**Shell Sort**

**Сортировка оболочкой** (Shell sort) — это в основном разновидность **сортировки вставкой** (Insertion Sort). При insertion sort мы перемещаем элементы только на одну позицию вперед. Когда элемент нужно переместить далеко вперед, задействовано много перемещений. Идея Shell Sort заключается в том, чтобы разрешить обмен далекими элементами. При Shell Sort мы делаем массив h-сортированным для большого значения h. Мы продолжаем уменьшать значение h, пока оно не станет равным 1. Массив называется h-сортированным, если все подсписки каждого h-го элемента отсортированы.

**Алгоритм:**

Шаг 1 − Начало

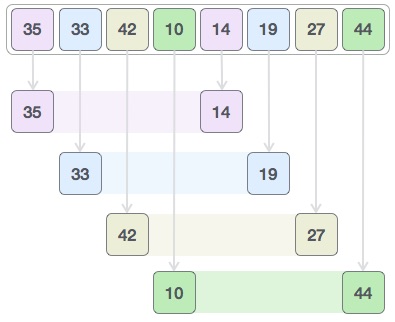
Шаг 2 − Инициализация значения размера зазора, скажем, h.

Шаг 3 − Разделение списка на меньшие части. Каждая должна иметь равные интервалы относительно h.

Шаг 4 − Отсортируйте эти подсписки с помощью сортировки вставкой.

Шаг 5 — Повторите этот шаг 2, пока список не будет отсортирован, отсортирован он будет в том случае, когда зазор будет равен h = 1.

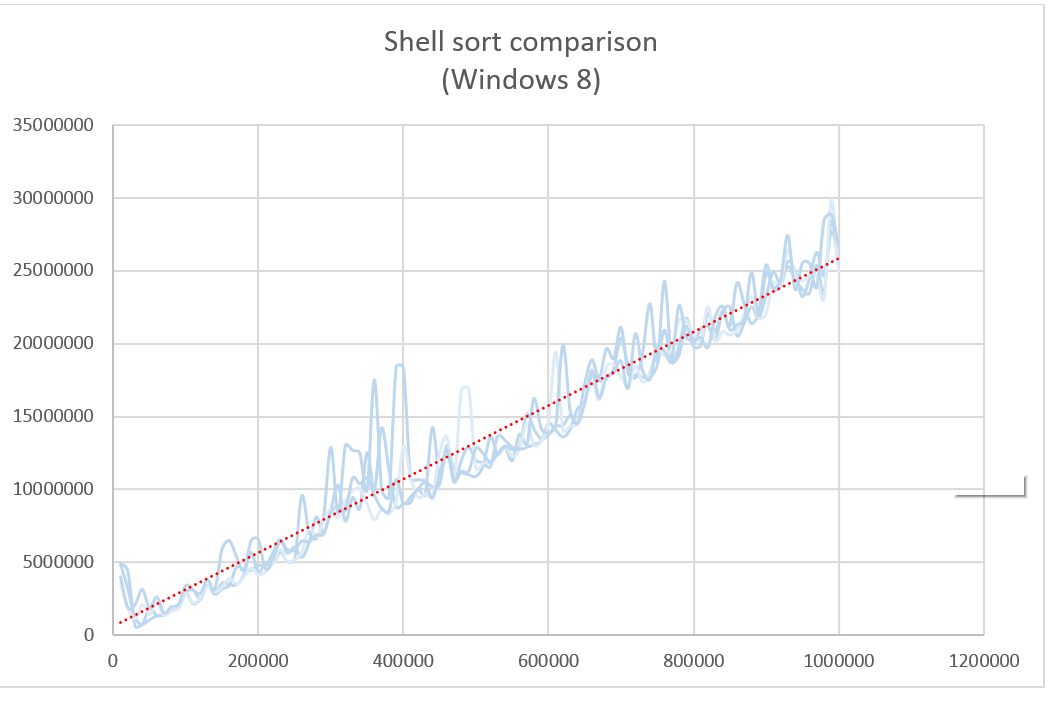
Шаг 6 — Отсортированный список готов.



**1. Какая сортировка более эффективна: Shell или Heap?**

Ответ. Согласно нотации big-O, Shell-сортировка имеет среднюю временную сложность O(n^{1.25}), тогда как Heap-сортировка имеет временную сложность O(N log N). Согласно строгой математической интерпретации нотации big-O, Heap-сортировка превосходит Shell-сортировку по эффективности, когда мы приближаемся к 2000 сортируемых элементов.

Примечание: Big-O — это округленное приближение, и аналитическая оценка не всегда на 100% верна, это зависит от реализации алгоритмов, которая может повлиять на фактическое время выполнения.



**Временная сложность**: Временная сложность вышеприведенной реализации Shell Sort составляет O(n2). В вышеприведенной реализации разрыв уменьшается вдвое в каждой итерации. Существует много других способов уменьшить разрывы, что приводит к лучшей временной сложности.

**Сложность в худшем случае:** Сложность в худшем случае для Shell Sort составляет O(n2). Это очевидно, ведь данное может произойти в том случае, когда алгоритм начнет повторять действия Insertion Sort. То есть, когда сортировке предстоит переставлять каждый элемент на свою позицию. Это может возникнуть, когда массив отсортирован в обратном порядке.

**Сложность в лучшем случае:** Когда заданный список массивов уже отсортирован, общее количество сравнений каждого интервала равно размеру заданного массива.

Таким образом, сложность в лучшем случае составляет Ω (n log(n))

**Сложность в среднем случае:** Сложность в среднем случае: O (n\*log n) ~ O(n1.25). При этом в худшем среднем случае может произойти сложность O (n1.5).

**Сложность по памяти**: Сложность по памяти Shell Sort составляет O (1).

**Применение Shell Sort**

1. Замена Insertion Sort, где выполнение заданной задачи занимает много времени.

2. Для вызова стековых накладных расходов.

3. Когда рекурсия превышает определенный предел.

4. Для наборов данных среднего и большого размера.

5. В Insertion Sort для сокращения количества операций.

**Недостатки Shell Sort**

1. Существуют более быстрые алгоритмы сортировок, такие как Quick Sort, Merge Sort или Heap Sort
2. Shell Sort является не стабильным алгоритмом сортировки, т.е. она может изменить порядок равных элементов на разных запусках.
3. Работает медленно на больших входных данных.
4. Относительная сложность реализации алгоритма по сравнению с более простыми алгоритмами, такими как Bubble Sort или Selection Sort
5. Shell Sort может уступать Insertion Sort, хотя и предполагался как его модификация. Уступает Shell Sort в случае, когда массив почти отсортирован. Для Insertion Sort нужно будет лишь немного пробежаться по элементам, пока Shell Sort будет выполнять свой алгоритм полностью