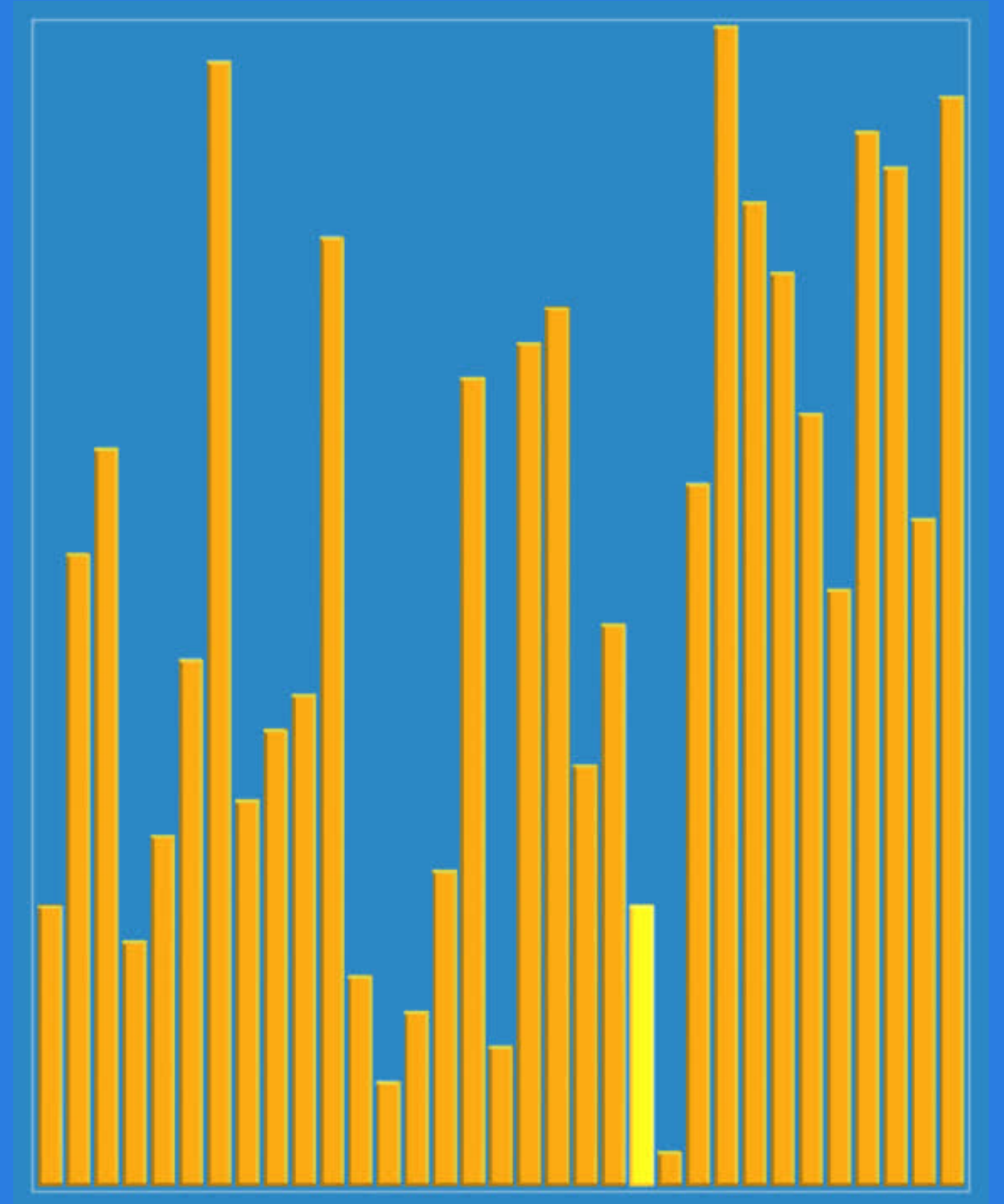


Shell Sort

Сортировка Шелла

Соловьев Данил 11-103



Описание и преимущества

Сортировка Шелла (англ. Shell sort) — улучшенная сортировка вставками. Соответственно, сложность у неё квадратичная.

