



Урок 18. Python: Быстрая сортировка

Часть 1. Материалы ученика

Программирование на Python и C++ · 9 класс

Урок 18 из 34 · 45 минут · Модуль «Сортировка»

Уровень: начинающий программист; Python 3.

Программа: <https://schoolotzyv.ru/blog/python-cpp-9-klass-programma>

Опорный элемент разделяет данные на меньшие, равные и большие. Равные значения нельзя терять или снова отправлять в ту же задачу без уменьшения размера.

Результат урока: Реализовать трёхчастную быструю сортировку и объяснить завершение на повторяющихся значениях и риск несбалансированных разбиений.

01. Проблемная ситуация

0–5 минут. Для [4,1,4,3,2] возьмите опорное значение 4. Разложите карточки в три группы <4, =4, >4, сохранив каждую карточку. Какие группы ещё надо сортировать?

02. Понятия и разобранный пример

5–14 минут: 4 минуты на понятия, 3 на пример, 2 на объяснение способа партнёру.

Три части вокруг опорного

В этом учебном варианте `pivot` — значение среднего по индексу элемента, а не медиана значений. Один проход раскладывает каждый элемент ровно в одну группу. Рекурсивно сортируются только `low` и `high`, а `equal` вставляется между ними. Поскольку хотя бы сам `pivot` попадает в `equal`, обе рекурсивные части короче исходной.

Для пустого или одноэлементного списка возвращаем копию. На всех одинаковых значениях рекурсия сразу получает две пустые части. Нельзя заменить `equal` единственным `[pivot]`: это уничтожит повторы. Здесь создаются новые списки; это не внутримассивная сортировка.

При сбалансированных частях время $O(n \log n)$, при неудачных последовательных разбиениях возможно $O(n^2)$. Средний индекс не гарантирует баланс значений. У этого простого рекурсивного варианта в худшем случае также растут глубина вызовов и память, вплоть до $O(n^2)$ для удерживаемых списков. Ограничиваем учебные входы 100 целыми; повышение предела рекурсии не исправляет плохое разбиение.

Отступы набирайте обычными пробелами: 4 на уровень. При копировании с веб-страницы замените неразрывные пробелы обычными; полный код с обычными пробелами есть в PDF.

Для [4,1,4,3,2] средний индекс 2, `pivot=4`: `low=[1,3,2]`, `equal=[4,4]`, `high=[]`. У `low pivot=3`: `low=[1,2]`, `equal=[3]`. Сортировка [1,2] вернёт [1,2]; сборка даёт [1,2,3,4,4]. Исходный список не изменяется.

```
def quick(a):
    if len(a) < 2:
        return a[:]
    pivot = a[len(a) // 2]
    low, equal, high = [], [], []
    for x in a:
        if x < pivot:
            low.append(x)
        elif x == pivot:
            equal.append(x)
        else:
            high.append(x)
    return quick(low) + equal + quick(high)
```

```
a = [4, 1, 4, 3, 2]
print(quick(a))
print(a)
```



Опорное	$a[2] = 4$
Меньше	$[1, 3, 2] \rightarrow [1, 2, 3]$
Равны / больше	$[4, 4] / []$
Сборка	$[1, 2, 3] + [4, 4] + []$

Три непересекающиеся группы сохраняют обе четвёрки.

03. Практика 1. Проверяем материал

Отлавливаем потерю повторов

14–25 минут. Сначала вручную заполните группы первого разбиения для каждого непустого входа.

Вход	Результат
$[4, 1, 4, 3, 2]$	$[1, 2, 3, 4, 4]$
$[]; [6]$	$[]; [6]$
$[5, 5, 5, 5]$	$[5, 5, 5, 5]$
$[-2, 0, -2, 3]$	$[-2, -2, 0, 3]$

2 минуты — прогноз, 4 — пять запусков, 3 — сравнение с `sorted(a)` и проверка длины, 2 — объяснение завершения. Форма: вход / `pivot` / группы / итог. Критерии: сохранены повторы, пустота безопасна, источник не изменён, каждая рекурсивная часть короче.

04. Практика 2. Перенос способа

Сравниваем выбор опорного

25–33 минуты. Сравните на $[1, 2, 3, 4, 5, 6, 7]$ две версии: `pivot=a[len(a)//2]` и `pivot=a[0]`. За 2 минуты нарисуйте первые группы, за 4 выпишите размеры всех непустых разбиений, за 2 сложите их. Считаем посещённые элементы только в цикле `for`, не отдельные операции сравнения. Критерии: две суммы и ограниченный вывод.

