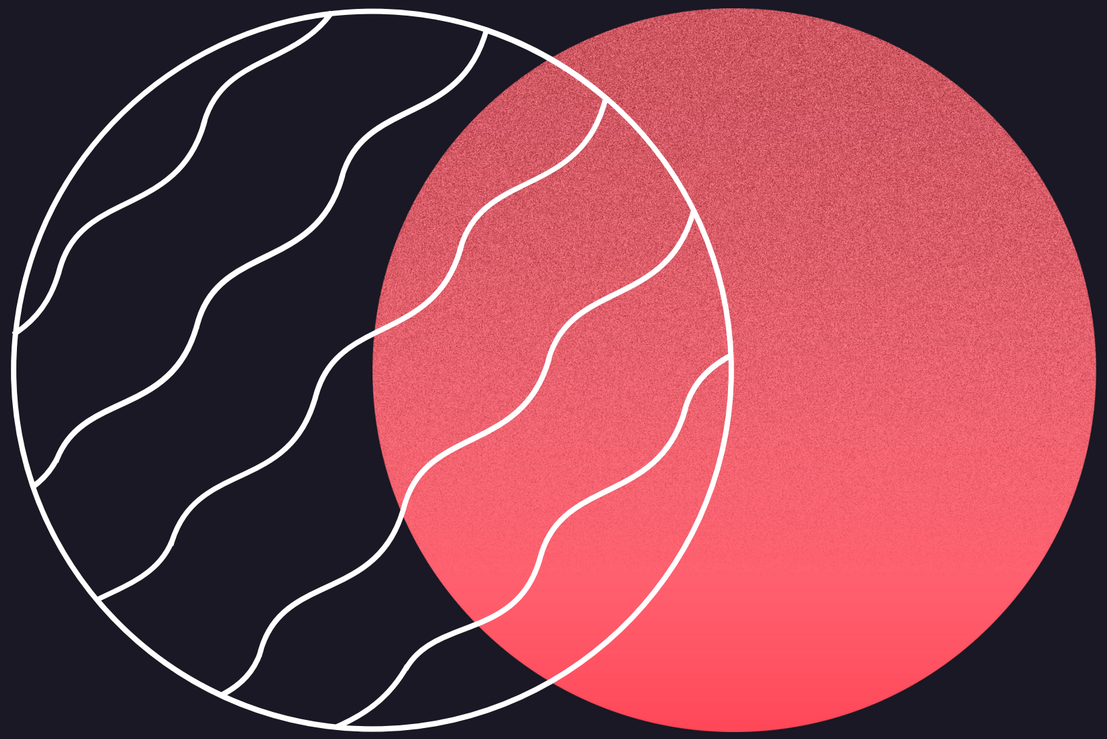


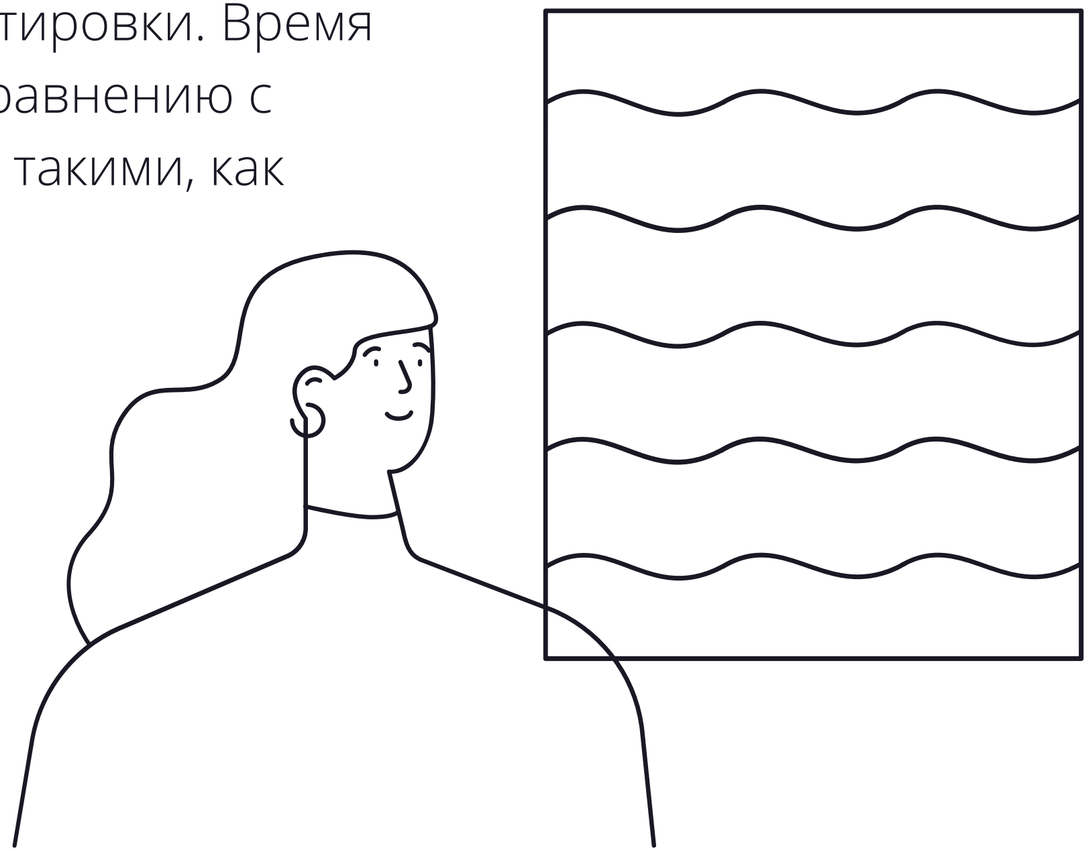
Алгоритмы сортировки Comb Sort & Stooge Sort

Работу выполнил :
Белобородов Андрей 11-103

Что такое Stooge Sort и Comb Sort?

Stoge Sort (Сортировка по частям, Блуждающая сортировка) - это рекурсивный алгоритм сортировки. Время