



Cálculo de números de Fibonacci con una Máquina de Turing determinista de una sola cinta

Análisis y Diseño de Algoritmos

Angel Andres Herrarte Lorenzana, 22873

2 de marzo del 2025

Introducción

La máquina de Turing, creada por Alan Turing en 1936, es un modelo básico de computación que consiste en una máquina de estados finitos con una cinta infinita. Este modelo permite entender el funcionamiento de algoritmos complejos. Este proyecto implementa una máquina de Turing de cinta única para calcular números de Fibonacci, demostrando aplicaciones prácticas de conceptos teóricos.

La secuencia de Fibonacci (0, 1, 1, 2, 3, 5, 8...) se define como $F(n) = F(n-1) + F(n-2)$ donde $F(0) = 0$ y $F(1) = 1$. Esta secuencia tiene aplicaciones en diversos campos. El cálculo de números de Fibonacci en una máquina de Turing requiere planificación debido a la naturaleza secuencial de las operaciones de la máquina.

El proyecto tiene como objetivos:

- Diseñar una máquina de Turing de cinta única que calcule el enésimo número de Fibonacci.
- Establecer convenciones para la representación de enteros en la cinta
- Desarrollar un programa en Python que ejecute máquinas de Turing según archivos de configuración.
- Analizar empíricamente la complejidad temporal del cálculo

Para la representación de números se utilizó el sistema unario, donde un número n corresponde a n símbolos "1" consecutivos. Este sistema facilita las operaciones de suma necesarias. El diseño funciona en fases: inicialización de casos base, adición iterativa y producción del resultado final.

El análisis empírico demostrará la complejidad temporal de $O(n^2)$ para el cálculo de Fibonacci en esta implementación, validando las predicciones teóricas mediante mediciones de tiempo de ejecución con distintas entradas.

Fundamentos teóricos

Máquinas de Turing Deterministas de Cinta Única

Una máquina de Turing determinista de cinta única es un modelo computacional que consiste en cuatro componentes principales:

- Cinta infinita dividida en celdas
- Cabezal que puede leer y escribir símbolos en la cinta
- Conjunto finito de estados
- Función de transición.

La cinta contiene símbolos de un alfabeto finito, y una celda especial contiene un símbolo "blanco" que representa la ausencia de información (Hopcroft et al., 2006).

En cada paso, la máquina lee el símbolo actual bajo el cabezal, y basándose en este símbolo y su estado actual, realiza tres acciones: cambia a un nuevo estado, escribe un nuevo símbolo en la celda actual, y mueve el cabezal una posición a la izquierda o a la derecha. El término "determinista" significa que para cada combinación de estado y símbolo leído, existe exactamente una acción posible (Sipser, 2012).

La máquina comienza en un estado inicial predefinido y continúa ejecutándose hasta alcanzar un estado de aceptación (indica que el proceso ha terminado con éxito) o hasta que no existe una transición definida para el estado y símbolo actuales.

Notación Big O y Complejidad Temporal

La notación Big O proporciona una medida de la eficiencia de un algoritmo en términos de cómo crece su tiempo de ejecución en relación con el tamaño de la entrada (Cormen et al., 2009). Específicamente, $O(f(n))$ indica que el tiempo de ejecución crece a lo como máximo proporcionalmente a la función $f(n)$, donde n representa el tamaño de la entrada.

Algunos ejemplos:

- $O(1)$: Tiempo constante
- $O(\log n)$: Tiempo logarítmico
- $O(n)$: Tiempo lineal
- $O(n \log n)$: Tiempo lineal-logarítmico

- $O(n^2)$: Tiempo cuadrático
- $O(2^n)$: Tiempo exponencial

Para analizar la complejidad de un algoritmo implementado en una máquina de Turing, es necesario contar el número de pasos (transiciones de estado) que realiza la máquina en función del tamaño de la entrada (Arora & Barak, 2009).

Complejidad Teórica del Cálculo de Fibonacci en una Máquina de Turing

La complejidad temporal del cálculo de Fibonacci en una máquina de Turing depende en su mayoría de la representación de números elegida y del diseño de la máquina. Con la representación unaria (donde cada número se representa como una secuencia de unos), el cálculo del n -ésimo número de Fibonacci tiene una complejidad teórica de $O(n^2)$ (Kozen, 2006).

