Computación y Estructuras Discretas I

Andrés A. Aristizábal P.
aaaristizabal@icesi.edu.co

Departamento de Computación y Sistemas Inteligentes

2024-2

Agenda del día

1. **Introducción al Análisis de Complejidad Temporal**
2. **Problemas y Clases de Complejidad**
3. **Contando líneas de código**
4. **Algoritmos y complejidad**
   - Introducción
   - Análisis de complejidad temporal
   - Ejercicios
5. **Modelos de computación**
   - Introducción
   - Modelo RAM
   - Circuitos combinacionales
   - Autómatas finitos
   - Máquinas de Turing

Introducción al Análisis de Complejidad Temporal

¿Cuál sería una primera aproximación para medir el desempeño en tiempo de un algoritmo?

Introducción al Análisis de Complejidad Temporal

¿Cuál sería una primera aproximación para medir el desempeño en tiempo de un algoritmo?

- Establecer distintos tamaños de entradas y tomar el tiempo que el algoritmo se demora en resolver el problema.

Introducción al Análisis de Complejidad Temporal

¿Cuál sería una primera aproximación para medir el desempeño en tiempo de un algoritmo?

- Establecer distintos tamaños de entradas y tomar el tiempo que el algoritmo se demora en resolver el problema.
- Para obtener una idea del comportamiento temporal del algoritmo se grafica tamaño de entrada vs tiempo para la solución.

Introducción al Análisis de Complejidad Temporal

¿Cómo compararía el desempeño temporal de dos algoritmos que solucionan el mismo problema?

Introducción al Análisis de Complejidad Temporal

¿Cómo compararía el desempeño temporal de dos algoritmos que solucionan el mismo problema?

- Graficando tamaño de entradas vs tiempo

Introducción al Análisis de Complejidad Temporal

¿Cuál es la sucesión de Fibonacci?

Introducción al Análisis de Complejidad Temporal

¿Cuál es la sucesión de Fibonacci?

- Es la sucesión que comienza con los números 0 y 1 y a partir de estos, cada término es la suma de los dos anteriores.

¿Cuál es la sucesión de Fibonacci?

- Es la sucesión que comienza con los números 0 y 1 y a partir de estos, cada término es la suma de los dos anteriores.
- Los número de esta sucesión se definen mediante la siguiente ecuación de recurrencia

$$f_n = f_{n-1} + f_{n-2} \text{ para } n \geq 2 \text{ y valores iniciales } f_0 = 0 \text{ y } f_1 = 1.$$

Introducción al Análisis de Complejidad Temporal

¿De que manera podría implementar un algoritmo que calcule el n-ésimo elemento de la sucesión de Fibonacci?

Introducción al Análisis de Complejidad Temporal

¿De que manera podría implementar un algoritmo que calcule el n-ésimo elemento de la sucesión de Fibonacci?

- De manera recursiva.

Introducción al Análisis de Complejidad Temporal

¿De que manera podría implementar un algoritmo que calcule el n-ésimo elemento de la sucesión de Fibonacci?

- De manera recursiva.
- De manera iterativa.

Recursivo

```
public static long fibonacci_rec(int n) {
    if (n <= 1) return n;
    else return fibonacci_rec(n-1) + fibonacci_rec(n-2);
}
```

Iterativo

```java
public static long fibonacci_iter(int n) {
    if(n <= 1) {
        return n;
    }
    long fib = 1;
    long prevFib = 1;

    for(long i=2; i<n; i++) {
        long temp = fib;
        fib+= prevFib;
        prevFib = temp;
    }
    return fib;
}
```

Problemas y Clases de Complejidad

¿Para qué utilizamos algoritmos?

Problemas y Clases de Complejidad

¿Para qué utilizamos algoritmos?

- En ciencias de la computación utilizamos algoritmos para resolver problemas.

Problemas y Clases de Complejidad

¿Para qué utilizamos algoritmos?

- En ciencias de la computación utilizamos algoritmos para resolver problemas.
- Siempre estamos en la búsqueda de los algoritmos mas eficientes que resuelvan cierta clase de problemas.

Problemas y Clases de Complejidad

¿Cómo son estos problemas a los que nos enfrentamos?

Problemas y Clases de Complejidad

¿Cómo son estos problemas a los que nos enfrentamos?

Problemas con diversos niveles de dificultad.

Problemas y Clases de Complejidad

¿Cómo son estos problemas a los que nos enfrentamos?

Problemas con diversos niveles de dificultad.

¿Cómo podemos distinguirlos?

Problemas y Clases de Complejidad

¿Cómo son estos problemas a los que nos enfrentamos?

Problemas con diversos niveles de dificultad.

¿Cómo podemos distinguirlos?

Estableciendo un criterio objetivo que permita diferenciar y categorizar cualquier problema de acuerdo con su nivel de dificultad.

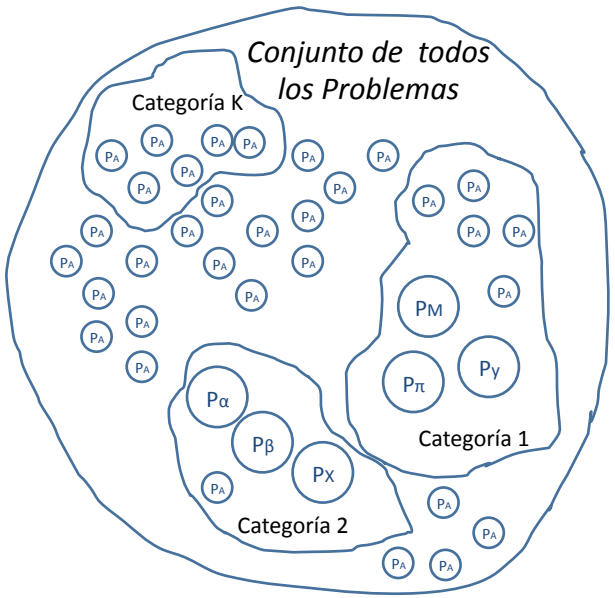
Problemas y Clases de Complejidad

¿Cómo son estos problemas a los que nos enfrentamos?

Problemas con diversos niveles de dificultad.

¿Cómo podemos distinguirlos?

Estableciendo un criterio objetivo que permita diferenciar y categorizar cualquier problema de acuerdo con su nivel de dificultad.



Problemas y Clases de Complejidad

¿Cómo se establece ese criterio?

Problemas y Clases de Complejidad

¿Cómo se establece ese criterio?

- Ese criterio de dificultad de cada problema se ha establecido como una relación directa con la complejidad temporal del algoritmo que lo resuelve.

¿Cómo se establece ese criterio?

