Universidad Rey Juan Carlos

Práctica 2: Divide y Vencerás

Diseño y Análisis de Algoritmos

- Los 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 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. Imprimir polinomios (1 punto)

1.1. Introducción

En este ejercicio implementaréis una función auxiliar sencilla que se utilizará en otro ejercicio. No debéis seguir un esquema de divide y vencerás en este ejercicio.

1.2. Enunciado del problema

En esta práctica trabajaremos con polinomios, los cuales vamos a especificar mediante listas. En concreto, una lista **a**, de longitud n, representará a un polinomio $P(x)$ de grado $n-1$. Por ejemplo, la lista $\mathbf{a} = [3, -5, 0, 1]$ especifica el polinomio $P(x) = x^3 - 5x^1 + 3$. Por tanto, a_i será el coeficiente asociado al término x^i.

En este ejercicio se pide desarrollar una función recursiva que reciba una lista **a** representando a un polinomio, y lo escriba en la consola de manera formateada. Por ejemplo, dado $\mathbf{a} = [3, -5, 0, 1]$, que representa a $x^3 - 5x^1 + 3$, el programa deberá imprimir:

```
␣+␣1x^3␣-␣5x^1␣+␣3↵
```

Para cada término $a_k x^k$ se escribirá la cadena "␣`+`␣" si $a_k \geqslant 0$, o "␣`-`␣" en caso contrario (notad que hay un espacio antes y después del carácter que indica el signo). Después se escribirá el coeficiente a_k seguido de una '`x`'. Después, si $k \geqslant 1$ se escribirá el carácter '`^`', seguido del exponente k.

Se asumirá que los coeficientes del polinomio (es decir, los elementos de **a**) serán números enteros. Además, el último elemento de la lista será distinto de 0 ($a_{n-1} \neq 0$), salvo que el polinomio sea la constante 0. Por último, asumid que $0 \leqslant n \leqslant 100$.

1.2.1. Descripción de la entrada

La primera línea de la entrada contendrá el grado del polinomio $P(x)$ (es decir, $n-1$). La segunda línea contendrá los n elementos de la lista **a**, los cuales estarán separados por espacios en blanco.

1.2.2. Descripción de la salida

La salida contendrá la salida formateada de $P(x)$ terminada en un salto de línea.

1.2.3. Ejemplo de entrada 1

```
3↵
3␣-5␣0␣1↵
```

1.2.4. Salida para el ejemplo de entrada 1

```
␣+␣1x^3␣-␣5x^1␣+␣3↵
```

1.2.5. Ejemplo de entrada 2

```
0↵
0↵
```

1.2.6. Salida para el ejemplo de entrada 2

```
␣+␣0↵
```

2. Sumar y restar polinomios (1 punto)

2.1. Introducción

En este ejercicio implementaréis una función auxiliar sencilla que se utilizará en otro ejercicio. No debéis seguir un esquema de divide y vencerás en este ejercicio.

2.2. Enunciado del problema

El objetivo de este ejercicio es crear dos métodos recursivos que sumen y resten polinomios, tal y como se han descrito en el primer ejercicio. Se pedirá al usuario que introduzca un polinomio $P(x)$ y otro $Q(x)$, y el método debe imprimir $P(x) + Q(x)$ y $P(x) - Q(x)$ de manera formateada, usando el método desarrollado en el primer ejercicio.

2.2.1. Descripción de la entrada

La primera línea de la entrada contendrá el grado del polinomio $P(x)$ (es decir, $n-1$). La segunda línea contendrá los n elementos de la lista que define al polinomio, los cuales estarán separados por espacios en blanco. La tercera línea contendrá el grado del polinomio $Q(x)$ (es decir, $m-1$). La cuarta línea contendrá los m elementos de la lista que define al polinomio, los cuales estarán separados por espacios en blanco.

2.2.2. Descripción de la salida

La salida contendrá dos líneas. En la primera se escribirá de manera formateada $P(x) + Q(x)$, mientras que la segunda (terminada en un salto de línea) contendrá $P(x) - Q(x)$.

