**Ловчикова Юлия, БПИ192**

Сравнительный анализ сортировок

1. BubbleSort

Сложность O(n2)

На массивах с большим количеством элементов работает дольше всех остальных сортировок. На массивах с небольшой вместимостью работает средне по отношению к другим сортировкам. В целом, график представляет собой параболу, даже вблизи нуля почти приближенную к эталонному графику.

1. InsertionSort

Сложность:

* худшее время O(n2)
* лучшее время O(n)
* среднее время O(n2)

Память: O(n) всего

Работает примерно так же, как пузырьковая сортировка, но почти всегда чуть быстрее (графики идут, примыкая друг к другу). Малейшее отставание по скорости от «пузырька» происходит лишь на почти отсортированном массиве с маленькой вместимостью, что объяснимо алгоритмами в вырожденных случаях: «пузырёк» в почти отсортированном массиве хоть и по-пержнему делает квадрат сравнений, но делает околонуля обменов. Алгоритм вставками же вынужден сдвигать часть массива на один индекс вправо столько раз, сколько нашлось элементов не на своих местах, что, естественно, проигрывает обычному обмену двух элементов.

1. BinaryInsertionSort

Сложность:

* худшее время O(n2)
* лучшее время O(n)
* среднее время O(n2)

Память: O(n) всего

Всегда работает быстрее остальных алгоритмов за счёт оптимизации поиска верной позиции бинарным поиском.

1. CountingSort

Почти во всех случаяях работает примерно так же, как и сортировка бинарными вставками, но чуть медленнее; имеет сложность в среднем 𝛩(n+k), где k – наибольшее число в массиве, поэтому на массивах с маленьким числом больших элементов (то есть k>>n) сортировка подсчётом начинает уступать почти половине оставшихся сортировок.

1. RadixSort

Почти линейная сортировка; на каждый из четырёх разрядов в 256-ричной системе использует линейную сортировку выбором, поэтому и у цифрового алгоритма относительно линейная сложность, что подтверждают графики: на всех данных цифровая сортировка работает медленнее сортировки подсчётом и бинарными вставками с небольшим отставанием. Исключения составляют массивы с малой вместимостью и большими значениями элементов, что закономерно и следует из зависимости сложности сортировки подсчётом от наибольшего элемента в массиве.

1. MergeSort

Сложность:

* худшее время O(n log n)
* лучшее время O(n log n)
* среднее время O(n log n)

Память: O(n) вспомогательной

На маленьких размерах массивов проигрывает, а на больших является «разделителем» между быстрыми и медленными сортировками, что явно следует из анализа функции n log n.

1. HoarSort

Сложность:

* худшее время O(n2)
* среднее время O(n log n)

Работает медленнее, чем сортировки бинарными вставками, цифровая и подсчётом, но всё ещё остаётся одной из самых быстрых. Отстаёт от них по времени за счёт того, что время выполнения определяется глубиной рекурсии. Хуже всего ведёт себя на массивах больших размеров с маленькими числами за счёт большого количества повторяющихся чисел и, соответсвенно, обменов и глубины рекурсии.

1. HeapSort

Сложность:

* худшее время O(n log n)

Память: O(1) дополнительной памяти

На массивах с маленькой вместимостью работает медленнее остальных алгоритмов. На почти отсортированных массивах работает так же долго, как и на ранодмизированных данных. Минимальный выигрыш в скорости происходит при большой размерности массива, но скудный.