МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота N6

З дисципліни

«Алгоритмізація та програмування»

Виконав:

Студент групи КН-108

Воробель Адріан

Викладач:

Грабовська Н.Р.

Львів – 2019р.

Мета:

● Ознайомлення з моделлю потоків Java.

● Організація паралельного виконання декількох частин програми.

● Вимірювання часу паралельних та послідовних обчислень.

● Демонстрація ефективності паралельної обробки.

1. Вимоги

1. Використовуючи програми рішень попередніх задач, продемонструвати можливість паралельної обробки елементів контейнера: створити не менше трьох додаткових потоків, на яких викликати відповідні методи обробки контейнера.

2. Забезпечити можливість встановлення користувачем максимального часу виконання (таймаута) при закінченні якого обробка повинна припинятися незалежно від того знайдений кінцевий результат чи ні.

3. Для паралельної обробки використовувати алгоритми, що не змінюють початкову колекцію.

4. Кількість елементів контейнера повинна бути досить велика, складність алгоритмів обробки колекції повинна бути зіставна, а час виконання приблизно однаковий, наприклад:

○ пошук мінімуму або максимуму;

○ обчислення середнього значення або суми;

○ підрахунок елементів, що задовольняють деякій умові;

○ відбір за заданим критерієм;

○ власний варіант, що відповідає обраній прикладної області.

5. Забезпечити вимірювання часу паралельної обробки елементів контейнера за допомогою розроблених раніше методів.

6. Додати до алгоритмів штучну затримку виконання для кожної ітерації циклів поелементної обробки контейнерів, щоб загальний час обробки був декілька секунд.

7. Реалізувати послідовну обробку контейнера за допомогою методів, що використовувались для паралельної обробки та забезпечити вимірювання часу їх роботи.

8. Порівняти час паралельної і послідовної обробки та зробити висновки про ефективність розпаралелювання:

○ результати вимірювання часу звести в таблицю;

○ обчислити та продемонструвати у скільки разів паралельне виконання швидше послідовного.

1.1 Розробник

Воробель Адріан, КН-108, номер варіанту індивідуального завдання – 5.

1.2 Задача

Опис програми:

Програма розділена на потоки, де кожен клас(потік) має свій конкретний алгоритм. Може працювати як паралельно, так і послідовно, а також встановлювати timeout.

2.1 Засоби ООП

Багатопоточність, декілька класів.

2.2 Ієрархія та структура класів

Клас Conteiner і клас Iterator – взяті з Лабораторної №2

Три класи(потоки) – ThreadOne(шукає найдовше і найкоротше слово), ThreadTwo(шукає кількість слів ,що містять букву 'g') та ThreadThree(шукає слова які мають таку ж кількість символів як і слово в контейнері від індексом 5).

```java
package com.company;

import java.util.Arrays;
/* найдовше і найменше слово */
public class ThreadOne extends Thread {

    protected Conteiner container = new Conteiner();
    int [] sizes = new int[container.size()];

    @Override
    public void run(){

        Thread.currentThread().setName("ThreadOne");
        long startTime = System.currentTimeMillis();
        int max=0, min=0, sum = 0;

        int [] sizes = new int[container.size()];

        for(int i=0; i< container.size(); i++ ){
            try {
                Thread.sleep( millis: 300);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            sizes[i] = container.get(i).length();
            sum += container.get(i).length();
        }

        Arrays.sort(sizes);
        min = sizes[0];
        max = sizes[sizes.length-1];
        for(int i=0; i< container.size(); i++ ){
            try {
                Thread.sleep( millis: 300);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
           if(container.get(i).length() == min)
               System.out.println("Найкоротше слово - "+container.get(i));
           else if(container.get(i).length() == max)
               System.out.println("Найдовше слово - "+container.get(i));
        }
        long timeSpent = System.currentTimeMillis() - startTime;
        System.out.println(Thread.currentThread().getName()+" виконувався "+timeSpent+" мілісекунд");
    }
}
```

```java
package com.company;
/*кількість слів ,що містять букву 'g'*/
public class ThreadTwo extends Thread {
    Conteiner container = new Conteiner();

    @Override
    public void run(){
        Thread.currentThread().setName("ThreadTwo");

        long startTime = System.currentTimeMillis();
        char bykwa = 'g';



        int counter=0;
        for (int i = 0; i < container.size(); i++)
            for (int j = 0; j < container.get(i).length(); j++) {
                try {
                    Thread.sleep( millis: 100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                if(container.get(i).charAt(j) == bykwa) {
                    counter++;
                    break;
                }
            }

        if(counter == 0)
            System.out.println("Нема слів з буквою '"+bykwa+"'");
        else
            System.out.println("З буквою '"+bykwa+"' знайдено ["+counter+"] слів/слово");

        long timeSpent = System.currentTimeMillis() - startTime;
        System.out.println(Thread.currentThread().getName()+" виконувався "+timeSpent+" мілісекунд");
    }
}
```

```java
package com.company;

import java.util.Arrays;
/*слова які мають таку ж кількість символів як і слово в контейнері від індексом 5*/
public class ThreadThree extends Thread {
    Conteiner container = new Conteiner();

    @Override
    public void run(){
        Thread.currentThread().setName("ThreadThree");

        String [] mas = new String[0];
        String slovo = container.get(5);
        long startTime = System.currentTimeMillis();
        for (int i = 0; i < container.size(); i++) {
            try {
                Thread.sleep( millis: 200);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            if(i == 5)
                continue;
            else if(container.get(i).length() == slovo.length()){
                mas = Arrays.copyOf(mas,  newLength: mas.length+1);
                mas[mas.length-1] = container.get(i);
            }
        }

        if(mas.length == 0)
            System.out.println("Слово "+container.get(5)+" - єдине яке має "+container.get(5).length()+" символів");
        else {
            System.out.println("Слова з такою ж кількістю символів як і в " + container.get(5));
            for (String s : mas)
                System.out.println(s);
        }

        long timeSpent = System.currentTimeMillis() - startTime;
        System.out.println(Thread.currentThread().getName()+" виконувався "+timeSpent+" мілісекунд");
    }
}
```

3. Варіанти використання

Програма може використовуватись для виконання одночасно декількох завдань.

ВИСНОВКИ

У ході роботи я навчився працювати з потоками, зупиняти, запускати, притримувати і тд. Зрозумів як запускати послідовно і паралельно, та проаналізував як оптимальніше.