|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Время в миллисекундах | |  |  |  |
|  | Вставками | Быстрая + выбор опорного числа |  |  |  |
| Маленький лучший | ~0.275ms | ~0.572ms | Лучший - отсортированный | | |
| Маленький средний | ~0.267ms | ~0.3002ms | Средний - не отсортирован | | |
| Маленький худший | ~0.914ms | ~0.312ms | Худший - отсортирован по убыванию | | |
| Большой лучший | ~701.979ms | ~187.343ms | Быстрая без выбора опорного числа (берём первый или последний элемент) переполняет стек на ~8000 элементах | | |
| Большой средний | ~3573.492ms | ~144.05ms |
| Большой худший | ~4821.792ms | ~131.773ms |
| Малый - 100 элем |  |  |  |  |  |
| Большой - 100000 элем |  |  |  |  |  |

На основании замеров времени, можно сделать вывод, что данные из таблицы подтвердились. Действительно, сортировка вставками справляется со средним и худшим случаем за N^2 времени (с погрешностью из-за случайного массива).

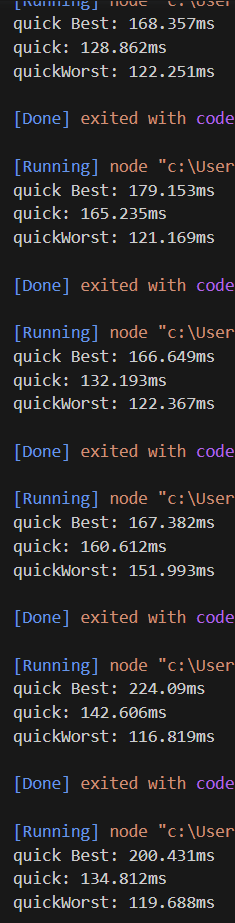
То же самое можно сказать и про быструю сортировку. Для неё худшим случаем на деле является заранее отсортированный массив, который выполняется ~N^2 времени.

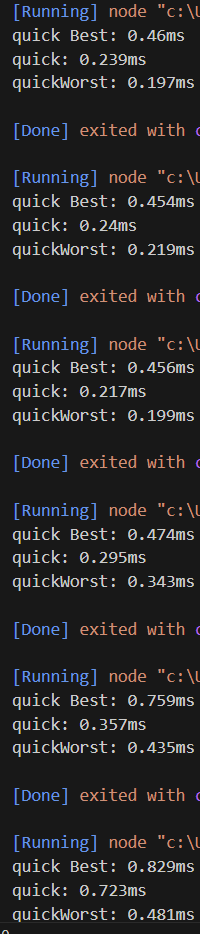
Исходя из исследования, можно сделать вывод что сортировка вставками лучше справляется с заранее упорядоченным массивом, а быстрая сортировка - с массивом с большими данными. При реализации быстрой сортировки с выбором первого элемента массива в виде опорного числа, показывает результат намного хуже, и переполняет стек на значениях ~8000 элементах

Ссылка на код - <https://codesandbox.io/p/sandbox/white-smoke-rlnx3m>

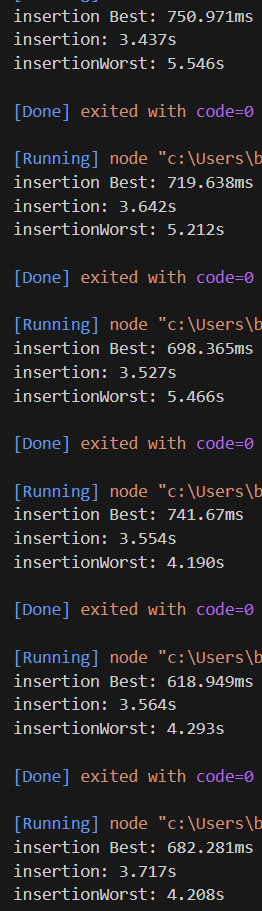
Если что-то сломается, Github - <https://github.com/Rolan335/2.2-semester/blob/master/Algorythms_Data_structure/kt4.js>

Быстрая сортировка

 Малый массив Большой массив



Сортировка вставками

 Малый массив Большой массив

