**Casos de test**

**String\_inverso**

|  |  |  |
| --- | --- | --- |
| Argumento | Salida esperada | Justificación de elección |
| ‘’ | ‘’ | String vacío. |
| ‘hola’ | ‘aloh’ | String solamente con caracteres del abecedario. |
| ‘h2! Ñ;‘ | ‘;Ñ !2h’ | String compuesto por caracteres de todo tipo. |
| ‘1000’ | ‘0001’ | String compuesto por números. |

**Decimal\_a\_binario**

|  |  |  |
| --- | --- | --- |
| Argumento | Salida esperada | Justificación de elección |
| 1 | ‘1’ | Primer número natural.  Números naturales no balanceados. |
| 8 | ‘1000’ |
| 13 | ‘1101’ |
| 2 | ‘10’ | Números naturales balanceados. |
| 9 | ‘1001’ |
| 35 | ‘100011’ |

**Es\_binario\_balanceado**

|  |  |  |
| --- | --- | --- |
| Argumento | Salida esperada | Justificación de elección |
| 2 | True | Primer natural balanceado. |
| 9 | True | Naturales balanceados. |
| 50 | True |
| 1 | False | Primer natural no balanceado. |
| 5 | False | Naturales no balanceados. |
| 84 | False |

**Cantidad\_binarios\_balanceados\_entre**

|  |  |  |
| --- | --- | --- |
| Argumento | Salida esperada | Justificación de elección |
| (9,9) | 1 | Ambos parámetros son iguales y balanceados. |
| (11,11) | 0 | Ambos parámetros son iguales, pero no balanceados. |
| (2,10) | 3 | Ambos parámetros son balanceados y tienen balanceado entre sí. |
| (1,3) | 1 | Ambos parámetros no son balanceados, pero tienen uno entre sí. |
| (2,8) | 1 | Un parámetro es balanceado, pero el otro no y no existen balanceados entre sí. |
| (41,60) | 7 | Intervalo grande entre parámetros, con balanceados entre sí. |
| (3,8) | 0 | Ambos parámetros no son balanceados y no contienen uno entre sí. |
| (15,30) | 0 |

**Siguiente\_binario\_balanceado**

|  |  |  |
| --- | --- | --- |
| Argumento | Salida esperada | Justificación de elección |
| 1 | 2 | Primer natural no balanceado. |
| 2 | 9 | Número natural balanceado. |
| 3 | 9 | Números naturales no balanceados. |
| 27 | 35 |

**Anterior\_binario\_balanceado**

|  |  |  |
| --- | --- | --- |
| Argumento | Salida esperada | Justificación de elección |
| 3 | 2 | Primer número natural elegible. |
| 5 | 2 | Natural no balanceado. |
| 10 | 9 | Número natural balanceado. |

**Binario\_balanceado\_más\_cercano**

|  |  |  |
| --- | --- | --- |
| Argumento | Salida esperada | Justificación de elección |
| 1 | 2 | Único valor natural elegible, cuyo valor de retorno será estrictamente mayor. |
| 2 | 2 | Naturales balanceados, por lo que esperan como valor de retorno a sí mismos. |
| 50 | 50 |
| 3 | 2 | Natural no balanceado, que se encuentra entre dos valores balanceados. |
| 20 | 12 | Naturales no balanceados, que se encuentran entre dos valores balanceados (12 y 35). |
| 28 | 35 |
| 51 | 52 | Natural no balanceado, que se encuentra entre dos valores a la misma distancia de cada uno (espera recibir el mayor de ellos). |

**Terminación y correctitud**

1. Es\_binario\_balanceado:

**Terminación**

* La variable i comienza valiendo 0 (cero).
* En cada iteración i se incrementa en 1.
* bina es una variable tipo string, que es resultado de la conversión de n a binario y no se modifica en ningún momento luego de ser definida. Además, por el requiere sabemos que n>0. Por lo tanto, su conversión a binario siempre tendrá necesariamente un dígito o más. Como resultado, la longitud de bina (len(bina)) siempre será >0.
* Por lo tanto, es inevitable que en algún momento i llegue al valor de la longitud de bina (len(bina)).
* En ese momento, la condición i < len(bina) es falsa, por lo que el ciclo termina.

**Correctitud**

Predicado invariante:

* 0 <= i <= len(bina)
* Ceros es la cantidad de ceros (0’s) en las primeras i-posiciones de bina (n, en su expresión binaria).
* Unos es la cantidad de unos (1’s) en las primeras i- posiciones de bina (n, en su expresión binaria).

