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»

ОТЧЕТ

по домашняя работа № 1

по курсу «Анализ алгоритмов»

на тему: «Трудоёмкость сортировок»

Студент ИУ7-54Б (Группа) _____ (Подпись, дата) Булдаков М. (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) Волкова Л. Л. (И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

1 Описание задачи

В качестве реализуемого алгоритма — вычисление расстояния Левенштейна.

1.1 Графовые модели программы

Программа представлена в виде графа: набор вершин и множество соединяющих их направленных дуг.

Выделяют 2 типа дуг:

1) операционное отношение — по передаче управления;

2) информационное отношение — по передаче данных.

Граф управления — модель, в который **вершины** —операторы, **дуги** — операционные отношения.

Информационный граф — модель, в которой **вершины**: операторы, **дуги** — информационные отношения.

Операционная история — модель, в которой **вершины**: срабатывание операторов, **дуги** — операционные отношения.

Информационная история — модель, в которой **вершины**: срабатывание операторов, **дуги** — информационные отношения.

Графы более компактны, однако менее информативны, чем истории. Истории менее комактны, однако более информативны, чем графы.

На листинге 1.1 приведена реализации функции, вычисляющей расстояние Левенштейна.

Листинг 1.1 – Реализация алгоритма вычисления расстояния Левенштейна

```
def levenstein():
    s1 = input("Введите 1-ую строку: ")  # 1
    s2 = input("Введите 2-ую строку: ")  # 2

    len1 = len(s1)  # 3
    len2 = len(s2)  # 4

    M = [[0] * (len2 + 1) for _ in range(len1 + 1)]  # 5

    for i in range(len1 + 1):  # 6
        M[i][0] = i  # 7

    for j in range(len2 + 1):  # 8
        M[0][j] = j  # 9

    for i in range(1, len1 + 1):  # 10
        for j in range(1, len2 + 1):  # 11
            A = M[i - 1][j] + 1  # 12
            D = M[i][j - 1] + 1  # 13
            C = M[i - 1][j - 1]  # 14

            if s1[i - 1] != s2[j - 1]:  # 15
                C += 1  # 16

            M[i][j] = min(A, D, C)  # 17

    return M[-1][-1]
```

На рисунке 1.1 представлен граф управления. На рисунке 1.2 представлен информационный граф.