La complejidad es porque:

1. Para calcular $F(n)$, la máquina debe realizar $n-1$ iteraciones (desde $F(2)$ hasta $F(n)$)
2. En cada iteración, la máquina debe realizar operaciones de suma que implican copiar y mover números en representación unaria
3. A medida que los números de Fibonacci crecen, las operaciones de copia y suma requieren más pasos.

En cada iteración i , las operaciones involucran trabajar con $F(i-1)$ y $F(i-2)$, y el tamaño de estos números crece aproximadamente linealmente con i . Por lo tanto, cada iteración toma un tiempo proporcional a i , resultando en una suma total de $O(1 + 2 + \dots + n) = O(n^2)$.

Es importante mencionar que esta complejidad es específica de la implementación en máquina de Turing con una representación unaria, y esto puede cambiar dependiendo de las implementaciones en lenguajes de programación de alto nivel, donde la representación de números es más eficiente.

Especificaciones de Diseño

Convención de Representación de Enteros

Para implementar una máquina de Turing que calcule la secuencia de Fibonacci, se ha decidido utilizar la representación unaria para los enteros. Esta representación ofrece ventajas significativas para las operaciones de adición requeridas en el cálculo de Fibonacci, aunque también cuenta con desventajas en términos de espacio.

Representación Unaria

- El número entero n se representa como n símbolos "1" consecutivos.
- Ejemplo: El número 5 se representa como "11111".
- El símbolo "B" se utiliza para representar celdas vacías (blanco).
- El símbolo "0" funciona como separador entre distintos números en la cinta.
- Estado inicial: Para una entrada n , la cinta comienza con n símbolos "1" seguidos de celdas en blanco.

La justificación principal para elegir este tipo de representación es que simplifica en gran medida la implementación de operaciones de adición, que son fundamentales para el cálculo de Fibonacci. Aunque también existen representaciones más eficientes en espacio de memoria como la binaria, la representación unaria permite transiciones de estado más simples y simplifica la implementación del algoritmo.

Diseño y Estructura de la Cinta

Suma ($a + b$): Concatenar las representaciones unarias de a y b . Por ejemplo:

- $3 + 2: "111" + "11" = "11111" (5)$
- Esta operación es fundamental para el cálculo de Fibonacci y se realiza mediante estados que copian secuencias de "1" de una sección a otra

Resta ($n - 1$): Eliminar un símbolo "1" de la secuencia. Por ejemplo:

- $4 - 1: "1111" \rightarrow "111" (3)$
- Esta operación se utiliza para el contador de iteraciones

Comparación con cero: Verificar si una secuencia está vacía

- Esta operación es crucial para determinar cuándo terminar las iteraciones

Diseño y Estructura de la Cinta

La estructura de la cinta está diseñada para facilitar el cálculo iterativo de los números de Fibonacci.

- **Contador:** Inicialmente contiene n , y decrece con cada iteración completa
- **$F(i-2)$:** Contiene el $(i-2)$ -ésimo número de Fibonacci en formato unario
- **$F(i-1)$:** Contiene el $(i-1)$ -ésimo número de Fibonacci en formato unario
- **$F(i)$:** Contiene el i -ésimo número de Fibonacci (resultado de la iteración actual)
- Los separadores "0" permiten a la máquina identificar los límites entre valores
- "B" representa las celdas en blanco (infinitas hacia la derecha)

