Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное

учреждения высшего образования

Университет науки и технологий МИСИС

«МИСиС»

Отчет по практической работе №1 на тему:

«Апертурное сжатие данных методом линейной интерполяции»

по дисциплине:

«Методы и средства защиты компьютерной информации»

Направление подготовки: 09.03.03

Семестр 5

|  |  |
| --- | --- |
| Выполнил:    Бобров Михаил Алексеевич  БПИ-20-2  (группа)  30.11.2022  (дата)  Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Проверил:  доцент, к. т. н.  Костин Виталий Николаевич  \_\_\_\_\_\_\_\_\_\_  (дата)  \_\_\_\_\_\_\_\_\_\_  (оценка)  Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва 2022

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc120730391)

[Определение входных и выходных данных 4](#_Toc120730392)

[1.1 Входные данные 4](#_Toc120730393)

[1.2 Выходные данные 4](#_Toc120730394)

[Алгоритм сжатия И ВОССТАНОВЛЕНИЯ данных 5](#_Toc120730395)

[2.1 Алгоритм сжатия 5](#_Toc120730396)

[2.2 Алгоритм восстановления 6](#_Toc120730397)

[Блок-схемы алгоритма 7](#_Toc120730398)

[3.1 Блок-схема всего алгоритма 7](#_Toc120730399)

[3.2 Алгоритм сжатия 8](#_Toc120730400)

[3.3 Алгоритм восстановления 9](#_Toc120730401)

[Листинг программы 10](#_Toc120730402)

[Тестирование программы 13](#_Toc120730403)

ВВЕДЕНИЕ

Отчёт составлен по результатам выполнения практической работы на тему «Апертурное сжатие данных методом линейной интерполяции» по дисциплине «Методы и средства защиты компьютерной информации».

Целью работы является создание алгоритма сжатия и восстановления данных методом линейной интерполяции, соответствующего поставленным требованиям, и его реализация на языке программирования Python.

Достижение указанной цели предполагает решение следующих задач:

1. Изучение метода линейной интерполяции;
2. Рассмотрение возможности его применения для данного случая;
3. Составление блок-схем, описывающих работу алгоритма;
4. Проверка корректности работы алгоритма путем его тестирования.

Определение входных и выходных данных

1.1 Входные данные

Требования, установленные преподавателем: ввод точек осуществляется с помощью клавиатуры через консоль. В первой строке задается число точек, в последующих строках вводится индекс точки и затем значение функции в ней. Далее задается значение апертуры. Число точек и апертура задаются целыми числами (int). Значение функции и апертура задаются значениями с плавающей запятой (в консоли точкой) (float), которая дальше сокращается до 4-х точек после запятой.

1.2 Выходные данные

Вывод выходных данных будет реализован двумя способами: посредством консоли и записью в текстовые файлы. Если текстовый файл не существует, то программа его автоматически создает.

Выходными данными, согласно требованиям, будут являться два массива: сжатый и восстановленный.

Сжатый массив – измененный исходный массив, полученный посредством

алгоритма сжатия с заданной погрешностью (т. е. значением апертуры).

Восстановленный массив – ранее сжатый массив, обработанный посредством алгоритма восстановления.

Индекс точки выводится целочисленным значением (int). Значение точки выводится числом с плавающей запятой (float). Значения функции в восстановленном массиве не должны отличаться от исходного набора данных в той же точке более, чем на значение апертуры.

Алгоритм сжатия И ВОССТАНОВЛЕНИЯ данных

2.1 Алгоритм сжатия

В начале работы алгоритма начальная точки функции добавляется в выходной массив, так как не подвергается сжатию.

Дальше происходит цикл в длину входного массива с сокращением на 2 (N-2). Здесь рассматривается интерполированный отрезок между двумя условно обозначенными точками с индексами *i* и *j+1*, где *i* – точка начала отрезка, *j+1* – точка конца и *j* – рассматриваемая точка.

Далее определяется, попадет ли точка с индексом *j* в выходной массив. Если значение функции в этой точки в исходном массиве будет отличаться на значение *y* апертуры и больше от значения функции этой точки в интерполированном отрезке, то данная точка попадет в выходной массив. Значение *y* находится по формуле: y = ((xj – xi)\*(yj+1 – yi))/(xj+1 – xi) + yi

Значение в этой точке нельзя будет в последствии восстановить с исходной погрешностью. Если точка попадает в промежуток апертуры, пропускаем её, оставив точку *i* такой же, а точку *j* устанавливаем следующей.

В конце работы алгоритма в конец сжатого массива добавляется конечная точка входного массива, так как не подвергается сжатию.

