

# Algoritmos y Estructuras de Datos



Universidad  
Católica del  
Uruguay

Clasificación u Ordenación  
Parte I

1

---

---

---

---

---

---

---

---

## Clasificación u Ordenación



- ¿por qué es tan importante?
  - Es la base sobre la que se construyen muchos otros algoritmos
  - Se aplican las distintas estrategias de diseño de algoritmos
  - Históricamente, se ha utilizado más tiempo de computación en ordenación que en cualquier otra tarea...
  - Es el problema más profundamente estudiado en ciencias de la computación...

Algoritmos y Estructuras de Datos

2

2

---

---

---

---

---

---

---

---

## Aplicaciones de la clasificación



- Una técnica importante de diseño de algoritmos es usar clasificación como componente base:
  - Búsqueda (ej, búsqueda binaria)
  - Par más cercano
  - Unicidad de elementos
  - Distribución de frecuencias (¿qué elemento ocurre la mayor cantidad de veces?)
  - Selección (¿cuál el el k-ésimo mayor elemento?)

Algoritmos y Estructuras de Datos

3

3

---

---

---

---

---

---

---

---

## Algunas consideraciones



- ¿cuántos elementos vamos a ordenar?
- ¿hay claves duplicadas?
- ¿qué sabemos de los datos?
- ¿conocemos las distribuciones de las claves?
- ¿son las claves muy largas o difíciles de comparar?
- ¿es el rango de claves posibles muy pequeño?

Algoritmos y Estructuras de Datos

4

4

---

---

---

---

---

---

---

---

## Citas ...



- *"Understanding the common sorting algorithms is incredibly valuable, as many sorting or searching solutions require tweaks of known sorting algorithms. A good approach when you are given a question like this is to run through the different sorting algorithms and see if one applies particularly well"...* (de "Cracking the coding interview..")
- *"Sorting is the most fundamental algorithmic problem in computer science. Learning the different sorting algorithms is like learning scales for a musician"* (Skiena)
- *"Searching & sorting algorithms form the back bone of coding acumen of developers"* (Searching & Sorting for Coding Interviews : With 100+ Interview questions, Kamal Rawat)

Algoritmos y Estructuras de Datos

5

5

---

---

---

---

---

---

---

---

## Clasificación, Introducción



- Existe un orden lineal definido para los elementos del conjunto a clasificar, por ejemplo, "menor que".
- La clasificación puede dividirse en interna y externa.
- Estabilidad del método (capacidad de mantener el orden relativo de elementos con iguales claves).
- Los algoritmos más simples requieren tiempos de  $O(n^2)$ , otros  $O(n * \log n)$  y algunos, para clases especiales de datos,  $O(n)$ .
- Los objetos a clasificar son estructuras complejas y contienen al menos un elemento del tipo para el cual se define la relación de ordenación (clave).

Algoritmos y Estructuras de Datos

6

6

---

---

---

---

---

---

---

---

## Clasificación Interna



- Ordenar un conjunto de elementos de forma tal que los valores de sus claves formen una secuencia no decreciente.
- Usar con cuidado el almacenamiento disponible.
- Medida de eficiencia: contar número de comparaciones de claves  $C$  y de movimientos de elementos  $M$ .
- Los buenos algoritmos de clasificación requieren  $O(n \log n)$  comparaciones, los más sencillos,  $O(n^2)$
- Métodos directos e indirectos.
  - Los directos son más cortos y fáciles de entender
  - Las operaciones de los indirectos son más complejas, por lo que los métodos directos pueden ser más rápidos para pequeños conjuntos de datos

Algoritmos y Estructuras de Datos

7

7

---

---

---

---

---

---

---

---

## Clasificación interna



Podemos agrupar los métodos de la siguiente forma:

- Clasificación por inserción
- Clasificación por enumeración o cuenta.
- Clasificación por intercambio.
- Clasificación por selección
- Especiales: ej.: Clasificación por urnas y por residuos

Algoritmos y Estructuras de Datos

8

8

---

---

---

---

---

---

---

---

## Clasificación



- Ordenar en forma no decreciente un vector con " $N$ " elementos que están desde la posición 1 a la  $N$ .
- Cada elemento posee un atributo de datos y un atributo "clave" por el cual se ordenará el vector:
  - $V[i]$  es el elemento de la posición " $i$ " del vector  $V$ .
  - $V[i].datos$  es el atributo de datos del elemento.
  - $V[i].clave$  es la clave por la que se ordenará el vector.

