Bubble

|  |  |  |  |
| --- | --- | --- | --- |
| n | Ordered | Reverse | Random |
| 10.000 | 473 | 2.179 | 1.440 |
| 20.000 | 1.875 | 8.669 | 5.768 |
| 40.000 | 7.477 | 34.691 | 23.208 |
| 80.000 | 29.799 | 138.387 | 92.914 |
| 160.000 | 117.173 | OT | 370.993 |
| 320.000 | OT | OT | OT |
| 640.000 | OT | OT | OT |

The ordered array will invariably bypass the vector.interchange() method, as it is inherently sorted. Conversely, the reverse sequence will consistently invoke it. As for the random sequence, its usage of vector.interchange() depends on the distribution of random numbers within the array. The more orderly the array, the fewer executions of the method and the quicker the Bubble method's runtime.

Selection

|  |  |  |  |
| --- | --- | --- | --- |
| n | Ordered | Reverse | Random |
| 10.000 | 440 | 494 | 440 |
| 20.000 | 1760 | 1942 | 1740 |
| 40.000 | 7084 | 7750 | 6945 |
| 80.000 | 28364 | 31103 | 27852 |
| 160.000 | 112798 | 123698 | 111494 |
| 320.000 | OT | OT | OT |
| 640.000 | OT | OT | OT |

­­­­­ Insertion

|  |  |  |  |
| --- | --- | --- | --- |
| n | Ordered | Reverse | Random |
| 10.000 | 0,225 | 825 | 309 |
| 20.000 | 0,438 | 2.870 | 1.238 |
| 40.000 | 0,874 | 10.641 | 4.869 |
| 80.000 | 1,729 | 40.773 | 19.488 |
| 160.000 | 3,413 | OT | 77.800 |
| 320.000 | 6,852 | OT | OT |
| 640.000 | 13,678 | OT | OT |
| 128.000 | 27,434 | OT | OT |

Quicksort

|  |  |  |  |
| --- | --- | --- | --- |
| n | Ordered | Reverse | Random |
| 160.000 | 37,84 | 41 | 82 |
| 320.000 | 77,92 | 87 | 172 |
| 640.000 | 160 | 178 | 367 |
| 1.280.000 | 329,30 | 364 | 797 |
| 2.560.000 | 679 | 744 | 1.756 |
| 5.120.000 | 1.386 | 1.528 | 4.017 |
| 10.240.000 | 2.842 | 3.125 | 9.933 |
| 20.480.000 | 5.808 | 6.384 | 27.226 |
| 40.960.000 | 11.854 | 13.075 | 82.390 |

Quicksort + Insertion

|  |  |  |
| --- | --- | --- |
| n | K | Random |
| 16.000.000 | 5 | 19.014 |
| 16.000.000 | 10 | 18.803 |
| 16.000.000 | 20 | 18.745 |
| 16.000.000 | 30 | 18.623 |
| 16.000.000 | 50 | 17.683 |
| 16.000.000 | 100 | 16.387 |
| 16.000.000 | 200 | 14.101 |
| 16.000.000 | 500 | 22.403 |
| 16.000.000 | 1000 | 42.275 |

With smaller values of k, the performance closely resembles that of quicksort. However, as we increment k, the ordered arrays produced by quicksort are swiftly refined by insertion, yielding superior performance within the range of 30 to 200. Nonetheless, once k surpasses 500, performance deteriorates anew, primarily due to the disorderliness of the array provided to the insertion method.