Shell Sort

**Сортировка Шелла** (англ. *Shell sort*) — алгоритм сортировки, являющийся усовершенствованным вариантом сортировки вставками. Идея метода Шелла состоит в сравнении элементов, стоящих не только рядом, но и на определённом расстоянии друг от друга. Иными словами — это сортировка вставками с предварительными «грубыми» проходами.

Сортировка Шелла была названа в честь её изобретателя — Дональда Шелла, который опубликовал этот алгоритм в 1959 году.

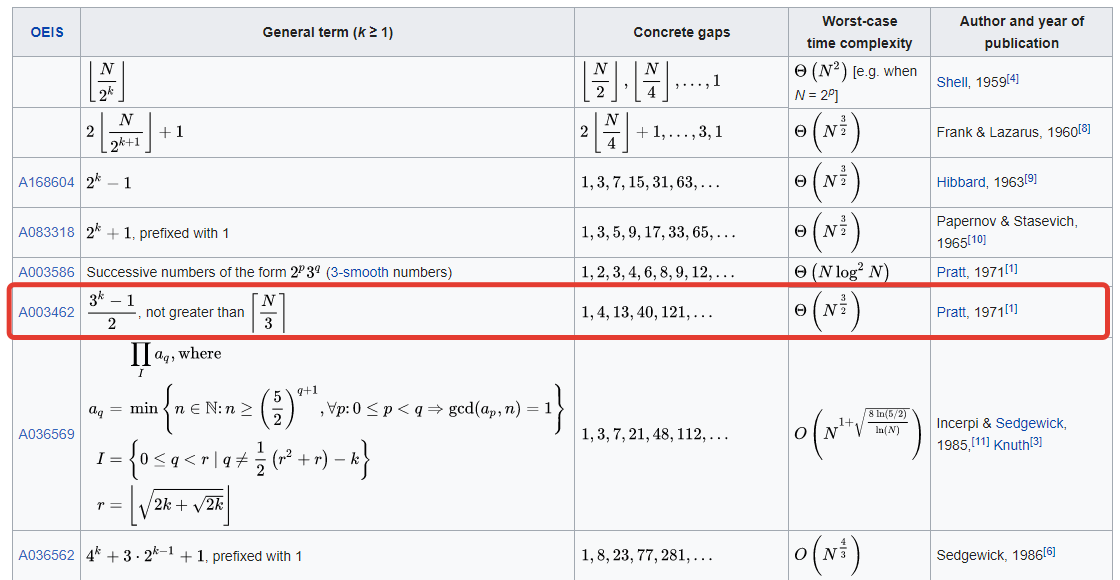
При сортировке Шелла сначала сравниваются и сортируются между собой значения, стоящие один от другого на некотором расстоянии d. После этого процедура повторяется для некоторых меньших значений d, а завершается сортировка Шелла упорядочиванием элементов при d=1 (то есть обычной сортировкой вставками). Эффективность сортировки Шелла в определённых случаях обеспечивается тем, что элементы «быстрее» встают на свои места (в простых методах сортировки, например, пузырьковой, каждая перестановка двух элементов уменьшает количество инверсий в списке максимум на 1, а при сортировке Шелла это число может быть больше).

Невзирая на то, что сортировка Шелла во многих случаях медленнее, чем быстрая сортировка, она имеет ряд преимуществ:

* отсутствие потребности в памяти под стек;
* отсутствие деградации при неудачных наборах данных — быстрая сортировка легко деградирует до O(n²), что хуже, чем худшее гарантированное время для сортировки Шелла.

Среднее время работы алгоритма зависит от длин промежутков — **d**, на которых будут   
находиться сортируемые элементы исходного массива ёмкостью **N** на каждом шаге алгоритма. Существует несколько подходов к выбору этих значений: 

* первоначально используемая Шеллом последовательность длин промежутков: https://www.pictshare.net/u1940wrcqz.pngв худшем случае, сложность алгоритма составит **O(N^2)**
* предложенная Хиббардом последовательность: все значенияhttps://www.pictshare.net/46ny6h49fq.png ; такая последовательность шагов приводит к алгоритму сложностью https://www.pictshare.net/6pohbbqqnd.png;
* эмпирическая последовательность Марцина Циура (последовательность [A102549](https://oeis.org/A102549) в [OEIS](https://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D1%86%D0%B8%D0%BA%D0%BB%D0%BE%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F_%D1%86%D0%B5%D0%BB%D0%BE%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9)): {\displaystyle d\in \left\{1,4,10,23,57,132,301,701,1750\right\}}{ 1, 4, 10, 23, 57, 132, 301, 701} является одной из лучших для сортировки массива ёмкостью приблизительно до 4000 элементов.



…

Таблицы графики полученных значений количества итераций и времени работы в зависимости от размера и характера данных можно найти по пути **…\ShellSort\files\output\*.xls**

Список использованной литературы:

* <https://habrahabr.ru/post/204968/>
* <https://habrahabr.ru/post/133996/>
* <https://en.wikipedia.org/wiki/Shellsort>
* <https://ru.wikipedia.org/wiki/Сортировка_Шелла>
* <http://www.javacodex.com/Sorting/Shell-Sort>

Код сортировки:

**public** **static** **void** sort(**int**[] array) {

**int** inner, outer;

**int** temp;

**int** h = 1;

**while** (h <= array.length / 3) {

h = h \* 3 + 1;

}

**while** (h > 0) {

**for** (outer = h; outer < array.length; outer++) {

temp = array[outer];

inner = outer;

**while** (inner > h - 1 && array[inner - h] >= temp) {

array[inner] = array[inner - h];

inner -= h;

}

array[inner] = temp;

}

h = (h - 1) / 3;

}

}

Код генерации входных файлов для сортировки:

**public void** generate(String fileName, **int** arraysAmount) {  
 **try** (BufferedWriter writer = **new** BufferedWriter(**new** FileWriter(**new** File(**path** + **"\\files\\"** + fileName + **".txt"**), **false**))) {  
 **int** dose = 0;  
 **int** t = 10000 / arraysAmount;  
 **for** (**int** j = 0; j < arraysAmount; j++) {  
 String text = **""**;  
 **for** (**int** i = 0; i < 100 + dose; i++) {  
 text += String.*valueOf*(**random**.nextInt(Integer.***MAX\_VALUE***)) + **" "**;  
 }  
 dose += t;  
 writer.write(text);  
 System.***out***.println(**"Файл сгенерирован на "** + (dose / arraysAmount) + **"%"**);  
 **if** (j == arraysAmount - 1) **break**;  
 writer.newLine();  
 }  
 writer.flush();  
 System.***out***.println(**"Файл сгенерирован"**);  
 } **catch** (IOException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
  
}

Все входные данные можно найти по пути: **…\ShellSort\files\file\*.txt**

