**DOCUMENTO TÉCNICO**

**Requisitos**

**Especificación**

Siendo la matriz de Hadamard una matriz de tamaño n x n (cuadrada), ese valor n debe ser potencia de 2 (2n) y además los elementos que lo contienen son valores booleanos, representados con [T] y [F].

Esta matriz cumple con una propiedad: los elementos de cada dos filas adyacentes difieren exactamente en n/2 elementos.

Por ejemplo, con n=1; la matriz Hadamard es un arreglo con un único elemento True: H1 = [T].

Entonces lo que hay que determinar es si una matriz es Hadamard o no de acuerdo a la información que se tenga.

**Entrada**

En la primera línea se encuentra el número n donde n >= 1, el cual indica el tamaño de la matriz (que debe ser cuadrada). En la siguiente línea encuentran n x n valores booleanos (T, F) que corresponden a la matriz a validar separados por espacio, el llenado de la matriz (específicamente por filas) completa la información necesaria para tener en cuenta para la salida.

**Salida**

Se debe indicar si la información leída cumple o no a una matriz de Hadamard, o si es imposible hacer un matriz con esos elementos, ya que, el número n no es potencia de 2.

**Diseño**

**Estrategia**

Teniendo una matriz con tamaño n >= 1, se crea una matriz de n x n llena de valores False (porque los elementos de la matriz son valores booleanos), también hubiera podido ser un número, pero por comodidad se puso False. Después se llena la matriz con [T] Y [F] y para verificar si la matriz es o no Hadamard, o es imposible crearla; compara cada par de filas y verifica si la cantidad de posiciones en las que tienen el mismo valor es igual a la mitad del tamaño de la matriz (n/2), es decir, aquí es donde se aplica la estrategia de “dividir y conquistar”, ya que el problema se divide en subproblemas más pequeños (comparar pares de filas) y las soluciones a estos subproblemas se combinan para resolver el problema original (verificar si la matriz es una matriz de Hadamard).

Se utilizaron diferentes estructuras de datos como:

* Listas: se utilizan para almacenar múltiples elementos en una sola variable. En el código, las listas se utilizan en varias partes:
* En la función crear\_matriz(n), se crea una matriz como una lista de listas, donde cada sublista representa una fila de la matriz.
* En la función llenar\_matriz (matriz, n), se llena la matriz (que es una lista de listas) con valores booleanos ingresados por el usuario.
* Variables booleanas: pueden tener dos valores posibles, True o False. En el código, las variables booleanas se utilizan para representar los valores en la matriz.

**Invariante #1 (ver código)**

Por cada caso se mira si la matriz es Hadamard o no es; o si es imposible.

* Iniciación: No hay ningún caso.
* Estabilidad: Por cada iteración dentro del rango establecido debe haber un caso, dependiendo también de los elementos.
* Terminación: Hubo tantos casos como el rango definido.

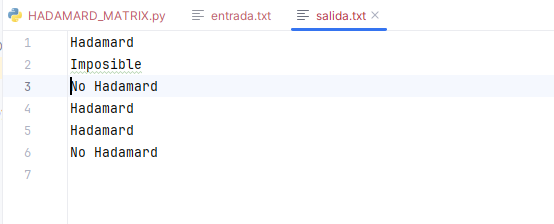
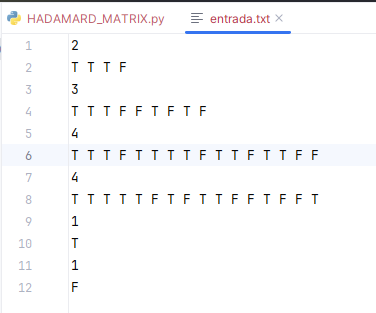
**Invariante #2 (ver código)**

La comparación entre los elementos de la matriz existirá sii tam > 0.

* Iniciación: Hay un caso base donde la matriz es de tamaño 1, entonces debe retornar la posición [0][0] de la matriz.
* Estabilidad: Por cada posición de la matriz dentro del bucle, esta se va comparando con los demás si cumple la condición: los elementos de cada dos filas adyacentes deben tener la mitad del tamaño de la matriz.
* Terminación: Ya recorrió toda la matriz, y terminó de comparar todos los elementos, así muestra la característica de la matriz (es, no es o imposible).