Преимущества:

- **отсутствие потребности в памяти под стек**
- **отсутствие деградации при неудачных наборах данных — быстрая сортировка легко деградирует до $O(n^2)$, что хуже, чем худшее гарантированное время для сортировки Шелла.**

Идея



Идея метода заключается в сравнение разделенных на группы элементов последовательности, находящихся друг от друга на некотором расстоянии. Изначально это расстояние равно d или $N/2$, где N — общее число элементов. На первом шаге каждая группа включает в себя два элемента расположенных друг от друга на расстоянии $N/2$; они сравниваются между собой, и, в случае необходимости, меняются местами. На последующих шагах также происходят проверка и обмен, но расстояние d сокращается на $d/2$, и количество групп, соответственно, уменьшается. Постепенно расстояние между элементами уменьшается, и на $d = 1$ проход по массиву происходит в последний раз.



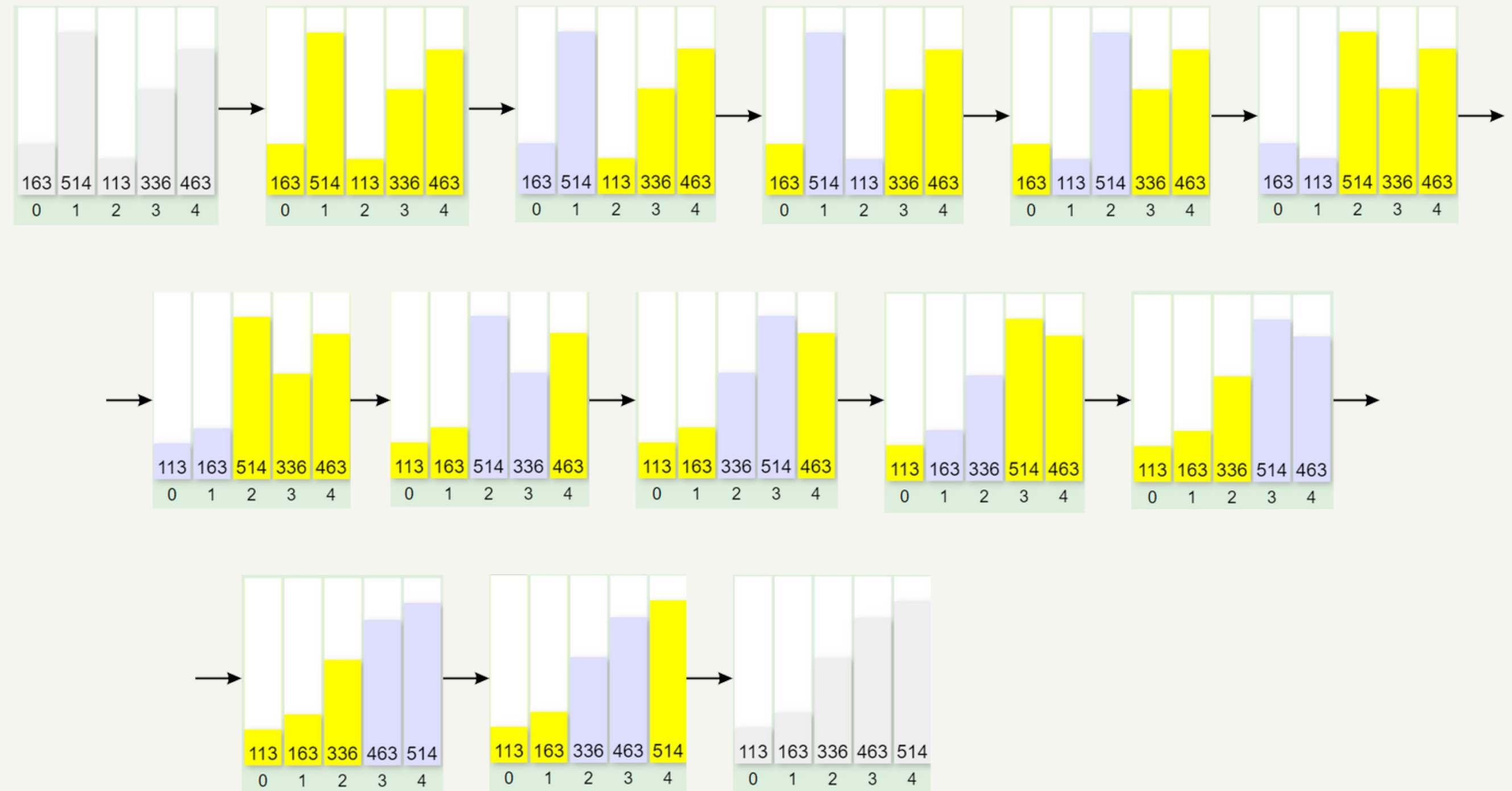
Назовем обменом перестановку местами двух элементов, тогда:

