**ESTRUCTURAS DE DATOS Y ALGORITMOS**

**ACTIVIDAD EN AULA 7 JUNIO 2019: RECURSIÓN**

**TRABAJO EN DUPLAS**

**PLAZO: Viernes 7 Junio, 12.40 hrs vía webcursos**

Mira el primer video de la semana y luego rutea las siguientes funciones recursivas (con el mismo nivel de detalle mostrado en el video) para descubrir qué tarea cumplen.

**DESAFÍO 1**

**FUNCIÓN 1**

|  |
| --- |
| 1.def F1(a,b):  2. if (b==0):  3. return 1  4. x=a\*F1(a,b-1)  5. return x |

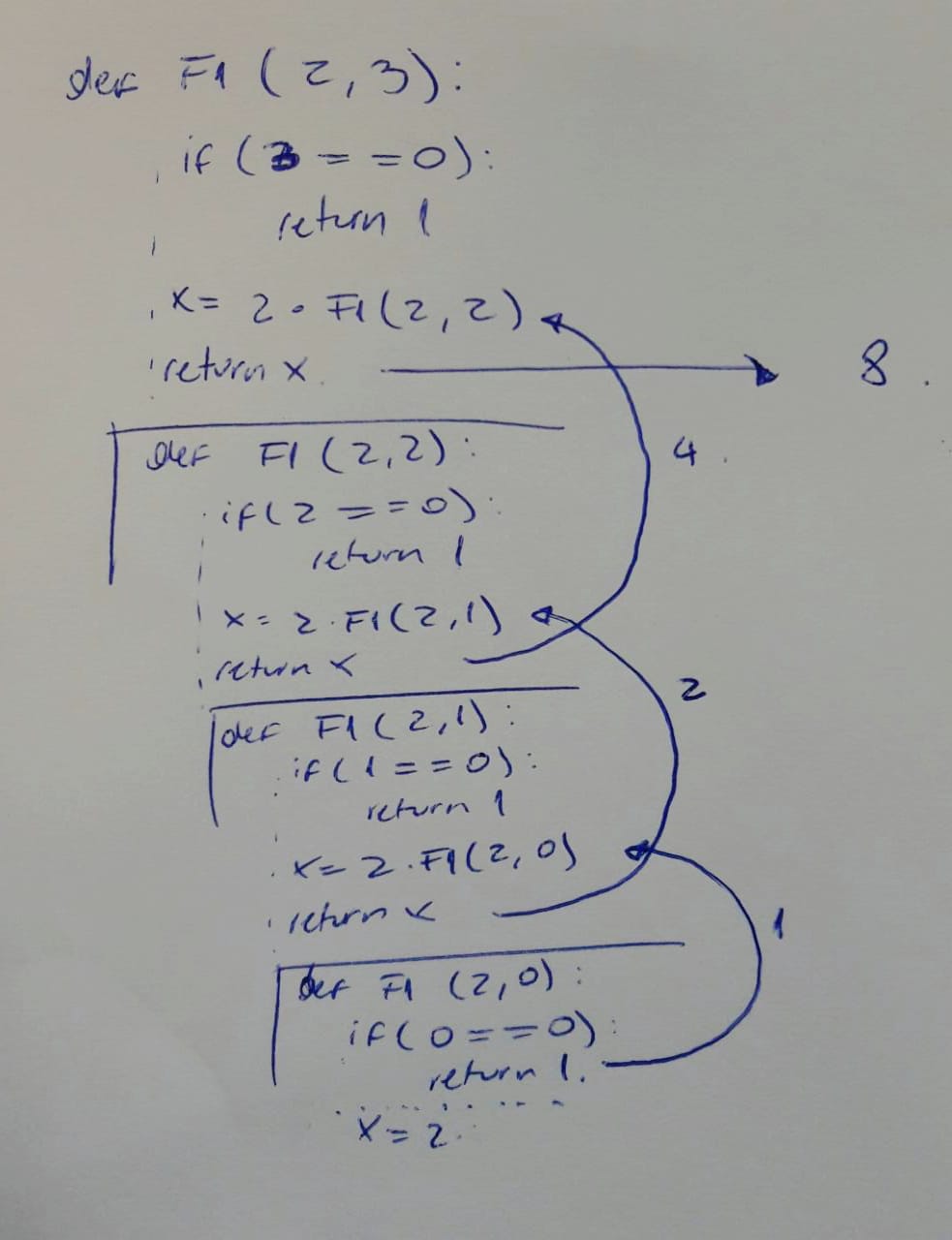
1. ¿qué líneas constituyen el caso base?

Lineas 2 y 3 consituyen un caso base, ya que es ahí donde se verifica que b se va acercando a 0 a medida de que avanza la recursión, y finalmente terminan la recursión.

1. ¿qué líneas corresponden a la parte recursiva?

Linea 4 y 5 corresponden a la parte recursiva, en donde la linea 4 hace el llamado de función y se espera para almacenar en x el valor de retorno, y luego retornarlo a su vez en la línea 5.

1. Inserta aquí una foto del ruteo, hecho como se muestra en el video (o hazlo en computador y lo copias aquí):



1. ¿Qué hace esta función recursiva?

