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»

на тему: «Параллельные вычисления на основе нативных потоков»

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

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

2023 г.

СОДЕРЖАНИЕ

1 Описание задачи

В данной части работы, будет описано полученное задание, а также смысл графового представления программы.

1.1 Задание

Описать четырьмя графовыми моделями (граф управления, информационный граф, опрерационная 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 17: в качестве реализуемого алгоритма — сортировка слиянием.

1.2 Графовые модели программы

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

Выделяют 2 типа дуг [1]:

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.

2 Выполнения задания

2.1 Выбор языка программирования

Для выполнения домашнего задания был выбран язык `C++`.

2.2 Код программы

Код программы приведен в листинге А.1 в приложении.

2.3 Графовые модели программы

2.3.1 Граф управления

На рисунке 2.1 представлен граф управления.

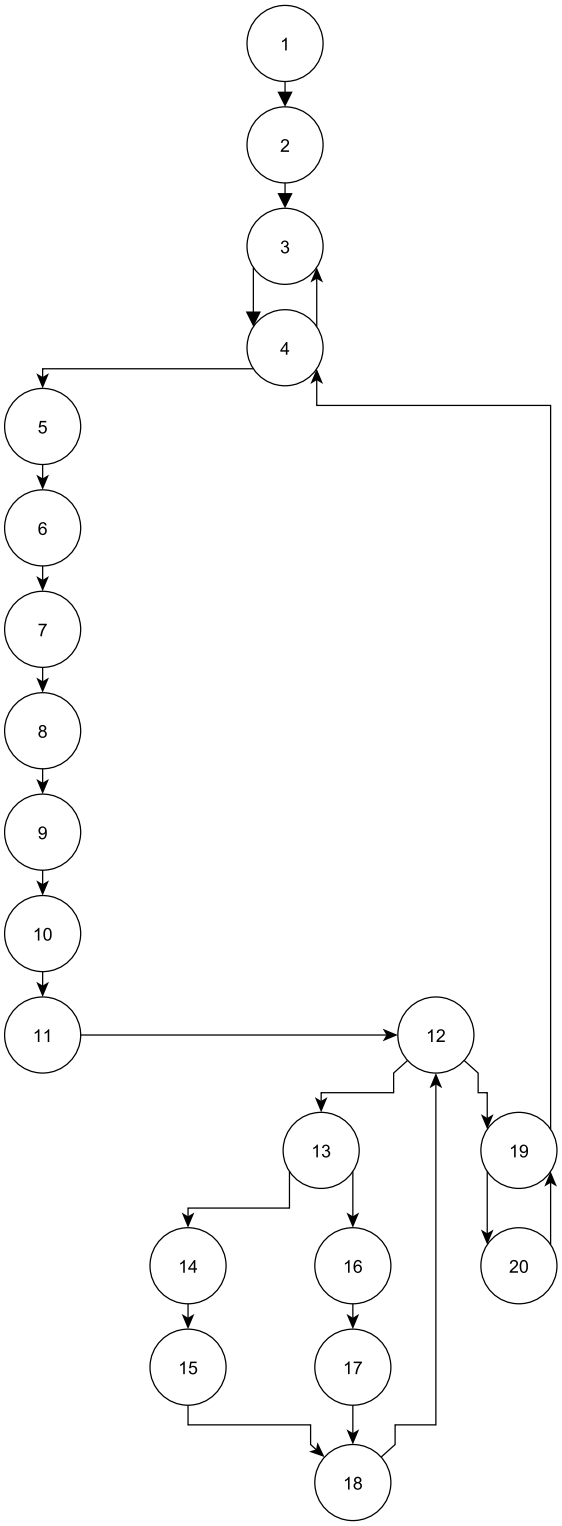

Рисунок 2.1 – Граф управления

2.3.2 Информационный граф

На рисунке 2.2 представлен информационный граф.

