Критерии сортировки:

1. Принцип работы (сравнение) (гибридность)
2. Сложность (л/х/ср) (частота)
3. Память – исп. Памяти + in place (out)
4. Распараллеливание
5. Устойчивость (допустим есть два одинаковых элемента -> устойчива, если эти элементы идут в том же порядке в каком и были)
6. Адаптивность (на отсортированных или почти отсортированных массивах работает быстрее)

Простейшие (такие сортировки работают быстрее на маленьких данных, пока квадратичная функция меньше логарифмической):

- **Выбором** (проходим по всему массиву, находим минимальный элемент и вставляем его в нулевую ячейку, дальше делаем тоже самое с оставшимися)
  Принцип работы = я так понимаю, что сравнение
  Сложность (л/х/ср) = O(n^2)
  Память = O(1)(const), in place
  Распараллеливание = нет, нельзя
  Устойчивость = может быть как устойчивой, так и не устойчивой
  Адаптивность = нет

- **Пузырьком** (если след элемент больше данного, то меняем их местами)
  Принцип работы = я так понимаю, что сравнение
  Сложность (л/х/ср) = л O(n) \ x O(n^2) \ c O(n^2)
  Память = O(1)(const), in place
  Распараллеливание = нет, нельзя
  Устойчивость = может быть как устойчивой, так и не устойчивой (написать >=)
  Адаптивность = да

- **Вставками** (разбиваем наш на две части (1-ая тип уже отсортирована) и вставляем элементы из 2-ой части в первую на нужное место)
  Принцип работы = я так понимаю, что сравнение
  Сложность (л/х/ср) = л O(n) \ x O(n^2) \ c O(n^2)
  Память = O(1)(const), out place
  Распараллеливание = нет, нельзя
  Устойчивость = да
  Адаптивность = нет

Сортировка слиянием (разбиванием наш массив 2 части(пока не останется массив из одного элемента), слияние осуществляется сравнением двух массиво)

Принцип работы = я так понимаю, что сравнение
Сложность (л/х/ср) = O(n*logn), n – проход по массиву, logn – разбиение массива
Память = O(n), out place
Распараллеливание = да
Устойчивость = да
Адаптивность = нет

Быстрая сортировка (А)

1. Выбрать элемент из массива. Назовём его опорным.

2. Разбиение: перераспределение элементов в массиве таким образом, что элементы меньше опорного помещаются перед ним, а больше или равные после.

3. Рекурсивно применить первые два шага к двум подмассивам слева и справа от опорного элемента. Рекурсия не применяется к массиву, в котором только один элемент или отсутствуют элементы.

Принцип работы = сравнение

Сложность л и ср = N logN / х = N^2

Память = O(logN), in place

Распараллеливание = да (правые и левые подмассивы)

Устойчивость = нет

Адаптивность = да

Выбор опорного элемента – бери случайный/медиана, чаще всего медиана трех (first + middle + last) /.3

Сортировка подсчетом (Н)

Ограничим входные данные (числа от 0 до k), нужен вспомогательный массив размером k, далее проходим по основному массиву и увеличиваем соответствующий счетчик, далее проходимся по вспомогательному массиву и складываем текущий+предыдущий (получаем индексы, это куда нужно вставить элемент – 1)
Принцип работы = подсчетом
Сложность (л/х/ср) = л O(n), х O(k), O(n+k)
Память = O(n+k), in place
Распараллеливание = …
Устойчивость = да
Адаптивность = нет

Гибридные сортировки (на примере интроспективной) (А)

Контролируем максимальную глубину рекурсии, допустимую для алгоритма быстрой сортировки (например, можно ориентироваться на величину log n). Если глубина рекурсии достигала этой величины, то дальнейшее упорядочивание подмассива, от которого поступил тревожный сигнал, производится с помощью пирамидальной сортировки. Пирамидальная сортировка характерна тем, что у неё нет ни вырожденных, ни лучших наборов

данных, любые массивы она сортирует всегда с одинаковой временной сложностью - O(n log n).

Принцип работы = сравнение (гибридный метод)

Сложность л = N / ср и х = N logN

Память = O(N), in place

Распараллеливание = ХЗ

Устойчивость = нет

Адаптивность = да

Поразрядная сортировка (Н)

Создаем вспомогательный массив и резервируем там места под наши числа (так сначала для 1-ого разряда, потом для 2-ого разряда и тд)
Принцип работы = поразрядно
Сложность (л/х/ср) = х O(w * n), где w – количество бит, требуемых для хранения каждого ключа
Память = O(w + n), out place
Распараллеливание = …
Устойчивость = да
Адаптивность = нет

Тимсорт (А)

Делим массив на подмассивы размера min_run, которые сортируются сортировкой вставками. Потом все отсортированные подмассивы соединяются сортировкой слияния.

(слияние кастомное, вставка бинарная - норм тема) Лучший min_run по-научному лежит в [32;64]

Принцип работы = сравнение (гибридный метод)

Сложность л = N / ср и х = N logN

Память = O(N), out place

Распараллеливание = да (сортировку слияния можно распараллелить вроде же)

Устойчивость = да

Адаптивность = да

Зачем нужны сортировки, если есть ДДП?

- Не всегда храним в ДДП
- Обращение по индексу
- Кэш-память – меньше доп данных
- Есть сортировки без сравнения
- O(n+ nlog(n)) vs n*O(log(n)) + O(n)
- Константа
- Распараллеливание

Расстояния Левенштейна (А)