САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИТМО
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №3
по курсу «Алгоритмы и структуры данных»
Тема: Быстрая сортировка, сортировки за линейное время
Вариант 17

Выполнил:
Останин А. С.
К3140

Проверил:
Афанасьев А. В.

Санкт-Петербург
2024 г.

Содержание отчета

Задачи по варианту

Задача №1. Улучшение Quick sort

Используя *псевдокод* процедуры `Randomized - QuickSort`, а так же `Partition` из презентации к Лекции 3 (страницы 8 и 12), напишите программу быстрой сортировки на Python и проверьте ее, создав несколько рандомных массивов, подходящих под параметры:

- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($1 \leq n \leq 10^4$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, ***по модулю*** не превосходящих 10^9.
- **Формат выходного файла (output.txt).** Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

```
def task1(array: list) -> list:
    if len(array) <= 1:
        return array

    elem = random.choice(array)
    left = []
    center = []
    right = []

    for i in array:
        if i < elem:
            left.append(i)
        elif i == elem:
            center.append(i)
        else:
            right.append(i)

    return task1(left) + center + task1(right)
```

Выбирает случайный опорный элемент (elem) из входного массива.
Разделяет массив на три части: элементы меньше опорного (left), равные опорному (center) и больше опорного (right).
Рекурсивно сортирует части left и right.
Объединяет отсортированные части left, center и right для получения окончательно отсортированного массива.

```
Input: 5
2 3 9 2 2
Output: 2, 2, 2, 3, 9
```

	Время выполнения	Затраты памяти

Нижняя граница диапазона значений входных данных из текста задачи		
Пример из задачи	[redacted]	[redacted]
Верхняя граница диапазона значений входных данных из текста задачи		

Вывод по задаче: Этот код имеет среднюю временную сложность O(nlog n), но может быть O(n^2) в худшем случае, если опорный элемент выбран неудачно.

Задача №2. Anti-quick sort

Для сортировки последовательности чисел широко используется быстрая сортировка - `QuickSort`. Далее приведена программа на языке ~~Pascal~~ Python, которая сортирует массив a, используя этот алгоритм.

```
def qsort (left, right):
        key = a [(left + right) // 2]
        i = left
        j = right
        while i <= j:
                while a[i] < key: # first while
                        i += 1
                while a[j] > key : # second while
                        j -= 1
                if i <= j :
                        a[i], a[j] = a[j], a[i]
                        i += 1
                        j -= 1
        if left < j:
                qsort(left, j)
        if i < right:
                qsort(i, right)

qsort(0, n - 1)
```

Хотя `QuickSort` является очень быстрой сортировкой в среднем, существуют тесты, на которых она работает очень долго. Оценивать время работы алгоритма будем числом сравнений с элементами массива (то есть, суммарным числом сравнений в первом и втором while). Требуется написать программу, генерирующую тест, на котором быстрая сортировка сделает наибольшее число таких сравнений.

```
def task2(n:int ) -> list:
    a = [i for i in range(1, n + 1)]
    for i in range(2, len(a)):
        a[i], a[i // 2] = a[i // 2], a[i]
    return a
```

Эта функция генерирует список чисел от 1 до n, затем меняет местами каждый элемент с четным индексом со своим родительским элементом (с индексом i // 2), создавая таким образом худший случай для алгоритма сортировки Quick Sort

```
Input: 5
Output: 1, 4, 5, 3, 2
```

	Время выполнения	Затраты памяти
Нижняя граница		

диапазона значений входных данных из текста задачи		
Пример из задачи	[redacted]	[redacted]
Верхняя граница диапазона значений входных данных из текста задачи		

Вывод по задаче: Написан алгоритм, создающий худший случай для алгоритма быстрой сортировки

Задача №6. Сортировка целых чисел

В этой задаче нужно будет отсортировать много неотрицательных целых чисел.

Вам даны два массива, A и B, содержащие соответственно n и m элементов. Числа, которые нужно будет отсортировать, имеют вид $A_i \cdot B_j$, где $1 \leq i \leq n$ и $1 \leq j \leq m$. Иными словами, каждый элемент первого массива нужно умножить на каждый элемент второго массива.

Пусть из этих чисел получится отсортированная последовательность C длиной $n \cdot m$. Выведите сумму каждого десятого элемента этой последовательности (то есть, $C_1 + C_{11} + C_{21} + ...$).

```
def combinations(A: list, B: list) -> list:
    res = []
    for a in A:
        for b in B:
            res.append(a * b)
    return res


def sum_tenth(array: list) -> int:
    res = 0
    for i in range(0, len(array), 10):
        res += array[i]
    return res


def task6(A: list, B: list) -> int:
    C = qsort(combinations(A, B))  # сортируем массив, состоящий из
произведений элементов списков A и B
    answer = sum_tenth(C)
    return answer
```

combinations(A, B): генерирует список произведений каждого элемента в A с каждым элементом в B.
qsort(...): сортирует список произведений в порядке возрастания (qsort мы берём из первого задания в этой лабораторной работе, где мы писали алгоритм быстрой сортировки).
sum_tenth(...): вычисляет сумму каждого 10-го элемента в отсортированном списке, оно и является ответом

```
Input: 4 4
7 1 4 9
2 7 8 11
Output: 51
```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи		
Пример из задачи		
Верхняя граница диапазона значений входных данных из текста задачи		

Вывод по задаче: Реализован алгоритм нахождения суммы каждого десятого элемента в сортированном массиве

Вывод

Реализован алгоритм быстрой сортировки, сортировки целых чисел, а также написан алгоритм, создающий худший случай для алгоритма быстрой сортировки