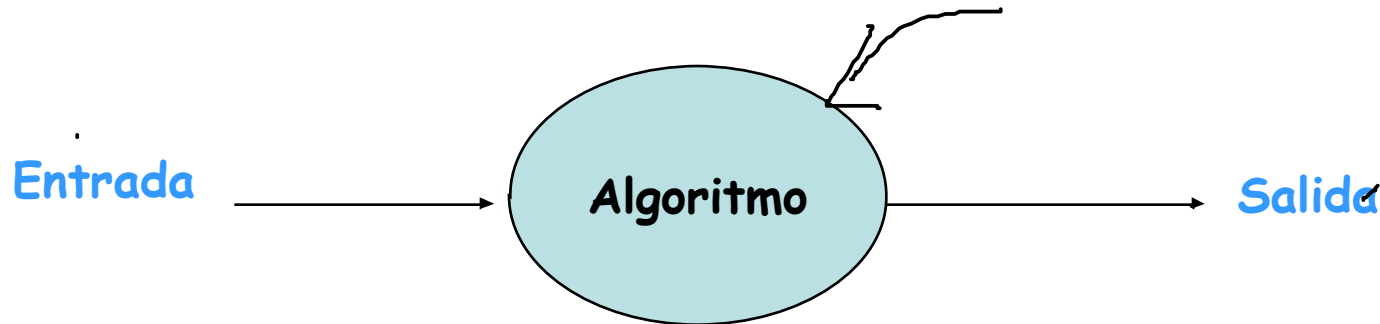


Fundamentos de análisis y diseño de algoritmos

Algoritmos en la computación

Algoritmos en la computación

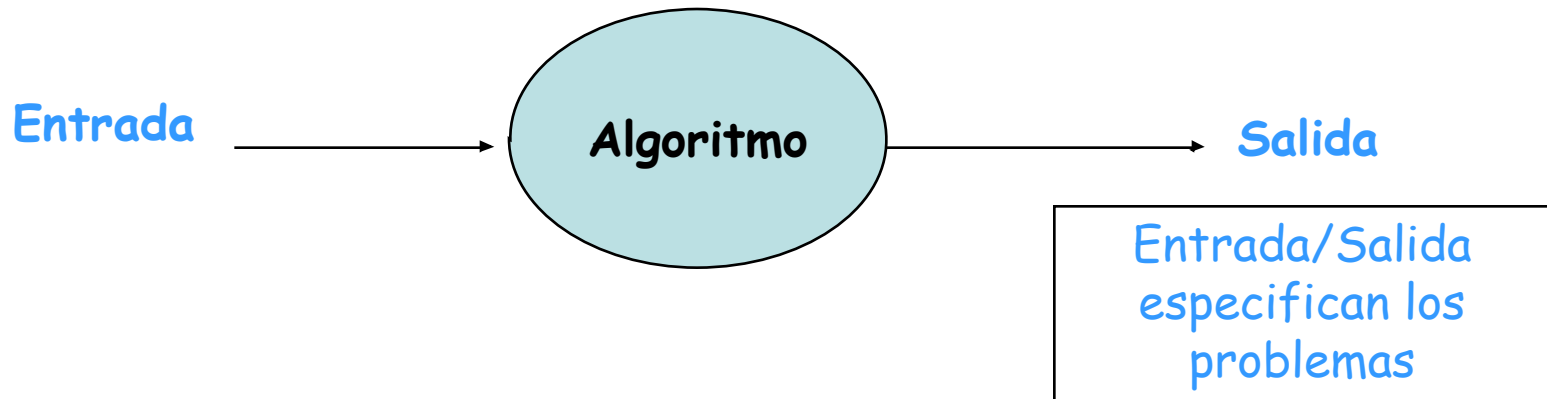
Un algoritmo es un **procedimiento computacional** que toma un valor o conjunto de valores como entrada y produce un valor o conjunto de valores como salida



"Un algoritmo se puede ver como una herramienta para resolver un problema computacional bien especificado"

Algoritmos en la computación

Un algoritmo es un **procedimiento computacional** que toma un valor o conjunto de valores como entrada y produce un valor o conjunto de valores como salida



"Un algoritmo se puede ver como una herramienta para resolver un problema computacional bien especificado"

Algoritmos en la computación

Problema 1: Cambio de monedas

Monedas

Entrada: $A = \{a_1, a_2, \dots, a_n\}$ tal que $P \in \mathbb{N}$

Salida: $A' = \langle a'_1, a'_2, \dots, a'_n \rangle$ tal que $\sum_{m \in A'} m = P$ y $|A'|$
es mínimo

Algoritmos en la computación

Problema 2: ??? $\mathcal{P}_{\text{Primos}}$
 $\Pi = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11$

Entrada: $n \in \mathbb{Z}^+$ $S = 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1$

Salida: 1, si $n=1$

1, si $n=2$

1, si $n > 2$ y $n \bmod i \neq 0 \quad \forall i \in \{2, \dots, n-1\}$

0, si $n > 2$ y $\exists x \mid n \bmod x = 0 \wedge x \in \{2, \dots, n-1\}$

$$4 \bmod 2 = 0$$

$$3 \bmod 2 = 1$$

Algoritmos en la computación

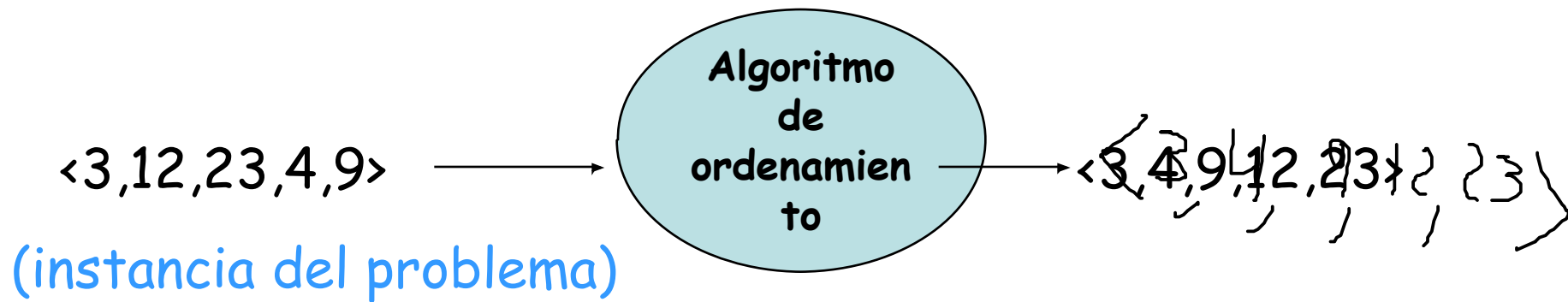
Problema 3: ??? Ordenamiento

Entrada: $S = \langle a_1, a_2, \dots, a_n \rangle$

Salida: una permutación de S , $S' = \langle a'_1, a'_2, \dots, a'_n \rangle$ tal que
 $a'_1 < a'_2 < \dots < a'_n$

Algoritmos en la computación

Instancia de un problema

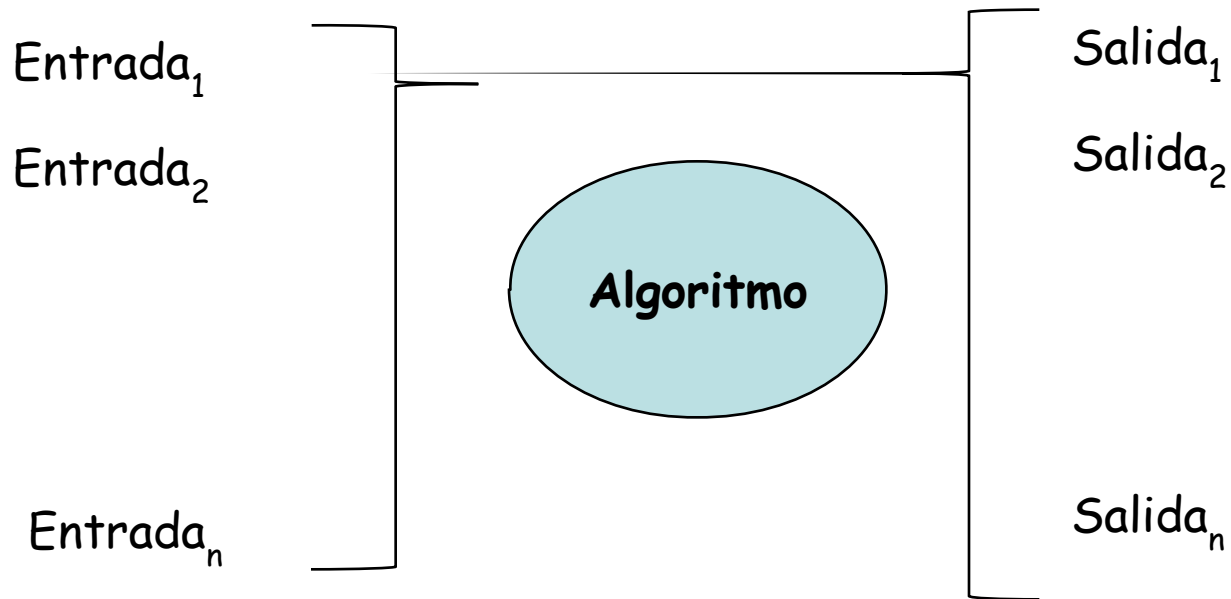


Una instancia es una entrada válida para el algoritmo

Algoritmos en la computación

Correctitud

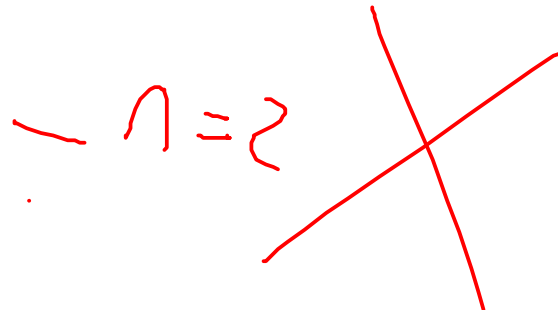
Se dice que un algoritmo es **correcto**, si para cada *instancia*, el algoritmo termina con la salida correcta



Algoritmos en la computación

```
primo(int n){  
    if (n==1)  
        return 1;  
    if (n%2==0)  
        return 0;  
    else  
        return 1;  
}
```

$n=2$



Para cada instancia, el algoritmo termina con la salida correcta

¿Es el algoritmo primo correcto?

