**Wniosek**

Wersja 1: insertsort we wszystkich sortowniach „co h” i „co 1”,

Wersja 2: insertsort w sortowaniu „co h” i bubblesort w sortowaniu „co 1”

Wersja 3: bubblesort w sortowaniu „co h” i insertsort w sortowaniu „co 1”.

Jak widać z tabelki niżej:

* dla pierwszej wersji algorytmu Shella najszybciej odsortowało drugi ciąg.
* dla drugiej wersji najszybciej odsortowało ciąg z liczbami Fibonacciego.
* dla trzeciej wersji najszybciej odsortowało też ciąg z liczbami Fibonacciego.

Najdłuższej działa sortowanie dla ciągu A, co jest oczywistym, ponieważ w innych przypadkach ciągi już od razu są posortowane.

Dla 5000 elementów:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ciąg \ Wersja** | **1** | **2** | **3** |
| **A** | 15422 ns | 11 ms | 233 ms |
| **B** | 4112 ns | 2 ms | 120 ms |
| **C** | 5655 ns | 6 ms | 114 ms |
| **D** | 9767 ns | 1 ms | 105 ms |

W tym przypadku najlepszym ciągiem dla pierwszej wersji algorytmu jest ciąg B i C, dla drugiej – ciąg B, dla trzeciej – ciąg C.

Dla 10000 elementów:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ciąg \ Wersja** | **1** | **2** | **3** |
| **A** | 9767 ns | 6 ms | 670 ms |
| **B** | 8225 ns | 0 ms | 345 ms |
| **C** | 8225 ns | 2 ms | 323 ms |
| **D** | 9767 ns | 1 ms | 385 ms |

Dla 50000 elementów:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ciąg \ Wersja** | **1** | **2** | **3** |
| **A** | 11823 ns | 22 ms | 41995 ms |
| **B** | 1028 ns | 3 ms | 8785 ms |
| **C** | 2056 ns | 9 ms | 9266 ms |
| **D** | 2570 ns | 32 ms | 53671 ms |

Dla 100000 elementów:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ciąg \ Wersja** | **1** | **2** | **3** |
| **A** | 2570 ns | 77 ms | 325678 ms |
| **B** | 1028 ns | 8 ms | 41612 ms |
| **C** | 1542 ns | 19 ms | 65760 ms |