2.2 Алгоритм восстановления

В начале работы алгоритма начальная точки функции добавляется в выходной массив.

Дальше происходит цикл в длину входного массива с сокращением на 2 (N-2). Аналогично алгоритму сжатия условно определяется отрезок, где точка начала отрезка – *i,* и точка конца - *j*, где *j = i + 1*. Отрезком является прямая между двумя последовательными точками входного массива. В цикле определяется, существует ли ранее сжатые точки между *i*-ой и *j*-ой сравнением длины отрезка по оси Х и значения шага, где шаг равен значению 1. Если длина отрезка больше длины шага, то восстанавливаются пропущенные точки нахождением каждой из них значением интерполированного отрезка по формуле из алгоритма сжатия.

В конце каждого шага цикла в выходной массив добавляется точка j.

Блок-схемы алгоритма

3.1 Блок-схема всего алгоритма

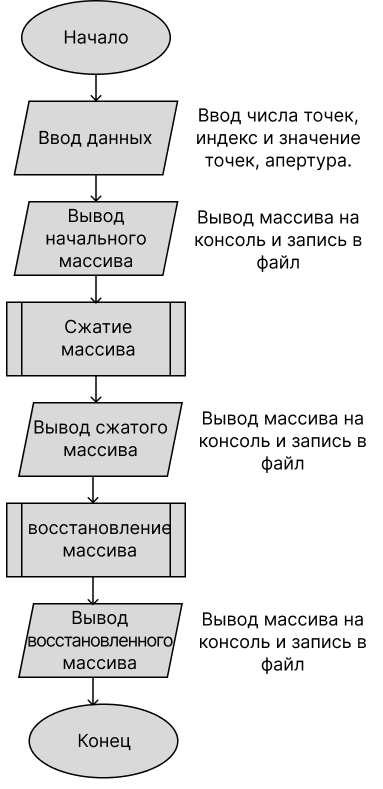


Рисунок 3.1 – блок-схема всего алгоритма

3.2 Алгоритм сжатия

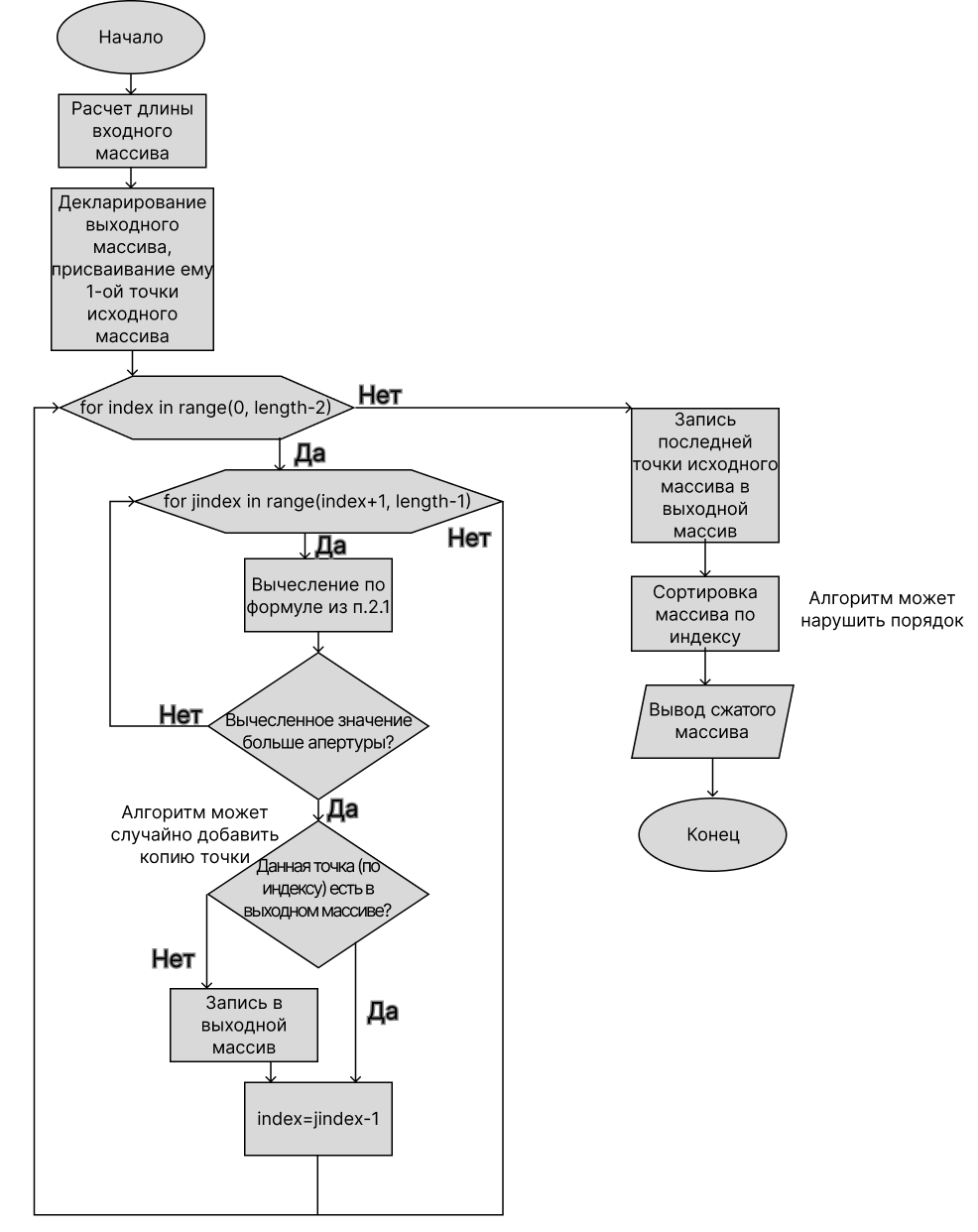


Рисунок 3.2 – алгоритм сжатия

3.3 Алгоритм восстановления

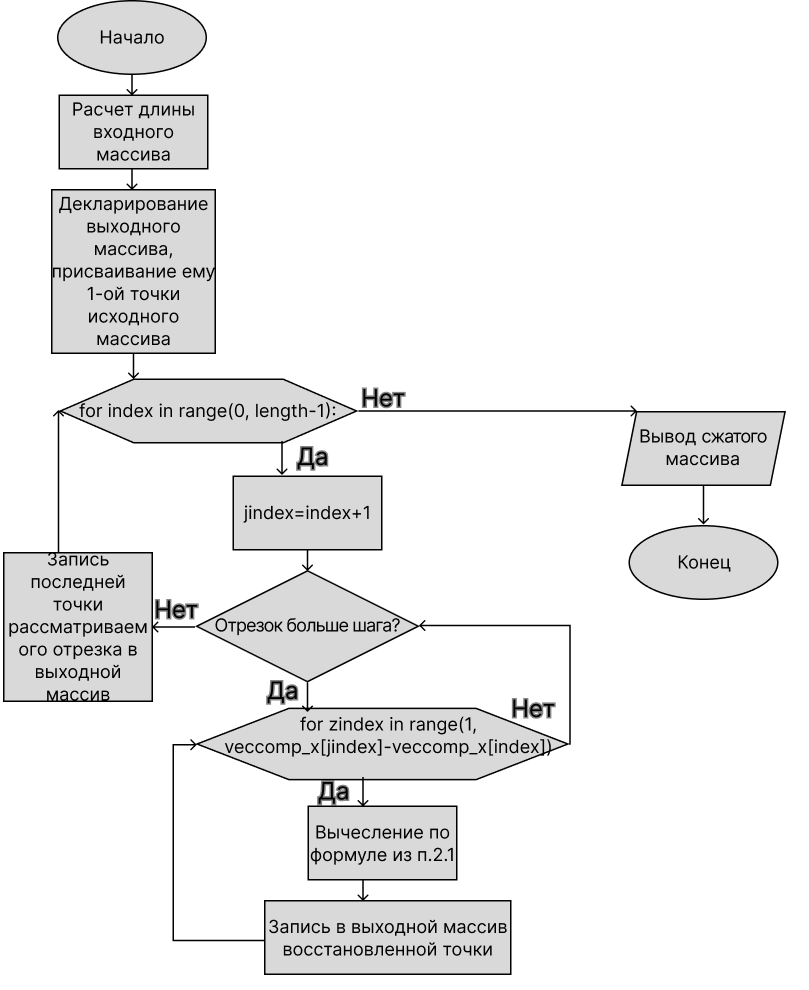


Рисунок 3.3 – алгоритм восстановления

Листинг программы

Ниже приведен листинг программы на языке программирования Python.

