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: Тема работы

Вариант 23

Выполнил:

Юшков А.М.

К3139

Санкт-Петербург

2025г.

# Содержание отчета

[**Содержание отчета**](#_10udeakjagvs) **2**

[**Задачи по варианту**](#_gf7yxvsg0zb) **3**

[Задача №1. Улучшение](#_pgzbaj56cf0o) Quick sort 3

[Задача №3. Сортировка](#_pgzbaj56cf0o) пугалом 4

[Задача №6. Сортировка](#_pgzbaj56cf0o) целых чисел 5

[**Вывод**](#_fu90fuyk873) **6**

# Задачи по варианту

## Задача №1. Улучшение быстрой сортировки

from lab3.utils import \*  
import random  
  
  
def partition(array, left, right):  
 pivot\_value = array[left]  
 left\_pointer = left  
 for current\_pointer in range(left + 1, right):  
 if array[current\_pointer] < pivot\_value:  
 left\_pointer += 1  
 array[left\_pointer], array[current\_pointer] = array[current\_pointer], array[left\_pointer]  
 array[left], array[left\_pointer] = array[left\_pointer], array[left]  
 return left\_pointer  
  
  
def quick\_sort(array, left=0, right=None):  
 if len(array) <= 1:  
 return array  
 if right is None:  
 right = len(array)  
  
 if left < right:  
 pivot = (left + right) // 2  
 array[pivot], array[left] = array[left], array[pivot]  
 pivot = partition(array, left, right)  
 random\_quick\_sort(array, left, pivot)  
 random\_quick\_sort(array, pivot + 1, right)  
 return array  
  
  
def random\_quick\_sort(array, left=0, right=None):  
 if len(array) <= 1:  
 return array  
 if right is None:  
 right = len(array)  
  
 if left < right:  
 pivot = random.randint(left, right - 1)  
 array[pivot], array[left] = array[left], array[pivot]  
 pivot\_index = partition(array, left, right)  
 random\_quick\_sort(array, left, pivot\_index)  
 random\_quick\_sort(array, pivot\_index + 1, right)  
 return array  
  
  
def partition3(array, left, right):  
 pivot\_value = array[left]  
 left\_pointer = left  
 right\_pointer = right - 1  
 current\_pointer = left + 1  
 while current\_pointer <= right\_pointer:  
 if array[current\_pointer] < pivot\_value:  
 array[left\_pointer], array[current\_pointer] = array[current\_pointer], array[left\_pointer]  
 left\_pointer += 1  
 current\_pointer += 1  
 elif array[current\_pointer] > pivot\_value:  
 array[right\_pointer], array[current\_pointer] = array[current\_pointer], array[right\_pointer]  
 right\_pointer -= 1  
 else:  
 current\_pointer += 1  
 return left\_pointer, right\_pointer  
  
  
def random\_quick\_sort3(array, left=0, right=None):  
 if len(array) <= 1:  
 return array  
 if right is None:  
 right = len(array)  
  
 if left < right:  
 pivot = random.randint(left, right - 1)  
 array[pivot], array[left] = array[left], array[pivot]  
 middle\_left, middle\_right = partition3(array, left, right)  
 random\_quick\_sort3(array, left, middle\_left)  
 random\_quick\_sort3(array, middle\_right + 1, right)  
 return array  
  
def qsort\_plus\_main():  
 n, arr = file\_read()  
 sorted\_array = random\_quick\_sort(arr)  
 file\_write(sorted\_array)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 measure(qsort\_plus\_main)

Текстовое объяснение решения.

В данной программе был улучшен алгоритм быстрой сортировки, это было достигнуто благодаря выбору случайного pivot элемента и последующего деления массива на три части

Результат работы кода на примере из текста задачи

Input.txt

5  
2 3 9 2 2

Output.txt

2 2 2 3 9

## Задача №3. Сортировка пугалом

from lab3.utils import \*  
from lab3.task1.src.task1 import random\_quick\_sort  
  
def sort\_dolls(dolls\_number, hand, dolls):  
 groups = [[] for \_ in range(hand)]  
  
 for i in range(dolls\_number):  
 groups[i % hand].append(dolls[i])  
  
 for group in groups:  
 random\_quick\_sort(group)  
  
 sorted\_dolls = []  
 index = [0] \* hand  
  
 for i in range(dolls\_number):  
 group\_index = i % hand  
 sorted\_dolls.append(groups[group\_index][index[group\_index]])  
 index[group\_index] += 1  
  
 if sorted\_dolls == random\_quick\_sort(dolls):  
 return "YES"  
 return "NO"  
  
  
def scarecrow\_sort():  
 (n, k), array = file\_read()  
 result = sort\_dolls(n, k, array)  
 file\_write([result])  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 measure(scarecrow\_sort)

Данная программа выполняет проверку возможности сортировки массива на основе входных данных, программа проверяет, можно ли отсортировать массив чисел, разделив их на группы, отсортировав каждую группу и затем объединив их обратно, чтобы получить полностью отсортированный массив.

Результат работы кода на примере из текста задачи

Input.txt

5 3  
1 5 3 4 1

Output.txt

YES

[**Задача №6. Сортировка**](#_pgzbaj56cf0o) **целых чисел**

from lab3.utils import \*  
from lab3.task1.src.task1 import random\_quick\_sort  
  
  
def sum\_tenth(combinations):  
 random\_quick\_sort(combinations)  
 return sum(combinations[i] for i in range(0, len(combinations), 10))  
  
  
def multiply(n, m, array1, array2):  
 combinations = []  
 for i in range(n):  
 for j in range(m):  
 combinations.append(array1[i] \* array2[j])  
 return combinations  
  
  
def solve(n, m, array1, array2):  
 combinations = multiply(n, m, array1, array2)  
 result = sum\_tenth(combinations)  
 return result  
  
  
def zahlen\_sort\_main():  
 (n, m), array\_a, array\_b = file\_read()  
 result = solve(n, m, array\_a, array\_b)  
 file\_write([result])  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 measure(zahlen\_sort\_main)

Программа вычисляет сумму каждого 10-го элемента отсортированного массива, который состоит из всех попарных произведений элементов двух входных массивов.

Результат работы кода на примере из текста задачи

Input.txt

4 4  
7 1 4 9  
2 7 8 11

Output.txt

51

# Вывод

В ходе выполнения данной лабораторной я ознакомился с новыми методами сортировок и научился усовершенствовать некоторые старые версии