

Complejidad Algorítmica

Unidad 2: Algoritmos voraces, programación dinámica y problemas P-NP

Módulo 14: Computabilidad

Complejidad Algorítmica

Semana 14 / Sesión 1

Objetivo

Comprender los conceptos de Computabilidad.

MÓDULO 14: Computabilidad



Contenido

1. Problemas Computables vs No Computables
2. Computación: Historia y modelos formales computables
 - Programas While
 - Máquinas de Turing



Preguntas / Conclusiones

1. Problemas Computables vs No Computables

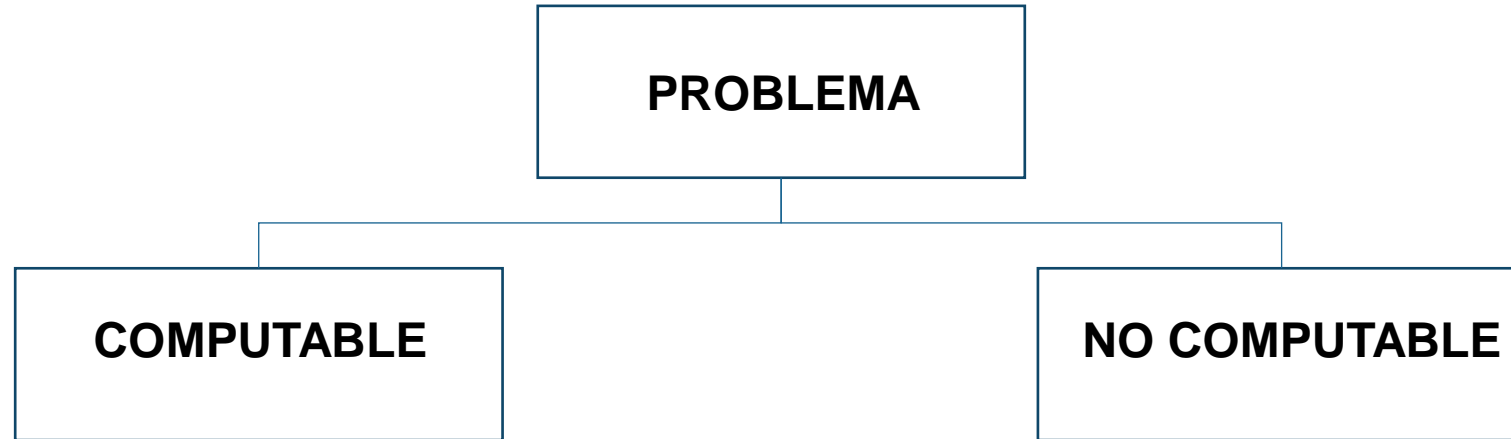
- Los humanos resolvemos problemas desde el inicio de nuestra existencia.
- La resolución de problemas matemáticos son un tipo de problemas que ocupan un lugar especial.
- El crecimiento exponencial de la capacidad de procesamiento computacional (a partir de los años 70), ha llevado a resolver en segundos, cuando antes demandaban mucho tiempo.

¿Cuál es la diferencia
entre resolver un
problema y reconocer
su solución?



- Pero a pesar de este crecimiento computacional, existen aun problemas que demandarían para ser resueltos, más años que la vida humana (ejemplo: factorizar el producto de números primos grandes).
- Esto nos lleva a determinar que:
 - ☐ Algunos problemas no tienen una solución algorítmica: **resultados indecidibles**
 - ☐ Incluso si tienen una solución algorítmica (i.e. son computables), puede ser intratables por su elevada complejidad.
 - ☐ Desafortunadamente, probar la computabilidad de un problema puede ser tan difícil como encontrar un algoritmo eficiente que lo resuelva

1. Problemas Computables vs No Computables



- Si tienen una solución algorítmica
- Pero puede ser intratables por su elevada complejidad.

