**TALLINNA TÖÖSTUSHARIDUSKESKUS**

**„Алгоритмы сортировки“  
Referaat**

**Õpilane: M. Sidorov  
Õpetaja: M. Oleinik**

**Tallinn  
2020**

# Вступление[[1]](#footnote-1)

На эту тему написано уже немало статей. Реферат посвящен тому, как работают все алгоритмы. Если говорить о самом программировании, то иногда могут возникнуть совершенно неожиданные трудности. Однако не менее трудно решить, какие именно алгоритмы пригодятся в данный момент, ведь существует множество алгоритмов сортировки, поэтому здесь описаны некоторые из них.

# Описание основных алгоритмов сортировок

Первый источник литературы: Объясняем известные алгоритмы сортировки на пальцах https://proglib.io/p/sort-algorithms/

## Самые известные алгоритмы сортировки:

### Сортировка пузырьком / Bubble Sort

Один из самых известных алгоритмов [[2]](#footnote-2)сортировки, реализовать который часто просят на технических собеседованиях. В этом алгоритме массив [[3]](#footnote-3)перебирается раз за разом, и каждое следующее значение сравнивается с предыдущим. После первого прохода по массиву самое большое число окажется в конце, а когда первый элемент будет не с чем сравнить, массив будет отсортирован по возрастанию.

**Визуализация: (чтобы увидеть, как работает сортировка по пунктам, перейдите на источник:** Визуализация алгоритмов сортировки, URL https://visualgo.net/en/sorting (11.04.2020)**)**

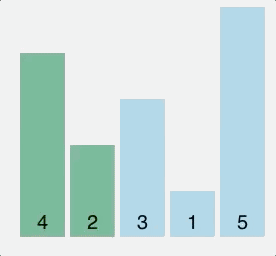


Рисунок 1 "Визуализация сортировки пузырьком"

### 

### Сортировка выбором / Selection Sort

Основная мысль этого метода заключается в том, чтобы создать отсортированную последовательность, присоединяя к ней один элемент за другим в правильном порядке.

Алгоритм сортировки выбором состоит из нескольких последовательных шагов. На каждом шаге сортировки текущий элемент массива меняется местами с элементом с наименьшим значением. Таким образом, получается массив значений, отсортированный по возрастанию.

**Визуализация: (чтобы увидеть, как работает сортировка по пунктам, перейдите на источник:** Визуализация алгоритмов сортировки, URL https://visualgo.net/en/sorting (11.04.2020)**)**

****

Рисунок 2 "Визуализация сортировки выбором"

### 

### Сортировка вставками / Insertion Sort

При сортировке вставками, массив перебирается последовательно. Каждый следующий рассматриваемый элемент размещается так, чтобы оказаться между ближайшим минимальным элементом и ближайшим максимальным.

**Визуализация: (чтобы увидеть, как работает сортировка по пунктам, перейдите на источник:** Визуализация алгоритмов сортировки, URL https://visualgo.net/en/sorting (11.04.2020)**)**



Рисунок 3 "Визаулизация сортировки вставками"

### 

### Быстрая сортировка / Quick Sort

Один из известных и быстрых алгоритмов сортировки, разработанный в МГУ английским информатиком Чарльзом Хоаром. Его часто называет quicksort или просто qsort – по названию стандартной библиотеки языка C. Является существенно переработанной версией пузырьковой сортировки.

### Сортировка слиянием / Merge Sort

Алгоритм сортировки слиянием помогает эффективно упорядочивать списки. Основная задача разбивается на подзадачи меньшего размера, каждая из которых решается отдельно. Затем их решения комбинируются, а результатом их слияния будет решение основной задачи.

### Время выполнения алгоритмов сортировки

Таблица 1 "Время выполнения алгоритмов сортировки"

|  |  |  |  |
| --- | --- | --- | --- |
| Алгоритм сортировки | Лучшее время выполнения | Среднее время выполнения | Худшее время выполнения |
| Bubble Sort | Ω(n) | θ(n2) | O(n2) |
| Selection Sort | Ω(n2) | θ(n2) | O(n2) |
| Insertion Sort | Ω(n) | θ(n2) | O(n2) |
| Quick Sort | Ω(nlogn) | θ(nlogn) | O(n2) |
| Merge Sort | Ω(nlogn) | θ(nlogn) | O(nlogn) |

## Другие алгоритмы сортировки

### Шейкерная сортировка / Shaker sort

Заметим, что сортировка пузырьком работает медленно на тестах, в которых маленькие элементы стоят в конце (их еще называют «черепахами»). Такой элемент на каждом шаге алгоритма будет сдвигаться всего на одну позицию влево. Поэтому будем идти не только слева направо, но и справа налево. Будем поддерживать два указателя begin и end, обозначающих, какой отрезок массива еще не отсортирован. На очередной итерации при достижении end вычитаем из него единицу и движемся справа налево, аналогично, при достижении begin прибавляем единицу и двигаемся слева направо. Асимптотика у алгоритма такая же, как и у сортировки пузырьком, однако реальное время работы лучше.

