UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE CIENCIAS
ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACION

APELLIDOS Y NOMBRES: Orihuela Contreras Jared CÓDIGO: 20220370F

Programación Paralela

PRACTICA CALIFICADA Nº 1

20/09/2024

LOS CODIGOS FUERON ADJUNTADOS EN UN ZIP.
el diagrama del problema 4 se incluye en este.

1. Efectuar un programa serial y paralelo (multi-threading para calcular la n-potencia de una matriz cuadrada A_{NxN}. Determinar las salidas y tiempos correspondientes.

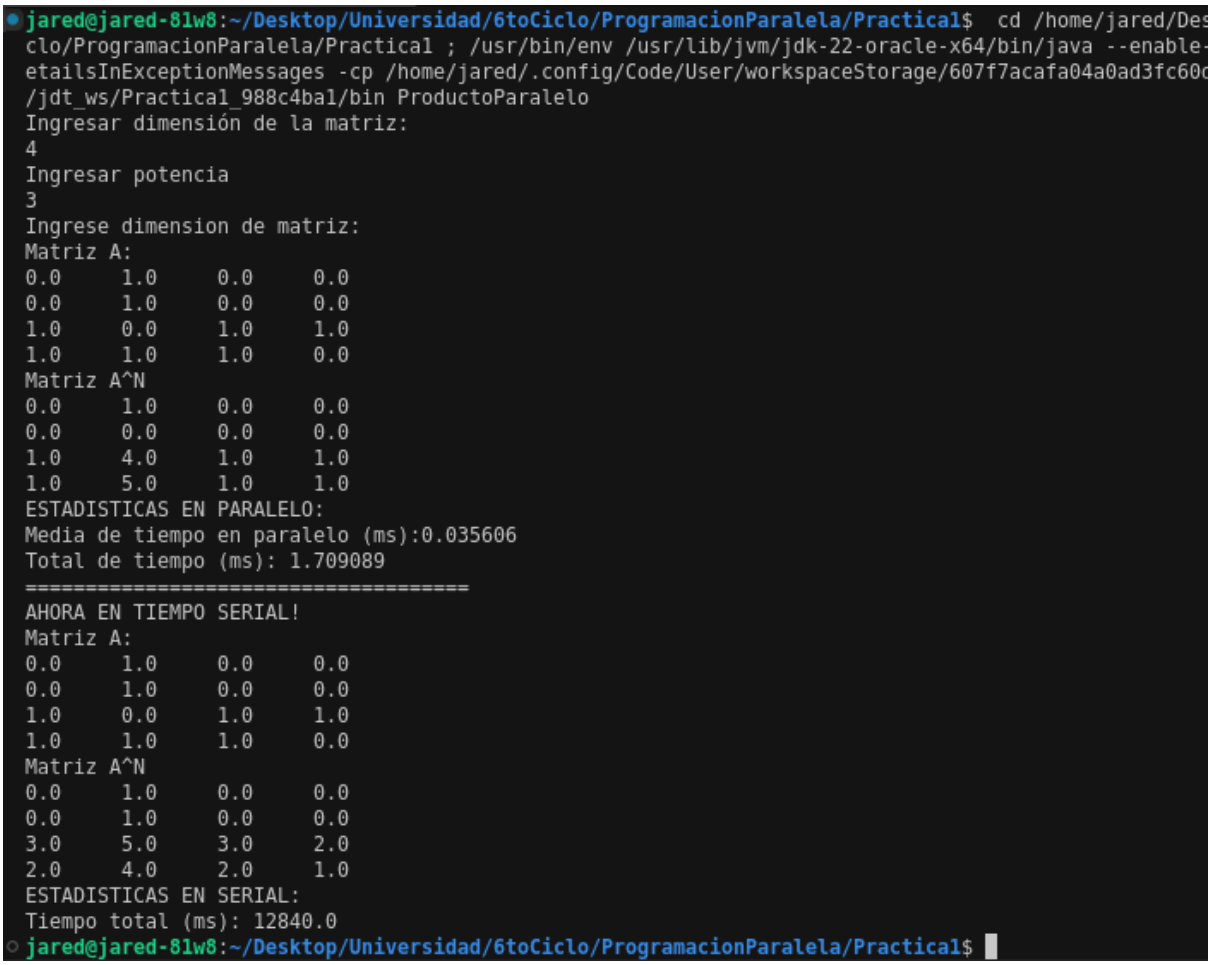
2. Sea P el número de hilos. Dado el problema anterior, efectuar el procesamiento en cada k-ésimo hilo, $1 \leq k \leq P$. para cada hilo determinar el tiempo de ejecución T_K y la salida correspondiente F_K. Realizar las estadísticas correspondientes respecto al procesamiento total serial y paralelo del ejemplo anterior.

