Лабораторная работа 3 - Улучшение QuickSort, Anti-QuickSort, индекс Хирша

Цель

Реализовать улучшенные варианты QuickSort по лекции 3, построить анти-тест для заданной реализации QuickSort, а также посчитать h-index для массива цитирований.

Задание 1 - Улучшение QuickSort

Сделано 2 варианта сортировки.

1. Randomized-QuickSort + Partition (как в лекции).

- Опорный элемент выбирается случайно (меняем местами `a[l]` и `a[k]`).
- `partition` делает разбиение на 2 части: `<= x` и `> x`.
- Массив сортируется на месте.

2. Randomized-QuickSort + Partition3 (по условию для одинаковых элементов).

- Разбиение на 3 части: `< x`, `= x`, `> x`.
- Рекурсия идет только в части `< x` и `> x`, что ускоряет случай с большим числом повторов.

Вариант 1 - Partition (2 части)

```
import random
import sys

sys.setrecursionlimit(1_000_000)

def partition(a, l, r):
    x = a[l]
    j = l
    for i in range(l + 1, r + 1):
        if a[i] <= x:
            j += 1
            a[j], a[i] = a[i], a[j]
    a[l], a[j] = a[j], a[l]
    return j

def randomized_quicksort(a, l, r):
    if l < r:
        k = random.randint(l, r)
        a[l], a[k] = a[k], a[l]
        m = partition(a, l, r)
        randomized_quicksort(a, l, m - 1)
        randomized_quicksort(a, m + 1, r)

def main():
```

```
    with open("n1/input_worst.txt", "r", encoding="utf-8") as f:
        n = int(f.readline())
        a = list(map(int, f.readline().split()))

    randomized_quicksort(a, 0, n - 1)

    with open("n1/output1_worst.txt", "w", encoding="utf-8") as f:
        f.write(" ".join(map(str, a)))

if __name__ == "__main__":
    main()
```

Вариант 2 - Partition3 (3 части)

```
import random
import sys

sys.setrecursionlimit(1_000_000)

def partition3(a, l, r):
    x = a[l]
    lt = l
    i = l
    gt = r

    while i <= gt:
        if a[i] < x:
            a[lt], a[i] = a[i], a[lt]
            lt += 1
            i += 1
        elif a[i] > x:
            a[i], a[gt] = a[gt], a[i]
            gt -= 1
        else:
            i += 1

    return lt, gt

def randomized_quicksort3(a, l, r):
    if l < r:
        k = random.randint(l, r)
        a[l], a[k] = a[k], a[l]
        m1, m2 = partition3(a, l, r)
        randomized_quicksort3(a, l, m1 - 1)
        randomized_quicksort3(a, m2 + 1, r)

def main():
    with open("n1/input_worst.txt", "r", encoding="utf-8") as f:
        n = int(f.readline())
        a = list(map(int, f.readline().split()))
```

```
    randomized_quicksort3(a, 0, n - 1)

    with open("n1/output2_worst.txt", "w", encoding="utf-8") as f:
        f.write(" ".join(map(str, a)))

if __name__ == "__main__":
    main()
```

Генератор тестов для задания 1

Генерируются 4 случая:

- best - уже отсортирован
- avg - случайная перестановка
- worst - обратный порядок
- few_unique - мало уникальных значений (для проверки Partition3)

```
import random

random.seed("chikirao")

def write_case(filename, a):
    with open(filename, "w", encoding="utf-8") as f:
        f.write(str(len(a)) + "\n")
        f.write(" ".join(map(str, a)) + "\n")

n = 10000

a_best = list(range(n))
a_worst = list(range(n, 0, -1))

a_avg = list(range(n))
random.shuffle(a_avg)

few_vals = [1, 2, 3, 4, 5]
a_few_unique = [random.choice(few_vals) for _ in range(n)]

write_case("n1/input_best.txt", a_best)
write_case("n1/input_avg.txt", a_avg)
write_case("n1/input_worst.txt", a_worst)
write_case("n1/input_few_unique.txt", a_few_unique)
```

Результат - для каждого входного файла формируются выходные файлы `output1_*.txt` (Partition) и `output2_*.txt` (Partition3). Во всех случаях массив на выходе отсортирован.

Задание 2 - Anti-QuickSort

Нужно вывести перестановку чисел `1..n`, на которой заданная реализация QuickSort делает максимальное число сравнений.

Идея:

- стартуем с `a = [1, 2, 3, ..., n]`
- для `i` от 2 до `n-1` делаем `swap(a[i], a[i//2])`

Так как `n` может быть до `10^6`, запись в `output.txt` идет чанками, чтобы не хранить огромные строки в памяти.

```
def main():
    with open("n2/input.txt", "r", encoding="utf-8") as f:
        n = int(f.readline().strip())

    a = list(range(1, n + 1))

    for i in range(2, n):
        j = i // 2
        a[i], a[j] = a[j], a[i]

    with open("n2/output.txt", "w", encoding="utf-8") as out:
        chunk = 20000
        first = True
        for start in range(0, n, chunk):
            part = a[start:start + chunk]
            s = " ".join(map(str, part))
            if first:
                out.write(s)
                first = False
            else:
                out.write(" " + s)

if __name__ == "__main__":
    main()
```

Скриншот проверки на acmp (Accepted, время и память в пределах лимитов):

Посылки решений:

ID	Дата	Язык	Результат	Тест	Время	Память
24748523	22.02.2026 3:35:17	Python	Accepted		0,062	4644 Кб

[Обсуждение] [Мои попытки] [Лучшие попытки]

Задание 5 - Индекс Хирша

Вход - одна строка `citations`, числа могут быть разделены пробелами или запятыми. Выход - одно число `h`.

Алгоритм:

- сортируем по убыванию
- ищем максимальное `h`, такое что `citations[h-1] >= h`

```
def h_index(citations):
    citations.sort(reverse=True)
    h = 0
    for i, c in enumerate(citations, start=1):
        if c >= i:
            h = i
        else:
            break
    return h

def main():
    with open("n3/input.txt", "r", encoding="utf-8") as f:
        s = f.read().strip()

    s = s.replace(",", " ")
    citations = [int(x) for x in s.split()] if s else []

    ans = h_index(citations)

    with open("n3/output.txt", "w", encoding="utf-8") as f:
        f.write(str(ans))

if __name__ == "__main__":
    main()
```

😎