We have 4 different initial array ordering situations: random, ordered, partially ordered and reversed ordered. For each type of the array, we choose 5 different lengths: 500, 1000, 2000, 4000, 8000. For each value of length, we run over 100 times and calculate the mean running time of each experiment. And the result are shown below:

Text

Description automatically generated

\*mean time of 100 time running

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Length=500 | Length=1000 | Length=2000 | Length=4000 | Length=8000 |
| Order | 0.08003963 | 0.06247623 | 0.08255641 | 0.10656276 | 0. 10583833 |
| Random | 0.68606424 | 1.8399524 | 6.65002588 | 25.5124338 | 104.30026096 |
| Reverse | 1.1489723 | 3.80038602 | 12.65826121 | 49.62944727 | 201.28857228 |
| Partially Order | 0.15643158 | 0.53373174 | 1.61946874 | 6.42083422 | 25.52431837 |

From the results shown above, we could clearly find that:

for random array, reversed array, partially ordered array, the running time of insertion sort to sort those arrays are close to O(n^2). For example, for random array, when the length of the array doubling, if the running time is O(N^2), the running time of the longer array is 4 times to the running time of the shorter array.

example: length=2000, running time of random array = 6.65

length=4000, running time of random array is 25.51 close to 6.65\*4=26.6

The same example as the other two types array.

for ordered array, the running time of to sort this array by using insertion sort should take O(n). And the test results also verify this conclusion.

In total, the running time of insertion sort should be O(N^2);

Benchmark.Test

Graphical user interface, text

Description automatically generated

TimerTest:

Text

Description automatically generated

InsertionSortTest:

Text

Description automatically generated