

Requerimientos.

El algoritmo para calcular el número de formas de cubrir un tablero $3 \times n$ con dominós 2×1 utiliza programación dinámica, ya que construye la solución del caso grande a partir de soluciones previamente calculadas de tableros más pequeños. Dado que solo se pueden cerrar correctamente bloques de dos columnas a la vez, la recurrencia depende de valores anteriores como $A(n-2)$ y $A(n-4)$. El procedimiento itera desde los casos base hasta n , realizando una cantidad constante de operaciones en cada paso, por lo que su complejidad temporal es $O(n)$. En cuanto al espacio, puede ser $O(n)$ si se almacena toda la tabla de resultados, o $O(1)$ si solo se conservan los últimos valores necesarios para calcular el siguiente término.

Requerimientos

1.1 Requerimientos Funcionales

RF-01

El sistema debe recibir como entrada un numero entero n .

RF-02

El sistema debe validar que n sea un numero entero mayor o igual a 0, para evitar errores en el proceso.

RF-03

El sistema debe verificar si n es impar. En caso de que sea impar, debe retornar 0 sin ejecutar el calculo, por lo que no es posible cubrir el tablero completamente.

RF-04

El sistema debe calcular la cantidad de formas de cubrir completamente un tablero de dimensiones $3 \times n$ utilizando programacion dinamica.

RF-05

El sistema debe implementar la siguiente recurrencia matematica:

$F(n) = 4F(n-2) - F(n-4)$ con los siguientes casos base:

$$F(0) = 1$$

$$F(2) = 3$$

RF-06

El sistema debe retornar el resultado final correspondiente a $F(n)$.

RF-07

El sistema debe ejecutar el algoritmo con una complejidad temporal $O(n)$, garantizando eficiencia en el tiempo de ejecucion.

1.2 Requerimientos No Funcionales

RNF-01

El sistema debe ejecutarse en menos de un segundo para valores de n menores o iguales a 10000000.

RNF-02

El sistema debe utilizar memoria eficiente, preferiblemente $O(n)$ o menor, para no consumir recursos innecesarios.

RNF-03

El codigo debe usar nombres de variables claros y faciles de entender.

RNF-04

El codigo debe contener comentarios explicativos que describan los pasos principales del algoritmo.

RNF-05

El sistema debe manejar entradas invalidas sin generar errores o que el programa se detenga inesperadamente.

2. Historias de Usuario HU-01

Ingreso de datos

quiero ingresar un numero entero n, para saber cuantas formas existen de cubrir un tablero $3 \times n$

Criterios de aceptacion:

- Si ingreso 8, el sistema devuelve 153
- Si ingreso 12, el sistema devuelve 213
- Si ingreso 7, el sistema devuelve 0

HU-02

Quiero que el sistema detecte cuando n es impar Para evitar que se hagan calculos innecesarios

Criterios de aceptacion:

- Si ingreso un numero impar, el sistema devuelve 0 inmediatamente.

- El algoritmo de programacion dinamica no se ejecuta cuando n es impar.

HU-03 Calculo eficiente

Quiero que el sistema realice el calculo de forma eficiente

Para poder trabajar con valores grandes de n sin que el programa sea lento

Criterios de aceptacion:

- El sistema responde correctamente para valores grandes como n = 1000.
- El algoritmo utilizado corresponde a una solucion de programacion dinamica.

HU-04 Claridad del codigo

Quiero que el codigo este organizado y bien documentado Para que sea mas facil de entender y mantener en el futuro

Criterios de aceptacion:

- El codigo incluye comentarios explicativos.
- Las variables tienen nombres claros y representativos.
- La estructura del programa es clara y modular.