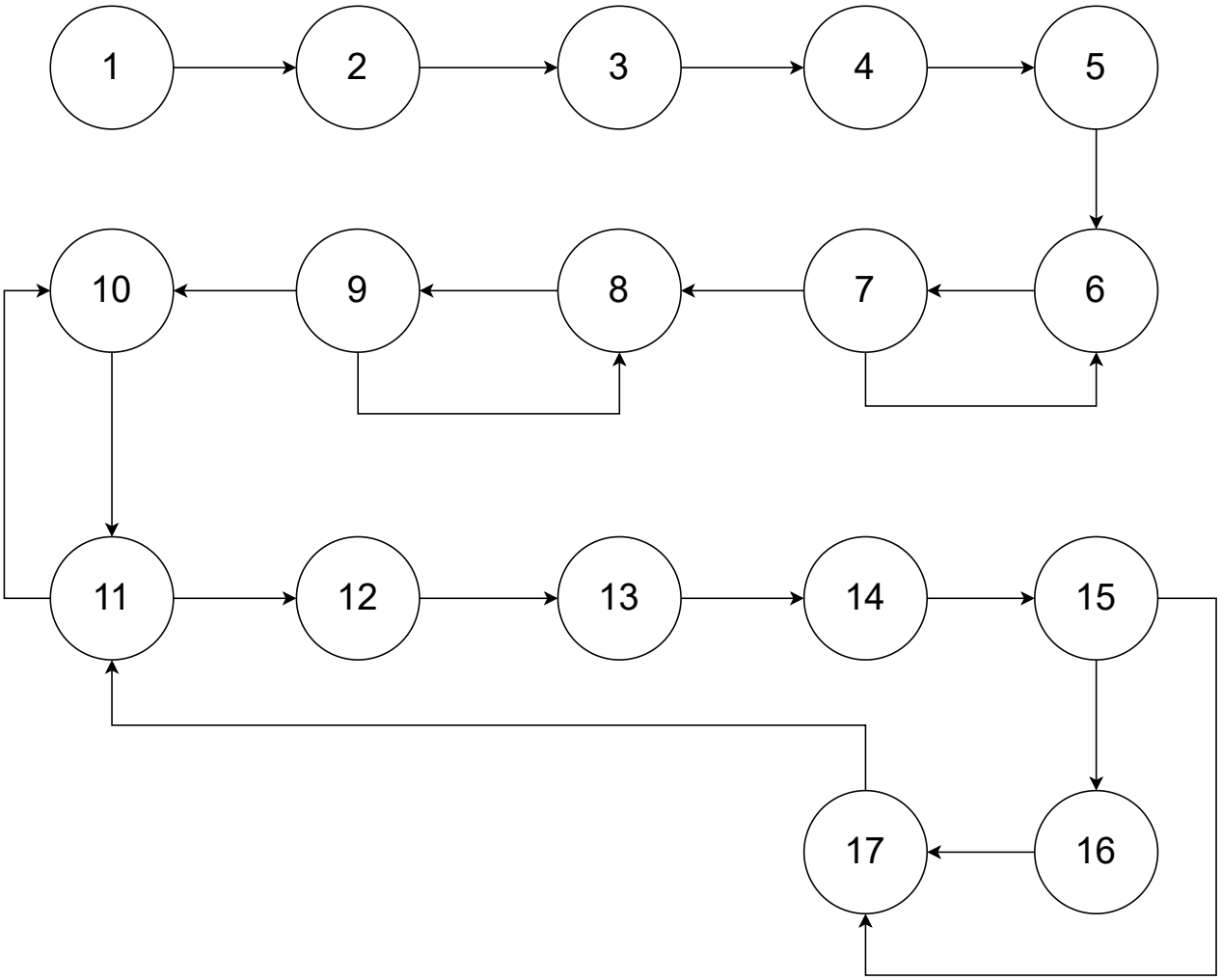

Рисунок 1.1 – Граф управления

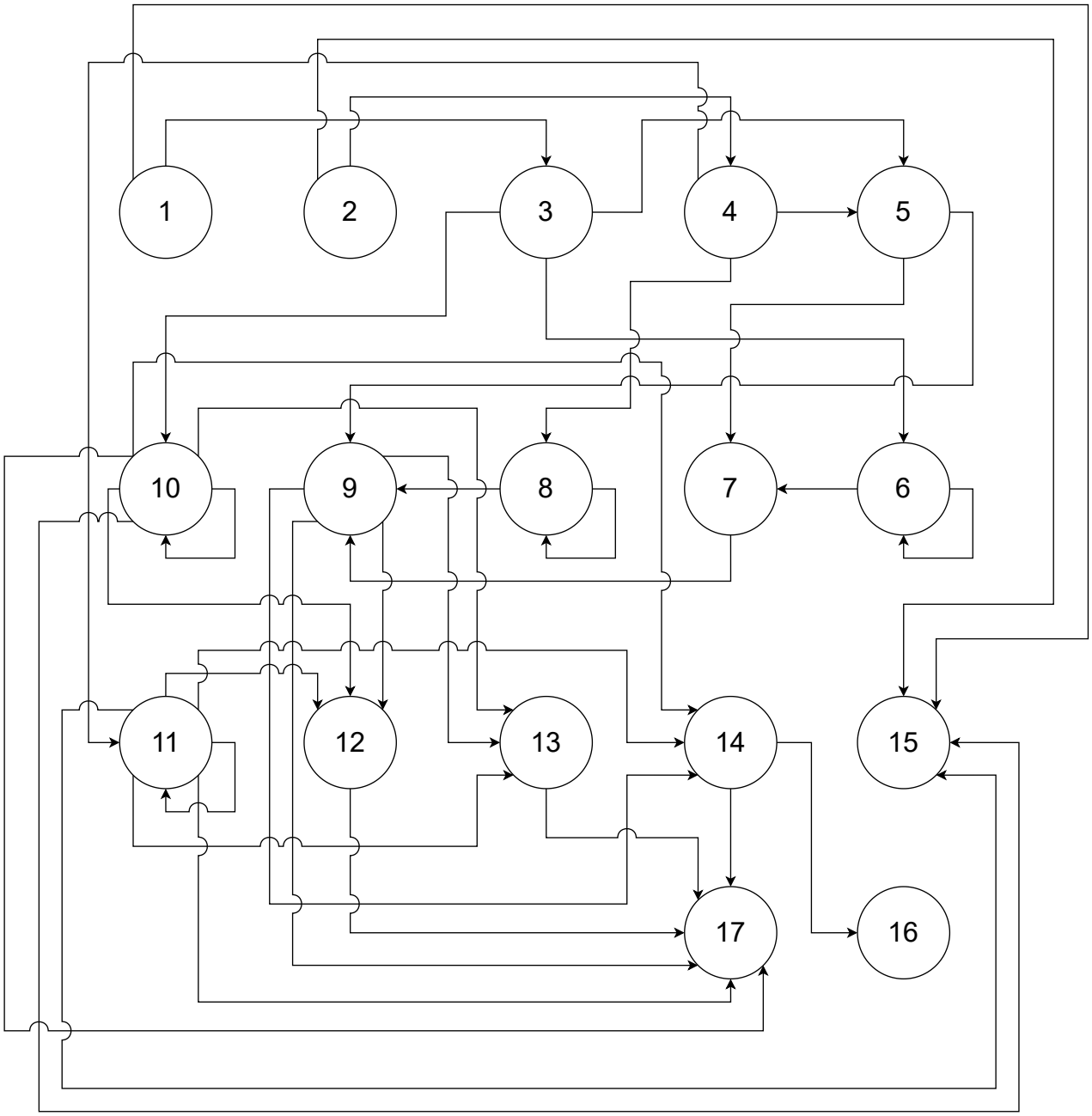

Рисунок 1.2 – Информационный граф

Приведем графы для строк «аба» и «ааа». На рисунке 1.3 представлена операционная история. На рисунке 1.4 представлена информационная история.