Algoritmos y Estructuras de Datos

9

9

---

---

---

---

---

---

---

---

## Métodos de Clasificación por Inserción



Algoritmos y Estructuras de Datos

10

10

---

---

---

---

---

---

---

---

### Clasificación por Inserción



- en el  $i$  - ésimo recorrido se inserta el  $i$  - ésimo elemento en el lugar correcto entre los  $(i-1)$  elementos anteriores, los cuales fueron ordenados previamente.
- Después de hacer la inserción, se encuentran clasificados los elementos  $V[1], \dots, V[i]$ .
- En un vector:

```
for i = 2 to n do
  mover V[i] hacia la posición j <= i tal que
    V[i].clave < V[j].clave para j <= i-1, y
    V[i].clave >= V[j-1].clave o j=1.
```

Algoritmos y Estructuras de Datos

11

11

---

---

---

---

---

---

---

---

### Clasificación – Algoritmo de Inserción Directa



- Inserción directa en vector.

#### Comienzo

- (1) Desde  $i = 2$  hasta  $N$  hacer
- (2)  $Aux \leftarrow V[i]$
- (3)  $j = i - 1$
- (4) mientras  $j > 0$  y  $Aux.clave < V[j].clave$  hacer
- (5)  $V[j+1] \leftarrow V[j]$
- (6)  $j \leftarrow j-1$
- (7) fin mientras
- (8)  $V[j+1] \leftarrow Aux$
- (9) fin desde

#### Fin

Algoritmos y Estructuras de Datos

12

12

---

---

---

---


---

---

---

---

Ordenar por inserción directa



1	2	3	4	5	6	7	8	
44	55	12	42	94	18	6	67	i
								2
								3
								4
								5
								6
								7
								8

Mostrando cómo queda el vector después de terminada la iteración para cada valor de "i".

Algoritmos y Estructuras de Datos

13

13

---

---

---

---

---

---


---

---

---

---

Ordenar por inserción directa



1	2	3	4	5	6	7	8	
44	55	12	42	94	18	6	67	i
44	55	12	42	94	18	6	67	2
								3
								4
								5
								6
								7
								8

Mostrando cómo queda el vector después de terminada la iteración para cada valor de "i".

Algoritmos y Estructuras de Datos

14

14

---

---

---

---

---

---


---

---

---

---

Ordenar por inserción directa



1	2	3	4	5	6	7	8	
44	55	12	42	94	18	6	67	i
44	55	12	42	94	18	6	67	2
12	44	55	42	94	18	6	67	3
								4
								5
								6
								7
								8

Mostrando cómo queda el vector después de terminada la iteración para cada valor de "i".

Algoritmos y Estructuras de Datos

15

15

---

---

---

---

---

---


---

---

---

---

Ordenar por inserción directa



1	2	3	4	5	6	7	8	i
44	55	12	42	94	18	6	67	2
44	55	12	42	94	18	6	67	3
12	44	55	42	94	18	6	67	4
12	42	44	55	94	18	6	67	5
								6
								7
								8

Mostrando cómo queda el vector después de terminada la iteración para cada valor de "i".

Algoritmos y Estructuras de Datos

16

16

---

---

---

---


---

---

---

---

Ordenar por inserción directa



1	2	3	4	5	6	7	8	i
44	55	12	42	94	18	6	67	2
44	55	12	42	94	18	6	67	3
12	44	55	42	94	18	6	67	4
12	42	44	55	94	18	6	67	5
12	42	44	55	94	18	6	67	6
								7
								8

Mostrando cómo queda el vector después de terminada la iteración para cada valor de "i".

