Análisis de Datos y Machine Learning, Tarea 1

Carlos Malanche

6 de febrero de 2018

Para la primer tarea (con fecha de entrega el **Jueves 15 de Febrero**), van a tener que implementar códigos que cumplan con las siguientes condiciones:

1. fibo.py: Serie de Fibonacci

Escriba una función que calcule la suma de los elementos impares de la serie de fibonacci hasta el primer elemento superior a n. Por ejemplo, $f(10) = 1 + 1 + 3 + 5 + 13 = 23$. El código debe imprimir el resultado de $f(100000)$.

2. maspalindromos.py: *Turbopalíndromos*

Sea x un número de n dígitos, $\{x_0, x_1, \ldots, x_{n-1}\}$, y sea r la cantinidad definida como:

$$r = x + \sum_{i=0}^{n-1} (-1)^i * rot(x, x_i) \tag{1}$$

donde $rot(x, \alpha)$ es la rotación de los dígitos de x en α posiciones. Entonces x será llamado un *turbopalíndromo* si r es un palíndromo (puede ser un número negativo).

Encuentre la cantidad de *turbopalíndromos* entre 1 y 200,000.

Ejemplo: 2639 es un turbopalíndromo, pues:

$$r = \underbrace{2639}_{x} + \underbrace{3926}_{rot(x,2)} - \underbrace{3926}_{rot(x,6)} + \underbrace{6392}_{rot(x,3)} - \underbrace{9263}_{rot(x,9)} = -232$$

3. Un algoritmo de *Sorting*

Un algoritmo de sorting (acomodo) es un algoritmo que toma una secuencia finita S de N elementos $s_i \in \mathbb{U}$ y genera una permutación de los índices $\sigma : \mathbb{N} \rightarrow \mathbb{N}$ (es decir, un nuevo arreglo de misma cardinalidad pero nuevo orden) tal que la secuencia resultado $\dot{S} = \{s_{\sigma(i)}\}_{i=1}^{N}$ cumple con $(\dot{s}_i \leq \dot{s}_{i+1}) = 1, \forall i = 1, \ldots, N-1$ (donde la operación $\leq: \mathbb{U} \times \mathbb{U} \rightarrow 0, 1$ está definida).

Estos algoritmos son de suma importancia pues su uso es extendido. Muchas veces en este curso vamos a tener que usar funciones de acomodo cuya implementación ya se encuentra en varios paquetes de `Python`, pero su estudio nos dará ventajas en un futuro.

Intente, sin utilizar internet, escribir un algoritmo de *sorting* en `Python` que reciba una secuencia de *naturales* (es decir, $\mathbb{U} = \mathbb{N}$ y $\leq$ un simple *menor o igual que*) separados por comas y devuelva la secuencia ordenada. En internet hay muchas soluciones, pero lo importante es que intenten hacer su propio algoritmo (ya después pueden verificar qué tan eficiente es el algoritmo que hicieron).

Para facilitar la tarea, el script `sortingavanzado9000.py` contiene unas funciones útiles para generar los arreglos e imprimirlos en pantalla.

Ejemplo:

$$mySorting([2, 7, 3, 8, 2, 9]) = [2, 2, 3, 7, 8, 9]$$