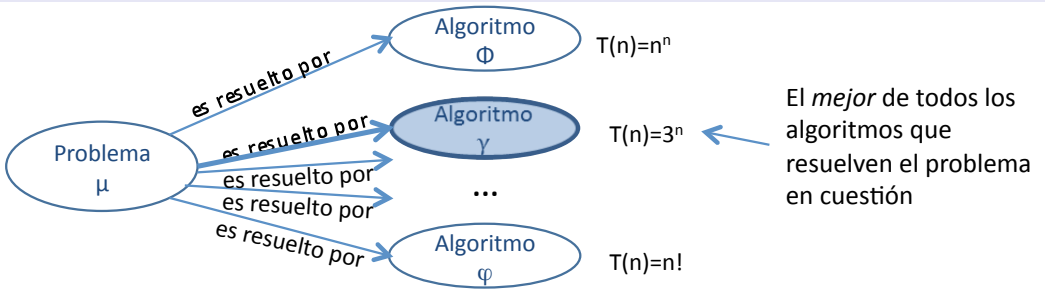
- Ese criterio de dificultad de cada problema se ha establecido como una relación directa con la complejidad temporal del algoritmo que lo resuelve.
- Existen diversos algoritmos que resuelven un mismo problema, cada uno con una complejidad temporal diferente.

Problemas y Clases de Complejidad

¿Cómo se establece ese criterio?

- Ese criterio de dificultad de cada problema se ha establecido como una relación directa con la complejidad temporal del algoritmo que lo resuelve.
- Existen diversos algoritmos que resuelven un mismo problema, cada uno con una complejidad temporal diferente.

¿cual de esas complejidades establece la complejidad del problema dado?



Problemas y Clases de Complejidad

¿Cuáles son los tipos básicos de problemas?

Problemas y Clases de Complejidad

¿Cuáles son los tipos básicos de problemas?

- Se considera entonces que el grado de dificultad para resolver un problema está dada por la complejidad del mejor de todos los algoritmos encontrados que lo resuelven.

Problemas y Clases de Complejidad

¿Cuáles son los tipos básicos de problemas?

- Se considera entonces que el grado de dificultad para resolver un problema está dada por la complejidad del mejor de todos los algoritmos encontrados que lo resuelven.
- Gran parte de los problemas que inicialmente se estudian en los cursos de algoritmos se pueden solucionar a través de algoritmos **eficientes**. A estos se les llama problemas tratables o fáciles.

Problemas y Clases de Complejidad

¿Cuáles son los tipos básicos de problemas?

- Se considera entonces que el grado de dificultad para resolver un problema está dada por la complejidad del mejor de todos los algoritmos encontrados que lo resuelven.
- Gran parte de los problemas que inicialmente se estudian en los cursos de algoritmos se pueden solucionar a través de algoritmos **eficientes**. A estos se les llama problemas tratables o fáciles.
- Sin embrago, aquellos problemas cuyo mejor algoritmo para resolverlo es **ineficiente** se les llama problemas intratables o difíciles.

Problemas y Clases de Complejidad

Ahora formalizando un poco, ¿qué es la teoría de la Complejidad Computacional?

Problemas y Clases de Complejidad

Ahora formalizando un poco, ¿qué es la teoría de la Complejidad Computacional?

Es aquella rama de la teoría de la computación que se enfoca en la clasificación de problemas computacionales de acuerdo a su dificultad inherente y en la relación que existe entre dichas clases de complejidad.

Problemas y Clases de Complejidad

Ahora formalizando un poco, ¿qué es la teoría de la Complejidad Computacional?

Es aquella rama de la teoría de la computación que se enfoca en la clasificación de problemas computacionales de acuerdo a su dificultad inherente y en la relación que existe entre dichas clases de complejidad.

¿Y qué son las clases de complejidad?

Problemas y Clases de Complejidad

Ahora formalizando un poco, ¿qué es la teoría de la Complejidad Computacional?

Es aquella rama de la teoría de la computación que se enfoca en la clasificación de problemas computacionales de acuerdo a su dificultad inherente y en la relación que existe entre dichas clases de complejidad.

¿Y qué son las clases de complejidad?

- Son las categorías en las cuales se han clasificado los problemas de acuerdo con la complejidad de los algoritmos que los resuelven de forma mas eficiente.

Problemas y Clases de Complejidad

Ahora formalizando un poco, ¿qué es la teoría de la Complejidad Computacional?

Es aquella rama de la teoría de la computación que se enfoca en la clasificación de problemas computacionales de acuerdo a su dificultad inherente y en la relación que existe entre dichas clases de complejidad.

¿Y qué son las clases de complejidad?

- Son las categorías en las cuales se han clasificado los problemas de acuerdo con la complejidad de los algoritmos que los resuelven de forma mas eficiente.
- Existen diversas clases de complejidad, incluso hay una clase para los problemas para los que NO existe un algoritmo que entregue la solución.

Problemas y Clases de Complejidad

Ahora formalizando un poco, ¿qué es la teoría de la Complejidad Computacional?

Es aquella rama de la teoría de la computación que se enfoca en la clasificación de problemas computacionales de acuerdo a su dificultad inherente y en la relación que existe entre dichas clases de complejidad.

¿Y qué son las clases de complejidad?

- Son las categorías en las cuales se han clasificado los problemas de acuerdo con la complejidad de los algoritmos que los resuelven de forma mas eficiente.
- Existen diversas clases de complejidad, incluso hay una clase para los problemas para los que NO existe un algoritmo que entregue la solución.
- A estos problemas se les llama indecidibles o irresolubles.

Problemas y Clases de Complejidad

¿Qué es una clase de complejidad P?

Problemas y Clases de Complejidad

¿Qué es una clase de complejidad P?

- Pertenecen a esta clase todos los problemas pueden ser resueltos por un algoritmo eficiente.

Problemas y Clases de Complejidad

¿Qué es una clase de complejidad P?

- Pertenecen a esta clase todos los problemas pueden ser resueltos por un algoritmo eficiente.
- Formalmente, la clase P consiste de todos aquellos problemas de decisión que pueden ser resueltos en una máquina determinista secuencial en un período de tiempo polinómico en proporción a los datos de entrada.

Problemas y Clases de Complejidad

¿Y una clase de complejidad NP?

Problemas y Clases de Complejidad

¿Y una clase de complejidad NP?

- Esta compuesta por los problemas que son verificables de manera eficiente. Es decir, si se ofrece una alternativa de solución, ésta puede ser verificada (indicar si es correcta o no) de manera eficiente.

Problemas y Clases de Complejidad

¿Y una clase de complejidad NP?

- Esta compuesta por los problemas que son verificables de manera eficiente. Es decir, si se ofrece una alternativa de solución, ésta puede ser verificada (indicar si es correcta o no) de manera eficiente.
- Primero, NP es el acrónimo en inglés de nondeterministic polynomial time, es decir, tiempo polinomial no determinista.

Problemas y Clases de Complejidad

¿Y una clase de complejidad NP?

- Esta compuesta por los problemas que son verificables de manera eficiente. Es decir, si se ofrece una alternativa de solución, ésta puede ser verificada (indicar si es correcta o no) de manera eficiente.
- Primero, NP es el acrónimo en inglés de nondeterministic polynomial time, es decir, tiempo polinomial no determinista.
- Es el conjunto de problemas que pueden ser resueltos en tiempo polinómico por una máquina de Turing no determinista.

Problemas y Clases de Complejidad

¿Y una clase de complejidad NP?

