Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины «Алгоритмизация»
Вариант 17

Выполнила:
Середа Кирилл Витальевич
1 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Ход выполнения заданий

1) Написал программу подсчета инверсий в массиве, которая работает за время O(n*log(n))

```
************************************************
******************** Алгоритм сортировки ********************
************************************************

Текущий массив:
93, 90, 83, 41, 22, 80, 43, 60

Начать сортировку? (y/n): y

Сортировка!

Сортировка - Шаг 1: 22

Сортировка - Шаг 2: 41

Сортировка - Шаг 3: 43

Сортировка - Шаг 4: 60

Сортировка - Шаг 5: 80

Сортировка - Шаг 6: 83

Сортировка - Шаг 7: 90

Сортировка - Шаг 8: 93

Отсортированный массив: [22, 41, 43, 60, 80, 83, 90, 93]
Количество инверсий = 21

Время выполнения merge: 0.000010 секунд
Время выполнения custom_sort: 0.000669 секунд

Функция custom_sort выполняется в 0.01 раз быстрее, чем merge.
************************************************
```

Рисунок 1 – Результат выполнения программы

Код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import random as r
from collections import deque
from time import sleep
import time
```

```
def generate_random_list(size, max_val):
    unique_numbers = list(range(max_val))
    r.shuffle(unique_numbers)
    return unique_numbers[:size]


def custom_sort(arr):
    if len(arr) <= 1:
        return arr, 0
    else:
        middle = len(arr) // 2
        left, inv_left = custom_sort(arr[:middle])
        right, inv_right = custom_sort(arr[middle:])
        merged, inv_merge = merge(deque(left), deque(right))
        return merged, inv_left + inv_right + inv_merge


def merge(left, right):
    merged = []
    inv_count = 0
    while left and right:
        if left[0] <= right[0]:
            merged.append(left.popleft())
        else:
            merged.append(right.popleft())
            inv_count += len(left)
    merged.extend(left)
    merged.extend(right)
    return merged, inv_count


def print_creative_header():
    print("*" * 50)
    print("*" * 20 + " Алгоритм сортировки " + "*" * 20)
    print("*" * 50)


def print_array_info(label, array):
    print("\n{}:".format(label))
    print(", ".join(map(str, array)))


def print_animation_step(step, array):
    print("\nСортировка - Шаг {}: {}".format(step, array))
    sleep(0.5)


if __name__ == '__main__':
    print_creative_header()

    custom_array_size = 8
    custom_array_max_val = 100

    original_custom_array = generate_random_list(custom_array_size,
custom_array_max_val)
    current_array = original_custom_array.copy()

    while True:
        print_array_info("Текущий массив", current_array)

        sort_choice = input("\nНачать сортировку? (y/n): ").lower()

        if sort_choice == 'y':
```

```
            start_time_custom_sort = time.time()
            sorted_array, inversions_count = custom_sort(current_array)
            end_time_custom_sort = time.time()
            custom_sort_time = end_time_custom_sort - start_time_custom_sort

            start_time_merge = time.time()
            _, _ = merge(deque(current_array[:custom_array_size // 2]),
deque(current_array[custom_array_size // 2:]))
            end_time_merge = time.time()
            merge_time = end_time_merge - start_time_merge

            print("\nСортировка!")

            for step, array_step in enumerate(sorted_array, 1):
                print_animation_step(step, array_step)

            print("\nОтсортированный массив:", sorted_array)
            print("Количество инверсий =", inversions_count)

            print("\nВремя выполнения merge: {:.6f}
секунд".format(merge_time))
            print("Время выполнения custom_sort: {:.6f}
секунд".format(custom_sort_time))

            if merge_time > 0:
                print("\nФункция custom_sort выполняется в {:.2f} раз
быстрее, чем merge.".format(
                    merge_time / custom_sort_time))
            else:
                print("\nФункция merge выполняется быстрее custom_sort.")
            break
        elif sort_choice == 'n':
            r.shuffle(current_array)
            print("\nМассив изменен.")
        else:
            print("\nОшибка")
            break

    print("*" * 50)
```

Вывод: В ходе выполнения лабораторной работы было исследовано, каким образом лучше подсчитывать инверсии в массиве. Интуитивно понятный метод работает за квадратичное время, поэтому необходимо использовать метод сортировки слиянием с подсчетом инверсий, благодаря которому удалось достичь времени O(n*log(n)).