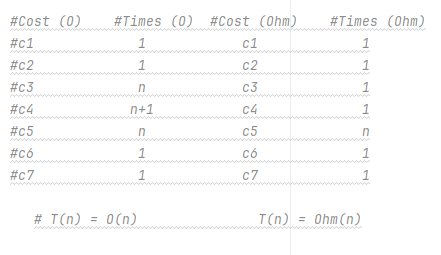
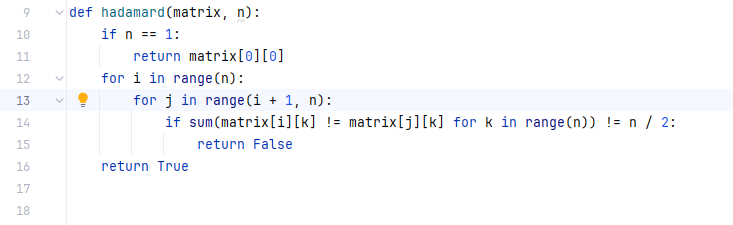
**Casos de prueba**

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Justificación** | **Salida** |
| 2  T T T F | n=2 | Hadamard |
| 3  T T T F F T F T F | n = 3 | Impossible |
| 4  T T T F T T T T F T T F T T F F | n = 4 | No Hadamard |
| 4  T T T T T F T F T T F F T F F T | n = 4 | Hadamard |
| 1  T | n = 1 | Hadamard |
| 1  F | n=1 | No Hadamard |