vecorig\_y=[]

vecorig\_x=[]

print('\nВведите количество точек:')

length=round(int(input()))

for index in range(0, length):

    print('\nВведите индекс точки №'+str(index+1)+':')

    vecorig\_x.append(int(input()))

    print('\nВведите значение точки №'+str(index+1)+':')

    vecorig\_y.append(round(float(input()), 4))

index=0

lines=[]

for point in vecorig\_x:

    line=''.join(str(vecorig\_x[index]))

    line+=' '

    line+=''.join(str(vecorig\_y[index]))

    lines.append(line)

    index+=1

with open('input.txt', 'w') as ipntsfile:

    ipntsfile.write('\n'.join(lines))

ipntsfile.close

print('\nВведите апертуру:')

apperture=round(float(input()), 4)

print('\nМассив до сжатия:\n'+str(vecorig\_x)+'\n'+str(vecorig\_y))

veccomp\_y=[]

veccomp\_x=[]

#сжатие

veccomp\_x.append(vecorig\_x[0])

veccomp\_y.append(vecorig\_y[0])

for index in range(0, length-2):

    for jindex in range(index+1, length-1):

        check=float((vecorig\_x[jindex]-vecorig\_x[index])\*(vecorig\_y[jindex+1]-vecorig\_y[index])/(vecorig\_x[jindex+1]-vecorig\_x[index]))

        check+=vecorig\_y[index]

        if abs(vecorig\_y[jindex]-check)>=apperture:

            if not vecorig\_x[jindex] in veccomp\_x:

                veccomp\_x.append(vecorig\_x[jindex])

                veccomp\_y.append(vecorig\_y[jindex])

            index=jindex-1

            break

veccomp\_x.append(vecorig\_x[-1])

veccomp\_y.append(vecorig\_y[-1])

#сортировка так как алгоритм может перепутать порядок

length=len(veccomp\_x)

for index in range(0, length-1):

    if veccomp\_x[index] > veccomp\_x[index+1]:

        veccomp\_x[index], veccomp\_x[index+1] = veccomp\_x[index+1], veccomp\_x[index]

        veccomp\_y[index], veccomp\_y[index+1] = veccomp\_y[index+1], veccomp\_y[index]

print('\nМассив после сжатия:\n'+str(veccomp\_x)+'\n'+str(veccomp\_y))

index=0

lines=[]

for point in veccomp\_x:

    line=''.join(str(veccomp\_x[index]))

    line+=' '

    line+=''.join(str(veccomp\_y[index]))

    lines.append(line)

    index+=1

with open('cpoints.txt', 'w') as cpntsfile:

    cpntsfile.write('\n'.join(lines))

cpntsfile.close

#восстановление

length=len(veccomp\_x)

vecrec\_x=[veccomp\_x[0]]

vecrec\_y=[veccomp\_y[0]]

for index in range(0, length-1):

    jindex=index+1

    if(veccomp\_x[jindex]-veccomp\_x[index]>1):

        for zindex in range(1, veccomp\_x[jindex]-veccomp\_x[index]):

            y = veccomp\_y[index]+(zindex\*(veccomp\_y[jindex]-veccomp\_y[index])/(veccomp\_x[jindex]-veccomp\_x[index]))

            vecrec\_x.append(veccomp\_x[index]+zindex)

            vecrec\_y.append(round(y, 4))

    vecrec\_x.append(veccomp\_x[jindex])

    vecrec\_y.append(veccomp\_y[jindex])

print('\nМассив после востановления:\n'+str(vecrec\_x)+'\n'+str(vecrec\_y))

index=0

lines=[]

for point in vecrec\_x:

    line=''.join(str(vecrec\_x[index]))

    line+=' '

    line+=''.join(str(vecrec\_y[index]))

    lines.append(line)

    index+=1

with open('rpoints.txt', 'w') as rpntsfile:

    rpntsfile.write('\n'.join(lines))

rpntsfile.close

Тестирование программы

Для тестирования программы выберем следующий набор значений некоторой функции: [0; 10] [1; 3] [2; 6] [3; 9] [4; 12] [5; 9.5] [6; 14] [7; 7.5] [8; 2] [9; 5.5]

Значение апертуры равно 1.

Изображение выглядит как текст, экран, снимок экрана

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5.1 и 5.2 – Ввод данных

Сжатие было произведено от 10 точек до 7, основные тенденции поведения функции не нарушены. Значение функции для каждой точки из восстановленного массива не отличается более чем на апертуру (1) от исходного набора, что свидетельствует о корректной работе программы.

Текстовые файлы по указанным путям будет иметь следующее содержимое, аналогично тому, что было выведено на консоль:

Изображение выглядит как текст, электроника

Автоматически созданное описание Изображение выглядит как текст, электроника

Автоматически созданное описание Изображение выглядит как текст, электроника

Автоматически созданное описание

Рисунок 5.3, 5.4 и 5.5 – текстовые файлы с исходными данными, сжатым массивом и восстановленным массивом.