3. Efectuar un programa serial y paralelo (multi-threading) para realizar la búsqueda secuencial de elementos (datos) almacenados en un archivo DATOS.TXT.

4. Dado un problema computacional, efectuar el algoritmo, programa y grafo de dependencias asociado.

Problema 1

El programa incluye un CLI para integrar los parametros.



CODIGO:

```
public class MultiplicacionMatrices {
    public static void main(String[] args) throws InterruptedException{

        for(int k = 0; k < potencia-1; k++){
            // Con cada iteración generar nuevos hilos
            // Lista de threads
            Thread[] my_threads = new Thread[n*n];
            int idx = 0;
            for(int i=0;i<n;i++){
                for(int j=0;j<n;j++){
                    final int a = i;
                    final int b = j;
                    my_threads[idx] = new Thread(()->{dotProd(a, b);});
                    //Calcula el tiempo
                    long startTimeWithThreads = System.nanoTime();
                    my_threads[idx].start();
                    long endTimeWithThreads = System.nanoTime();
                    long durationWithThreads = (endTimeWithThreads - startTimeWithThreads);
                    times[k][idx] = durationWithThreads;
                    idx++;
                }
            }
            //Hacemos que cada hilo espere que termine el otro y así evitamos la asincronía
            for(Thread x: my_threads){
                x.join();
            }
            //Ahora igualamos a nuestra matriz Auxiliar con la matriz n ésima
            n_esima = aux;
        }

        imprimir(n_esima);
        System.out.println(x:"ESTADÍSTICAS EN PARALELO:");
        long media = 0;
        for(int i=0; i<potencia; i++){
                for(int k=0;k<n*n;k++){
                    media += times[i][k];
                }
        }
        System.out.println("Media de tiempo en paralelo (ms):" + (double)((media/(potencia*n*n)))/1_000_000);
        System.out.println("Total de tiempo (ms): " + (double)(media)/1_000_000);
        scan.close();

        System.out.println(x:"==================================");
        System.out.println(x:"AHORA EN TIEMPO SERIAL!");
        double[][] serialA = new double[n][n];
        double[][] serialB = new double[n][n];
        double[][] serialC = new double[n][n];
        copiarMatriz(matrixA, serialA);
        copiarMatriz(matrixA, serialB);

        System.out.println(x:"Matriz A:");
        imprimir(serialB);
        System.out.println(x:"Matriz A^b");

        long startTimeWithThreads = System.nanoTime();
        for(int k = 0; k < potencia-1; k++){
            serialMultiplicacion(serialB, serialA, serialC);
            copiarMatriz(serialC, serialB);
        }
        long endTimeWithThreads = System.nanoTime();
        long durationWithThreads = (endTimeWithThreads - startTimeWithThreads);

        imprimir(serialB);
        System.out.println(x:"ESTADÍSTICAS EN SERIAL:");
        System.out.println("Tiempo total (ms): " + (double)(durationWithThreads));
    }

    //Lo hago de esta manera por problemas de referencia
    public static void serialMultiplicacion(double[][] A, double[][] B, double[][] C) {
        for(int i=0; i<n;i++) {
            for(int j=0;j<n;j++) {
                C[i][j] = 0;
                for(int k=0;k<n;k++) {
                    C[i][j] += A[i][k] * B[k][j];
                }
            }
        }
    }

    public static void copiarMatriz(double[][] fuente, double[][] destino) {
        for (int i=0; i<n;i++) {
            for (int j=0; j<n;j++) {
                destino[i][j] = fuente[i][j];
            }
        }
    }
}
```