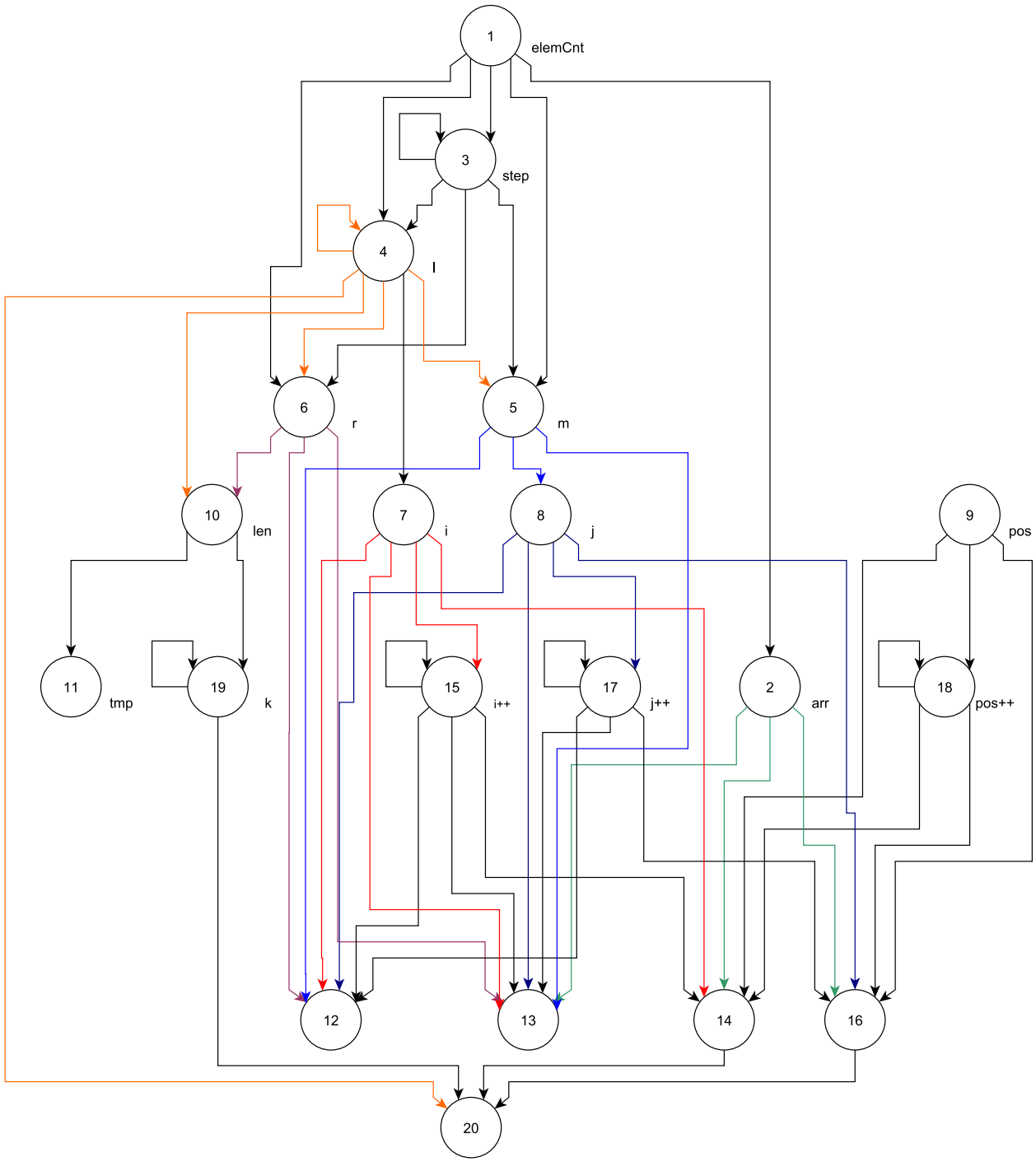

Рисунок 2.2 – Информационный граф

2.3.3 Информационная история

Рассмотрим следующий массив: `a = [7, 4, 2, 1]`.

На рисунках 2.5–2.6 представлена операционная история программы для этого массива, а также и информационная история 2.3–2.4.

