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 домашней работе по курсу «Анализ алгоритмов»

Тема Графовые модели алгоритмов ‘

Студент Пискунов П.

Группа ИУ7-56Б

Преподаватель Волкова Л.Л., Строганов Д.В.

Москва — 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной работе будут расмотрены графовые модели алгоритмов.

Задание

Описать четырьмя графовыми моделями (граф управления, информационный граф, операционная история, информационная история) последовательный алгоритм либо фрагмент алгоритма, содержащий от 15 значащих строк кода и от двух циклов, один из которых является вложенным в другой.

Вариант – Исправление орфографических ошибок в тексте.

1 Аналитическая часть

1.1 Графовые модели программы

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

- Вершины – процедуры, циклы, линейные участки, операторы, итерации циклов, срабатывание операторов и т.д.
- Дуги – отражают связь (отношение между вершинамы).

Выделяют 2 типа отношений:

- операционное отношение – по передаче управления;
- 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.

2 Выполнение

2.1 Выбор языка программирования

Для выполнения домашнего задания был выбран язык C++.

2.2 Код программы

В листинге 2.1 представлен код программы.

```
1  void single_thread(const string &input_text, const vector<string> &
    dictionary)
2  {
3    string temp_text = input_text; // 1
4    istringstream iss(temp_text); // 2
5    vector<string> words(istream_iterator<string>{iss}, istream_iterator<
   string>{}); // 3
6    for (auto &word : words) // 4
7    {
8      int min_distance = numeric_limits<int>::max(); // 5
9      string best_match = word; // 6
10
11     for (auto &dict_word : dictionary) // 7
12     {
13       int distance = alg_lev(word, dict_word); // 8
14
15       if (distance < min_distance) // 9
16       {
17         min_distance = distance; // 10
18         best_match = dict_word; // 11
19       }
20     }
21     size_t pos = temp_text.find(word); // 12
22
23     while (pos != string::npos) // 13
24     {
25       temp_text.replace(pos, word.length(), best_match); // 14
26       pos = temp_text.find(word, pos + best_match.length()); // 15
27     }
28   }
29 }
```

Листинг 2.1 – Алгоритм исправления орфографических ошибок в тексте

2.3 Графовые модели программы

2.3.1 Граф управления

На рисунке 2.1 представлен граф управления.

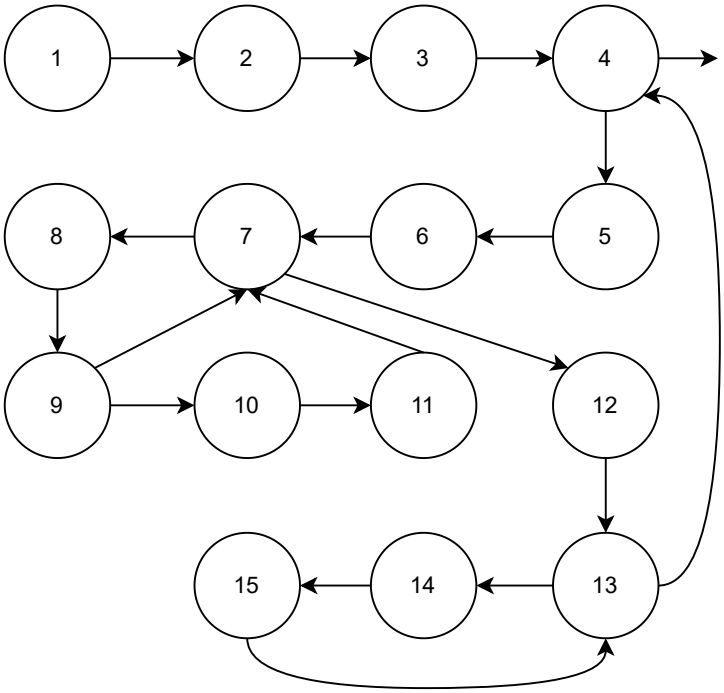

Рисунок 2.1 – Граф управления

2.3.2 Информационный граф

На рисунке 2.2 представлен информационный граф.

