ESTRUCTURA DE DATOS 1
Código ST0245

Laboratorio Nro. 2
Complejidad de Algoritmos

Sebastian Velez Galeano
Universidad Eafit
Medellín, Colombia
svelezg4@eafit.edu.co

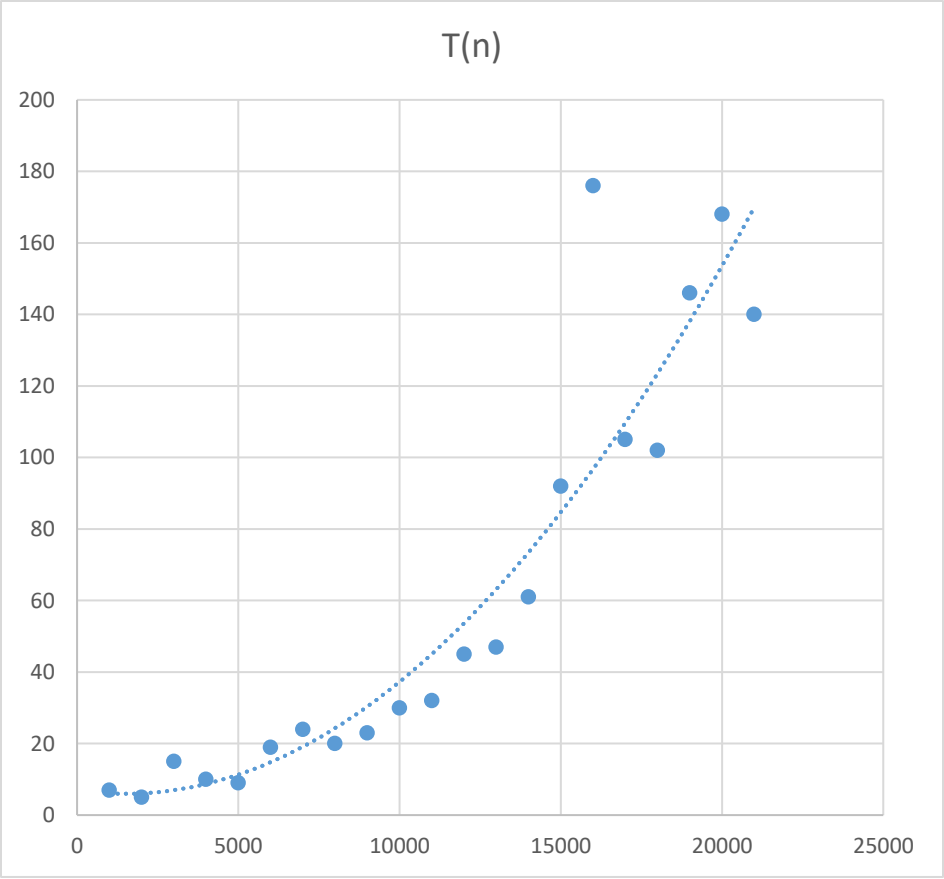
Tomas Atehortua Cefeirno
Universidad Eafit
Medellín, Colombia
tatehortuc@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1 y 3.2

insertionSort(): Complejidad O(n^2)

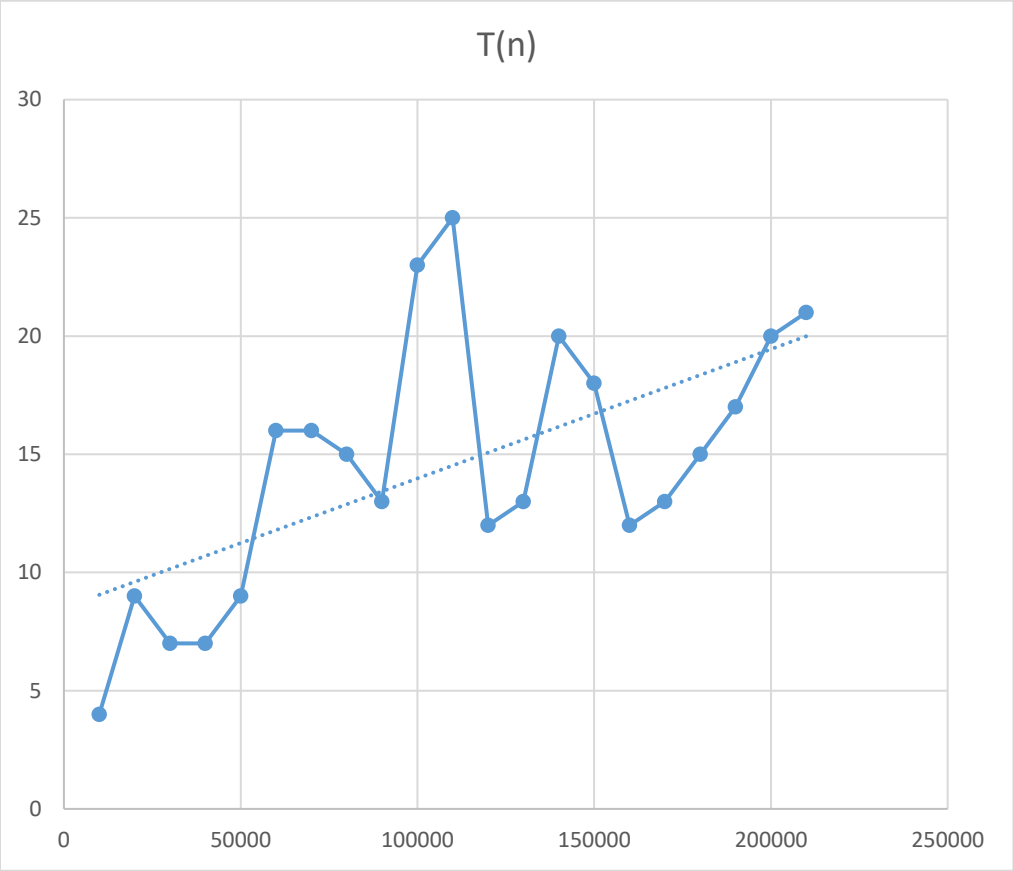
n	T(n)
1000	7
2000	5
3000	15
4000	10
5000	9
6000	19
7000	24
8000	20
9000	23
10000	30
11000	32
12000	45
13000	47
14000	61
15000	92
16000	176
17000	105
18000	102
19000	146
20000	168
21000	140



