ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

“ВЫСШАЯ ШКОЛА ЭКОНОМИКИ”

Факультет компьютерных наук

Образовательная программа бакалавриата “Программная инженерия”

Домашнее задание 3 по дисциплине “Архитектура вычислительных систем”

Пояснительная записка

Выполнил

Студент группы БПИ191

Дадугин Егор Артемович

Москва 2020

Оглавление

1. Введение ... 3

1.1 Текст задания ... 3

1.2 Алгоритм программы ... 3

2. Входные и выходные данные ... 4

2.1 Входные данные ... 4

2.2 Выходные данные ... 4

3. Тестирование программы ... 6

3.1 Ввод 2 потоков на массиве из 9 элментов (0,0,0,0,1,2,3,4,9) для наглядности работы ... 6

3.2 Ввод 2 потоков на случайно сгенерированном массиве из 1000 элментов ... 6

3.3 Ввод 3 потоков на случайно сгенерированном массиве из 1000 элментов ... 7

4. Список использованных источников ... 8

ПРИЛОЖЕНИЕ 1 ... 9

ТЕКСТ ПРОГРАММЫ ... 9

1. Введение

1.1 Текст задания

Определить индексы i, j, для которых существует наиболее длинная последовательность A[i] < A[i+1] < A[i+2] < A[i+3] < ... < A[j]. Входные данные: массив чисел A, произвольной длины большей 1000. Количество потоков является входным параметром.

1.2 Алгоритм программы

Работа выполнена с помощью библиотеки thread.

Выполнение программы осуществлено с помощью итеративного подхода(массив делится на части в зависимости от количества потоков, для каждого потока предусмотрен цикл с определенными границами). Программа состоит из функции func, которая ищет возрастающую последовательность наибольшей длинны в заданном интервале(для каждого потока). Предусмотрена ситуация, когда последовательность заканчивается в одном потоке и продолжается в следующем.

Функция принимает по ссылке параметры: вектор, начальный индекс последовательности, конечный индекс последовательности, максимальную длину, предыдущую длину последовательности. Также передаются параметры: начало и конец интервала для потока.

Функция сравнивает элементы вектора и изменяет параметры максимальной длины и индексов начала и конца последовательности. Функция хранит длину последней последовательности из предыдущего потока, и в начале нового сравнивает крайние элементы в интервалах потоков, что позволяет не пропустить последовательность, разбитую двумя потоками.

2. Входные и выходные данные

2.1 Входные данные

На вход программа получает целое число больше - количество потоков.

2.2 Выходные данные

На выходе программа выводит начальный и конечный индексы наибольшей возрастающей последовательонсти.

3. Тестирование программы

3.1 Ввод 2 потоков на массиве из 9 элментов (0,0,0,0,1,2,3,4,9) для наглядности работы

```
Enter the number of threads: 2
0
0
0
0
1
2
3
4
9
Start index: 3
End index: 8
```

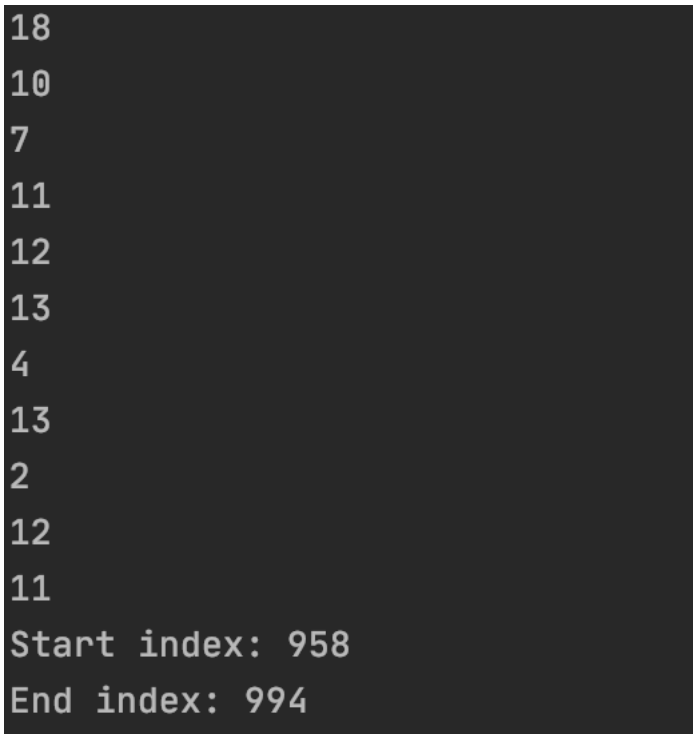
Программа выводит начальный и конечный индексы наибольшей возрастающей последовательонсти.

