



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»

НАПРАВЛЕНИЕ ПОДГОТОВКИ \_\_\_\_\_ «09.03.04 Программная инженерия»

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

Название: \_\_\_\_\_ Распараллеливание алгоритма DBSAN

Дисциплина: \_\_\_\_\_ Анализ алгоритмов

Студент	<u>ИУ7-56Б</u>	_____	<u>Ковель А.Д.</u>
	Группа	Подпись, дата	И. О. Фамилия
Преподаватель		_____	<u>Волкова Л.Л.</u>
Преподаватель		_____	<u>Строганов Ю.В.</u>
		Подпись, дата	И. О. Фамилия

Москва, 2022 г.

# Оглавление

<b>1</b>	<b>Выполнение задания</b>	<b>3</b>
1.1	Средства реализации . . . . .	3
1.2	Программный код . . . . .	3
1.3	Графовые представления . . . . .	5
	<b>Список использованных источников</b>	<b>9</b>

# 1 Выполнение задания

В этом разделе будут представлены код реализации алгоритма и его графовые представления.

## 1.1 Средства реализации

В данной работе для реализации был выбран язык программирования [?].

## 1.2 Программный код

На листинге 1.1 демонстрируется реализация простого алгоритма DBSCAN.

# Листинг 1.1 – Реализация простого алгоритма DBSCAN

```

1 pub fn dbscan(points: &Vec<Vec<bool>>, min_ptx: usize, eps: f64,
2     mut imgbuf: RgbImage) -> u32 {
3     let mut cluster_count = 0; // 1
4     let mut current_point = points.clone(); // 2
5     for i in 0..points.len() { // 3
6         for j in 0..points[i].len() { // 4
7             if current_point[i][j] { // 5
8
9                 let mut v = Vec::<[usize; 2]>::new(); // 6
10                v.push([i, j]); // 7
11                let mut neighbor_count_check = 0; // 8
12                while !v.is_empty() { // 9
13                    let p = v.pop().unwrap(); // 10
14                    if !current_point[p[0]][p[1]] { // 11
15                        continue; // 12
16                    }
17                    current_point[p[0]][p[1]] = false; // 13
18
19                    let mut neighbor_count = 0; // 14
20
21                    regionquery(points, min_ptx, &mut v, &mut
22                        neighbor_count, p, eps); // 15
23
24                    if neighbor_count >= min_ptx { // 16
25                        neighbor_count_check += 1; // 17
26                    }
27                    if neighbor_count_check > 0 { // 18
28                        cluster_count += 1; // 19
29                    }
30                    current_point[i][j] = false; // 20
31                }
32            }
33        }
34
35        cluster_count
36    }

```

## 1.3 Графовые представления

На рисунках 1.1, 1.2, 1.3, 1.4 представлен операционный граф, информационный граф, граф операционной истории, граф информационной истории.