#A

* 0 <= i <= len(bina) es correcto porque i = 0.
* Ambas variables ceros como unos, representan la cantidad de tales números dentro de las primeras i-posiciones recorridas. Como i = 0, es correcto que valgan 0 al no haber recorrido ninguna posición todavía.

#B -> #C

* Entramos al ciclo por lo que la guarda se cumple, es decir, i < len(bina). Así, 0 <= i <= len(bina) sigue siendo válido.
* Llamando Φ al valor de i en #B, es verdadero que Φ < len(bina). Como entre #B y #C i se incrementa en uno, vale Φ + 1. Por eso, i puede seguir siendo menor a len(bina) o pasar a ser igual a len(bina), por lo que 0 <= i <= len(bina) se cumple.
* Si el carácter analizado en la posición i es igual a ‘1’, unos se incrementa en uno. En el caso contrario tendrá el valor de ‘0’, ya que bina únicamente está compuestos por unos y ceros, por el devuelve de la función decimal\_a\_binario. En ese caso ceros aumenta en 1. Por tal razón, unos y ceros siguen representando la cantidad de sus números respectivos en las primeras i-posiciones de bina.

#D

* Podemos llegar a #D desde el punto #A, evaluando la guarda y que ésta resulte falsa o partiendo desde #C, evaluando la guarda y que ésta también resulte falsa. En ambos casos, el invariante sigue siendo válido, ya que tanto entre #A y #D como entre #C y #D no se modifican los valores de i, ceros y unos.
* Cuando la guarda es falsa, i >= len(bina). Por el invariante sabemos que i <= len(bina). Así, necesariamente i = len(bina).
* Por el invariante sabemos que ceros y unos valen la cantidad de 0’s y 1’s respectivamente en las primeras i-posiciones, es decir, en las primeras len(bina) posiciones, es decir, en todo bina. Por eso, en la última línea de la función el valor de retorno, que es unos == ceros, será True si la cantidad de unos y ceros en toda la función es la misma y False, si es diferente (que es lo que especifica el Devuelve). Así, podemos afirmar que la función hace lo especificado.

1. Siguiente\_binario\_balanceado:

**Terminación**

* La variable i comienza valiendo n.
* La variable encontrado comienza valiendo False.
* En cada iteración i se incrementa en 1 (uno).
* Por lo tanto, en la condición if(es\_binario\_balanceado(i)) voy a analizar los números naturales > n, aumentando de uno en uno.
* Como existen infinitos balanceados, i al ir aumentando en una unidad, en algún momento será balanceado estrictamente mayor a n, sin importar de qué número iniciemos.
* Por lo anterior, inevitablemente la condición (if es\_binario\_balanceado(i)) será verdadera en algún momento y allí encontrado tomará el valor de True.
* En ese momento, la guarda es falsa y el ciclo termina.

**Correctitud**

Predicado invariante:

* n <= i
* encontrado es True si i es un número balanceado y todos los números entre n no incluido e i – 1 inclusive no son balanceados.

#A

* El invariante n <= i, se cumple, ya que i comienza valiendo n.
* Como n y i – 1 (que es n – 1) no hay ningún número natural, se cumple trivialmente que no hay balanceado entre ellos, por lo que la segunda parte del invariante es valida.

#B -> #C

* Suponemos que en #B el invariante es válido.
* Llamando Φ al valor de i en #B, en cada iteración del ciclo se le suma 1 a i, por lo que en #C i vale de Φ + 1.
* En cada iteración encontrado toma el valor de es\_binario\_balanceado(i), es decir, se vuelve True si Φ + 1 es balanceado, y permanece False si no lo es.
* Como existen infinitos balanceados mayores a n (sea cual sea n), al i irse incrementando de uno en uno, eventualmente será un balanceado.
* En esa iteración (la llamaremos iteración original) encontrado toma el valor True. Siendo que iteración original sea la primera iteración del ciclo o cualquier iteración posterior, se puede afirmar que Φ no es balanceado. De ser así, encontrado se hubiera vuelto True en la iteración anterior a la original y el ciclo hubiera terminado (por lo explicado en la terminación), nunca llegando así a la iteración original.
* Así, podemos afirmar que i > n, y que encontrado es True si i es binario balanceado y todos los números entre n e i – 1 no son balanceados, por lo que el invariante sigue siendo válido.

#D

* Se puede decir que a #D se llega únicamente desde #C y como entre #C y #D no hubo modificación a i, n < i.
* Según lo explicado en #C, es verdadero que cunado encontrado es True, i es el menor balanceado estrictamente mayor a n. Como el valor de retorno de la función es i, la función cumple con lo especificado.