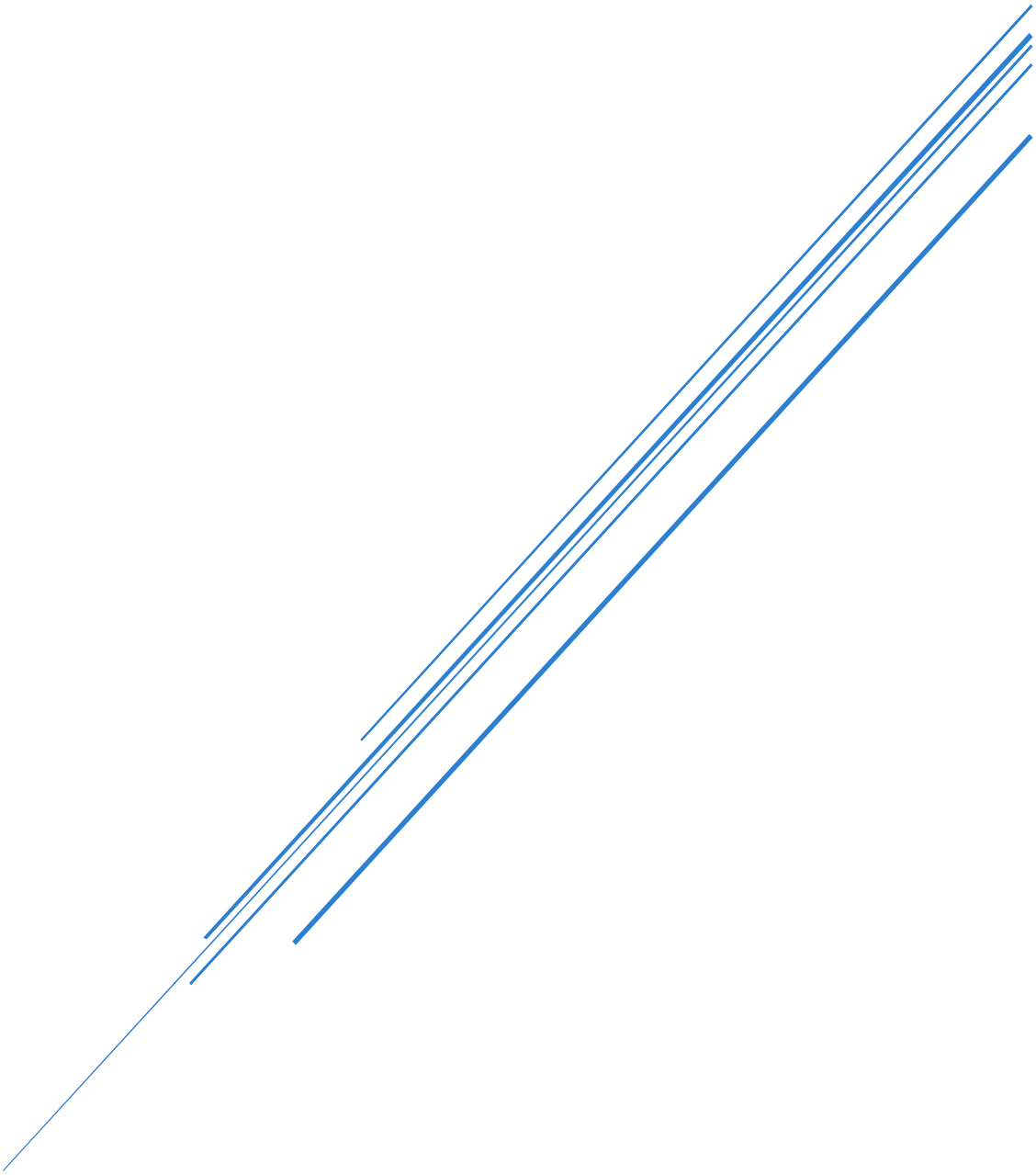
PRÁCTICA 3

Laboratorio de Algoritmia



Jorge Blanco Sánchez
Uo293697

Contenido

Jorge Blanco Sánchez
Uo293697

Divide y Vencerás por SUSTRACCIÓN

La clase sustraccion1 y sustraccion2 consiguen realizar las operaciones hasta los 8000 elementos, cuando se avanza al siguiente valor, 16000 el programa al realizar llamadas recursiva con n-1 elementos, se consigue que se generen 16000 llamadas, haciendo que la pila, donde se almacenan estas llamadas, se desborde ya que se tarda en llegar a la parada.