Algoritmos en la computación

```
primo(int n){  
  if (n==1)  
    return 1;  
  else{  
    int c=0;  
    for (int i=2; i<n; i++)  
      if (n%i==0) c++;  
    if (c==0)  
      return 1;  
    else return 0;  
  }  
}
```

$$n=1, s=1$$

$$n=2, s=1$$

$$n=3, s=1$$

$$3 \bmod 2 == 0$$

$$n=4$$

$$4 \bmod 2 == 0 \checkmark \quad c=1$$

$$4 \bmod 3 == 0 \times$$

$$s=0$$

Para cada instancia, el algoritmo termina con la salida correcta

¿Es el algoritmo primo correcto?

$$\begin{array}{l}
 \text{CTH} \\
 \downarrow \\
 0
 \end{array}
 \equiv
 \left.
 \begin{array}{l}
 n \bmod 2 \\
 n \bmod 3 \\
 \vdots \\
 n \bmod (n-1)
 \end{array}
 \right\}
 \neq 0$$

$$S=1$$

Algoritmos en la computación

```
primo(int n){  
    if (n==1)  
        return 1;  
    else{  
        int c=0;  
        for (int i=3; i<n; i++)  
            if (n%i==0) c++;  
        if (c==0)  
            return 1;  
        else return 0;  
    }  
}
```

Para cada instancia, el algoritmo termina con la salida correcta

¿Es el algoritmo primo correcto?

Algoritmos en la computación

Tipos de problemas solucionados utilizando algoritmos:

- **Genoma humano:** Identificar genes en secuencias de ADN que llegan hasta los 3200 millones de pares de bases nitrogenadas (A,T,C,G). ¿Que pasaría si lo hacemos manualmente?
- **Búsquedas en Internet:** Dada la cantidad de información indexada, responder de forma correcta la solicitud de una búsqueda en Internet. ¿Que pasaría si buscamos manualmente en cada página?

Algoritmos en la computación

Tipos de problemas solucionados utilizando algoritmos:

- **Tratamiento de colisiones de objetos:** Detectar una colisión entre dos cuerpos en un espacio 3D
- **Búsquedas sobre videos:** En un biblioteca, un usuario desea encontrar todos los videos donde aparezca la mascota de Univalle (La ardilla extraña)

Algoritmos en la computación

Análisis de algoritmos

Meta: Comparar algoritmos que resuelven un mismo problema

- Correctitud
- Eficiencia
 - Tiempo
 - Espacio
- Estructuras de datos utilizadas
- Modelo computacional
- El tipo y número de datos con los cuales trabaja (escalabilidad)

Algoritmos en la computación

¿Cómo hacer análisis de algoritmos?

- Calcular tiempo de computación Complejidad computacional
 - Espacio (memoria) Complejidad especial
 - Analizar las estructuras de datos utilizadas
 - Identificar el tipo y número de datos de entrada al algoritmo
- + medidas de análisis (tiempo de ejecución)
- medidas experimentales

Algoritmos en la computación

¿Cómo hacer análisis de algoritmos?

- Calcular tiempo de computación*
 - Espacio (memoria)
 - Analizar las estructuras de datos utilizadas
 - Identificar el tipo y número de datos de entrada al algoritmo
- + medidas de análisis (tiempo de ejecución)
- medidas experimentales

Algoritmos en la computación

```
primo(int n){
```

```
  if (n==1)
```

```
    return 1;
```

```
  else{
```

```
    int c=0;
```

```
    for (int i=2; i<n; i++)
```

```
      if (n%i==0) c++;
```

```
    if (c==0)
```

```
      return 1;
```

```
    else return 0;
```

```
  }
```

¿De qué depende la cantidad de operaciones básicas que realizará el algoritmo para un llamado específico?

$n \in \mathbb{Z}^+$

Algoritmos en la computación

El tiempo de computo depende del tamaño de la entrada, los tiempos serán diferentes si se ordenan 10 números que si se ordenan 10000.

{ Además, es posible que para dos entradas de igual tamaño, el tiempo sea diferente. Esto depende, de qué tan ordenado ya se encontraba la secuencia de entrada

Mejor caso

Caso promedio

Peor caso

Algoritmos en la computación

El tiempo de computo T de un algoritmo depende del tamaño de la entrada,

$T(n)=f(n)$, donde n es el tamaño de la entrada

n es el tamaño de la entrada -----> valor numérico

Algoritmos en la computación

El **tiempo de computo** T de un algoritmo depende del tamaño de la entrada:

$T(n)=f(n)$, donde n es el **tamaño de la entrada**

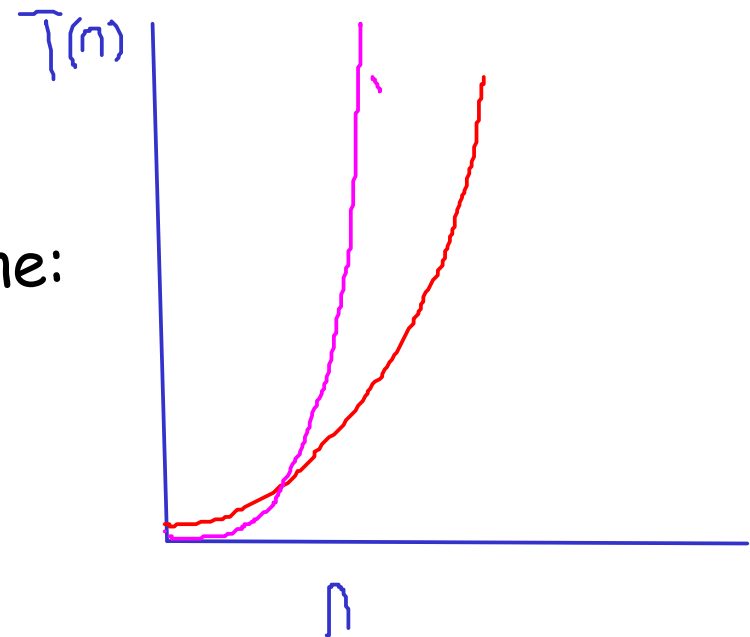
— $T_1(n)=3n^2$

— $T_2(n)=6n^3$

por ejemplo, para $n=100$, se tiene:

$$T_1(n)=3 \cdot 100^2 = 30.000$$

$$T_2(n)=6 \cdot 100^3 = 6.000.000$$



Algoritmos en la computación

El **tiempo de computo** T de un algoritmo depende del tamaño de la entrada,

$T(n)=f(n)$, donde n es el **tamaño de la entrada**

$$T_1(n)=3n^2$$

$$T_2(n)=6n^3$$

por ejemplo, para $n=100$, se tiene:

$$T_1(n)=3*100^2=30.000$$

$$T_2(n)=6*100^3=6.000.000$$

Operaciones primitivas

Pasos

Instrucciones

Algoritmos en la computación

Note que los pasos ejecutados se calculan independientemente de la máquina y de la implementación

Algoritmos en la computación

Analicemos un ejemplo (Algoritmo de ordenamiento insertion-sort)

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	5	3 5 7 9
2 do $\text{key} \leftarrow A[j]$	3	7 9
3 $i \leftarrow j-1$		
4 while $i > 0$ and $A[i] > \text{key}$		
5 do $A[i+1] \leftarrow A[i]$		
6 $i \leftarrow i-1$		
7 $A[i+1] \leftarrow \text{key}$		

¡Sin temor!, vamos a explorar este algoritmo.

Algoritmos en la computación

Este algoritmo recibe un arreglo de tamaño n y retorna el mismo arreglo ordenado de menor a mayor.

