Algoritmia

ALICIA FERNÁNDEZ PUSHKINA

DyV DIVISIÓN

$a < b^k \longrightarrow O(n^k)$
$a = b^k \longrightarrow O(n^k \cdot \log n)$
$a > b^k \longrightarrow O(n^{\log_b a})$

- $a=2, b=2, k=1 \rightarrow O(n \cdot \log n)$

DyV SUBSTRACCIÓN

$a < 1 \longrightarrow O(n^k)$
$a = 1 \longrightarrow O(n^{k+1})$
$a > 1 \longrightarrow O(a^{n/b})$

"Un logaritmo es a que número tiene que estar elevada la base para que te de el número que te dice"

Ej: $\log_2 1024 = 10$ $\log_3 81 = 4$

PARALELIZACION

0) extends RecursiveAction

1) import Java.Util.Concurrent.ForkJoinPool
   " " " " .RecursiveAction
2) eliminar "static"
3) crear variables de instancia para los índices
4) crear tantas instancias como llamadas recursivas tenga
5) invokeAll (instancia1, instancia2, ...)
6) crear constructor (vector e índices)
7) crear instancia inicial
8) crear pool [ForkJoinPool pool = new ForkJoinPool ()]
9) invocar pool.invoke (instInicial)

4)–5): el método recursivo se llama ahor compute()

7)–9): En el Main, antes de imprimir el final

NOTA: Si es algo como "return recDiv(x) + recDiv(y)" tenemos que:
- crear las 2 instancias
- invocarlas
- juntarlas en la variable

QUICKSHORT

mejor → $O(n \cdot \log n)$
peor → $O(n^2)$
medio → $O(n \cdot \log n)$

buen pivote y n grande
- mej → $O(n \cdot \log n)$
- pe: → $O(n^2)$ o $O(n \cdot \log n)$
- med → $O(n \cdot \log n)$

mal pivote y n grande
- mej → $O(n^2)$ o $O(n \cdot \log n)$
- pe → $O(n^2)$
- med → $O(n^2)$

- lista ordenada y pivote (primer o último) elemento → $O(n^2)$. Crea sublistas tamaño (n-1)
- lista ordenado y pivote central → $O(n \cdot \log n)$. Arbol perfecto. Dos partes iguales de tamaño

INSERCIÓN DIRECTA

mejor → $O(n)$. Lista ordenada
peor → $O(n^2)$
medio → $O(n^2)$

Ejemplo:

8	5	1	6	9	2	7	8	4
3	5	1	6	9	2	7	8	4
1	3	5	6	9	2	7	8	4
1	3	5	6	9	2	7	8	4
1	3	5	6	9	2	7	8	4
1	2	3	5	6	9	7	8	4
1	2	3	5	6	7	9	8	4
1	2	3	5	6	7	8	9	4
1	2	3	4	5	6	7	8	9

SELECCIÓN

mejor, peor, medio → $O(n^2)$

TRAZA sobre n=8 → 9 (1) 8 3 5 4 7 2

1	9	8	3	5	4	7	2
1	2	8	3	5	4	7	9
1	2	3	8	5	4	7	9
1	2	3	4	5	8	7	9
1	2	3	4	5	7	8	9
1	2	3	4	5	7	8	9
1	2	3	4	5	7	8	9

1º Buscamos el elemento más pequeño y lo intercambiamos con la 1ª posición
2º Buscamos el más pequeño de los restantes y lo intercambiamos por la 2ª pos
...
Hasta terminar

BURBUJA

mejor, peor, medio → $O(n^2)$

1) Comparamos cada elemento con el siguiente e intercambiamos si están desordenados
2) Repetimos excepto con la última posición, ya ordenada.

MEZCLA

Mergeshort

mejor, peor, medio → $O(n \cdot \log n)$

Vs QUICKSORT

+ ambos algoritmos DyV
+ "misma" complejidad

- Mezcla = alg. externo (memoria extra)
- Quicksort = alg. interno (no memoria extra)
- Quicksort puede llegar a $O(n^2)$