Figura 1: Disposición inicial de la cinta

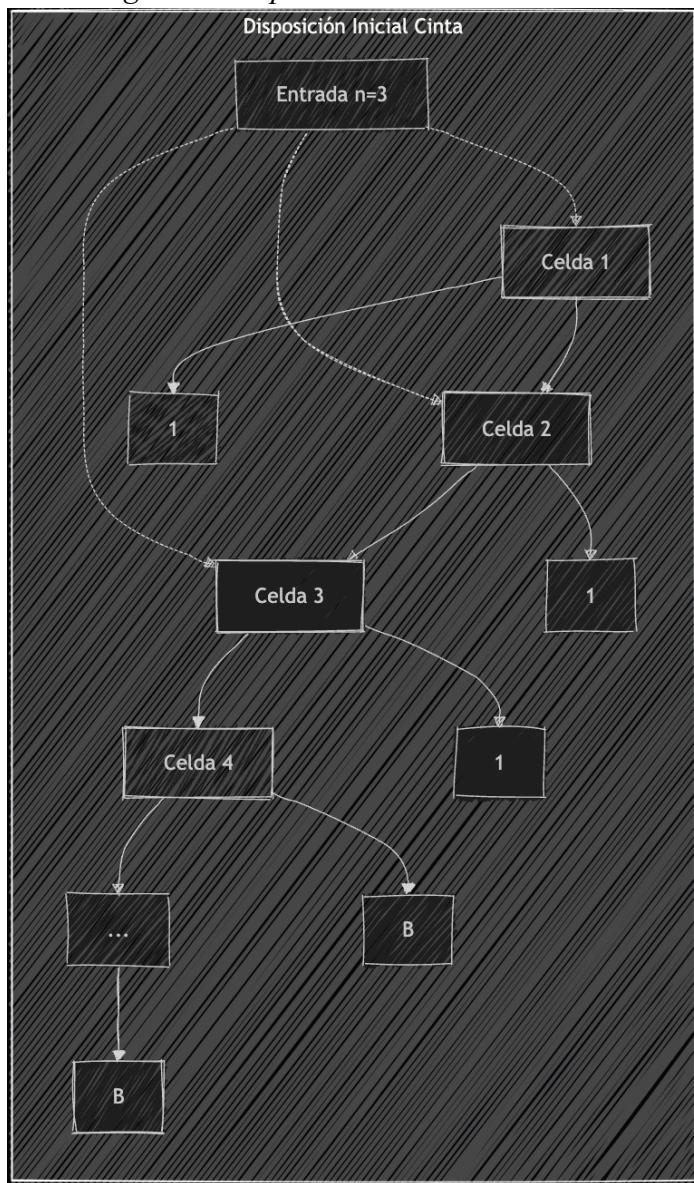
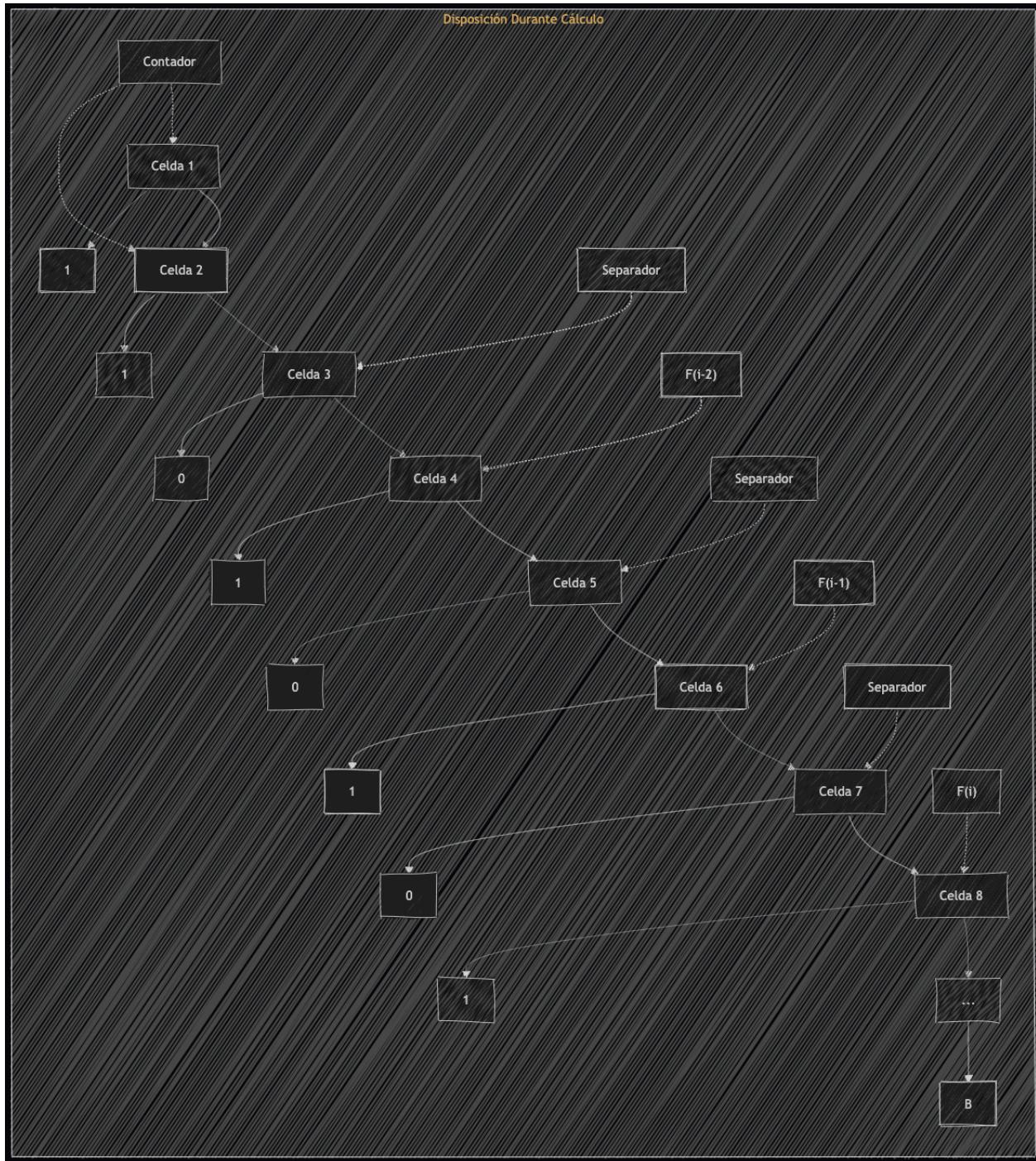