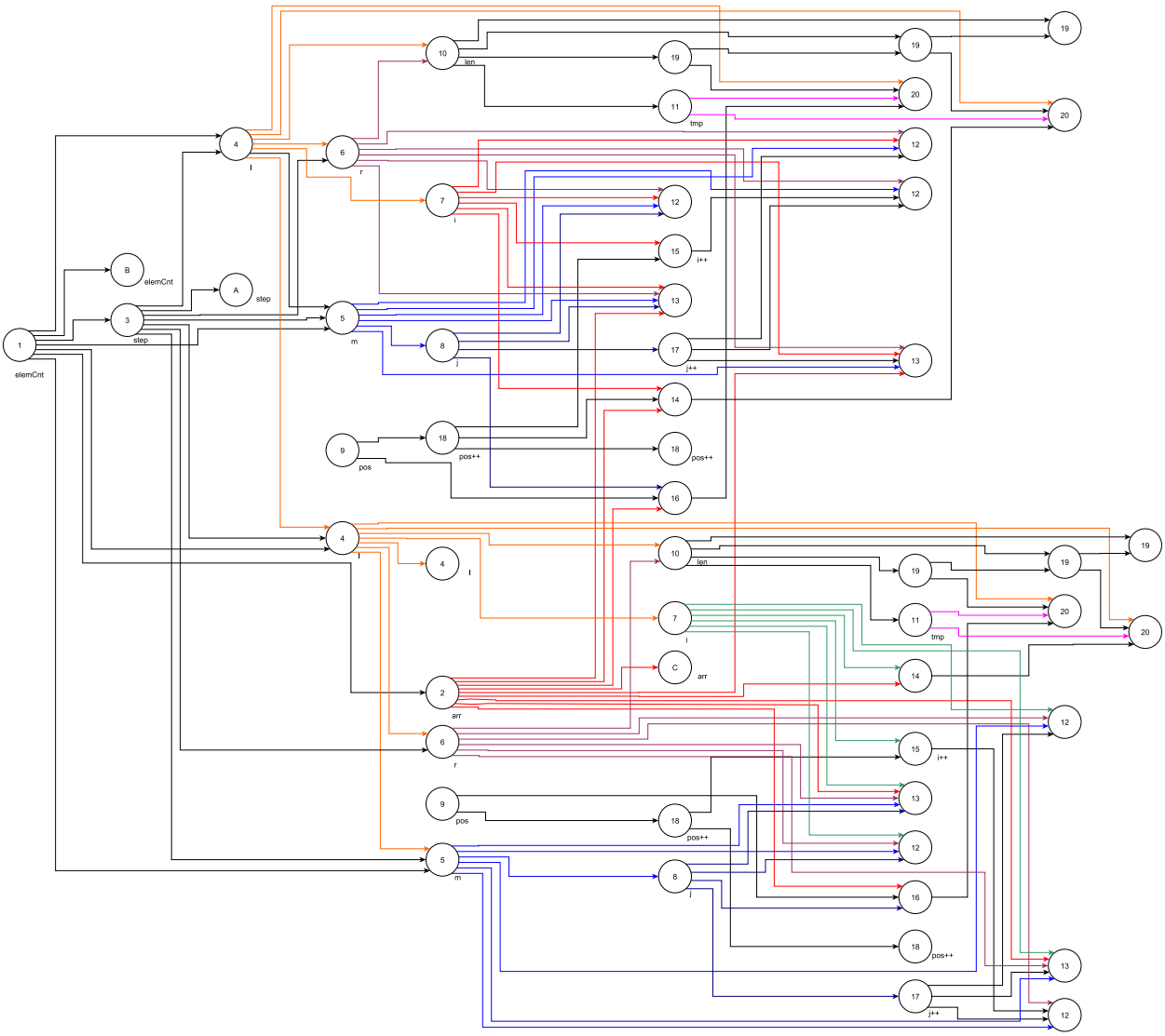

Рисунок 2.3 – Информационная история (начало)