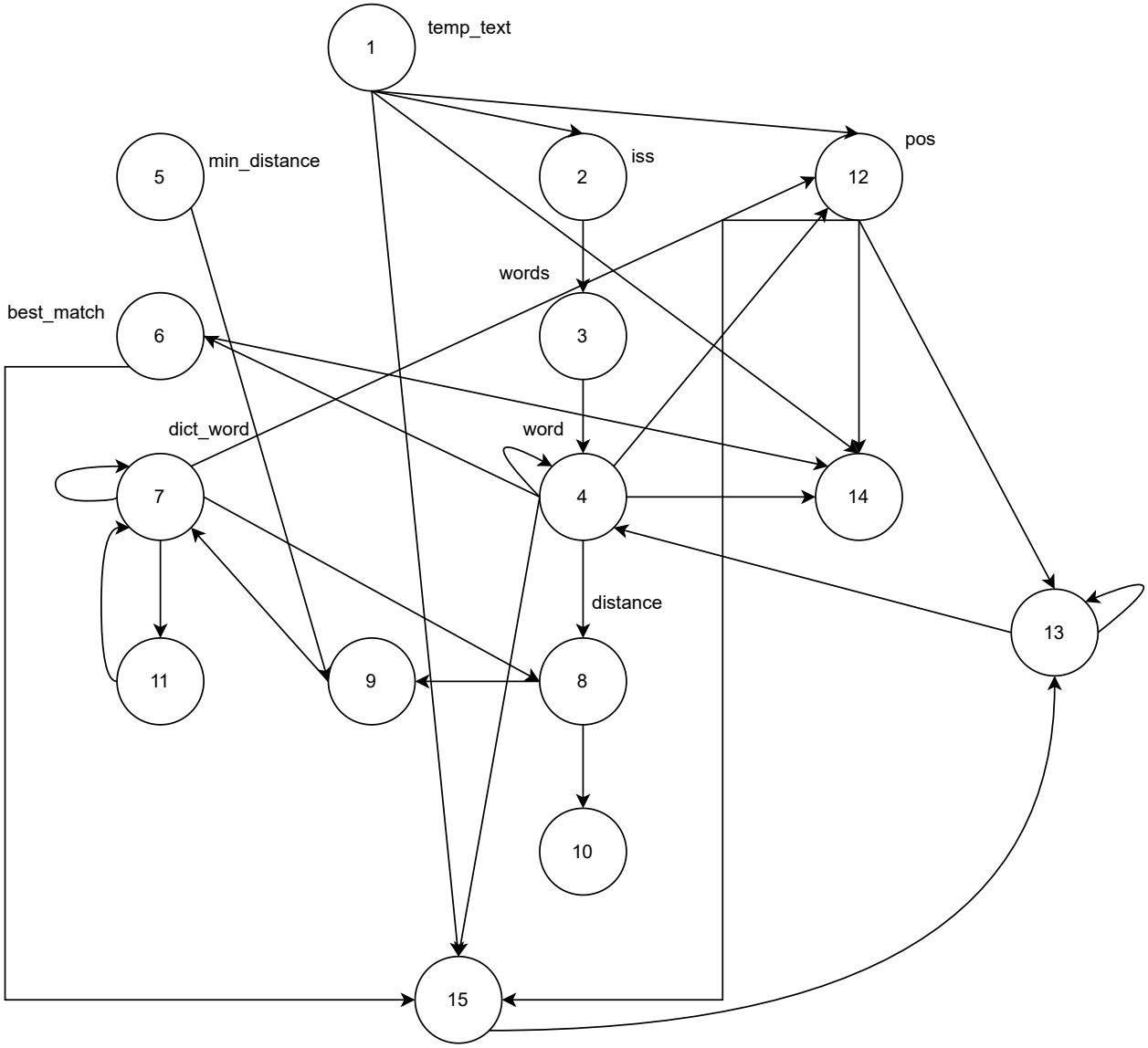

Рисунок 2.2 – Информационный граф

2.3.3 Операционная история

Дан следующий короткий текст «hellw rorld».
На русунке 2.3 представленна операционная история.