- Esta compuesta por los problemas que son verificables de manera eficiente. Es decir, si se ofrece una alternativa de solución, ésta puede ser verificada (indicar si es correcta o no) de manera eficiente.
- Primero, NP es el acrónimo en inglés de nondeterministic polynomial time, es decir, tiempo polinomial no determinista.
- Es el conjunto de problemas que pueden ser resueltos en tiempo polinómico por una máquina de Turing no determinista.

¿Qué es la clase de complejidad NP-completo?

Problemas y Clases de Complejidad

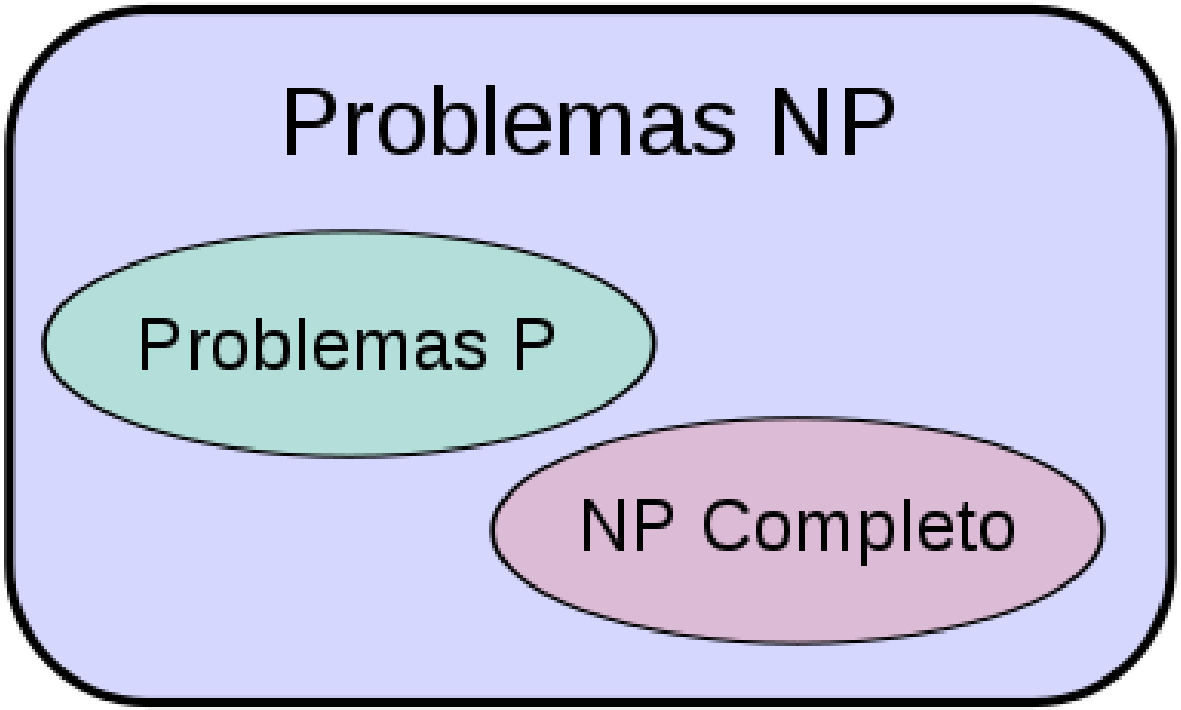
¿Y una clase de complejidad NP?

- Esta compuesta por los problemas que son verificables de manera eficiente. Es decir, si se ofrece una alternativa de solución, ésta puede ser verificada (indicar si es correcta o no) de manera eficiente.
- Primero, NP es el acrónimo en inglés de nondeterministic polynomial time, es decir, tiempo polinomial no determinista.
- Es el conjunto de problemas que pueden ser resueltos en tiempo polinómico por una máquina de Turing no determinista.

¿Qué es la clase de complejidad NP-completo?

Es el subconjunto de los problemas en NP tal que todo problema en NP se puede reducir en cada uno de los problemas de NP-completo.

Problemas y Clases de Complejidad



Problemas y Clases de Complejidad

¿Por qué resultan tan interesantes los problemas NP-completo?

¿Por qué resultan tan interesantes los problemas NP-completo?

- 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.

Problemas y Clases de Complejidad

¿Por qué resultan tan interesantes los problemas NP-completo?

- 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.

Problemas y Clases de Complejidad

¿Por qué resultan tan interesantes los problemas NP-completo?

- 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.

Problemas y Clases de Complejidad

- Varios problemas NP-completo son similares a otros problemas a los que se les conoce algoritmos eficientes para solucionarlos.

Problemas y Clases de Complejidad

- Varios problemas NP-completo son similares a otros problemas a los que se les conoce algoritmos eficientes para solucionarlos.
- Un pequeño cambio a la declaración del problema puede causar un gran cambio en lo que respecta a a la eficiencia del mejor algoritmo para solucionar dicho problema.

Problemas y Clases de Complejidad

- Varios problemas NP-completo son similares a otros problemas a los que se les conoce algoritmos eficientes para solucionarlos.
- Un pequeño cambio a la declaración del problema puede causar un gran cambio en lo que respecta a a la eficiencia del mejor algoritmo para solucionar dicho problema.
- Es importante conocer acerca de problemas NP-completo ya que, sorprendentemente, su aparición resulta frecuente en aplicaciones de la vida real.

Problemas y Clases de Complejidad

- Varios problemas NP-completo son similares a otros problemas a los que se les conoce algoritmos eficientes para solucionarlos.
- Un pequeño cambio a la declaración del problema puede causar un gran cambio en lo que respecta a a la eficiencia del mejor algoritmo para solucionar dicho problema.
- Es importante conocer acerca de problemas NP-completo ya que, sorprendentemente, su aparición resulta frecuente en aplicaciones de la vida real.
- Al poder reconocer un problema como NP-completo se puede evitar el realizar un trabajo infructuoso tratando de conseguir la mejor solución.

Problemas y Clases de Complejidad

- Varios problemas NP-completo son similares a otros problemas a los que se les conoce algoritmos eficientes para solucionarlos.
- Un pequeño cambio a la declaración del problema puede causar un gran cambio en lo que respecta a a la eficiencia del mejor algoritmo para solucionar dicho problema.
- Es importante conocer acerca de problemas NP-completo ya que, sorprendentemente, su aparición resulta frecuente en aplicaciones de la vida real.
- Al poder reconocer un problema como NP-completo se puede evitar el realizar un trabajo infructuoso tratando de conseguir la mejor solución.
- Si se prueba que el problema encontrado es NP-completo se puede dedicar el tiempo a encontrar un algoritmo que de una solución correcta, aunque no sea la óptima.