### Сортировка расческой / 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).

### Сортировка Шелла / Shellsort

Используем ту же идею, что и сортировка с расческой, и применим к сортировке вставками. Зафиксируем некоторое расстояние. Тогда элементы массива разобьются на классы – в один класс попадают элементы, расстояние между которыми кратно зафиксированному расстоянию. Отсортируем сортировкой вставками каждый класс. В отличие от сортировки расческой, неизвестен оптимальный набор расстояний. Существует довольно много последовательностей с разными оценками. Последовательность Шелла – первый элемент равен длине массива, каждый следующий вдвое меньше предыдущего. Асимптотика в худшем случае – O(n2). Последовательность Хиббарда – 2n — 1, асимптотика в худшем случае – O(n1,5), последовательность Седжвика (формула нетривиальна, можете ее посмотреть по ссылке ниже) — O(n4/3), Пратта (все произведения степеней двойки и тройки) — O(nlog2n). Отмечу, что все эти последовательности нужно рассчитать только до размера массива и запускать от большего от меньшему (иначе получится просто сортировка вставками). Также я провел дополнительное исследование и протестировал разные последовательности вида si = a \* si — 1 + k \* si — 1 (отчасти это было навеяно эмпирической последовательностью Циура – одной из лучших последовательностей расстояний для небольшого количества элементов). Наилучшими оказались последовательности с коэффициентами a = 3, k = 1/3; a = 4, k = 1/4 и a = 4, k = -1/5.

### Сортировка деревом / Tree sort

Будем вставлять элементы в двоичное дерево поиска. После того, как все элементы вставлены достаточно обойти дерево в глубину и получить отсортированный массив. Если использовать сбалансированное дерево, например красно-черное, асимптотика будет равна O(nlogn) в худшем, среднем и лучшем случае. В реализации использован контейнер multiset.

### Гномья сортировка / Gnome sort

Алгоритм похож на сортировку вставками. Поддерживаем указатель на текущий элемент, если он больше предыдущего или он первый — смещаем указатель на позицию вправо, иначе меняем текущий и предыдущий элементы местами и смещаемся влево.

### Пирамидальная сортировка / Heapsort

Развитие идеи сортировки выбором. Воспользуемся структурой данных «куча» (или «пирамида», откуда и название алгоритма). Она позволяет получать минимум за O(1), добавляя элементы и извлекая минимум за O(logn). Таким образом, асимптотика O(nlogn) в худшем, среднем и лучшем случае. Реализовывал кучу я сам, хотя в С++ и есть контейнер priority\_queue, поскольку этот контейнер довольно медленный.

### Сортировка подсчетом / Counting sort

Создадим массив размера r – l, где l – минимальный, а r – максимальный элемент массива. После этого пройдем по массиву и подсчитаем количество вхождений каждого элемента. Теперь можно пройти по массиву значений и выписать каждое число столько раз, сколько нужно. Асимптотика – O(n + r — l). Можно модифицировать этот алгоритм, чтобы он стал стабильным: для этого определим место, где должно стоять очередное число (это просто префиксные суммы в массиве значений) и будем идти по исходному массиву слева направо, ставя элемент на правильное место и увеличивая позицию на 1. Эта сортировка не тестировалась, поскольку большинство тестов содержало достаточно большие числа, не позволяющие создать массив требуемого размера. Однако она, тем не менее, пригодилась.

### Блочная сортировка / Bucket sort

(также известна как корзинная и карманная сортировка). Пусть l – минимальный, а r – максимальный элемент массива. Разобьем элементы на блоки, в первом будут элементы от l до l + k, во втором – от l + k до l + 2k и т.д., где k = (r – l) / количество блоков. В общем-то, если количество блоков равно двум, то данный алгоритм превращается в разновидность быстрой сортировки. Асимптотика этого алгоритма неясна, время работы зависит и от входных данных, и от количества блоков. Утверждается, что на удачных данных время работы линейно. Реализация этого алгоритма оказалась одной из самых трудных задач. Можно сделать это так: просто создавать новые массивы, рекурсивно их сортировать и склеивать. Однако такой подход все же довольно медленный и меня не устроил.

