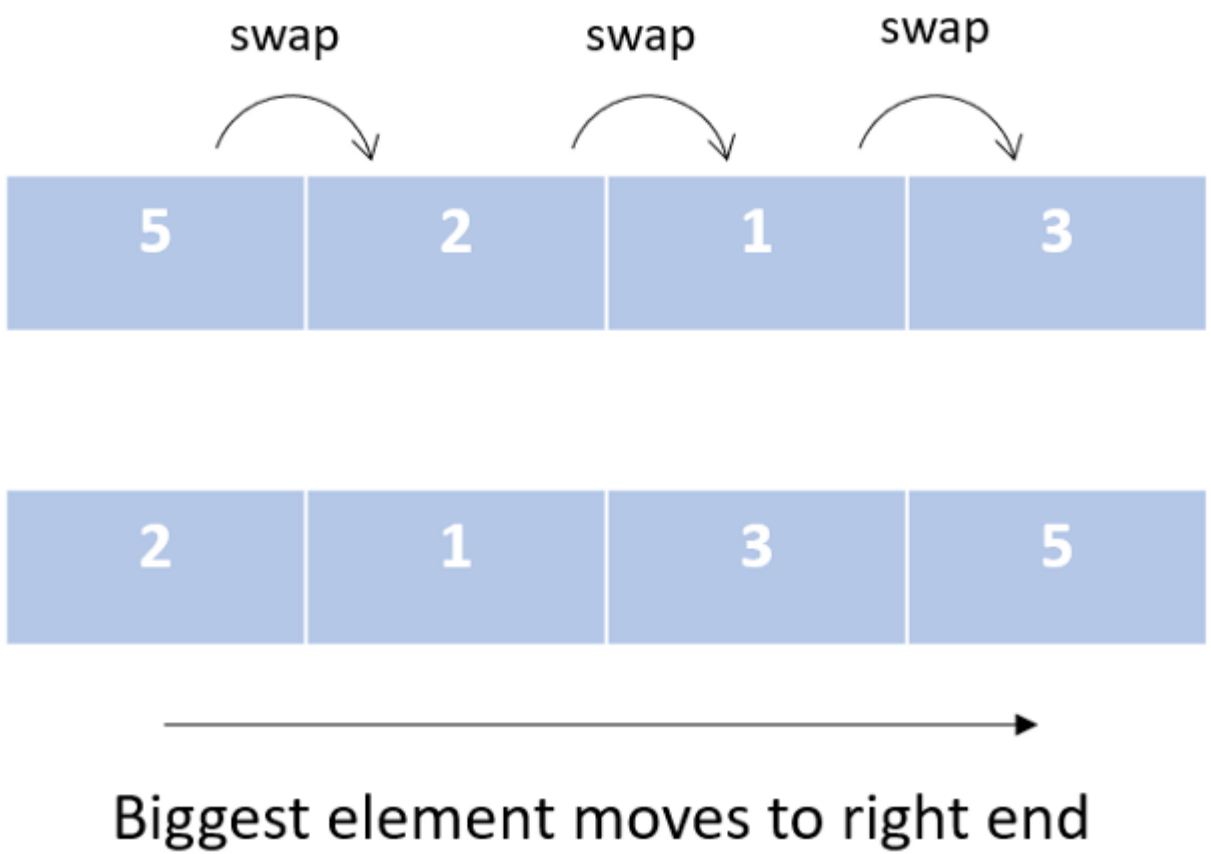
Алгоритмы сортировок

- Сортировка означает размещение элементов в определенном порядке. Этот конкретный порядок определяется свойством сравнения элементов.
- В случае целых чисел мы говорим, что сначала идет меньшее число, а потом — большее.

Сортировка методом пузырька
Bubble sort

При этом подходе осуществляется перебор по списку и сравнение соседних элементов. Они меняются местами в том случае, если порядок неправильный. Так продолжается до тех пор, пока все элементы не расположатся в нужном порядке. Из-за большого количества повторений у пузырьковой сортировки его сложность в худшем случае — O(n^2).

Сортировка методом пузырька
Bubble sort



Сортировка методом пузырька
Bubble sort

6 5 3 1 8 7 2 4

```python
def bubble_sort(nums):
    # Устанавливаем swapped в True, чтобы цикл запустился хотя бы
один раз
    swapped = True
    while swapped:
        swapped = False
        for i in range(len(nums) - 1):
            if nums[i] > nums[i + 1]:
                # Меняем элементы
                nums[i], nums[i + 1] = nums[i + 1], nums[i]
                # Устанавливаем swapped в True для следующей
итерации
                swapped = True

# Проверяем, что оно работает
random_list_of_nums = [5, 2, 1, 8, 4]
bubble_sort(random_list_of_nums)
print(random_list_of_nums)
```

Сортировка выбором

Этот алгоритм сегментирует список на две части: отсортированную и неотсортированную. Наименьший элемент удаляется из второго списка и добавляется в первый.
Затраты времени на сортировку выборкой в среднем составляют $O(n^2)$, где n — количество элементов списка.

Сортировка выбором

На практике не нужно создавать новый список для отсортированных элементов. В качестве него используется крайняя левая часть списка. Находится наименьший элемент и меняется с первым местами. Теперь, когда нам известно, что первый элемент списка отсортирован, находим наименьший элемент из оставшихся и меняем местами со вторым. Повторяем это до тех пор, пока не останется последний элемент в списке.

Сортировка выбором

	8
	5
	2
	6
	9
	3
	1
	4
	0
	7

```
def selection_sort(nums):
    # Значение i соответствует кол-ву отсортированных значений
    for i in range(len(nums)):
        # Исходно считаем наименьшим первый элемент
        lowest_value_index = i
        # Этот цикл перебирает несортированные элементы
        for j in range(i + 1, len(nums)):
            if nums[j] < nums[lowest_value_index]:
                lowest_value_index = j
        # Самый маленький элемент меняем с первым в списке
        nums[i], nums[lowest_value_index] =
nums[lowest_value_index], nums[i]

# Проверяем, что оно работает
random_list_of_nums = [12, 8, 3, 20, 11]
selection_sort(random_list_of_nums)
print(random_list_of_nums)
```

Сортировка вставками

Этот алгоритм сегментирует список на две части: отсортированную и неотсортированную. Алгоритм перебирает второй сегмент и вставляет текущий элемент в правильную позицию первого сегмента.

Сортировка вставками

Предполагается, что первый элемент списка отсортирован. Переходим к следующему элементу, обозначим его x. Если x больше первого, оставляем его на своём месте. Если он меньше, копируем его на вторую позицию, а x устанавливаем как первый элемент.

Переходя к другим элементам несортированного сегмента, перемещаем более крупные элементы в отсортированном сегменте вверх по списку, пока не встретим элемент меньше x или не дойдём до конца списка. В первом случае x помещается на правильную позицию.

Сортировка вставками

6 5 3 1 8 7 2 4

```
def insertion_sort(nums):
    # Сортировку начинаем со второго элемента, т.к. считается, что
первый элемент уже отсортирован
    for i in range(1, len(nums)):
        item_to_insert = nums[i]
        # Сохраняем ссылку на индекс предыдущего элемента
        j = i - 1
        # Элементы отсортированного сегмента перемещаем вперёд, если они
больше
        # элемента для вставки
        while j >= 0 and nums[j] > item_to_insert:
            nums[j + 1] = nums[j]
            j -= 1
        # Вставляем элемент
        nums[j + 1] = item_to_insert

# Проверяем, что оно работает
random_list_of_nums = [9, 1, 15, 28, 6]
insertion_sort(random_list_of_nums)
print(random_list_of_nums)
```