Оглавление

[Сортировка вставками(Insertion sort) 2](#_Toc511063555)

[Сортировка вставками 2](#_Toc511063556)

[Описание 2](#_Toc511063557)

[Анализ наихудшего случая 2](#_Toc511063558)

[Анализ среднего случая 3](#_Toc511063559)

[Временная сложность 3](#_Toc511063560)

[Сортировка Шелла (Shell Sort) 4](#_Toc511063561)

[Сортировка Шелла 4](#_Toc511063562)

[Описание 4](#_Toc511063563)

[Выбор диапазона поиска 4](#_Toc511063564)

[Анализ алгоритма 5](#_Toc511063565)

[Временная сложность 5](#_Toc511063566)

[Сравнение сортировок: 5](#_Toc511063567)

[Зависимость количества итераций от размерности массива 6](#_Toc511063568)

[Зависимость времени работы алгоритма (в мил. Сек) от размерности массива 6](#_Toc511063569)

[Минусы и плюсы сортировки 7](#_Toc511063570)

[Сортировка вставками 7](#_Toc511063571)

[Литература 7](#_Toc511063572)

[Статьи 7](#_Toc511063573)

**Отчет по алгоритмам сортировок**

**второй семестровой работы второго семестра**

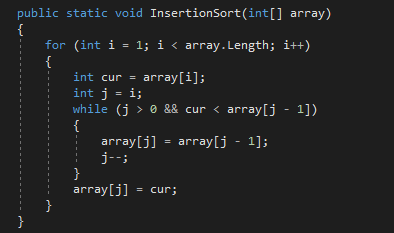
**по предмету Алгоритмы и Структуры Данных**

**Отчет выполнила студентка 1 курса группы 11-708**

**Сизова Дарья**

# Сортировка вставками(Insertion sort)

Сортировка вставками (*Insertion sort*) — алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.



## Описание

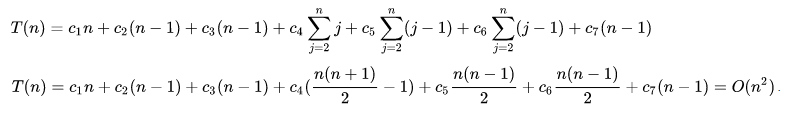
Сортировка вставками (*Insertion Sort*) — это простой алгоритм сортировки. Суть его заключается в том что, на каждом шаге алгоритма мы берем один из элементов массива, находим позицию для вставки и вставляем. Стоит отметить что массив из 1-го элемента считается отсортированным.

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

Данный алгоритм можно ускорить при помощи использования бинарного поиска для нахождения места текущему элементу в отсортированной части. Проблема с долгим сдвигом массива вправо решается при помощи смены указателей.

## Анализ наихудшего случая

Наихудшим случаем является массив, отсортированный в порядке, обратном нужному. При этом каждый новый элемент сравнивается со всеми в отсортированной последовательности. Это означает, что все внутренние циклы состоят из j итераций, то есть {\displaystyle t\_{j}=j}t(j) = j для всех {\displaystyle j}. Тогда время работы алгоритма составит:

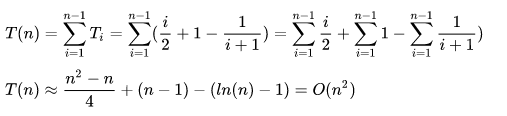


## Анализ среднего случая

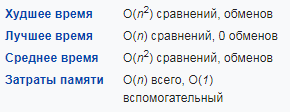
Для анализа среднего случая нужно посчитать среднее число сравнений, необходимых для определения положения очередного элемента. При добавлении нового элемента потребуется, как минимум, одно сравнение, даже если этот элемент оказался в правильной позиции. {\displaystyle i}i-й добавляемый элемент может занимать одно из {\displaystyle i+1}i+1 положений. Предполагая случайные входные данные, новый элемент равновероятно может оказаться в любой позиции. Среднее число сравнений для вставки {\displaystyle i}i-го элемента:



Для оценки среднего времени работы для n элементов нужно просуммировать

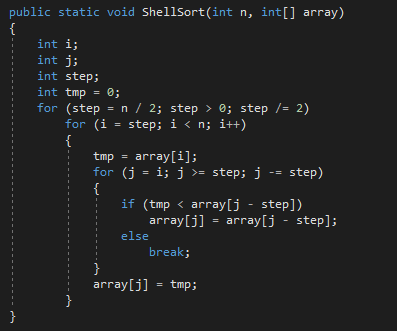


## Временная сложность



# Сортировка Шелла (Shell Sort)

Сортировка Шелла ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *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.



## Описание

При сортировке Шелла сначала сравниваются и сортируются между собой значения, стоящие один от другого на некотором расстоянии {\displaystyle d}k. После этого процедура повторяется для некоторых меньших значений {\displaystyle d}k, а завершается сортировка Шелла упорядочиванием элементов при k=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 инверсии в списке максимум на 1, а при сортировке Шелла это число может быть больше).