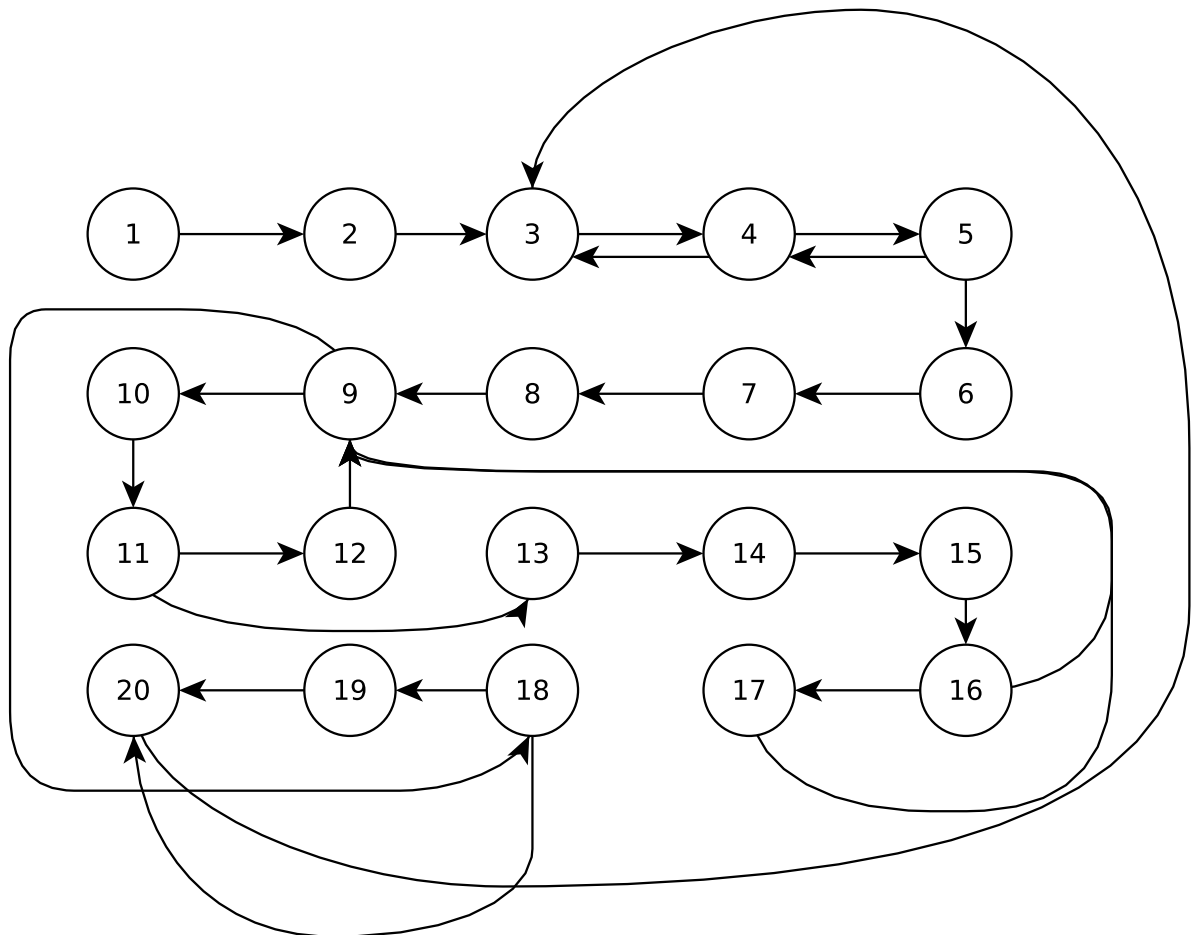


Рисунок 1.1 – Операционный граф

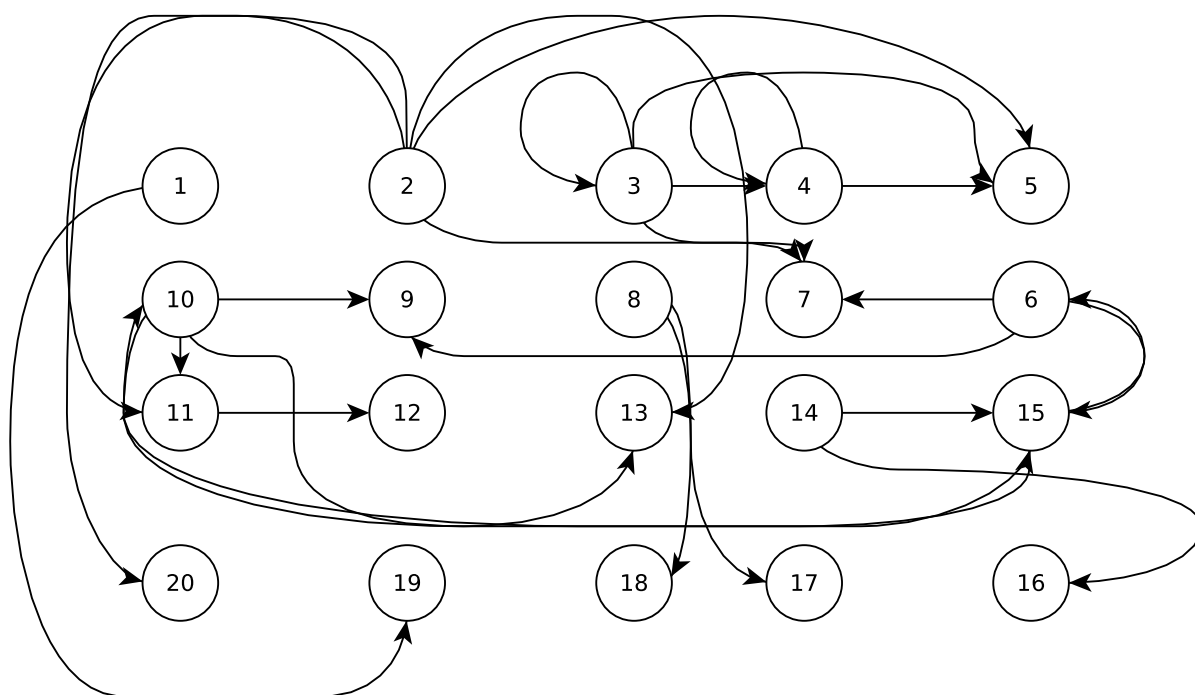