Algoritmos y Estructuras de Datos

17

17

---

---

---

---


---

---

---

---

Ordenar por inserción directa



1	2	3	4	5	6	7	8	i
44	55	12	42	94	18	6	67	2
44	55	12	42	94	18	6	67	3
12	44	55	42	94	18	6	67	4
12	42	44	55	94	18	6	67	5
12	42	44	55	94	18	6	67	6
12	18	42	44	55	94	6	67	7
								8

Mostrando cómo queda el vector después de terminada la iteración para cada valor de "i".

Algoritmos y Estructuras de Datos

18

18

---

---

---

---


---

---

---

---

Ordenar por inserción directa



1	2	3	4	5	6	7	8	i
44	55	12	42	94	18	6	67	2
44	55	12	42	94	18	6	67	3
12	44	55	42	94	18	6	67	4
12	42	44	55	94	18	6	67	5
12	42	44	55	94	18	6	67	6
12	18	42	44	55	94	6	67	7
6	12	18	42	44	55	94	67	8

Mostrando cómo queda el vector después de terminada la iteración para cada valor de "i".

Algoritmos y Estructuras de Datos

19

19

---

---

---

---


---

---

---

---

Ordenar por inserción directa



1	2	3	4	5	6	7	8	i
44	55	12	42	94	18	6	67	2
44	55	12	42	94	18	6	67	3
12	44	55	42	94	18	6	67	4
12	42	44	55	94	18	6	67	5
12	42	44	55	94	18	6	67	6
12	18	42	44	55	94	6	67	7
6	12	18	42	44	55	94	67	8

Mostrando cómo queda el vector después de terminada la iteración para cada valor de "i".

Algoritmos y Estructuras de Datos

20

20

---

---

---

---

---

---

---

---

Inserción directa en vector: análisis del método



Comienzo

(1) Desde i = 2 hasta N  
hacer

(2) Aux ← V[i]

(3) j = i - 1

(4) mientras j > 0 y  
Aux.clave < V[j].clave  
hacer

(5) V[j+1] ← V[j]

(6) j ← j-1

(7) fin mientras

(8) V[j+1] ← Aux

(9) fin desde

Fin

Comienzo

(1) Desde: exactamente N-1  
veces.

(2) O(1)

(3) O(1)

(4) mientras: peor caso N-i veces,  
mejor caso 1 vez

(5) O(1)

(6) O(1)

(7) fin mientras

(8) O(1)

(9) fin desde

Fin

Algoritmos y Estructuras de Datos

21

21

---

---

---

---

---

---

---

---

7

## Inserción directa en vector: análisis del método



- El bucle interior “mientras”:
  - si el conjunto ya está ordenado, ejecuta solamente la condición y no su bloque interno,  $O(1)$ .
  - Si el conjunto está ordenado al revés, ejecuta  $N-i$  veces para cada valor de  $i$ ,  $O(N)$ .
  - Si las claves del conjunto se encuentran ubicadas al azar, en promedio será  $O(N)$ .
- Por lo tanto, se demuestra que el orden del tiempo de ejecución de este algoritmo presenta dos casos:
  - Mejor caso  $O(N)$  cuando el conjunto ya está ordenado.
  - Peor caso y caso promedio:  $O(N^2)$
- Una mejora que no afecta el orden: uso de un centinela.

Algoritmos y Estructuras de Datos

22

22

---

---

---

---

---

---

---

---

## Trabajo de Aplicación 1, Ejercicio 1



Algoritmos y Estructuras de Datos

23

23

---

---

---

---

---

---

---

---

## Inserción en Lista



- Un método efectivo para mejorar un algoritmo dado es examinar cuidadosamente la estructura de los datos a clasificar.
- ¿Cuál es la estructura más adecuada para la inserción directa ?
- La inserción directa implica dos operaciones básicas:
  - exploración de un conjunto ordenado para hallar la posición en la que hay que insertar.
  - inserción del elemento en el lugar hallado.
- La mejor estructura es la lista encadenada simple.