2.2.3. Ejemplo de entrada

```
3↵
3 5 0 1↵
4↵
2 1 4 9 1↵
```

2.2.4. Salida para el ejemplo de entrada

```
 + 1x^4 + 10x^3 + 4x^2 + 6x^1 + 5↵
 - 1x^4 - 8x^3 - 4x^2 + 4x^1 + 1↵
```

3. Multiplicar polinomios (4 puntos)

3.1. Introducción

En este ejercicio implementaréis una variante del algoritmo de Karatsuba.

3.2. Enunciado del problema

El objetivo de este ejercicio es implementar un método que multiplique dos polinomios P, de grado $p-1$, y Q, de grado $q-1$:

$$P = u_{p-1}x^{p-1} + \cdots + u_1 x + u_0$$

$$Q = v_{q-1}x^{q-1} + \cdots + v_1 x + v_0$$

El método deberá estar basado necesariamente en la estrategia de divide y vencerás en la que se basa el algoritmo de Karatsuba. Para ello, observad que los polinomios se pueden escribir de la siguiente manera:

$$P = (\underbrace{u_{p-1}x^{p-m-1} + \cdots + u_m}_{P_a})x^m + \underbrace{u_{m-1}x^{m-1} + \cdots + u_0}_{P_b} = P_a x^m + P_b,$$

$$Q = (\underbrace{v_{q-1}x^{q-m-1} + \cdots + v_m}_{Q_a})x^m + \underbrace{v_{m-1}x^{m-1} + \cdots + v_0}_{Q_b} = Q_a x^m + Q_b,$$

donde $m = \text{mín}(\lfloor p/2 \rfloor, \lfloor q/2 \rfloor)$. De esta manera, podríamos expresar el producto como:

$$PQ = (P_a x^m + P_b)(Q_a x^m + Q_b) = P_a Q_a x^{2m} + (P_a Q_b + P_b Q_a)x^m + P_b Q_b$$

lo cual implica hallar cuatro productos de polinomios más sencillos que el original. Sin embargo, mediante la estrategia del algoritmo de Karatsuba podemos calcular el producto de manera más eficiente:

$$PQ = P_a Q_a x^{2m} + \left[(P_a + P_b)(Q_a + Q_b) - P_a Q_a - P_b Q_b\right] x^m + P_b Q_b$$

que solo requiere tres multiplicaciones de polinomios. En este ejercicio tendréis que implementar el producto siguiendo esta estrategia.

Los polinomios se especificarán tal y como se han descrito en el primer ejercicio. Además, el método deberá usar las funciones auxiliares para sumar y restar polinomios desarrolladas en el ejercicio 2. Por último, observad que multiplicar un polinomio $P(x)$ por x^m simplemente consiste en concatenar m ceros al principio de la lista que defina a $P(x)$.

3.2.1. Descripción de la entrada

La primera línea de la entrada contendrá el grado del polinomio $P(x)$ (es decir, p). La segunda línea contendrá los $p+1$ elementos de la lista que define al polinomio, los cuales estarán separados por espacios en blanco. La tercera línea contendrá el grado del polinomio $Q(x)$ (es decir, q). La cuarta línea contendrá los $q+1$ elementos de la lista que define al polinomio, los cuales estarán separados por espacios en blanco.

3.2.2. Descripción de la salida

La salida contendrá el producto $P(x) \cdot Q(x)$, escrito de manera formateada, y terminado en un salto de línea.