работы алгоритма крайне большое по сравнению с эффективными алгоритмами сортировки, такими, как Сортировка слиянием.

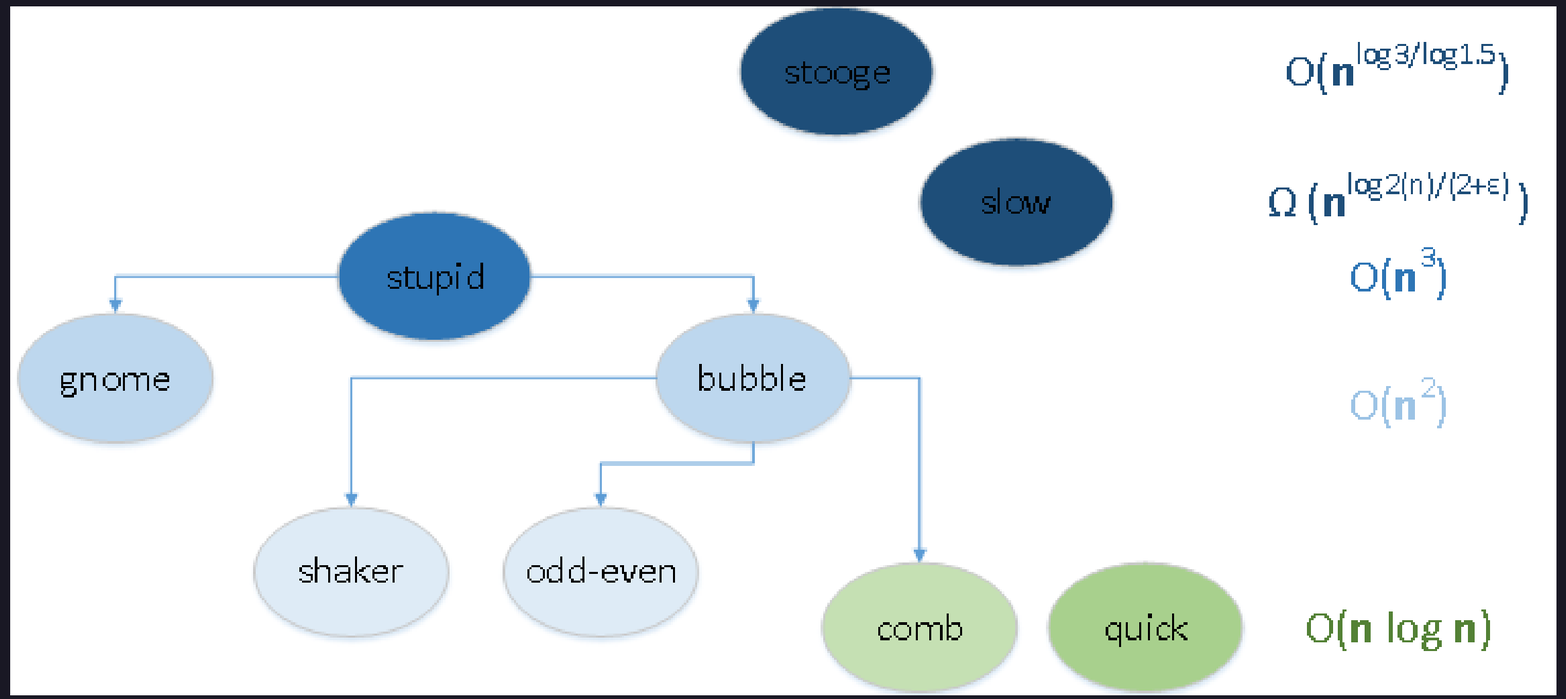
Comb Sort - это ещё одна модификация сортировки пузырьком, упрощающая данную сортировку. Конкурирует с алгоритмами, подобные быстрой сортировке.



