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:

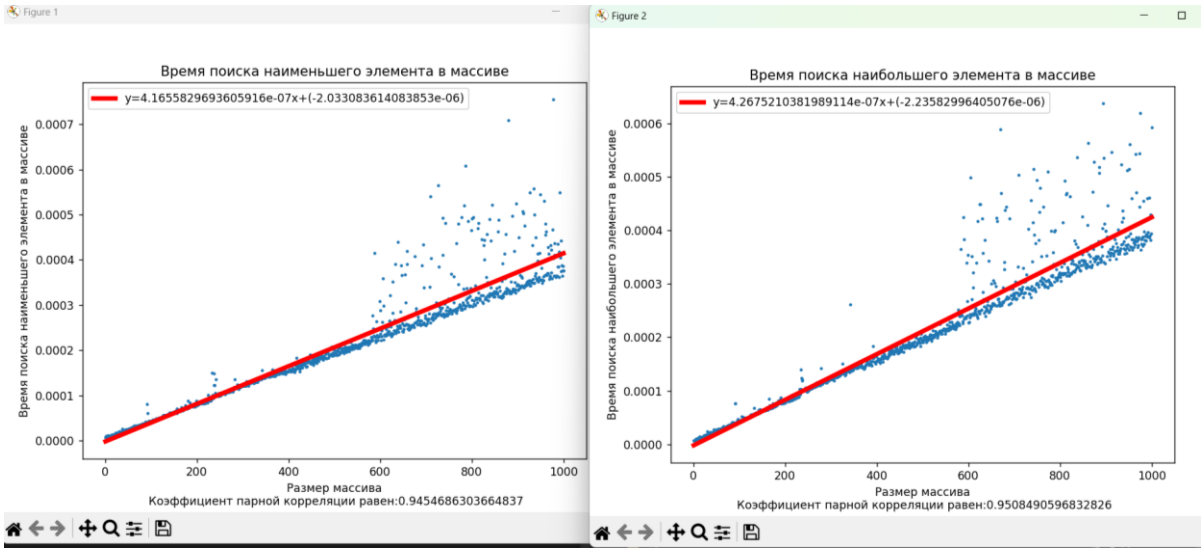

Рисунок 1 – Результат работы программы

Код программы:

```
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 min_element(arr):
```

```
    temp = arr[0]

    for i in range(0,len(arr)):
        if arr[i] < temp:
            temp = arr[i]

    return temp
def max_element(arr):
    temp = arr[0]

    for i in range(0,len(arr)):
        if arr[i] > temp:
            temp = arr[i]

    return temp
arr_time_min = []
x = []
arr_time_max = []
for i in range(1,1001):
    arr = [0 for i in range(0,i)]
    x.append(i)
    for j in range(0,len(arr)):
        arr[j] = random.randint(500,1000)
    search_time = (timeit.timeit(lambda: min_element(arr), number = 50))/50
    print("Время поиска наименьшего элемента в массиве из ",i,
          " элементов: ",search_time)
    arr_time_min.append(search_time)
    search_time = (timeit.timeit(lambda: max_element(arr), number = 50))/50
    print("Время поиска наибольшего элемента в массиве из ",i,
          " элементов: ",search_time,"\n")
    arr_time_max.append(search_time)
sum_arr_time = sum(arr_time_min)
sum_x = sum(x)
sum_sqr_x = 0
sum_mltp_x_arr_time_min = 0
bn = len(x)
for i in x:
    sum_sqr_x += i*i
for i in range(0,len(arr_time_min)):
    sum_mltp_x_arr_time_min += x[i]*arr_time_min[i]
matrix = np.array([[sum_sqr_x, sum_x],
                   [sum_x, bn]])
det = np.linalg.det(matrix)
first_matrix_kramer = np.array([[sum_mltp_x_arr_time_min, sum_x],
                                [sum_arr_time, bn]])
first_det=np.linalg.det(first_matrix_kramer)
second_matrix_kramer = np.array([[sum_sqr_x, sum_mltp_x_arr_time_min],
                                 [sum_x, sum_arr_time]])
second_det=np.linalg.det(second_matrix_kramer)
first_coefficient=first_det/det
second_coefficient=second_det/det
func=[]
for i in range(1,1001):
    func.append(first_coefficient*i + second_coefficient)
sum_arr_time_max=sum(arr_time_max)
sum_mltp_x_arr_time_max = 0
for i in range(0,len(arr_time_max)):
    sum_mltp_x_arr_time_max += x[i]*arr_time_max[i]
matrixMax = np.array([[sum_sqr_x, sum_x],
                      [sum_x, bn]])
det_max = np.linalg.det(matrix)
first_matrix_kramer_max = np.array([[sum_mltp_x_arr_time_max, sum_x],
                                    [sum_arr_time_max, bn]])
first_det_max = np.linalg.det(first_matrix_kramer_max)
second_matrix_kramer_max = np.array([[sum_sqr_x, sum_mltp_x_arr_time_max],
                                     [sum_x, sum_arr_time_max]])
second_det_max = np.linalg.det(second_matrix_kramer_max)
```

```
first_coefficient_max = first_det_max/det_max
second_coefficient_max = second_det_max/det_max
func_max = []
for i in range(1,1001):
    func_max.append(first_coefficient_max*i + second_coefficient_max)
plt.figure(figsize=(8,6))
plt.figure(1)
plt.title("Время поиска наименьшего элемента в массиве")
plt.plot(x,func,color='red',linewidth=4)
plt.scatter(x, arr_time_min,s=3)
plt.xlabel("Размер массива\n Коэффициент парной корреляции равен:"
           +str(correlation(x,arr_time_min)))
plt.legend(['y='+str(first_coefficient)+"x+("+str(second_coefficient)+")"])
plt.ylabel("Время поиска наименьшего элемента в массиве")
plt.figure(figsize=(8,6))
plt.figure(2)
plt.title("Время поиска наибольшего элемента в массиве")
plt.plot(x,func_max,color='red',linewidth=4)
plt.scatter(x,arr_time_max,s=3)
plt.xlabel("Размер массива\n Коэффициент парной корреляции равен:"
           +str(correlation(x,arr_time_max)))
plt.legend(["y="+str(first_coefficient_max)+"x+("
            +str(second_coefficient_max)+")"])
plt.ylabel("Время поиска наибольшего 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.