Рисунок 1.2 – Информационный граф

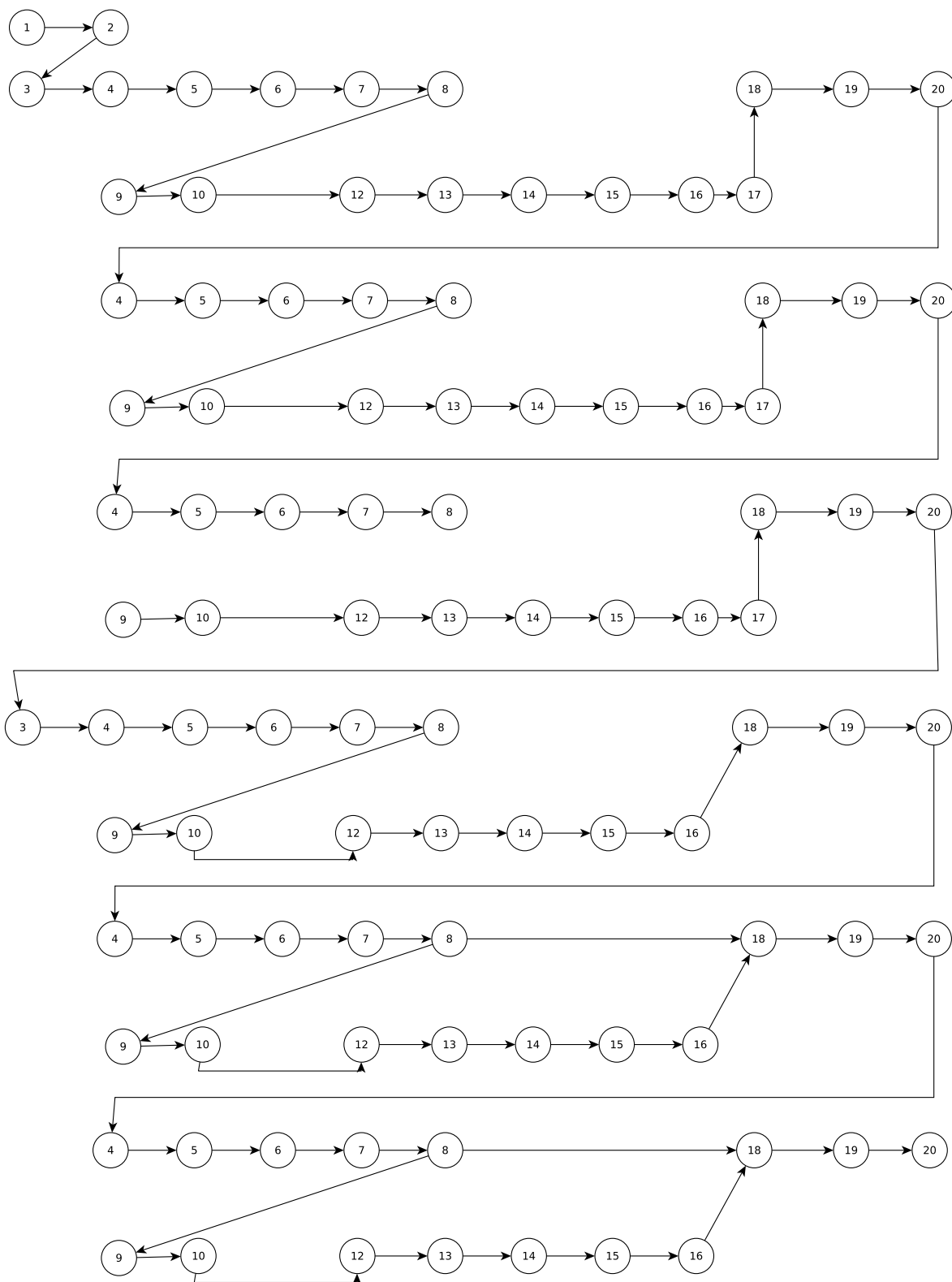


Рисунок 1.3 – Граф операционной истории