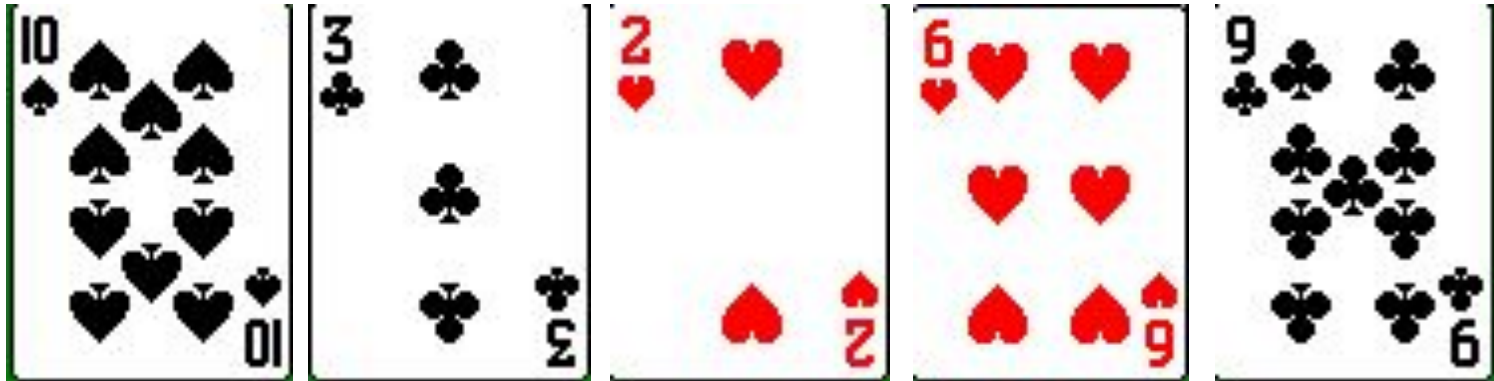
Ejemplo

Entrada = {10,3,2,6,9}

Salida = {2,3,6,9,10}

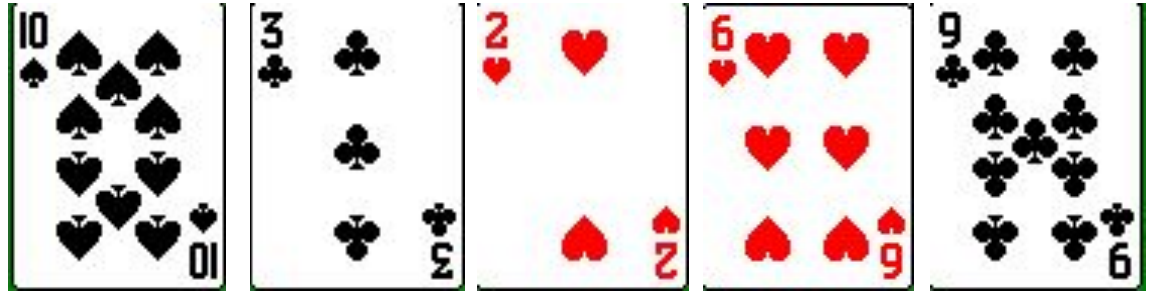
Algoritmos en la computación

Insertion sort



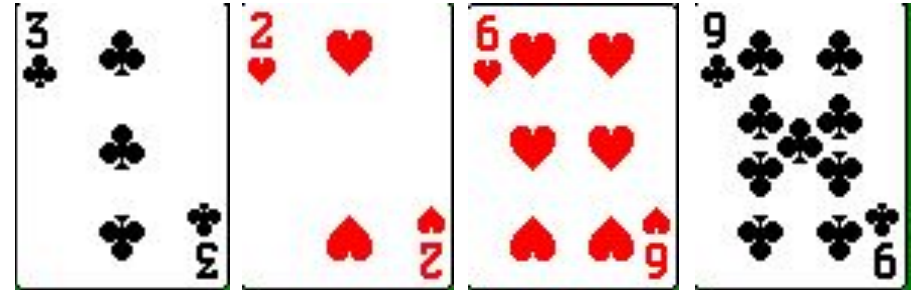
Algoritmos en la computación

Insertion sort



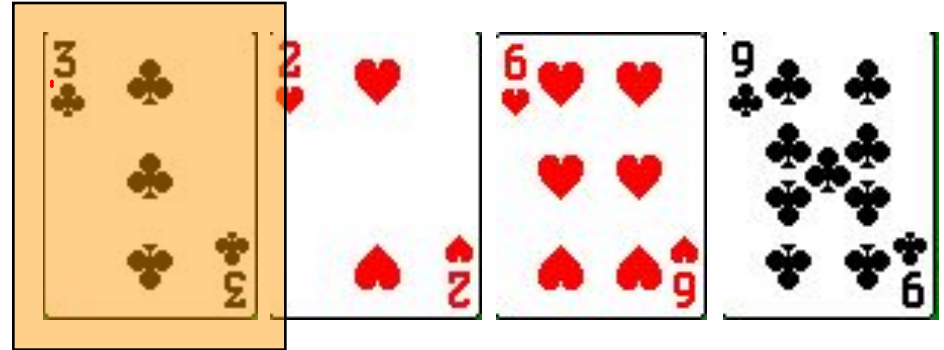
Algoritmos en la computación

Insertion sort



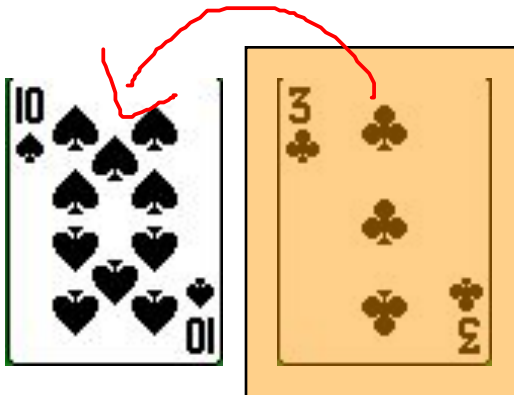
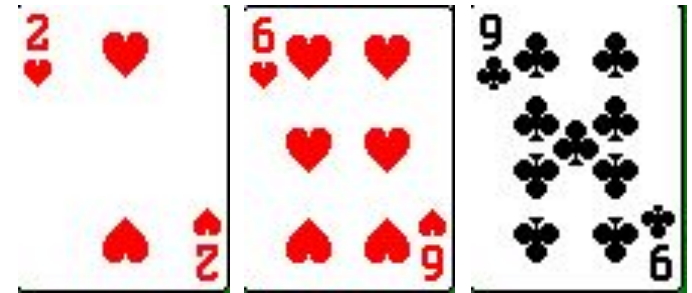
Algoritmos en la computación

Insertion sort



Algoritmos en la computación

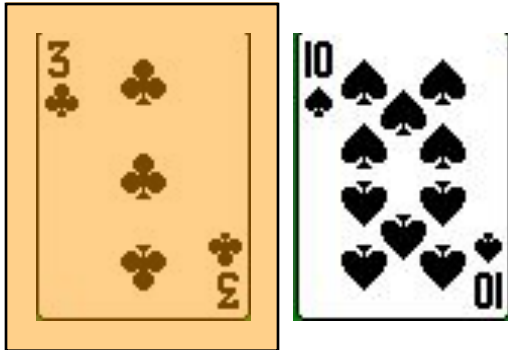
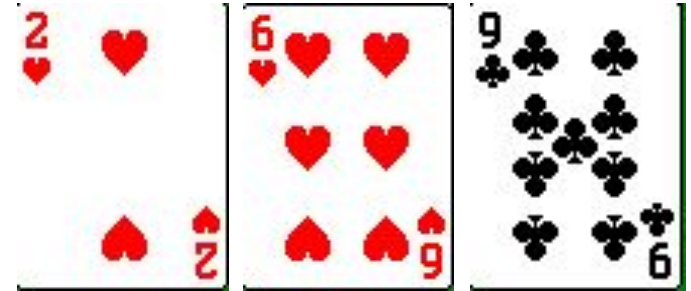
Insertion sort



Se recorre de derecha a izquierda buscando el lugar que debe ocupar

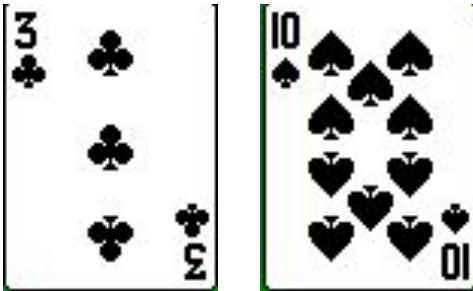
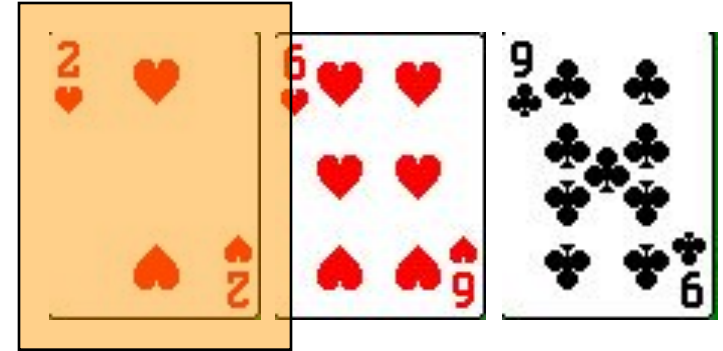
Algoritmos en la computación

Insertion sort



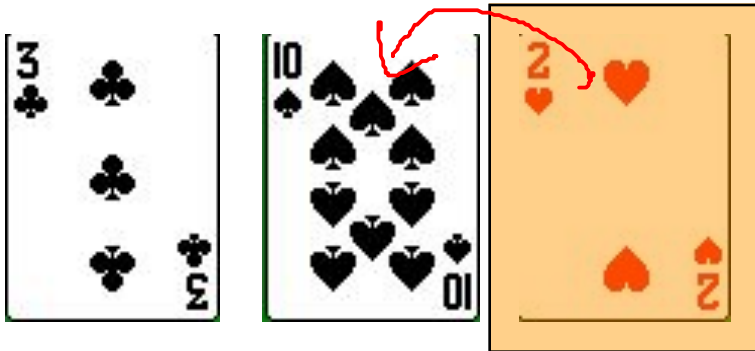
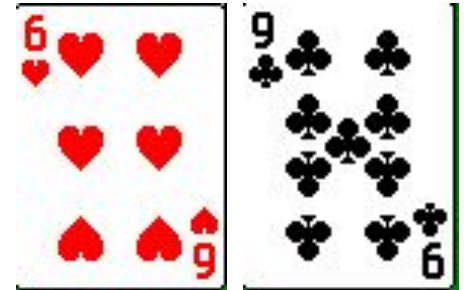
Algoritmos en la computación

Insertion sort



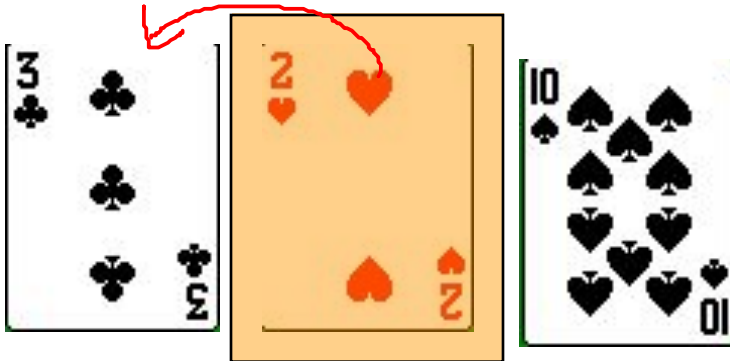
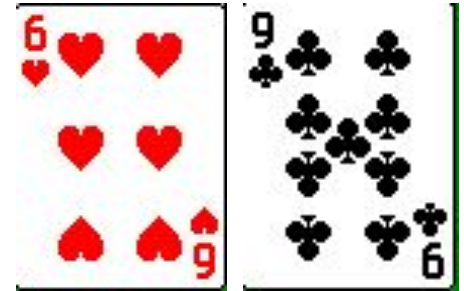
Algoritmos en la computación

Insertion sort



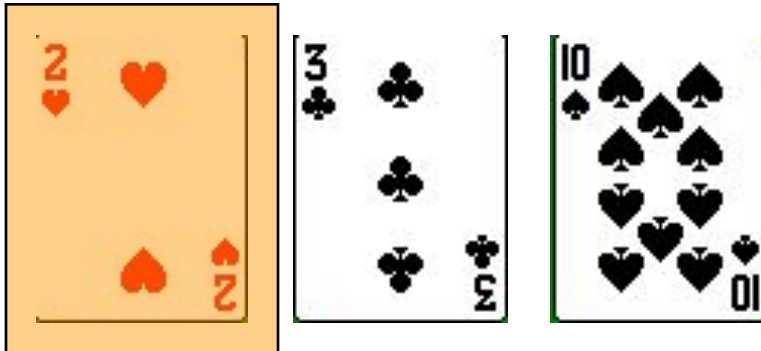
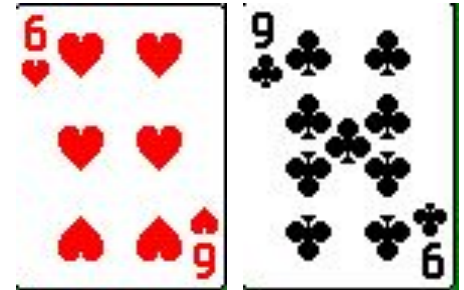
Algoritmos en la computación