- No tienen una solución algorítmica
- Sus resultados pueden ser indecidibles

Desafortunadamente, probar la computabilidad de un problema puede ser tan difícil como encontrar un algoritmo eficiente que lo resuelva

1. Problemas Computables vs No Computables

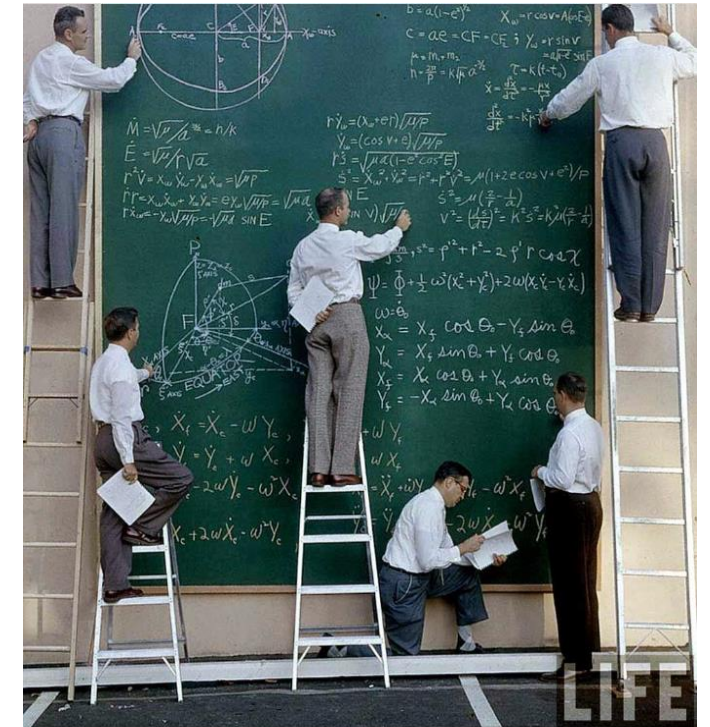
EJEMPLO

Implementar un programa de computadora que imprima “**hola**” cuando encuentre un entero positivo $n > 2$, que cumpla: $x^n + y^n = z^n$, para x, y, z enteros positivos.

SOLUCION:

La solución entera de esta ecuación se conoce como el último teorema de Fermat, que llevó a los matemáticos 300 años resolver.

- El poder analizar cualquier programa de computadora y decidir si va a imprimir un resultado como “hola” es en general indecidable.
- Para ello, existe **la teoría de la indecibilidad** basada en modelos computacionales sencillos (i.e., Programas WHILE, Máquina de Turing, ...).



Científicos de la NASA con su tabla de cálculos, Life Magazine, 1957

2. Computación: Historia y modelos formales

HISTORIA

- Siglo. XIX **D. Hilbert**. es posible encontrar un algoritmo que determinara el valor de verdad de una fórmula en lógica de primer orden.
- 1931 **K. Godel**: demostró su teoría de incompletitud para probar que no se podía construir dicho algoritmo.
- 1936 **A. Turing**: publicó su máquina de Turing como un modelo para cualquier tipo de computación (aunque todavía no existían las computadoras).

MODELOS



2. Computación: Historia y modelos formales

MODELOS FORMALES DE COMPUTACION

- Programas LOOP
- **Programas WHILE**
- Programas GOTO
- **Máquinas de Turing**
- Máquinas Post
- Máquinas RAM
- Cálculo λ , etc.

Programas WHILE

- Los programas LOOP, WHILE y GOTO están estructurados como lenguajes de programación (simples) tradicionales.
- Usan un número finito de variables del conjunto $\{x_0, x_1, x_2, \dots\}$ pueden tomar valores N.
- Difieren entre ellos en las “expresiones” permitidas.

Máquinas de Turing

- Para hablar de “cualquier máquina de computación realizable”, se necesita un modelo formal de computación.
- El autómata estándar para este trabajo es la máquina de Turing, nombrada en honor a Alan Turing, el “Padre las Ciencias de la Computación”.

2. Computación: Historia y modelos formales

MODELOS FORMALES DE COMPUTACION

Programas While

- Se define como todo programa que pueda ser implementado con la semántica de un programa WHILE y las operaciones básicas:

- $x_i \leftarrow 0 \ \forall i, c \in \mathbb{N}$ (asignación a 0)
- $x_i \leftarrow x_i + 1 \ \forall i \in \mathbb{N}$ (incrementar)
- $x_i \leftarrow x_i - 1 \ \forall i \in \mathbb{N}$ (decrementar)

Semántica de los programas WHILE

```
1 while  $x_i \neq 0$  do
2   | P;
3 end
```

// P es un programa WHILE.

