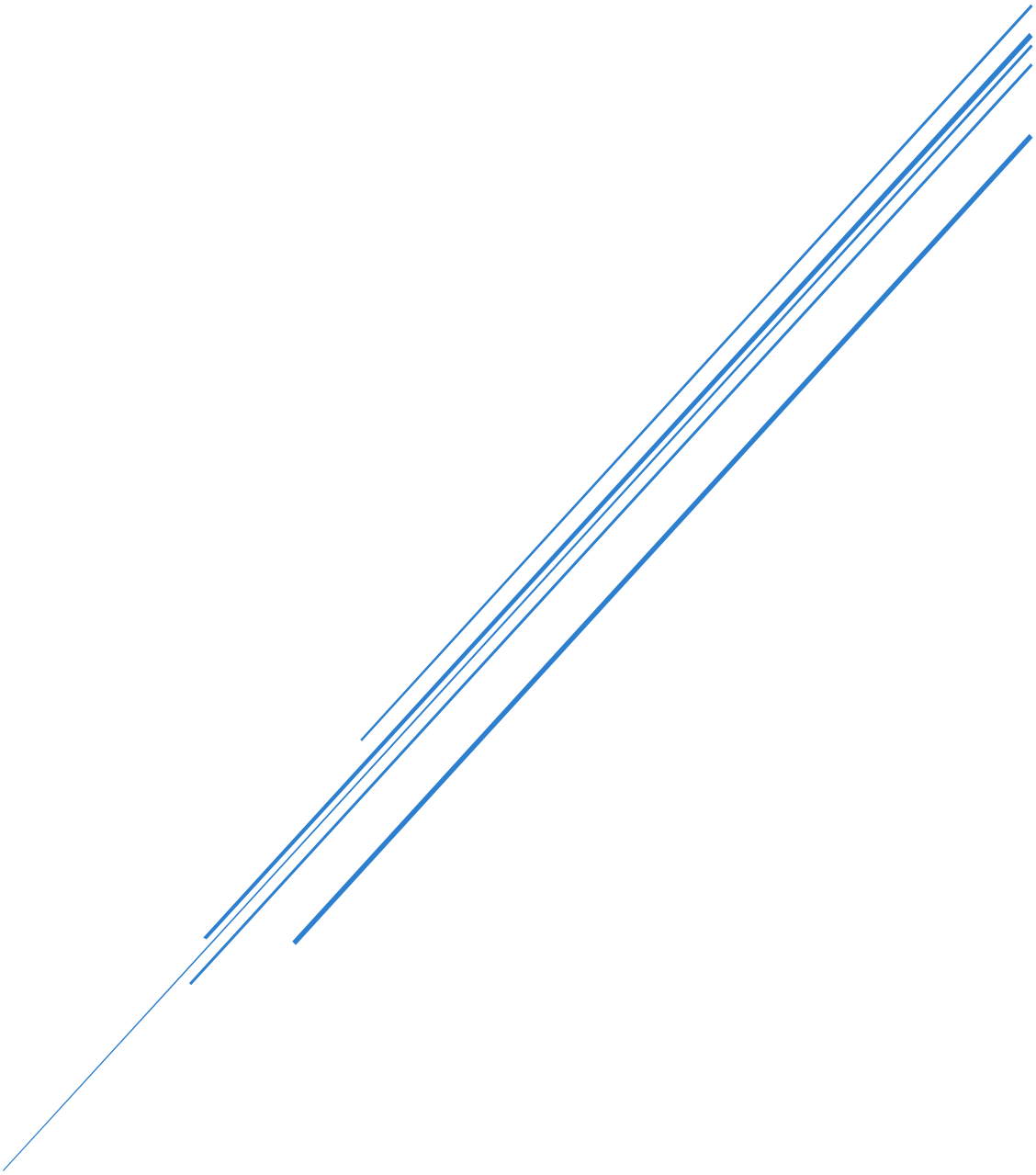
SESIÓN 1.2

Algoritmia-Prácticas de Laboratorio



Jorge Blanco Sánchez
UO293697

Contenido

Jorge Blanco Sánchez
UO293697

MODELOS ITERATIVOS

En esta primera parte de la sesión se dedica al análisis de 4 clases java para obtener su complejidad y sus tiempos de ejecución:

n	t bucle 1	t bucle 2	t bucle 3	t bucle 4
100	0,011	0,505	1,275	1,36
200	0,023	1,800	5,525	9,92
400	0,050	8,190	23,720	80
800	0,116	31,795	109,000	601
1600	0,247	124,695	449,000	4798
3200	0,526	561	1943,000	38010
6400	1,119	2197	8242,000	FdT
12800	2,541	9853	35062,000	FdT
25600	5,344	44132	FdT	FdT
51200	11,240	FdT	FdT	FdT
Complejidad:	O(n log2)	O(n^2 log n)	O(n^2 log n)	O(n^3)