Insertion sort



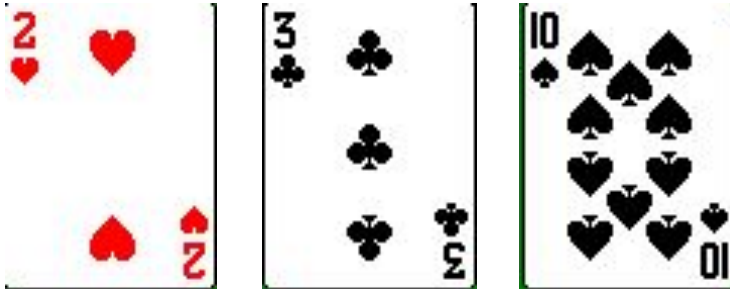
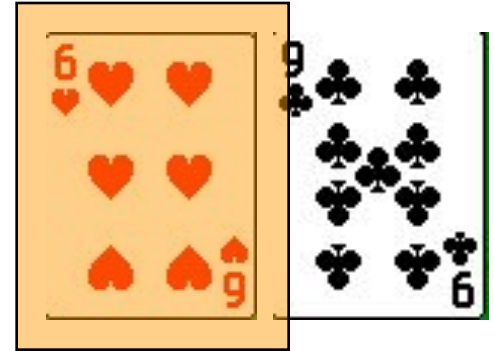
Algoritmos en la computación

Insertion sort



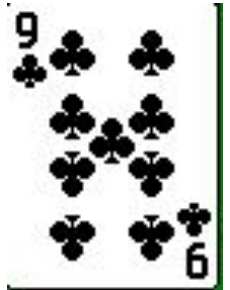
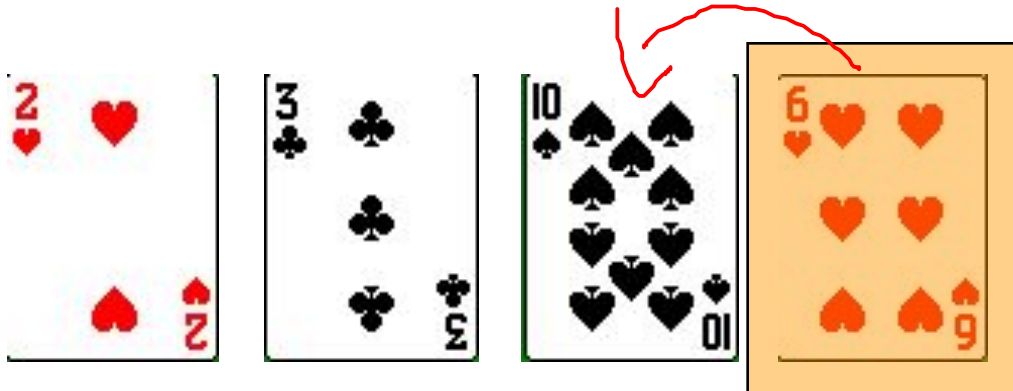
Algoritmos en la computación

Insertion sort



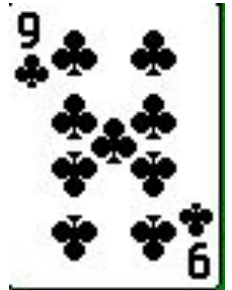
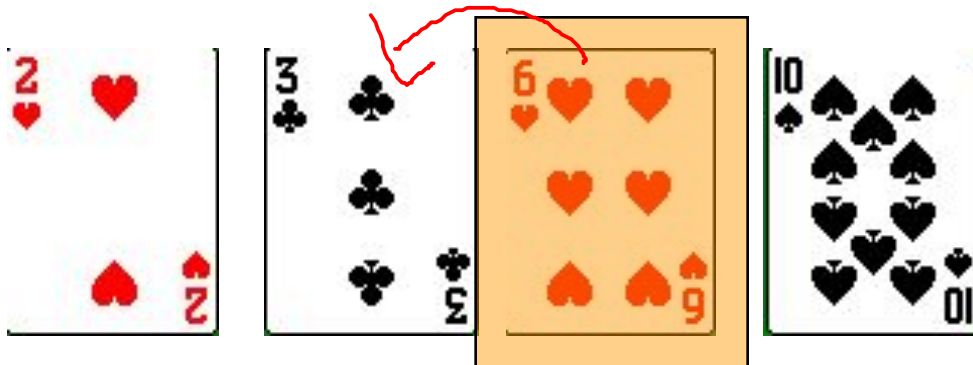
Algoritmos en la computación

Insertion sort



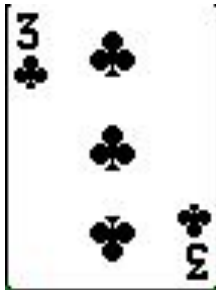
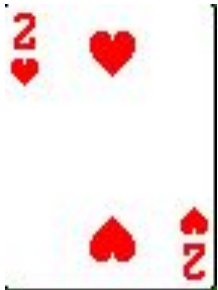
Algoritmos en la computación

Insertion sort



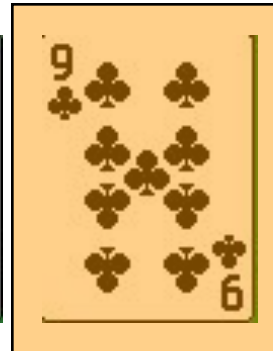
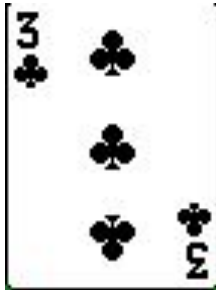
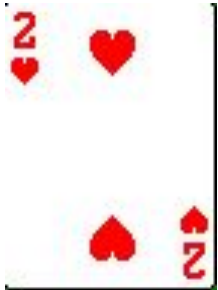
Algoritmos en la computación

Insertion sort



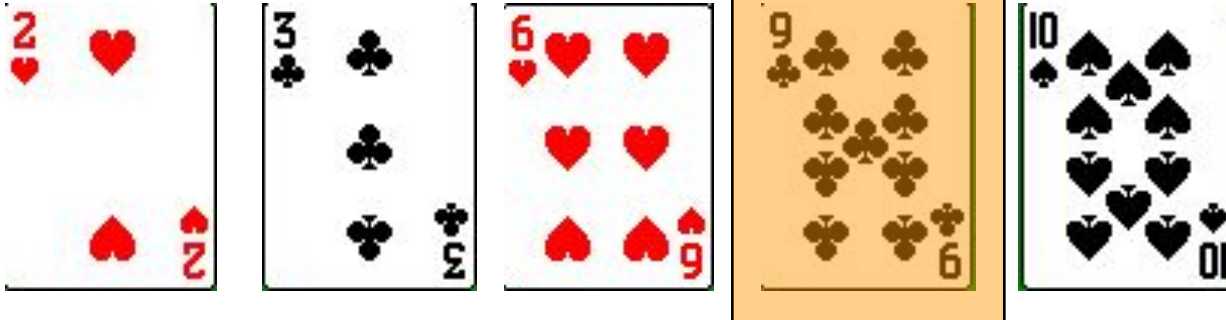
Algoritmos en la computación

Insertion sort



Algoritmos en la computación

Insertion sort



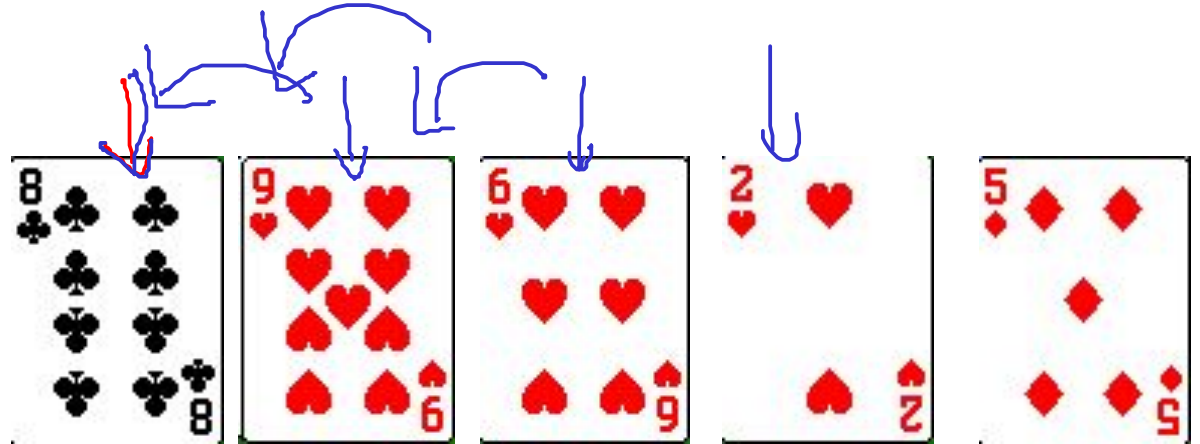
Algoritmos en la computación

Insertion sort



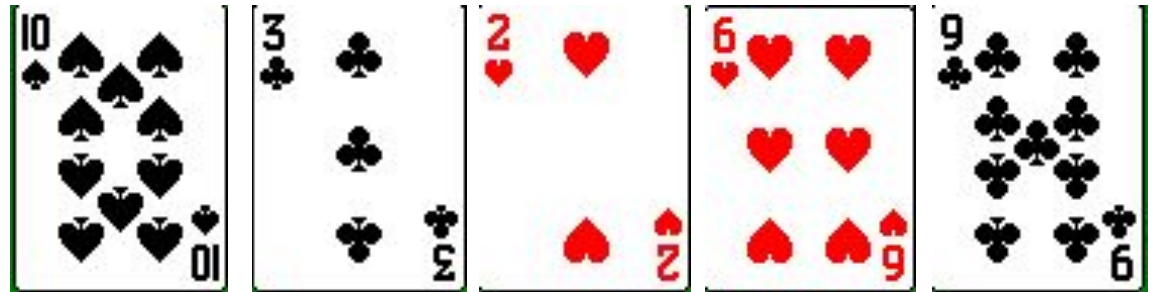
Algoritmos en la computación

Insertion sort



Algoritmos en la computación

Insertion sort



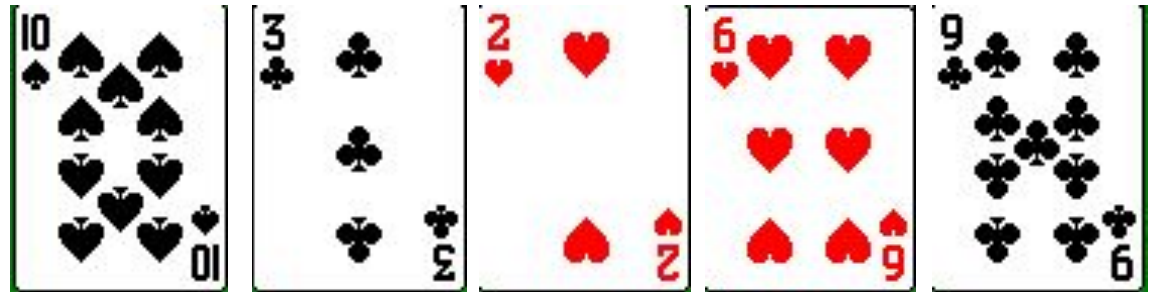
A:

10	3	2	6	9
----	---	---	---	---

Algoritmos en la computación

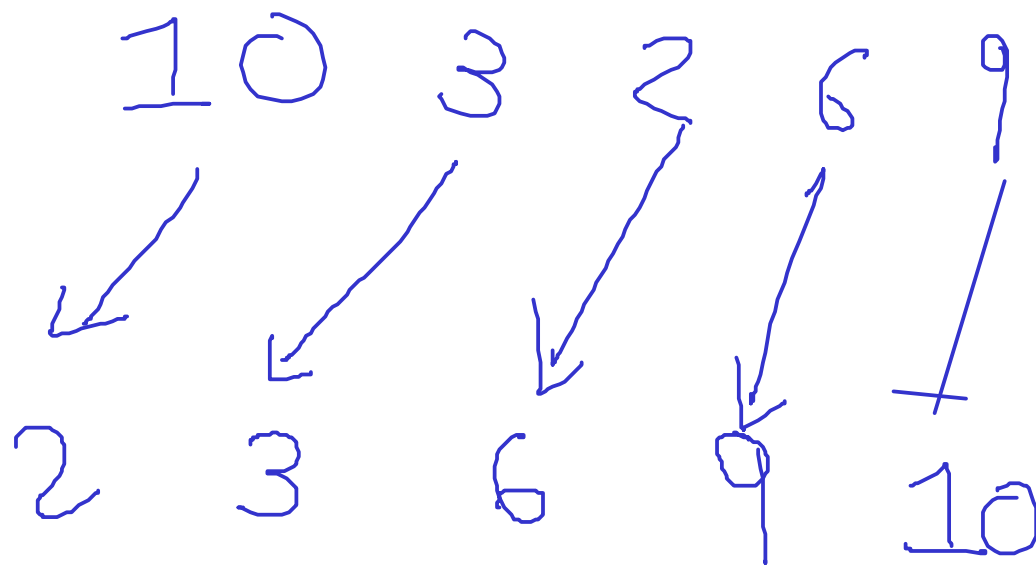
Insertion sort

Desarrolle el algoritmo
INSERTION-SORT(A)



A:

10	3	2	6	9
----	---	---	---	---



3 5 1 4 8

1 3 4 5 8

1 2 3 4 5
✓
1 2 3 4 5

5 4 3 2 1

1 2 3 4 5

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	
2 do $\text{key} \leftarrow A[j]$	c_2	
3 $i \leftarrow j-1$	c_3	
4 while $i > 0$ and $A[i] > \text{key}$	c_4	
5 do $A[i+1] \leftarrow A[i]$	c_5	
6 $i \leftarrow i-1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for j ← 1 to length[A]	c_1	$n + 1$
2 print A[j]	c_2	n

1 2 3 4 5 n $n+1$ - romper r/
ciclo

Algoritmos en la computación

for $j \leftarrow 1$ to 3 \longrightarrow for (int $j=1$; $j \leq 3$; $j++$)

$j=1$ ✓

$j=2$ ✓

$j=3$ ✓

$j=4$ ✗

Algoritmos en la computación

for $j \leftarrow 1$ to 3 \longrightarrow for (int $j=1$; $j \leq 3$; $j++$)

$j=1$ ✓

$j=2$ ✓

$j=3$ ✓

$j=4$ ✗

La cantidad de comparaciones en un for es:

cantidad de números válidos + 1

Algoritmos en la computación

for $j \leftarrow 1$ to 3 \longrightarrow for (int $j=1$; $j \leq 3$; $j++$)

$j=1$ ✓

$j=2$ ✓

$j=3$ ✓

$j=4$ ✗

Cantidad de comparaciones:

$3 + 1$

La cantidad de comparaciones en un for es:

cantidad de números válidos + 1

Algoritmos en la computación

for $j \leftarrow 1$ to n

¿Cuántas veces se repite?

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 1$ to $\text{length}[A]$	c_1	$n+1$
2 print $A[j]$	c_2	n

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to length[A]	c_1	n ?
2 print A[j]	c_2	$n - 1$?

2, 3, 4, ..., n , $n+1$ — Salida /
parada

Algoritmos en la computación

for $j \leftarrow 2$ to 4 \longrightarrow for (int $j=2$; $j \leq 4$; $j++$)

$j=2$ ✓

$j=3$ ✓

$j=4$ ✓

$j=5$ ✗

La cantidad de comparaciones en un for es:

cantidad de números válidos + 1

Algoritmos en la computación

for $j \leftarrow 2$ to 4 \longrightarrow for (int $j=2$; $j \leq 4$; $j++$)

$j=2$ ✓

$j=3$ ✓

$j=4$ ✓

$j=5$ ✗

La cantidad de comparaciones en un for es:

$$(4-2+1) + 1$$

Comparación
inicial

Comparación de
salida

Algoritmos en la computación

for $j \leftarrow 2$ to 6 \longrightarrow for (int j=2; j<=6; j++)

???

Algoritmos en la computación

for $j \leftarrow \underline{2}$ to n $n - 2 + 1$ Validos $+ 1$ Parada

La cantidad de comparaciones en el for es:

Algoritmos en la computación

for j ← 2 to n

$$n - 2 + 1$$

Período

La cantidad de comparaciones en el for es:

$$(n - 2 + 1) + 1 = n$$

Validez

② 3, 4, 5, 6 $n=6$

$$6 - 2 = 4$$

$$(6 - 2 + 1) + 1 = 6$$

For($i=x$, $i \leq n$, $i++$)

$x, x+1, x+2, \dots, n, n+1$
|
paradig

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	$n - 2 + 1 + 1 = n$
2 do $\text{key} \leftarrow A[j]$	c_2	
3 $i \leftarrow j - 1$	c_3	
4 while $i > 0$ and $A[i] > \text{key}$	c_4	
5 do $A[i+1] \leftarrow A[i]$	c_5	
6 $i \leftarrow i - 1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n - 1$
3 $i \leftarrow j - 1$	c_3	$n - 1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	
5 do $A[i+1] \leftarrow A[i]$	c_5	
6 $i \leftarrow i - 1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	???
3 $i \leftarrow j-1$	c_3	
4 while $i > 0$ and $A[i] > \text{key}$	c_4	
5 do $A[i+1] \leftarrow A[i]$	c_5	
6 $i \leftarrow i-1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	
4 while $i > 0$ and $A[i] > \text{key}$	c_4	
5 do $A[i+1] \leftarrow A[i]$	c_5	
6 $i \leftarrow i-1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	
5 do $A[i+1] \leftarrow A[i]$	c_5	
6 $i \leftarrow i-1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	
5 do $A[i+1] \leftarrow A[i]$	c_5	
6 $i \leftarrow i-1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Depende de qué tan ordenados
se encuentran los datos en A

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	
5 do $A[i+1] \leftarrow A[i]$	c_5	
6 $i \leftarrow i-1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Para cada j , se puede repetir una cantidad diferente de veces

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	
5 do $A[i+1] \leftarrow A[i]$	c_5	
6 $i \leftarrow i-1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Sea t_j , la cantidad de comparaciones que se hacen en el while para cada valor de j

Por ejemplo, t_2 , es un número que indica cuántas veces se cumple la condición cuando $j=2$

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$t_2 + t_3 + t_4 + \dots + t_n$
5 do $A[i+1] \leftarrow A[i]$	c_5	
6 $i \leftarrow i-1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Sea t_j , la cantidad de comparaciones que se hacen en el while para cada valor de j

Por ejemplo, t_2 , es un número que indica cuántas veces se cumple la condición cuando $j=2$

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$t_2 + t_3 + t_4 + \dots + t_n$
5 do $A[i+1] \leftarrow A[i]$	c_5	$(t_2-1) + (t_3-1) + (t_4-1) + \dots + (t_n-1)$
6 $i \leftarrow i-1$	c_6	
7 $A[i+1] \leftarrow \text{key}$	c_7	

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$t_2 + t_3 + t_4 + \dots + t_n$
5 do $A[i+1] \leftarrow A[i]$	c_5	$(t_2-1) + (t_3-1) + (t_4-1) + \dots + (t_n-1)$
6 $i \leftarrow i-1$	c_6	$(t_2-1) + (t_3-1) + (t_4-1) + \dots + (t_n-1)$
7 $A[i+1] \leftarrow \text{key}$	c_7	

Algoritmos en la computación

$$\sum_{j=2}^n t_j$$

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$t_2 + t_3 + t_4 + \dots + t_n$
5 do $A[i+1] \leftarrow A[i]$	c_5	$(t_2-1) + (t_3-1) + (t_4-1) + \dots + (t_n-1)$
6 $i \leftarrow i-1$	c_6	$(t_2-1) + (t_3-1) + (t_4-1) + \dots + (t_n-1)$
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$\sum_{j=2}^n t_j$
5 do $A[i+1] \leftarrow A[i]$	c_5	$\sum_{j=2}^n (t_j - 1)$
6 $i \leftarrow i-1$	c_6	$\sum_{j=2}^n (t_j - 1)$
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

$T(n) = ???$

Algoritmos en la computación

Los algoritmos debemos analizarlos, con respecto a:

- **Mejor caso:** Configuración de instancia(s) para las cuales el algoritmo realiza el menor número de pasos para dar la solución.
- **Peor caso:** Configuración de instancia(s) para las cuales el algoritmo realiza el mayor número de pasos para dar la solución.
- **Caso promedio:** Configuración típica o más frecuente de las instancias, este caso se puede analizar
 - Suponer configuraciones de instancias entre el mejor y peor caso, por ejemplo, si en el peor caso se hacen x comprobaciones y en el mejor 1 comprobación, suponer $x/2$ comprobaciones.
 - Con métodos estadísticos, para determinar la configuración esperada de las instancias

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$\sum_{j=2}^n t_j$
5 do $A[i+1] \leftarrow A[i]$	c_5	$\sum_{j=2}^n (t_j - 1)$
6 $i \leftarrow i-1$	c_6	$\sum_{j=2}^n (t_j - 1)$
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

En el mejor de los casos, cuánto vale t_j ?

