Práctica Individual 1 – Ejercicios iterativos, recursivos y notación funcional. Análisis empírico de la complejidad.

EJEMPLOS:

1. Un punto es un tipo con las siguientes propiedades:
   - X, Double, básica, individual
   - Y, Double, básica, individual
   - Cuadrante, Cuadrante, derivada, individual. Enumerado {PRIMER_CUADRANTE, SEGUNDO_CUADRANTE, TERCER_CUADRANTE, CUARTO_CUADRANTE}.

```
public static Map<Cuadrante, Double> solucionFuncional (List<Punto2D> ls) {
            return ls.stream()
                  .collect(Collectors.groupingBy(Punto2D::cuadrante,
                  Collectors.reducing(0., p -> p.x(), (a, b) -> a + b)));
}
```

Analice el código que se muestra y proporcione una solución iterativa y otra recursiva final equivalentes.

2. Dada la siguiente definición recursiva de la función f (que toma como entrada 2 números enteros positivos y devuelve una cadena):

$$f(a,b) = \begin{cases} "(" + toString(a*b) + ")", & a < 5 \lor b < 5 \\ toString(a+b) \ + \ f(a/2, b-2), & en\ otro\ caso \end{cases}$$

siendo + un operador que representa la concatenación de cadenas, y toString(i) un método que devuelve una cadena a partir de un entero. Proporcione una solución iterativa usando while, una recursiva no final, una recursiva final, y una en notación funcional.

3. Dada la siguiente definición recursiva de la función g (que toma como entrada 2 números enteros positivos y devuelve un entero):

$$g(a,b) = \begin{cases} a^2 + b, & a < 2 \ \lor b < 2 \\ g\left(\frac{a}{2}, b-1\right) + g\left(\frac{a}{3}, b-2\right), & en\ otro\ caso \end{cases}$$

Proporcione una solución recursiva sin memoria, otra recursiva con memoria, y otra iterativa.

4. Implementar de forma recursiva e iterativa un algoritmo para el cálculo de la potencia a^n, siendo a de tipo Double y n de tipo Integer. Analizar sus tiempos de ejecución.

5. Analizar los tiempos de ejecución de las versiones recursivas sin y con memoria, iterativa y haciendo uso de la potencia de matrices para el cálculo de los números de Fibonacci. Comparar según los resultados sean de tipo Double o BigInteger.

EJERCICIOS:

1. Analice el código que se muestra a continuación:

```
public static Map<Integer,List<String>>  ejercicioA (Integer varA, String varB, Integer varC, String
varD, Integer varE) {
        UnaryOperator<EnteroCadena> nx = elem ->
                {
                        return EnteroCadena.of(elem.a()+2,
                                        elem.a()%3==0?
                                        elem.s()+elem.a().toString():
                                        elem.s().substring(elem.a()%elem.s().length()));
                };

        return Stream.iterate(EnteroCadena.of(varA,varB), elem -> elem.a() < varC, nx)
                                .map(elem -> elem.s()+varD)
                                .filter(nom -> nom.length() < varE)
                                .collect(Collectors.groupingBy(String::length));
}
```

donde EnteroCadena es una clase con una propiedad entera *a* y otra de tipo cadena *s*, la cual debe implementar como un record.

SE PIDE: Proporcione una solución iterativa y otra recursiva final equivalentes.

2. Dada la siguiente definición recursiva de la función f (que toma como entrada 2 números enteros positivos, y devuelve una lista de enteros):

$$f(a,b) = \begin{cases} \text{[a*b]}, & a < 2 \lor b < 2 \\ f(a\%b, b-1) + a, & a > b \\ f(a-2, b/2) + b, & eoc \end{cases}$$

donde:

- *[elem]*: representa una lista de un único elemento *elem*
- *list* + *elem*: representa la inserción del elemento *elem* al final de la lista *list*

SE PIDE: Proporcione una solución recursiva no final, una recursiva final, una iterativa usando while, y una en notación funcional.

3. Dada la siguiente definición recursiva de la función g:

$$g(a,b,c) = \begin{cases} a + b^2 + 2 * c, & a < 3 \lor b < 3 \lor c < 3 \\ g(a-1, b/2, c/2) + g(a-3, b/3, c/3), & a \text{ es múltiplo de } b \\ g\left(\frac{a}{3}, b-3, c-3\right) + g\left(\frac{a}{2}, b-2, c-2\right), & \text{en otro caso} \end{cases}$$

siendo a, b y c números enteros positivos.

SE PIDE: Proporcione una solución recursiva sin memoria, otra recursiva con memoria, y otra iterativa.

4. Dada la siguiente definición recursiva de la función f (que toma como entrada un número entero positivo):

$$h(a) = \begin{cases} 5, & a < 10 \\ Math.sqrt(3 * a) * h(a-2), & en\ otro\ caso \end{cases}$$

SE PIDE: Analizar los tiempos de ejecución de la versiones recursiva e iterativa de dicha función, comparando según los resultados sean de tipo Double o BigInteger. Para ello, debe obtener 5 gráficas:

a) La que se obtiene cuando los resultados son de tipo Double usando un esquema recursivo.
b) La que se obtiene cuando los resultados son de tipo BigInteger usando un esquema recursivo.
c) La que se obtiene cuando los resultados son de tipo Double usando un esquema iterativo.
d) La que se obtiene cuando los resultados son de tipo BigInteger usando un esquema iterativo.
e) La que se obtiene combinando las 4 anteriores.

5. Dada la siguiente definición recursiva de la función f (que toma como entrada un número entero positivo):

$$k(n) = \begin{cases} 1, & n \leq 6 \\ 1 + k(n-1) * log2(n-1), & n > 6 \end{cases}$$

```
public static int log2(int n){
    if(n <= 0) throw new IllegalArgumentException();
    return 31 - Integer.numberOfLeadingZeros(n);
}
```

SE PIDE: Analizar los tiempos de ejecución de las versiones recursiva e iterativa de dicha función, comparando según los resultados sean de tipo Double o BigInteger. Para ello, debe obtener 5 gráficas:

a) La que se obtiene cuando los resultados son de tipo Double usando un esquema recursivo.
b) La que se obtiene cuando los resultados son de tipo BigInteger usando un esquema recursivo.
c) La que se obtiene cuando los resultados son de tipo Double usando un esquema iterativo.
d) La que se obtiene cuando los resultados son de tipo BigInteger usando un esquema iterativo.
e) La que se obtiene combinando las 4 anteriores.

Debe resolver todos los ejercicios de forma <u>eficiente</u>. Tenga en cuenta que:

- Para cada ejercicio debe leer los datos de entrada de un fichero, y mostrar la salida por pantalla. Dicha lectura debe ser independiente del algoritmo concreto que resuelva el ejercicio.
- La solución tiene que ser acorde al material de la asignatura proporcionado.