Algoritmos y Estructuras de Datos

24

24

---

---

---

---

---

---

---

---



## Inserción directa con listas



### Comienzo

- (1) Mientras no (ListaDeEntrada.Vacia) hacer
- (2) ElementoAMover  $\leftarrow$  ListaDeEntrada.PrimerO
- (3) ListaDeEntrada.EliminarPrimerO
- (4) ListaDeSalida.InsertarOrdendado(ElementoAMover)
- (5) Fin mientras
- (6) Devolver ListaDeSalida.

### Fin

Algoritmos y Estructuras de Datos

25

25

---

---

---

---

---

---

---

---

## Inserción directa con listas: análisis del método



### Comienzo

- (1) Mientras no (ListaDeEntrada.Vacia) hacer
- (2) ElementoAMover  $\leftarrow$  ListaDeEntrada.PrimerO
- (3) ListaDeEntrada.EliminarPrimerO
- (4) ListaDeSalida.InsertarOrdendado (ElementoAMover)
- (5) Fin mientras
- (6) Devolver ListaDeSalida.

### Fin

### Comienzo

- (1) Mientras:  $O(N)$
- (2)  $O(1)$
- (3)  $O(1)$
- (4)  $O(N)$ , ¡pero  $O(1)$  si estaba ordenada al revés!
- (5) Fin mientras
- (6)  $O(1)$

### Fin

$O(N^2)$ : peor caso y caso promedio

$O(N)$ : mejor caso, la lista estaba ordenada al revés.

Algoritmos y Estructuras de Datos

26

26

---

---

---

---

---

---

---

---

## Inserción en listas múltiples.



- Si se conoce de antemano el rango de las claves, y éstas se encuentran uniformemente distribuidas en el entorno, se pueden utilizar varias listas abarcando cada una un sub-rango.
- Se mantienen varias listas en vez de una.
- Se requiere espacio para las  $M$  cabeceras de listas.
- Dentro de cada lista se utiliza el método anterior.

Algoritmos y Estructuras de Datos

27

27

---

---

---

---

---

---

---

---

## Clasificación



- Ejemplo de listas múltiples.
  - $M = 4$ .
  - Rangos: 0 - 249, 250 - 499, 500 - 749, 750 - 999

Conjunto original:

503 087 512 061 908 170 897 275 653 426 154 509 612 677 765 703

Resultado final:

Lista 1: 061 087 154 170

Lista 2: 275 426

Lista 3: 503 509 512 612 653 677 703

Lista 4: 765 897 908

Algoritmos y Estructuras de Datos

28

28

---

---

---

---

---

---

---

---

## inserción directa en listas múltiples



¿cuál es el orden del tiempo de ejecución ?

- $n / M$
- $N^2 / M$
- $n$
- Otro?

Algoritmos y Estructuras de Datos

29

29

---

---

---

---

---

---

---

---

## Método de Shell (Shellsort)



O clasificación por disminución de incrementos.

- Si el algoritmo mueve los elementos sólo una posición por vez su tiempo de ejecución será proporcional a  $N^2$ .
- Buscamos un mecanismo para que los elementos puedan dar grandes saltos en vez de pequeños pasos.
- Ejemplo: primero dividimos los 16 registros en 8 grupos de dos,  $(R_1, R_9), (R_2, R_{10}), \dots, (R_8, R_{16})$  y clasificamos cada grupo por separado.
- A continuación dividimos los elementos en 4 grupos de 4, y clasificamos cada grupo por separado.
- Continuamos así hasta tener un sólo grupo con los 16 elementos.