03

Придурковая сортировка представляет собой чисто академический интерес из-за своей неэффективности по времени выполнения, она медленнее, чем сортировка пузырьков.

А сортировка расчёской — это довольно упрощённый алгоритм, она улучшает сортировку пузырьком, и конкурирует с алгоритмами, подобными быстрой сортировке.

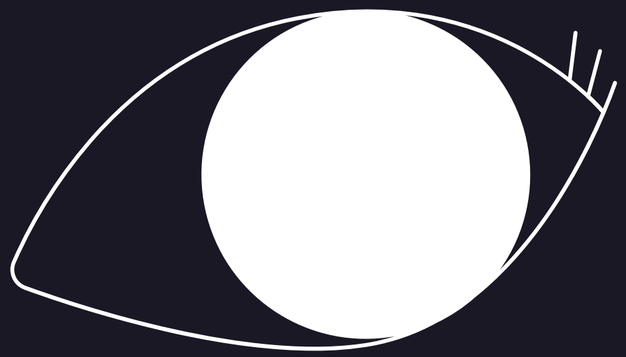


ОЦЕНОЧНОЕ ВРЕМЯ АЛГОРИТМОВ

Время выполнения алгоритма Stooge Sort можно записать как O (n (log3 / log1.5)) = O (n 2.709). Следовательно, она медленнее, чем Сортировка Пузырьком. Данная сортировка считается нетрадиционной и неэффективной, её мало где используют.

А вот Comb Sort наоборот будет быстрее сортировки пузырьком. O(nlogn), в худшем – O(n2).