Lo que hace es elevar *a* a *b* (*a*^*b*).

**DESAFÍO 2**

Escribe el código de una función recursiva (en Python o en C) que:

1. calcule la resta de dos números
2. calcule la multiplicación de dos números
3. calcule el módulo de dos números

Inserta los códigos a continuación:

**RESTA RECURSIVA**

|  |
| --- |
| def resta(a,b):  if (b == 0):  return a  if (b < 0):  return resta(a+1,b+1)  if (b > 0):  return resta(a-1,b-1) |

**MULTIPLICACIÓN RECURSIVA**

|  |
| --- |
| def mult(a,b):  if (b==0):  return 0  x=a + mult(a,b-1)  return x |

**MÓDULO RECURSIVO**

|  |
| --- |
| def mod(a,b):  if (a < b):  return a  x = mod(a-b,b)  return xl |

**DESAFÍO 3**

Considera la siguiente función recursiva, que recibe como argumento de entrada un arreglo de números A, el índice del primer elemento del arreglo (low) y el índice del último elemento del arreglo (high).

|  |
| --- |
| **function** X(A,low,high)  **if** low < high **then**  p ← **particion**(A,low,high)  X(A,low,p-1)  X(A,p+1,high)  **end if**  **end function** |

La función partition, no es recursiva, y se define como sigue:

|  |
| --- |
| **function** particion(A,low,high)  pivot ← A[high]  i ← low  **for** i ≤ j < high **do**  **if** A[j] ≤ pivot **then**  swap(A[i],A[j])  i ← i + 1  **end if**  **end for**  swap(A[high],A[i])  **return** i  **end function** |

1. Rutea la función particion() para un arreglo cualquiera de 5 números. Describe el ruteo a continuación.

Entra a función partición con valores index low 0: 4; index high 4: 3

---

[4, 7, 2, 1, 3]; index low 0: 4; index high 4: 3

[2, 7, 4, 1, 3]; index low 0: 2; index high 4: 3

[2, 1, 4, 7, 3]; index low 0: 2; index high 4: 3

Swap entre A[4] = 3 y A[2] = 4

[2, 1, 3, 7, 4]; index low 0: 2; index high 4: 4

---

retorna 2

1. ¿Qué hace la función particion()?

Toma como pivote el último valor del arreglo, y sitúa a los valores menores a ese a la izquierda y los mayores a la derecha.

1. Rutea la función X (como se mostró en el video) para un arreglo cualquiera de 7 números.

Entra a función X

.....

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 6: 9

Entra a función partición con valores index low 0: 4; index high 6: 9

---

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 6: 9

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 6: 9

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 6: 9

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 6: 9

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 6: 9

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 6: 9

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 6: 9

Swap entre A[6] = 9 y A[6] = 9

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 6: 9

---

retorna 6

Entra a función X

.....

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 5: 8

Entra a función partición con valores index low 0: 4; index high 5: 8

---

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 5: 8

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 5: 8

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 5: 8

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 5: 8

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 5: 8

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 5: 8

Swap entre A[5] = 8 y A[5] = 8

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 5: 8

---

retorna 5

Entra a función X

.....

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 4: 3

Entra a función partición con valores index low 0: 4; index high 4: 3

---

[4, 7, 2, 1, 3, 8, 9]; index low 0: 4; index high 4: 3

[2, 7, 4, 1, 3, 8, 9]; index low 0: 2; index high 4: 3

[2, 1, 4, 7, 3, 8, 9]; index low 0: 2; index high 4: 3

Swap entre A[4] = 3 y A[2] = 4

[2, 1, 3, 7, 4, 8, 9]; index low 0: 2; index high 4: 4

---

retorna 2

Entra a función X

.....

[2, 1, 3, 7, 4, 8, 9]; index low 0: 2; index high 1: 1

Entra a función partición con valores index low 0: 2; index high 1: 1

---

[2, 1, 3, 7, 4, 8, 9]; index low 0: 2; index high 1: 1

Swap entre A[1] = 1 y A[0] = 2

[1, 2, 3, 7, 4, 8, 9]; index low 0: 1; index high 1: 2

---

retorna 0

Entra a función X

.....

......

Entra a función X

.....

......

......

Entra a función X

.....

[1, 2, 3, 7, 4, 8, 9]; index low 3: 7; index high 4: 4

Entra a función partición con valores index low 3: 7; index high 4: 4

---

[1, 2, 3, 7, 4, 8, 9]; index low 3: 7; index high 4: 4

Swap entre A[4] = 4 y A[3] = 7

[1, 2, 3, 4, 7, 8, 9]; index low 3: 4; index high 4: 7

---

retorna 3

Entra a función X

.....

......

Entra a función X

.....

......

......

......

Entra a función X

.....

......

......

Entra a función X

.....

......

......

1. ¿Qué hace la función X?

Es un algoritmo de ordenamiento que con recursión va dividiendo el arreglo en subarreglos y con la función de partición va ordenando de adentro hacia afuera el arreglo entero.

Suban este archivo completo al enlace habilitado en webcursos. El archivo debe tener el nombre Apellido1\_Apellido2.doc/pdf.