Burbuja (nVeces=10, ordenado), inverso y aleatorio = 1

Inserción (nVeces=105, ordenado), inverso y aleatorio = 1

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(ms) ordenado | T(ms) inverso | T(ms) aleatorio |
| 10 000 | 0,00379 | 69 | 39 |
| 20 000 | 0,00702 | 311 | 141 |
| 40 000 | 0,01445 | 263 | 134 |
| 80 000 | 0,02825 | 1557 | 759 |
| 160 000 | 0,05637 | 5845 | 2733 |
| 320 000 | 0,11638 | 21907 | 10738 |
| 640 000 | 0,23772 | 96419 | 42094 |
| 1 280 000 | 0,5917 | 372840 | 175309 |

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(ms) ordenado | T(ms) inverso | T(ms) aleatorio |
| 10 000 | 19 | 66 | 129 |
| 20 000 | 69,9 | 234 | 527 |
| 40 000 | 278,1 | 1094 | 2428 |
| 80 000 | 1301,1 | 5017 | 10705 |
| 160 000 | 5457 | 18742 | 47783 |
| 320 000 | 21785,9 | 73829 | 188943 |
| 640 000 | 84747,1 | 290886 | 719872 |
| 1 280 000 | 332208,8 | 1157727 | 2879491 |

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(ms) ordenado | T(ms) inverso | T(ms) aleatorio |
| 10 000 | 11,1 | 44 | 59 |
| 20 000 | 48,0 | 168 | 177 |
| 40 000 | 209,9 | 674 | 481 |
| 80 000 | 939,4 | 2408 | 2497 |
| 160 000 | 3893 | 9654 | 9871 |
| 320 000 | 15961,3 | 37757 | 39901 |
| 640 000 | 62089,4 | 151491 | 151491 |
| 1 280 000 | 246494,9 | 596874 | 627239 |

Selección (nVeces=10, ordenado), inverso y aleatorio = 1

QuickSort Mediana (nVeces=103, ordenado), inverso y aleatorio = 1)

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(ms) ordenado | T(ms) inverso | T(ms) aleatorio |
| 10 000 | 0,144 | 3 | 2 |
| 20 000 | 0,293 | 3 | 4 |
| 40 000 | 0,626 | 3 | 3 |
| 80 000 | 1,626 | 2 | 5 |
| 160 000 | 2,904 | 5 | 12 |
| 320 000 | 6,503 | 11 | 26 |
| 640 000 | 13,372 | 23 | 53 |
| 1 280 000 | 28,472 | 51 | 113 |

Gráficas:

Comparación de algoritmos:

Conclusiones:

Después de la implementación y la ejecución de todos los algoritmos, se ve claramente que la mejor eficiencia es la del algoritmo de QuickSort, seguida de los algoritmos de Inserción, Selección y Burbuja, salvo en el caso de que el vector esté ordenado, pues en ese caso es mejor el de Inserción.

**Análisis algoritmo Burbuja:**

Aunque es el algoritmo más conocido, se comprueba que claramente es un mal algoritmo, pues al ser de complejidad cuadrática en todos los casos, es verdaderamente lento. Es el algoritmo más lento de todos, pues sí o si ha de recorrer dos bucles for.

**Análisis algoritmo Selección:**

Se comprueba que no es un algoritmo eficiente, ya que en todos sus casos posee una complejidad cuadrática. Es el segundo algoritmo más lento, por debajo del algoritmo de burbuja.

**Análisis algoritmo Inserción:**

Se comprueba que es un algoritmo lento para todos los casos salvo que el vector pasado como parámetro esté ordenado. En dicho caso se comprueba que su complejidad es lineal, en lugar de cuadrática, siendo mejor incluso que el algoritmo Quicksort.

**Análisis algoritmo Quicksort:**

Es el algoritmo más eficiente de todos, pues su complejidad es logarítmica, salvo en el caso peor, es decir, que el vector esté inversamente ordenado, en el cual tiene una complejidad cuadrática. De todas formas, pese a compartir complejidad con los demás algoritmos en dicho caso, tarda mucho menos que el resto.

**Describe en el documento en que consiste este método de selección, cuándo funciona mal y cuando no y que efecto tiene en el tiempo de ejecución.**

El método rápido fatal funciona mal cuando aumentas el tamaño del problema, pues al tomar como pivote el elemento de la izquierda obtiene una partición izquierda vacía y una partición derecha con el resto del vector. Únicamente funciona “bien” cuando el tamaño del vector es muy reducido. El efecto que tiene en tiempo de ejecución es su elevado tiempo de ejecución, así como la alta probabilidad de desbordamiento de la pila.