# Как работает пузырьковая сортировка

**Самый простой, но не самый эффективный алгоритм.**

**Контекст:**мы начали говорить об алгоритмах сортировки — смысл в том, что есть алгоритмы, которые помогают упорядочить разные неорганизованные данные. И данные, и алгоритмы могут быть разными, поэтому разработчики разбираются в этих алгоритмах, чтобы подобрать лучшее.

**Что разбираем:** пузырьковую сортировку — самый простой способ отсортировать массив, написав всего 6 строк кода. Она самая простая, но не самая эффективная.

**Зачем это нужно:**пузырьковая сортировка проще остальных, поэтому изучение всех сортировок лучше начинать именно с неё — так будет легче понять, что происходит в остальных алгоритмах. А ещё пузырьковая сортировка применяется внутри других, более эффективных алгоритмов.

## Принцип работы

На каждом шаге мы находим наибольший элемент из двух соседних и ставим этот элемент в конец пары. Получается, что при каждом прогоне цикла большие элементы будут всплывать к концу массива, как пузырьки воздуха — отсюда и название.

Алгоритм выглядит так:

1. Берём самый первый элемент массива и сравниваем его со вторым. Если первый больше второго — меняем их местами с первым, если нет — ничего не делаем.
2. Затем берём второй элемент массива и сравниваем его со следующим — третьим. Если второй больше третьего — меняем их местами, если нет — ничего не делаем.
3. Проходим так до предпоследнего элемента, сравниваем его с последним и ставим наибольший из них в конец массива. Всё, мы нашли самое большое число в массиве и поставили его на своё место.
4. Возвращаемся в начало алгоритма и делаем всё снова точно так же, начиная с первого и второго элемента. Только теперь даём себе задание не проверять последний элемент — мы знаем, что теперь в конце массива самый большой элемент.
5. Когда закончим очередной проход — уменьшаем значение финальной позиции, до которой проверяем, и снова начинаем сначала.
6. Так делаем до тех пор, пока у нас не останется один элемент.

## Эскиз работы

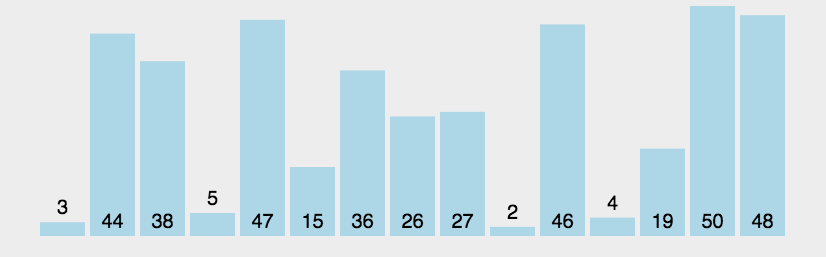
Возьмём массив из шести чисел и сделаем первый прогон:



После первого прогона у нас «всплыл» самый большой элемент массива (число 11) и отправился в конец. После второго прогона на предпоследнем месте будет стоять число 9 — оно «всплывёт» наверх как самое большое число из оставшихся, потому что последние элементы мы не проверяем.

Так, шаг за шагом, мы пройдём все прогоны, каждый раз отправляя ближе к концу массива самый большой на этом этапе элемент. В итоге на последнем прогоне мы сравним первый и второй элементы, найдём из них наибольший и так получим полностью отсортированный массив.

Работает это примерно так:



## Эффективность работы

Этот алгоритм считается учебным и в чистом виде на практике почти не применяется. Дело в том, что среднее количество проверок и перестановок в массиве — это количество элементов в квадрате. Например, для массива из 10 элементов потребуется 100 проверок, а для массива из 100 элементов — уже в сто раз больше, 10 000 проверок.

Получается, что если у нас в массиве будет 10 тысяч элементов (а это не слишком большой массив с точки зрения реальных ИТ-задач), то для его сортировки понадобится 100 миллионов проверок — на это уйдёт какое-то время. А что, если сортировать нужно несколько раз в секунду?

В программировании эффективность работы алгоритма в зависимости от количества элементов n обозначают так: O (n). В нашем случае эффективность работы пузырьковой сортировки равна O (n²). По сравнению с другими алгоритмами это очень большой показатель (чем он больше — тем больше времени нужно на сортировку).