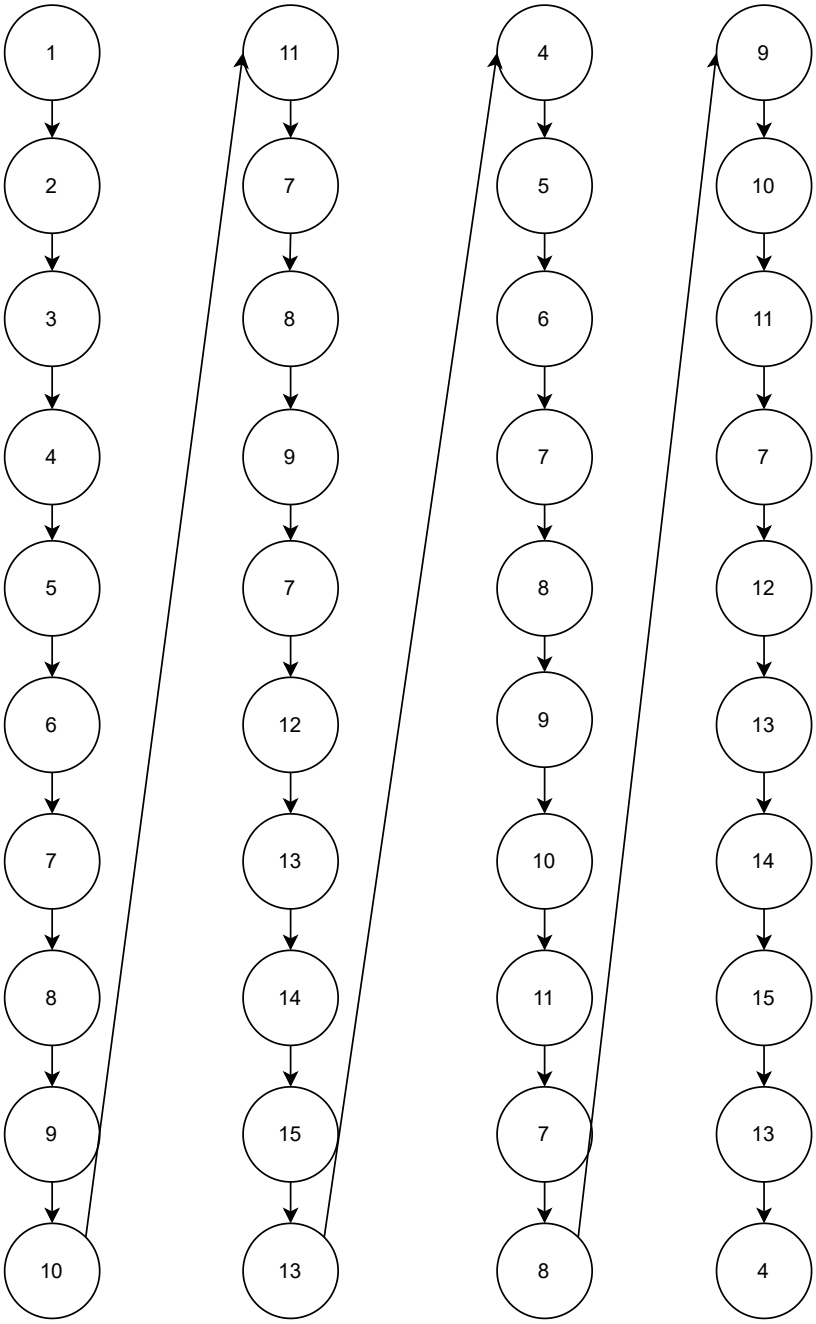

Рисунок 2.3 – Операционная история

2.3.4 Информационная история

На рисунке 2.4 представленна информационная история.