Исходный массив	32 95 16 82 24 66 35 19 75 54 40 43 93 68	
После сортировки с шагом 5	32 35 16 68 24 40 43 19 75 54 66 95 93 82	6 обменов
После сортировки с шагом 3	32 19 16 43 24 40 54 35 75 68 66 95 93 82	5 обменов
После сортировки с шагом 1	16 19 24 32 35 40 43 54 66 68 75 82 93 95	15 обменов

Более подробная визуализация на произвольном примере

-  - Не активные
-  - Сравниваются

Здесь
расстояние
 $d = 2$



Интересно, но качество данного алгоритма зависит от последовательности значений расстояния d . Существует несколько подходов к выбору этих значений:

1. Первоначально используемая Шеллом последовательность длин промежутков:

$d_1 = N/2, d_i = d_{i-1}/2, d_k = 1$ в худшем случае, сложность алгоритма составит $O(N^2)$

2. Предложенная Хиббардом последовательность: все значения $2^i - 1 \leq N, i \in \mathbb{N}$ такая последовательность шагов приводит к алгоритму сложностью $O(N^{3/2})$

3. Предложенная Седжвиком последовательность:

$d_i = 9 \cdot 2^i - 9 \cdot 2^{i/2} + 1$, если i четное и

$d_i = 8 \cdot 2^i - 6 \cdot 2^{(i+1)/2} + 1$, если i нечетное.

При использовании таких приращений средняя сложность алгоритма составляет: $O(n^{7/6})$, а в худшем случае порядка $O(n^{4/3})$.

●●● Java Realization

```
public static void AscendingShellSort(Integer[] a , int len ) {
    for (int d = len / 2; d > 0; d /= 2) {
        for (int i = d; i < len; i++) {
            for (int j = i - d; j >= 0 && a[j] > a[j + d] ; j -= d) {
                int temp = a[j];
                a[j] = a[j + d];
                a[j + d] = temp;
            }
        }
    }
}

public static void DescendingShellSort(Integer[] a , int len ) {
    for (int d = len / 2; d > 0; d /= 2) {
        for (int i = d; i < len; i++) {
            for (int j = i - d; j >= 0 && a[j] < a[j + d] ; j -= d) {
                int temp = a[j];
                a[j] = a[j + d];
                a[j + d] = temp;
            }
        }
    }
}
```

Так как шаг выбран в соответствии с оригинальным алгоритмом ($d = \text{length}/2$), то сложность по времени $O(N^2)$

