

ordenación



20 de febrero de 2018

Gema rico pozas

UO238096

TRABAJO PEDIDO

Vamos a comparar diversos algoritmos de ordenación. Para ello programamos lo necesario para rellenar 4 tablas de tiempos (para Burbuja , Quicksort con pivote utilizando mediana a tres , Selección e Inserción).

BURBUJA

|  |  |  |  |
| --- | --- | --- | --- |
| Carga de trabajo (n) | tiempo "ordenado" | tiempo "inverso" | tiempo "aleatorio" |
| 10000 | 17 | 48 | 86 |
| 20000 | 53 | 179 | 401 |
| 40000 | 224 | 810 | 1718 |
| 80000 | 855 | 2876 | 6988 |
| 160000 | 3454 | 11533 | 28224 |
| 320000 | 13915 | 46239 | 112660 |
| 640000 | 55810 | 184222 | 449609 |
| 1280000 | 226002 | 737584 | 1800385 |

Caso mejor: O(n2) 🡪 ordenado

Caso peor: O(n2) 🡪aleatorio

Caso medio: O(n2) 🡪 inverso

El peor de los algoritmos aquí estudiados, ya que tiene que iterar por todo el array, por tanto presenta complejidad O(n2).

QUICKSORT (RAPIDO MEDIANA 3)

|  |  |  |  |
| --- | --- | --- | --- |
| Carga de trabajo (n) | tiempo "ordenado" | tiempo "inverso" | tiempo "aleatorio" |
| 10000 | 69 | 69 | 85 |
| 20000 | 117 | 138 | 134 |
| 40000 | 248 | 268 | 282 |
| 80000 | 512 | 553 | 589 |
| 160000 | 1071 | 1160 | 1256 |
| 320000 | 2207 | 2400 | 2682 |
| 640000 | 4610 | 5027 | 5916 |
| 1280000 | 9535 | 10512 | 13278 |

Caso mejor: O(n logn) 🡪 ordenado

Caso peor: O(n2) 🡪aleatorio

Caso medio: O(n logn) 🡪 inverso

El mejor de los algoritmos aquí estudiados. Utilizamos como pivote la mediana de una muestra de 3 elementos lo que evita que ordenado sea el caso peor como en el caso de utilizar el elemento central como pivote.

SELECCIÓN

|  |  |  |  |
| --- | --- | --- | --- |
| Carga de trabajo (n) | tiempo "ordenado" | tiempo "inverso" | tiempo "aleatorio" |
| 10000 | 14 | 37 | 36 |
| 20000 | 47 | 139 | 135 |
| 40000 | 109 | 423 | 293 |
| 80000 | 762 | 2183 | 2151 |
| 160000 | 3101 | 8838 | 8625 |
| 320000 | 12517 | 35409 | 34351 |
| 640000 | 50060 | 141318 | 137561 |
| 1280000 | 201958 | 568591 | 549692 |

Caso mejor: ordenado 🡪 O(n2)

Caso peor: inverso 🡪 O(n2)

Caso medio: aleatorio 🡪 O(n2)

Todos presentan una complejidad cuadrática, ya que el algoritmo necesita encontrar la posición del más pequeño (O(n)) e intercambiar; es decir tenemos O(n) \* O(n) == O(n2).

INSERCIÓN

|  |  |  |  |
| --- | --- | --- | --- |
| Carga de trabajo (n) | tiempo "ordenado" | tiempo "inverso" | tiempo "aleatorio" |
| 10000 | 1 | 17 | 12 |
| 20000 | 0 | 52 | 27 |
| 40000 | 1 | 268 | 134 |
| 80000 | 0 | 1076 | 542 |
| 160000 | 0 | 4371 | 2163 |
| 320000 | 0 | 17462 | 8733 |
| 640000 | 0 | 70220 | 35060 |
| 1280000 | 0 | 283789 | 142319 |

Caso mejor: ordenado 🡪 O(n) 🡪los elementos ya están ordenados

Caso peor: inverso 🡪 O(n2) 🡪 todos los elementos necesitan ser reordenados.

Caso medio: aleatorio 🡪 O(n2) 🡪no todos necesitan ser reordenados.

CONCLUSIÓN:

Hemos comprobado que realmente los algoritmos con complejidad O(n2) son algoritmos “malos” y que el más eficiente es el algoritmo Quicksort con mediana a 3.