Los tiempos pertenecen a las clases bulce1, bucle2, bucle3 y bucle4, respectivamente

Como podemos observar los tiempos van acorde a la complejidad del algoritmo, por ejemplo, para el caso de bucle 1 y siguiendo la siguiente fórmula $T_{resultado} = \frac{O(k*n \log k*n)}{O(n \log n)} * T_n$, esperamos para el valor 200 un resultado de 0,025 ms, lo que es un valor muy aproximado al resultado obtenido

MODELOS ITERATIVOS CON UNA COMPLEJIDAD DADA

En esta sección se crearan tres clases en java con diferentes complejidades y se analizará como los tiempos depende de estos:

n	t bucle 5	t bulce 6	t bucle 7
100	6,270	91,000	980,000
200	34,145	826,000	15622,000
400	172,000	7197,000	FdT
800	823,000	63068,000	FdT
1600	3891,000	FdT	FdT
3200	18309,000	FdT	FdT
6400	FdT	FdT	FdT
12800	FdT	FdT	FdT
25600	FdT	FdT	FdT
51200	FdT	FdT	FdT
Complejidad:	O(n^2 (log^2)(n))	O(n^3log n)	O(n^4)

Siguiendo la formula del anterior apartado y para el caso de bucle6, se espera un tiempo para n=200 de 830 ms, con los que podemos afirmar que los tiempos están acorde a sus complejidades

Jorge Blanco Sánchez
UO293697

RELACION TEMPORAL DE ALGORITMOS

En esta última parte de la sesión, analizaremos como los algoritmos, los cuales resuelven el mismo problema, son más rápidos que el resto

DOS ALGORITMOS DE DIFERENTE COMPLEJIDAD

Se empezara analizando dos algoritmos que poseen distinta complejidad:

n	t bucle 1	t bucle 2	t1/t2
100	0,0113	0,5050	0,0223
200	0,0230	1,8000	0,0128
400	0,0496	8,1900	0,0061
800	0,1161	31,7950	0,0037
1600	0,2467	124,6950	0,0020
3200	0,5257	561,0000	0,0009
6400	1,1194	2197,0000	0,0005
12800	2,5411	9853,0000	0,0003
25600	5,3439	44132,0000	0,0001
51200	11,2399	FdT	
Complejidad:	O(n log2)	O(n^2 log n)	

Gracias a estos tiempos podemos concluir que el algoritmo de bulce1 es más rápido cuanto mayor sea el tamaño, esto es debido a la complejidad, ya que bucle1 aumenta de forma casi línea, mientras que bucle2, aumenta de forma casi cuadrática

DOS ALGORITMOS CON LA MISMA COMPLEJIDAD

n	t bucle 2	t bucle 3	t3/t2
100	0,505	1,275	2,52475248
200	1,8	5,525	3,06944444
400	8,19	23,720	2,8962149
800	31,795	109	3,42821198
1600	124,695	449	3,60078592
3200	561	1943	3,46345811
6400	2197	8242	3,75147929
12800	9853	35062	3,5585101
25600	44132	FdT	
51200	FdT	FdT	
Complejidad:	O(n^2 log n)	O(n^2 log n)	

Como podemos observar, el algoritmo 2 es más rápido que el algoritmo 3, esto se debe a constantes que no se representan cuando se da la complejidad, es decir en el caso del algoritmo 2, su complejidad real seria de O((n^2)/2 * log3 (n)), y dichas constantes provocan esas variaciones de tiempo

Jorge Blanco Sánchez
UO293697

MISMO ALGORITMO EN ENTORNOS DE DESARROLLO y/o EJECUCIÓN DIFERENTES

Por último, analizaremos como el entorno afecta al tiempo de ejecución:

n	Bucle4 Python t41	Bucle4 Java (Sin_opti) t42	Bucle4 Java (Con_opti) t43	t42/t41	t43/t42
200	59	1,36	0,077	0,023	0,057
400	422	9,92	0,297	0,024	0,030
800	3603	80	1,35	0,022	0,017
1600	26917	601	25,472	0,022	0,042
3200	FdT	4798	109,442		0,023
6400	FdT	38010	476		0,013
Complejidad:	O(n^3)	O(n^3)	O(n^3)		

Gracias a estos tiempos podemos concluir que el entorno afecta al tiempo de ejecución, dándonos tiempos de aproximadamente 42 veces más rápido Java sin optimizar que Python y tiempos de 33 veces más rápido Java optimizado que Java sin Optimizar