La clase sustraccion3 teniendo una complejidad de 2^n y tardaría aproximadamente unos 2,47E+51 años en procesar 80 elementos

Sustraccion4 la cual posee una complejidad de n^3, nos ha permitido medir tiempos y obtener los siguientes resultados

n	T(ms)
100	FdF
200	FdF
400	335
800	2628
1600	20985
3200	FDT

Midiendo Sustracción 5, el cual posee una complejidad de 3^(n/2), hemos obtenido los siguientes tiempos:

n	T(ms)
30	674
32	2007
34	5847
36	17526
38	52605

Si n fuera igual a 80 sustraccion5 tardaría aproximadamente unos 18 108 años

Jorge Blanco Sánchez
Uo293697

Divide y Vencerás por DIVISIÓN

Tras analizar las complejidades de division1(O(n)), division2(O(n log n)) y divison3(O(n)), podemos apreciar como los tiempos obtenidos son aproximadamente el doble respecto al n anterior, ya que se está duplicando la n con cada llamada, con esto podemos confirmar que siguen la complejidad deducida

Division4 con O(n^2) con($a<b^k$)

n	T(ms)
1000	FdF
2000	90
4000	349
8000	1396
16000	5626
32000	22778

Division5 con O(n^2) con ($a>b^k$)

n	T(ms)
1000	51
2000	198
4000	779
8000	3194
16000	12540
32000	50169

Como podemos ver ambos algoritmo peses a tener diferente estructura con los valores a, b y k, se consigue una complejidad cuadrática, la cual se confirma con los tiempos dados

Diferentes Implementaciones de Algoritmos

Para esta parte se usaran algoritmos de suma de vector y Fibonacci

Para Vector suma a un vector de 100 elementos

	T(nanosegundos)
Iterativo	960
Sustracción	2321
División	4776

Como podemos ver, pese a tener la misma complejidad, es mucho mas eficiente resolver el problema de forma iterativa.

Para Fibonacci con el valor 40

	T(nanosegundos)
Iterativo 1	353
Iterativo 2	502
Sustracción	713
Sustracción 2	5200

Para los 3 primeros algoritmos, la complejidad es de O(n), mientras que para el último es de O(1,6^n), la mejor solución para este problema es hacerlo de forma iterativa.

Con estos dos ejemplos, podemos concluir, que a veces la mejor manera de resolver un problema es usar un algoritmo iterativo y no de divide y vencerás

Jorge Blanco Sánchez
Uo293697

Algoritmo de Ordenación por Mezcla

Tras completar este algoritmo y medir tiempos, se han obtenido los siguientes resultados:

n	ordenado	inverso	aleatorio
31250	FdF	FdF	FdF
62500	FdF	FdF	50
125000	81	84	101
250000	160	174	200
500000	333	361	424
1000000	686	750	879
2000000	1423	1483	1816
4000000	3054	3086	3762
8000000	6166	6454	7780
16000000	12677	13483	16062
32000000	26185	27680	33190
64000000	53814	56743	68539

La unidad de tiempo es milisegundos.

Y si lo comparamos con el algoritmo de ordenación Rápida

n	aleatorio(Mezcla)	aleatorio(Rápido)	Mezcla/Rápido
31250	FdF	FdF	
62500	50	FdF	
125000	101	FdF	
250000	200	178	1,123595506
500000	424	355	1,194366197
1000000	879	717	1,225941423
2000000	1816	1616	1,123762376
4000000	3762	3345	1,124663677
8000000	7780	6801	1,143949419
16000000	16062	14230	1,128742094
32000000	33190	28926	1,147410634
64000000	68539	59614	1,149713155

La unidad de tiempo es milisegundos.

Como podemos ver el algoritmo de ordenación Rápido es, aproximadamente, 1,15 veces de media, más rápido que el algoritmo de Mezcla