Tecnología Digital 1: Introducción a la Programación
Trabajo Práctico 1

Escuela de Negocios, UTDT

Segundo semestre 2022

Las computadoras no entienden las palabras o los números como lo hacen las personas. El *software* moderno nos permite no prestar atención a esto, pero en los niveles más bajos los componentes físicos de las computadoras representan todo mediante señales que distinguen dos estados posibles: presencia o ausencia de voltaje. Habitualmente, para referirnos a estos estados se utilizan los dígitos 0 y 1. Su uso se ha vuelto tan habitual que a partir de ellos se inventó la palabra *bit*, formada por las dos primeras letras y la última letra de dígito binario en inglés, *binary digit*. Debido a su utilidad práctica, la codificación de números en base 2 ha sido ampliamente estudiada.

Se dice que un número $n \in \mathbb{N}$ (es decir, un entero mayor que 0) es *binario balanceado* si en su codificación en base 2 (sin ceros no significativos a la izquierda) hay la misma cantidad de ceros que de unos. El menor binario balanceado es el 2 (10 en binario). También es binario balanceado el 11921: su desarrollo en base 2 es 10111010010001. Por su parte, el 54 **no** lo es, porque en base 2 es escribe 110110.

El objetivo de este trabajo es crear un programa en Python que ofrezca las siguientes funcionalidades:

- dado $n \in \mathbb{N}$, obtener un *string* con su desarrollo binario;
- dado $n \in \mathbb{N}$, determinar si n es binario balanceado;
- dados $n, m \in \mathbb{N}$ con $n \leq m$, obtener la cantidad de binarios balanceados mayores o iguales que n y menores o iguales que m.
- dado $n \in \mathbb{N}$, obtener el menor binario balanceado estrictamente mayor que n;
- dado $n \in \mathbb{N}$ y $n \geq 3$, obtener el mayor binario balanceado estrictamente menor que n;
- dado $n \in \mathbb{N}$, obtener el binario balanceado más cercano a n, teniendo en cuenta que:

  (I) si n es binario balanceado, se espera obtener n;

  (II) en caso de que hayan dos más cercanos (uno menor que n y otro mayor que n a la misma distancia), se espera obtener el mayor de ellos.

Para cada una de estas funcionalidades requeridas, se pide:

(a) Escribir la especificación de una función que resuelva el problema.

(b) Construir un conjunto de casos de test que se considere adecuado para ilustrar y poner a prueba el comportamiento esperado.

(c) Pensar un algoritmo y llevarlo a un programa en Python que resuelva el problema.

(d) Asegurarse de que la función pasa los casos de test construidos.

Además, se pide demostrar que son correctas (i) la función que determina si un natural n es binario balanceado; y (ii) la función que dado un natural n obtiene el menor binario balanceado estrictamente mayor que n. En particular, se espera la demostración de que los ciclos terminan y son correctos.

Al ejecutar el programa se deberá desplegar un menú desde el cual seleccionar la funcionalidad a la que se desea acceder. Dependiendo de la opción elegida y los argumentos ingresados, el programa tendrá que mostrar la respuesta que corresponde. A continuación puede observarse parte de una ejecución del programa en la cual el usuario consulta si el número 11921 es o no binario balanceado:

```
Funciones disponibles
---------------------
A. Desarrollo binario [n]
B. Es binario balanceado [n]
C. Cantidad de binarios balanceados entre [n,m]
D. Siguiente binario balanceado [n]
E. Anterior binario balanceado [n]
F. Binario balanceado más cercano [n]
G. Finalizar

Seleccione una opciónn: B
Ingrese n: 11921
El 11921 es binario balanceado.

Presione ENTER para continuar.
```

La ejecución debe concluir únicamente si se selecciona la opción `Finalizar`; caso contrario, tiene que desplegar nuevamente el menú de opciones.

En la siguiente tabla se ilustran los mensajes que debe imprimir el programa para algunas funciones y valores ingresados:

Función seleccionada	Argumento(s)	Mensaje por pantalla
`Desarrollo binario [n]`	`3`	`El 3 en binario es 11.`
`Desarrollo binario [n]`	`11921`	`El 11921 en binario es 10111010010001.`
`Es binario balanceado [n]`	`3`	`El 3 no es binario balanceado.`
`Es binario balanceado [n]`	`11921`	`El 11921 es binario balanceado.`
`Cantidad binarios balanceados entre [n,m]`	`2,10`	`Hay 3 binarios balanceados entre 2 y 10.`
`Siguiente binario balanceado [n]`	`2`	`El siguiente binario balanceado de 2 es 9.`
`Anterior binario balanceado [n]`	`9`	`El anterior binario balanceado de 9 es 2.`
`Binario balanceado más cercano [n]`	`5`	`El binario balanceado más cercano a 5 es 2.`
`Binario balanceado más cercano [n]`	`6`	`El binario balanceado más cercano a 6 es 9.`

Se deben entregar los siguientes cuatro archivos:

- `binario_balanceado.py`, con el código de las funciones: `decimal_a_binario`, `es_binario_balanceado`, `cantidad_binarios_balanceados_entre`, `siguiente_binario_balanceado`, `anterior_binario_balanceado`, `binario_balanceado_mas_cercano` y cualquier otra función auxiliar que se defina. Es obligatorio especificar el tipo de todas las variables y funciones mediante sugerencias de tipos (*type hints*). Las especificaciones de las funciones se deben incluir con el formato de *docstrings* presentado en clase. Este archivo **sólo** debe incluir la implementación de las funciones; es decir, **no** debe ser directamente ejecutable.
- `binario_balanceado_test.py`, con los tests de unidad de todas las funciones definidas.
- `tp1.py`, con el código principal para ejecutar el programa. Este código es el encargado de imprimir el menú, invocar a las funciones definidas en `binario_balanceado.py` y mostrar los resultados por pantalla.
- `informe.pdf`, un documento con (i) la justificación de los casos de test elegidos (es decir, por qué son buenos candidatos para revelar la presencia de errores); y (ii) las demostraciones de correctitud

solicitadas más arriba. Además debe incluir cualquier aclaración adicional que se considere necesaria sobre cualquier parte del trabajo. Se espera que este documento sea conciso, de no más de cuatro páginas, y en formato PDF.

La carpeta adjunta `templates` contiene archivos de ejemplo para usar como referencia.

Observaciones:

- El trabajo se debe realizar en grupos de **tres personas**. La entrega consistirá en trabajo original realizado íntegra y exclusivamente por las personas que integran el grupo.
- La fecha límite de entrega es el **martes 11 de octubre a las 23:55**. Los TPs entregados fuera de término serán aceptados hasta 48h pasada esta fecha, pero la demora incidirá negativamente en la nota.
- Los archivos deben subirse al formulario *TP1: Entrega* de la página de la materia en el campus virtual.
- El programa entregado debe correr correctamente en `Python3`.
- Solo está permitido importar la biblioteca `unittest` (para correr los tests definidos en `binario_balanceado_test.py`).
- La función `print` solamente puede invocarse en el cuerpo principal del código, en el archivo `tp1.py`. Es decir, no puede usarse en las funciones requeridas ni en las funciones auxiliares.
- Sólo pueden usarse las instrucciones y estructuras de control vistas en clase previo al primer parcial.
- Puede suponerse que el usuario siempre invocará a las funciones de manera correcta. Es decir, si hay errores de tipo o valor en los argumentos provistos, no se espera comportamiento alguno del programa (podría colgarse o terminar en un error, por ejemplo).