Contando líneas de código

Algoritmo sólo con estructura de control secuencial

```
//pseudocódigo

calcular(x:real)
    y ← x + x
    z ← y + y
    w ← y + z
    retorne w
```

```
//en Java

public double calcular(double x){
    double y = x + x;
    double z = y + y;
    double w = y + z;
    return w;
}
```

¿Qué valor es retornado en la variable cantidad?

```
//en Java
//contando cada línea de código
//y retornando esa cantidad
public int calcular(double x){
    int cantidad = 0;

    cantidad++;
    double y = x + x;

    cantidad++;
    double z = y + y;

    cantidad++;
    double w = y + z;

    cantidad++;
    return cantidad;
}
```

Contando líneas de código

Algoritmo sin estructura de control repetitiva

```
//pseudocódigo

cumplir(a:caracter)
    es ← falso
    si(a=‘A’)
        es ← verdadero
    retorne es
```

```
//en Java

public boolean cumplir(char a){
    boolean es = false;
    if(a==‘A’){
        es = true;
    }
    return es;
}
```

¿Qué valor es retornado en la variable cantidad?

```
//en Java
//contando cada línea de código
//y retornando esa cantidad
public int cumplir(char a){
    int cantidad = 0;

    cantidad++;
    boolean es = false;

    cantidad++;
    if(a==‘A’){
        cantidad++;
        es = true;
    }

    cantidad++;
    return cantidad;
}
```

Contando líneas de código

Algoritmo con estructura de control repetitiva

```
//pseudocódigo

acumular(d:caracter)
    c ← “”
    b ← 5
    repita mientras b>0
        c ← c + d
        b ← b - 1
    retorne c
```

```
//en Java

public String acumular(char d){
    String c = “”;
    int b = 5;
    while(b>0){
        c = c + d;
        b = b – 1;
    }
    return c;
}
```

¿Qué valor es retornado en la variable cantidad?

```
//en Java
//contando cada línea de código
//y retornando esa cantidad
public int acumular(char d){
    int cantidad = 0;

    cantidad++;
    String c = “”;

    cantidad++;
    int b = 5;

    cantidad++;
    while(b>0){
        cantidad++;
        c = c + d;

        cantidad++;
        b = b – 1;

        cantidad++;
    }

    cantidad++;
    return cantidad;
}
```

Contando líneas de código

Algoritmo con estructura de control repetitiva

```
//pseudocódigo

iterar(r:arreglo de reales)
    p ← 0
    t ← 0
    repita mientras t<r.tamaño
        p ← p + r[t]
        t ← t + 1
    p ← p / t
    retorne p
```

```
//en Java

public double iterar(double[] r){
    double p = 0;
    int t = 0;
    while(t<r.length){
        p = p + r[t];
        t = t + 1;
    }
    p = p / t;
    return p;
}
```

¿Qué valor es retornado en la variable cantidad?

```
//en Java
//contando cada línea de código
//y retornando esa cantidad
public int iterar(double[] r){
    int cantidad = 0;

    cantidad++;
    double p = 0;

    cantidad++;
    int t = 0;

    cantidad++;
    while(t<r.length){
        cantidad++;
        p = p + r[t];

        cantidad++;
        t = t + 1;

        cantidad++;
    }
    cantidad++;
    p = p / t;

    cantidad++;
    return cantidad;
}
```

Agenda del día

Objetivo

¿Cuál es el objetivo del análisis de algoritmos?

Objetivo

¿Cuál es el objetivo del análisis de algoritmos?

Comparar algoritmos que resuelven un mismo problema.

Objetivo

¿Cuál es el objetivo del análisis de algoritmos?

Comparar algoritmos que resuelven un mismo problema.

¿A través de que se pueden comparar?

Objetivo

¿Cuál es el objetivo del análisis de algoritmos?

Comparar algoritmos que resuelven un mismo problema.

¿A través de que se pueden comparar?

- Correctitud

Objetivo

¿Cuál es el objetivo del análisis de algoritmos?

Comparar algoritmos que resuelven un mismo problema.

¿A través de que se pueden comparar?

- Correctitud
- Eficiencia

Objetivo

¿Cuál es el objetivo del análisis de algoritmos?

Comparar algoritmos que resuelven un mismo problema.

¿A través de que se pueden comparar?

- Correctitud
- Eficiencia
  - Tiempo

Objetivo

¿Cuál es el objetivo del análisis de algoritmos?

Comparar algoritmos que resuelven un mismo problema.

¿A través de que se pueden comparar?

- Correctitud
- Eficiencia
  - Tiempo
  - Espacio

Objetivo

¿Cuál es el objetivo del análisis de algoritmos?

Comparar algoritmos que resuelven un mismo problema.

¿A través de que se pueden comparar?

- Correctitud
- Eficiencia
  - Tiempo
  - Espacio
- Estructuras de datos

Objetivo

¿Cuál es el objetivo del análisis de algoritmos?

Comparar algoritmos que resuelven un mismo problema.

¿A través de que se pueden comparar?

- Correctitud
- Eficiencia
  - Tiempo
  - Espacio
- Estructuras de datos
- El tipo y número de datos con los que se trabaja

Agenda del día

Análisis de complejidad temporal

Indique cuantas líneas de código se ejecutan en el siguiente algoritmo

```
1. s=0
2. x=1
3. while(x<=n)
4.   y=0
5.   while(y<=x)
6.     s=s+1
7.     y=y+1
8.   x=x+1
```

Agenda del día

1. Introducción al Análisis de Complejidad Temporal
2. Problemas y Clases de Complejidad
3. Contando líneas de código
4. **Algoritmos y complejidad**
   - Introducción
   - Análisis de complejidad temporal
   - Ejercicios
5. Modelos de computación
   - Introducción
   - Modelo RAM
   - Circuitos combinacionales
   - Autómatas finitos
   - Máquinas de Turing

Agenda del día

Modelos de computación

¿Qué es un modelo de computación?

Modelos de computación

¿Qué es un modelo de computación?

- Una definición formal y abstracta de un computador.

Modelos de computación

¿Qué es un modelo de computación?

- Una definición formal y abstracta de un computador.
- Un modelo abstracto que describe una forma de computar.

Modelos de computación

¿Qué es un modelo de computación?

- Una definición formal y abstracta de un computador.
- Un modelo abstracto que describe una forma de computar.
- Es la definición de un conjunto de operaciones permitidas utilizadas en el cómputo y sus respectivos costos.

Modelos de computación

¿Qué es un modelo de computación?

- Una definición formal y abstracta de un computador.
- Un modelo abstracto que describe una forma de computar.
- Es la definición de un conjunto de operaciones permitidas utilizadas en el cómputo y sus respectivos costos.
- Al asumir un cierto modelo de computación es posible analizar los recursos de cómputo requeridos, como el tiempo de ejecución o el espacio de memoria, o discutir las limitaciones de algoritmos o computadores.

Modelos de computación

¿Qué aspectos se deben tener en cuenta al momento de definir un modelo de computación?

Modelos de computación

¿Qué aspectos se deben tener en cuenta al momento de definir un modelo de computación?

- Representación de las entradas y salidas

Modelos de computación

¿Qué aspectos se deben tener en cuenta al momento de definir un modelo de computación?

- Representación de las entradas y salidas
- Operaciones elementales.

Modelos de computación

¿Qué aspectos se deben tener en cuenta al momento de definir un modelo de computación?

- Representación de las entradas y salidas
- Operaciones elementales.
- Combinación de las operaciones para el desarrollo del programa.

Modelos de computación

¿Qué modelos de computación existen?

