САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

# Отчет по лабораторной работе №3

по курсу «Алгоритмы и структуры данных»

Тема: Быстрая сортировка, сортировка за линейное время

Выполнил:
Волжева М.И.
К3141

Проверила:
Артамонова В.Е.

Санкт-Петербург
2022 г.

## Содержание отчета

## Задачи по варианту

### Задача №1. Улучшение Quick sort

Текст задачи:
Используя псевдокод процедуры Randomized - QuickSort, а так же Partition из презентации к Лекции 3 (страницы 8 и 12), напишите программу быстрой сортировки на Python. Цель задачи - переделать данную реализацию рандомизированного алгоритма быстрой сортировки, чтобы она работала быстро даже с последовательностями, содержащими много одинаковых элементов.

- Формат входного файла (input.txt). В первой строке входного файла содержится число n ($1 \le n \le 10^4$ ) — 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. 2сек.
- Ограничение по памяти. 256 мб.
- Пример:

| | |
|---|---|
| Input.txt | 10<br>1 0 5 1 -10 2 7 -5 -5 4 |
| Output.txt | -10 -5 -5 0 1 1 2 4 5 7 |

Листинг кода:

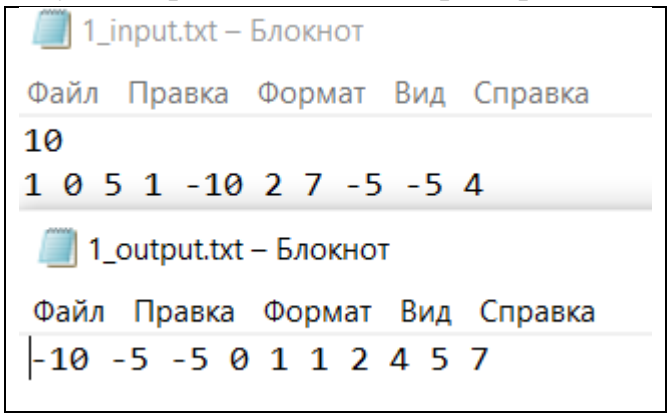
```
import time
import os, psutil
import random


def quicksort(nums):
    if len(nums) <= 1:
        return nums
    else:
        q = random.choice(nums)
        s_nums = []
        m_nums = []
        e_nums = []
        for n in nums:
            if n < q:
                s_nums.append(n)
            elif n > q:
                m_nums.append(n)
            else:
```

```
                e_nums.append(n)
        return quicksort(s_nums) + e_nums + quicksort(m_nums)


def test(a, b):
    k = []
    for i in range(0, a):
        k.append(random.randint(-b, b))
    return k


t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("1_input.txt", "w")
m = open("1_output.txt", "w")

f.write("10\n")
numbers = test(10, 10)
f.write(" ".join(map(str, numbers)))
m.write(" ".join(map(str, quicksort(numbers))))

f.close()
m.close()

print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Я переписала псевдокод данный в презентации, далее передала его, так чтобы элементы, равные ключу, сохранялись отдельно и не сортировались далее. Далее написала функцию для генерации массива и провела тестирование.

Результат работы кода на примерах из текста задачи:




Тестирование алгоритма:

| | Время выполнения | Затраты памяти |
|---|---|---|
| Нижняя граница | 0.0019472999999999999 second | 13.703125 mb |

| | | |
|---|---|---|
| диапазона значений входных данных из текста задачи | | |
| Пример из задачи | 0.013001299999999993 second | 13.57421875 mb |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.015651599999999988 second | 13.76953125 mb |

Вывод по задаче:.Мы научились писать “быструю сортировку” и убедились, что она работает быстрее других, ранее изученных сортировок.

**Задача №5. Сортировка слиянием+**

Текст задачи:
Для заданного массива целых чисел citations, где каждое из этих чисел – число цитирований i-ой статьи ученого-исследователя, посчитайте индекс Хирша этого ученого.

- Формат ввода или входного файла (input.txt). Одна строка citations, содержащая n целых чисел, по количеству статей ученого (длина citations), разделенных пробелом или запятой.
- Формат выхода или выходного файла (output.txt). Одно число - индекс Хирша (h-индекс).
- Ограничения: $1 \leq n \leq 5000$, $0 \leq citations[i] \leq 1000$.
- Пример:

| Input.txt | 3 0 6 1 5 |
|---|---|
| Output.txt | 3 |

Листинг кода:

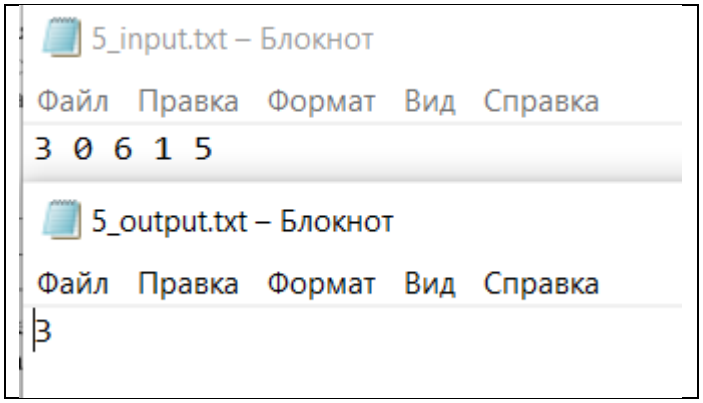
```
import time
import os, psutil
import random


def quicksort(nums):
    if len(nums) <= 1:
        return nums
    else:
        q = random.choice(nums)
        s_nums = []
        m_nums = []
        e_nums = []
        for n in nums:
            if n < q:
                s_nums.append(n)
            elif n > q:
                m_nums.append(n)
            else:
                e_nums.append(n)
        return quicksort(s_nums) + e_nums + quicksort(m_nums)


def test(a, b):
    k = []
    for i in range(0, a):
        k.append(random.randint(-b, b))
    return k


t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("5_input.txt")
m = open("5_output.txt", "w")
```

