

Universidad de Oriente Núcleo Anzoátegui

Materia: Teoría de Autómatas y Lenguajes Formales

Profesor: José Bastardo

Sección: 01

Alumno: Luis Correa CI: xx.xxx.xxx

Departamento de Computación y Sistemas

Ingeniería en Computación

MAQUINAS DE TURING Y COMPLEJIDAD

Fecha 06 – 08 - 12

**INDICE**

|  |
| --- |
| [INTRODUCCIÓN 1](#_Toc331792521)  [1.- Maquinas de Turing: 2](#_Toc331792522)  [ Defina formalmente una Maquina de Turing Determinística: 3](#_Toc331792523)  [La entrada de una máquina de Turing viene determinada por el estado actual y el símbolo leído, un par (estado, símbolo), siendo el cambio de estado, la escritura de un nuevo símbolo y el movimiento del cabezal, las acciones a tomar en función de una entrada. En el caso de que para cada par (estado, símbolo) posible exista a lo sumo una posibilidad de ejecución, se dirá que es una máquina de Turing determinista. 3](#_Toc331792524)  [ Defina formalmente una Maquina de Turing No Determinística. 3](#_Toc331792525)  [2.-Defina algoritmo, según la teoría de Autómatas: 3](#_Toc331792526)  [3.- Como se denominan los lenguajes reconocibles usando maquinas de turing: 4](#_Toc331792527)  [4.- Construya una maquina de turing determinista capaz de reconocer el lenguaje L definido sobre ∑ = {0,1} y que está formado por todas las cadenas de ∑\* que responden al siguiente patrón 1n0n, con n> 0: 4](#_Toc331792528)  [5.- Construya una Maquina de Turing determinista capaz de reconocer el lenguaje L definido sobre Σ={0,1} y que está formado por todas las cadenas de Σ\* que responden al siguiente patrón 1n 02n1n, con n>0. 4](#_Toc331792529)  [6.- Que es el Orden de ejecución de un algoritmo: 4](#_Toc331792530)  [7.- Defina problemas p y np: 5](#_Toc331792531)  [8.-¿Que es un problema NP-Completo?: 5](#_Toc331792532)  [9.- Haga una lista de 5 problemas NP-completos: 6](#_Toc331792533)  [10.- ¿P = NP?: 6](#_Toc331792534)  [BIBLIOGRAFIA 7](#_Toc331792535) |
|  |
|  |
|  |

# INTRODUCCIÓN

Los autómatas son modelos para diseñar circuitos digitales, Analizadores léxicos de un compilador. Buscar por palabras claves en un archivo o en internet, como en protocolos de comunicación.

Los autómatas son muy importantes ya que sin ellos no podría saberse los cambios de estado de una maquina.

Por ejemplo tenemos la Tesis de Church: Todo lo que es computable se puede calcular con una Máquina de Turing. Existen problemas que no son computables.

Es decir, la Teoría de los Lenguajes Formales (y de los Autómatas) permite responder a preguntas esenciales de la Informática.

## 1.- Maquinas de Turing:

Definición formal:

  Es un [modelo computacional](http://es.wikipedia.org/wiki/Modelo_computacional) que realiza una [lectura](http://es.wikipedia.org/wiki/Lectura)/[escritura](http://es.wikipedia.org/wiki/Escritura) de manera automática sobre una [entrada](http://es.wikipedia.org/wiki/Entrada) llamada cinta, generando una [salida](http://es.wikipedia.org/wiki/Salida_(inform%C3%A1tica)) en esta misma.

Este modelo está formado por un [alfabeto](http://es.wikipedia.org/wiki/Alfabeto) de entrada y uno de salida, un símbolo especial llamado blanco (normalmente *b*, *\Delta* o *0*), un conjunto de [estados](http://es.wikipedia.org/wiki/Estado_(inform%C3%A1tica)) finitos y un conjunto de transiciones entre dichos estados. Su funcionamiento se basa en una [función de transición](http://es.wikipedia.org/wiki/Funci%C3%B3n_de_transici%C3%B3n), que recibe un *estado inicial* y una [cadena de caracteres](http://es.wikipedia.org/wiki/Cadena_de_caracteres) (la cinta, la cual puede ser infinita) pertenecientes al [alfabeto](http://es.wikipedia.org/wiki/Alfabeto) de entrada. La máquina va leyendo una celda de la cinta en cada paso, borrando el símbolo en el que se encuentra posicionado su cabezal y escribiendo un nuevo símbolo perteneciente al alfabeto de salida, para luego desplazar el cabezal a la izquierda o a la derecha (solo una celda a la vez). Esto se repite según se indique en la [función de transición](http://es.wikipedia.org/wiki/Funci%C3%B3n_de_transici%C3%B3n), para finalmente detenerse en un estado final o de aceptación, representando así la salida.