Modelos de computación

¿Qué modelos de computación existen?

- Existe una amplia variedad de modelos de computación que difieren en el conjunto de operaciones permitidas y su costo de computación.

Modelos de computación

¿Qué modelos de computación existen?

- Existe una amplia variedad de modelos de computación que difieren en el conjunto de operaciones permitidas y su costo de computación.
- Máquinas de acceso aleatorio RAM

Modelos de computación

¿Qué modelos de computación existen?

- Existe una amplia variedad de modelos de computación que difieren en el conjunto de operaciones permitidas y su costo de computación.
- Máquinas de acceso aleatorio RAM
- Circuitos combinacionales

Modelos de computación

¿Qué modelos de computación existen?

- Existe una amplia variedad de modelos de computación que difieren en el conjunto de operaciones permitidas y su costo de computación.
- Máquinas de acceso aleatorio RAM
- Circuitos combinacionales
- Autómatas finitos

Modelos de computación

¿Qué modelos de computación existen?

- Existe una amplia variedad de modelos de computación que difieren en el conjunto de operaciones permitidas y su costo de computación.
- Máquinas de acceso aleatorio RAM
- Circuitos combinacionales
- Autómatas finitos
- Máquinas de Turing

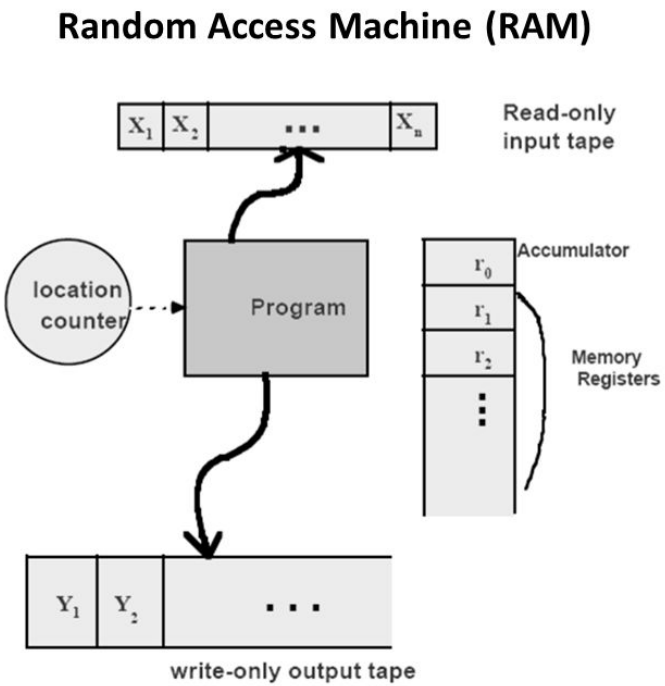
Modelos de computación

¿Qué modelos de computación existen?

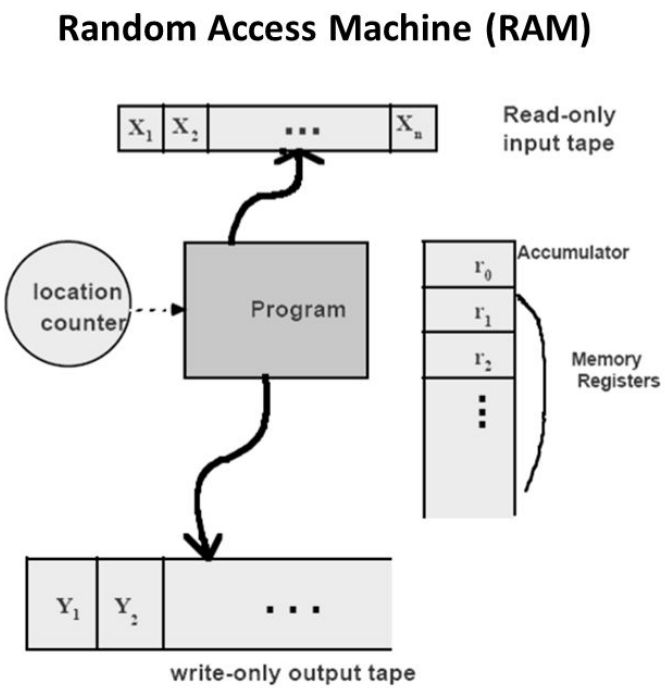
- Existe una amplia variedad de modelos de computación que difieren en el conjunto de operaciones permitidas y su costo de computación.
- Máquinas de acceso aleatorio RAM
- Circuitos combinacionales
- Autómatas finitos
- Máquinas de Turing
- ...

Agenda del día

Modelo RAM

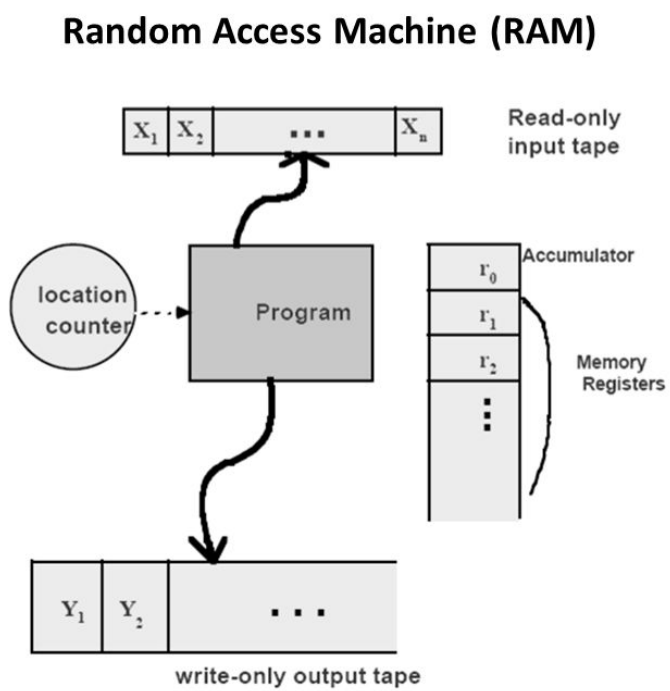

Random Access Machine (RAM)
X1 X2 ... Xn
Read-only input tape
location counter
Program
r0
r1
r2
Accumulator
Memory Registers
Y1 Y2 ...
write-only output tape

Modelo RAM



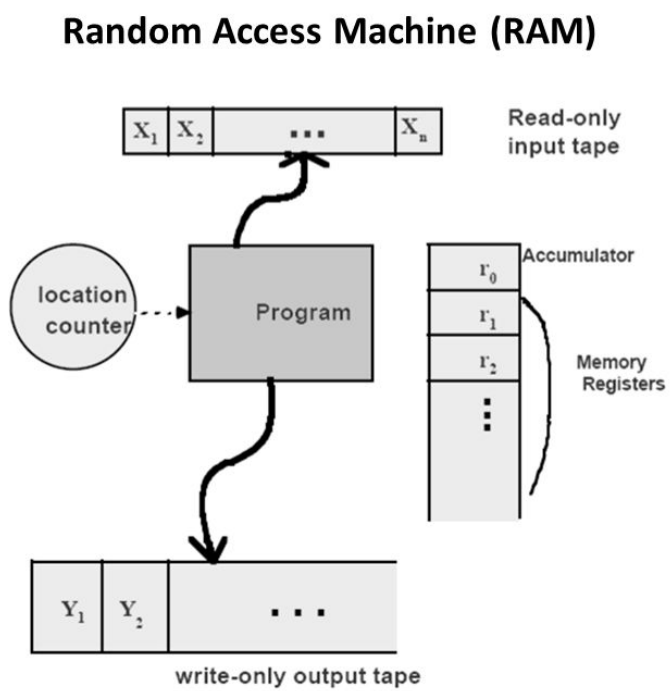
- Es un modelo simple de como los computadores se desempeñan.

