Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Домашняя работа №1 по курсу "Анализ алгоритмов"

Тема Графовые модели алгоритмов

Студент Артюхин Н.П.

Группа ИУ7-51Б

Преподаватели Волкова Л.Л.

Москва — 2022 г.

Оглавление

1 Технологическая часть

1.1 Выбор языка программирования

При выполнении домашнего задания использовался язык программирования — Python [1].

1.2 Исходный код программы

В листинге 1.1 представлена реализация поиска расстояния Дамерау-Левенштейна с заполнением матрицы расстояний.

Листинг 1.1 – Реализация итеративного алгоритма поиска расстояния Дамерау-Левенштейна с заполнением матрицы расстояний

```
def damerau_lowenstein_dist_non_recursive(s1, s2, flag=False):
    n = len(s1) + 1                                                    # 1
    m = len(s2) + 1                                                    # 2

    matrix = [[0 for i in range(m)] for j in range(n)]                 # 3

    for j in range(0, m):                                              # 4
        matrix[0][j] = j                                               # 5
    for i in range(0, n):                                              # 6
        matrix[i][0] = i                                               # 7

    for i in range(1, n):                                              # 8
        for j in range(1, m):                                          # 9
            insert = matrix[i][j - 1] + 1                              # 10
            delete = matrix[i - 1][j] + 1                              # 11
            tmp = int(s1[i - 1] == s2[j - 1])                          # 12
            replace = matrix[i - 1][j - 1] + tmp                       # 13
            matrix[i][j] = min(insert, delete, replace)                # 14
            if i > 1 and j > 1 and
            s1[i - 1] == s2[j - 2] and s1[i - 2] == s2[j - 1]:         # 15
                exchange = matrix[i - 2][j - 2] + 1                    # 16
                matrix[i][j] = min(matrix[i][j], exchange)             # 17

    return matrix[n - 1][m - 1]
```

1.3 Графовые модели алгоритма

На рисунке 1.1 представлен граф управления алгоритма.

Рисунок 1.1 – Граф управления алгоритма

На рисунке 1.2 представлен информационный граф алгоритма.