Algoritmos en la computación

2	3	10	16	19
---	---	----	----	----

↑
 $j=2$
 $t_2=?$

INSERTION-SORT(A)

```
1 for  $j \leftarrow 2$  to  $\text{length}[A]$ 
2   do  $\text{key} \leftarrow A[j]$ 
3      $i \leftarrow j-1$ 
4     while  $i > 0$  and  $A[i] > \text{key}$ 
5       do  $A[i+1] \leftarrow A[i]$ 
6          $i \leftarrow i-1$ 
7      $A[i+1] \leftarrow \text{key}$ 
```

Algoritmos en la computación

2	3	10	16	19
---	---	----	----	----

↑
 $j=3$
 $t_3=?$

INSERTION-SORT(A)

```
1 for  $j \leftarrow 2$  to  $\text{length}[A]$ 
2   do  $\text{key} \leftarrow A[j]$ 
3      $i \leftarrow j-1$ 
4     while  $i > 0$  and  $A[i] > \text{key}$ 
5       do  $A[i+1] \leftarrow A[i]$ 
6          $i \leftarrow i-1$ 
7      $A[i+1] \leftarrow \text{key}$ 
```

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] < \text{key}$	c_4	$\sum_{j=2}^n t_j$
5 do $A[i+1] \leftarrow A[i]$	c_5	$\sum_{j=2}^n (t_j - 1)$
6 $i \leftarrow i-1$	c_6	$\sum_{j=2}^n (t_j - 1)$
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

En el mejor de los casos, $t_j = 1$.

$T(n) = ???$

$t_j = 1$

$j=2$ $j=3$ $j=4$... $j=n$
 1 1 1 ... 1

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$\sum_{j=2}^n 1$
5 do $A[i+1] \leftarrow A[i]$	c_5	0
6 $i \leftarrow i-1$	c_6	0
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

En el mejor de los casos, $t_j = 1$.

$T(n) = ???$


$$\sum_{j=2}^n 1$$

Algoritmos en la computación

Para solucionar este caso, recordemos la forma cerrada de la sumatoria:

$$\sum_{i=1}^n C = C * n$$

Debido a que no la tenemos en la forma cerrada, debemos convertirla:

Entonces operando tenemos:

$$\sum_{i=2}^n 1 = \sum_{i=1}^n 1 - 1$$

$$\sum_{i=2}^n 1 = n - 1$$

Handwritten expansion of the sum:

$$1 + 1 + 1 + 1 + 1 + 1 + 1$$

Indices written below the terms: 2, 3, 4, 5, 6, 7, 8

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$n-1$
5 do $A[i+1] \leftarrow A[i]$	c_5	0
6 $i \leftarrow i-1$	c_6	0
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

En el mejor de los casos, $t_j=1$.

$$T(n) = n + 4(n - 1) = 5n - 4$$

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$\sum_{j=2}^n t_j$
5 do $A[i+1] \leftarrow A[i]$	c_5	$\sum_{j=2}^n (t_j - 1)$
6 $i \leftarrow i-1$	c_6	$\sum_{j=2}^n (t_j - 1)$
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

En el peor de los casos, cuánto vale t_j ?

$$t_j = j$$

2	101	2
3	040	m
5	010	r
5	040	s
5	040	o
...		
0	0-2	c

Algoritmos en la computación

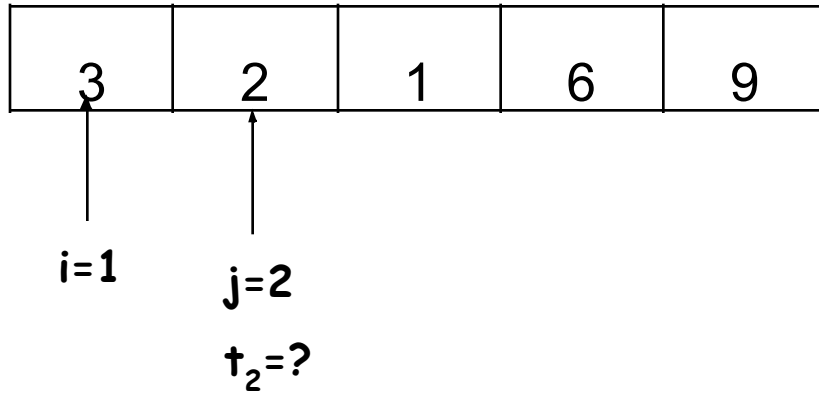
3	2	1	6	9
---	---	---	---	---

↑
 $j=2$
 $t_2=?$

INSERTION-SORT(A)

```
1 for  $j \leftarrow 2$  to  $\text{length}[A]$ 
2   do  $\text{key} \leftarrow A[j]$ 
3      $i \leftarrow j-1$ 
4     while  $i > 0$  and  $A[i] > \text{key}$ 
5       do  $A[i+1] \leftarrow A[i]$ 
6          $i \leftarrow i-1$ 
7      $A[i+1] \leftarrow \text{key}$ 
```

Algoritmos en la computación



INSERTION-SORT(A)

```
1 for j ← 2 to length[A]
2   do key ← A[j]
3     i ← j-1
4     while i > 0 and A[i] > key
5       do A[i+1] ← A[i]
6         i ← i-1
7     A[i+1] ← key
```

Algoritmos en la computación

2	3	1	6	9
---	---	---	---	---

$i=0$

$j=2$

$t_2=?$

INSERTION-SORT(A)

```
1 for  $j \leftarrow 2$  to  $\text{length}[A]$ 
2   do  $\text{key} \leftarrow A[j]$ 
3      $i \leftarrow j-1$ 
4     while  $i > 0$  and  $A[i] > \text{key}$ 
5       do  $A[i+1] \leftarrow A[i]$ 
6          $i \leftarrow i-1$ 
7      $A[i+1] \leftarrow \text{key}$ 
```

Algoritmos en la computación

2	3	1	6	9
---	---	---	---	---

↑
 $j=3$

$t_3=?$

INSERTION-SORT(A)

```
1 for  $j \leftarrow 2$  to  $\text{length}[A]$ 
2   do  $\text{key} \leftarrow A[j]$ 
3      $i \leftarrow j-1$ 
4     while  $i > 0$  and  $A[i] > \text{key}$ 
5       do  $A[i+1] \leftarrow A[i]$ 
6          $i \leftarrow i-1$ 
7      $A[i+1] \leftarrow \text{key}$ 
```


Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$\sum_{j=2}^n j$
5 do $A[i+1] \leftarrow A[i]$	c_5	$\sum_{j=2}^n (j-1)$
6 $i \leftarrow i-1$	c_6	$\sum_{j=2}^n (j-1)$
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

En el peor de los casos, $t_j = j$.

$T(n) = ???$

Algoritmos en la computación

Para solucionar este caso, recordemos la forma cerrada de la sumatoria:

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^n C = C * n$$

Debido a que no la tenemos en la forma cerrada, debemos convertirla:

$$\sum (a+b) = \sum a + \sum b$$

Entonces operando tenemos:

$$\sum_{j=2}^n 1 = \sum_{j=1}^n 1 - 1$$

$$\sum_{j=2}^n 1 = n - 1$$

$$\sum_{j=2}^n j = \frac{n(n+1)}{2} - 1$$

$$\sum_{j=2}^n (j-1) = \sum_{j=2}^n j - \sum_{j=2}^n 1$$

Dividimos la sumatoria
Aprovechamos el anterior caso

$$\sum_{j=2}^n (j-1) = \frac{n(n+1)}{2} - 1 - (n-1)$$

$$\sum_{j=2}^n j \rightarrow \underset{2}{2} + \underset{3}{3} + \underset{4}{4} + \dots + n$$

$n \leftarrow \text{index}$

$$(1 + 2 + 3 + \dots + n) - 1$$

$$\left(\sum_{j=1}^n j \right) - 1$$

$$\frac{n(n+1)}{2} - 1$$

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$\frac{n(n+1)}{2} - 1$
5 do $A[i+1] \leftarrow A[i]$	c_5	$\frac{n(n+1)}{2} - n$
6 $i \leftarrow i-1$	c_6	$\frac{n(n+1)}{2} - n$
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

En el peor de los casos, $t_j = j$.

$$T(n) = n + 3(n - 1) + 0,5 \cdot 3(n(n + 1)) - 2n - 1$$

$$T(n) = 1,5n^2 + 3,5n - 4$$

Algoritmos en la computación

Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$\sum_{j=2}^n j/2$
5 do $A[i+1] \leftarrow A[i]$	c_5	$\sum_{j=2}^n (j/2-1)$
6 $i \leftarrow i-1$	c_6	$\sum_{j=2}^n (j/2-1)$
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

En el caso promedio, se supone que se necesitan $j/2$ comparaciones, esto es, $t_j = j/2$.

