Análisis y Diseño de Algoritmos

Carlos Eduardo Atencio Torres
Universidad Nacional de San Agustín
mailto: catencio@episunsa.edu.pe

Empezamos: Ordenación

$A[1..n]$ es creciente si $A[1] \leq ... \leq A[n]$.

Problema

Ordenar un arreglo $A[1..n]$ de modo que quede en forma creciente.

Empezamos: Ordenación

$A[1..n]$ es creciente si $A[1] \leq ... \leq A[n]$.

Problema

Ordenar un arreglo $A[1..n]$ de modo que quede en forma creciente.

Entra

1										n
33	55	33	44	33	22	11	99	22	55	77

Empezamos: Ordenación

$A[1..n]$ es creciente si $A[1] \leq ... \leq A[n]$.

Problema

Ordenar un arreglo $A[1..n]$ de modo que quede en forma creciente.

Entra

1										n
33	55	33	44	33	22	11	99	22	55	77

Sale

1										n
11	22	22	33	33	33	44	55	55	77	99

Ordenación por Inserción

Clave = 38

1					i	j				n
20	25	35	40	44	55	38	99	10	65	50

Ordenación por Inserción

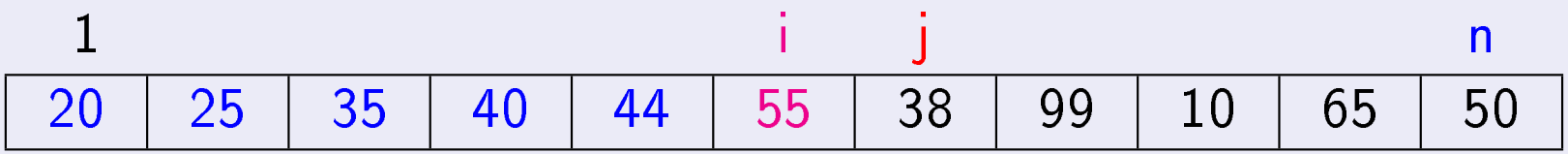
Clave = 38

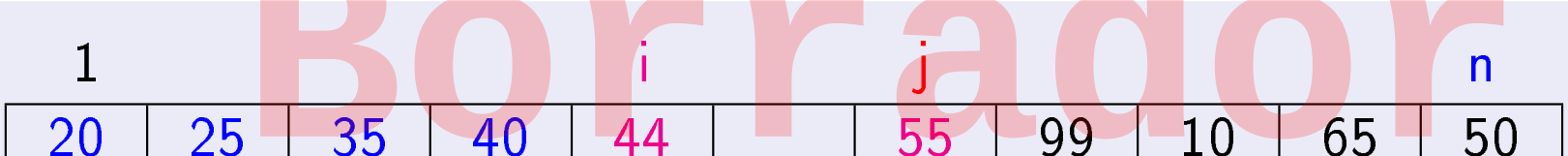
1					i	j				n
20	25	35	40	44	55	38	99	10	65	50

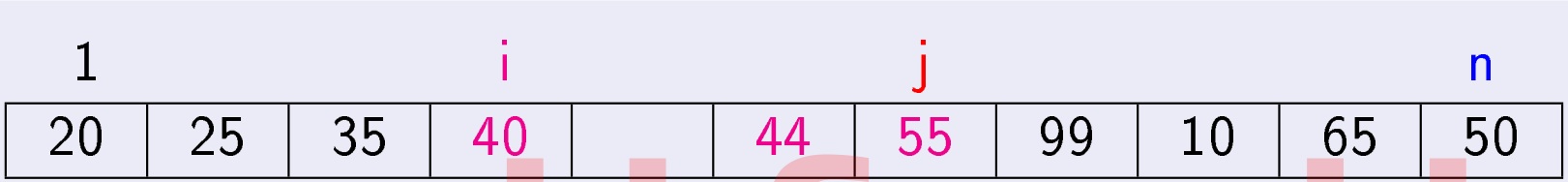
1				i		j				n
20	25	35	40	44		55	99	10	65	50

