F210 Programación Aplicada a Finanzas

TP 0: Recursividad

1. **Torre de Hanoi**

   Bajo la cúpula que marca el centro del mundo en un gran templo de Benarés (India), hay tres varas. Al principio del mundo, Dios apiló 64 discos de oro puro en una de las varillas. Los discos fueron colocados y numerados en orden creciente de radio, de arriba hacia abajo. Al disco más grande de todos le fue asignado por tanto el número 64 y al más chico el 1.

   Se cree que desde el principio del tiempo los monjes pasan días y noches transfiriendo discos de la vara número 1 a la 3 siguiendo una regla determinada por Dios.

   Se supone que el mundo acabará en cuanto los monjes terminen de mover todos los discos.

   Las reglas para trasladar los discos son sencillas:

   (a) sólo se puede mover un disco a la vez

   (b) un disco no puede colocarse sobre otro de radio menor

   Los monjes registran cada movimiento que hacen mediante la terna (a,b,c) donde a es el disco movido, b la vara de donde se saca el disco y c la vara donde se lo inserta.

   (a) 1. Construya una función recursiva Hanoi(n,i,j) que resuelva el problema de la Torre de Hanoi y devuelva como resultado una lista con todas las transferencias cuando n discos deben ser movidos de la vara i a la vara j. (asumiendo que hay tres varas). Verifique que el algoritmo efectivamente funciona para el caso Hanoi(5,1,3)
   2. Compute la cantidad de transferencias t(n) necesarias para los casos n=2,4,8,16,32,64. Grafique t(n) vs n. ¿Qué tipo de relación puede establecer entre la cantidad de movidas y la cantidad de discos?
   3. Un monje tibetano hizo una cuenta en el piso con un palito y le dio que el numero de movidas que va a encontrar en el punto anterior deberia ser $S_n = 2^n - 1$, siendo n la cantidad de discos. Fíjese si puede demostrar de alguna manera (ya sea por induccion completa o en forma directa) que el monje está en lo cierto.
   4. ¿Asumiendo que cada movida tarda 1 minuto, cuanto tiempo debería pasar según su algoritmo hasta que ocurra el fin del mundo?
   5. Cual sería la mínima cantidad de varas que permite resolver el problema de la torre de Hanoi en la menor cantidad de movimientos posibles? Cuales serían esa cantidad de movimientos y la secuencia?

2. **Secuencia de Fibonacci**

   Leonardo Fibonacci fue un matemático italiano cuyo verdadero nombre era Leonardo da Pisa (Leonardo de Pisa). Leonardo Fibonacci, que significa "Leonardo, hijo de

Bonacci", no inventó la secuencia de números que lleva su nombre. Se utilizó su nombre debido a la popularidad que adquirió la secuencia después de que la mencionara en su libro "Liber Abaci" (libro sobre el ábaco).

La sucesión de Fibonacci $F : \mathbb{N} -> \mathbb{N}$ se define recursivamente a partir de los primeros dos términos:

$$\begin{aligned} F(1) &= 1 \\ F(2) &= 1 \\ F(n) &= F(n-1) + F(n-2) \qquad n > 2, n \in \mathbb{N} \end{aligned}$$

(a) Construya una función recursiva `Fib1(n)` que compute la secuencia de Fibonacci.

(b) Construya una función `Fib2(n)`que compute la secuencia de Fibonacci usando la solución analítica, y construya una tabla de $F(n)$ para $1 \leq n \leq 20$ con ambos métodos. Compare los resultados obtenidos.

$$F(n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Demuestre por inducción completa que, efectivamente, la fórmula arriba presentada es la solución a la secuencia de Fibonacci.

(c) Modifique el siguiente script

```
import time
start = time.perf_counter_ns()
print("hello")
end = timeit.perf_counter_ns()
print(end - start)
```

Y adáptelo para comparar el tiempo que tarda la funcion recursiva en computar los primeros 40 términos de la secuencia de Fibonacci, versus lo que tarda la fórmula explícita.

(d) A fin de mejorar la performance de la forma recursiva pura, vamos a buscar otras estrategias:

1. Usando un loop hacia adelante, que para calcular $F(n)$ inicializa dos valores $F(n-1) = F(n-2) = 1$ y que se va calculando de manera iterativa haciendo $F(n) = F(n-1) + F(n-2)$, y con cada valor de $F(n)$ nuevo calculado se van sustituyendo los dos valores previos $F(n-1) = F(n)$ y $F(n-2) = F(n-1)$ para calcular al siguiente $F(n)$. Llame a esta funcion `Fib3(n)`
2. Usando una forma recursiva, como en $Fib1(n)$ pero con una lista/diccionario de apoyo con los valores ya encontrados de $F(n)$, y que cada vez que se llama a la función, si el valor $F(n)$ ya fue encontrado, que lo tome directamente de la memoria en lugar de recalcular todo de nuevo. Llame a esta funcion `Fib4(n)`

(e) Use nuevamente el script para medir performance y compare el tiempo de cómputo para las 4 versiones de la sucesion de Fibonacci.