$T(n) = ???$

Algoritmos en la computación

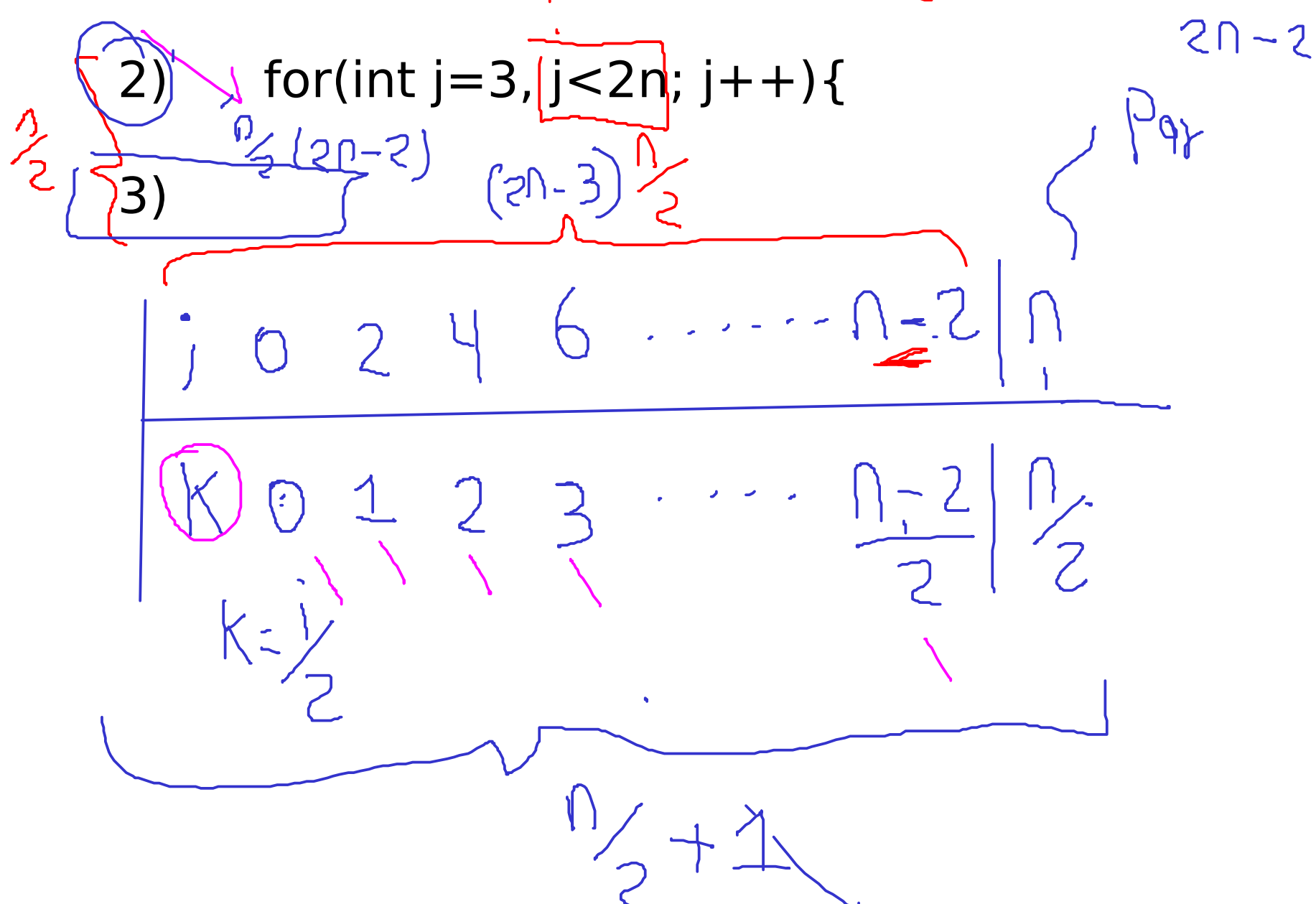
Instrucción	Costo	Veces que se repite
1 for $j \leftarrow 2$ to $\text{length}[A]$	c_1	n
2 do $\text{key} \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 while $i > 0$ and $A[i] > \text{key}$	c_4	$\frac{n(n+1)}{4} - \frac{1}{2}$
5 do $A[i+1] \leftarrow A[i]$	c_5	$\frac{n(n+1)}{4} - n$
6 $i \leftarrow i-1$	c_6	$\frac{n(n+1)}{4} - n$
7 $A[i+1] \leftarrow \text{key}$	c_7	$n-1$

En el caso promedio, se supone que se necesitan $j/2$ comparaciones, esto es, $t_j = j/2$.

$$T(n) = n + 3(n-1) + 0.25 \cdot 3(n(n+1)) - 0.5 - 2n$$

$$T(n) = 0,75n^2 + 2,75n - 3,5$$

1) for(int i=0; i<n; i+=2){ $\frac{n}{2} + 1$ $2n-1+3+2$



$t_i \rightarrow$ ejecuciones i dado

$i = 2, 4, 6, \dots, n-2$

$t_2, t_4, t_6, \dots, t_{n-2}$

k	0	1	2	3	...	$\frac{n-2}{2}$
j	t_0	t_1	t_2	t_3	...	$t_{\frac{n-2}{2}}$
	$2n-2$	$2n-2$	$2n-2$	$2n-2$...	$2n-2$

$$2n-1-3+1+1$$

$$\boxed{2n-2}$$

$$\sum_{k=0}^{\frac{n-2}{2}} 2n-2$$

$$(2n-2) + (2n-2) + (2n-2) + \dots + (2n-2)$$

0 1 2 $\frac{n-2}{2}$

$$(2n-2) + \sum_{k=1}^{\frac{n-2}{2}} (2n-2)$$

$$\sum_{i=1}^n c = cn$$

$$(2n-2) + \left(\frac{n-2}{2}\right)(2n-2)$$

$$(2n-2)\left(\frac{n}{2}-1+1\right)$$

$$(2n-2)\left(\frac{n}{2}\right)$$

$$3) (2n-3)\left(\frac{n}{2}\right)$$

Algoritmos en la computación

Calcule el tiempo de cómputo para el algoritmo

def programa1(mat1, mat2)

suponga que $\text{len}(\text{mat1})=\text{len}(\text{mat2})=n$

Instrucción	Costo
1 $i=1$	c_1
2 while $i \leq \text{len}(\text{mat1})$	c_2
3 $j \leftarrow 1$	c_3
4 while $j \leq \text{len}(\text{mat2})$	c_4
5 $\text{mat3}[i][j] \leftarrow \text{mat1}[i][j] + \text{mat2}[i][j]$	c_5
6 $j \leftarrow j+1$	c_6
7 $i \leftarrow i+1$	c_7

Instrucción		Costo
1	$i \leftarrow 1$	c_1
2	while $i \leq \text{len}(\text{mat1})$	c_2
3	$j \leftarrow 1$	c_3
4	while $j \leq \text{len}(\text{mat2})$	c_4
5	$\text{mat3}[i][j] \leftarrow \text{mat1}[i][j] + \text{mat2}[i][j]$	c_5
6	$j \leftarrow j + 1$	c_6
7	$i \leftarrow i + 1$	c_7

$\leftarrow 1$
 $\leftarrow n+1$
 $\leftarrow n$
 $\leftarrow n(n+1)$
 $\leftarrow n^2$
 $\leftarrow n^2$
 $\leftarrow n$

$i = 1, 2, 3, 4, \dots, n, n+1$

$n-1+1 \mid +1 \leftarrow n+1$

$j = 1, 2, 3, 4, \dots, n \mid n+1 \leftarrow n+1$

4) $t_1, t_2, t_3, \dots, t_n$

$\begin{matrix} n+1 & n+1 & n+1 & & n+1 \\ n & n & n & & n \end{matrix}$

$\sum_{j=1}^n n+1 = n(n+1)$
 $\sum_{j=1}^n n = n^2$

Algoritmos en la computación

Calcule el tiempo de
cómputo para el
algoritmo

def programa2(n)

i	1	2	3	4	...	n
j	1	1	1	1		1
	2	2	2	2		2
		3	3	3		...
			4	4		...
				5		n+1

i+1

1
1
n+1
n
n
t_i
t_{i-1}
t_{i-1}
n
n

Instrucción	Costo
1 $s \leftarrow 0$	c_1
2 $i \leftarrow 1$	c_2
3 while $i \leq n$	c_3
4 $t \leftarrow 0$	c_4
5 $j \leftarrow 1$	c_5
6 while $j \leq i$	c_6
7 $t \leftarrow t+1$	c_7
8 $j \leftarrow j+1$	c_8
9 $s \leftarrow s+t$	c_9
10 $i \leftarrow i+1$	c_{10}

$\frac{n(n+1)+n}{2}$
 $\frac{n(n+1)}{2}$

$$t_i = i + 1$$

$$\sum_{i=1}^n i + 1$$

$$\sum_{i=1}^n i + \sum_{i=1}^n 1 = \frac{n(n+1)}{2} + n$$

Algoritmos en la computación

Calcule el tiempo de
cómputo para el
algoritmo

def programa3(n)

Instrucción	Costo
1 $i \leftarrow 1$	c_1
2 while $i \leq n$ $1 \dots n, n+1$	c_2
3 $k \leftarrow i$	c_3
4 while $k \leq n$	c_4
5 $k \leftarrow k+1$	c_5
6 $k \leftarrow 1$	c_6
7 while $k \leq i$	c_7
8 $k \leftarrow k+1$	c_8
9 $i \leftarrow i+1$	c_9

i	1	2	3	...	n
4	1	2	3		
	⋮	⋮	⋮		
	n	n	n		
	n+1	n+1	n+1		

$n+1$ n $n-1$

$n-i+2$

Instrucción	Costo
1 $i \leftarrow 1$	c_1
2 while $i \leq n$ 1...n, n+1	c_2
3 $k \leftarrow i$	c_3
4 while $k \leq n$ 1...n, n+1	c_4
5 $k \leftarrow k+1$	c_5
6 $k \leftarrow 1$	c_6
7 while $k \leq i$	c_7
8 $k \leftarrow k+1$	c_8
9 $i \leftarrow i+1$	c_9

$$\sum_{i=1}^n (n-i+2) = n^2 - \frac{n(n+1)}{2} + 2n$$

$$n-i+1 = n^2 - \frac{n(n+1)}{2} + n$$

1	2	3	n
<u>1</u>	1	1	1
2	2	2	2
	<u>3</u>	3	3
		<u>4</u>	⋮
			n
			<u>n+1</u>

2 3 4 n+1

 i+1

$$\sum_{i=1}^n i + 1 = \frac{n(n+1)}{2} + n$$

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}$$

Algoritmos en la computación

Calcule el tiempo de
cómputo para el
algoritmo

def programa4(n)