Algoritmos y Estructuras de Datos

30

30

---

---

---

---

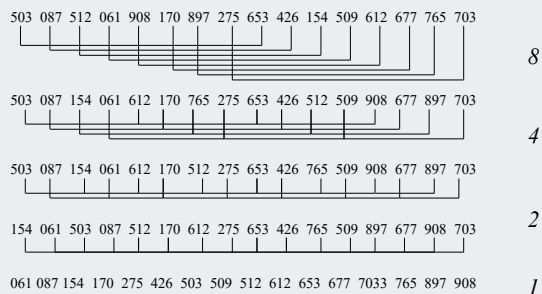
---

---

---

---

## Clasificación por disminución de incrementos



Algoritmos y Estructuras de Datos

31

31

---

---

---

---

---

---

---

---

## Algoritmo de Shell



### Comienzo

Desde  $k = t$  hasta 1 hacer

$h \leftarrow \text{Inc}[k]$

Desde  $i = 1 + h$  hasta  $N$  hacer

$\text{Aux} \leftarrow V[i]$

$j = i - h$

mientras  $j > 0$  y  $\text{Aux.clave} < V[j].\text{clave}$  hacer

$V[j+h] \leftarrow V[j]$

$j \leftarrow j - h$

fin mientras

$V[j+h] \leftarrow \text{Aux}$

fin desde

fin desde

Fin

Vector de incrementos



Algoritmos y Estructuras de Datos

32

32

---

---

---

---

---

---

---

---

## Análisis del algoritmo de Shell.



- Para elegir una buena secuencia de incrementos es necesario analizar el tiempo de ejecución en función de estos incrementos.
- No se conoce la mejor secuencia para grandes valores de  $N$ .
- Los incrementos no deben ser múltiplos de sí mismos, de forma que las cadenas se mezclen entre sí lo más a menudo posible.
- Secuencias razonables (en orden inverso) :
  - 1, 4, 13, 40, 121 ...
  - 1, 3, 7, 15, 31 ...
- El orden es de  $n^{1.26}$ .

Algoritmos y Estructuras de Datos

33

33

---

---

---

---

---

---

---

---

Ejemplo de Shellsort

Universidad  
Pública de  
Uruguay

• Incrementos: {5, 3, 1}

1	2	3	4	5	6	7	8	9	10	11	12	13
81	94	11	96	12	35	17	95	28	58	41	75	15

Algoritmos y Estructuras de Datos

34

34

---

---

---

---

---

---

---

---

Ejemplo de Shellsort

Universidad  
Pública de  
Uruguay

• Primero con el incremento = 5

1	2	3	4	5	6	7	8	9	10	11	12	13
81	94	11	96	12	35	17	95	28	58	41	75	15

Algoritmos y Estructuras de Datos

35

35

---

---

---

---

---

---

---

---

Ejemplo de Shellsort

Universidad  
Pública de  
Uruguay

• Primero con el incremento = 5

1	2	3	4	5	6	7	8	9	10	11	12	13
81	94	11	96	12	35	17	95	28	58	41	75	15
35	17	11	28	12	41	75	15	96	58	81	94	95

Algoritmos y Estructuras de Datos

36

36

---

---

---

---

---

---

---

---

[illegible]

Algoritmos y Estructuras de Datos

37



Universidad  
Católica del  
Uruguay

1	2	3	4	5	6	7	8	9	10	11	12	13
35	17	11	28	12	41	75	15	96	58	81	94	95
28	12	11	35	15	41	58	17	94	75	81	96	95

Algoritmos y Estructuras de Datos

38



Universidad  
Católica del  
Uruguay

1	2	3	4	5	6	7	8	9	10	11	12	13
28	12	11	35	15	41	58	17	94	75	81	96	95

Algoritmos y Estructuras de Datos

39

## Ejemplo de Shellsort



- Por último con incremento = 1

1	2	3	4	5	6	7	8	9	10	11	12	13
28	12	11	35	15	41	58	17	94	75	81	96	95
11	12	15	17	28	35	41	58	75	81	94	95	96

Algoritmos y Estructuras de Datos

40

---

---

---

---

---

---

---

---

## Ejemplo de Shellsort



- Array ordenado

1	2	3	4	5	6	7	8	9	10	11	12	13
11	12	15	17	28	35	41	58	75	81	94	95	96

Algoritmos y Estructuras de Datos

41

---

---

---

---

---

---

---

---

## Trabajo de Aplicación 1, Ejercicio 2



Algoritmos y Estructuras de Datos

42

---

---

---

---

---

---

---

---

## Métodos de Inserción, resumen



- Inserción directa en vector:  $O(N^2)$  en el peor y en el caso promedio,  $O(N)$  en el mejor caso.
- Inserción directa en lista encadenada:  $O(N^2)$  en el peor y en el caso promedio,  $O(N)$  en el mejor caso. No hay movimientos, hay cambios de referencias.
- Inserción en listas múltiples: mejora comparaciones a  $N^2 / M$ .
- Shell:  $O(N^{1.26})$ .