	Best	Average	Worst
Quick Sort	$\Omega (n \log (n))$	$\Theta (n \log(n))$	$O (n^2)$
Merge Sort	$\Omega (n \log (n))$	$\Theta (n \log(n))$	$O (n \log(n))$
Timsort	$\Omega (n)$	$\Theta (n \log(n))$	$O (n \log(n))$
Heap Sort	$\Omega (n \log (n))$	$\Theta (n \log(n))$	$O (n \log(n))$
Bubble Sort	$\Omega (n)$	$\Theta (n^2)$	$O (n^2)$
Insertion Sort	$\Omega (n)$	$\Theta (n^2)$	$O (n^2)$
Selection Sort	$\Omega (n^2)$	$\Theta (n^2)$	$O (n^2)$
Tree Sort	$\Omega (n \log (n))$	$\Theta (n \log(n))$	$O (n^2)$
Shell Sort	$\Omega (n \log (n))$	$\Theta (n (\log(n))^2)$	$O (n (\log(n))^2)$
Bucket Sort	$\Omega (n+k)$	$\Theta (n+k)$	$O (n^2)$
Radix Sort	$\Omega (nk)$	$\Theta (nk)$	$O (nk)$
Counting Sort	$\Omega (n+k)$	$\Theta (n+k)$	$O (n+k)$
Cubesort	$\Omega (n)$	$\Theta (n \log(n))$	$O (n \log(n))$
Smooth Sort	$\Omega (n)$	$\Theta (n \log(n))$	$O (n \log(n))$
Tournament Sort	-	$\Theta (n \log(n))$	$O (n \log(n))$
Stooge sort	-	-	$O(n^{\log 3 / \log 1.5})$
Gnome/Stupid sort	$\Omega (n)$	$\Theta (n^2)$	$O (n^2)$
Comb sort	$\Omega (n \log (n))$	$\Theta (n^2/p^2)$	$O (n^2)$
Odd – Even sort	$\Omega (n)$	-	$O (n^2)$



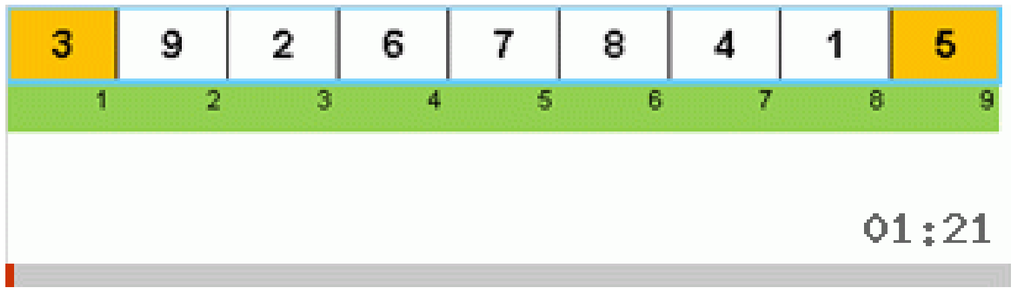
КАК РАБОТАЮТ ДАННЫЕ АЛГОРИТМЫ?

STOOGE SORT

- Сравниваем элементы на концах отрезка (первоначально это весь массив).

- Если на левом конце больше чем на правом, то меняем местами.

- Рекурсивно применяем сортировку для первых 2/3 элементов списка.



- Рекурсивно применяем сортировку для последних 2/3 элементов списка.

- Снова рекурсивно применяем сортировку для первых 2/3 элементов списка.

Stooge Sort - это рекурсивный алгоритм сортировки. Это неэффективный, но интересный алгоритм сортировки. Он делит массив на две перекрывающиеся части (по 2/3 каждая). Затем он выполняет сортировку в первой части 2/3, а затем выполняет сортировку в последней части 2/3. После этого сортировка выполняется на первых 2/3 части, чтобы убедиться, что массив отсортирован.

```
public class StoogeSort<T> implements Sorting<T> {

    private void StoogeSort(T[] array, int l, int r, Comparator<? super T> c) {
        if (c.compare(array[l], array[r]) > 0) new Swap<T>().swap(array, l, r);
        if (r - l < 2) return;
        int k = (r - l + 1) / 3;
        StoogeSort(array, l, r - k, c);
        StoogeSort(array, l + k, r, c);
        StoogeSort(array, l, r - k, c);
    }

    @Override
    public void sort(T[] array, Comparator<? super T> c) {
        StoogeSort(array, 0, array.length - 1, c);
    }
}
```

Это неэффективный алгоритм сортировки и он даже медленнее, чем сортировка пузырьками, но он привносит новую идею сортировки.

