Guía de Ejercicios 10: Recursión algorítmica

Objetivos:

- Presentar la recursión algorítmica como una forma de encarar la resolución de problemas.
- Identificar los casos base y recursivo en problemas sencillos sobre enteros y listas.

Ejercicio 1. Escribir funciones recursivas para resolver los siguientes problemas.

- (a) pot2(n): Dado un entero $n \ge 0$, calcular 2^n . Ejemplos: pot2(0) $\to 1$; pot2(7) $\to 128$.
- (b) $pot_a(a, n)$: Dados dos enteros $n \ge 0$ y a, calcular a^n . Ejemplos: $pot_a(3, 0) \to 1$; $pot_a(-2, 7) \to -128$. Además, en particular $pot_a(0, 0)$ debe devolver 1.
- (c) producto(n, m): Dados dos enteros $n \ge 0$, m ≥ 0 , calcular n * m. Sugerencia: hacer la recursión sobre el parámetro n, dejando fijo el valor de m.
- (d) es_par(n): Determinar si un entero $n \ge 0$ es par (o sea, debe devolver True si es par y False en caso contrario). Sugerencia: pensar cómo podría ayudar restarle 2 a n.

Ejercicio 2. Escribir funciones recursivas para resolver los siguientes problemas.

- (a) productoria(xs): Dada una lista no vacía de enteros xs, calcular el resultado de multiplicar todos los números de xs.
- (b) cantidad_ocurrencias(x, xs): Dada una lista de enteros xs y un entero x, devolver la cantidad de veces que aparece x en xs.
- (c) max_pos(xs): Dada una lista no vacía de enteros xs, devolver la posición del elemento más grande. En caso de empate, devolver la posición de la primera aparición.
- (d) contar_coincidencias(xs): Dada una lista de enteros xs, contar cuántas veces es cierto que la i-ésima posición tiene el número i (es decir, cuántas veces xs[i]==i).
- (e) sumar_posiciones_pares(xs): Dada una lista de enteros, sumar los elementos en sus posiciones pares. Ejemplo: sumar_posiciones_pares([1,2,9,4,3]) devuelve 13 = 1 + 9 + 3.

Ejercicio 3. La sucesión de Fibonacci es una sucesión en la cual cada término se define como la suma de los dos anteriores, comenzando con 0 y 1. A continuación se observan los primeros diez términos de la sucesión:

$$F_0 = 0; F_1 = 1; F_2 = 1; F_3 = 2; F_4 = 3; F_5 = 5; F_6 = 8; F_7 = 13; F_8 = 21; F_9 = 34; \dots$$

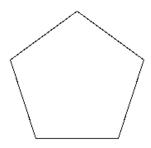
- (a) Escribir una función <u>recursiva</u> fibonacci (n) que dado un entero $n \ge 0$, devuelva el término F_n de la sucesión de Fibonacci.
- (b) Ejecutar fibonacci(50). ¿Por qué tarda tanto? (Probablemente sea necesario cortar la ejecución después de un tiempo.) Para buscar una respuesta (informal), hacer un seguimiento de la evaluación de un ejemplo chico, como fibonacci(6).

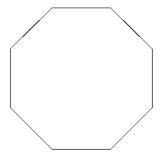
Módulo turtle de Python

En estos ejercicios, recomendamos ejecutar lo siguiente al comienzo de cada programa:

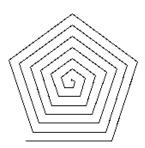
Ejercicio 4.

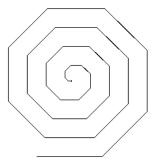
(a) Definir una función <u>iterativa</u> que, dados d:float y n:int, dibuje un polígono regular con n lados de longitud d. Por ejemplo, para n=5 y n=8:



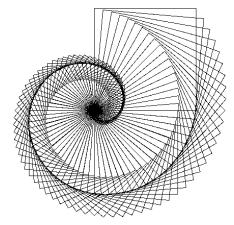


(b) Definir una función <u>recursiva</u> que, dados d:float, n:int e incr:float, dibuje una espiral con n lados, comenzando con longitud d y reduciéndose en incr cada vez hasta llegar a 0. Por ejemplo, para n=5 y n=8:





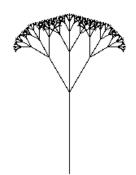
Ejercicio 5. Definir una función que dibuje la siguiente espiral de cuadrados. Escribir una variante iterativa y otra recursiva.



Ejercicio 6. Sea la función arbol definida de la siguiente manera:

```
import turtle as t
def arbol(altura:float, nivel:int):
    t.forward(altura)
    if nivel>0:
        t.left(30)
        arbol(altura/2, nivel-1)
        t.right(2 * 30)
        arbol(altura/2, nivel-1)
        t.left(30)
        t.left(30)
        t.backward(altura)
```

- (a) Ejecutar a mano arbol(100,0), arbol(100,1) y arbol(100,2), suponiendo que se empieza en cada caso con la tortuga apuntando hacia arriba.
- (b) Ejecutar esas instrucciones en Python, y comparar con los resultados del punto anterior. Antes hacer que la tortuga apunte hacia arriba con el método t.setheading(90).
- (c) ¿En qué consiste el caso base en esta función recursiva?
- (d) Ejecutar arbol(100,8). Para acelerar la ejecución, reemplazar la línea 4 por if nivel>0 and altura>1:, así la tortuga no pierde el tiempo dibujando cosas imperceptibles. ¿Cómo cambia la definición del caso base al hacer ese cambio?
- (e) Modificar la función arbol, para que dibuje árboles con ramificaciones triples, en lugar de dobles. Por ejemplo, ver la figura de abajo a la izquierda.
- (f) Randomizar la función arbol, de modo que los ángulos y longitudes sean elegidos al azar de ciertos intervalos (para esto, usar el método random.uniform(a,b)¹). Por ejemplo, ver la figura de abajo en el centro.
- (g) Modificar la función arbol, de modo cambie el color (ej: t.pencolor('brown')) y el grosor (ej: t.pensize(3)) del trazo, para dibujar árboles en colores como el de la figura de abajo a la derecha.







¹https://docs.python.org/es/3/library/random.html