Algoritmos y Estructuras de Datos

43

43

---

---

---

---

---

---

---

---

## Ejercicios



- ¿Qué es la clasificación interna y cuáles son sus objetivos? ¿Cómo se pueden clasificar los distintos métodos existentes? Explique las características de cada grupo. Ilustre con ejemplos.
- Explique los métodos de Clasificación por Inserción Directa y Shellsort sobre un vector. Desarrolle el algoritmo para este último. Ilustre con ejemplos.
- Utilizando el método de Inserción directa sobre el siguiente conjunto de datos, mostrar en cada paso cómo se van clasificando.  
256 458 365 298 043 648 778 621 655 019 124 847
- Utilizando el método de Shell con la secuencia de incrementos (4, 2, 1) sobre el siguiente conjunto de datos, mostrar en cada paso cómo se van clasificando.  
256 458 365 298 043 648 778 621 655 019 124 847

Algoritmos y Estructuras de Datos

44

44

---

---

---

---

---

---

---

---

## Métodos de Clasificación por Intercambio



Algoritmos y Estructuras de Datos

45

45

---

---

---

---

---

---

---

---

## Métodos de Intercambio



- Llamados así porque el intercambio o trasposición de elementos es su característica dominante.
- El algoritmo se basa en el principio de comparar e intercambiar pares subsiguientes de elementos, hasta que todos estén ordenados.
- Poco aptos para su implementación con listas encadenadas.

Algoritmos y Estructuras de Datos

46

46

---

---

---

---

---

---

---

---

## Burbuja



- (1) Desde  $i = 1$  hasta  $N-1$  hacer
- (2) Desde  $j = N$  hasta  $i+1$  hacer
- (3) Si  $V[j].clave < V[j-1].clave$  entonces
- (4) intercambiar ( $V[j]$ ,  $V[j-1]$ )
- Fin si
- Fin desde
- Fin desde

Algoritmos y Estructuras de Datos

47

47

---

---

---

---

---

---

---

---

## Burbuja - Análisis



- (1) Desde  $i = 1$  hasta  $N-1$  hacer
- (2) Desde  $j = N$  hasta  $i+1$  hacer
- (3) Si  $V[j].clave < V[j-1].clave$  intercambiar ( $V[j]$ ,  $V[j-1]$ )
- Fin si
- (4) Fin desde
- Fin desde

- Intercambia lleva un tiempo constante  $O(1)$ , igual que la prueba "Si".
- Los pasos 2 a 4 llevan un tiempo proporcional a  $(N-i)$ .
- Estos se ejecutan desde  $i = 1$  hasta  $N-1$ .
- Por lo tanto, la cantidad de comparaciones que se realizan serán de orden:  $O(N^2)$  siempre, ya que el algoritmo no detecta diferencia entre casos.

Algoritmos y Estructuras de Datos

48

48

---

---

---

---

---


---

---

---



Ordenar por burbuja



1	2	3	4	5	6	7	8	i
44	55	12	42	94	18	6	67	1
								2
								3
								4
								5
								6
								7

Mostrando cómo queda el vector después de terminada la iteración para cada valor de "i".

Algoritmos y Estructuras de Datos

49

49

---

---

---

---


---

---

---

---

Ordenar por burbuja



1	2	3	4	5	6	7	8	i
44	55	12	42	94	18	6	67	1
6	44	55	12	42	94	18	67	2
6	12	44	55	18	42	94	67	3
6	12	18	44	55	42	67	94	4
6	12	18	42	44	55	67	94	5
6	12	18	42	44	55	67	94	6
6	12	18	42	44	55	67	94	7

Mostrando cómo queda el vector después de terminada la iteración para cada valor de "i".