- Si x_i tiene el valor 0, la ejecución del programa termina.
- En caso contrario, ejecutar P.
- Hay que repetir estos pasos hasta que la ejecución termine (potencialmente infinitamente).
- Si P1 y P2 son programas WHILE, entonces P1; P2 (composición).

2. Computación: Historia y modelos formales

MODELOS FORMALES DE COMPUTACION

Ejemplos de Programas While

$x_i \leftarrow c \ \forall i, c \in \mathbb{N}$
(asignación)

Algoritmo 1: $x_i \leftarrow x_j + c$

```
1  $x_i \leftarrow 0$ ;  
2 while  $c \neq 0$  do  
3    $x_i \leftarrow x_i + 1$ ;  
4    $c \leftarrow c - 1$ ;  
5 end
```

$x_i \leftarrow x_j - c \ \forall i, j, c \in \mathbb{N}$
(resta)

Algoritmo 3: $x_i \leftarrow x_j - c$

```
1  $x_k \leftarrow x_j$ ;  
2 while  $c \neq 0$  do  
3    $x_i \leftarrow x_i - 1$ ;  
4    $c \leftarrow c - 1$ ;  
5 end
```

$x_i \leftarrow x_j + c \ \forall i, j, c \in \mathbb{N}$
(suma)

Algoritmo 2: $x_i \leftarrow x_j + c$

```
1  $x_i \leftarrow x_j$ ;  
2 while  $c \neq 0$  do  
3    $x_i \leftarrow x_i + 1$ ;  
4    $c \leftarrow c - 1$ ;  
5 end
```

$x_i \leftarrow x_j * c \ \forall i, j, c \in \mathbb{N}$
(multiplicación)

Algoritmo 4: $x_i \leftarrow x_j * c$

```
1  $x_i \leftarrow 0$ ;  
2 while  $c \neq 0$  do  
3    $x_i \leftarrow x_i + x_j$ ;  
4    $c \leftarrow c - 1$ ;  
5 end
```

2. Computación: Historia y modelos formales

MODELOS FORMALES DE COMPUTACION

La Máquina de Turing



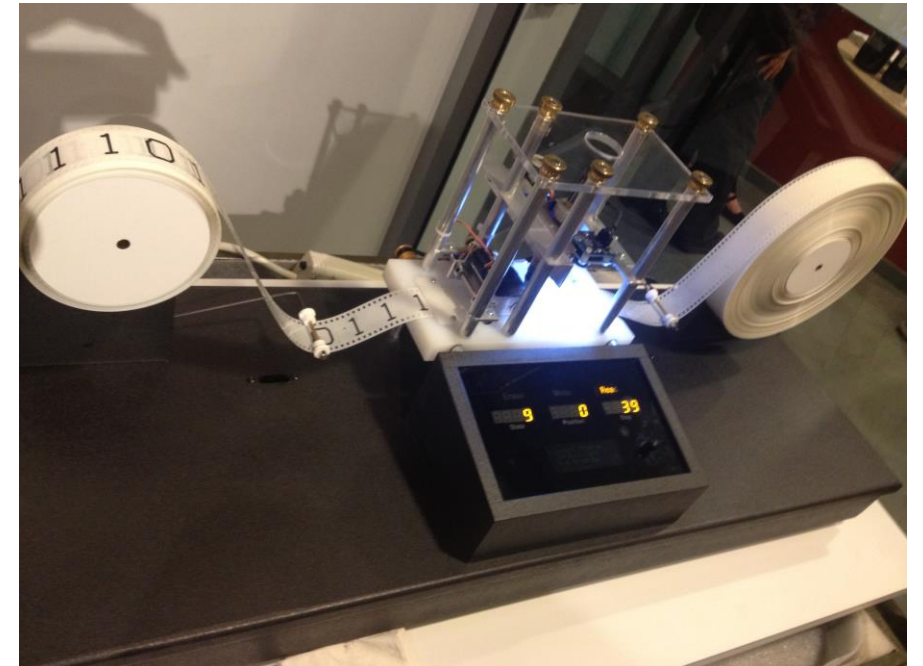
Alan Turing llamado el padre de la Informática y su máquina para resolver algoritmos.