Una máquina de Turing con una sola cinta puede definirse como una 7-[tupla](http://es.wikipedia.org/wiki/Tupla)

M=(Q, \Sigma, \Gamma, s, b, F, \delta),\!

Donde:

Q\! es un conjunto finito de [estados](http://es.wikipedia.org/wiki/Estado_f%C3%ADsico).

\Sigma\! es un conjunto finito de símbolos distinto del espacio en blanco, denominado alfabeto de máquina o de entrada.

\Gamma\! es un conjunto finito de símbolos de cinta, denominado alfabeto de cinta (\Sigma \subseteq\Gamma).

s \in Q es el estado inicial.

b \in \Gamma es un símbolo denominado blanco, y es el único símbolo que se puede repetir un número infinito de veces.

F \subseteq Q es el conjunto de estados finales de aceptación.

\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\} es una [función parcial](http://es.wikipedia.org/wiki/Funci%C3%B3n_parcial) denominada función de transición, donde L\! es un movimiento a la izquierda y R\! es el movimiento a la derecha.

Existen en la literatura un abundante número de definiciones alternativas, pero todas ellas tienen el mismo poder computacional, por ejemplo se puede añadir el símbolo S\! como símbolo de "no movimiento" en un paso de cómputo.

## Defina formalmente una Maquina de Turing Determinística:

## La entrada de una máquina de Turing viene determinada por el estado actual y el símbolo leído, un par (estado, símbolo), siendo el cambio de estado, la escritura de un nuevo símbolo y el movimiento del cabezal, las acciones a tomar en función de una entrada. En el caso de que para cada par (estado, símbolo) posible exista a lo sumo una posibilidad de ejecución, se dirá que es una máquina de Turing determinista.

## Defina formalmente una Maquina de Turing No Determinística.

Mientras que en el caso de que exista al menos un par (estado, símbolo) con más de una posible combinación de actuaciones se dirá que se trata de una máquina de Turing no determinista.

La función de transición \delta en el caso no determinista, queda definida como sigue:

δ :Q×Γ→P( Q×Γ×{L,R})

## 2.-Defina algoritmo, según la teoría de Autómatas:

Definición formal de algoritmo determinista:

Formalmente los algoritmos deterministas se pueden definir en términos de una [máquina de estado](http://es.wikipedia.org/wiki/M%C3%A1quina_de_estados); un «estado» describe qué está haciendo la máquina en un instante particular de tiempo. Justo cuando se produce la entrada, la máquina comienza en su «estado inicial» y, posteriormente, si la máquina es determinista, comenzará la ejecución de la secuencia de estados predeterminados. Una máquina puede ser determinista y no tener límite temporal para la ejecución o quedarse en un bucle de estados cíclicos eternamente.

Ejemplos de [máquinas abstractas](http://es.wikipedia.org/wiki/M%C3%A1quina_abstracta) deterministas son las [máquinas de Turing deterministas](http://es.wikipedia.org/wiki/M%C3%A1quina_de_Turing_determinista) y los [autómatas finitos deterministas](http://es.wikipedia.org/wiki/Aut%C3%B3mata_finito_determinista).

Definición formal de algoritmo no determinista:

En [Ciencias de la computación](http://es.wikipedia.org/wiki/Ciencias_de_la_computaci%C3%B3n), un algoritmo no determinístico es un algoritmo que con la misma entrada ofrece muchos posibles resultados. No se puede saber de antemano cuál será el resultado de la ejecución de un algoritmo no determinístico.

## 3.- Como se denominan los lenguajes reconocibles usando maquinas de turing:

Aceptan lenguajes formales que pueden ser generados por una gramática de tipo 0: recursivamente innumerable. Las maquinas de Turing son los reconocedores de lenguaje más poderosos que existen.

Un lenguaje que es aceptado por una máquina de Turing se conoce como lenguaje recursivamente enumerable. El término enumerable proviene que dichos lenguajes son aquellos cuyas cadenas pueden ser listadas (enumeradas) por una máquina de Turing. Esta clase de lenguajes es bastante grande, incluyendo los lenguajes libres de contexto.

## 4.- Construya una maquina de turing determinista capaz de reconocer el lenguaje L definido sobre ∑ = {0,1} y que está formado por todas las cadenas de ∑\* que responden al siguiente patrón 1n0n, con n> 0:

## 5.- Construya una Maquina de Turing determinista capaz de reconocer el lenguaje L definido sobre Σ={0,1} y que está formado por todas las cadenas de Σ\* que responden al siguiente patrón 1n 02n1n, con n>0.

## 6.- Que es el Orden de ejecución de un algoritmo:

El orden de ejecución será casi siempre crítico para su funcionamiento en general, se asume que las instrucciones se enumeran explícitamente, y deben ejecutarse desde arriba hacia abajo.