Ordenación por Inserción

Clave = 38







Ordenación por Inserción

Clave = 38

1					i	j				n
20	25	35	40	44	55	38	99	10	65	50

1				i		j				n
20	25	35	40	44		55	99	10	65	50

1			i			j				n
20	25	35	40		44	55	99	10	65	50

1		i				j				n
20	25	35		40	44	55	99	10	65	50

Ordenación por Inserción

Clave = 38

1					i	j				n
20	25	35	40	44	55	38	99	10	65	50

1				i		j				n
20	25	35	40	44		55	99	10	65	50

1			i			j				n
20	25	35	40		44	55	99	10	65	50

1		i				j				n
20	25	35		40	44	55	99	10	65	50

1		i				j				n
20	25	35	38	40	44	55	99	10	65	50

Ordenación por Inserción

Clave	1							j			n
99	20	25	35	38	40	44	55	99	10	65	50

Ordenación por Inserción

Clave	1							j			n
99	20	25	35	38	40	44	55	99	10	65	50

Clave	1								j		n
10	20	25	35	38	40	44	55	99	10	65	50

Ordenación por Inserción

Clave	1							j			n
99	20	25	35	38	40	44	55	99	10	65	50

Clave	1								j		n
10	20	25	35	38	40	44	55	99	10	65	50

Clave	1									j	n
65	10	20	25	35	38	40	44	55	99	65	50

Ordenación por Inserción

Clave	1							j			n
99	20	25	35	38	40	44	55	99	10	65	50

Clave	1								j		n
10	20	25	35	38	40	44	55	99	10	65	50

Clave	1									j	n
65	10	20	25	35	38	40	44	55	99	65	50

Clave	1										j
50	10	20	25	35	38	40	44	55	65	99	50

Ordenación por Inserción

Clave 99

1							j			n
20	25	35	38	40	44	55	99	10	65	50

Clave 10

1								j		n
20	25	35	38	40	44	55	99	10	65	50

Clave 65

1									j	n
10	20	25	35	38	40	44	55	99	65	50

Clave 50

1										j
10	20	25	35	38	40	44	55	65	99	50

FIN

10	20	25	35	38	40	44	50	55	65	99

Ordenación por Inserción

ORDENA-POR-INSERCION(A,n)

```
1: para j ← 2 hasta n hacer
2:     clave ← A[j]
3:     i ← j − 1
4:     mientras i ≥ 1 AND A[i] > clave hacer
5:         A[i + 1] ← A[i] , ▷ Haciendo campo
6:         i ← i − 1
7:     A[i + 1] ← clave , ▷ Insertando
```

Corrección del algoritmo

Para probar la correctitud de un algoritmo usamos invariantes.

Corrección del algoritmo

Para probar la correctitud de un algoritmo usamos invariantes.

Invariante i0

En la línea 1, se cumple que A[1..j-1] es creciente.

1						j				
20	25	35	40	44	55	38	99	10	65	50

Corrección del algoritmo

Para probar la correctitud de un algoritmo usamos invariantes.

Invariante i0

En la línea 1, se cumple que A[1..j-1] es creciente.

1						j				
20	25	35	40	44	55	38	99	10	65	50

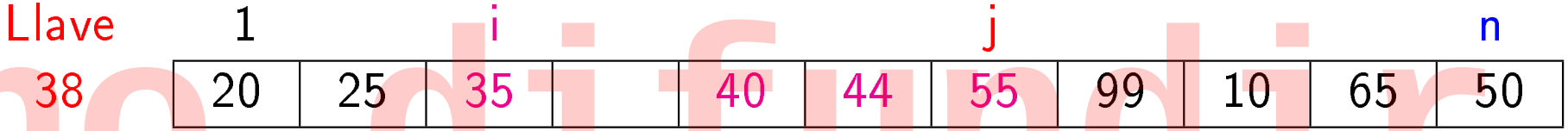
Conclusión

Suponiendo que i0 siempre es válida, cuando j sea n+1, se entiende que A[1..n] es creciente.

Más invariantes

En la línea 4 ocurren las siguientes invariantes:

- i1 A[1..i] y A[i+2..j] son crecientes
- i2 A[1..i] $\leq$ A[i+2..j] son crecientes
- i3 A[i+2..j] $>$ llave.
- i4 A[1..i] + A[i+2..j] + llave no cambian.



Concluyendo

Las demostraciones en un algoritmo iterativo siempre seguirán los siguientes pasos:

1. Verificar que la relación vale al inicio de la primera iteración.
2. Demostrar que si la relación vale al inicio de la iteración, entonces ella valdrá al final.
3. Demostrar que si la relación vale al inicio de la última iteración, entonces la relación junto a la condición de parada demuestran la correctitud del algoritmo.