Modelo RAM



- Es un modelo simple de como los computadores se desempeñan.
- Bajo el modelo RAM se mide el tiempo de ejecución de un algoritmo al contar la cantidad de pasos que se toma para una instancia de problema dada.

Modelo RAM



- Es un modelo simple de como los computadores se desempeñan.
- Bajo el modelo RAM se mide el tiempo de ejecución de un algoritmo al contar la cantidad de pasos que se toma para una instancia de problema dada.
- Está formado por una cinta de entrada, una de salida, un conjunto de registros y un programa (secuencia de instrucciones).

Modelos RAM

¿Qué se debe tener en cuenta cuando nos encontramos bajo el modelo RAM?

Modelos RAM

¿Qué se debe tener en cuenta cuando nos encontramos bajo el modelo RAM?

- Cada operación simple solo se toma un paso.

Modelos RAM

¿Qué se debe tener en cuenta cuando nos encontramos bajo el modelo RAM?

- Cada operación simple solo se toma un paso.
- Los ciclos y las subrutinas no se consideran operaciones simples.

Modelos RAM

¿Qué se debe tener en cuenta cuando nos encontramos bajo el modelo RAM?

- Cada operación simple solo se toma un paso.
- Los ciclos y las subrutinas no se consideran operaciones simples.
- Este tipo de operaciones se consideran como una composición de operaciones de un solo paso.

Modelos RAM

¿Qué se debe tener en cuenta cuando nos encontramos bajo el modelo RAM?

- Cada operación simple solo se toma un paso.
- Los ciclos y las subrutinas no se consideran operaciones simples.
- Este tipo de operaciones se consideran como una composición de operaciones de un solo paso.
- Cada acceso a memoria toma un solo paso.

Modelos RAM

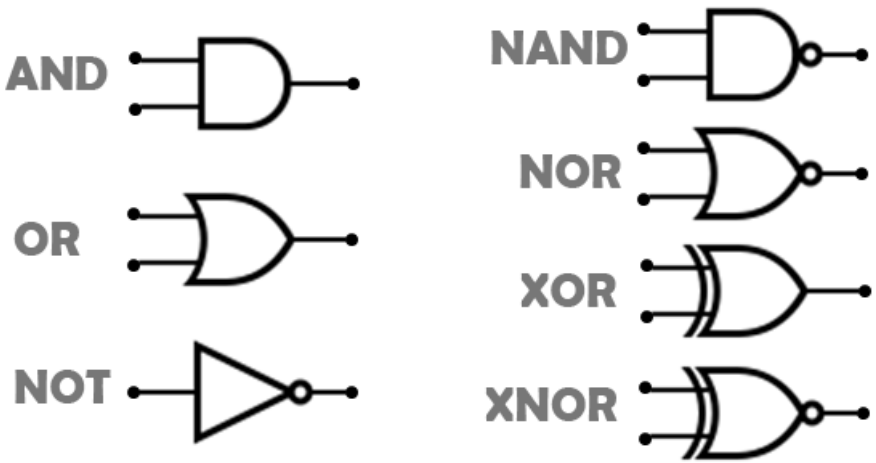
¿Qué se debe tener en cuenta cuando nos encontramos bajo el modelo RAM?

- Cada operación simple solo se toma un paso.
- Los ciclos y las subrutinas no se consideran operaciones simples.
- Este tipo de operaciones se consideran como una composición de operaciones de un solo paso.
- Cada acceso a memoria toma un solo paso.
- El modelo RAM no tiene en cuenta si un elemento esta en caché o en disco, lo cual simplifíca el análisis.

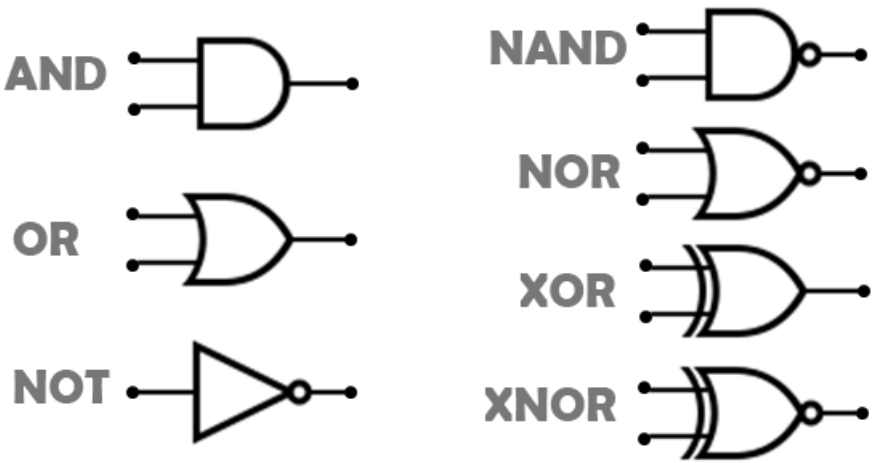
Agenda del día

1. Introducción al Análisis de Complejidad Temporal
2. Problemas y Clases de Complejidad
3. Contando líneas de código
4. Algoritmos y complejidad
   - Introducción
   - Análisis de complejidad temporal
   - Ejercicios
5. **Modelos de computación**
   - Introducción
   - Modelo RAM
   - Circuitos combinacionales
   - Autómatas finitos
   - Máquinas de Turing

