**Tecnología Digital 1: Introducción a la Programación - TP1**

**Autores:** Estrada, Lucas

Rehmann, Lucas

Aulicino, Lucas

**Función distancia\_binaria:**

distancia\_binaria contiene un ciclo “while”. El ciclo termina ya que:

* i vale 0 al igual que “distance”
* len(bin\_n) no cambia su valor
* Ambos aumentan en 1 por iteración mientras se cumpla la condición de i<len(bin\_n)
* Cuando i>=len(bin\_n) entonces el while da falso, termina el ciclo

**INV**: luego de k iteraciones, ‘distance’ representa el número de bits diferentes entre los primeros k bits de ‘bin\_n’ y ‘bin\_m’, i vale k

**Para el testing de esta función elegimos cuatro casos:**

* distancia\_binaria(6,18) = 2, porque demuestra distancias mayores a 1 (“00110” y ”10010”)
* distancia\_binaria(12,12) = 0, porque demuestra que la función puede retornar 0, algo que puede pasar desapercibido (“1100”,”1100”)
* distancia\_binaria(1,3) = 1, porque muestra que podemos tener distancias iguales a 1 (“01”,”11”)
* distancia\_binaria(127,128) = 8, porque aunque la distancia entre estos dos números en decimal es 1, en binario es distinto (“01111111” y “10000000”)

**Función son\_aledaños:**

‘son\_aledaños’ no contiene ciclos “while”. Llamamos a la función **distancia\_binaria** con los parámetros recibidos para calcular la distancia entre estos.

**Para el testing de esta función elegimos seis casos:**

* son\_aledaños(14,3) = False y son\_aledaños(1,100) = False, porque demuestran que para cualquier par de números que no sean aledaños devuelve False, sin importar el orden de ingreso *(es decir, el primero ingresado no tiene que ser mayor al segundo)*.
* son\_aledaños(3,3) = False y son\_aledaños(50,50) = False, porque demuestra el correcto funcionamiento aunque dos números sean iguales.
* son\_aledaños(3,2) = True, y son\_aledaños(5,4) = True, porque demuestran que para cualquier pares de números que sean aledaños devuelve True.

**Función aledaños\_menores:**

‘aledaños\_menores’ contiene un ciclo “while”. El ciclo termina ya que:

* i vale 1
* n no cambia
* i aumenta 1 por iteración hasta que se deje de cumplir i<n
* cuando i es mayor o igual a n, termina el ciclo.

**INV**: Después de k iteraciones, 'total' contiene todos los números menores a ‘n’ que cumplen con la condición 'son\_aledaños(n, i)'. i vale k

Creamos un while que por cada intervalo, mientras i sea menor a n, use la función son\_aledaños() entre n e i para confirmar que sean vecinos. Si se cumple esto, appendear el valor de i en ese intervalo a la lista.

**Para el testing de esta función elegimos cuatro casos:**

* aledaños\_menores(127) = [63, 95, 111, 119, 123, 125, 126] y aledaños\_menores(5) = [1, 4]; porque demuestran que puede retornar una lista con 1 o más valores.
* aledaños\_menores(64) = [], aledaños\_menores(1) = []; porque demuestran que la lista final puede ser vacía. Esto puede pasar de largo si i, la variable usada para realizar el ciclo, es igual a 0 en vez de 1.

**Función cant\_aledaños:**

“cant\_aledaños”contiene un ciclo “while”. El ciclo termina ya que:

* i = a
* b no cambia
* i aumenta 1 por iteración mientras sea menor o igual a b
* Cuando i sea mayor a b, termina el ciclo.

**INV**: Después de k iteraciones: i vale k, cant vale la cantidad de vecinos binarios entre n y los números en el intervalo [a,b]

**Para el testing de esta función elegimos cuatro casos:**

* cant\_aledaños(10,8,13) = 2, cant\_aledaños(10,1,100) = 7; porque demuestran el uso correcto de los parámetros y el resultado correcto de los vecinos que existen en el intervalo.
* cant\_aledaños(64,1,63) = 0, cant\_aledaños(10,100,1) = 0 porque demuestran que existe la posibilidad de que no hayan vecinos en el intervalo [a,b]

**Función densidad\_intervalo:**

“densidad\_intervalo”no contiene ciclos “while”.

Para el testing de esta función elegimos varios casos:

* densidad\_intervalo(64,1,63) = 0.0 y densidad\_intervalo(64,1,63) = 0.0, comprueban de que no haya errores a la hora de buscar un resultado que sea 0.
* densidad\_intervalo(10,8,13) = 0.33333 y densidad\_intervalo(100,50,100) = 0.03922, ya que demuestran efectivamente la densidad de vecinos aledaños a en en el intervalo ingresado, y a la vez muestra la compresión correcta de la densidad usando round().