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 ЛАБОРАТОРНОЙ РАБОТЕ №7

Группа ИУ7-51Б

Тема работы **Графовые модели**

Студент Баранов Николай Алексеевич

Преподаватель Волкова Лилия Леонидовна

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – получение навыка работы с графовыми моделями.

Задачи работы:

— выбрать фрагмент кода для анализа и входные данные для построения информационной и операционной историй;

— построить граф управления, информационный граф, операционную историю и информационную историю;

— сделать вывод о применимости графовых моделей.

1 Выбор фрагмента кода и входных данных

В листингах 1.1-1.2 представлен выбранный для анализа фрагмент кода. В комментариях строкам присвоены номера для обозначения в графовых моделях.

Листинг 1.1 – Выбранный для анализа фрагмент кода (начало)

```
var queuesWaiting = (double[]) null; // 1
var worktime = .0; // 2
var stageTimes = new LinkedHashMap<String, Double>(); // 3
var records = 0; // 4
var queues = 0; // 5
for (var value : tasks.values()) { // 6
    value.sort(Comparator.comparing(a -> a[0])); // 7
    if (records == 0) { // 8
        records = value.size(); // 9
        queues = records >> 1; // 10
        queuesWaiting = new double[queues]; // 11
    }
    var times = new double[records]; // 12
    for (var i = 0; i < records; ++i) { // 13
        times[i] = Double.parseDouble(value.get(i)[0]); // 14
    }
    for (var i = 0; i < records; i += 2) { // 15
        queuesWaiting[i >> 1] += times[i + 1] - times[i]; // 16
        if (i > 0) { // 17
            var stage = value.get(i)[2].split("_")[1]; // 18
```

Листинг 1.2 – Выбранный для анализа фрагмент кода (окончание)

```
                stageTimes.put(stage, stageTimes.getOrDefault(stage,
                    .0) + times[i] - times[i - 1]); // 19
            }
        }
        worktime += times[records - 1] - times[0]; // 20
}
```

Фрагмент взят из статического метода $getStats()$ класса $LogsMerger$. Метод $getStats()$ принимает на вход словарь $tasks$, содержащий историю для каждой задачи в виде списка массивов. В таблице 1.1 представлены ключи и значения в словаре $tasks$, которые использовались для построения операционной и информационной историй.

Таблица 1.1 – Ключи и значения, хранящиеся в словаре tasks

Ключ	Значение
1	$[[0.0, 1, created], [0.01, 1, start\_first], [0.05, 1, end\_first],$ $[0.06, 1, start\_second], [0.09, 1, end\_second], [0.1, 1, destroyed]]$
2	$[[0.01, 2, created], [0.06, 2, start\_first], [0.11, 2, end\_first],$ $[0.12, 2, start\_second], [0.18, 2, end\_second], [0.19, 2, destroyed]]$
3	$[[0.02, 3, created], [0.12, 3, start\_first], [0.15, 3, end\_first],$ $[0.19, 3, start\_second], [0.22, 3, end\_second], [0.23, 3, destroyed]]$

2 Графовые модели

Информационное отношение — отношение по передаче данных между вершинами такое, что во второй вершине используются данные, полученные в первой вершине. Информационный граф — граф, вершинами которого являются строки кода, а рёбрами — информационное отношение. Информационная история — граф, вершинами которого являются срабатывания операторов, команд и строк кода, а рёбрами — информационное отношение. Операционное отношение — отношение по передаче управления между вершинами такое, что вторая вершина будет выполнена после первой. Граф управления — граф, вершинами которого являются строки кода, а рёбрами — операционное отношение. Операционная история — граф, вершинами которого являются

срабатывания операторов, команд и строк кода, а рёбрами — операционное отношение. Операционная история строго линейна.

На рисунке 2.1 представлен информационный граф. На рисунке 2.2 представлена информационная история. На рисунке 2.3 представлен граф управления. На рисунке 2.4 представлена операционная история.