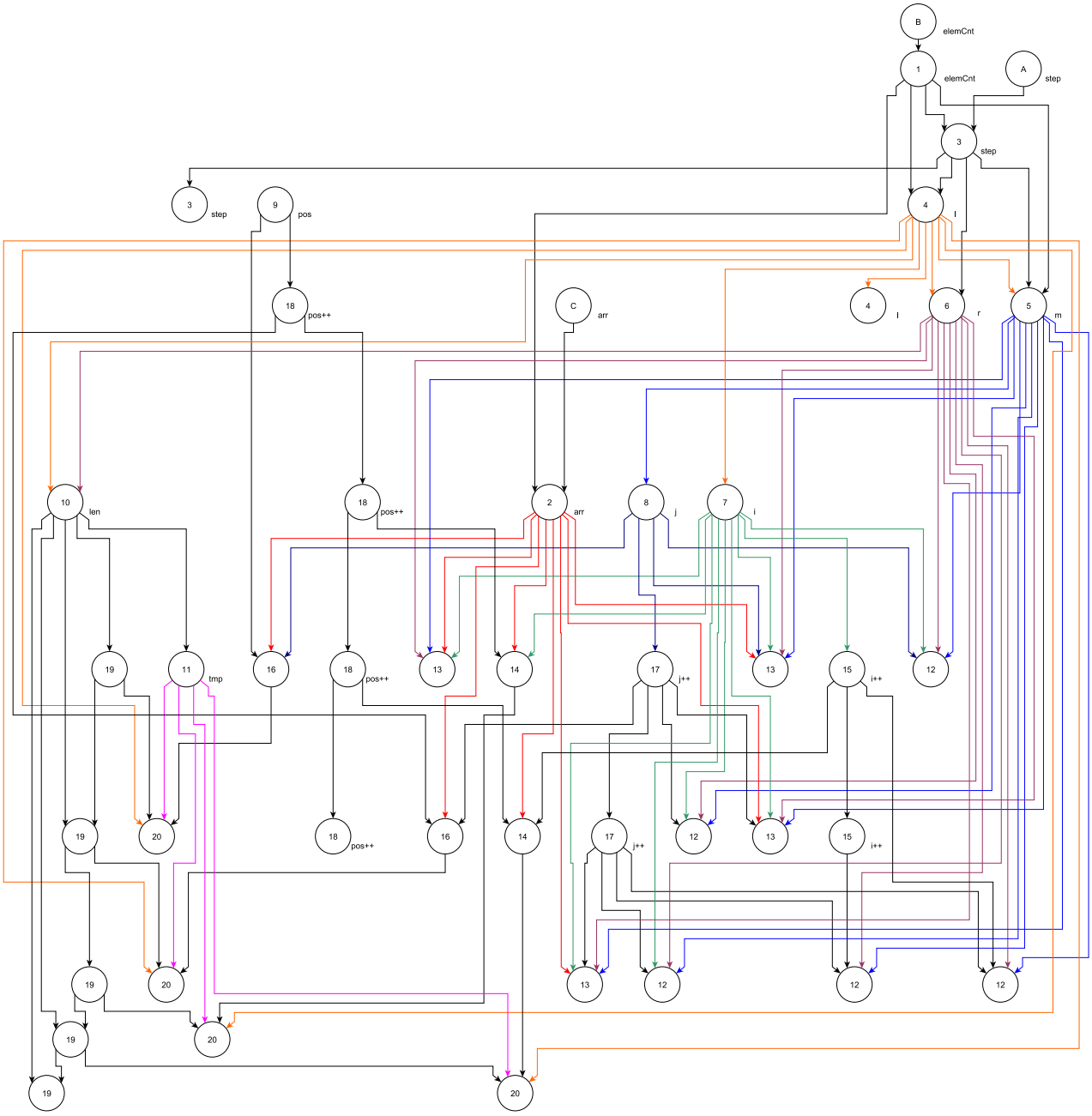

Рисунок 2.4 – Информационная история (продолжение)

2.3.4 Операционная история

На рисунках 2.5–2.6 представлена операционная история программы для данного массива.

