Programación Concurrente

Trabajo Práctico

Se desea implementar en Java un programa concurrente que verifique si una permutación de $n \times n$ números representa un cuadrado latino. Un cuadrado latino es una matriz de $n \times n$ celdas en la cual cada celda contiene un número entre 1 y n, donde en ninguna fila ni en ninguna columna se repiten números. El archivo de entrada contiene una entrada por linea, salvo en la primer línea que contiene un entero k con la cantidad total de entradas en el archivo. Cada entrada (linea) en el archivo comienza con un número n que indica la dimensión del cuadrado a validar, luego contiene una secuencia de $n \times n$ números separada por espacios.

Por ejemplo el archivo:

```
1
3 1 2 3 3 1 2 2 3 1
```

representa un archivo de entrada con un único cuadrado a validar ($k = 1$), con un cuadrado de 3×3 ($n = 3$) y representa la siguiente matriz:

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}$$

Esta matriz es un cuadrado latino y, dado que es el primer cuadrado en el archivo, el programa debe reportar que la entrada de índice 1 cumple la propiedad.

El programa debe leer de un archivo una lista de entradas a validar. El programa debe generar como output los índices de las entradas que corresponden a cuadrados latinos en orden creciente.

Se debe respetar la siguiente estructura:

1. Una clase `Main` con el punto de entrada del programa. El programa debe terminar únicamente cuando todas las entradas del archivo fueron procesadas, y finalmente informar el tiempo transcurrido desde el inicio de la ejecución.

2. Una clase `Buffer` (implementada como un monitor utilizando métodos synchronized) que actúa como una cola FIFO concurrente. Es decir, bloquea a un lector intentando sacar un elemento cuando está vacía.

3. Una clase `Countdown` (implementada como un monitor utilizando métodos synchronized) que se inicializa con un valor entero y expone dos métodos:

   - `dec` que decrementa el contador; y
   - `zero` que bloquea a los procesos que lo invocan hasta que el contador tenga un valor de cero (o menos).

4. Una clase `SortedList` (implementada con un monitor utilizando métodos synchronized) que permite insertar números de manera que siempre quede ordenada.

5. Una clase `LatinWorker` que extiende de `Thread` y realiza la verificación de si un cuadrado es latino o no. Un `LatinWorker` debe tomar las tareas de verificación de cuadrados de un `Buffer` conocido al momento de su creación. Por cada tarea resuelta debe decrementar un `Countdown` que registra las tareas restantes. Por cada cuadrado que cumple la propiedad de ser latino, el worker debe insertar el índice del cuadrado en la `SortedList`.

6. Una clase `ThreadPool`, que se encarga de instanciar e iniciar la cantidad de `LatinWorkers` pedida por un usuario. La clase `ThreadPool` debe proveer un método para descartar a los threads cuando ya no sean necesarios (e.g., mediante la implementación de una tarea de tipo `PoisonPill`).

Al iniciar el programa se debe delegar la iniciación de los threads necesarios en la clase `ThreadPool` y luego introducir de a uno los cuadrados a verificar en el `Buffer`. Cada `LatinWorker` en funcionamiento debe tomar cuadrados de a uno del `Buffer` y verificar si son o no perfectos. Cabe destacar que es

inadmisible utilizar una cantidad de threads menor a la solicitada por el usuario (salvo cuando se tienen menos cuadrados que threads).

Junto con este enunciado, se incluye un archivo `inputs-ejemplo` con algunos candidatos a cuadrados latinos para usar como referencia. Para ese archivo, el output deberían ser los números `1, 4, 7, 9`.

Además del código, se debe entregar un informe corto que incluya los tiempos de ejecución del programa con los datos de test y distintos valores para la cantidad de threads (al menos deben probarse cantidades de threads entre 1 y 8). El informe debe incluir las características del equipo donde se realizó la prueba y una conclusión relativa a los tiempos obtenidos.

Para entregar, envíen a las casillas de email de los docentes (que están en el grupo de Discord) un email con el proyecto Java del TP en un archivo `.zip`. Procuren no usar librerías externas, el TP debe poder compilarse con los archivos incluídos en el mail. Se debe poner en copia a ambos docentes de la materia y a todos los integrantes del grupo. En el cuerpo del mail también deben figurar los datos de los integrantes del grupo. El plazo para realizar la entrega es hasta el 13 de Junio a las 23:59.