Figura 2: Disposición durante el cálculo



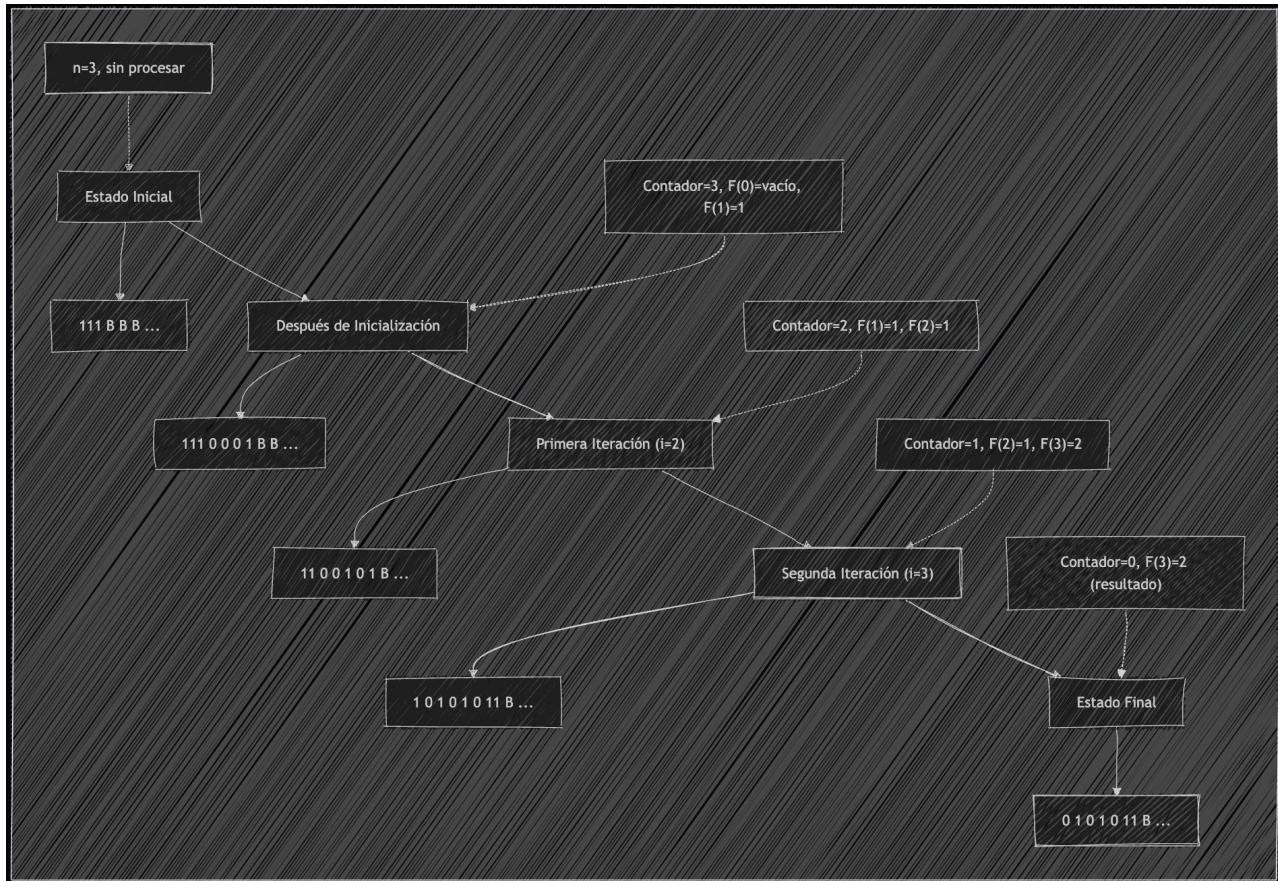
Ejemplo de la Cinta para $n=3$

- Estado inicial:** 111 B B B ... (Entrada $n=3$)
- Después de la inicialización:** 111 0 0 0 1 B B ... (Contador=3, $F(0)=\epsilon$ (vacío), $F(1)=1$)

3. **Primera iteración (i=2):** 11 0 0 1 0 1 B ... (Contador=2, F(1)=1, F(2)=1)
4. **Segunda iteración (i=3):** 1 0 1 0 1 0 11 B ... (Contador=1, F(2)=1, F(3)=2)
5. **Estado final:** 0 1 0 1 0 11 B ... (Contador=0, F(3)=2 como resultado final)

Para interpretar el resultado final, después de que la máquina termina su ejecución, el n-ésimo número de Fibonacci se encuentra en la última sección de la cinta, representado en formato unario. En este ejemplo, F(3) = "11" que representa el valor 2.

Figura 3: Ejemplo de la cinta para n = 3



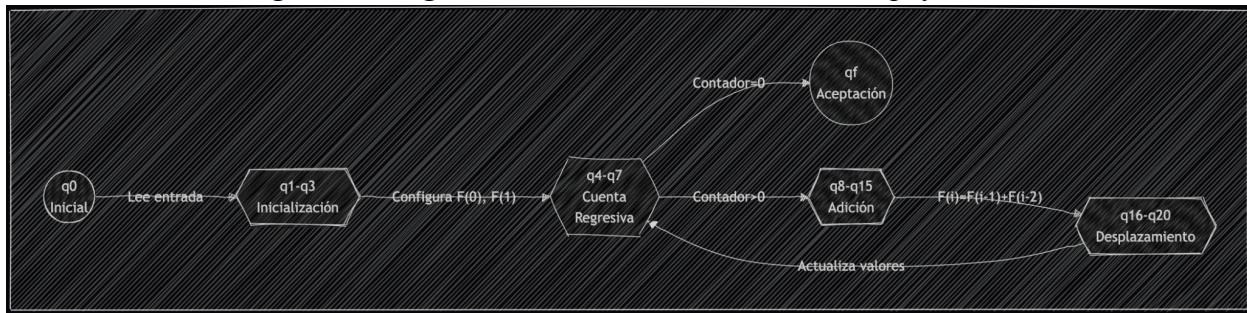
Arquitectura de la Máquina de Turing