Instrucción	Costo
1 $i \leftarrow 1$	c_1
2 while $i \leq n$	c_2
3 $k \leftarrow i$	c_3
4 while $k \leq n$	c_4
5 $k \leftarrow k+2$	c_5
6 $k \leftarrow 1$	c_6
7 while $k \leq i$	c_7
8 $k \leftarrow k+1$	c_8
9 $i \leftarrow i+2$	c_9

1
 $\frac{n}{2} + 1$
 $\frac{n}{2}$
 $\frac{n}{2}$

1 3 5 7 9 ... n | $n+2$
 $\frac{n}{2} + 1$
 $\frac{n}{2}$

Instrucción	Costo
1 $i \leftarrow 1$	c_1
2 while $i \leq n$	c_2
3 $k \leftarrow i$	c_3
4 while $k \leq n$	c_4
5 $k \leftarrow k+2$	c_5
6 $k \leftarrow 1$	c_6
7 while $k \leq i$	c_7
8 $k \leftarrow k+1$	c_8
9 $i \leftarrow i+2$	c_9

k	1	2	3	...	$\frac{n}{2}$
i	1	3	5	...	n
	1	3	5	...	
	3	5	
	5	
	
	n	n	n	...	
	$n+2$	$n+2$	$n+2$...	

$$K = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$x = \frac{n}{2}$$

$$\frac{n}{2} + 1 \quad \frac{n}{2} \quad \frac{n}{2} - 1$$

$$\left\lceil \frac{n}{2} \right\rceil \quad \frac{n}{2} - k + 2$$

$$\sum_{k=1}^{\left\lceil \frac{n}{2} \right\rceil} \frac{n}{2} - k + 2 = \left\lceil \frac{n}{2} \right\rceil \left(\frac{n}{2} - 2 \right) = \left\lceil \frac{\frac{n}{2} \left(\frac{n}{2} + 1 \right)}{2} \right\rceil$$

$$k = 1$$

Instrucción	Costo
1 $i \leftarrow 1$	c_1
2 while $i \leq n$	c_2
3 $k \leftarrow i$	c_3
4 while $k \leq n$	c_4
5 $k \leftarrow k+2$	c_5
6 $k \leftarrow 1$	c_6
7 while $k \leq i$	c_7
8 $k \leftarrow k+1$	c_8
9 $i \leftarrow i+2$	c_9

$k: 1, 2, 3, 4, \dots, \lceil \frac{n}{2} \rceil$

1	1	3	5	7	n
	1	1	1	1	1
	$\frac{1}{2}$	2	2	\vdots	\vdots
		3	3	\vdots	\vdots
		4	5	7	\vdots
				8	$n+1$
					$n+1$
	2	4	6	8	$n+1$

\sim impressions. done

$$\sum_{k=1}^{\lceil \frac{n}{2} \rceil} 2k = \frac{2 \lceil \frac{n}{2} \rceil \times (\lceil \frac{n}{2} \rceil + 1)}{2} = 2 \lceil \frac{n}{2} \rceil$$

$$\sum_{k=1}^{\lceil \frac{n}{2} \rceil} (2k-1) = \lceil \frac{n}{2} \rceil$$

Algoritmos en la computación

Calcule el tiempo de
cómputo para el
algoritmo

def programa5(n)

Instrucción	Costo
1 $i \leftarrow 1$	c_1
2 while $i \leq n$	c_2
3 $k \leftarrow i$	c_3
4 while $k \leq n$	c_4
5 $k \leftarrow k+2$	c_5
6 $k \leftarrow 1$	c_6
7 while $k \leq i$	c_7
8 $k \leftarrow k+1$	c_8
9 $i \leftarrow 2i$	c_9

$i = 1, 2, 4, 8, 16$

$k = 2, 2, 2, 2, 2$

$$k = \log_2(i)$$

$$i \leftarrow i^2 \quad i = 2^{\log(\log(n))}$$

Instrucción	Costo
1 $i \leftarrow 1$	c_1
2 while $i \leq n$	c_2
3 $k \leftarrow i$	c_3
4 while $k \leq n$	c_4
5 $k \leftarrow k+2$	c_5
6 $k \leftarrow 1$	c_6
7 while $k \leq i$	c_7
8 $k \leftarrow k+1$	c_8
9 $i \leftarrow 2i$	c_9

$$n = 2^k$$

1

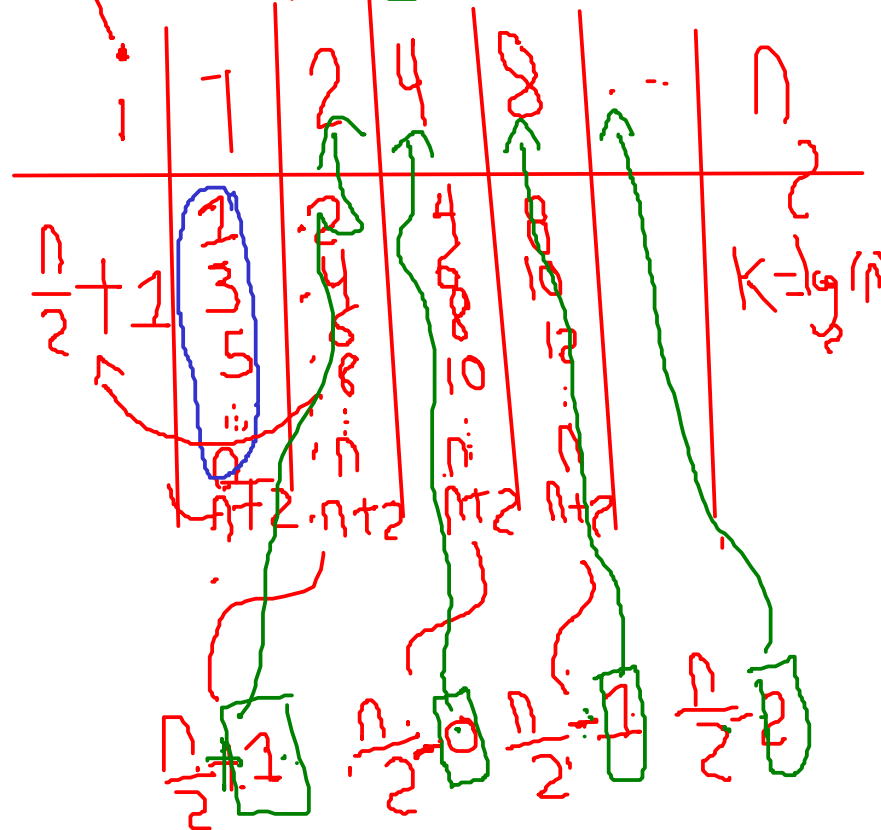
$\log(n)+1$ 1, 3, 4, 8, 16, 32, 64... n

$\log(n)$ $2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6$

$$K = \log(i)$$

$0, 1, 2, 3, \dots, \log(n)$

$\log(n)+1$



$$\frac{n}{2} + 1 + \sum_{k=1}^{\log(n)} \frac{n}{2} + 2 - k$$

$$\frac{n}{2} = k + 2$$

$$\frac{n}{2} + 1 + \frac{\log(n) \cdot n}{2} + \log(n) \times 2 - \frac{\log(n)(\log(n)+1)}{2}$$

$$O(n \log(n))$$

Instrucción	Costo
1 $i \leftarrow 1$	c_1
2 while $i \leq n$	c_2
3 $k \leftarrow i$	c_3
4 while $k \leq n$	c_4
5 $k \leftarrow k+2$	c_5
6 $k \leftarrow 1$	c_6
7 while $k \leq i$	c_7
8 $k \leftarrow k+1$	c_8
9 $i \leftarrow 2i$	c_9

1
 $\log(n) + 1$
 $\log(n)$
 $O(n \log(n))$
 $O(n \log(n))$
 $\log(n)$
 $2n + \log_2(n)$
 $2n - 1$
 $\log(n)$

$2O(n \log_2(n))$

i	2^0	2^1	2^2	2^3	2^4	\dots	n	$2^{\log_2(n)}$
	1	2	4	8	16			
	1	1	1	1	1		1	
	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$		\vdots	
		$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$		\vdots	
			$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$		$\frac{1}{n}$	
				$\frac{1}{7}$	$\frac{1}{7}$		$\frac{1}{n+1}$	

2 3 5 9 17 $n+1$

$$2^k + 1$$

$\log_2(n)$

$$\sum_{k=0}^{\log_2(n)}$$

$$2^k + 1 =$$

$$2^{\log_2(n)+1}$$

$$= 1$$

$$+ \log_2(n) + 1$$

$$\frac{1}{1}$$

$$2[2^{\log_2(n)} - 1 + \log_2(n) + 1]$$

$$2n + \log_2(n)$$

$$\sum_{k=0}^n a r^k = \frac{a r^{n+1} - a}{r - 1}$$

Algoritmos en la computación

Diseño de algoritmos

Otras alternativas para el diseño de algoritmos son:

- Solución ingenua
- Dividir y conquistar
- Programación dinámica
- Técnicas voraces

Algoritmos en la computación

Análisis de algoritmos ordenamiento

Computador de la Abuela
10^9 instrucciones/seg (100MHz)

Implementación 1	Implementación 2
$2n^2$	$50n \cdot \lg n$

Ordenar un arreglo de 10^8 números

Tiempo 1	Tiempo 2
$2(10^8)^2 / 10^9 = 2 \times 10^7 \text{ segs} = 5555,6 \text{ horas}$	$(50 \cdot 10^8 \lg 10^8) / 10^9 = 40 \text{ segs} = 0,66 \text{ mins}$

Algoritmos en la computación

Análisis de algoritmos ordenamiento

Computador Ultima generación
10^{11} instrucciones/seg (10GHz)

Implementación 1	Implementación 2
$2n^2$	$50n \cdot \lg n$

Ordenar un arreglo de 10^8 números

Tiempo 1	Tiempo 2
$2(10^8)^2 / 10^{11} = 2 \times 10^5 \text{ segs} = 55,56 \text{ horas}$	$(50 \cdot 10^8 \lg 10^8) / 10^{11} = 0,4 \text{ segs}$

Referencias

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd ed.). The MIT Press. Pages 5-29

Gracias

Próximo tema:

Computación iterativa:

- Concepto de estado
- Transición de estados
- Invariante de ciclo