Universidad del Valle de Guatemala

Facultad de Ingeniería Ciencias de la Computación

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 que almacenan símbolos de un alfabeto finito (incluyendo un símbolo "blanco" B).
- Cabezal móvil: Lee/escribe símbolos y se desplaza izquierda/derecha.
- Estados finitos: Incluyen un estado inicial (q0) y estados de aceptación/rechazo.
- Función de transición: Reglas deterministas de la forma (*estado_actual, símbolo_leído)* → *(estado_nuevo, símbolo_escrito, movimiento*).

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 de la representación de números elegida y del diseño de la máquina. Con la representación unaria, 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 $O(1 + 2 + ... + n) = O(n^2)$.

$$\text{Pasos totales} = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \implies O(n^2)$$

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 "X" se utiliza para representar celdas vacías.
- El símbolo "0" funciona como separador entre distintos números en la cinta.
- El símbolo "M" se utiliza como marcador durante operaciones de copia y desplazamiento.
- Estado inicial: Para una entrada “n”, la cinta comienza con “n” símbolos "1" seguidos de celdas "X".

La representación unaria simplifica la implementación de operaciones de adición, fundamentales para el cálculo de Fibonacci. Aunque existen representaciones más eficientes en espacio como la binaria, la unaria permite transiciones de estado más simples y facilita la visualización del proceso.

Diseño y Estructura de la Cinta

La implementación utiliza una cinta con la siguiente estructura:

Durante la ejecución, la cinta cambia para almacenar:

- Los valores de Fibonacci anteriores
- El contador de iteraciones
- Marcadores temporales durante las operaciones

Las operaciones básicas implementadas incluyen:

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

- 3 + 2: "111" + "11" = "11111" (5)
- Esta operación se realiza mediante estados que copian secuencias de "1" de una sección a otra

Marcado de posiciones: El símbolo "M" se utiliza para marcar posiciones específicas durante el procesamiento.

Comparación con cero: Verificar si una secuencia está vacía.

- Esta operación es crucial para determinar cuándo terminar las iteraciones.

Arquitectura de la Máquina de Turing

En el archivo *config.json* se define la estructura que tendrá la máquina de Turing.

Se define los siguiente:

- 18 estados diferentes (desde el estado "0" al estado "18")
- Un alfabeto de cinta que incluye: "0", "1", "X", "M"
- El estado "0" como estado inicial
- El estado "18" como estado de aceptación

La implementación está estructurada como una máquina de estados determinista, donde cada transición específica:

1. El nuevo estado
2. El símbolo a escribir
3. La dirección de movimiento del cabezal (1 para derecha, -1 para izquierda)

El simulador mantiene control del estado actual, la posición del cabezal y el contenido de la cinta, permitiendo visualizar el proceso paso a paso durante la ejecución.

Análisis Empírico

Los resultados empíricos obtenidos al ejecutar la máquina de Turing para calcular los primeros 10 números de Fibonacci validan la corrección del algoritmo, ya que todas las salidas coinciden con los valores esperados. Sin embargo, el análisis de rendimiento revela que el rendimiento del algoritmo es deficiente. Como se indicó anteriormente, se necesita bastante espacio en la memoria.

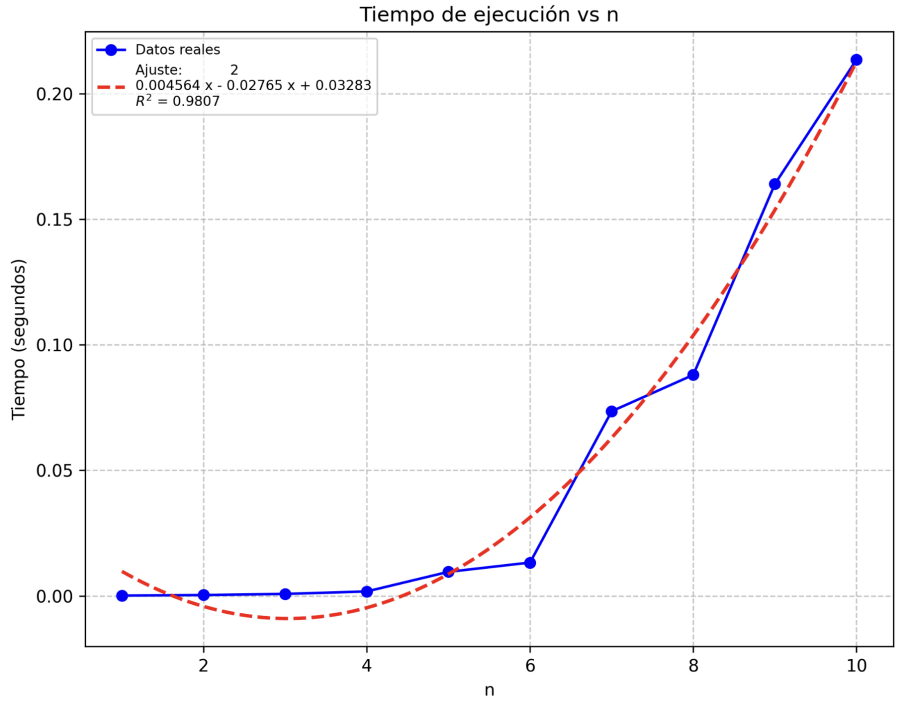
Entradas de prueba

n	Pasos	Tiempo (s)
1	2	0.000115
2	19	0.000332
3	52	0.000771
4	107	0.001757
5	214	0.009559
6	427	0.013262
7	894	0.073515
8	1973	0.088002
9	4590	0.164029
10	11117	0.213565