- Alan Turing fue un matemático que, junto con sus compañeros, se vio desafiado por la cuestión de la computabilidad.
- En 1936, escribió un artículo, **“On Computable Numbers, with an Application to the Entscheidungs problema”**, que proponía que se podía especificar una máquina hipotética para resolver cualquier problema soluble, usando reglas simples.
- **Alan Turing** se dio cuenta de que necesitaba una descripción matemática de una máquina que pudiera resolver algoritmos.
- Esta máquina es conocida como la máquina de Turing.

2. Computación: Historia y modelos formales

Componentes de una Máquina de Turing

- Una **cinta infinitamente larga dividida en cuadrados** (cada cuadrado puede estar en blanco o contener un símbolo)
- Un **alfabeto finito de símbolos**, por ejemplo corchete izquierdo, 0, coma, 1, corchete derecho, { 0 , 1 }, el conjunto de dígitos binarios
- Un **cabezal sensor de lectura/escritura**: esto sería capaz de mirar un cuadrado en la cinta y leer su símbolo.
 - ✓ En respuesta a lo que se leyó, la máquina podría dejar el símbolo sin cambios o reemplazarlo con otro símbolo (incluso borrar el símbolo para dejar la celda en blanco).
 - ✓ Luego, la cabeza se movería hacia la izquierda o hacia la derecha un cuadrado a la vez.
- Un **conjunto finito de estados**: un estado es el estado de inicio y habría uno o más estados de parada (sin transiciones salientes que hagan que la máquina se detenga o se detenga)
- Un **conjunto de reglas** (reglas de transición): para especificar cómo funciona la máquina.




2. Computación: Historia y modelos formales

Reglas de transición de una Máquina de Turing

- Las reglas de transición para una máquina de Turing se expresan en forma de una función de transición.
- Por convención, la letra griega 'delta' (δ) se usa para representar la función de transición, que se escribe en la forma:

$$\delta (\text{Estado actual, símbolo de entrada}) = (\text{nuevo estado, símbolo de salida, movimiento})$$

- Así la regla $\delta (S0, 0) = (S3, 1, \leftarrow)$  Si la máquina está actualmente en el estado 0 (S0) y el símbolo de entrada es 0, ENTONCES la máquina pasará al estado 3 (S3), escriba un 1 en la cinta y mueva una celda a la izquierda.

Una función o regla de transición también se puede expresar como un **diagrama de transición de estado**. Veamos un ejemplo!

2. Computación: Historia y modelos formales

Reglas de transición/Diagrama de estado de una Máquina de Turing

Consideremos el siguiente conjunto de reglas:

Función de transición

$\delta(S1, 0) = (S3, 0, \rightarrow)$
 $\delta(S1, 1) = (S2, 1, \rightarrow)$
 $\delta(S1, \square) = (S5, \square, \rightarrow)$
 $\delta(S2, 0) = (S2, 0, \rightarrow)$
 $\delta(S2, 1) = (S3, 1, \rightarrow)$
 $\delta(S2, \square) = (S4, 0, \rightarrow)$
 $\delta(S3, 0) = (S3, 0, \rightarrow)$
 $\delta(S3, 1) = (S2, 1, \rightarrow)$
 $\delta(S3, \square) = (S4, 1, \rightarrow)$

- En estas reglas, el símbolo \square se usa para representar una celda en blanco en la cinta
- Las flechas representan el movimiento (izquierda o derecha) del cabezal de lectura/escritura.
- Para dibujar el diagrama de transición de estado equivalente, debemos seguir una secuencia lógica de pasos.

¡Dibujemos el diagrama de transición de la MT a partir de la Función de transición!