Algoritmos y Estructuras de Datos

50

50

---

---

---

---


---

---

---

---

Trabajo de Aplicación 1,  
Ejercicio 3



Algoritmos y Estructuras de Datos

51

51

---

---

---

---

---

---

---

---

## Quicksort



- Es tal vez el algoritmo más eficiente para clasificación interna. Su orden es de  $n \cdot \log n$ .
- La idea es clasificar un conjunto de elementos  $V[1]..V[N]$  tomando uno de ellos, de clave  $V[p].clave$ , como *pivote*, procurando que sea la mediana del conjunto, de forma que esté precedido y sucedido por más o menos la mitad de los elementos del conjunto.
- Se permutan los elementos de forma que, para algún valor de  $j$ , todos los que tienen clave menor que  $V[p].clave$  se encuentran a la izquierda de  $j$ , y los de clave mayor o igual están a la derecha.

Algoritmos y Estructuras de Datos

52

52

---

---

---

---

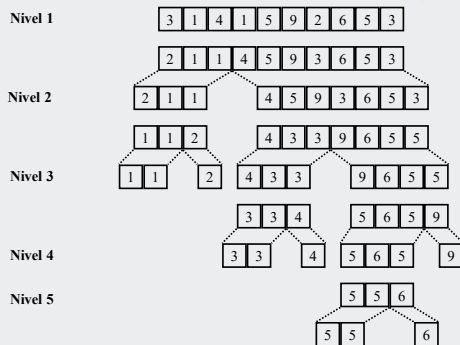
---

---

---

---

## Quicksort



Algoritmos y Estructuras de Datos

53

53

---

---

---

---

---

---

---

---

## Desarrollo del algoritmo de Quicksort



- El algoritmo opera sobre un conjunto de elementos  $V[1]..V[N]$  definido de manera externa.
- El procedimiento **quicksort(i,j)** ordena desde  $V[i]$  hasta  $V[j]$  en el mismo lugar.

### quicksort(i,j)

(1) Pivote  $\leftarrow$  ObtenerClavePivote(i,j)

(2) Si existe un Pivote **entonces**

(3) permutar  $V[i]..V[j]$  de forma que, para alguna  $k$  tal que  $i+1 \leq k < j$ ,  
 $V[i].clave..V[k-1].clave < \text{Pivote}$  y  
 $V[k].clave..V[j].clave \geq \text{Pivote}$

(5) quicksort(i,k-1)

(6) quicksort(k,j)

Fin si

Algoritmos y Estructuras de Datos

54

54

---

---

---

---

---

---

---

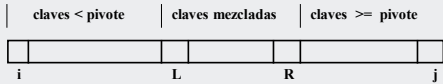
---

### Desarrollo de Quicksort



Aplicar la línea (3) del algoritmo, permutando en el lugar.

- Introducimos dos cursores, L y R, en los extremos izquierdo y derecho del conjunto considerado, respectivamente.
- Los elementos a la izquierda de L ( $V[i]..V[L-1]$ ) siempre tendrán claves menores que el pivote, y los elementos a la derecha de R ( $V[R+1]..V[j]$ ) tendrán claves mayores o iguales que el pivote.
- Los elementos del centro estarán mezclados.



55

---

---

---

---

---

---

---

---

### Quicksort, Pasos del algoritmo



- Rastrear.
  - Mover L a la derecha en los registros cuyas claves sean menores que el pivote. Mover R a la izquierda en las claves mayores o iguales que el pivote.
- Probar.
  - Si  $L > R$  ( $L = R + 1$ ), entonces el conjunto ya se ha dividido, salir.
- Desviar.
  - Si  $L < R$ , intercambiar  $V[L]$  con  $V[R]$ . Después de hacerlo,  $V[L].clave$  es menor que el pivote y  $V[R].clave$  es mayor o igual.