Asignaciones del algoritmo ($\leftarrow$)

Recordando el algoritmo de INSERCIÓN:

LINEAS 3-6(A,n)

3: $i \leftarrow j - 1$
4: **mientras** $i \geq 1$ AND $A[i] > clave$ **hacer**
5: $\quad A[i+1] \leftarrow A[i]$
6: $\quad i \leftarrow i - 1$
7: $A[i+1] \leftarrow clave$

Contando las asignaciones para el caso máximo:

Asignaciones del algoritmo ($\leftarrow$)

Recordando el algoritmo de INSERCIÓN:

LINEAS 3-6(A,n)

3: $i \leftarrow j - 1$
4: **mientras** $i \geq 1$ AND $A[i] > clave$ **hacer**
5: $\quad A[i+1] \leftarrow A[i]$
6: $\quad i \leftarrow i - 1$
7: $A[i+1] \leftarrow clave$

Contando las asignaciones para el caso máximo:

línea	**asignaciones**
3	$= 1$
4	
5	
6	
total	

Asignaciones del algoritmo ($\leftarrow$)

Recordando el algoritmo de INSERCIÓN:

LINEAS 3-6(A,n)

3: $i \leftarrow j - 1$
4: **mientras** $i \geq 1$ AND $A[i] > clave$ **hacer**
5: $\quad A[i+1] \leftarrow A[i]$
6: $\quad i \leftarrow i - 1$
7: $A[i+1] \leftarrow clave$

Contando las asignaciones para el caso máximo:

línea	**asignaciones**
3	$= 1$
4	$= 0$
5	
6	
total	

Asignaciones del algoritmo ($\leftarrow$)

Recordando el algoritmo de INSERCIÓN:

LINEAS 3-6(A,n)

3: $i \leftarrow j - 1$
4: **mientras** $i \geq 1$ AND $A[i] > clave$ **hacer**
5: $\quad A[i+1] \leftarrow A[i]$
6: $\quad i \leftarrow i - 1$
7: $A[i+1] \leftarrow clave$

Contando las asignaciones para el caso máximo:

línea	**asignaciones**
3	$= 1$
4	$= 0$
5	$\leq$ j-1
6	
total	

Asignaciones del algoritmo ($\leftarrow$)

Recordando el algoritmo de INSERCIÓN:

LINEAS 3-6(A,n)

3: $i \leftarrow j - 1$
4: **mientras** $i \geq 1$ AND $A[i] > clave$ **hacer**
5: $\quad A[i+1] \leftarrow A[i]$
6: $\quad i \leftarrow i - 1$
7: $A[i+1] \leftarrow clave$

Contando las asignaciones para el caso máximo:

línea	**asignaciones**
3	$= 1$
4	$= 0$
5	$\leq$ j-1
6	$\leq$ j-1
total	$\leq$ 2j-1 $\leq$ 2n - 1

Asignaciones del Algoritmo ORDENA-POR-INSERCION

línea		**Asignaciones (número máximo)**
1	$=$	$n-1+1$
2	$=$	$n-1$
3	$=$	$n-1$
4	$=$	0
5	$\leq$	$1+2+\ +(n-1)=n(n-1)/2$
6	$\leq$	$1+2+\ +(n-1)=n(n-1)/2$
7	$=$	$n-1$
total	$\leq$	n^2+3n-3

Consumo de tiempo del Algoritmo ORDENA-POR-INSERCION

- Suponiendo que cada línea de código consume 1 unidad de tiempo.

línea		**Asignaciones (número máximo)**
1	$=$	n
2	$=$	$n-1$
3	$=$	$n-1$
4	$\leq$	$2+3+\cdots+n=(n-1)(n+2)/2$
5	$\leq$	$1+2+\cdots+(n-1)=n(n-1)/2$
6	$\leq$	$1+2+\cdots+(n-1)=n(n-1)/2$
7	$=$	$n-1$
total	$\leq$	

Consumo de tiempo del Algoritmo ORDENA-POR-INSERCION

- Suponiendo que cada línea de código consume 1 unidad de tiempo.

línea		**Asignaciones (número máximo)**
1	$=$	n
2	$=$	$n-1$
3	$=$	$n-1$
4	$\leq$	$2+3+\cdots+n=(n-1)(n+2)/2$
5	$\leq$	$1+2+\cdots+(n-1)=n(n-1)/2$
6	$\leq$	$1+2+\cdots+(n-1)=n(n-1)/2$
7	$=$	$n-1$
total	$\leq$	$(3/2)n^2+(7/2)n-4$

Consumo de tiempo del Algoritmo ORDENA-POR-INSERCION

- Suponiendo que cada línea de código consume 1 unidad de tiempo.
- Suponiendo que cada línea consume un tiempo t_i.

línea		**Asignaciones (número máximo)**	
1	$=$	n	$\times t_1$
2	$=$	$n-1$	$\times t_2$
3	$=$	$n-1$	$\times t_3$
4	$\leq$	$2+3+\cdots+n=(n-1)(n+2)/2$	$\times t_4$
5	$\leq$	$1+2+\cdots+(n-1)=n(n-1)/2$	$\times t_5$
6	$\leq$	$1+2+\cdots+(n-1)=n(n-1)/2$	$\times t_6$
7	$=$	$n-1$	$\times t_7$
total	$\leq$		

