**INFO 6205**

**Program Structures & Algorithms**

**Summer Full 2018**

**Assignment 3**

1. **CONCLUSION:**

Sort efficiency various by method, the size of arrays and the ordering which we input.

* **conclusion1**: when the arrays’ ordering and size are fixed, normally, sort efficiency rank is: shell(3h+1)>shell(2h+1)>shell(2h)>insertion>selection;
* **conclusion2:** when the sort method and arrays’ size are fixed, sort efficiency rank is: ordered>partial>random>reverse.
* **conclusion3:** when the sort method and arrays’ ordering are fixed, sort efficiency is higher when array’s size are smaller.

In this program, I implemented 3 different sort methods (insertion sort, selection sort, shell sort). As for shell sort, it is divided into 3 modes according to the H (gap value). Therefore, there are 5 sorts methods in total.

For each sort method, the arrays are initialized in four different orderings which are random, ordered, partial, reverse ordering. Arrays’ sizes are in 5 various numbers (1000,2000,4000,8000,16000).

Therefore, there are 5\*4\*5=100 sets of data of average time been output.

1. **Tables and Graphs of various sort method.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| INSERTION | random | ordered | partial | reverse |
| 1000 | 3928958.96 | 6518.1 | 1755191.82 | 3897766.4 |
| 2000 | 7465265.54 | 11154.21 | 1.58E+07 | 1.44E+07 |
| 4000 | 2.98E+07 | 25567.16 | 2.67E+07 | 6.03E+07 |
| 8000 | 1.23E+08 | 50812.23 | 1.10E+08 | 2.37E+08 |
| 16000 | 5.10E+08 | 114211.55 | 4.40E+08 | 1.05E+09 |

table1 Insertion Sort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SELECTION | random | ordered | partial | reverse |
| 1000 | 8323714.49 | 4271675.53 | 4251224.23 | 4349155.18 |
| 2000 | 2.19E+07 | 1.70E+07 | 1.58E+07 | 1.71E+07 |
| 4000 | 9.01E+07 | 6.95E+07 | 6.49E+07 | 6.73E+07 |
| 8000 | 3.73E+08 | 2.72E+08 | 2.64E+08 | 2.63E+08 |
| 16000 | 1.52E+09 | 1.08E+09 | 1.05E+09 | 1.24E+09 |

table2 Selection Sort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SHELL (h=2h) | random | ordered | partial | reverse |
| 1000 | 3726318.21 | 7146 | 1869983.84 | 4179060.1 |
| 2000 | 8003805.42 | 13543.48 | 7472163.16 | 1.60E+07 |
| 4000 | 3.09E+07 | 32250.06 | 2.79E+07 | 6.39E+07 |
| 8000 | 1.27E+08 | 77718.2 | 1.14E+08 | 2.42E+08 |
| 16000 | 5.39E+08 | 81803.64 | 4.55E+08 | 9.89E+08 |

table3 Shell Sort (h=2h)