3.2.3. Ejemplo de entrada

```
3↵
3 5 0 1↵
4↵
2 1 4 9 1↵
```

3.2.4. Salida para el ejemplo de entrada

```
 + 1x^7 + 9x^6 + 9x^5 + 49x^4 + 49x^3 + 17x^2 + 13x^1 + 6↵
```

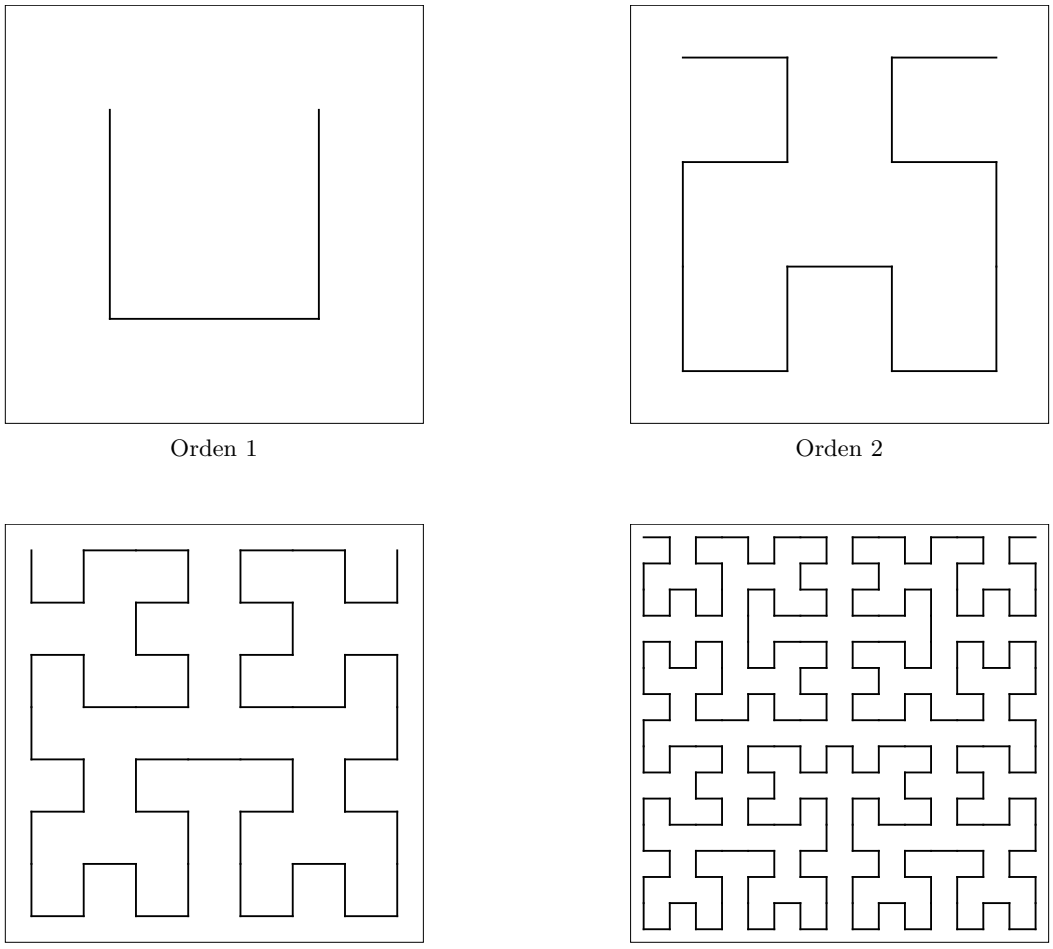

Figura 1: Hilbert curves of orders 1–6.

4. Curva de Hilbert (4 puntos)

Una curva de Hilbert es una curva fractal continua que rellena un área cuadrada. La figura muestra las primeras curvas de Hilbert. Mientras que aumenta el orden la longitud de la curva aumenta de manera exponencial y termina rellenando el cuadrado. Para construir el fractal de manera recursiva el problema para un determinado orden n se descompone en cuatro subproblemas de orden $n-1$, que deben orientarse apropiadamente. Además, los cuatro "subfractales" deben conectarse mediante tres segmentos, y son precisamente esos segmentos los que forman la totalidad de la curva. En este ejercicio se pide implementar un método recursivo que dibuje una curva de Hilbert de orden n. 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.