La solución de los más variados problemas en un agente de cómputo se reduce inevitablemente a la realización de simples operaciones aritméticas y lógicas. Sin embargo, en muchas situaciones una misma expresión matemática admite varias posibilidades para su cálculo, que difieren solamente en el orden de ejecución de las operaciones. Ahora bien, los resultados de los cálculos en un agente de cómputo dependen del orden de realización de las operaciones y la diferencia en los errores puede ser altamente significativo.

## 7.- Defina problemas p y np:

* La [clase de complejidad P](http://es.wikipedia.org/wiki/Tiempo_polin%C3%B3mico):

Es el conjunto de los problemas de decisión que pueden ser resueltos en una [máquina determinista](http://es.wikipedia.org/wiki/M%C3%A1quina_de_Turing) en tiempo polinómico, lo que corresponde intuitivamente a problemas que pueden ser resueltos aún en el peor de sus casos.

* La [clase de complejidad NP](http://es.wikipedia.org/wiki/NP_(Complejidad_computacional)):

Es el conjunto de los problemas de decisión que pueden ser resueltos por una máquina no determinista en tiempo polinómico. Esta clase contiene muchos problemas que se desean resolver en la práctica, incluyendo el [problema de satisfacibilidad booleana](http://es.wikipedia.org/wiki/Problema_de_satisfacibilidad_booleana) y el [problema del viajante](http://es.wikipedia.org/wiki/Problema_del_viajante), un camino Hamiltoniano para recorrer todos los vértices una sola vez. Todos los problemas de esta clase tienen la propiedad de que su solución puede ser verificada efectivamente.

## 8.-¿Que es un problema NP-Completo?:

En [teoría de la complejidad computacional](http://es.wikipedia.org/wiki/Complejidad_computacional), la [clase de complejidad](http://es.wikipedia.org/wiki/Clase_de_complejidad) NP-completo es el subconjunto de los [problemas de decisión](http://es.wikipedia.org/wiki/Problema_de_decisi%C3%B3n) en [NP](http://es.wikipedia.org/wiki/NP_(Complejidad_computacional)) tal que todo problema en NP se puede [reducir](http://es.wikipedia.org/wiki/Transformaci%C3%B3n_polin%C3%B3mica) en cada uno de los problemas de NP-completo. Se puede decir que los problemas de NP-completo son los problemas más difíciles de NP y muy probablemente no formen parte de la clase de complejidad [P](http://es.wikipedia.org/wiki/Tiempo_polin%C3%B3mico).

La razón es que de tenerse una solución polinómica para un problema NP-completo, todos los problemas de NP tendrían también una solución en tiempo polinómico. Si se demostrase que un problema NP-completo, llamémoslo *A*, no se pudiese resolver en tiempo polinómico, el resto de los problemas NP-completos tampoco se podrían resolver en tiempo polinómico.

Esto se debe a que si uno de los problemas NP-completos distintos de *A*, digamos *X*, se pudiese resolver en tiempo polinómico, entonces *A* se podría resolver en tiempo polinómico, por definición de NP-completo. Ahora, pueden existir problemas en *NP* y que no sean NP-completos para los cuales exista solución polinómica aún no existiendo solución para *A*.

## 9.- Haga una lista de 5 problemas NP-completos:

Note que los nombres de los problemas están escritos con letras mayúsculas y corresponden a abreviaciones del nombre en inglés, como es lo usual junto a ellos entre paréntesis se escribe la traducción del nombre en español.

EXACT COVER ([Problema de la cobertura exacta](http://es.wikipedia.org/w/index.php?title=Problema_de_la_cobertura_exacta&action=edit&redlink=1))

KNAPSACK ([Problema de la mochila](http://es.wikipedia.org/wiki/Problema_de_la_mochila))

JOB SEQUENCING ([Problema de las secuencias de trabajo](http://es.wikipedia.org/w/index.php?title=Problema_de_las_secuencias_de_trabajo&action=edit&redlink=1))

PARTITION ([Problema de la partición](http://es.wikipedia.org/wiki/Problema_de_la_partici%C3%B3n))

MAX-CUT ([Problema del corte máximo](http://es.wikipedia.org/w/index.php?title=Problema_del_corte_m%C3%A1ximo&action=edit&redlink=1))

## 10.- ¿P = NP?:

Si ya que p puede ser reconocido por un autómata finito determinista y np por uno finito no determinista y los lenguajes regulares están dentro de los libres de contexto.

# BIBLIOGRAFIA

* [http://www.sc.ehu.es/jiwhehum2/TC/temas/[2]turing.pdf](http://www.sc.ehu.es/jiwhehum2/TC/temas/%5b2%5dturing.pdf)
* <http://www.mitecnologico.com/Main/LaMaquinaDeTuring>
* [http://www.mitecnologico.com](http://www.mitecnologico.com/)
* <http://www.wikiteka.com/trabajos/algoritmo-4/>