3. Casos de Prueba

CP-01

Entrada: 2

Salida esperada: 3

CP-02

Entrada: 4

Salida esperada: 11

CP-03

Entrada: 8

Salida esperada: 153

CP-04

Entrada: 12

Salida esperada: 2131

CP-05

Entrada: 5

Diagrama de flujo

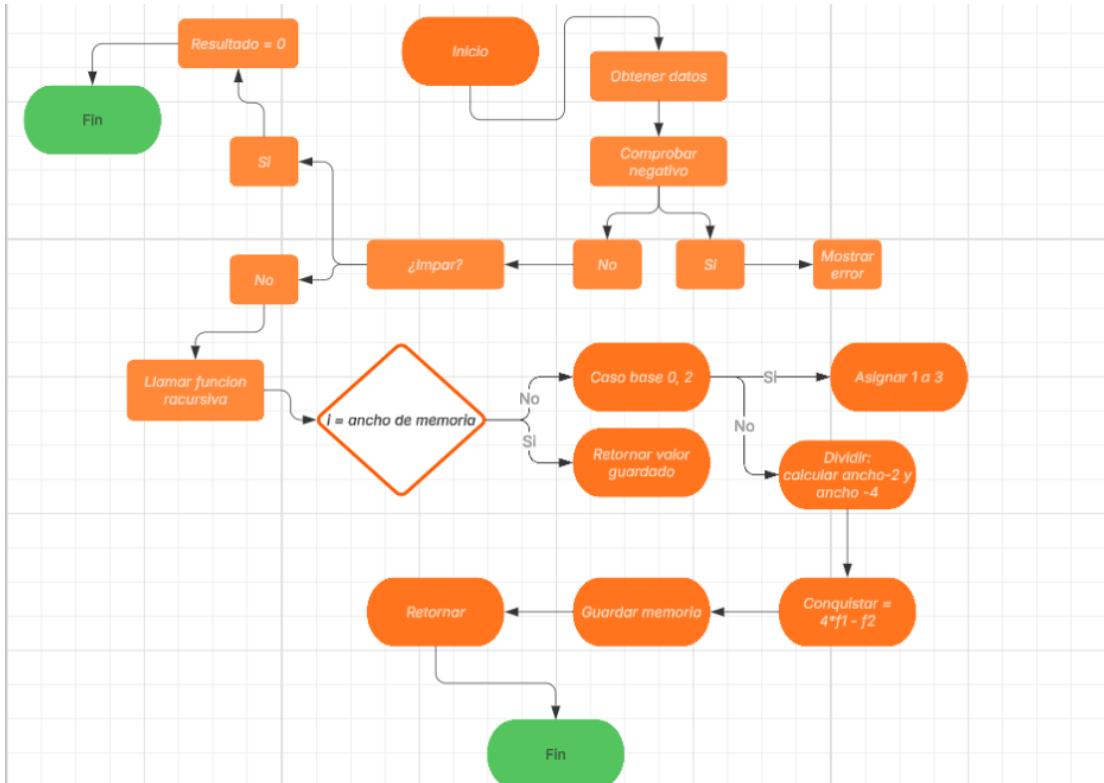


Diagrama de secuencia

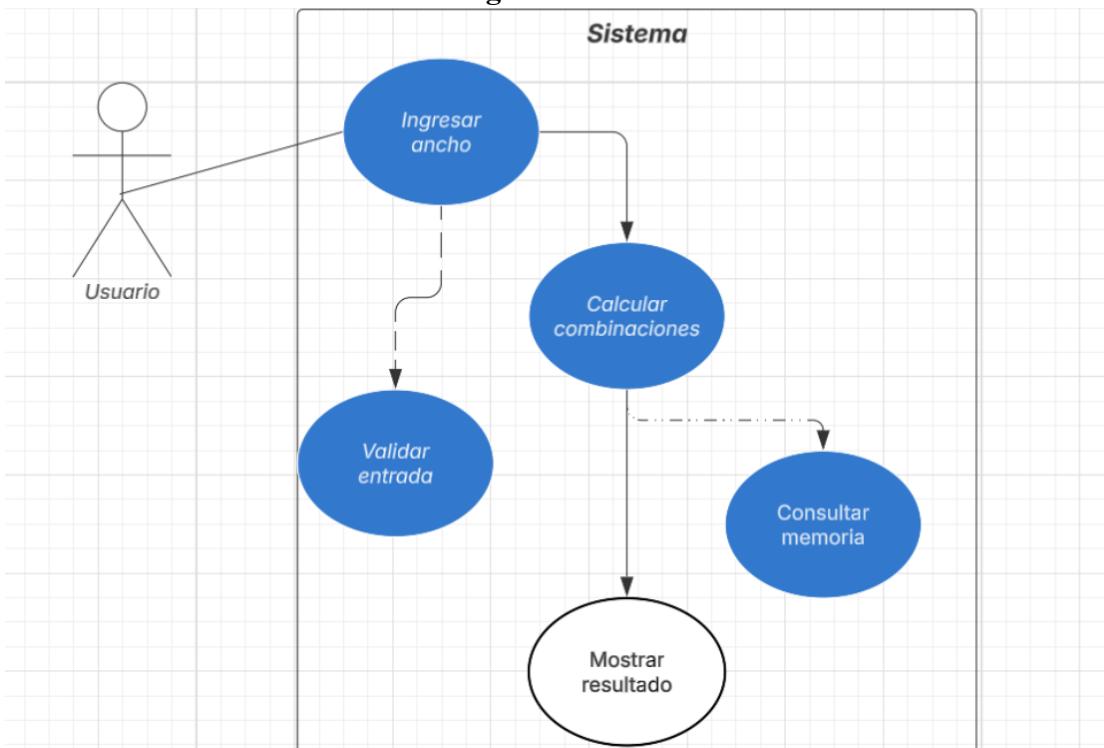


Diagrama de caso de uso

