

* Алгоритмы, основанные на сравнениях: ***Insertion Sort, Bubble Sort, Selection Sort, Shell Sort, Quick Sort, Merge Sort, Heap Sort и другие***
* Алгоритмы, не основанные на сравнениях: ***Counting Sort, Radix Sort*** — используют структуру ключа

1. Сортировка подсчетом

* Целочисленная сортировка
* Без использования операции сравнения
* Линейная сложность – Θ(n + k)
* Не рекомендуется для сортировки широких диапазонов чисел

Шаги:

1. Определить диапазон сортировки: обойти исходный массив arr, найти максимальный элемент max

2. Создать вспомогательный массив count размера k + 1 (первый индекс – 0, последний – k), заполнить его нулями

3. Обойти исходный массив arr, записывая в count информацию о количестве вхождений каждого элемента в arr: count[arr[i]]++

4. Начиная с индекса 1, увеличить каждый count[i] на count[i – 1] – теперь в каждом count[i] записано число элементов, не превышающих i

5. Используя массив count, заполнить результирующий массив output элементами исходного

1. Быстрая сортировка

* Один из наиболее распространенных на практике алгоритмов
* Рекурсивный подход на основе метода декомпозиции
* Сложность в среднем случае – Θ(nlogn), в худшем – O(n^2 )(качестве опорного элемента выбирает максимальный в подмассиве)

Шаги:

1. Выбрать в массиве опорный элемент (pivot)

2. Разбить массив на две части относительно опорного элемента: положить слева все элементы не больше, а справа – не меньше опорного

3. Повторить шаги 1 и 2 рекурсивно для левой и правой частей

1. Сортировка вставками

* Относится к простейшим алгоритмам сортировки на основе сравнений
* Основан на методе уменьшения задачи
* Сложность в среднем и худшем случаях — O(n^2 ). В худшем случае цикл while всегда доходит до первого элемента массива — на вход поступил массив, упорядоченный по убыванию

Шаги:

1. Двигаемся по массиву слева направо со 2-го до n-го элемента.

2. Сравниваем элемент с элементами перед ним и, если правый элемент меньше, вставляем его в правильное положение.

1. Сортировка слиянием

* Асимптотически оптимальный алгоритм сортировки сравнением
* Рекурсивный подход на основе метода декомпозиции
* Сложность в любом случае – Θ(n logn)

Шаги:

1. Рекурсивное разбиение исходного массива из n элементов на n частей по 1 элементу – фаза разбиения

2. Слияние упорядоченных подмассивов в один – фаза слияния

1. Пирамидальная сортировка (Heap)

* Сортировка на базе бинарной кучи
* Требует O(nlogn) времени, не требует дополнительной памяти

Бинарная куча (binary heap):

• Древовидная структура данных

• Полное бинарное дерево, в котором каждый элемент по ключу не меньше своих потомков

• Корень содержит максимальный элемент

(про max-heap, в min-heap – наоборот, не больше и минимальный)

Шаги:

1. Построить из исходного массива бинарную кучу вида max-heap (с максимальным элементом в корне)

2. Поочередно извлекая корень и восстанавливая кучу из оставшихся элементов, заполнять результирующий массив с конца

1. Поразрядная сортировка (Radix)

* В любом случае — O(n)

Сравнение производится поразрядно: сначала сравниваются значения одного крайнего разряда, и элементы группируются по результатам этого сравнения, затем сравниваются значения следующего разряда, соседнего, и элементы либо упорядочиваются по результатам сравнения значений этого разряда внутри образованных на предыдущем проходе групп, либо переупорядочиваются в целом, но сохраняя относительный порядок, достигнутый при предыдущей сортировке. Затем аналогично делается для следующего разряда, и так до конца.

1. Чётно-нечётная сортировка (Odd-even)

* Устойчивая
* Сложность в среднем и худшем случаях — O(n^2 )

Производится многократный прогон по массиву, соседние элементы сравниваются и, в случае необходимости, меняются местами. В отличие от пузырьковой сортировки шаг по массиву равен двум, а не единице.

Сначала элементы с нечётными индексами сравниваются/обмениваются с элементами с чётными индексами Затем элементы с чётными индексами сравниваются/обмениваются с соседними элементами с нечётными индексами. Затем снова нечётные сравниваются с чётными, потом снова чётные с нечётными и т.д.

1. Сортировка выбором

* В любом случае — O(n^2 ). Худший —когда приходится делать перестановку на каждой итерации

В неотсортированном подмассиве ищется локальный максимум (минимум).

Найденный максимум (минимум) меняется местами с последним (первым) элементом в подмассиве.

1. Сортировка пузырьком

* Сложность в среднем и худшем случаях — O(n^2 ). Худший — массив отсортирован в обратном порядке

Выполняется некоторое количество проходов по массиву — начиная от начала массива, перебираются пары соседних элементов массива. Если 1-й элемент пары больше 2-го, элементы переставляются