2. Computación: Historia y modelos formales

Reglas de transición/Diagrama de estado de una Máquina de Turing

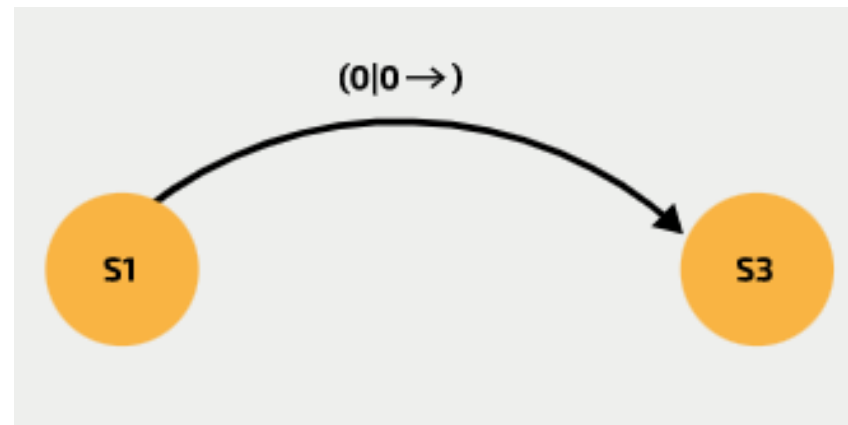
Consideremos el siguiente conjunto de reglas (continuación)

Función de transición

$\delta(S1, 0) = (S3, 0, \rightarrow)$
 $\delta(S1, 1) = (S2, 1, \rightarrow)$
 $\delta(S1, \square) = (S5, \square, \rightarrow)$
 $\delta(S2, 0) = (S2, 0, \rightarrow)$
 $\delta(S2, 1) = (S3, 1, \rightarrow)$
 $\delta(S2, \square) = (S4, 0, \rightarrow)$
 $\delta(S3, 0) = (S3, 0, \rightarrow)$
 $\delta(S3, 1) = (S2, 1, \rightarrow)$
 $\delta(S3, \square) = (S4, 1, \rightarrow)$

Paso 1

- Comienza tomando la primera regla: $\delta(S1, 0) = (S3, 0, \rightarrow)$.
- Necesitas dibujar los dos estados y la línea de transición;
- Etiquetar la entrada, la salida y el movimiento en la línea de transición.



2. Computación: Historia y modelos formales

Reglas de transición/Diagrama de estado de una Máquina de Turing

Consideremos el siguiente conjunto de reglas (continuación)

Función de transición

$\delta(S1, 0) = (S3, 0, \rightarrow)$
 $\delta(S1, 1) = (S2, 1, \rightarrow)$
 $\delta(S1, \square) = (S5, \square, \rightarrow)$
 $\delta(S2, 0) = (S2, 0, \rightarrow)$
 $\delta(S2, 1) = (S3, 1, \rightarrow)$
 $\delta(S2, \square) = (S4, 0, \rightarrow)$
 $\delta(S3, 0) = (S3, 0, \rightarrow)$
 $\delta(S3, 1) = (S2, 1, \rightarrow)$
 $\delta(S3, \square) = (S4, 1, \rightarrow)$

Paso 2

- Tomamos la siguiente regla: $\delta(S1, 1) = (S2, 1, \rightarrow)$.
- Debe agregar un nuevo estado (S2) al diagrama y una transición de S1 a S2.
- Etiquetar la entrada, la salida y el movimiento en la línea de transición.



2. Computación: Historia y modelos formales

Reglas de transición/Diagrama de estado de una Máquina de Turing

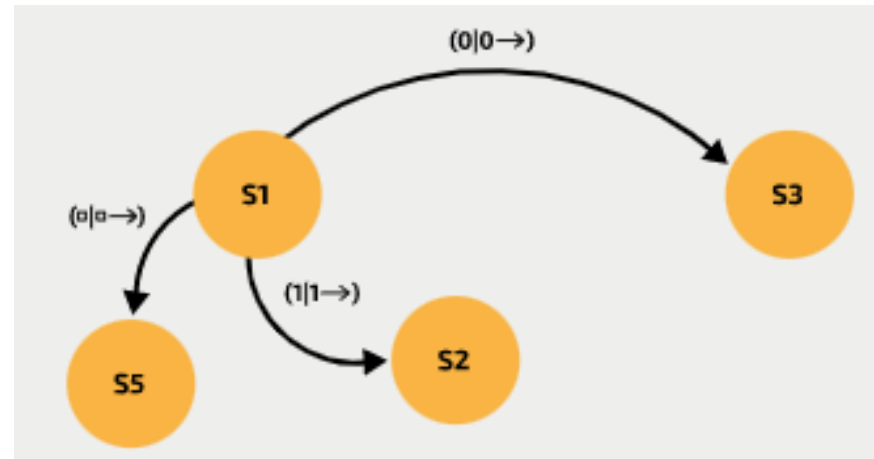
Consideremos el siguiente conjunto de reglas (continuación)