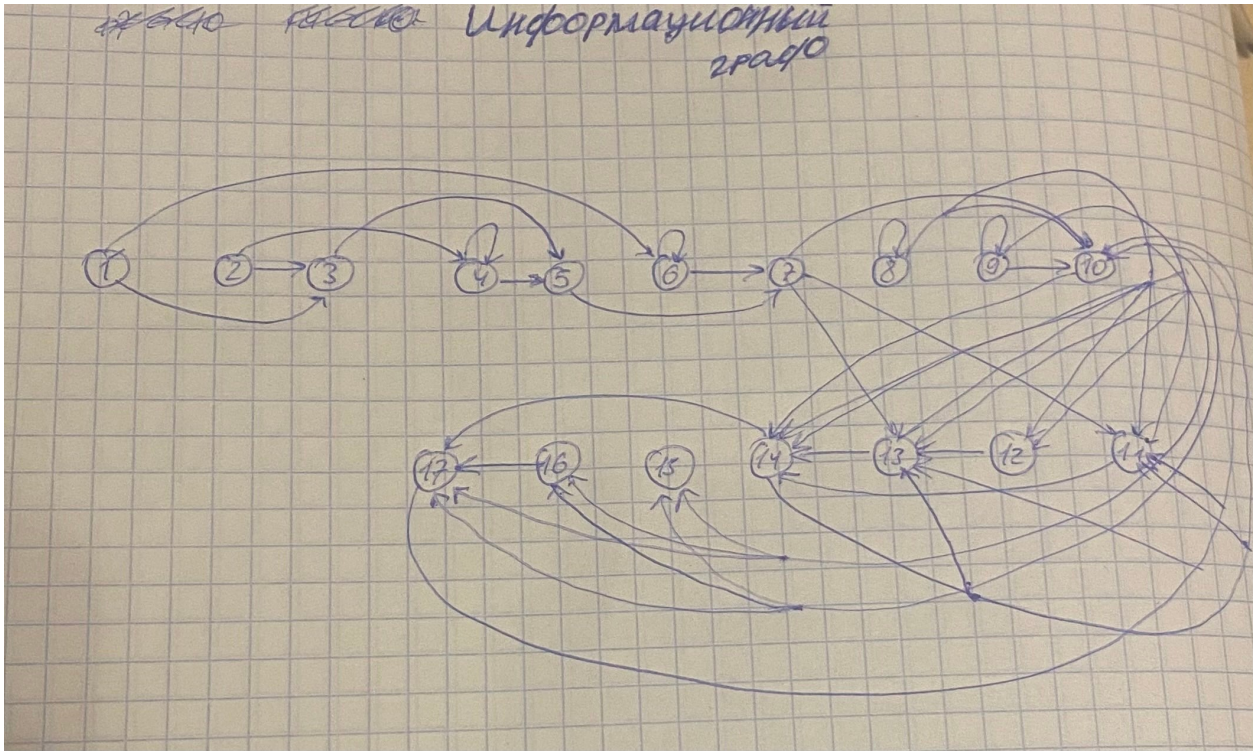

Рисунок 1.2 – Информационный граф алгоритма

На рисунке 1.3 представлена операционная история алгоритма для случая, когда условие в строке 15 листинга 1.1 выполняется, то есть принимает значение "истина". В противном случае (значение "ложь"в условии) во вложенном цикле не будут выполняться строки 16 и 17 листинга 1.1, не будут задействоваться узлы 16 и 17 в операционной истории (узел 15 станет последним в цикле), в остальном она никак не изменится.

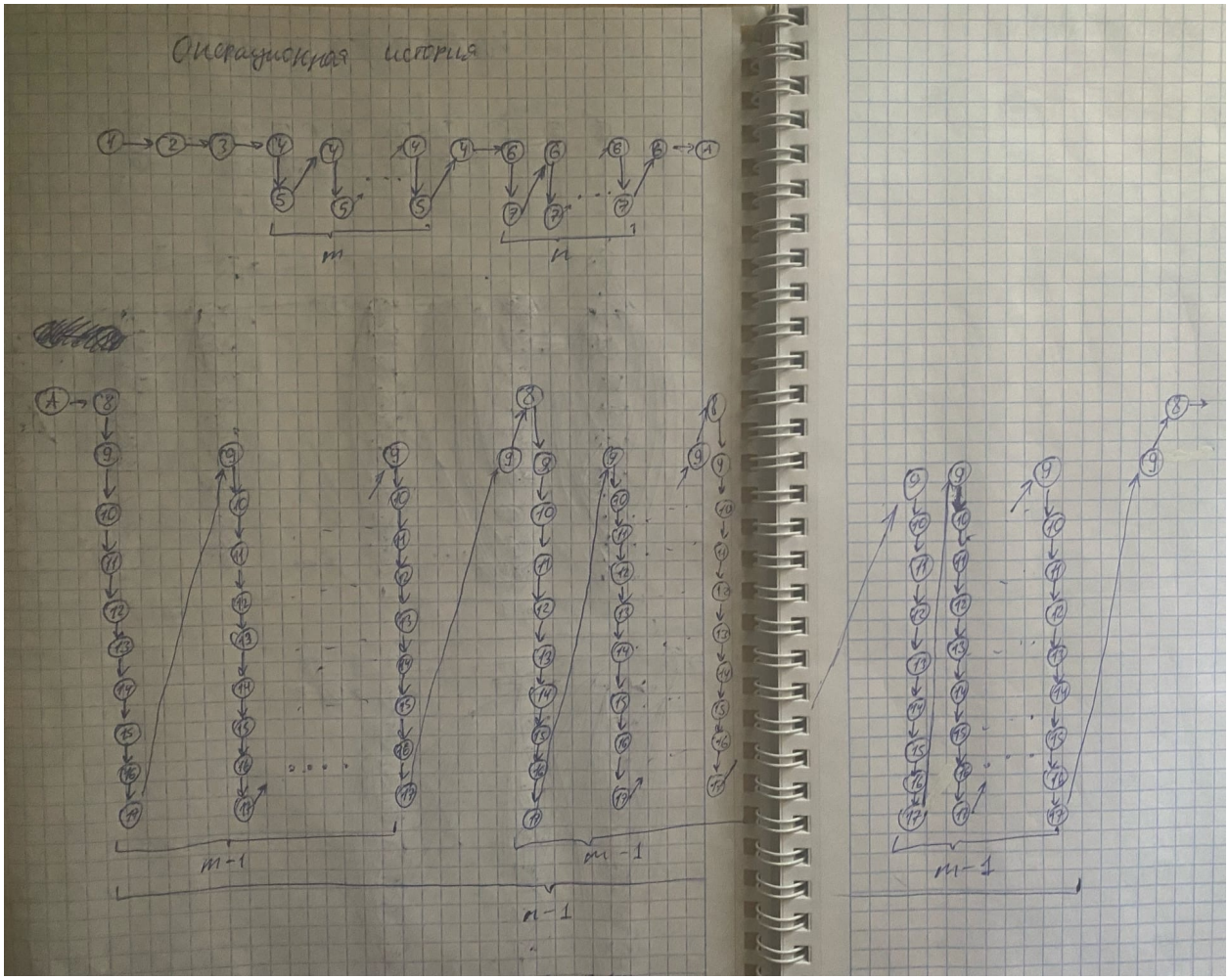

Рисунок 1.3 – Операционная история алгоритма

На рисунке 1.4 представлена информационная история алгоритма для случая, когда условие в строке 15 листинга 1.1 выполняется, то есть принимает значение "истина". В противном случае (значение "ложь"в условии) во вложенном цикле не будут выполняться строки 16 и 17 листинга 1.1, не будут задействоваться узлы 16 и 17 в информационной истории (узел 15 станет последним в цикле, измененная матрица будет передаваться на следующую итерацию цикла с узла 14 вместо узла 17), в остальном она никак не изменится.