3.2 Ввод 2 потоков на случайно сгенерированном массиве из 1000 элментов

```
18
10
7
11
12
13
4
13
2
12
11
Start index: 942
End index: 980
```

Программа выводит начальный и конечный индексы наибольшей возрастающей последовательонсти.

3.3 Ввод 3 потоков на случайно сгенерированном массиве из 1000 элментов



Программа выводит начальный и конечный индексы наибольшей возрастающей последовательонсти.

4. Список использованных источников

1) RIP Tutorial [Электронный ресурс] URL: https://riptutorial.com/ru/cplusplus/example/2330/создание-std----thread (Режим доступа: свободный)
2) SecurityLab.ru [Электронный ресурс] URL: https://www.securitylab.ru/news/514134.php (Режим доступа: свободный)

ПРИЛОЖЕНИЕ 1

ТЕКСТ ПРОГРАММЫ

```
//Дадугин Егор Артемович
//БПИ191
//Вариант 12
//Определить индексы i, j, для которых существует наиболее длинная
последовательность
// A[i] < A[i+1] < A[i+2] < A[i+3] < ... < A[j].
// Входные данные: массив чисел A, произвольной длины большей 1000.
Количество потоков является входным параметром.

#include <iostream>
#include <vector>
#include <thread>

void
func(std::vector<int> &array, int currentInd, int lastInd, int &finalI, int
&finalJ, int &maxLength, int &prevLength,
     int &currentI) {
    int currentLength = 1;
    if (currentInd - 1 >= 0)
        if (array[currentInd - 1] < array[currentInd]) {
            currentLength = prevLength + 1;
        } else
            currentI = currentInd;
    for (int i = currentInd; i < lastInd - 1; ++i) {
        if (array[i] < array[i + 1]) {
            currentLength++;
        } else if (maxLength <= currentLength) {
            maxLength = currentLength;
            finalI = currentI;
            finalJ = i;
            currentI = i + 1;
            currentLength = 1;
        }
        if (i == array.size() - 2) {
            if (maxLength <= currentLength) {
                maxLength = currentLength;
                finalI = currentI;
                finalJ = i+1;
                currentI = i + 1;
                currentLength = 1;
            }
        }
    }
    prevLength = currentLength;
}

int main() {
    int threadsNumber, maxThreadsNumber = 1000;
    int finalI = -1, finalJ = -1, maxLength = 0, currentInd = 0, lastInd,
prevLength = 0, currentI = 0;
    std::cout << "Enter the number of threads: ";
    std::cin >> threadsNumber;
    if (threadsNumber > maxThreadsNumber) {
        threadsNumber = maxThreadsNumber;
    }
    std::vector<int> array(1000);
```

```
    for (int i = 0; i < array.size(); ++i) {
        array[i] = rand() % 20 + 1;
        std::cout << array[i] << std::endl;
    }
    lastInd = array.size() / threadsNumber;
    for (int i = 0; i < threadsNumber; ++i) {
        if (i == threadsNumber - 1)
            lastInd = array.size();
        (new std::thread{func, std::ref(array), currentInd, lastInd,
std::ref(finalI), std::ref(finalJ),
                         std::ref(maxLength), std::ref(prevLength),
std::ref(currentI)})->join();
        currentInd += array.size() / threadsNumber;
        lastInd += array.size() / threadsNumber;
    }

    std::cout << "Start index: " << finalI << std::endl;
    std::cout << "End index: " << finalJ;
    return 0;
}
```