|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название Сортировки** | **Скорость работы (время / итерации)** | | | |
| **size=10** | size=105 | | |
|  | **массив состоит из случайных элементов** | **массив почти упорядочен в нужном порядке** | **массив обладает малым "размахом"** |
| **INSERT SORT** | 8,9983 ms/21 =0,42 | 59445,6233 ms / 178895058= | 4, 9665/ | 410230,881 ms/ 2042546618= |
| **QUICK SORT** | 3,9907 ms/31= 0,1287322 | 31060,2715 ms/ 35779011,6 | 2, 2877/ | 40304,9721 ms/ |
| **MERGE SORT** | 9,9952 ms/39= 0,25 | 345235 ms/1967845,605 | 3, 7372/ | 50434,2156 ms/ |
| **COUNT SORT** | 5,998 ms/49=0,122 | 423359 ms/ 8916,843495 | 3, 3485/ | 11911,4972 ms/ 499999=0,023 |

Выводы:

В работы и полученных данных, для неотсортированного массива из 10 элементов оптимальнее всех оказались алгоритмы **подсчетом и быстрая сортировка.**

При почти упорядоченого массиве результаты отличаются не сильно, все алгоритмы сортировки показывают время примерно на 3-4 миллисекунды меньше.

Однако при сортировке почти упорядоченого массива **быстрая сортировка** срабатывает намного быстрее.

При малых “размахах” быстрее всех срабатывает сортировка посчетом.