Función de transición

$\delta(S1, 0) = (S3, 0, \rightarrow)$
 $\delta(S1, 1) = (S2, 1, \rightarrow)$
 $\delta(S1, \square) = (S5, \square, \rightarrow)$
 $\delta(S2, 0) = (S2, 0, \rightarrow)$
 $\delta(S2, 1) = (S3, 1, \rightarrow)$
 $\delta(S2, \square) = (S4, 0, \rightarrow)$
 $\delta(S3, 0) = (S3, 0, \rightarrow)$
 $\delta(S3, 1) = (S2, 1, \rightarrow)$
 $\delta(S3, \square) = (S4, 1, \rightarrow)$

Paso 3

- Tomamos la siguiente regla: $\delta(S1, \square) = (S5, \square, \rightarrow)$.
- Agregar un nuevo estado (S5) al diagrama y etiquetar la transición de S1 a S5.



2. Computación: Historia y modelos formales

Reglas de transición/Diagrama de estado de una Máquina de Turing

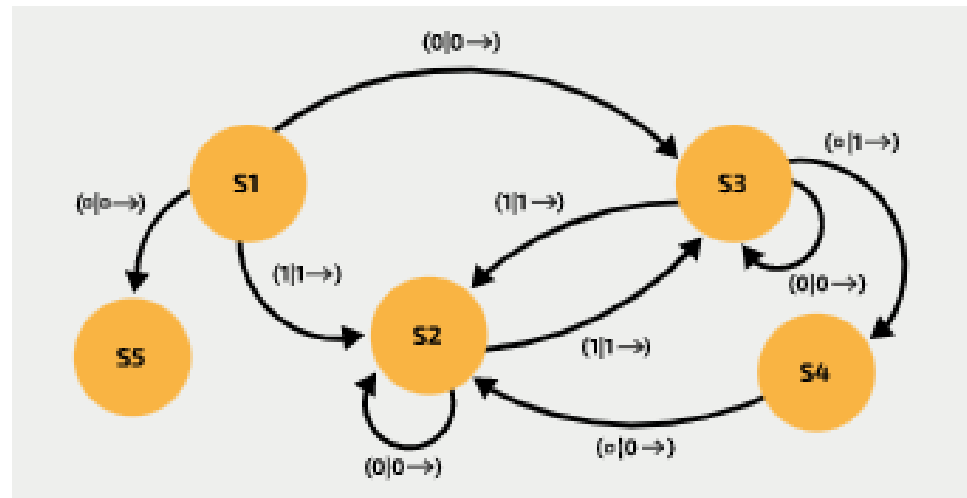
Consideremos el siguiente conjunto de reglas (continuación)

Función de transición

$\delta(S1, 0) = (S3, 0, \rightarrow)$
 $\delta(S1, 1) = (S2, 1, \rightarrow)$
 $\delta(S1, \square) = (S5, \square, \rightarrow)$
 $\delta(S2, 0) = (S2, 0, \rightarrow)$
 $\delta(S2, 1) = (S3, 1, \rightarrow)$
 $\delta(S2, \square) = (S4, 0, \rightarrow)$
 $\delta(S3, 0) = (S3, 0, \rightarrow)$
 $\delta(S3, 1) = (S2, 1, \rightarrow)$
 $\delta(S3, \square) = (S4, 1, \rightarrow)$