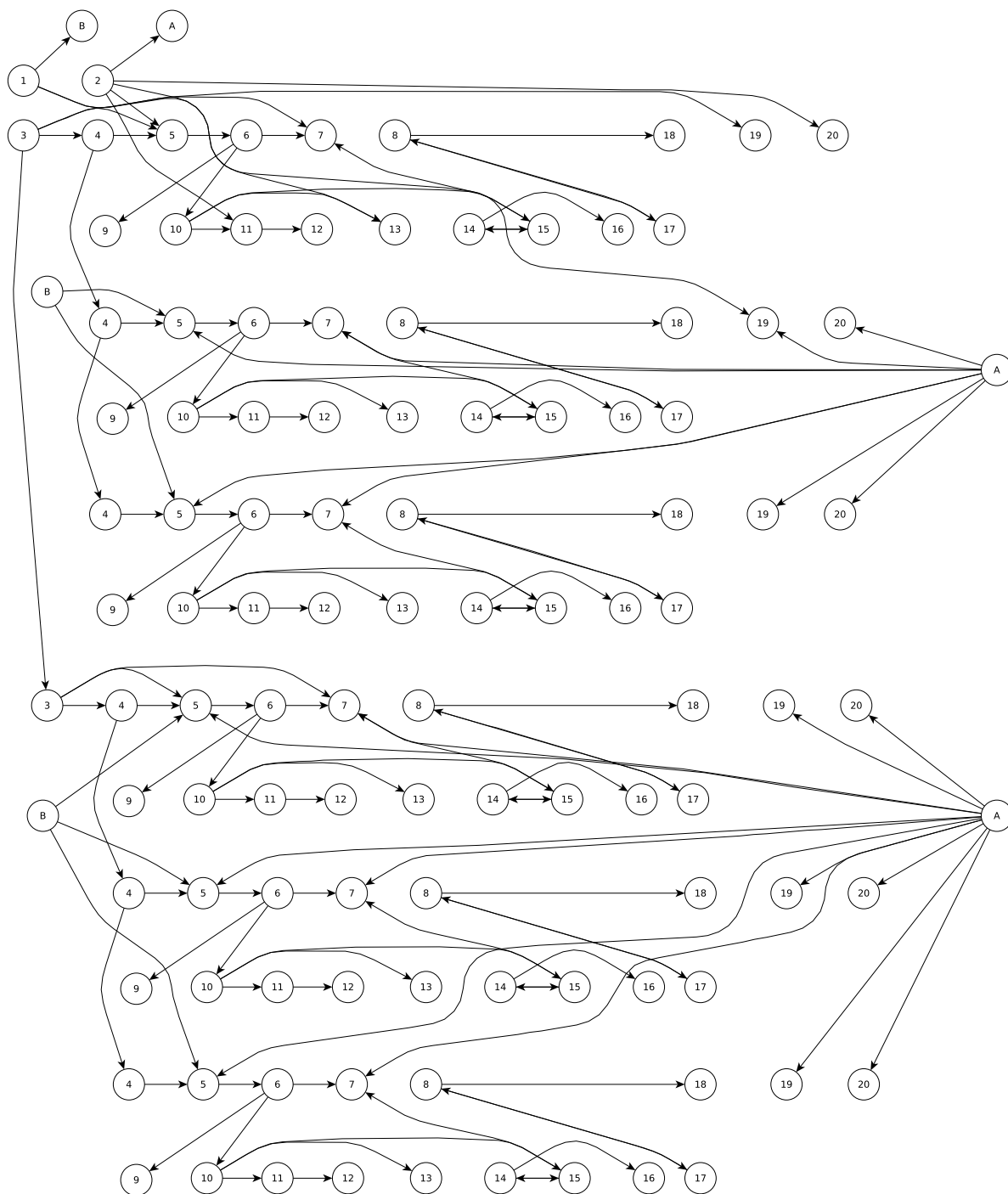


Рисунок 1.4 – Граф информационной истории



## Список использованных источников