Laboratorio Nro. 2
Complejidad de algoritmos

Julián Gómez Benítez
Universidad Eafit
Medellín, Colombia
jgomezb11@eafit.edu.co

Juan Pablo Rincón Usma
Universidad Eafit
Medellín, Colombia
jprinconu@eafit.edu.co

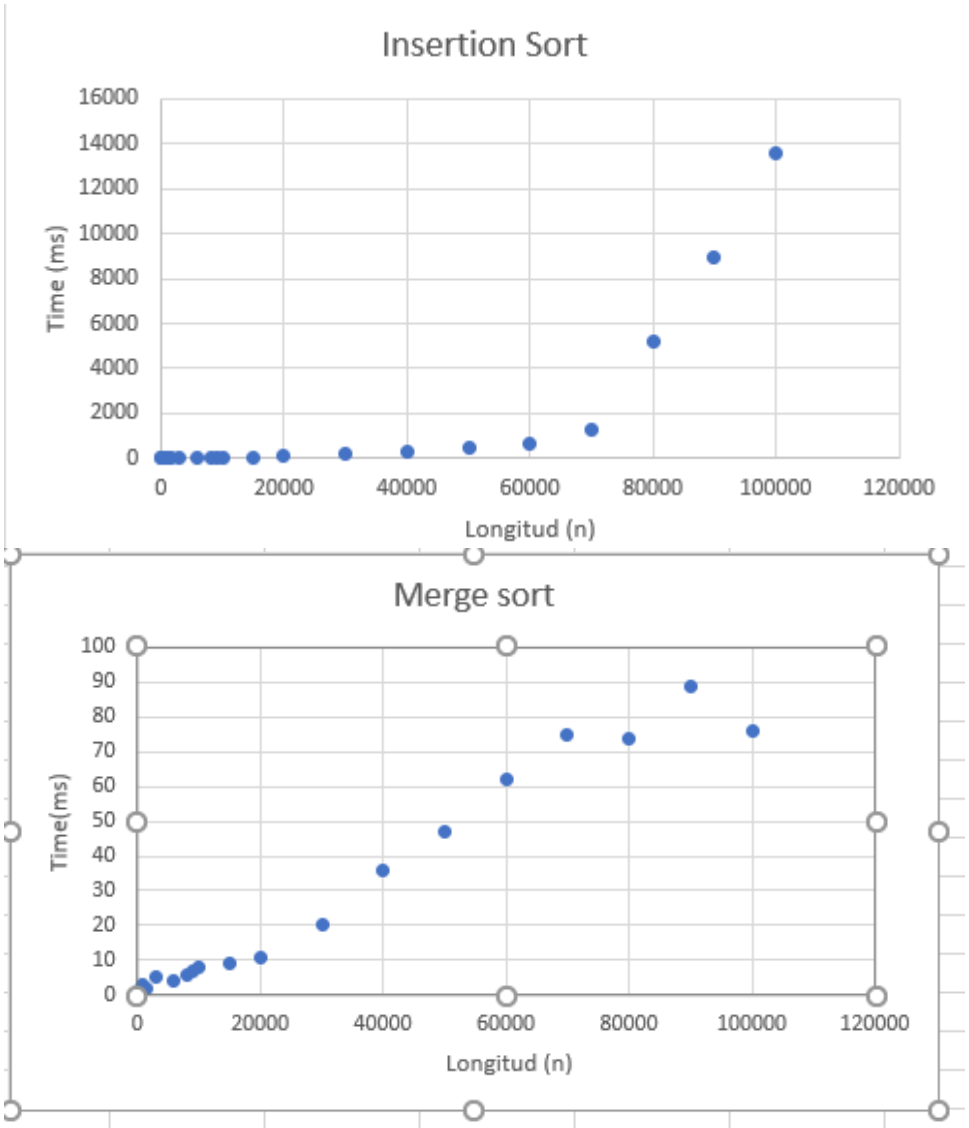
3) Simulacro de preguntas de sustentación de Proyectos

3.1

Insertion Sort			Merge	
Time(ms)	Longitud		Time(ms)	Longitud
0	10		0	10
1	100		0	100
3	500		2	500
16	1000		3	1000
9	1500		2	1500
28	3000		5	3000
17	6000		4	6000
23	8000		6	8000
28	9000		7	9000
47	10000		8	10000
59	15000		9	15000
115	20000		11	20000
200	30000		20	30000
259	40000		36	40000
466	50000		47	50000
633	60000		62	60000
1284	70000		75	70000
5162	80000		74	80000
8909	90000		89	90000
13561	100000		76	100000