Paso 4

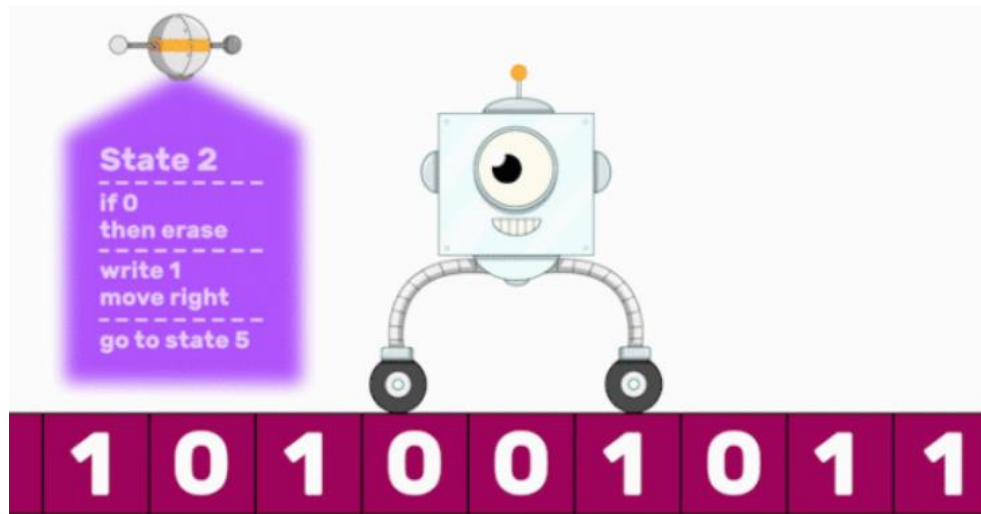
- Continuamos tomando cada regla por turno.
- Etiquetamos las transiciones con cuidado, hasta que hayamos completado el diagrama.



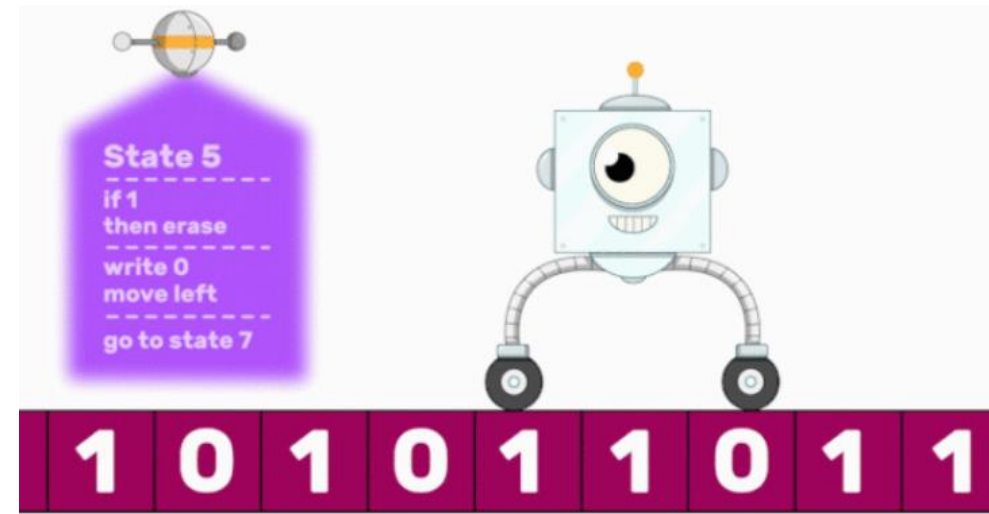
2. Computación: Historia y modelos formales

Máquina de Turing: ¿Como funciona?

Ejemplo: En esta máquina, leamos el estado y las instrucciones a seguir:



- La cabeza está en el estado 2
- Si lee un 0, bórralo, escriba un 1
- Mueva una celda a la derecha y cambie su estado a 5.



- La cabeza esta ahora en el estado 5.
- Si lee un 1, bórralo, escriba un 0 y muévase una celda a la izquierda. Luego vaya al estado 7.

Alan Turing demostró que cualquier problema que sea computable puede ser calculado por una máquina de Turing utilizando este sencillo sistema.

2. Computación: Historia y modelos formales

Máquina de Turing: Como construir una MT

Ejemplo #1:

Symbol	Write	Move
Blank	None	None
0	1	Left
1	0	Left



Tenemos la cinta, el cabezal, algunas instrucciones iniciales y, para empezar, colocaremos algunos 1s y 0s en la cinta.

Pasos según la tabla:

- El primer símbolo que ve la cabeza es a 0, y las instrucciones dicen que al leer a 0, debe escribir a 1 y moverse hacia la izquierda.
- Moviéndose a la izquierda, luego lee a 1, lo que de acuerdo con las instrucciones significa que debe escribir a 0 y moverse a la izquierda.
- Luego lee otro 1, escribe a 0 y se mueve hacia la izquierda, y finalmente lee un espacio en blanco en el cual el programa se detiene.

Esta máquina de Turing está diseñada para voltear bits. Cambia 0s por 1s y 1s por 0s.