Сортировка Шелла более эффективна при работе с большими объемами данных

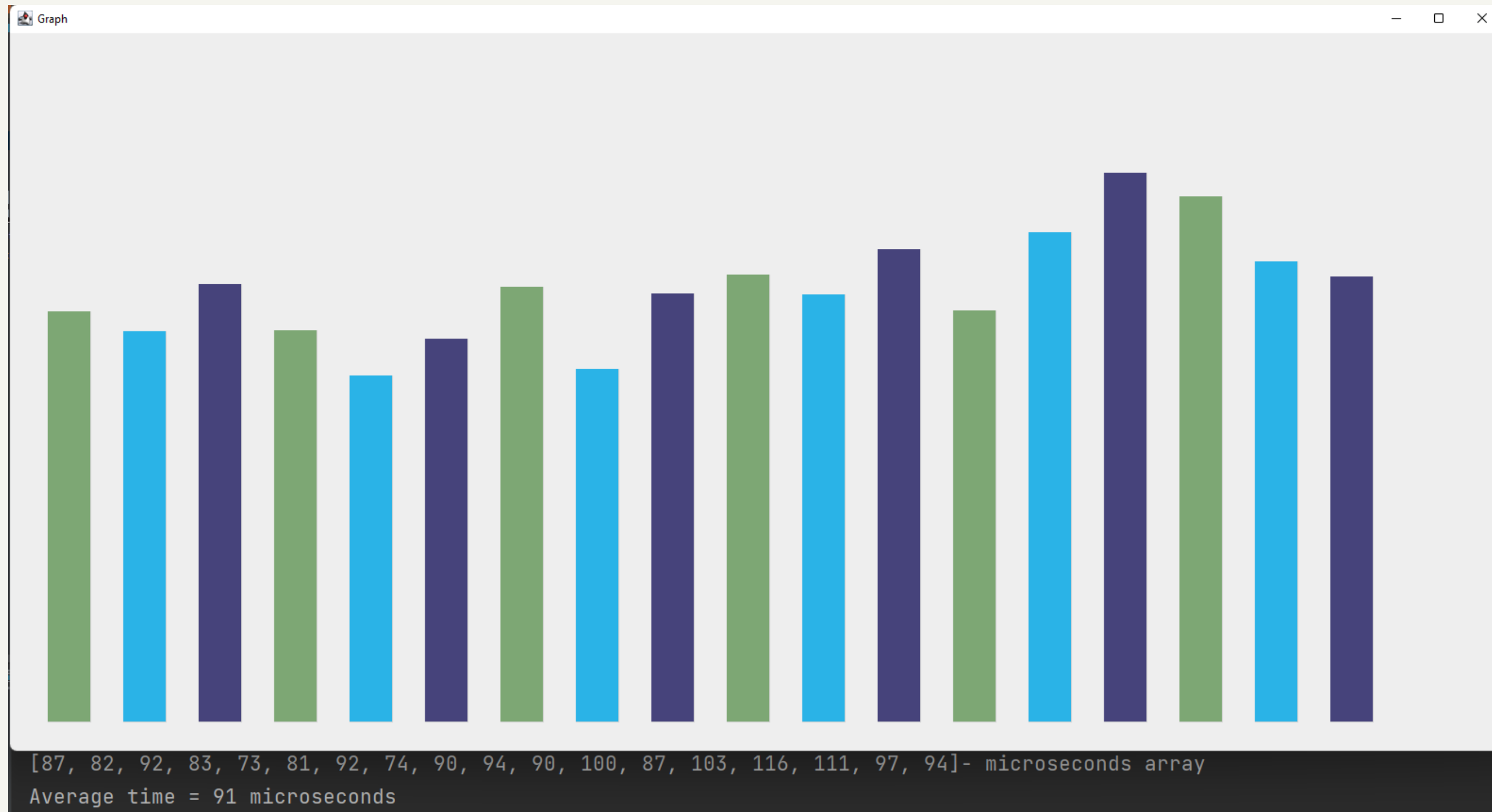
Результат расчета среднего времени сортировки массива из 100 элементов - **1000 измерений**

```
Array containing the values of the element Ascending/Descending Shell sorting time measurements in some of MEASURING_QUANTITY arrays
[213, 109, 98, 94, 97, 77, 84, 91, 100, 103, 127, 96, 135, 165, 96, 94, 101, 92, 102, 93, 104, 96, 92, 100, 93, 94, 87, 78, 76, 88, 84,
Average time = 18 microseconds
81 - random ShellSorted array`s time
```

Результат расчета среднего времени сортировки массива из 100 элементов - **50 измерений**

```
Array containing the values of the element Ascending/Descending Shell sorting time measurements in some of MEASURING_QUANTITY arrays
[136, 95, 92, 81, 87, 91, 97, 102, 85, 96, 103, 93, 88, 80, 101, 95, 84, 91, 90, 78, 83, 99, 93, 87, 93, 82, 91, 81, 85, 89, 79, 88, 98,
Average time = 91 microseconds
99 - random ShellSorted array`s time
```


... Визуализация для 18 измерений



●●● Визуализация для 1000 измерений