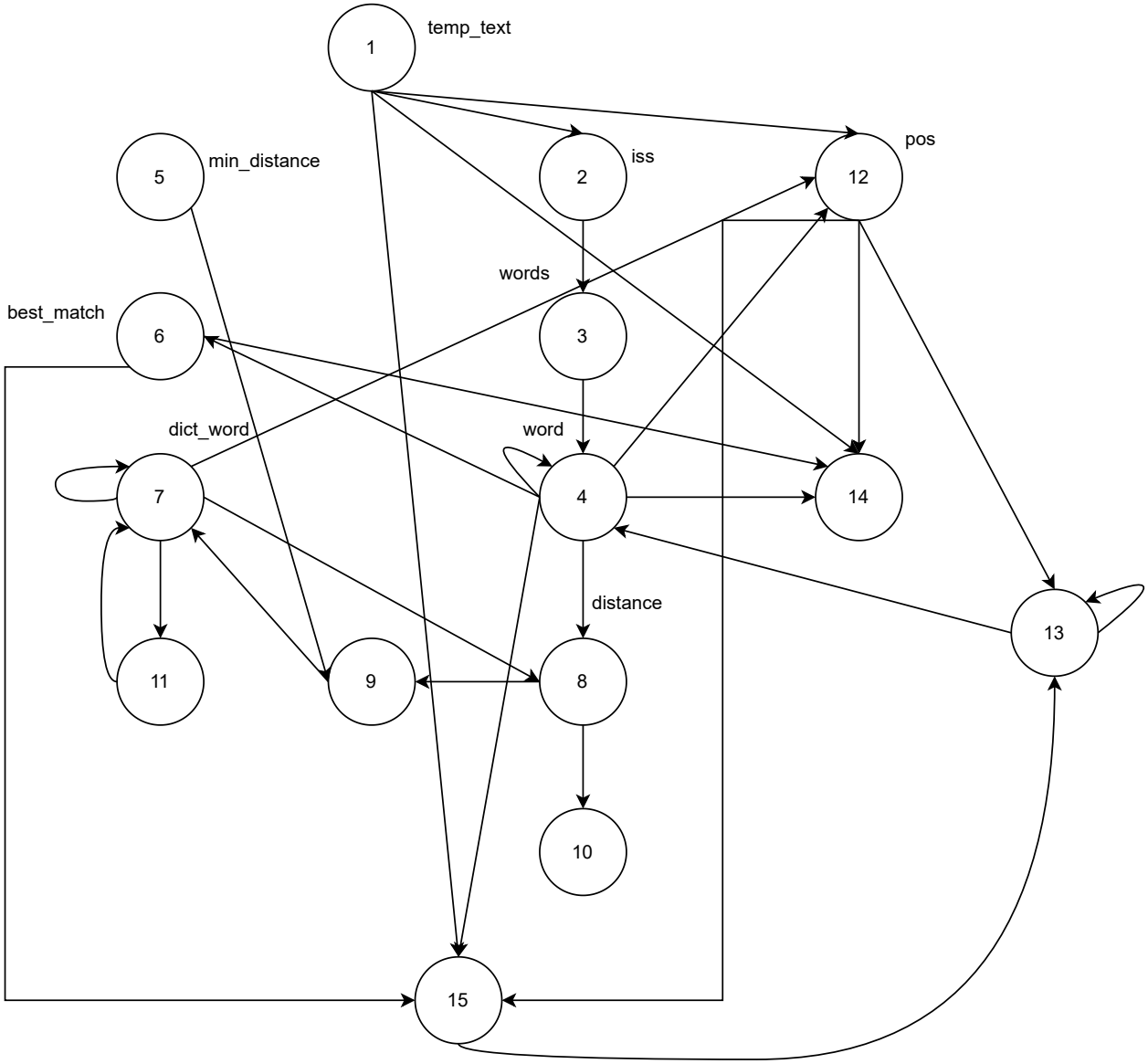

Рисунок 2.4 – Информационная история

Возможность распараллеливания

Можно разделить текст, который содержит орфографические ошибки на несколько частей, и запустить каждую часть в отдельном потоке, а затем объединить результаты.