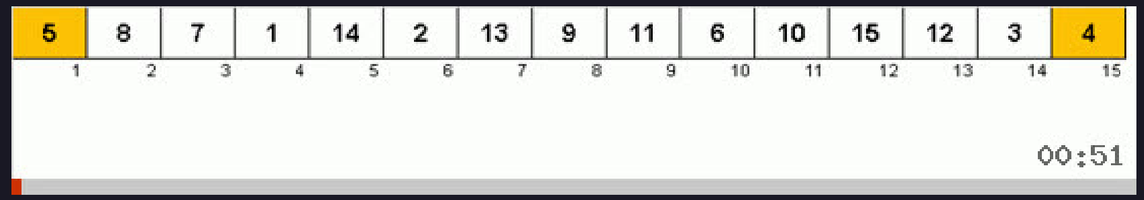
COMB SORT

Еще одна модификация сортировки пузырьком. Для того, чтобы избавиться от «черепах», будем переставлять элементы, стоящие на расстоянии.

Зафиксируем его и будем идти слева направо, сравнивая элементы, стоящие на этом расстоянии, переставляя их, если необходимо. Очевидно, это позволит «черепахам» быстро добраться в начало массива.

Оптимально изначально взять расстояние равным длине массива, а далее делить его на некоторый коэффициент, равный примерно 1.247. Когда расстояние станет равно единице, выполняется сортировка пузырьком.

В лучшем случае асимптотика равна O(nlogn), в худшем – O(n2). Какая асимптотика в среднем мне не очень понятно, на практике похоже на O(nlogn).

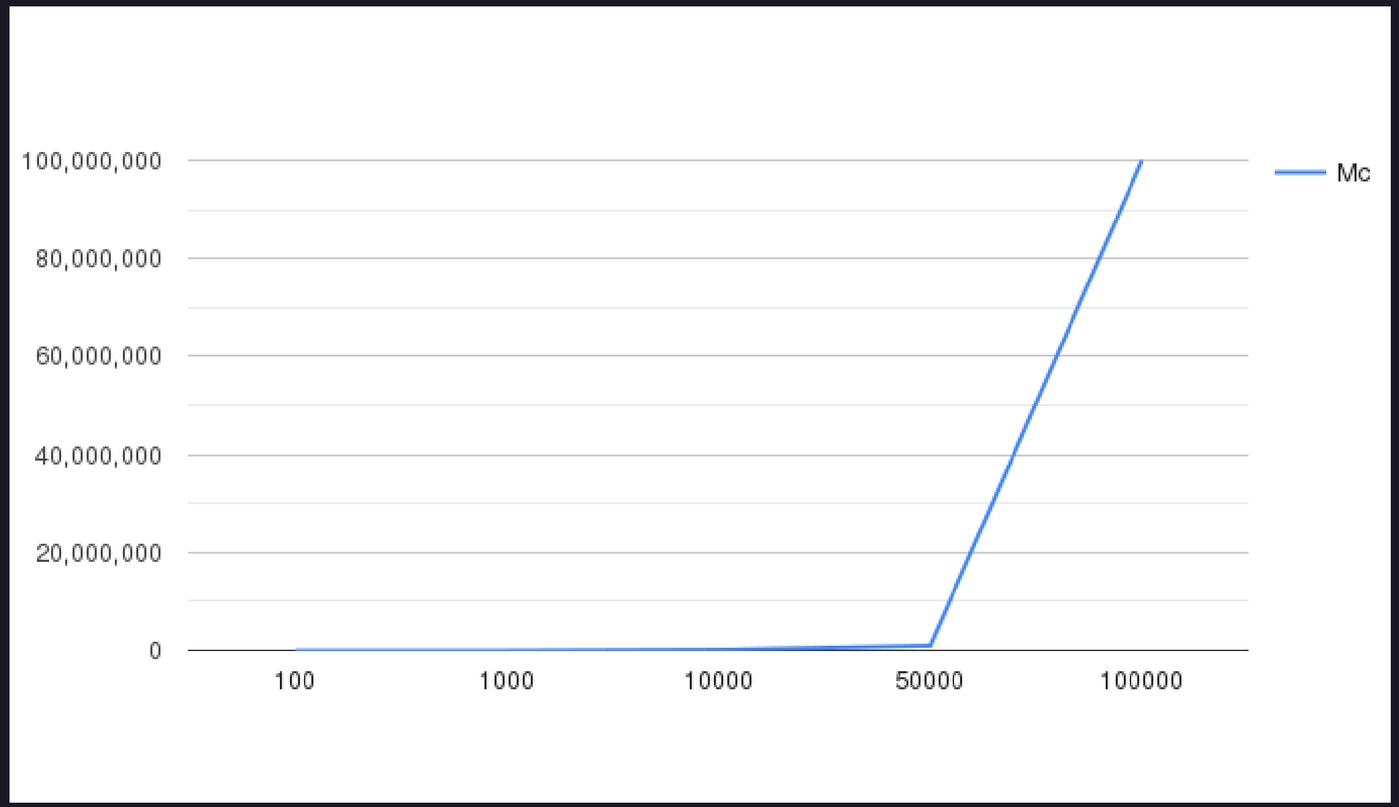


```java
public class CombSort<T> implements Sorting<T> {


    @Override
    public void sort(T[] array, Comparator<? super T> c) {

        for (int len = array.length - 1; len > 0; len--)
            for (int i = 0; i + len < array.length; i++)
                if (c.compare(array[i], array[i + len]) > 0)
                {
                    new Swap<T>().swap(array, i, i + len);
                }


    }
}
```