Problema 2

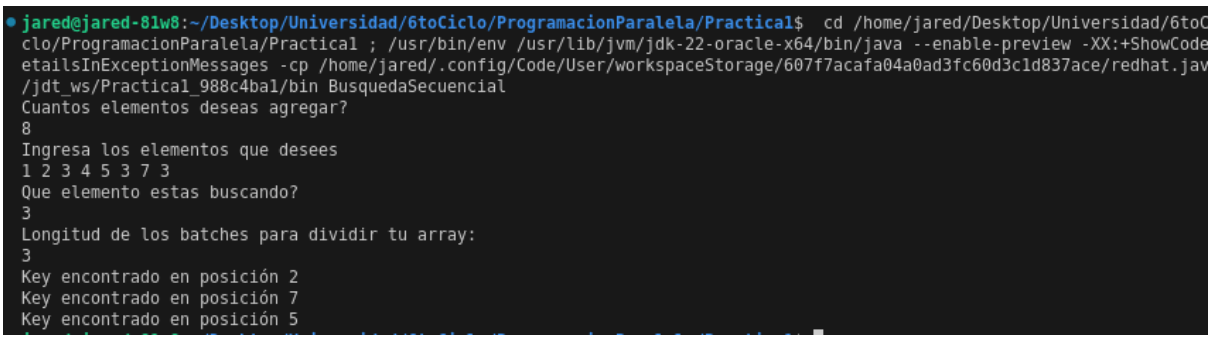
Se incluyeron las estadísticas y se guardo los tiempos de cada hilo, se muestra a continuación la media del tiempo por hilo.