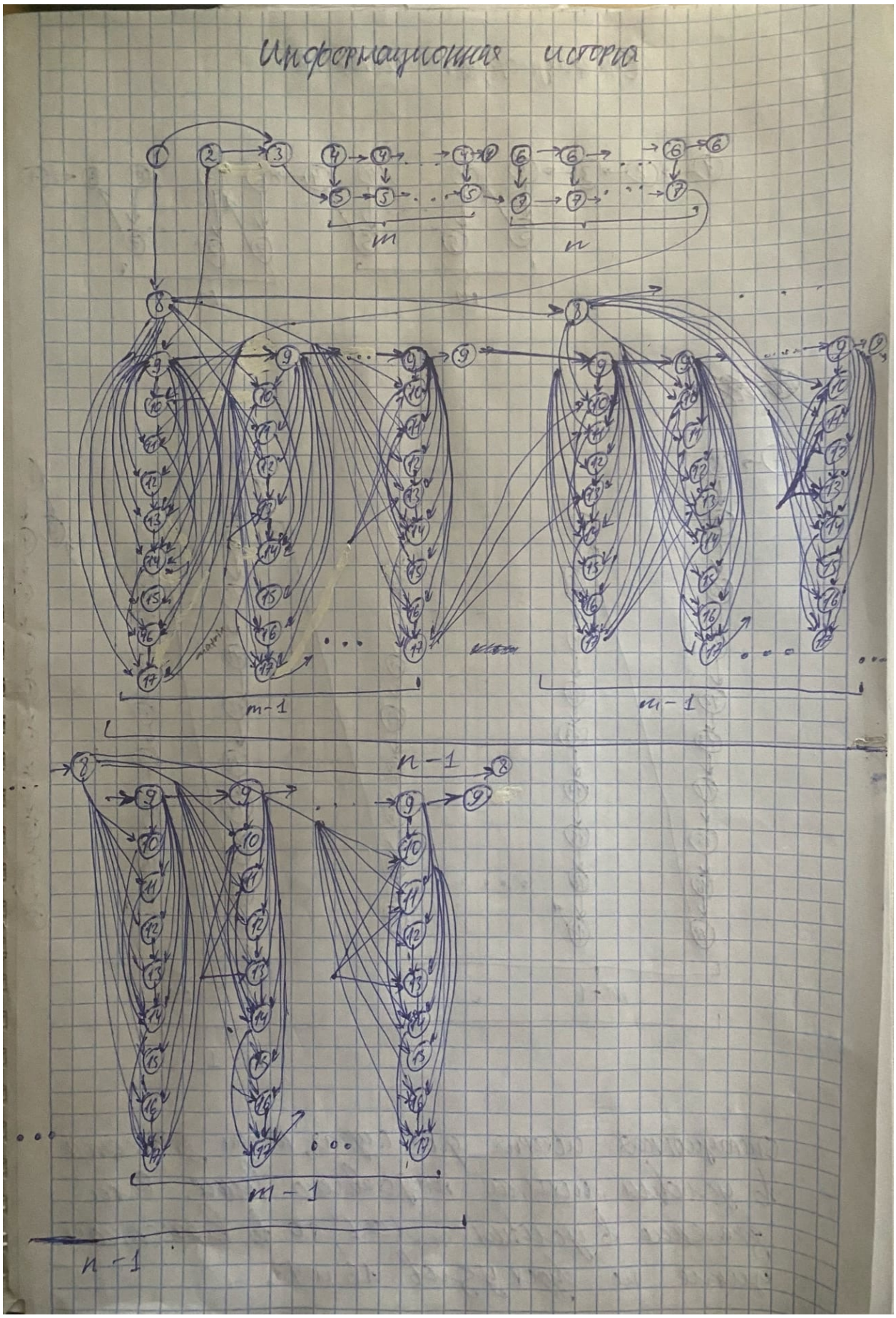

Рисунок 1.4 – Информационная история алгоритма

Список использованных источников

1. Лутц, Марк. Изучаем Python, том 1, 5-е изд. Пер. с англ. — СПб.: ООО “Диалектика”, 2019 — С. 832.