



Урок 17. Python: Сортировка слиянием

Часть 1. Материалы ученика

Программирование на Python и C++ · 9 класс

Урок 17 из 34 · 45 минут · Модуль «Сортировка»

Уровень: начинающий программист; Python 3.

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

Две упорядоченные стопки карточек можно объединить без поиска минимума по всей стопке. Достаточно сравнивать две первые ещё не взятые карточки.

Результат урока: Написать сортировку слиянием, объяснить шаг объединения и проверить сохранение всех элементов, включая повторы.

01. Проблемная ситуация

0–5 минут. Даны две уже упорядоченные серии [1,4,7] и [2,4,6]. Выпишите общий порядок, каждый раз выбирая один из двух первых доступных элементов. Пометьте две четвёрки буквами Л и П: какая будет первой при равенстве?

02. Понятия и разобранный пример

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

Разделение, база и слияние

Сортировка делит список примерно пополам, сортирует половины и сливает их. База — длина 0 или 1: такой список уже упорядочен. Размер каждой рекурсивной части меньше исходного, поэтому рекурсия заканчивается. Срез `a[:m]` создаёт новый список, исходный `a` не меняется.

При слиянии `i` и `j` указывают на первые неиспользованные элементы левой и правой частей. В `out` уже лежат наименьшие взятые элементы в порядке неубывания. Выбор меньшей головы сохраняет это свойство. Если одна часть закончилась, остаток другой уже отсортирован и переносится целиком.

Условие `<=` при равенстве берёт левый элемент первым. Это сохраняет исходный порядок равных ключей при соответствующем сравнении записей: устойчивость. На каждом уровне слияний суммарная работа пропорциональна n , уровней порядка $\log_2(n)$; время $O(n \log n)$, дополнительная память этой реализации $O(n)$ в пике. Учебный контракт: список из 0...100 целых чисел; на огромных входах учитывают память.

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 [7,1,4,2] делим на [7,1] и [4,2], получаем [1,7] и [2,4]. На верхнем слиянии сравниваем 1 и 2, затем 7 и 2, затем 7 и 4; остаток [7] дописываем. Итог [1,2,4,7], три сравнения на последнем слиянии. Это не общее число сравнений всей сортировки.

```
def merge_sort(a):
    if len(a) < 2:
        return a[:]
    m = len(a) // 2
    left = merge_sort(a[:m])
    right = merge_sort(a[m:])
    i = j = 0
    out = []
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            out.append(left[i])
            i += 1
        else:
            out.append(right[j])
            j += 1
    return out + left[i:] + right[j:]
```

```
a = [7, 1, 4, 2]
print(merge_sort(a))
print(a)
```



Разделение	<code>[7,1,4,2] → [7,1] [4,2]</code>
После рекурсии	<code>[1,7] [2,4]</code>
Выбор голов	<code>1 → 2 → 4; остаток 7</code>
Результат	<code>[1,2,4,7]; исходник не изменён</code>

Трасса сортировки слиянием: половины сортируются до объединения.

03. Практика 1. Проверяем материал

Проверяем результат и исходный список

14–25 минут. Все серии учебные. Используйте готовый файл примера, меняйте `a` и сохраняйте исходный список для сравнения.

Вход	Ожидаемый результат
<code>[7,1,4,2]</code>	<code>[1,2,4,7]</code>
<code>[]; [5]</code>	<code>[]; [5]</code>
<code>[2,-1,2,0]</code>	<code>[-1,0,2,2]</code>
<code>[3,2,1]; [1,2,3]</code>	<code>[1,2,3]</code> в обоих случаях

2 минуты — прогноз по таблице, 4 — шесть запусков, 3 — проверка длины и повторов, 2 — журнал «вход / результат / исходный список после вызова». Критерии: шесть верных результатов, две двойки не потеряны, а не изменён; объясните назначение дописывания остатков.

04. Практика 2. Перенос способа

Переходим к убыванию

25–33 минуты. Требуется убывание для `[3,1,3,-2]`. За 2 минуты определите, какую строку сравнения заменить; за 4 измените копию функции и проверьте также `[]` и `[9]`; за 2 объясните, почему недостаточно развернуть только одну половину. Результат: изменённая строка, три вывода и аргумент.

05. Обсуждение и границы вывода

33–40 минут. Один проверяющий смотрит только первый и последний элементы, другой сравнивает весь результат с `sorted(a)`. За 2 минуты найдите недостаток первого подхода, за 3 предложите три свойства проверки, за 2 объясните роль эталонной функции.

06. Индивидуальная проверка

40–45 минут. Слейте `[0,5]` и `[1,5,8]` с выбором левого при равенстве. Укажите порядок с метками 5Л/5П, число сравнений голов, остаток и базовый случай сортировки. Четыре балла.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Все результаты указаны в таблице, исходные списки остаются прежними. Если забыть `+ left[i:] + right[j:]`, для `[7,1,4,2]` потеряется последний 7, а некоторые двухэлементные списки укоротятся. Правильный порядок сам по себе не доказывает сохранение элементов: `[]` тоже упорядочен.

Практика 2: разбор и критерии

Замените `if left[i] <= right[j]` на `if left[i] >= right[j]`: во всех вызовах той же функции. Получится `[3,3,1,-2]`, `[]`, `[9]`. Обе половины и операция слияния должны следовать одному направлению; иначе сравнение голов не выбирает нужный следующий элемент. Равенство по-прежнему отдаёт приоритет левой части.

Обсуждение: возможный вывод

Правильные края не исключают беспорядок внутри: `[1,3,2,4]`. Проверяем неубывание каждой соседней пары, сохранение количества каждого значения и неизменность исходника по контракту. `sorted(a)` удобен как независимый эталон в тесте, но не заменяет написание `merge_sort` в учебном задании.

Индивидуальная проверка: ответ и критерии

0,1,5Л,5П,8; три сравнения: 0/1, 5/1, 5/5; остаток правой части `[5П,8]`; база `len(a)<2`. По одному баллу 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: списки, индексы, циклы, функции и базовый случай рекурсии. 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	Проверяем результат и исходный список	Написать сортировку слиянием, объяснить шаг объединения и проверить сохранение всех элементов, включая повторы.
Структурирование решения и отладка	Переходим к убыванию	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: сортировка слиянием

<https://xlinux.nist.gov/dads/HTML/mergesort.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.