Сортировка расчёской улучшает сортировку пузырьком, и конкурирует с алгоритмами, подобными быстрой сортировке. Основная идея — устранить черепах, или маленькие значения в конце списка, которые крайне замедляют сортировку пузырьком (кролики, большие значения в начале списка, не представляют проблемы для сортировки пузырьком).

В сортировке пузырьком, когда сравниваются два элемента, промежуток (расстояние друг от друга) равен 1. Основная идея сортировки расчёской в том, что этот промежуток может быть гораздо больше, чем единица.

Тестирование на сгенерированных данных

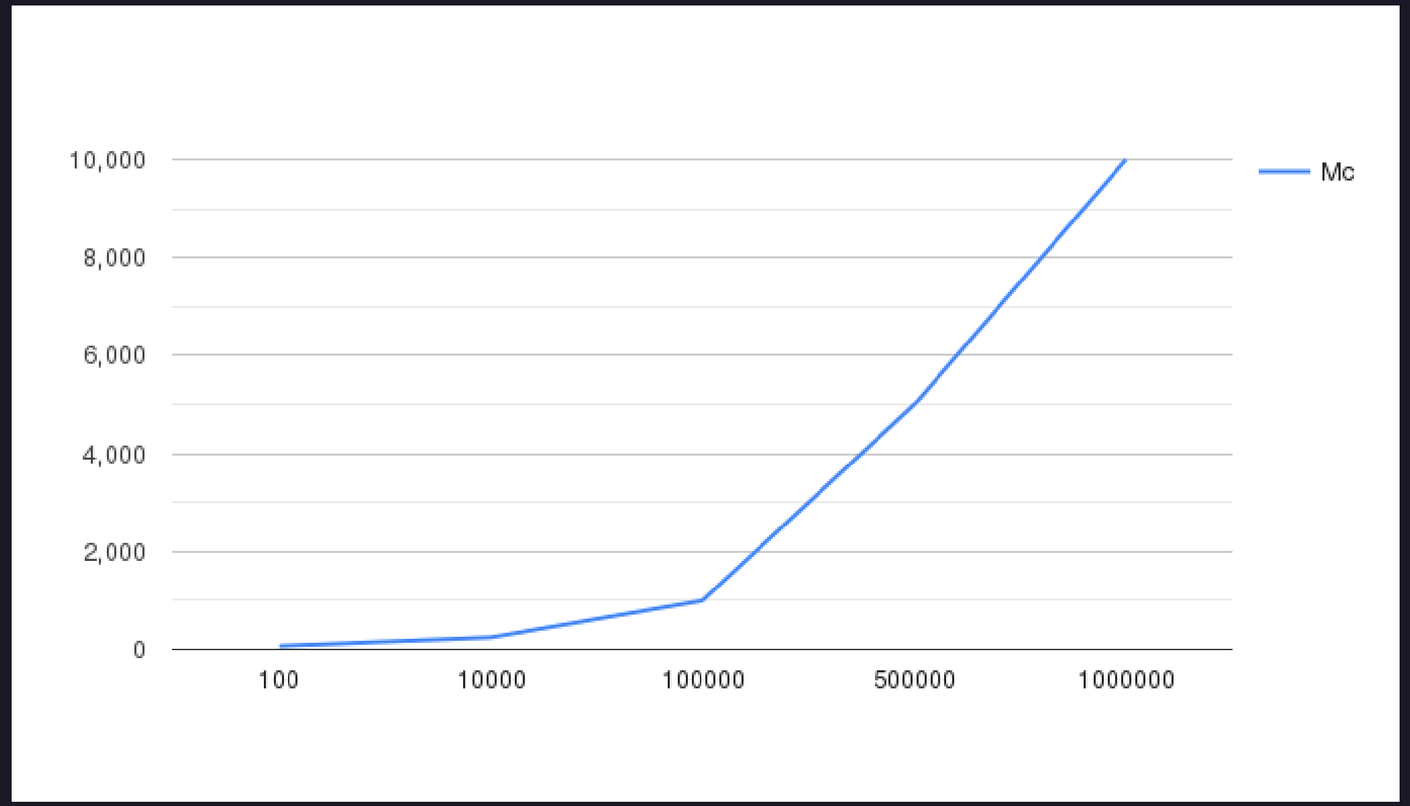


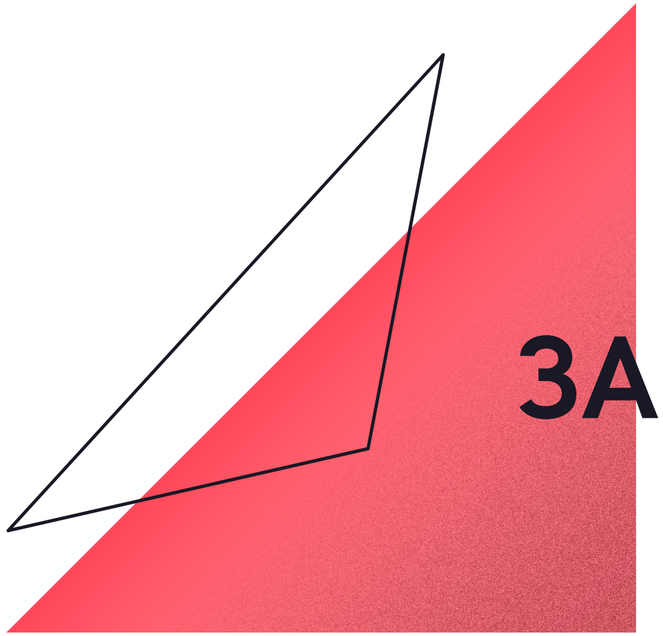
Были сгенерированы различные числа от 0 до 5000 и массивы с замерами от 100 до 1000000 элементов. Высчитывалось время выполнения алгоритма в миллисекундах, за которое выполнялся алгоритм.

По итогам подсчёта Stooge Sort получилось, что:
- на 100 эл заняло около 16 мс
- на 1000 элементов - 250 мс
- на 10000 эл. - 173651 мс
- на 50000 эл. - около 1000000 мс
- на 1000000 эл. - 1000000000 мс

Так же были сгенерированы различные числа от 0 до 5000 и массивы с замерами от 100 до 1000000 элементов. Высчитывалось время выполнения алгоритма в миллисекундах, за которое выполнялся алгоритм.

По итогам подсчёта Comb Sort получилось, что:
- на 100 эл заняло около 25 мс
- на 1000 элементов - 75 мс
- на 10000 эл. - 250 мс
- на 50000 эл. - около 1000 мс
- на 1000000 эл. - 10000 мс





ЗАКЛЮЧЕНИЕ

Как итог можно отметить, что мы рассмотрели две противоположные сортировки.

Одна сортировка, Comb Sort, часто используется, считается достаточно быстрой в своём исполнении, она улучшает привычную Сортировку Пузырьком.

Вторая, Stooge Sort, создана просто так, представляет собой академический интерес. Она не эффективна, мало где используется, существует просто так.