Práctica 3 – Aproximación diofántica

Definición del problema y objetivo

El objetivo de esta práctica es estudiar la aproximación diofántica, un algoritmo para aproximar un número real con una fracción (un número racional). En el documento DnC-lab.pdf (disponible en el campus virtual) puedes encontrar la definición del problema y el algoritmo.

Realización de la práctica

El estudiante tiene que implementar el algoritmo de aproximación diofántica descrito en las transparencias en el método `_approximate` de la clase `BinarySearchDiophantineApproximator`. Este método recibe como parámetro el número real que hay que aproximar y la tolerancia (épsilon) con el que hay que aproximarlo. Este método devuelve la fracción que aproxima al número real como un objeto RationalNumber. Además del método _approximate, hay que implementar el método `getMiddlePoint` de la clase `MediantApproximation` que recibe como parámetro dos números racionales (que representan los extremos de un intervalo) y devuelve un racional que es el mediante.

Además de las clases `BinarySearchDiophantineApproximator` y `MediantApproximation`, se proporcionan las siguientes clases y librerías:

- diophantineApprox1.6.jar: esta librería contiene clases necesarias para la implementación de los métodos _approximate y getMiddlePoint, como por ejemplo la clase RationalNumber. Para importar la librería en el proyecto eclipse:
  - Copiar el fichero jar en el proyecto. Por ejemplo, puedes crear una carpeta lib dentro del proyecto para mantener los ficheros organizados.
  - Sobre el nombre del proyecto, pulsa el botón secundario y selecciona Configure Build Path > Pestaña Libraries
  - Si en el área central te aparecen Module Path y Class Path selecciona Class Path. Si no te aparecen estas opciones, sigue con el siguiente paso.
  - Pulsa el botón Add JARs y selecciona el fichero diophantineApprox1.6.jar
- `TestBinarySearchApproximation`: en esta clase se encuentra en método main (no hay que modificarlo). El método main recibe en los argumentos de entrada el modo de ejecución, que puede ser uno de los siguientes:
  - –n <numero> <epsilon>: el algoritmo aproxima el *número* que recibe como parámetro con un número racional con la tolerancia dada.
  - –f <input-file>: El fichero *input-file* contiene en cada línea un número real y una tolerancia. Con el modo -f el algoritmo aproxima todos los números del fichero con la tolerancia correspondiente.
  - – r <max_digits> <max_num>: en este modo se generan *max_num* números aleatorios con longitud *max_digits* y calcula las aproximaciones con diferentes tolerancias, comenzando con tolerancia 0.1 y hasta alcanzar tolerancia 10^{-16}. Es decir que calcula la aproximación de los *max_num* con tolerancia 0.1, luego con 0.01 y así hasta tolerancia 10^{-16}. El programa genera un fichero con los tiempos de ejecución que se puede utilizar para realizar un estudio experimental de la complejidad del algoritmo.
- Fichero de prueba (numbers.txt) para usar con el modo -f y la correspondiente salida (numbers.out).
- analyze.R: este fichero contiene un script que procesa los ficheros que se generan con el modo –r. Hay una guía en el campus que explica como instalar el entorno R y ejecutar el script.

Entrega y evaluación

La evaluación de la práctica consiste en la entrega y evaluación automática del código del alumno y una memoria (de no más de 3 páginas) en la que el alumno presenta los objetivos, el entorno de ejecución donde se ha realizado las pruebas, los resultados teóricos esperados y los resultados que ha obtenido en el estudio experimental de la complejidad del algoritmo. A continuación, se detalla la nota que se puede obtener:

Nota	Descripción
0	No pasa el test en laboratorio ni entrega memoria o el sistema detecta plagio en el código o la memoria
[1,4]	No pasa el test en laboratorio. Entrega la memoria y pasa los test fuera de plazo
[4,6]	Pasa parcialmente los test de laboratorio. Entrega memoria y pasa test fuera de plazo
[6,10]	Pasa los test en laboratorio y entrega memoria

Para la evaluación automática del código el estudiante debe subir el fichero MediantApproximation.java y el fichero BinarySearchDiophantineApproximator.java a una actividad del campus que realizará una serie de pruebas sobre la implementación. El código hay que subirlo siempre al final de la sesión de laboratorio correspondiente, y si no se pasan todas las pruebas, el alumno tendrá una segunda oportunidad fuera de plazo. La fecha se comunicará al final de la sesión de laboratorio.