CountingSort
Создается массив подсчета (countArray) размером (maxNumber + 1), где maxNumber - это максимальное число во входном массиве. Этот массив будет использоваться для подсчета количества каждого уникального элемента во входном массиве.

2. Инициализируются все элементы массива подсчета нулями.

3. Происходит первый проход по входному массиву, в котором для каждого элемента подсчитывается количество его вхождений. То есть, countArray[array[i]] будет содержать количество вхождений числа array[i].

4. Далее, происходит накопление суммы в массиве подсчета, чтобы получить информацию о количестве элементов, которые меньше или равны текущему элементу.

5. После этого происходит второй проход по входному массиву. Здесь элементы размещаются в выходном массиве в соответствии с их порядком, определяемым с помощью массива подсчета.

6. В конце работы алгоритма значения выходного массива копируются обратно во входной массив.

Этот алгоритм позволяет эффективно сортировать массив целых чисел, если известен диапазон чисел, которые могут встречаться в данном массиве.
Сложность по времени: n+k, k - разность между самым малым и самым большим членом массива
stable
not in place

BubbleSort
1. Внешний цикл проходит по всем элементам массива, кроме последнего, потому что после каждой итерации внутреннего цикла самый большой элемент "всплывает" на правильную позицию.
2. Внутренний цикл сравнивает каждую пару соседних элементов и, если они стоят в неправильном порядке, меняет их местами с помощью функции `swap`.
3. Таким образом, на каждой итерации внешнего цикла самый большой элемент "всплывает" на правильную позицию, алгоритм получает название "пузырьковой сортировки" из-за того, что большие элементы "всплывают" наверх, как пузырьки в воде.
Сложность по времени: Лучший случай - n, худший и др n^2
stable
in place

HeapSort
Ваша функция `swap` используется для обмена значений двух переменных.

2. Функция `heapify` принимает массив `arr[]`, его размер `N` и индекс `i`, который представляет текущий узел для "кучевой" операции. Функция превращает заданное поддерево с корнем в узле `i` во валидную "кучу".

3. Функция `heapSort` начинает сортировку, преобразовывая входной массив в "кучу" и затем поочередно извлекая наибольший элемент из "кучи" и уменьшая размер "кучи".

Вот как это работает:
- Сначала строится "куча" из входного массива, вызывая функцию `heapify`.
- Затем происходит сортировка путем поочередного извлечения наибольшего элемента, помещения его в конец массива, уменьшения размера "кучи", и затем восстановления "кучевого" свойства с помощью функции `heapify`.
nlogn для всех случаев
not stable
in place

MergeSort
nlogn
stable
RadixSort
nk
stable
SelectionSort
n^2
not stable
InsertionSort
best - n, average and worst - n^2
stable
Odd-EvenSort
n best, else - n^2

QuickSort
nlogn - best and average, n^2 - worst
not stable