Circuitos combinacionales

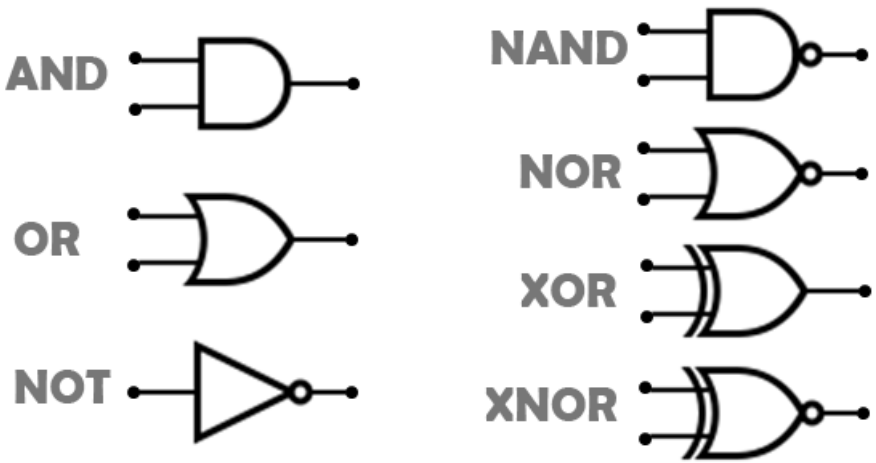


Circuitos combinacionales



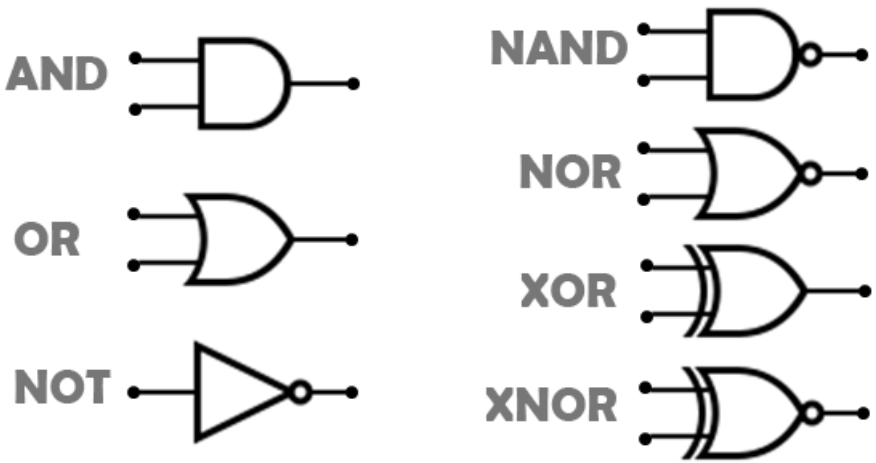
- Entradas: codificación binaria.

Circuitos combinacionales



- Entradas: codificación binaria.
- Salidas: codificación binaria.

Circuitos combinacionales

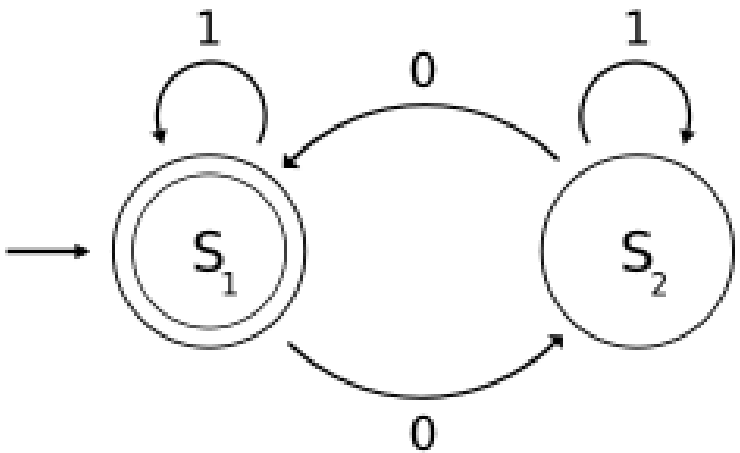


- Entradas: codificación binaria.
- Salidas: codificación binaria.
- Operaciones elementales: compuertas lógicas

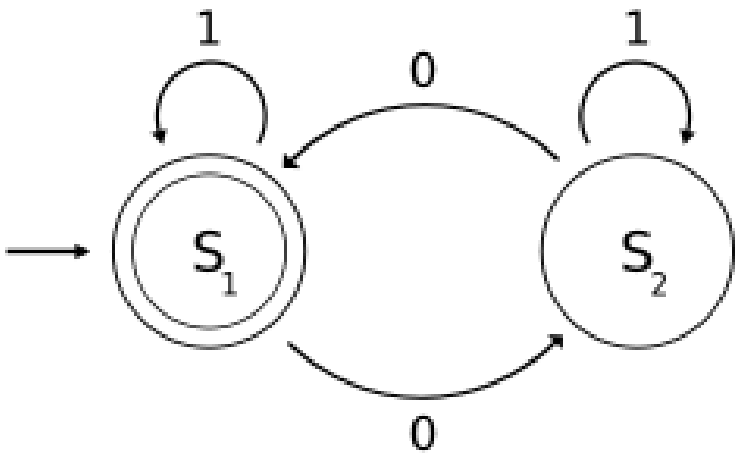
Agenda del día

1. Introducción al Análisis de Complejidad Temporal
2. Problemas y Clases de Complejidad
3. Contando líneas de código
4. Algoritmos y complejidad
   - Introducción
   - Análisis de complejidad temporal
   - Ejercicios
5. **Modelos de computación**
   - Introducción
   - Modelo RAM
   - Circuitos combinacionales
   - Autómatas finitos
   - Máquinas de Turing

Autómatas finitos

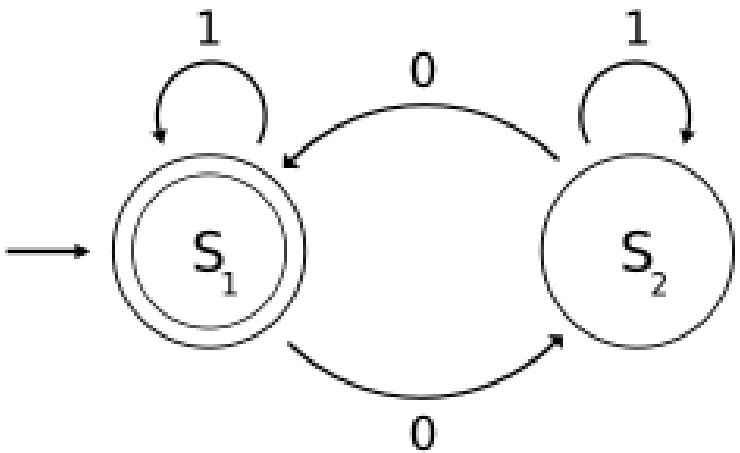


Autómatas finitos



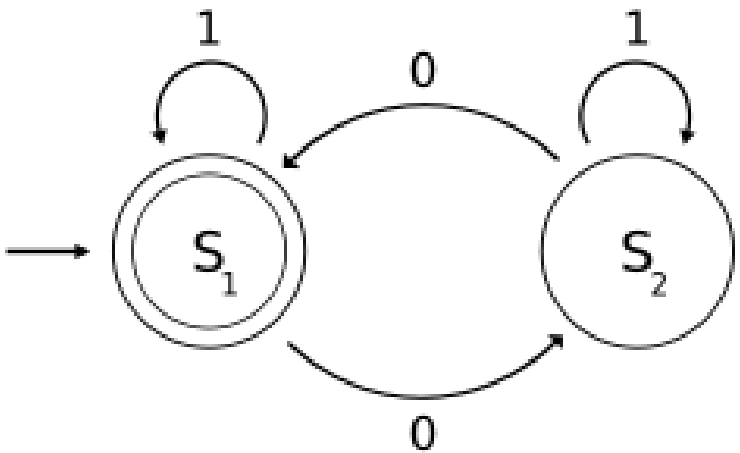
- Procesan cadenas de entrada, las cuales son aceptadas o rechazadas.