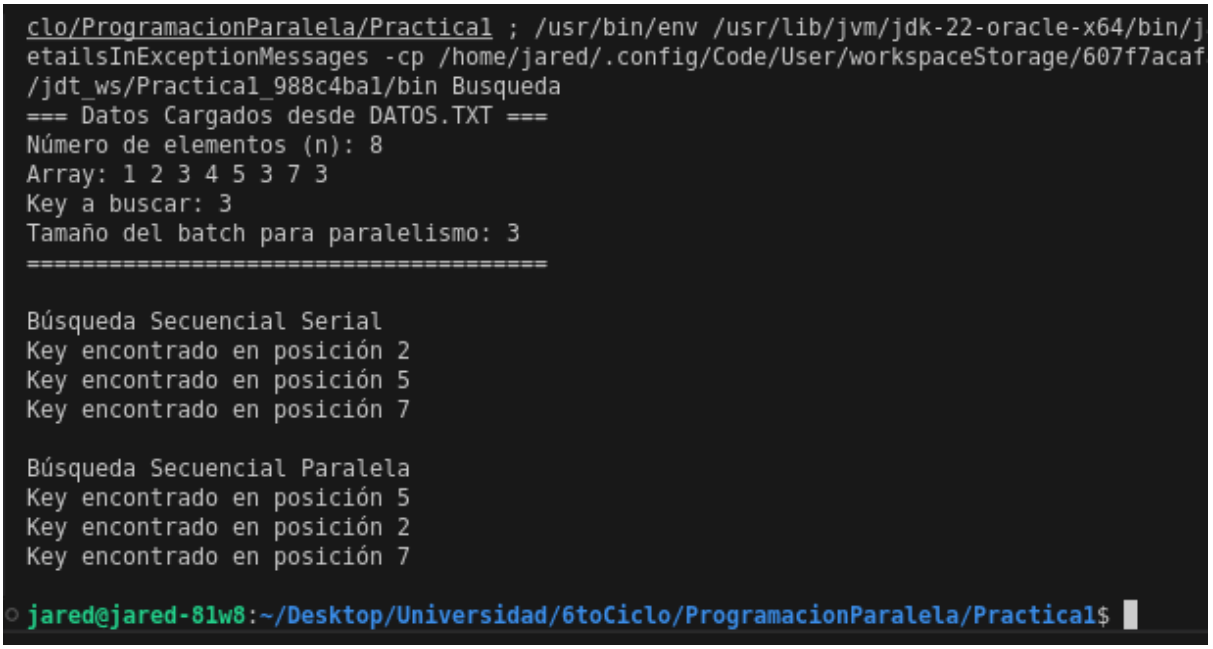
Se logró calcular el tiempo del k-esimo hilo en estado “Run” y gracias a eso se pudo calcular la media, fue posible agregar la desviación estándar pero por motivos de tiempo.

```
4168.0  2500.0  4136.0  4784.0  3892.0  3570.0  4744.0  5162.0  3890.0  7520.0
ESTADISTICAS EN PARALELO:
Media de tiempo en paralelo (ms):0.034362
Total de tiempo (ms): 10.308628
==================================
AHORA EN TIEMPO SERIAL!
Matriz A:
0.0     1.0     1.0     0.0     1.0     1.0     1.0     1.0     0.0     1.0
1.0     0.0     1.0     0.0     0.0     1.0     1.0     0.0     0.0     1.0
1.0     1.0     0.0     0.0     1.0     1.0     1.0     1.0     0.0     1.0
1.0     0.0     1.0     0.0     0.0     0.0     1.0     1.0     1.0     1.0
0.0     1.0     1.0     1.0     1.0     1.0     1.0     1.0     1.0     1.0
0.0     0.0     1.0     1.0     0.0     1.0     0.0     0.0     0.0     1.0
0.0     0.0     0.0     1.0     0.0     0.0     0.0     0.0     0.0     1.0
1.0     0.0     0.0     1.0     0.0     0.0     1.0     1.0     1.0     1.0
0.0     1.0     1.0     1.0     1.0     1.0     1.0     0.0     0.0     1.0
1.0     0.0     0.0     1.0     1.0     0.0     0.0     1.0     1.0     1.0
Matriz A^N
21.0    13.0    21.0    25.0    20.0    19.0    24.0    26.0    19.0    39.0
15.0    8.0     14.0    17.0    13.0    13.0    16.0    18.0    14.0    26.0
22.0    13.0    20.0    25.0    20.0    19.0    24.0    26.0    19.0    39.0
19.0    11.0    19.0    23.0    17.0    17.0    22.0    23.0    18.0    35.0
28.0    18.0    27.0    33.0    27.0    26.0    32.0    34.0    24.0    52.0
12.0    9.0     12.0    15.0    13.0    12.0    15.0    16.0    11.0    23.0
6.0     6.0     6.0     8.0     8.0     6.0     9.0     9.0     6.0     12.0
19.0    12.0    17.0    22.0    18.0    16.0    22.0    23.0    17.0    34.0
21.0    14.0    20.0    25.0    21.0    20.0    24.0    26.0    18.0    39.0
22.0    15.0    21.0    27.0    21.0    21.0    27.0    26.0    19.0    41.0
ESTADISTICAS EN SERIAL:
Tiempo total (ms): 98864.0
```

Problema 3

Logre hacerlo de 2 formas, uno con CLI y otro desde un archivo TXT





CODIGO PRUEBA:

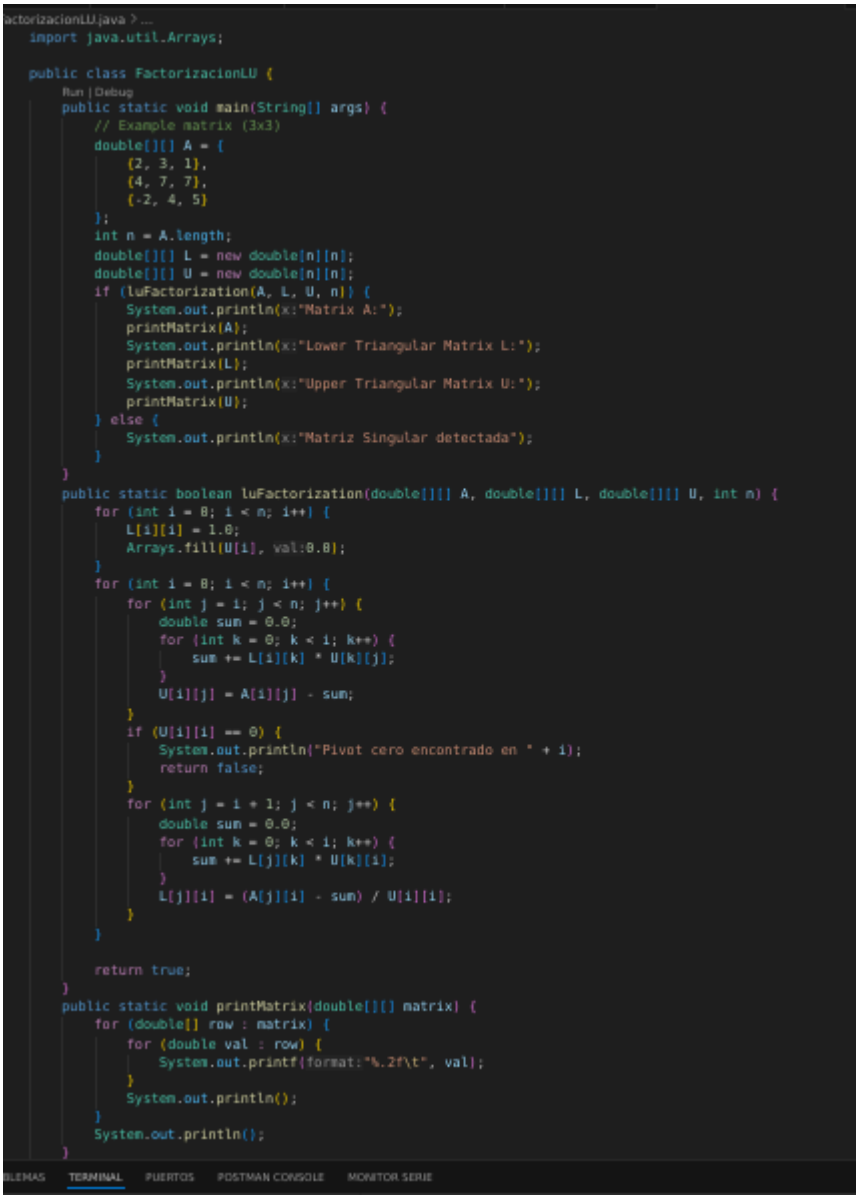
```java
//Nuestra estrategia será separar la secuencia en batches

import java.util.Scanner;

public class BusquedaSecuencial {
    public static int key;
    public static int[] arr;
    public static void serialSearch(int a,int b){
        for(int i=a;i<b;i++){
            if(arr[i] == key){
                System.out.println("Key encontrado en posición "+ String.valueOf(i));
            }
        }
    }
    Run | Debug
    public static void main(String[] args) throws InterruptedException {
        Scanner scan = new Scanner(System.in);
        int n;
        System.out.println(x:"Cuantos elementos deseas agregar?");
        n = scan.nextInt();
        System.out.println(x:"Ingresa los elementos que desees");
        arr = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = scan.nextInt();
        }
        System.out.println(x:"Que elemento estas buscando?");
        key = scan.nextInt();

        //ALGORITMO
        //Eligiré k batches
        System.out.println(x:"Longitud de los batches para dividir tu array:");
        int len = scan.nextInt();
        int batches = (n % len == 0)? n/len: n/len + 1;

        //Instanciando mis hilos a usar
        Thread[] hilos = new Thread[batches];

        for(int i=0;i<batches;i++){
            final int st = i * len; // inicio
            final int end = ((i+1)*len > n)? n:(i+1)*len; //En caso me exceda!
            hilos[i] = new Thread(()->{
                serialSearch(st, end);
            });
            hilos[i].start();
        }
        for(Thread x: hilos){
            x.join();
        }
        scan.close();
    }
}
```

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Busqueda{
    public static int key;
    public static int[] arr;
    public static void serialSearch() {
        System.out.println(x:"Busqueda Secuencial Serial\");
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == key) {
                System.out.println("Key encontrado en posicion " + i);
            }
        }
        System.out.println();
    }

    public static void serialSearch_por_Batch(int a,int b){
        for(int i=a;i<b;i++){
            if(arr[i] == key){
                System.out.println("Key encontrado en posicion "+ String.valueOf(i));
            }
        }
    }

    public static void parallelSearch(int len) throws InterruptedException {
        System.out.println(x:"Busqueda Secuencial Paralela");
        int n = arr.length;
        int batches = (n%len==0)? n/len: n/len+1;

