ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

“ВЫСШАЯ ШКОЛА ЭКОНОМИКИ”

Факультет компьютерных наук

Образовательная программа бакалавриата “Программная инженерия”

Домашнее задание 4 по дисциплине “Архитектура вычислительных систем”

Пояснительная записка

Выполнил

Студент группы БПИ191

Дадугин Егор Артемович

Москва 2020

Оглавление

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 OpenMP.

Выполнение программы осуществлено с помощью итеративного подхода. В ходе выполнения происходит распараллеливание на потоки. Наибольшая возрастающая последовательность находится посредством заполнения вспомогательного массива, в котором для каждого элемента хранится длина возрастающей последовательности, заканчивающаяся на данном элементе.

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 последовательности.

3. Тестирование программы

3.1 Ввод 3 потоков на массиве из 10 элементов (0,0,0,0,1,2,3,4,9) для наглядности работы

```
Enter the number of threads: 3
8
10
14
19
11
13
5
19
4
10
Start index: 5
End index: 8
```

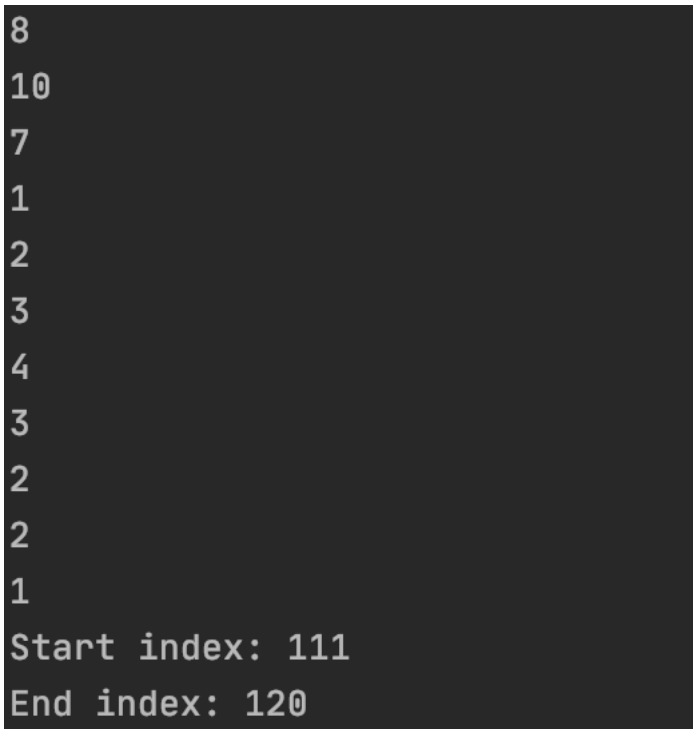
Программа выводит начальный и конечный индексы наибольшей возрастающей последовательности.

3.2 Ввод 3 потоков на случайно сгенерированном массиве из 1000 элементов

```
18
10
7
11
12
13
4
13
2
12
11
Start index: 406
End index: 425
```

Программа выводит начальный и конечный индексы наибольшей возрастающей последовательности.

3.3 Ввод 10 потоков на случайно сгенерированном массиве из 1000 элементов



Программа выводит начальный и конечный индексы наибольшей возрастающей последовательности.

4. Список использованных источников

1) Habr [Электронный ресурс] URL: https://habr.com/ru/post/71296/ (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;
}
```