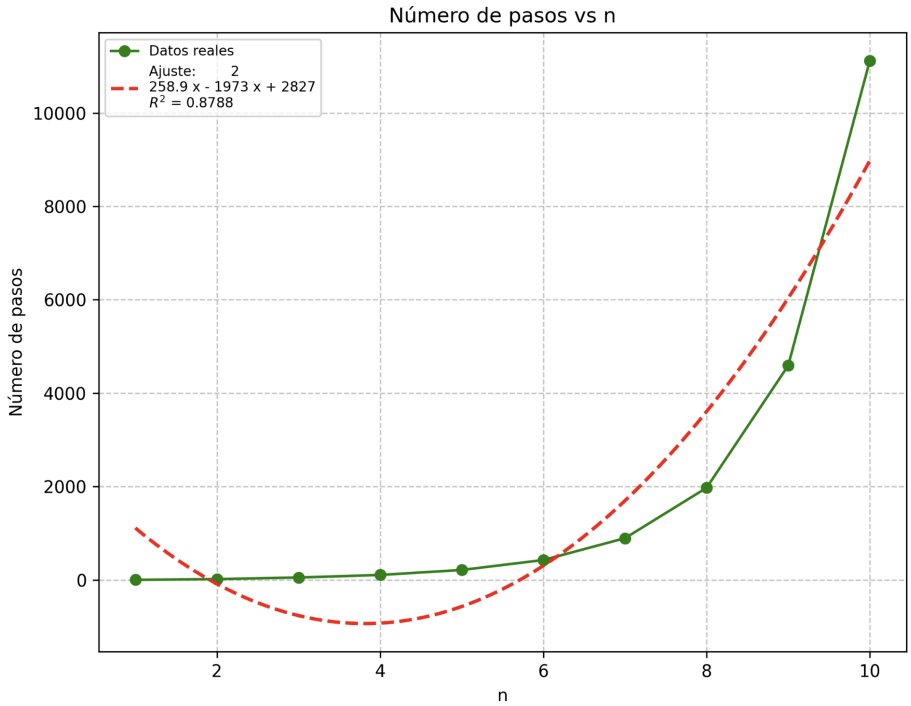
Cada entrada representa el índice n del número de Fibonacci en notación unaria, siguiendo la convención establecida en la sección de diseño.

Diagrama de dispersión con tiempos de ejecución

Los puntos azules representan los tiempos medidos, mientras que la línea punteada roja corresponde a la regresión polinomial ajustada. Se observa un crecimiento acelerado a partir de n=5, lo que concuerda con la predicción teórica de complejidad cuadrática.



La siguiente gráfica muestra el número de pasos (transiciones de estado) ejecutados por la máquina de Turing en función de n. Este análisis complementario confirma el crecimiento cuadrático del algoritmo, con una curva más pronunciada para valores grandes de n.



Regresión polinomial

Para cuantificar la complejidad temporal del algoritmo, se realizó un ajuste polinomial de segundo grado sobre los datos de tiempo y número de pasos.

Tiempo de ejecución (s)

$$Tiempo(n) = 0.004564x^2 - 0.02765x + 0.03283$$

El resultado cuenta con un coeficiente de correlación R^2 = 0.9807, lo que indica un excelente ajuste al modelo cuadrático.

Número de pasos

$$Pasos(n) = 258.9x^2 - 1973x + 2827$$

El coeficiente de correlación es de R^2 = 0.8788, por lo que también se ajusta al modelo planteado.

Los coeficientes positivos de los términos cuadráticos en ambos casos confirman la complejidad O(n²) predicha teóricamente. Es interesante notar que mientras el tiempo crece de manera más uniforme, el número de pasos muestra más variabilidad para valores pequeños de n, lo que explica el menor valor de R² en este caso.

Análisis de rendimiento

La tabla siguiente muestra los tiempos de ejecución y número de pasos para cada entrada:

n	Pasos	Tiempo (s)
1	2	0.000115
2	19	0.000332
3	52	0.000771
4	107	0.001757
5	214	0.009559
6	427	0.013262
7	894	0.073515
8	1973	0.088002
9	4590	0.164029
10	11117	0.213565

Estos resultados concuerdan con la fórmula teórica para la suma de los primeros n números $\frac{(n(n+1))}{2}$, que predice un crecimiento cuadrático.

Los resultados empíricos confirman que el cálculo de Fibonacci en una máquina de Turing determinista de una sola cinta con representación unaria tiene una complejidad temporal de $O(n^2)$, validando el análisis teórico anteriormente expuesto.

Apéndice

Repositorio: https://github.com/aherrarte2019037/big-o-fibonnaci

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.