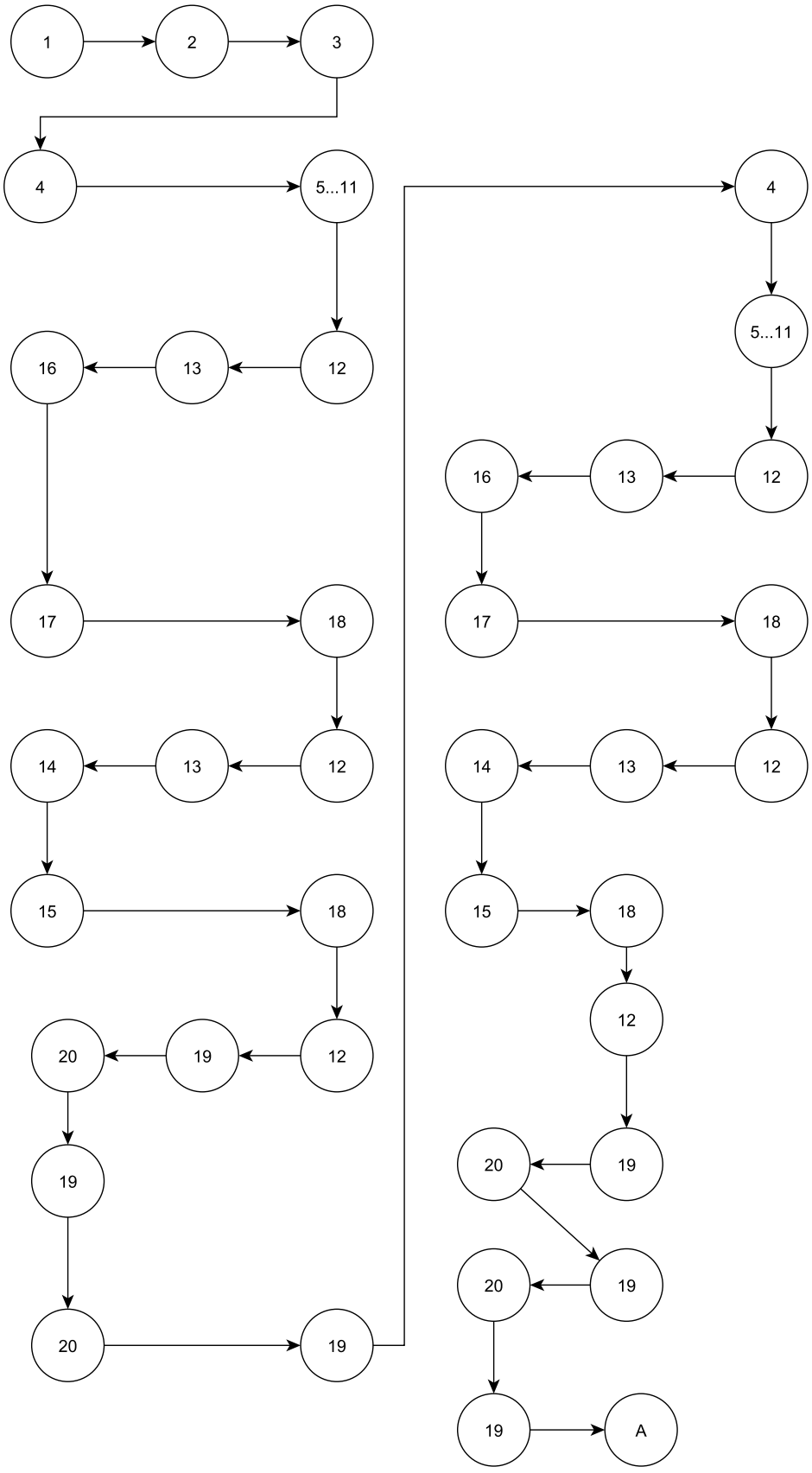

Рисунок 2.5 – Операционная история (начало)