В эффективной реализации используется несколько идей:

* Не будем создавать новых массивов. Для этого воспользуемся техникой сортировки подсчетом – подсчитаем количество элементов в каждом блоке, префиксные суммы и, таким образом, позицию каждого элемента в массиве.
* Не будем запускаться из пустых блоков. Занесем индексы непустых блоков в отдельный массив и запустимся только от них.
* Проверим, отсортирован ли массив. Это не ухудшит время работы, так как все равно нужно сделать проход с целью нахождения минимума и максимума, однако позволит алгоритму ускориться на частично отсортированных данных, ведь элементы вставляются в новые блоки в том же порядке, что и в исходном массиве.
* Поскольку алгоритм получился довольно громоздким, при небольшом количестве элементов он крайне неэффективен. До такой степени, что переход на сортировку вставками ускоряет работу примерно в 10 раз.

Осталось только понять, какое количество блоков нужно выбрать. На рандомизированных тестах мне удалось получить следующую оценку: 1500 блоков для 107 элементов и 3000 для 108. Подобрать формулу [[4]](#footnote-4)не удалось – время работы ухудшалось в несколько раз.

# Заключение[[5]](#footnote-5)

Больше всего времени уходит на сноски, таблицы и схемы, потому что все надо строить верно, а делать их долго. Также не все задания понятно описаны, чтобы понять, было бы неплохо поставить картинку с примером этого задания. В основном задания не сложные.

# Список использованной литературы

1. Описание алгоритмов сортировки и сравнение их производительности, URL <https://habr.com/ru/post/335920/> (09.04.2020)
2. Объясняем известные алгоритмы сортировки на пальцах, URL <https://proglib.io/p/sort-algorithms/> (09.04.2020)
3. Визуализация алгоритмов сортировки, URL <https://visualgo.net/en/sorting> (11.04.2020)

# Приложения

[Таблица 1 "Время выполнения алгоритмов сортировки" 5](#_Toc37516596)

# Содержание

[Вступление 2](#_Toc37516609)

[Описание основных алгоритмов сортировок 3](#_Toc37516610)

[1. Самые известные алгоритмы сортировки: 3](#_Toc37516611)

[1.1. Сортировка пузырьком / Bubble Sort 3](#_Toc37516612)

[1.2. Сортировка выбором / Selection Sort 3](#_Toc37516613)

[1.3. Сортировка вставками / Insertion Sort 4](#_Toc37516614)

[1.4. Быстрая сортировка / Quick Sort 5](#_Toc37516615)

[1.5. Сортировка слиянием / Merge Sort 5](#_Toc37516616)

[1.6. Время выполнения алгоритмов сортировки 5](#_Toc37516617)

[2. Другие алгоритмы сортировки 6](#_Toc37516618)

[2.1. Шейкерная сортировка / Shaker sort 6](#_Toc37516619)

[2.2. Сортировка расческой / Comb sort 6](#_Toc37516620)

[2.3. Сортировка Шелла / Shellsort 6](#_Toc37516621)

[2.4. Сортировка деревом / Tree sort 7](#_Toc37516622)

[2.5. Гномья сортировка / Gnome sort 7](#_Toc37516623)

[2.6. Пирамидальная сортировка / Heapsort 7](#_Toc37516624)

[2.7. Сортировка подсчетом / Counting sort 7](#_Toc37516625)

[2.8. Блочная сортировка / Bucket sort 8](#_Toc37516626)

[Заключение 10](#_Toc37516627)

[Список использованной литературы 11](#_Toc37516628)

[Приложения 12](#_Toc37516629)

[Содержание 13](#_Toc37516630)

[Список иллюстраций 14](#_Toc37516631)

# Список иллюстраций

[Рисунок 1 "Визуализация сортировки пузырьком" 3](#_Toc37516605)

[Рисунок 2 "Визуализация сортировки выбором" 4](#_Toc37516606)

[Рисунок 3 "Визаулизация сортировки вставками" 4](#_Toc37516607)

1. Вступление — начальная часть чего-н., введение к чему-н. [↑](#footnote-ref-1)
2. Алгоритм— конечная совокупность точно заданных правил решения произвольного класса задач или набор инструкций, описывающих порядок действий исполнителя для решения некоторой задачи. [↑](#footnote-ref-2)
3. Массив — структура данных, хранящая набор значений (элементов массива), идентифицируемых по индексу или набору индексов, принимающих целые (или приводимые к целым) значения из некоторого заданного непрерывного диапазона. [↑](#footnote-ref-3)
4. Формула — правило, предписание. [↑](#footnote-ref-4)
5. Заключение — утверждение, сделанное на основе вывода. [↑](#footnote-ref-5)