La máquina de Turing para calcular la secuencia de Fibonacci está compuesta por 22 estados, incluyendo los estados inicial y de aceptación. Estos estados están organizados en grupos que realizan operaciones específicas durante el cálculo.

Grupos de estados principales

1. **Estado Inicial (q0)**: Lee la entrada n y prepara la cinta para el cálculo.
2. **Estados de Inicialización (q1-q3)**: Configuran los valores iniciales $F_0=0$ (representado como una cadena vacía) y $F_1=1$ (representado como un solo "1") en la cinta.
3. **Estados de Cuenta Regresiva (q4-q7)**: Controlan el número de iteraciones a realizar, disminuyendo el contador n con cada ciclo completo.
4. **Estados de Adición (q8-q15)**: Realizan la operación $F(i) = F(i-1) + F(i-2)$ mediante la copia de $F(i-1)$ para crear $F(i)$ y luego añadiendo $F(i-2)$.
5. **Estados de Desplazamiento (q16-q20)**: Actualizan los valores para la siguiente iteración, desplazando $F(i-1)$ a la posición de $F(i-2)$ y $F(i)$ a la posición de $F(i-1)$.
6. **Estado de Aceptación (qf)**: Estado final alcanzado cuando se completa el cálculo del n-ésimo número de Fibonacci.