```
string = f.readline()
numbers = list(map(int, string.split()))
numbers_new = quicksort(numbers)
m.write(str(numbers_new[int(len(numbers_new)/2)]))

f.close()
m.close()

print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Мы сортируем массив методом быстрой сортировки, а затем берем “средний ” элемент. Он и является ответом.

Результат работы кода на примерах из текста задачи:




Тестирование алгоритма:

| | Время выполнения | Затраты памяти |
|---|---|---|
| Пример из задачи | 0.014959600000000073 second | 13.61328125 mb |

Вывод по задаче:.Мы поняли, что такое индекс Хирша.

**Задача №6. Сортировка целых чисел**

Текст задачи:

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 Ai  Bj , где 1 ≤ i ≤  и 1 ≤ j ≤ m. Иными словами, каждый элемент первого массива нужно умножить на каждый элемент второго массива.

Пусть из этих чисел получится отсортированная последовательность C длиной n*m. Выведите сумму каждого десятого элемента этой последовательности (то есть, C1 + C11 + C21 + ...)

- Формат входного файла (input.txt). В первой строке содержатся числа n и m (1 ≤ n, m ≤ 6000) – размеры массивов. Во второй строке содержится n чисел – элементы массива A. Аналогично, в третьей строке содержится m чисел — элементы массива B. Элементы массива неотрицательны и не превосходят 40000.
- Формат выходного файла (output.txt). Выведите одно число — сумму каждого десятого элемента последовательности, полученной сортировкой попарных произведений элементов массивов A и B.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.
- Пример:

| Input.txt | 4 4<br>7 1 4 9<br>2 7 8 11 |
|---|---|
| Output.txt | 51 |

Листинг кода

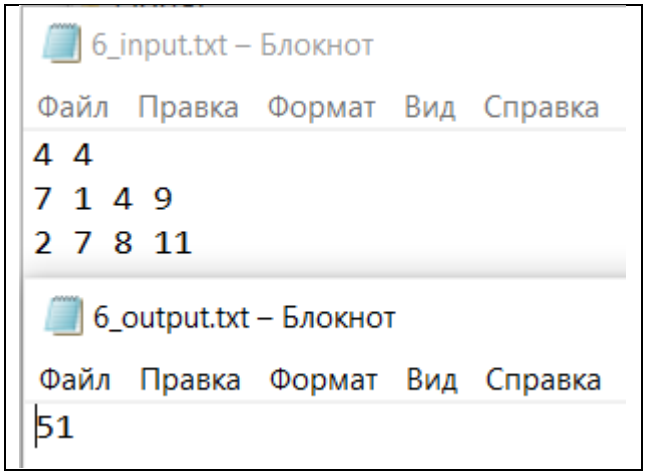
```
import time
import os, psutil
import random


def quicksort(nums, fst, lst):
    if fst >= lst: return

    i, j = fst, lst
    pivot = nums[random.randint(fst, lst)]

    while i <= j:
        while nums[i] < pivot: i += 1
        while nums[j] > pivot: j -= 1
        if i <= j:
```

```
            nums[i], nums[j] = nums[j], nums[i]
            i, j = i + 1, j - 1
    quicksort(nums, fst, j)
    quicksort(nums, i, lst)


def test(a, b):
    k = []
    for i in range(0, a):
        k.append(random.randint(-b, b))
    return k


t_start = time.perf_counter()
process = psutil.Process(os.getpid())
f = open("6_input.txt")
m = open("6_output.txt", "w")

string1 = f.readline()
string2 = f.readline()
string3 = f.readline()
numbers_1 = list(map(int, string2.split()))
numbers_2 = list(map(int, string3.split()))
numbers = []
for i in numbers_1:
    for j in numbers_2:
        numbers.append(i*j)
quicksort(numbers, 0, len(numbers) - 1)
print(numbers)
numbers_good = []
a = len(numbers)//10
print(a)
b = 0

for i in range(0, a+1):
    numbers_good.append(numbers[i*10])


m.write(str(sum(numbers_good)))
f.close()
m.close()

print("Time of working: %s second" % (time.perf_counter() - t_start))
print("Memory", process.memory_info().rss/(1024*1024), "mb")
```

Текстовое объяснение решения:

Сначала было написано перемножение массивов, а затем произведение “быстрая сортировка”, с использованием улучшения с минимизацией затрачиваемой памяти. В конце были выведены каждые 10 элементы массива.

Результат работы кода на примерах из текста задачи:




Тестирование алгоритма:

| | Время выполнения | Затраты памяти |
|---|---|---|
| Пример из задачи | 0.0048245999999999845 second | 13.65234375 mb |

Вывод по задаче: Я научилась использовать алгоритм “быстрой сортировки” в прикладных задачах.

**Вывод**

Я узнала как работает быстрая сортировка и научилась ее реализовывать и применять на некоторых прикладных задачах.