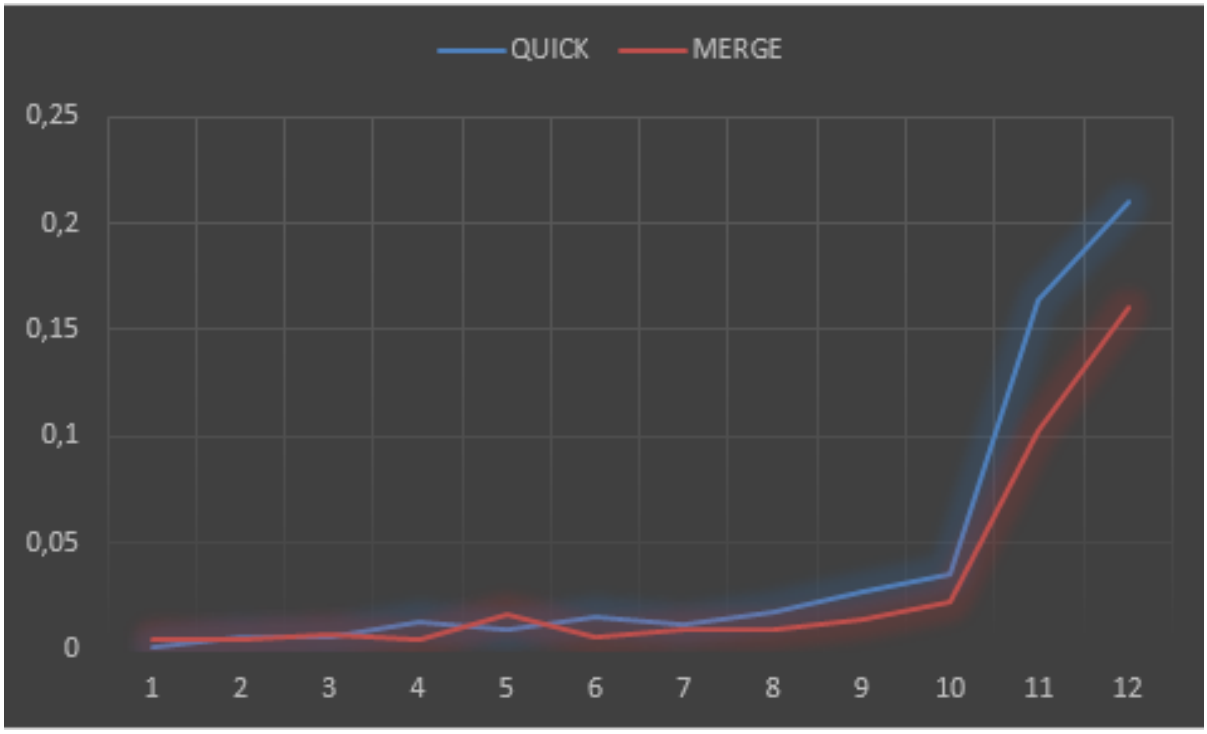

Figura 2.2-Prueba exhaustiva entre Quick y Merge

Sabiendo que ambos tienen la misma clase de complejidad, eso no significa precisamente que ambos tengan el mismo tiempo de ejecución. Quicksort suele ser más rápido que mergesort, porque es más fácil codificar las operaciones.. Pero en la gráfica 2.2 se ve que no se da en todos los casos, para elementos aleatorios y de muchos elementos No debemos usar QuickSort, usaremos MergeSort.

Quicksort es solo O (n log n) en promedio y en el peor de los caso es O (n ^ 2)! Y Mergesort es siempre O (n log n).

2. CONCLUSIÓN

Tomando en cuenta la cantidad de elementos de cada arreglo llegamos a concluir que Insert Sort será el mejor para una pequeña cantidad de elementos, en cambio para las pruebas de MergeSort y QuickSort con elementos aleatorios y variando el tamaño de elementos deberemos utilizar MergeSort por conveniencia.