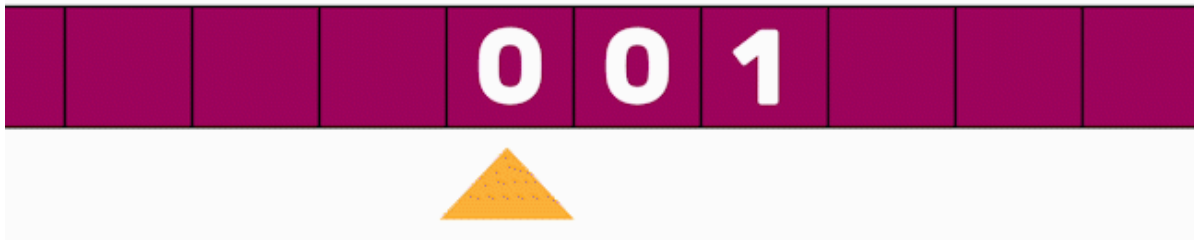
2. Computación: Historia y modelos formales

Máquina de Turing: Como construir una MT

Ejemplo #2:

Las instrucciones de la MT anterior solo tenían un estado, pero se pueden construir máquinas de Turing más complejas usando múltiples estados.

State	Symbol	Write	Move	Next state
A	Blank	None	Left	B
	0	None	Right	A
	1	None	Right	A
B	Blank	1	Right	C
	0	1	Left	C
	1	0	Left	B
C	Blank	None	Left	Halt
	0	None	Right	C
	1	None	Right	C



Pasos según la tabla:

- Comienza en el estado A, con la cabeza colocada debajo del primer símbolo.
- En el estado A, con un símbolo de 0, no hay ninguna instrucción de escritura, la cabeza simplemente debe moverse hacia la derecha y la máquina de Turing debe permanecer en el estado A.
- Luego, lee un 1, y se mueve hacia la derecha nuevamente.
- Ahora, lee un espacio en blanco, lo que significa que debe moverse hacia la izquierda, pero también cambia al estado B.
- Para el estado B, si está leyendo a 1, lo borra y escribe a 0, y luego se mueve hacia la izquierda, permaneciendo en el estado B.
- Luego, lee a 0, lo que significa que escribe a 1, se mueve hacia la izquierda y cambia al estado C.
- En el estado C, no se realizan acciones de escritura.
- La cabeza sigue moviéndose hacia la derecha hasta que finalmente llega a un espacio en blanco, momento en el cual se mueve hacia la izquierda y el programa se detiene.

2. Computación: Historia y modelos formales

Importancia de las Máquinas de Turing

- La máquina de Turing abstracta es un concepto simple, pero muy importante para la teoría de las ciencias de la computación, ya que proporciona una definición de computabilidad:

Si se puede definir una máquina de Turing para realizar una tarea, el problema es computable.
- Una máquina de Turing es una descripción de un solo algoritmo. Sin embargo, Turing fue más allá al describir una máquina universal (conocida como máquina universal de Turing) que podría abordar cualquier problema computable.
- En una máquina de este tipo, las funciones de transición, así como los datos, se almacenarían en la cinta . Con una cinta infinitamente larga y una cantidad infinita de tiempo, todos los problemas computables podrían resolverse.
- El trabajo de Turing en esta área se produjo en un momento muy importante en la historia de la informática.
- Su trabajo conduciría posteriormente al desarrollo de las computadoras tal como las conocemos hoy.

Conclusiones

1. COMPUTACION:

- Pregunta principal: ¿Que puede hacer una computadora?
- Enfoque: Investigar modelos de computación.

2. WHILE

- ✓ Estructura simple que permite resolver muchos problemas.
- ✓ Muy cercano a los típicos lenguajes de programación.

3. Maquinas de Turing

- MT son dispositivos potentes de computación, pero no tan evidentes a diseñar.
- Implementa 3 técnicas útiles:
 - ❖ Recursión: Tratar de resolver los problemas simplificándolos recursivamente.
 - ❖ Subrutinas: Tener diferentes partes de la MT para diferentes tareas.
 - ❖ Almacenamiento: Mantener un monto constante de información en la cinta.

4. MT y programas WHILE son igualmente potentes.

PREGUNTAS

Dudas y opiniones