Práctica 2
Complejidad

Estructuras de datos

1. Meta

Que el alumno visualice el concepto de "función de complejidad computacional".

2. Objetivos

Al finalizar la práctica el alumno será capaz de:

- Medir la complejidad en número de operaciones de un método de manera experimental.
- Comparar el desempeño entre las versiones iterativas y recursivas de un método.

3. Antecedentes

3.1. Sucesión de Fibonacci.

La sucesión de fibonacci fue descubierta por Fibonacci en relación a un problema de conejos. Supongamos que se tiene una pareja de conejos y cada mes esa pareja cría una nueva pareja. Después de dos meses, la nueva pareja se comporta de la misma manera. Entonces, el número de parejas nuevas nacidas a_n en el n-ésimo mes es $a_{n\text{-}1} + a_{n\text{-}2}$, ya que nace una pareja por cada pareja nacida en el mes anterior y cada pareja nacida hace dos meses cria una nueva pareja. Por convención consideremos $a_0 = 0$ y $a_1 = 1$.

Actividad 1

Define, con estos datos, la función de fibonacci.

3.2. Triángulo de Pascal.

En la figura Figura 1 se muestran algunos términos del Triángulo de Pascal.

Matemáticamente, podemos definir el elemento $Pascal_{ij}$ que corresponde al elemento en la fila i, columna j de la siguiente manera:

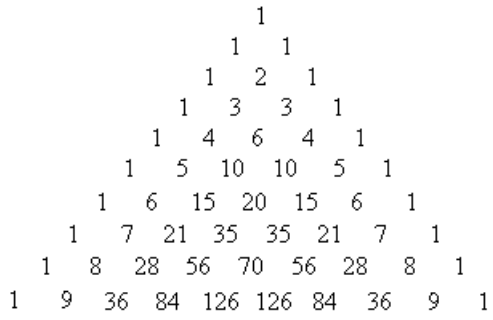

Figura 1: Triángulo de Pascal

$$Pascal_{ij} = \begin{cases} 1 & si\ i = 0 \text{ ó } j = 0 \\ Pascal_{(i-1)(j-1)} + Pascal_{(i-1)(j)} & si\ i! = 0 \text{ y } j! = 0 \end{cases} \quad (1)$$

4. Desarrollo

La práctica consiste en implementar métodos que calculen el triángulo de pascal y el n-ésimo número de fibonacci. Esto se realizará en forma recursiva e iterativa. Además se deberá entregar un reporte sobre el número de operaciones que realiza cada método. Para esto se deberá implementar la interfaz `RecursionInterfaz` en una clase llamada `Recursion`.

```
1  package Practica2;
2
3  /**
4   * Calcula el número de operaciones que tarda un método en ser ejecutado.
5   */
6  public interface RecursionInterfaz{
7
8      /**
9       * Asigna el nombre del archivo en el cual se añadirá el reporte
10      * del número de operaciones que tardó el último método ejecutado.
11      */
12     public void asignaNombreDeArchivo(String archivo);
13
14     /**
15      * Método para calcular, de forma recursiva, el elemento en la fila
16      * <code>i</code>, en la columna <code>j</code> del triángulo de Pascal
17      * @param i el número de fila
18      * @param j el número de columna
19      * @return El elemento en la <code>i</code>-ésima fila y la
20      * <code>j</code>-ésima columna del triángulo de Pascal.
21      */
22     public int tPascalRec(int i, int j);
23
24     /**
25      * Método para calcular, iterativamente, el elemento en la fila
26      * <code>i</code> y la columna <code>j</code> del triángulo de Pascal
27      * @param i el número de fila
28      * @param j el número de columna
29      * @return El elemento en la i-ésima fila y la j-ésima columna del triángulo
30      * de Pascal.
```

```
31          */
32         public int tPascalIt(int i,int j);
33
34         /**
35          * Devuelve el n-ésimo elemento, calculado de forma recursiva,
36          * de la sucesión de Fibonacci
37          * @param n el índice del elemento que se desea calcular
38          * @return el n-ésimo elemento de la sucesión de Fibonacci
39          */
40         public int fibonacciRec(int n);
41
42         /**
43          * Devuelve el n-ésimo elemento, calculado de forma iterativa,
44          * de la sucesión de Fibonacci
45          * @param n el índice del elemento que se desea calcular
46          * @return el n-ésimo elemento de la sucesiónd de Fibonacci
47          */
48         public int fibonacciIt(int n);
49
50 }
```

5. Comandos útiles para la práctica. Gnuplot

Gnuplot es una herramienta interactiva que permite generar gráficas a partir de archivos de datos planos. Para esta práctica, los datos deben ser guardados en un archivo de este tipo y graficados con gnuplot. Supongamos que el archivo donde se guardan es llamado **datos.dat**. Por ejemplo, para el método de fibonacci el archivo **datos.dat** tendría algo semejante al siguiente contenido:

```
1 100
2 250
3 300
5 575
```

donde la primer columna es el valor del argumento y la segunda, el número de operaciones.