        Thread[] hilos = new Thread[batches];
        for (int i = 0; i < batches; i++) {
            final int st = i * len; // inicio
            final int end = ((i + 1) * len > n) ? n : (i + 1) * len; // En caso se exceda
            hilos[i] = new Thread(() -> {
                serialSearch_por_Batch(st, end);
            });
            hilos[i].start();
        }

        // Esperar a que todos los hilos terminen
        for (Thread x : hilos) {
            x.join();
        }
        System.out.println();
    }

    Run | Debug
    public static void main(String[] args) throws InterruptedException {
        String fileName = "DATOS.txt";

        try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            //Leer numero de elementos
            String line = br.readLine();
            int n = Integer.parseInt(line.trim());
            //Leer elementos del array
            line = br.readLine();
            String[] elements = line.trim().split(regex:"\\s+");
            arr = new int[n];
            for (int i = 0; i < n; i++) {
                arr[i] = Integer.parseInt(elements[i]);
            }
            //En esta linea debo estar la key
            line = br.readLine();
            key = Integer.parseInt(line.trim());

            //Leer el tamaño del batch para búsqueda paralela
            line = br.readLine();
            int len = Integer.parseInt(line.trim());

            // Mostrar datos leídos
            System.out.println(x:"=== Datos Cargados desde DATOS.TXT ===");
            System.out.println("Número de elementos (n): " + n);
            System.out.print(s:"Array: ");
            for (int num : arr) {
                System.out.print(num + " ");
            }
            System.out.println("\nKey a buscar: " + key);
            System.out.println("Tamaño del batch para paralelismo: " + len);
            System.out.println(x:"=====================================\n");

            serialSearch();
            parallelSearch(len);
        } catch (IOException e) {
            System.err.println("Error al abrir archivo: " + e.getMessage());
        }
    }
}
```

Pregunta 4:

Hice el análisis de la factorizacion LU:

CODIGO PRUEBA:



```
actorizacionLU.java > ...
import java.util.Arrays;

public class FactorizacionLU {
    Run | Debug
    public static void main(String[] args) {
        // Example matrix (3x3)
        double[][] A = {
            {2, 3, 1},
            {4, 7, 7},
            {-2, 4, 5}
        };
        int n = A.length;
        double[][] L = new double[n][n];
        double[][] U = new double[n][n];
        if (luFactorization(A, L, U, n)) {
            System.out.println(x:"Matrix A:");
            printMatrix(A);
            System.out.println(x:"Lower Triangular Matrix L:");
            printMatrix(L);
            System.out.println(x:"Upper Triangular Matrix U:");
            printMatrix(U);
        } else {
            System.out.println(x:"Matriz Singular detectada");
        }
    }
    public static boolean luFactorization(double[][] A, double[][] L, double[][] U, int n) {
        for (int i = 0; i < n; i++) {
            L[i][i] = 1.0;
            Arrays.fill(U[i], val:0.0);
        }
        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                double sum = 0.0;
                for (int k = 0; k < i; k++) {
                    sum += L[i][k] * U[k][j];
                }
                U[i][j] = A[i][j] - sum;
            }
            if (U[i][i] == 0) {
                System.out.println("Pivot cero encontrado en " + i);
                return false;
            }
            for (int j = i + 1; j < n; j++) {
                double sum = 0.0;
                for (int k = 0; k < i; k++) {
                    sum += L[j][k] * U[k][i];
                }
                L[j][i] = (A[j][i] - sum) / U[i][i];
            }
        }

        return true;
    }
    public static void printMatrix(double[][] matrix) {
        for (double[] row : matrix) {
            for (double val : row) {
                System.out.printf(format:"%.2f\t", val);
            }
            System.out.println();
        }
        System.out.println();
    }
BLEMAS  TERMINAL  PUERTOS  POSTMAN CONSOLE  MONITOR SERIE
```