# 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 №4 дисциплины «Алгоритмизация» Вариант 29

Выполнил: Саенко Андрей Максимович 2 курс, группа ИВТ-б-о-22-1, 09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», заочная форма обучения (подпись) Руководитель практики: Воронкин Р.А., канд. технических наук, доцент кафедры инфокоммуникаций (подпись) Отчет защищен с оценкой Дата защиты Ставрополь, 2023 г.

Тема: Алгоритмы поиска наименьшего и наибольшего элемента в массиве

Цель: Научиться применять метод наименьших квадратов и находить коэффициент парной корреляции, исследовать алгоритм поиска наименьшего или наибольшего элемента в массиве.

### Порядок выполнения работы:

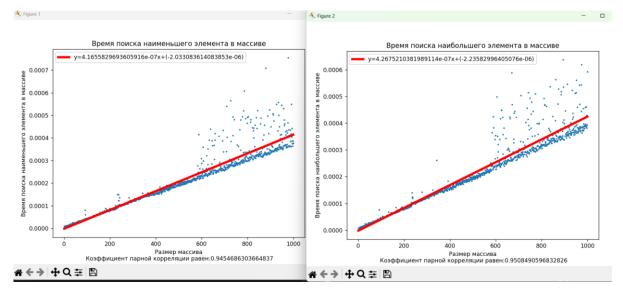


Рисунок 1 – Результат работы программы

## Код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import random
import numpy as np
import timeit
import matplotlib.pyplot as plt
from math import sqrt
def correlation(array of values x,array of values y):
    sigma first = 0
    sigma second = 0
    sigma_third = 0
    average value x = 0
    average_value_y = 0
    sum x = 0
    sum y = 0
    for i in range(len(array_of_values_x)):
        sum x += array of values x[i]
        sum_y += array_of_values_y[i]
    average value x = sum x/len(array of values x)
    average value y = sum y/len(array of values y)
    for i in range(len(array_of_values_x)):
        sigma_first += ((array_of_values_x[i] - average_value_x)*
                         (array_of_values_y[i] - average_value_y))
        sigma_second += (array_of_values_x[i]-average_value_x)**2
        sigma third += (array of values y[i]-average value y) **2
    pair correlation coefficient = (sigma_first/(sqrt(sigma_second) *
                                                  sqrt(sigma third)))
```

```
return pair correlation coefficient
def linear search(array, element):
    for i in range(0,len(array)):
        if array[i] == element:
            return 1
    return 0
array of search time bad = []
amount of elements = []
array of search time avr = []
for i in range(1,1001):
    arr = [0 \text{ for i in } range(0,i)]
    amount of elements.append(i)
    for j in range(0,len(arr)):
        arr[j] = random.randint(500,1000)
    linear search(arr,1)
    search time = (timeit.timeit(lambda: linear search(arr,1), number=50))/50
    print("Время поиска в массиве из ",i,
          " элементов в худшем случае: ", search_time)
    array of search time bad.append(search time)
    search time=(timeit.timeit(lambda: linear search())
        arr, arr[round(len(arr)/2)]), number=50))/50
    print("Время поиска в массиве из ",i,
          " элементов в среднем случае: ", search time, "\n")
    array_of_search_time_avr.append(search_time)
sum_search_time_bad = sum(array_of_search_time_bad)
sum_amt_of_el = sum(amount_of_elements)
sum_of_sqr_amt_of_el = 0
sum mltp amt time = 0
bn = len(amount of elements)
for i in amount of elements:
    sum of sqr amt of el += i*i
for i in range(0,len(array of search time bad)):
    sum mltp amt time += amount of elements[i]*array of search time bad[i]
matrix bad = np.array([[sum_of_sqr_amt_of_el, sum_amt_of_el],
                        [sum_amt_of_el, bn]])
det bad = np.linalg.det(matrix bad)
first mt bad = np.array([[sum mltp amt time, sum amt of el],
                          [sum search time bad, bn]])
first_det_bad=np.linalg.det(first mt bad)
second_mt_bad = np.array([[sum_of_sqr_amt_of_el, sum_mltp_amt_time],
                           [sum_amt_of_el, sum_search_time_bad]])
second det bad = np.linalg.det(second mt bad)
first coefficient bad=first det bad/det bad
second coefficient bad = second det bad/det bad
func bad = []
for i in range(1,1001):
    func bad.append(first coefficient bad*(i)+second coefficient bad)
sum search time avr = sum(array of search time avr)
sum mltp amt time avr = 0
for i in range(0,len(array_of_search time bad)):
    sum mltp amt time avr += amount of elements[i] *array of search time avr[i]
matrix avr = np.array([[sum of sqr amt of el, sum amt of el],
                        [sum amt of el, bn]])
det avr = np.linalg.det(matrix avr)
first mt avr = np.array([[sum mltp amt time avr, sum amt of el],
                          [sum search time avr, bn]])
first det avr = np.linalg.det(first mt avr)
second_mt_avr = np.array([[sum_of_sqr_amt_of_el, sum_mltp_amt_time_avr],
                           [sum_amt_of_el, sum_search_time_avr]])
second det avr = np.linalg.det(second_mt_avr)
first coefficient avr = first det avr/det avr
second coefficient avr = second det avr/det avr
func average = []
for \overline{i} in range(1,1001):
```

```
func_average.append(first_coefficient avr*i +
                        second coefficient avr)
plt.figure(figsize=(8,6))
plt.figure(1)
plt.title("Время поиска элемента в худшем случае")
plt.plot(amount of elements, func bad, color='red', linewidth=4)
plt.scatter(amount of elements, array of search time bad, s=3)
plt.xlabel('Размер массива\n Коэффициент парной корреляции равен:'+
           str(correlation(amount of elements, array of search time bad)))
plt.legend(['y='+str(first coefficient bad)+"x+("+
            str(second coefficient bad)+")"])
plt.ylabel("Время поиска элемента в массиве")
plt.figure(figsize=(8,6))
plt.figure(2)
plt.title("Время поиска элемента в среднем случае")
plt.plot(amount of elements, func average, color='red', linewidth=4)
plt.scatter(amount_of_elements,array_of_search_time_avr,s=3)
plt.xlabel('Размер массива\n Коэффициент парной корреляции равен:'+
           str(correlation(amount of elements, array of search time avr)))
plt.legend(['y='+str(first coefficient avr)+"x+("+
            str(second coefficient avr)+")"])
plt.ylabel("Время поиска элемента в массиве")
plt.show()
```

### Вывод

Время поиска наименьшего и наибольшего элемента в массиве увеличивается линейно при увеличении размера массива. Коэффициент парной корреляции позволяет определить связь между величинами, а метод наименьших квадратов позволяет построить график зависимости на основе экспериментальных данных.