5.1. Gráficas en 2D

Al iniciar el programa `gnuplot` aparecerá un promt y se puede iniciar la sesión de trabajo. A continuación se muestra cómo crear una gráfica 2D.

- gnuplot> set title “Mi gráfica” //Título para la gráfica
- gnuplot> set xlabel “Eje X” //Título para el eje X
- gnuplot> set ylabel “Eje y” //Título para el eje Y
- gnuplot> plot “datos.dat” using 1:2 with lines lc rgb 'blue' //graficamos los datos
- gnuplot> set terminal pngcairo size 1000,1000 //algunas características de la imagen que se guardará
- gnuplot> set output 'fib.png' //nombre de la imagen que se guardará
- replot //lo graficamos para que se guarde en la imagen

5.2. Gráficas en 3D

A continuación se muestra cómo crear una gráfica 3D. Observa que, en este caso, el archivo de datos requiere tres columnas.

- gnuplot> set Title "Mi gráfica"
- gnuplot> set xlabel "Eje X"
- gnuplot> set ylabel "Eje Y"
- gnuplot> set zlabel "Eje Z"
- gnuplot> splot "datos.dat" using 1:2:3 with lines lc rgb 'blue'
- gnuplot> set terminal pngcairo size 1000,1000
- gnuplot> set output 'fib.png'
- replot

6. Ejercicios

1. Crea la clase `Recursion`, que implemente `RecursionInterfaz`. Agrega las firmas de los métodos requeridos y asegúrate de que compile, aunque aún no realize los cálculos.
2. Programa los métodos indicados en la interfaz.
3. Crea un método `main` que mande llamar los métodos programados para diferentes valores de sus parámetros. Entre método y método cambia el nombre del archivo en el cual serán guardados (Para ello es el método `asignaNombreDeArchivo`.
4. Para realizar el análisis de complejidad, modifica los métodos agregando una variable, que sea incrementada en cada iteración/recursión y guarde el número de operaciones realizadas en el archivo. Observa que para los métodos recursivos necesitarás que esta variable sea un atributo de clase. Para los iterativos puede ser una variable local. ¿A qué se debe la diferencia?
5. Para el método de fibonacci, genera las gráficas n(entrada) vs número de operaciones y haz un análisis de lo que sucede.
6. Para el método recursivo del cálculo del triángulo de Pascal, genera una gráfica en 3-D en donde el parámetro renglón se encontrará en el eje X, el parámetro columna se encontrará en el eje Y, el número de operaciones en el eje Z y haz un análisis de lo que sucede.

7. Forma de entrega

- Asegúrate de borrar todos los archivos generados, incluyendo archivos de respaldo usando `ant clean`.
- Copia el directorio `Practica2` dentro de un directorio <`apellido1_appellido2`> donde `apellido1` es el primer apellido de un miembro del equipo y `apellido2` es el primer apellido del otro miembro. Por ejemplo: `marquez_vazquez`.
- Borra el directorio libs en la copia.

- Agrega un archivo `reporte.pdf` dentro del directorio `Practica2` de la copia, con el nombre completo de los integrantes del equipo, la resolución de la sección anterior y cualquier comentario que quieran hacer sobre la práctica.
- Comprime el directorio `<apellido1_appelido2>` en un archivo `<apellido1_apellido2>.tar.gz`. Por ejemplo: `marquez_vazquez.tar.gz`.
- Sube este archivo en la sección correspondiente del aula virtual.

NOTA: Sólo una persona lo sube a la plataforma. No olviden incluir el nombre de ambas personas del equipo dentro del `reporte.pdf`.

8. Forma de evaluación

Para tu calificación final se tomarán en cuenta los aspectos siguientes:

50 %	Calificación generada por las pruebas automáticas. Usaremos nuestros archivos, por lo que si realizas modificaciones a tus copias, éstas no tendrán efecto al momento de calificar.
40 %	Documentación completa y adecuada. Entrega en el formato requerido, sin archivos `.class`, respaldos, bibliotecas de `JUnit` u otros no requeridos.
10 %	Revisión manual del código, para verificar que se cumpla con las especificaciones, que no se haya copiado, etc.