05. Обсуждение и границы вывода

33–40 минут. Ученик говорит: «Средний элемент всегда делает половины равными». За 2 минуты найдите контрпример из пяти различных чисел, за 3 объясните отличие индекса от ранга, за 2 предложите практический выбор для обычной программы.

06. Индивидуальная проверка

40–45 минут. Для $[2, 2, 1]$ назовите `pivot`, три группы, итог и причину завершения. Четыре балла.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Для четырёх пятёрок `pivot=5`, `low=high=[]`, `equal` содержит все четыре значения, итог той же длины. Для `[-2,0,-2,3]` `pivot=-2`, `low=[]`, `equal=[-2,-2]`, `high=[0,3]`. Ошибка `quick(a)` вместо `quick(low)` не уменьшает задачу и может вызвать бесконечную рекурсию до ограничения интерпретатора.

Практика 2: разбор и критерии

Средний индекс: размеры 7,3,3, сумма 13; одиночки не входят в цикл. Первый элемент: 7,6,5,4,3,2, сумма 27. Это точные результаты для данного входа и данного счётчика, а не универсальное время в секундах. Чтобы проверить программно, увеличивайте общий счётчик на `len(a)` сразу после базового случая и обнуляйте перед каждым вызовом.

Обсуждение: возможный вывод

В `[2,3,1,4,5]` средний элемент равен 1: `low` пуст, `high` длины 4. Средний индекс — положение в текущей записи; медиана — значение среднего ранга после упорядочения. Для прикладной сортировки обычно выбирают проверенную встроенную функцию, а собственный алгоритм используют для изучения и специальных задач.

Индивидуальная проверка: ответ и критерии

`pivot=2`; `low=[1]`, `equal=[2,2]`, `high=[]`; итог `[1,2,2]`; сортируемые части имеют длины 1 и 0, поэтому срабатывает база. Один балл за каждый пункт.

Учителю: подготовка и план на 45 минут

Подготовьте компьютер, редактор, папку с правом записи, распечатку ученической части PDF, бумагу и ручку. Python 3.11+ и IDLE или редактор; запуск `python lesson.py` (либо `python3 lesson.py`). Внешние библиотеки не нужны. Установку и проверку запуска учитель выполняет заранее. Все программы, данные и ожидаемые ответы даны; регистрация детей и интернет на занятии не нужны.

Уровень — начинающий программист, 9 класс. Предпосылки: рекурсия, списки, сравнения и сортировка слиянием. В паре автор прогнозирует и вводит код, проверяющий ведёт журнал, затем роли меняются. Без компьютера выполните трассировку полного кода на бумаге и сопоставьте с готовыми результатами; учитель демонстрирует подготовленный запуск. Бумажный вариант не считается фактическим запуском ученика. Перед практикой учитель выдаёт готовый файл разобранного примера, чтобы дети тратили время на изменение и проверку, а не длинное перепечатывание. Матрица программы предназначена учителю.

Минуты	Работа	Результат
0–5	Проблема	Вопрос и первое решение
5–14	Понятия и пример	Способ проверки
14–25	Практика 1	Готовый продукт и проверка
25–33	Практика 2	Перенос способа
33–40	Обсуждение	Аргумент и ограничение
40–45	Индивидуальная проверка	Самостоятельный ответ



07. Соответствие исходной программе

Пункт программы	Действие ученика	Проверяемый результат
Алгоритмы сортировки	Отлавливаем потерю повторов	Реализовать трёхчастную быструю сортировку и объяснить завершение на повторяющихся значениях и риск несбалансированных разбиений.
Структурирование решения и отладка	Сравниваем выбор опорного	Исходник, журнал обычных и граничных тестов, самостоятельное объяснение

Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-9-klass-programma>

NIST: быстрая сортировка

<https://xlinux.nist.gov/dads/HTML/quicksort.html>

Тексты, учебные данные, реплики, программы и схемы подготовлены для уроков портала «Школы России». Числовые серии — учебные модели, не результаты проведённых классом опытов. Схемы авторские, сторонние фотографии не используются. Методические ссылки приведены для учителя; ученик получает все материалы здесь. Проверка первичных источников: 18.10.2026.