## Выбор диапазона поиска

Есть огромное количество методов выбора числа D:

1. Самый просто пример это D = n / 2, D2 = D /2 … Dn =1 . В худшем случае сложность алгоритма O(n) = N ^ 2
2. Предложение Хиббарда: проверить на всем N ^i — 1<= N. В таком случае сложность алгоритма O(n) = n ^ (3/2)

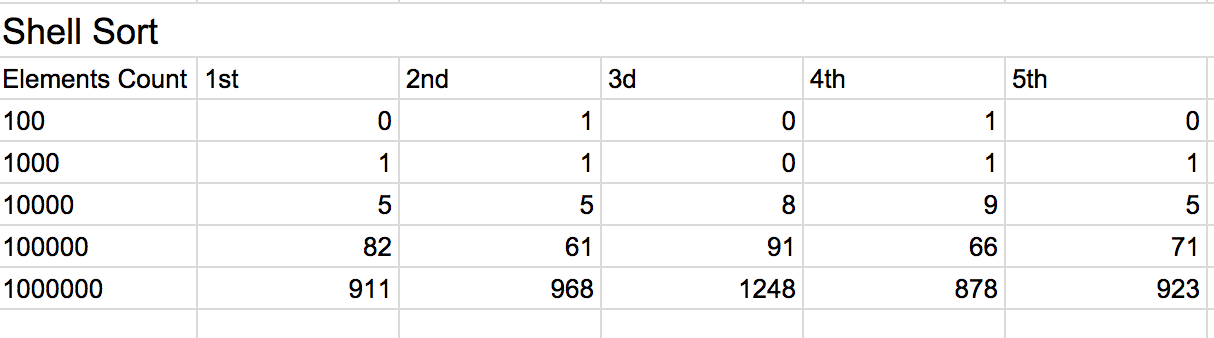
Анализ алгоритма

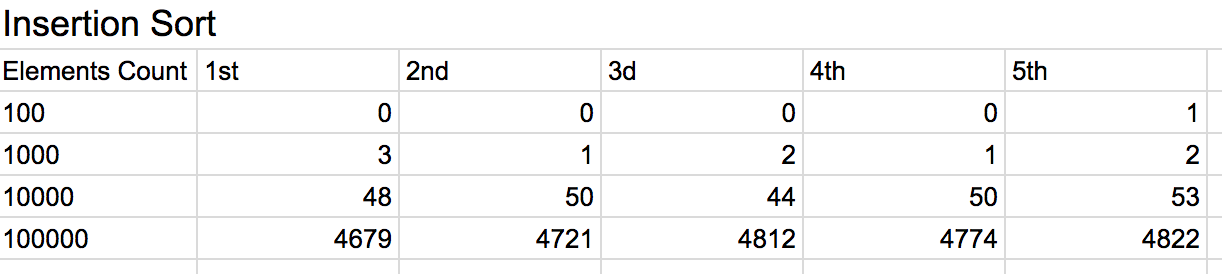
Анализ этого алгоритма - сложная математическая задача, у которой до сих пор нет полного решения. В настоящий момент неизвестно, какая последовательность расстояний даёт наилучший результат, но известно, что расстояния не должны быть кратными друг другу. Это необходимо, чтобы очередной проход не объединял две последовательности, до того никак не пересекавшиеся. На самом деле желательно, чтобы последовательности пересекались как можно чаще. Это связано со следующей теоремой: если последовательность после ki сортировке подвергается ki+1 сортировке, то она при этом остаётся ki сортированной.  
Кнут предлагает в качестве последовательно уменьшающихся расстояний использовать одну из следующих последовательностей (приведены в обратном порядке): 1,4,13,40,121,…, где hi-1=3\*hi+1 или 1,3,7,15,31,…, где hi-1=2\*hi+1. В последнем случае математическое исследование показывает, что при сортировке N элементов алгоритмом Шелла затраты пропорциональны N1.2. Хотя это гораздо лучше, чем N2, мы не будем углубляться в этот метод, так как есть ещё более быстрые алгоритмы.

## Временная сложность



# Сравнение сортировок:





## Зависимость количества итераций от размерности массива

## Зависимость времени работы алгоритма (в мил. Сек) от размерности массива

## Минусы и плюсы сортировки

Из минусов:

* Во многих случаях сортировка Шелла медленнее, чем быстрая сортировка

Но она также 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.

Сортировка вставками наиболее эффективна, когда массив уже частично отсортирован и когда элементов массива не много. Если же элементов меньше 10, то данный алгоритм является лучшим.

# Литература

* Н. Вирт **Алгоритмы и структуры данных** — Невский Диалект, 2008. — 352 с. — ISBN 978-5-7940-0065-8
* *Д. Кнут*. Искусство программирования. Том 3. Сортировка и поиск, 2-е изд. Гл. 5.2.1.

Статьи

* <https://habrahabr.ru/post/104697/>
* <https://habrahabr.ru/post/181271/>
* https://habrahabr.ru/post/133996/