Autómatas finitos



- Procesan cadenas de entrada, las cuales son aceptadas o rechazadas.
- Lee símbolos escritos sobre una cinta semi infinita, dividida en celdas, sobre la cual se escribe una cadena de entrada.

Autómatas finitos

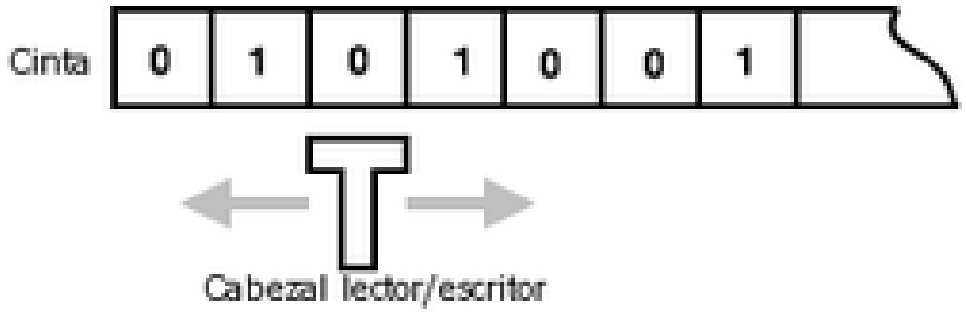


- Procesan cadenas de entrada, las cuales son aceptadas o rechazadas.
- Lee símbolos escritos sobre una cinta semi infinita, dividida en celdas, sobre la cual se escribe una cadena de entrada.
- Posee una cabeza lectora que contiene configuraciones internas llamadas estados.

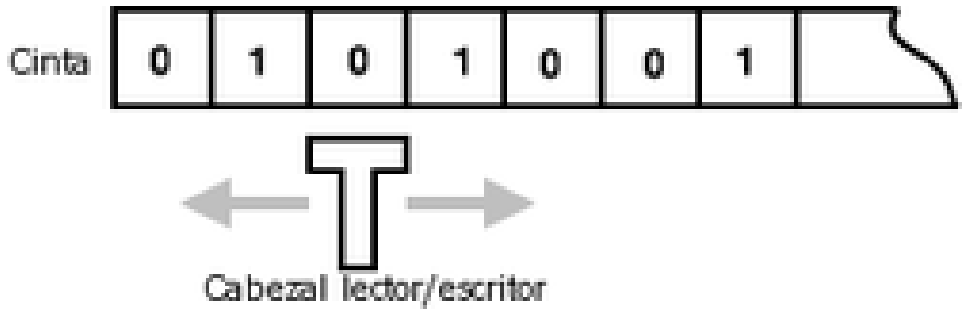
Agenda del día

1. Introducción al Análisis de Complejidad Temporal
2. Problemas y Clases de Complejidad
3. Contando líneas de código
4. Algoritmos y complejidad
   - Introducción
   - Análisis de complejidad temporal
   - Ejercicios
5. **Modelos de computación**
   - Introducción
   - Modelo RAM
   - Circuitos combinacionales
   - Autómatas finitos
   - Máquinas de Turing

Máquinas de Turing

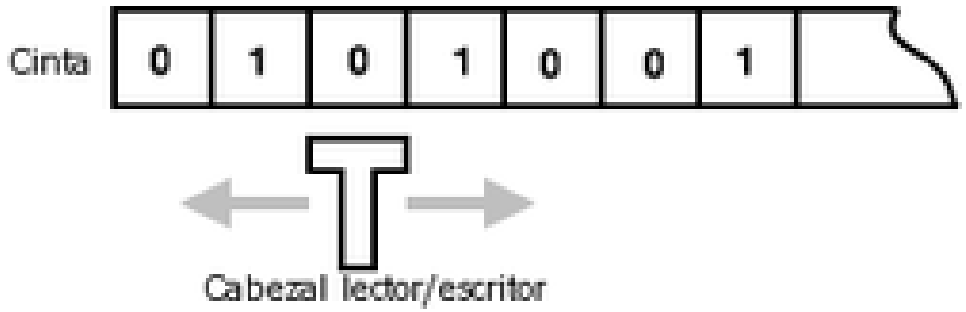


Máquinas de Turing



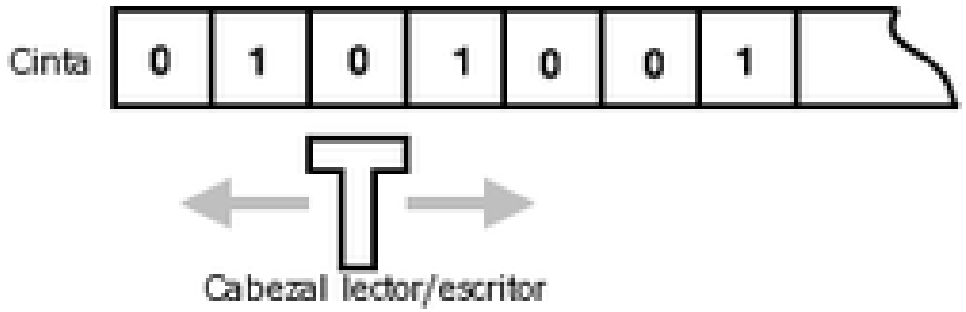
- Es el modelo de autómata con máxima capacidad computacional.

Máquinas de Turing



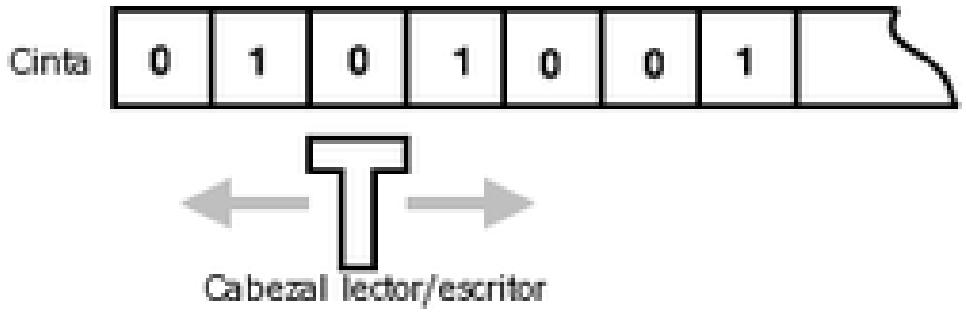
- Es el modelo de autómata con máxima capacidad computacional.
- Entradas: cinta sin fin formada por celdas que almacenan símbolos

Máquinas de Turing



- Es el modelo de autómata con máxima capacidad computacional.
- Entradas: cinta sin fin formada por celdas que almacenan símbolos
- Salidas: contenido final de la cinta.

Máquinas de Turing



- Es el modelo de autómata con máxima capacidad computacional.
- Entradas: cinta sin fin formada por celdas que almacenan símbolos
- Salidas: contenido final de la cinta.
- Operaciones elementales: transición de estado, leyendo un símbolo de la cinta, escribiendo un símbolo en la cinta y realizando un movimiento sobre la cinta (izquierda o derecha).