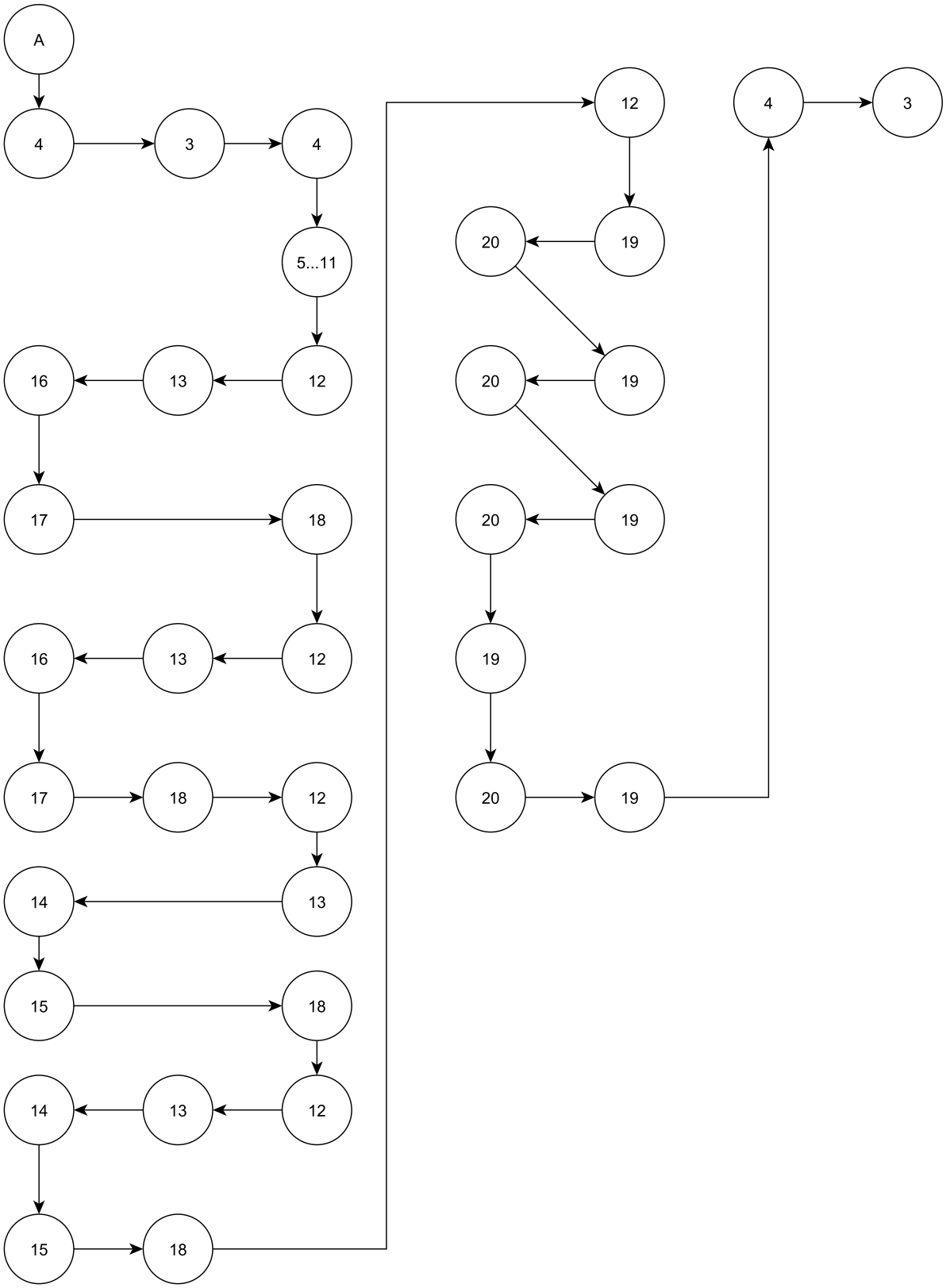

Рисунок 2.6 – Операционная история (продолжение)

Вывод

Сортировка разделяет массив на непересекающиеся подмассивы, а затем объединяет их, возможно организовать объединение непересекающихся подмассивов в отдельных потоках.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Математические основы параллельных вычислений [Электронный ресурс]. — — Режим доступа: https://intuit.ru/studies/courses/4447/983/lecture/14919 (дата обращения: 06.12.2023).

ПРИЛОЖЕНИЕ А

Листинг А.1 – Реализация алгоритма сортировки слиянием

```
}


int main() {

    srand(time(0));

    int elemCnt = 4;
                                                                    //(1)
    std::vector<int> arr(elemCnt);
                                                          //(2)

    for (int i = 0; i < elemCnt; ++i)
        arr[i] = rand() % 1000 - 500;

    std::cout << "Prev: ";
    printVector(arr);
    std::cout << std::endl;

    for (int step = 1; step < elemCnt; step = step * 2){
                 //(3)

        for (int l = 0; l < elemCnt - 1; l += 2 * step){   //(4)

            int m = std::min(l + step - 1, elemCnt - 1),
                          //(5)
            r = std::min(l + 2 * step - 1, elemCnt - 1);
               //(6)

            int i = l,
               //(7)
            j = m + 1,                                          //(8)
            pos = 0,
               //(9)
            len = r - l + 1;                                //(10)
```

```
            std::vector<int> tmp(len);
               //(11)

            while (i <= m || j <= r) {                          //(12)

                if (i <= m && (j > r || arr[i] < arr[j])) {//(13)
                    tmp[pos] = arr[i];
                       //(14)
                    i++;
                       //(15)
                }
                else {
                    tmp[pos] = arr[j];
                       //(16)
                    j++;
                       //(17)
                }
                pos++;
                   //(18)
            }

            for (int k = 0; k < len; ++k)
               //(19)
                arr[k + l] = tmp[k];
                   //(20)
        }
    }

    std::cout << "After: ";
    printVector(arr);
}
```