



Facultad de Ciencia

Departamento de Matemática y Ciencia de la Computación

Paradigmas de programación

Semestre 2020-01

Ejercicios de ayudantía

Nicolás Honorato Droguett
Matías Hurtado Carrasco

Pablo Perez Lantero
22 de mayo de 2020

Índice

1	2020-05-11	2
1.1	Ayudantía Nicolás	2
	Ejercicios	2
1.2	Ayudantía Matías	2
	Ejercicios	2
2	2020-05-18	3
2.1	Ayudantía Nicolás	3
	Ejercicios	3
2.2	Ayudantía Matías	3
	Ejercicios	3

1 2020-05-11

1.1 Ayudantía Nicolás

Ejercicios

Pregunta 1 Programe en Python la función `bloques(g, key)` donde `g` es un generador de elementos, y `key` es una función de un argumento que recibe elementos de `g` y devuelve valores comparables entre sí usando los operadores habituales de comparación. La función genera los bloques de elementos consecutivos generados por `g` que están ordenados ascendentemente de acuerdo al valor de `key`. Cada vez que se rompe el orden se genera un nuevo bloque. Cada bloque es generado como una lista. Por ejemplo, si `g` genera los elementos 1,5,6,4,8,9,2,3,7 en este orden y `key` es la función identidad, entonces se generarán los bloques [1,5,6], [4,8,9], [2,3,7] en este mismo orden. Tenga presente en la implementación que `g` podría generar infinitos elementos, o una cantidad finita pero extremadamente grande.

Pregunta 2 La función `merge(A,B)` recibe dos listas A y B de números, cada una ordenada ascendentemente, y devuelve una tercera lista con la mezcla ordenada de A y B. Usando la función `merge`, realice dos implementaciones distintas de la función `mega_merge(*L)` que recibe varias listas de números como parámetros, cada lista ordenada, y devuelve la mezcla de todas las listas. La primera implementación debe usar la función `reduce`, mientras que la segunda implementación no puede usarla.

1.2 Ayudantía Matías

Ejercicios

Pregunta 1 Implemente en Python la función `merge(A,B,comp=cmp)`, pero en lugar de devolver explícitamente la mezcla en una lista, devuelve una generación de (o define una forma de generar o iterar) los elementos de la mezcla.

Pregunta 2 Implemente en Python la función `modulo(A,k)` que recibe una lista A y un entero positivo k, y devuelve una lista compuesta por k listas. La lista número i de la lista resultado contiene de la lista original los elementos que se encuentran en las posiciones que dejan resto i en la división por k. Por ejemplo, para `A = [0,1,2,3,4,5,6,7,8,9,10]` y `k = 3`, la función debe devolver la lista `[[0,3,6,9], [1,4,7,10], [2,5,8]]`.

2 2020-05-18

2.1 Ayudantía Nicolás

Ejercicios

Pregunta 1 La **sucesión de Thue-Morse** es una sucesión de números binarios que al concatenarlos se produce una secuencia con segmentos iniciales alternos. Implemente en **Python** una función que entregue los n primeros números de la sucesión, siendo n un **parámetro opcional**. La secuencia comienza con los números 0, 01, 0110, 01101001, \dots , si tomamos t_n como el binario n de la sucesión, el binario t_{n+1} es una **concatenación** de la forma $t_n + C_1(t_n)$, tal que $C_1(x)$ es la función complemento 1.

Pregunta 2 Implemente en **Python** una función que reciba una matriz **M** y entregue la sub-matriz de suma máxima de **M**. Muestre el valor de la suma.

2.2 Ayudantía Matías

Ejercicios

Pregunta 1 Implemente en **Python** un generador **fun_stream(s,e)**, que devuelva el los resultados de la ecuacion $\text{fun}(x) = 2x^3 + 3x^2 + 4$ para todos los naturales entre **s** y **e** incluyente, utilizando el generador **exp_stream(s,e,exp)** que entrega x^{exp} siendo x todos los naturales entre **s** y **e** incluyente.

Pregunta 2 Implemente en **Python** la función **recorrer(M)** donde **M** es una matriz rectangular llena de 1 y 0, la funcion recorrer debe generar un camino desde la posición (0,0) de la matriz **M** hasta la esquina contraria, con los 0 como camino valido y los 1 no, el orden de preferencia de movimiento es $(x,y) \rightarrow (x+1,y+1)$, $(x,y) \rightarrow (x+1,y)$, $(x,y) \rightarrow (x,y+1)$

$$\begin{bmatrix} \$ & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} \$ & \$ & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} \$ & \$ & 0 & 0 & 0 & 0 \\ 1 & 1 & \$ & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \xrightarrow{*} \begin{bmatrix} \$ & \$ & 0 & 0 & 0 & 0 \\ 1 & 1 & \$ & \$ & 0 & 0 \\ 0 & 0 & 0 & 1 & \$ & 0 \\ 0 & 0 & 0 & 1 & 0 & \$ \\ 0 & 0 & 0 & 1 & 0 & \$ \\ 0 & 0 & 0 & 1 & 0 & \$ \end{bmatrix}$$