3. **Máximo común divisor**

   El MCD de dos enteros positivos a y b es el máximo entero c que divide exactamente a ambos números a y b. Se le pide construir una funcion mcd(a,b) que encuentre el MCD entre a y b. Construirá dos versiones de mcd:

   - Utilizando la descomposicion en factores primos de a y b (mediante una funcion recursiva) y construyendo c como el producto de los factores comunes elevados a su menor exponente.
   - Utilizando el algoritmo de Euclides. Euclides fue un matemático griego que vivió hace aproximadamente 2.300 años. Su algoritmo para calcular el máximo común divisor de dos enteros positivos, a y b, es

     Si b es 0 entonces devuelve a

     Si no establece c igual al resto cuando a se divide por b

     Devuelve el máximo común divisor de b y c

   (a) Construya ambas soluciones. Llamelas mcd1 y mcd2 respectivamente.

   (b) Muestre mediante ejemplos que el algoritmo de Euclides es mucho mas eficiente que el otro, en particular cuando se trata de numeros grandes.

   (c) Muestre matematicamente que las soluciones del algoritmo de Euclides y del otro son coincidentes.

4. **Números romanos**

   Como su nombre indica, los números romanos se desarrollaron en la antigua Roma. Incluso después

   de la caída del Imperio Romano, sus números siguieron utilizándose en Europa hasta finales de la Edad Media hasta la Baja Edad Media, y en la actualidad se siguen utilizando en circunstancias limitadas. Los números romanos se construyen a partir de las letras M, D, C, L, X, V e I que representan 1000, 500, 100, 50, 10, 5 y 1 respectivamente.

   Los números se escriben se escriben de mayor a menor valor. Cuando esto ocurre, el valor del número es la suma de los valores de todos sus numerales. Si un valor menor precede a un valor mayor mayor, el valor menor se resta del valor mayor al que precede inmediatamente, y esa diferencia se suma al valor del número.[1]

   - Cree una función r2i que convierta de manera recursiva un número romano entre 1 y 3999 en un número entero.

[1] Sólo C, X e I se utilizan de forma sustractiva. El número al que precede una C, una X o una I debe tenerun valor que no sea más de 10 veces el valor que se resta. Por lo tanto, I puede preceder a V o X, pero no puede preceder a L, C, D o M. Esto significa, por ejemplo, que 99 debe representarse por XCIX y no por IC.

- Construya una función i2r que convierta de manera recursiva un numero entero entre 1 y 3999 en su contraparte romana.

5. **Algunas series importantes**

   Pruebe por induccion completa la validez de las siguientes expresiones y desarrolle funciones recursivas que las calculen

   - $S_N = \sum_{i=1}^{N} i = \frac{(N+1)\ N}{2}$
   - $S_N = \sum_{i=1}^{N} i^2 = \frac{(2N+1)(N+1)\ N}{6}$. Los curiosos tienen una linda discusion en https://math.stackexcha
     proof-for-the-sum-of-squares/663527#663527.
   - $S_N = \sum_{i=1}^{N} \frac{1}{i(i+1)} = 1 - \frac{1}{N+1}$
   - $S_N = \sum_{i=1}^{N} q^i = \frac{1-q^{N+1}}{1-q}$
   - $S_N = \sum_{i=1}^{N} \frac{1}{4i^2-1} = \frac{N}{2N+1}$

6. **Secuencias de Elementos Químicos**

   A algunas personas les gusta jugar a un juego que construye una secuencia de elementos químicos donde cada elemento de la secuencia comienza con la última letra de su predecesor. Por ejemplo, si una secuencia comienza con Hidrógeno, entonces el siguiente elemento debe ser un elemento que empiece por N, como el Níquel. El elemento siguiente al Níquel debe comenzar

   con L, como el Litio. La secuencia de elementos construida no puede contener duplicados. Cuando se juega solo, el objetivo del juego es construir la secuencia de elementos más larga posible.Cuando se juega con dos jugadores, el objetivo es seleccionar

   un elemento que deje a tu oponente sin opción de añadirlo a la secuencia. Utilice la lista de elementos quimicos en ingles.

   Escribe un programa que lea el nombre de un elemento ingresado por el usuario y utilice una función recursiva para encontrar la secuencia más larga posible.

   Muestre la secuencia una vez calculada. Asegúrese de que su programa responda de forma razonable si el usuario no introduce un nombre de elemento válido.

7. **Raíz Cuadrada Recursiva**

   Crea una función de raíz cuadrada con dos parámetros. El primer parámetro, n, será el número para el que se calcula la raíz cuadrada. El segundo parámetro, guess, será la estimación actual de la raíz cuadrada. El parámetro guess debe tener un valor por defecto de 1.0. No proporcione un valor por defecto para el primer parámetro.

Su función de raíz cuadrada será recursiva. El caso base ocurre cuando $guess^2$ está dentro de 10^{-12} de n. En este caso su función debe devolver guess porque está lo suficientemente cerca a la raíz cuadrada de n. En caso contrario, la función devolverá el resultado de llamarse a sí misma recursivamente con n como primer parámetro y $\frac{guess+\frac{n}{guess}}{2}$ como segundo parámetro.

Escribe un programa principal que demuestre tu función raíz cuadrada calculando la raíz cuadrada de varios valores diferentes. Cuando llames a tu función raíz cuadrada desde el programa principal sólo debes pasarle un parámetro para que se use el valor por defecto.