Universidad Rey Juan Carlos

Práctica 2: Divide y Vencerás

Diseño y Análisis de Algoritmos

- Algunos códigos tendrán que probarse con el juez automático **DOMjudge**
  - gibson.escet.urjc.es
  - El nombre de usuario será "team-XXX", donde XXX es un número de 3 cifras único por cada alumno. Podéis ver qué número os corresponde en un documento subido al aula virtual que describe la relación entre el nombre de usuario y el nombre de un alumno.
  - La contraseña es vuestro DNI (incluida la letra final en mayúsculas).
- Además de probar vuestros códigos con DOMjudge debéis subir los ficheros fuente al aula virtual en un solo archivo (.zip o .rar)
- El ejercicio sobre el fractal NO se probará en DOMjudge
- No se entregará una memoria
- Fecha límite: Se especificará en el campus virtual
- 10 % de la nota final

Índice

1. Ordenar impares-pares (5 puntos)

1.1. Introducción

En este ejercicio el objetivo es implementar una variante del algoritmo *mergesort*.

1.2. Enunciado del problema

Dada una lista de números enteros, el problema consiste en reordenar sus elementos de izquierda a derecha de tal forma que primero aparezcan los números impares ordenados de menor a mayor, y después los pares, también ordenados de menor a mayor.

El algoritmo a desarrollar seguirá la estrategia de **divide y vencerás**, pero no podrá usar el algoritmo *mergesort* ni otro algoritmo de ordenación conocido. Por ejemplo, no se dará por válido un algoritmo que primero extraiga los elementos pares e impares, y luego los ordene independientemente.

1.2.1. Descripción de la entrada

La entrada contiene, en su primera línea, el número de enteros que contendrá la lista de entrada. En la siguiente línea aparecerán los n enteros de la lista, separados por espacios en blanco.

1.2.2. Descripción de la salida

La salida contendrá una línea con los enteros de la lista "ordenada", separados por espacios en blanco. Después del último entero no hay un espacio en blanco, sino un salto de línea.

1.2.3. Ejemplo de entrada

```
13↵
4 -5 6 1 8 13 0 -2 2 4 3 1 7↵
```

1.2.4. Salida para el ejemplo de entrada

```
-5 1 1 3 7 13 -2 0 2 4 4 6 8↵
```

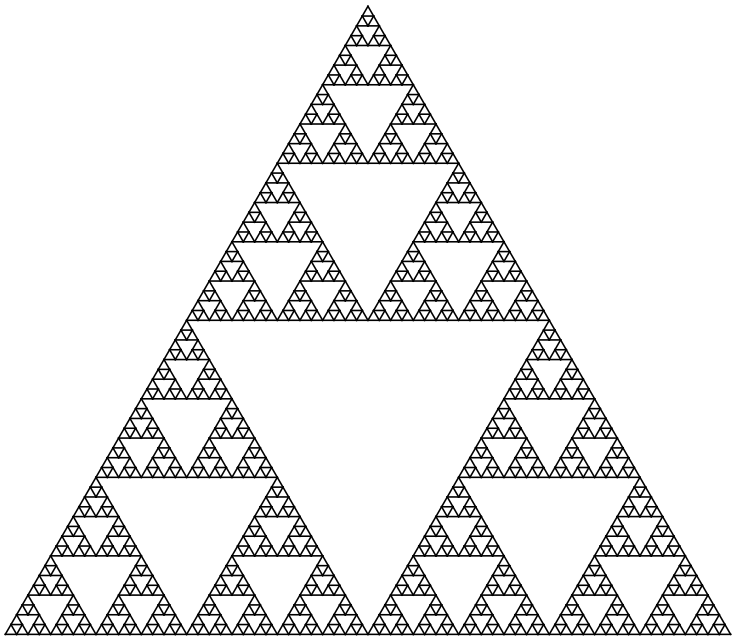

Figura 1: Triángulo de Sierpiński.

2. Triángulo de Sierpiński (5 puntos)

El triángulo de Sierpiński (véase la Figura 1) es uno de los fractales más conocidos. En este ejercicio se pide implementar un método recursivo que dibuje un triángulo de Sierpiński de orden n. Se considerará que un triángulo de Sierpiński de orden 0 es un simple triángulo, mientras que el de orden 1 estaría formado por tres triángulos, tal y como se ilustra en la Figura 2. El código debe ser similar a los ejemplos del libro "Introduction to Recursive Programming". Por ejemplo, se deberá usar la librería matplotlib para dibujar imágenes y segmentos. El código NO debe probarse en DOMjudge.

Orden 0 Orden 1

Figura 2: Triángulos de Sierpiński de orden 0 y 1.