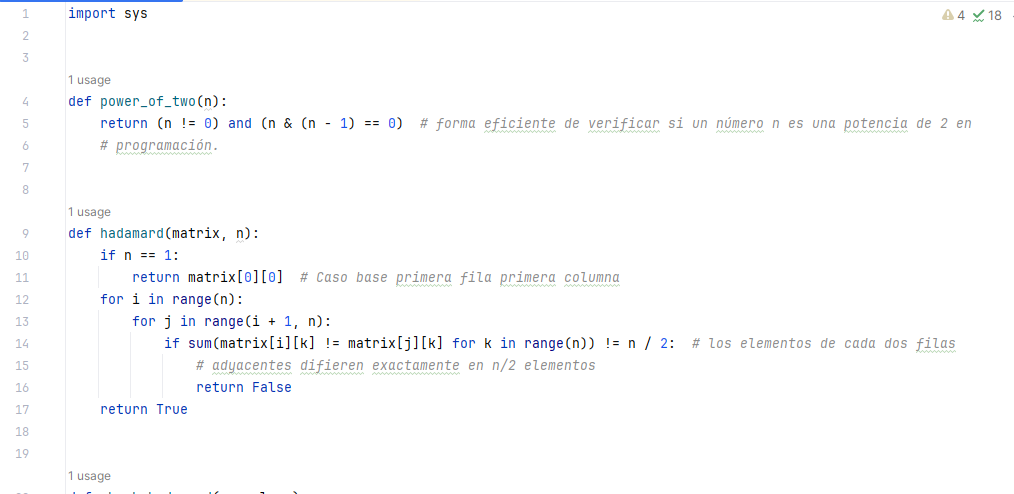
****

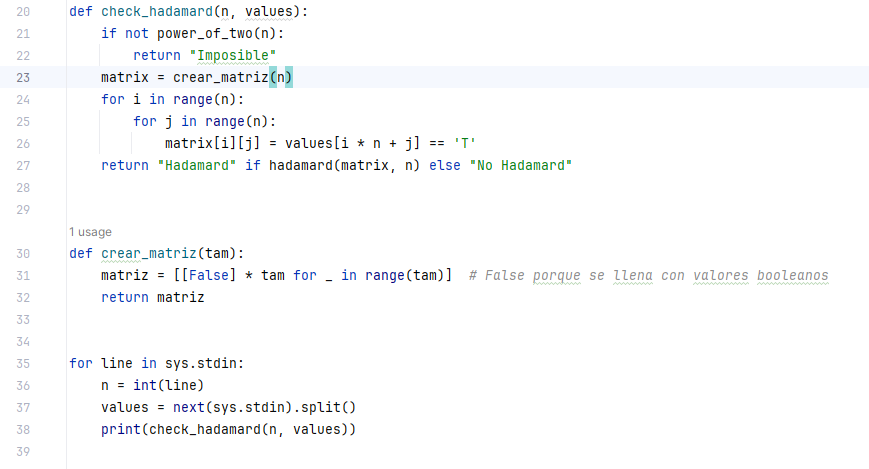
**Análisis**

**Temporal**

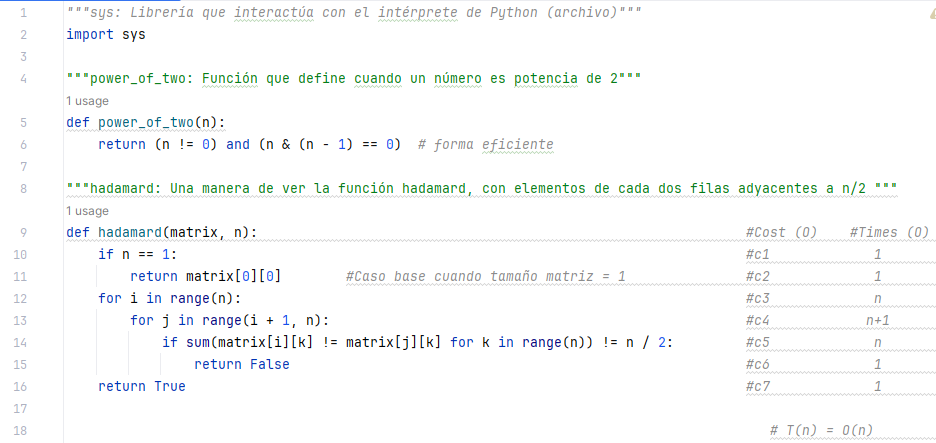
Analizando la función que resuelve la propiedad de la matriz Hadamard, se tomó la complejidad usando análisis asintótico para el peor y mejor de los casos:

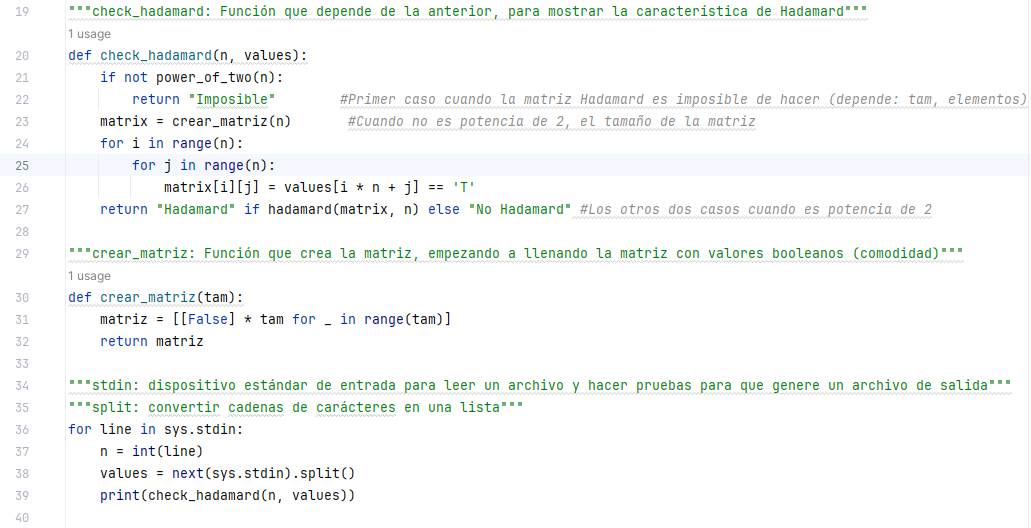
**Código**





**Documentación**





**Fuentes**

/HADAMARD\_MATRIZ.py