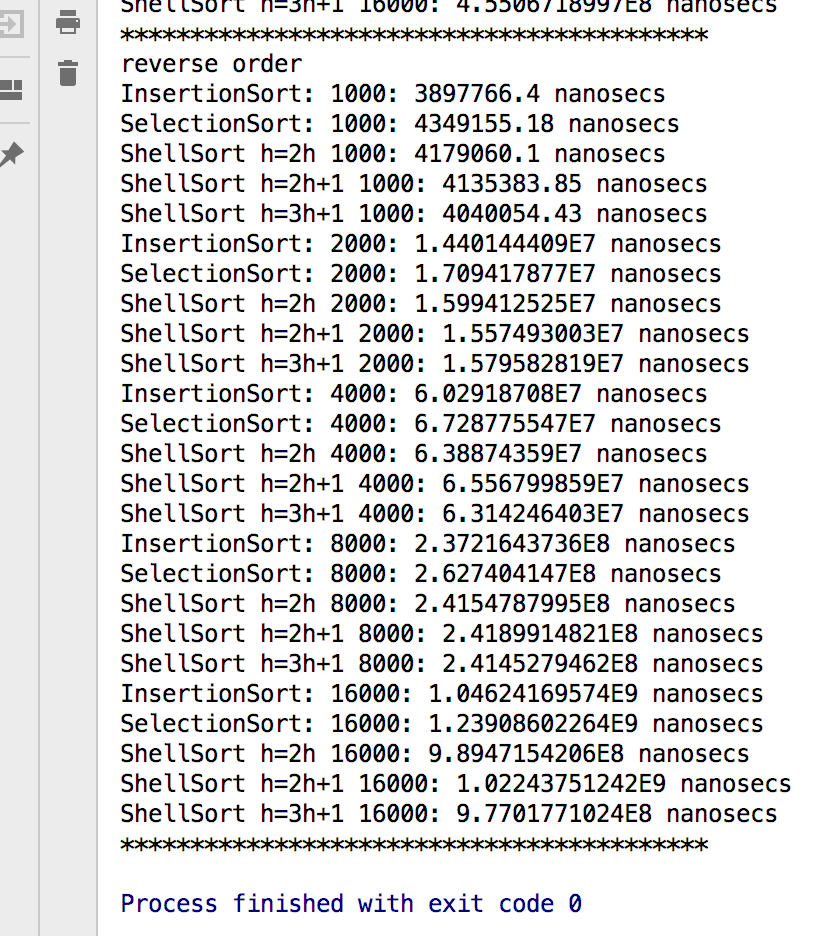
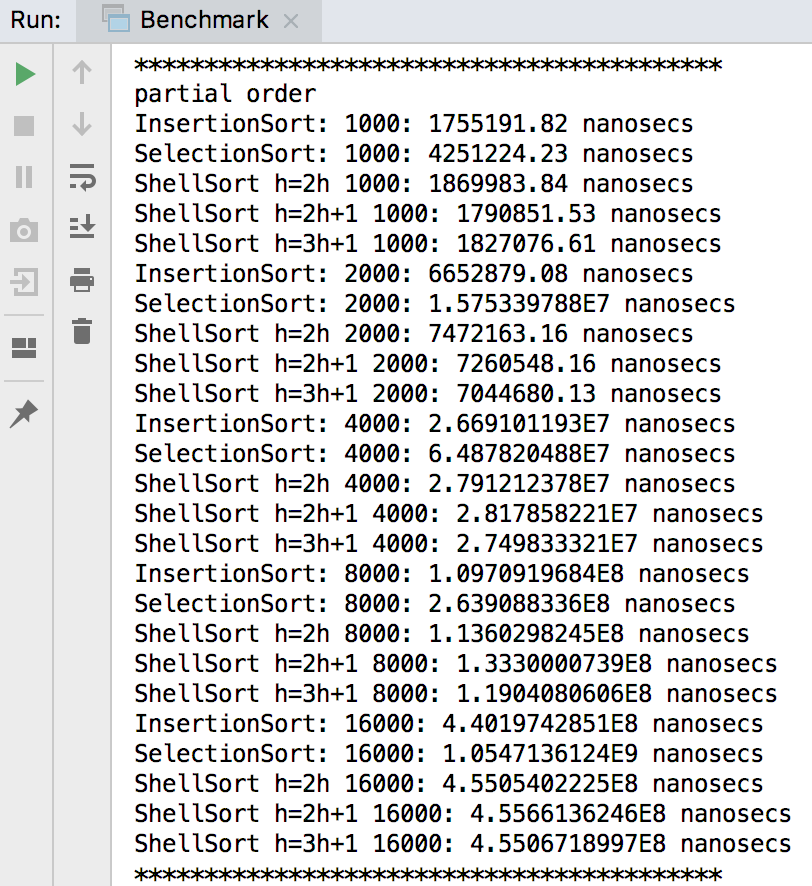
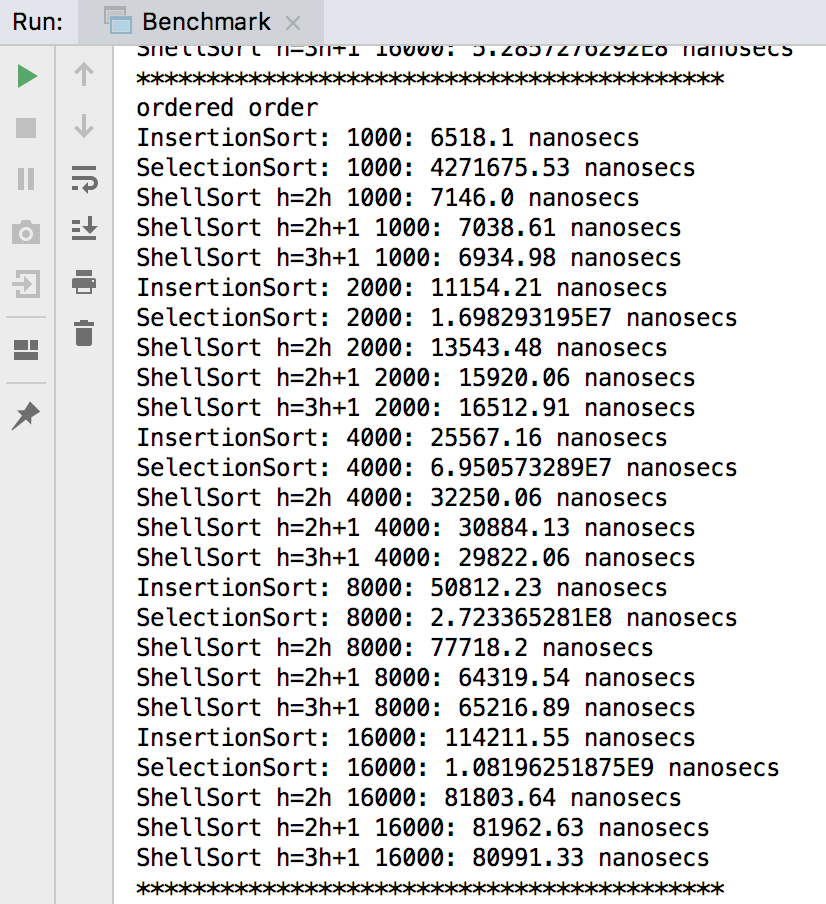
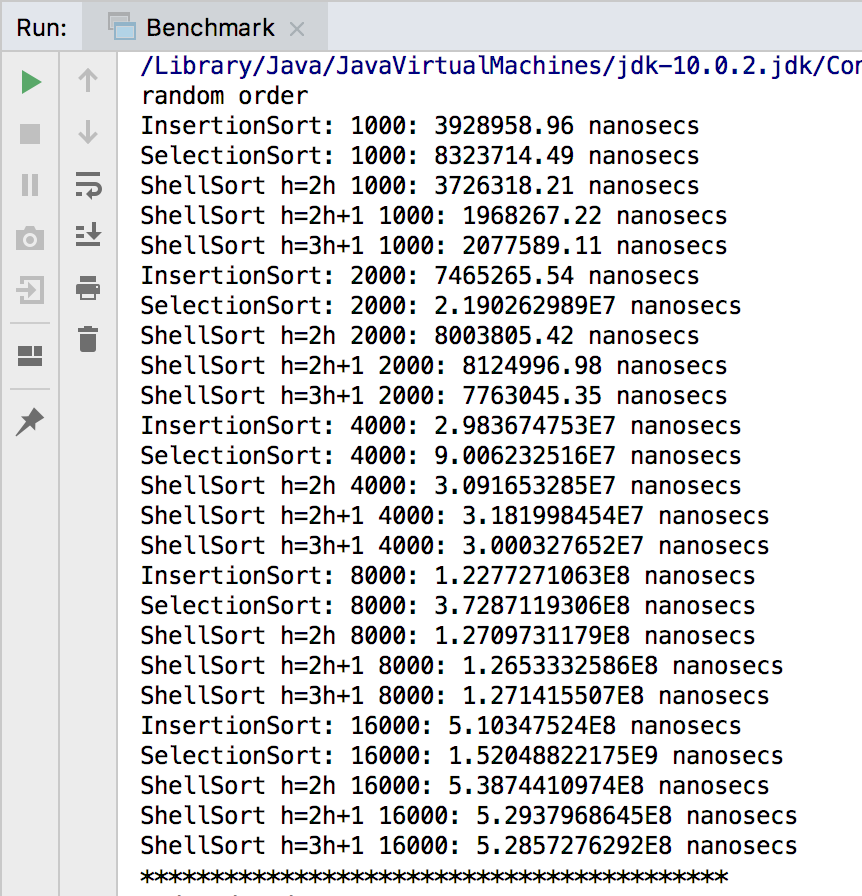
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SHELL(h=2h+1) | random | ordered | partial | reverse |
| 1000 | 1968267.22 | 7038.61 | 1790851.53 | 4135383.85 |
| 2000 | 8124996.98 | 15920.06 | 7260548.16 | 1.56E+07 |
| 4000 | 3.18E+07 | 30884.13 | 2.82E+07 | 6.56E+07 |
| 8000 | 1.27E+08 | 64319.54 | 1.33E+08 | 2.42E+08 |
| 16000 | 5.29E+08 | 81962.63 | 4.56E+08 | 1.02E+09 |

Table4 Shell Sort (h=2h+1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SHELL(h=3h+1) | random | ordered | partial | reverse |
| 1000 | 2077589.11 | 6934.98 | 1827076.61 | 4040054.43 |
| 2000 | 7763045.35 | 16512.91 | 7044680.13 | 1.58E+07 |
| 4000 | 3.00E+07 | 29822.06 | 2.75E+07 | 6.31E+07 |
| 8000 | 1.27E+08 | 65216.89 | 1.19E+08 | 2.41E+08 |
| 16000 | 5.29E+08 | 80991.33 | 4.55E+08 | 9.77E+08 |

Table5 Shell Sort (h=3h+1)

Screen shot of data I obtained:



According to observing and comparing these data, we can find that they matches the three conclusions above.

**Hence proved.**