3.2

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

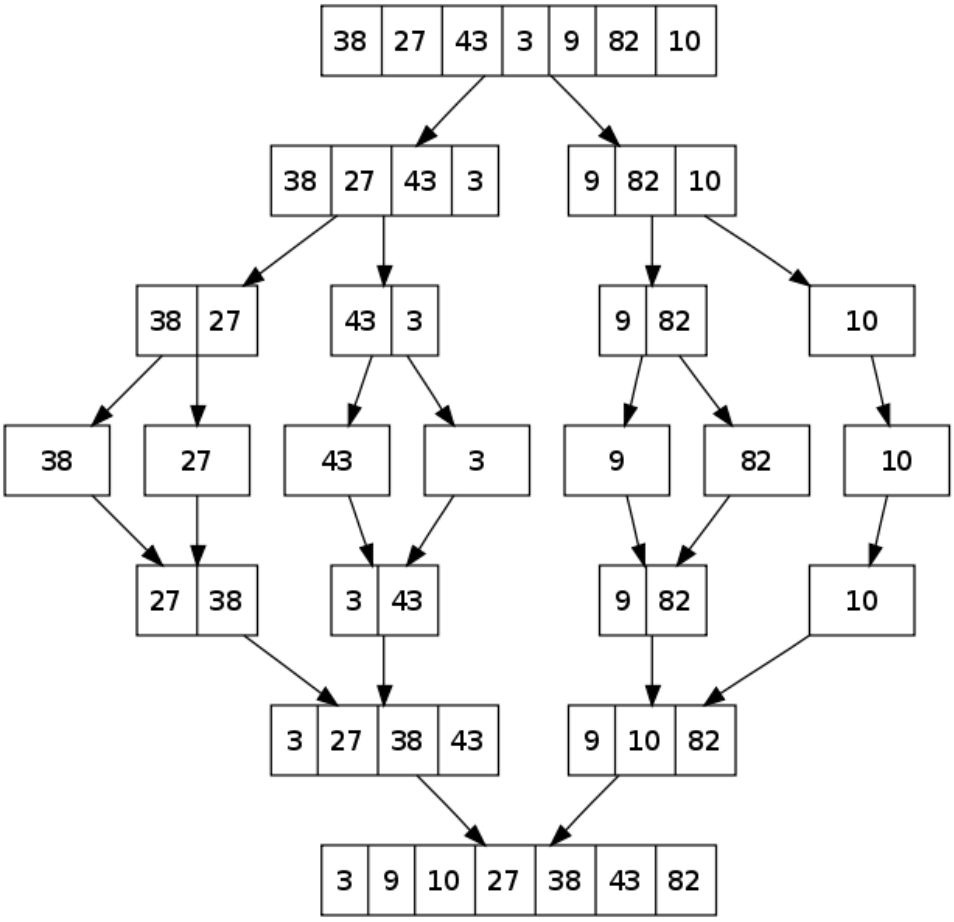


3.3 Puesto que la complejidad en el tiempo del Insertion sort es de $O(n^2)$, es demasiado ineficiente, especialmente para números n que sean muy grandes. Es por esto que para un videojuego que requiere ordenar millones de n y una respuesta en tiempo real, no es apropiado este algoritmo. La mejor opción sería el Merge sort.

3.4 El log aparece en la complejidad del algoritmo conocido como Merge sort (O(n*log(n))), este algoritmo es relacionado con la frase divide y conquistarás, es un método para organizar arreglos recursivamente que se basa en dividir el arreglo por la mitad una cantidad de log(n) veces, siendo n el tamaño del arreglo. Luego la n que multiplica el log se refiere al paso en donde los subarreglos se combinan en uno solo de nuevo.
Adjunto imagen para que se entienda mejor.
Sacada de: https://laingenieria.info/questions/3752/por-que-mergesort-o-log-n#:~:text=Eso%20es%20O%20(nlogn)%20para,algoritmo%20de%20clasificaci%C3%B3n%

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

20de%20mezcla.&text=La%20complejidad%20temporal%20de%20la,lineal%20para%20combinar%20dos%20mitades.