Figura 4: Diagrama de Transición de Estados Simplificado



Algoritmo de Fibonacci Implementado

El algoritmo de Fibonacci implementado en esta máquina de Turing sigue un enfoque iterativo utilizando tres registros (secciones de la cinta) para almacenar los valores consecutivos de la secuencia.

Figura 5: Pseudocódigo del Algoritmo:

```
Función CalcularFibonacci(n):
    1. Inicializar contador = n
    2. Inicializar F(0) = ε (cadena vacía, representa 0 en unario)
    3. Inicializar F(1) = "1" (representa 1 en unario)

    4. Mientras contador > 0:
        5. F(i) = F(i-1) + F(i-2) // Suma mediante copia
        6. F(i-2) = F(i-1)       // Desplazamiento
        7. F(i-1) = F(i)         // Desplazamiento
        8. contador = contador - 1 // Decrementar contador

    9. Retornar F(n) // Resultado final
```

La complejidad temporal teórica de esta implementación es $O(n^2)$, donde n es el número de entrada. Cada iteración requiere copiar y sumar números en representación unaria, lo que toma $O(n)$ pasos, y realizamos n iteraciones (Kozen, 2006).

Detalles de implementación

Formato del Archivo de Configuración

El formato JSON fue seleccionado para el archivo de configuración debido a su legibilidad, facilidad de parseo y estructura jerárquica que representa naturalmente los elementos de la máquina.

Figura 6: Archivo de configuración

```
{  
    "states": ["q0", "q1", "q2", "q3", "q4", "q5", "q6", "q7", "q8",  
              "q9", "q10", "q11", "q12", "q13", "q14", "q15", "q16",  
              "q17", "q18", "q19", "q20", "qf"],  
    "alphabet": ["0", "1", "B"],  
    "blank": "B",  
    "initial_state": "q0",  
    "accepting_states": ["qf"],  
    "transitions": {  
        "q0": {  
            "1": {"next_state": "q0", "write": "1", "move": "R"},  
            "B": {"next_state": "q1", "write": "0", "move": "R"}  
        },  
        "q1": {  
            "0": {"next_state": "q1", "write": "0", "move": "R"},  
            "B": {"next_state": "q2", "write": "0", "move": "R"}  
        },  
        "q2": {  
            "0": {"next_state": "q2", "write": "0", "move": "R"},  
            "B": {"next_state": "q3", "write": "0", "move": "R"}  
        },  
        "q3": {  
            "0": {"next_state": "q3", "write": "0", "move": "R"},  
            "B": {"next_state": "q4", "write": "1", "move": "L"}  
        }  
    }  
}
```

Componentes del archivo de configuración

- **Estados:** Lista de todos los estados de la máquina.
- **Alfabeto:** Conjunto de símbolos que la máquina puede leer y escribir.
- **Símbolo blanco:** Símbolo que representa una celda vacía.
- **Estado inicial:** Estado en el que la máquina comienza la ejecución.
- **Estados de aceptación:** Estados que indican la finalización exitosa.
- **Función de transición:** Define para cada combinación de estado y símbolo leído:
 - El próximo estado
 - El símbolo a escribir
 - La dirección de movimiento del cabezal (L: izquierda, R: derecha)

Referencias

- Arora, S., & Barak, B. (2009). Complejidad computacional: Un enfoque moderno. Cambridge University Press.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introducción a algoritmos (3ra edición.). MIT Press.
- Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). Introduction to automata theory, languages, and computation (3rd ed.). Pearson Education.
- Kozen, D. C. (2006). Teoría de la computación. Springer.
- Papadimitriou, C. H. (1994). Computational complexity. Addison-Wesley.
- Sipser, M. (2012). Introduction to the theory of computation (3rd ed.). Cengage Learning.