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

mergeSort(): Complejidad O(n*logn)

n	T(n)
10000	4
20000	9
30000	7
40000	7
50000	9
60000	16
70000	16
80000	15
90000	13
100000	23
110000	25
120000	12
130000	13
140000	20
150000	18
160000	12
170000	13
180000	15
190000	17
200000	20
210000	21



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 Usar insertion sort para organizar millones de datos es una mala idea debido a que su complejidad en tiempo es O(n^2) y tardaría demasiado ordenándolo, en un caso de estos donde se cuente con tantos datos, es mejor usar el merge sort, este ayudaría a organizar esa cantidad de datos en menor cantidad de tiempo.

3.4 Aparece un logaritmo en la complejidad de merge sort ya que va partiendo el n en 2 cada vez que se llama en la recursión. Cuando un algoritmo hace esto su complejidad es O (log n), y como después debe organizarse cada partición del array su complejidad es O (n log n).

Explicación ejerciciosEnLinea:
Array2:
countEvens(): Recorre todo el array y hace cont++ si el número es par.
bigDiff(): Recorre todo el array y busca el Math.max y Math min entre el el array[i] y el máximo o mínimo que tenía antes.
centeredAvarege(): Se ordena los números del arreglo de menor a mayor y solo se imprimen del desde array [0] hasta array[longitud-2].
sum13(): Se recorre el array y en el caso de que se encuentre un 13, hace i+2 y sigue sumando normalmente los demás dígitos del array.
sum67(): Recorre el array, en el caso de que se encuentre un 7 activará un booleano llamado seven, si este booleano es true, significa que no se puede sumar. Cuando en el recorrido del array se encuentre un 7 se pondrá el booleano en falso, para que pueda sumar.

3.7 Array2:
3.7.1 countEvens() : Complejidad O(n)
3.7.2 bigDiff() : Complejidad O(n)
3.7.3 centeredAvarage: Complejidad O(n^2)
3.7.4 sum13: Complejidad O(n)
3.7.5 sum67: Complejidad O(n)
Array3:
3.7.6 maxSpan(): Complejidad O(n)
3.7.7 fix34(): Complejidad O(n^2)
3.7.8 fix45(): Complejidad O(n^2)
3.7.9 canBalance(): Complejidad O(n)
3.7.10 seriesUp(): Complejidad O(n)

3.8 Array2:
3.8.1 countEvens(): n es el tamaño del array.
3.8.2 bigDiff(): n es el tamaño del array.
3.8.3 centeredAvarage: n es el tamaño del array.
3.8.4 sum13: n es el tamaño del array.
3.8.5 sum67: n es el tamaño del array.
Array3:
3.8.6 maxSpan():n es el tamaño del array.
3.8.7 fix34():n es el tamaño del array.

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

3.8.8 fix45():n es el tamaño del array.
3.8.9 canBalance():n es el tamaño del array.
3.8.10 seriesUp():n es el tamaño del array.

4) Simulacro de Parcial

4.1 *c*
4.2 *b*
4.5.a d
4 .5.b a
4.6 T(10000) sería = 100, y como 1 = 1000, entonces 100= 100000
4.7 1,2,4
4.9 a
4.14 a

5) Lectura recomendada (opcional)

Mapa conceptual

6) Trabajo en Equipo y Progreso Gradual (Opcional)

6.1 *Actas de reunión*
6.2 *El reporte de cambios en el código*
6.3 *El reporte de cambios del informe de laboratorio*

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