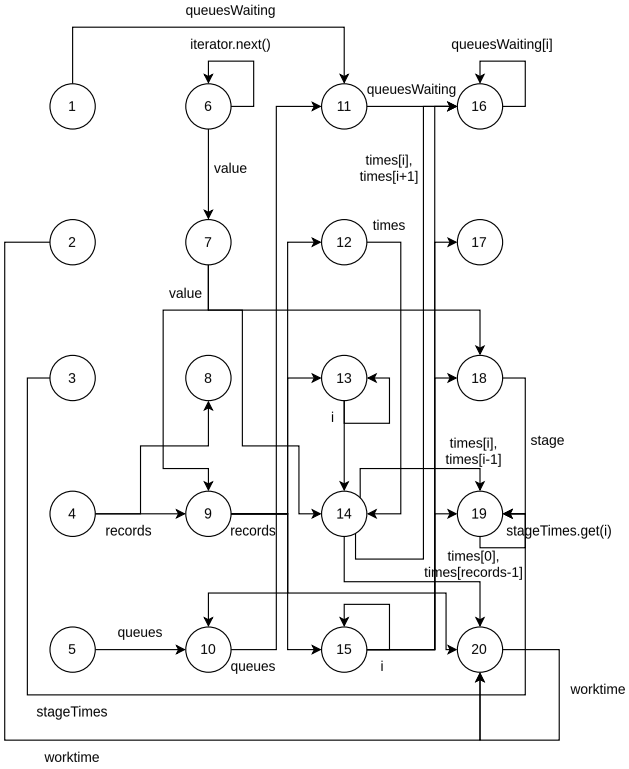

Рисунок 2.1 – Информационный граф

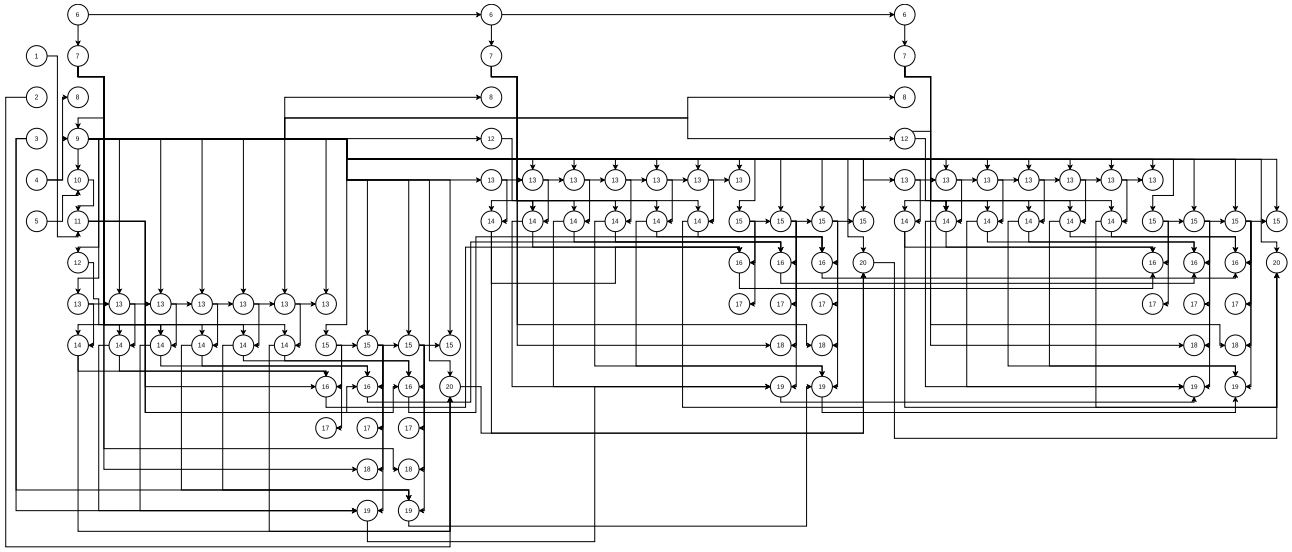

Рисунок 2.2 – Информацинная история

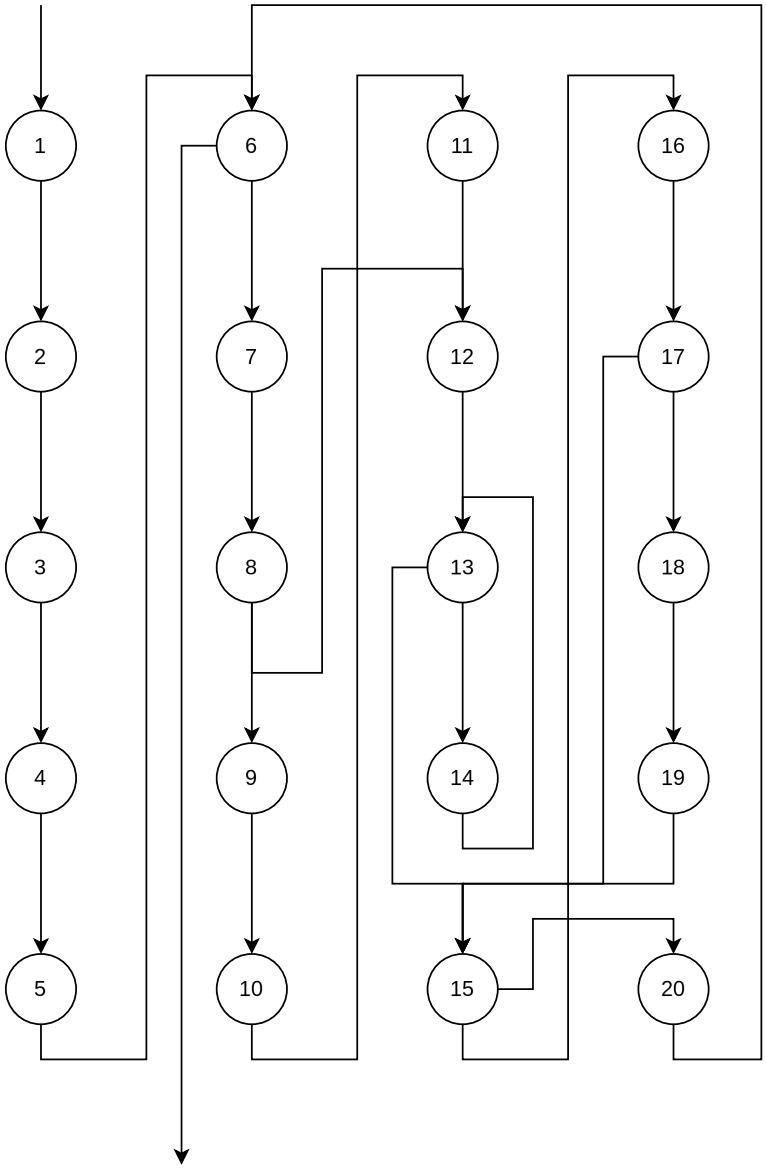

Рисунок 2.3 – Граф управления

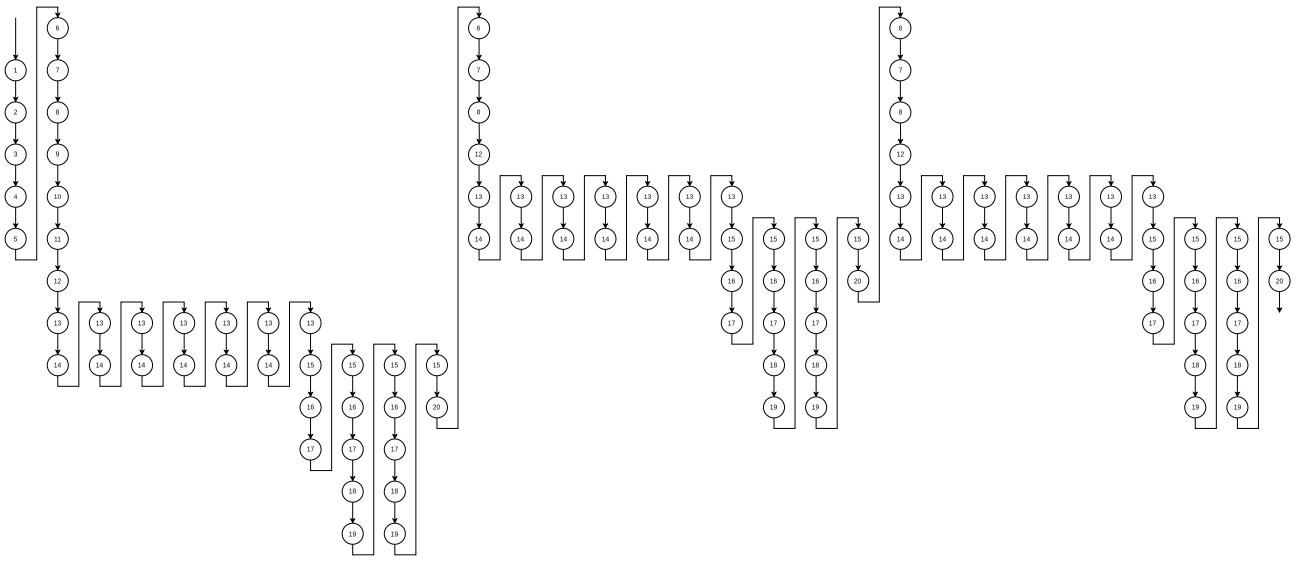

Рисунок 2.4 – Операционная история

Графовые модели показывают зависимости по данным и передаче управления между участками кода. Анализ данных моделей позволяет эффективнее распараллеливать программы путём выявления последовательных участков кода, между которыми либо нет конфликта по доступу к данным, либо такие конфликты могут быть решены с использованием средств взаимоисключения.

ЗАКЛЮЧЕНИЕ

Цель работы достигнута. Решены все поставленные задачи:

— выбраны фрагмент кода для анализа и входные данные для построения информационной и операционной историй;

— построены граф управления, информационный граф, операционную историю и информационную историю;

— сделан вывод о применимости графовых моделей.