Анализ проводился в соответствии с полученными результатами, время указано в секундах :

Slow sort test

On random array:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10^3 элементов | 10^4 элементов | 10^5 элементов |
| Bubble sort | 0.021 | 1.463 | 143.003 |
| Bubble sort(lvl up) | 0.02 | 1.377 | 142.612 |
| Selection sort | 0.009 | 0.643 | 59.219 |
| Insertion sort | 0.009 | 0.668 | 72.864 |

Лучше всех оказалась сортировка выбором, скорее всего потому что в ней меньше всего присваиваний, хуже всех – сортировка пузырьком, также из-за того что происходит больше всего присваиваний.

On sorted array:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10^3 элементов | 10^4 элементов | 10^5 элементов |
| Bubble sort | 0.009 | 0.617 | 63.866 |
| Bubble sort(lvl up) | 0 | 0 | 0.002 |
| Selection sort | 0.009 | 0.604 | 60.999 |
| Insertion sort | 0 | 0 | 0.001 |

Быстрее всего оказалась сортировка вставками, т.к. внутренний цикл ни разу не выполнился и сортировка выполнилась за n итераций.

On sorted reverse array:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10^3 элементов | 10^4 элементов | 10^5 элементов |
| Bubble sort | 0.022 | 1.467 | 164.298 |
| Bubble sort(lvl up) | 0.022 | 1.654 | 164.652 |
| Selection sort | 0.01 | 0.692 | 76.52 |
| Insertion sort | 0.018 | 1.334 | 153.287 |

На сортированных в обратном порядке массивах ситуация такая же как и в первом случае, т.к. сортированный в обратном порядке массив – худший случай для этих сортировок.

On almost sorted array:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10^3 элементов | 10^4 элементов | 10^5 элементов |
| Bubble sort | 0.009 | 0.726 | 69.54 |
| Bubble sort(lvl up) | 0.001 | 0.138 | 17.387 |
| Selection sort | 0.011 | 0.631 | 60.252 |
| Insertion sort | 0.002 | 0.104 | 11.02 |

На почти сортированном быстрее всех оказалась сортировка вставками, т.к. она на почти отсортированных массивах работает за O(n).

Fast sort test

On random array:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10^5 элементов | 10^6 элементов | 10^7 элементов |
| Quick sort with selection of the middle element of the array as a pivot | 0.032 | 0.364 | 4.345 |
| Merge sort | 0.141 | 1.106 | 12.563 |
| Heap sort | 0.122 | 1.255 | 17.877 |
| Default sort from STL | 0.046 | 0.491 | 6.507 |
| Stable sort from STL | 0.05 | 0.752 | 7.657 |
| Improved quick sort | 0.011 | 0.187 | 2.471 |

On sorted array:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10^5 элементов | 10^6 элементов | 10^7 элементов |
| Quick sort with selection of the middle element of the array as a pivot | 0.021 | 0.188 | 2.498 |
| Merge sort | 0.105 | 0.886 | 10.793 |
| Heap sort | 0.129 | 0.967 | 10.908 |
| Default sort from STL | 0.041 | 0.303 | 4.004 |
| Stable sort from STL | 0.023 | 0.321 | 3.394 |
| Improved quick sort | 0.022 | 0.191 | 2.573 |

On sorted reverse array:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10^3 элементов | 10^4 элементов | 10^5 элементов |
| Quick sort with selection of the middle element of the array as a pivot | 0.025 | 0.235 | 2.621 |
| Merge sort | 0.09 | 0.933 | 10.247 |
| Heap sort | 0.122 | 0.958 | 12.068 |
| Default sort from STL | 0.024 | 0.294 | 4.173 |
| Stable sort from STL | 0.044 | 0.319 | 3.844 |
| Improved quick sort | 0.018 | 0.184 | 2.273 |

On almost sorted array:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10^3 элементов | 10^4 элементов | 10^5 элементов |
| Quick sort with selection of the middle element of the array as a pivot | 0.04 | 0.276 | 2.912 |
| Merge sort | 0.072 | 0.867 | 11.334 |
| Heap sort | 0.102 | 1.044 | 13.688 |
| Default sort from STL | 0.059 | 0.477 | 18.866 |
| Stable sort from STL | 0.048 | 0.355 | 3.804 |
| Improved quick sort | 0.01 | 0.161 | 2.306 |

На всех массивах быстрее всех сработала написанная мной улучшенная быстрая сортировка, она оказалась быстрее чем встроенная возможно потому что встроенная написана и для других структур данных, а не только для векторов. Самой медленной оказалась heap sort, скорее всего потому что кроме самой сортировки за nlogn, происходит построение кучи почти за n(в реальности больше) . На сортированном и почти отсортированном массивах stable sort была быстрее чем sort, возможно потому что она лишний раз не перемещает элементы, которые уже на своих местах.