3.4 El método maxSpan se encarga de encontrar un intervalo de números dentro de un arreglo, para ello se tiene en cuenta que es inclusivo y que el intervalo se empieza a tener en cuenta desde un número n hasta encontrar el último número k que sea igual a n, cada número que este entre este n y k valdrá 1, y se incluyen los números n y k. Pero recordando que el número con el que se empiece el intervalo debe ser el mismo que lo cierre, un ejemplo de este método es:

Valor apertura, Valor cierre
intervalo intervalo
| |
{1, 3, 6, 2, 1, 5}
La longitud seria de 5 porque los elementos de intervalo son: {1,2,6,2,1}

3.5 Array 2
CountEvens :

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

T(n) = c1 + c2 * n + c3 + c4 * n + c5 * n
T(n) = O(c2 * n + c3) ==> T(n) = O(n)
Donde n es el tamaño del arreglo.

BigDiff :
T(n) = c1 + c2 + c3 + c4 + c4 * n + c5 + c6 * n^2 + c7 * n + c8 * n^2 + c9 * n^2 + c10 * n^2 + c11 * n^2 + c12 * n^2 + c13 * n^2
T(n) = O(c6 * n^2 + c7 * n) ==> T(n) = O(n^2)
/Donde n es el tamaño del arreglo.

Lucky13 :
T(n) = c1 + c2+ c3 * n + c4 + c5 * n + c6 * n + c7 * n + c8 * n
T(n) = O(c3 * n + c4) ==> T(n) = O(n)
Donde n es el tamaño del arreglo.

Has77 :
T(n) = c1 * n + c2 + c3 * n + c4 * n + c5 * n
T(n) = O(c1 * n + c2) ==> T(n) = O(n)
Donde n es el tamaño del arreglo.

Has12 :
T(n) = c1 + c2 + c3 * n + c4 + c5 * n + c6 * n + c7 * n + c8 * n
T(n) = O(c3 * n + c4) ==> T(n) = O(n)
Donde n es el tamaño del arreglo.

Array 3

MaxSpan :
T(n) = c1 + c2 + c3 * n + c4 + c5 * n + c6 n^2 + c7 * n + c8 * n^2 + c9 * n^2 + c10 * n^2 + c11 * n^2
T(n) = O(c6 n^2 + c7 * n) ==> T(n) = O(n^2)
Donde n es el tamaño del arreglo.

Fix34 :
T(n) = c1 + c2 + c3 + c4 + c5 + c6 + c7 * n + c8 + c9 * n + c10 * n + c11 * n + c12 * n + c13 * n + c14 * n + c15 + c16 + c17 * n + c18 + c19 * n + c20 * n + c21 + c22 * n + c23 * n + c24 *n
T(n) = c7 * n + c8 ==> T(n) = O(n)
Donde n es el tamaño del arreglo.

CanBalance :
T(n) = c1 + c2 * n + c3 + c4 * n + c5 * n + c6 * n^2 + c7 * n^2 + c8 * n^2 + c9 * n
T(n) = O(c6 * n^2 + c7) ==> T(n) = O(n^2)
Donde n es el tamaño del arreglo.

LinearIn :
T(n) = c1 + c2 + c3 * n + c4 + c5 * n + c6 * n
T(n) = O(c3 * n + c4) ==> T(n) = O(n)
Donde n es el tamaño del arreglo outer.

SquareUp :
T(n) = c1 + c2 * n c3 + c4 * n^2 + c5 * n + c6 * n^2 + c7 * n^2 + c8 * n^2
T(n) = O(c4 * n^2 + c5 * n) ==> T(n) = O(n^2)
Donde n es el tamaño del arreglo creado por la entrada n y seteado como n * n.

3.6 La n en el numeral anterior es la cantidad de procesos que realiza el algoritmo para poder solucionar el problema, en este caso para la mayoría de los problemas anteriores el valor de

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

la n estaba dado por la longitud de los arreglos de cada método, en caso de la m es muy similar respecto a la n, también es usado para saber cuántos procesos realiza.

4) Simulacro de Parcial

4.1 *c*
4.2 *d*
4.3 *b*
4.4 *b*
4.5 *d*
1.5.2 *a*
4.6 *Se tarda 100 segundos en completar el procesamiento de 10000 datos.*
4.7 *1) Verdadera 2) Verdadera 3) Verdadera 4) Verdadera*
4.8 *a*
4.9 *c*
4.10 *a*
4.11 *c*
4.12 *b*
4.13 *c*
4.14 *b*

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473