Consumo de tiempo del Algoritmo ORDENA-POR-INSERCION

- Suponiendo que cada línea de código consume 1 unidad de tiempo.
- Suponiendo que cada línea consume un tiempo t_i.

línea		**Asignaciones (número máximo)**	
1	$=$	n	$\times t_1$
2	$=$	$n-1$	$\times t_2$
3	$=$	$n-1$	$\times t_3$
4	$\leq$	$2+3+\cdots+n=(n-1)(n+2)/2$	$\times t_4$
5	$\leq$	$1+2+\cdots+(n-1)=n(n-1)/2$	$\times t_5$
6	$\leq$	$1+2+\cdots+(n-1)=n(n-1)/2$	$\times t_6$
7	$=$	$n-1$	$\times t_7$
total	$\leq$	$((t_4+t_5+t_6)/2)\times n^2+$	
		$(t_1+t_2+t_3+t_4/2-t_5/2-t_6/2+t_7)\times n-$	
		$(t_2+t_3+t_4+t_7)$	

Consumo de tiempo del Algoritmo ORDENA-POR-INSERCION

$$\begin{aligned}\textbf{total} = & ((t_4 + t_5 + t_6)/2) \times n^2 + \\ & (t_1 + t_2 + t_3 + t_4/2 - t_5/2 - t_6/2 + t_7) \times n - \\ & (t_2 + t_3 + t_4 + t_7)\end{aligned}$$

Reemplazando t_i por una constante, tenemos:

Consumo de tiempo del Algoritmo ORDENA-POR-INSERCION

$$\begin{aligned}\textbf{total} = & ((t_4 + t_5 + t_6)/2) \times n^2 + \\ & (t_1 + t_2 + t_3 + t_4/2 - t_5/2 - t_6/2 + t_7) \times n - \\ & (t_2 + t_3 + t_4 + t_7)\end{aligned}$$

Reemplazando t_i por una constante, tenemos:

$$c_1 \times n^2 + c_2 \times n + c_3$$

Consumo de tiempo del Algoritmo ORDENA-POR-INSERCION

$$\begin{aligned}\textbf{total} = & ((t_4 + t_5 + t_6)/2) \times n^2 + \\ & (t_1 + t_2 + t_3 + t_4/2 - t_5/2 - t_6/2 + t_7) \times n - \\ & (t_2 + t_3 + t_4 + t_7)\end{aligned}$$

Reemplazando t_i por una constante, tenemos:

$$c_1 \times n^2 + c_2 \times n + c_3$$

- c_1, c_2, c_3 dependen del computador.
- n^2 se repetirá siempre, es algo propio del algoritmo.

Notación Asintótica

Intuitivamente...

$O(f(n)) \approx$ Funciones que no crecen más rápido que $f(n)$

$\approx$ Funciones menores o iguales a un múltiplo de $f(n)$

$n^2 \quad 100n^2 + 0.00000001 \quad n^2/9378 \quad$ etc.

- $n^2 + 3n - 5$ tiene el mismo crecimiento asintótico que n^2
- $n^2 + 3n - 5$ no crece más rápido que n^2
- $n^2 + 3n - 5$ es $O(n^2)$
- $n^2 + 3n - 5 = O(n^2)$
- $n^3 + n^2 - 18n + 65$ No es $O(n^2)$

Ejercicios Propuestos

Ejercicio 1

En función de n, cuanto vale S al final del siguiente algoritmo?

```
1: S ← 0
2: para i ← 2 hasta n − 2 hacer
3:     para j ← i hasta n hacer
4:         S ← S + 1
```

Ejercicios Propuestos

Ejercicio 1

En función de n, cuanto vale S al final del siguiente algoritmo?

```
1: S ← 0
2: para i ← 2 hasta n − 2 hacer
3:     para j ← i hasta n hacer
4:         S ← S + 1
```


Solución

$S = (1/2)n^2 - (1/2)n - 3$

Ejercicios Propuestos

Ejercicio 2

En función de n, cuanto vale S al final del siguiente algoritmo?. Responda con una cota superior cercana.

1: $S \leftarrow 0$
2: $i \leftarrow n$
3: **mientras** $i > 0$ **hacer**
4: **para** $j \leftarrow 1$ hasta i **hacer**
5: $S \leftarrow S + 1$
6: $i \leftarrow \lfloor i/2 \rfloor$

Ejercicios Propuestos

Ejercicio 3

Para el siguiente algoritmo que recibe un arreglo A, se pide el número de...

1. Asignaciones, y
2. Comparaciones.

```
1: s ← 0
2: para i ← hasta n hacer
3:     s ← s + A[i]
4: m ← s/n
5: k ← 1
6: para i ← 2 hasta n hacer
7:     si (A[i] − m)^2 < (A[k] − m)^2 entonces
8:         k ← i
```

Gracias