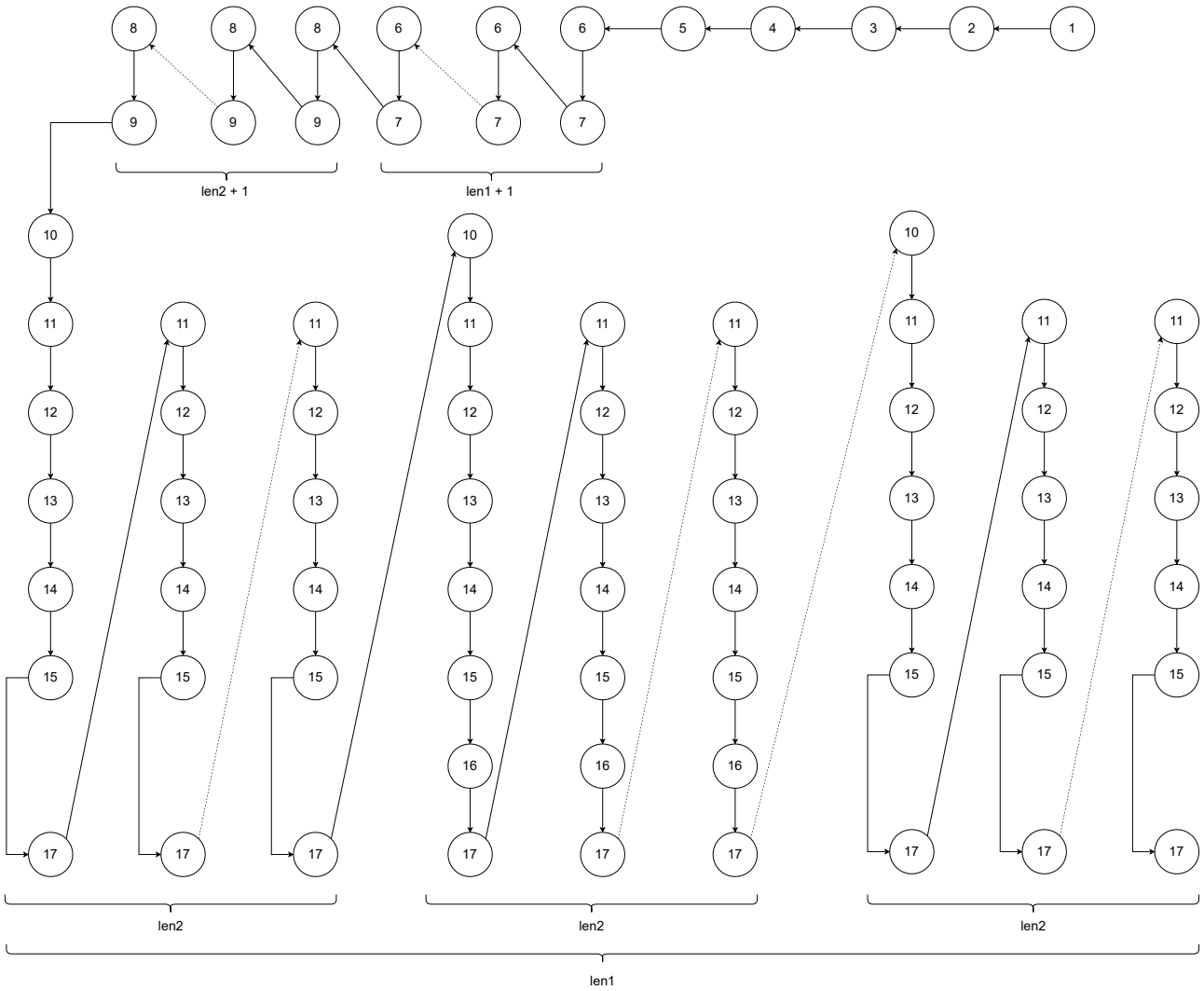

Рисунок 1.3 – Операционная история

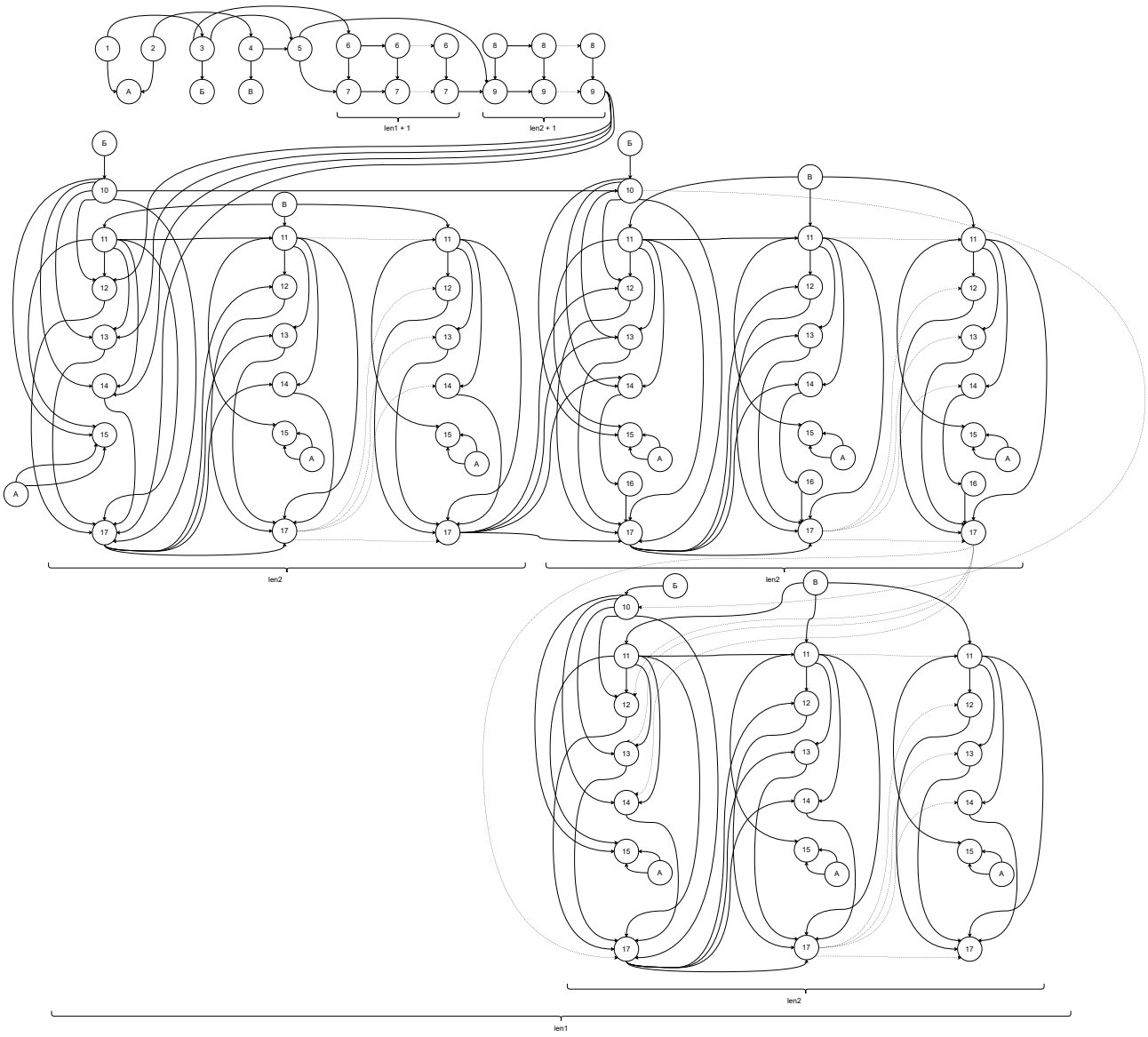

Рисунок 1.4 – Информационная история

Вывод

В реализуемом алгоритме наблюдается скошенный параллелизм. Можно вычислять диагонали, параллельные побочной в матрице Левенштейна, в различных потоках, поскольку элементы диагонали не зависят друг от друга.