56

---

---

---

---

---

---

---

---

### Quicksort - partición.



```

function particion( i, j: integer; pivote: TipoClave): integer:
    {divide V[i], ..., V[j] para que las claves menores que pivote estén a la izquierda y las mayores o iguales a la derecha. Devuelve el lugar donde se inicia el grupo de la derecha.}
    var
        L, R : de tipo enteros; {cursores de acuerdo a la descripción anterior}
    COM
    (1) L := i;
    (2) R := j;
    (3) Repetir
    (4)     intercambia(V[L],V[R]);
        {ahora se inicia la fase de rastreo}
    (5)     mientras V[L].clave < pivote hacer L := L + 1; fin mientras
    (6)     mientras V[R].clave >= pivote hacer R := R - 1; fin mientras
    (7)     Hasta que L > R
    (8)     Devolver L;
    end; {particion}
    
```

57

---

---

---

---

---

---

---

---

## Quicksort: método principal



```
quicksort( i,j: tipo entero);
//clasifica los elementos V[i]...V[j] del arreglo externo V

pivote : TipoClave; {el valor del pivote}
IndicePivote : tipo entero; {el índice de un elemento de V donde clave es el pivote}
k : tipo entero; {índice al inicio del grupo de elementos >= pivote}

COM
IndicePivote ← EncuentraPivote(i,j);
SI IndicePivote <> 0 entonces {no hacer nada si todas las claves son iguales}
    pivote ← V[IndicePivote].clave;
    k ← particion(i,j,pivote);
    quicksort(i,k-1);
    quicksort(k,j);
FIN SI;
FIN; //quicksort
```

(1)  
(2)  
(3)  
(4)  
(5)  
(6)

58

---

---

---

---

---

---

---

---

## Quicksort: Análisis del tiempo de ejecución



- El algoritmo insume en el mejor caso y en el caso promedio un tiempo  $O(n \log n)$  y en el peor caso,  $O(n^2)$ .
- El mejor caso se da cuando el pivote es en cada elección la mediana del conjunto.
- El peor caso se da cuando el pivote elegido es un extremo del conjunto.
- En todos los casos las sentencias del método principal pueden ser  $O(1)$ , excepto **partición** que será  $O(j-i)$ . El orden de cada llamada será entonces  $O(j-i)$ .
- Se ejecutarán  $2N-1$  llamadas al algoritmo.

59

---

---

---

---

---

---

---

---

## Clasificación: Quicksort



- Análisis del tiempo de ejecución para el mejor caso, el subrango se parte siempre a la mitad.
- Analizando la cantidad de llamadas recursivas, cada una con su tamaño de entrada, se obtiene:

$$N + \left(\frac{N}{2} + \frac{N}{2}\right) + \left(\frac{N}{4} + \frac{N}{4} + \frac{N}{4} + \frac{N}{4}\right) + \dots + N * \left(\frac{N}{N}\right) =$$

$$N + N * \left(\frac{2}{2}\right) + N * \left(\frac{4}{4}\right) + N * \left(\frac{8}{8}\right) \dots + N * \left(\frac{N}{2^K}\right) =$$

$$N * \left(\frac{2^0}{2^0} + \frac{2^1}{2^1} + \frac{2^2}{2^2} + \frac{2^3}{2^3} + \dots + \frac{2^K}{2^K}\right) = N * (1 + K)$$

Como  $K = \log_2 N$ , entonces el orden es  $O(N \log N)$

60

---

---

---

---

---

---

---

---

## Clasificación: Quicksort



- Análisis del tiempo de ejecución para el peor caso, el subrango se parte siempre en un extremo.
- Analizando la cantidad de llamadas recursivas, cada una con su tamaño de entrada, se obtiene:

$$N + (1 + N - 1) + (1 + N - 2) + \dots + (1 + N - (N - 2)) + (1 + N - (N - 1)) =$$

$$N + \sum_{i=1}^{N-1} (1 + (N - i)) = 2N - 1 + \frac{N(N-1)}{2} \Rightarrow O(N^2)$$

Algoritmos y Estructuras de Datos

61

61

---

---

---

---

---

---

---

---

## Quicksort - Ejercicios



- Ordenar por Quicksort los siguientes elementos:  
256 458 365 298 043 648 778 621 655 019 124 847
- ¿Cuál es la profundidad de llamadas recursivas que se realiza?
- Combinación con métodos simples cuando el conjunto de datos es pequeño
- Control de la profundidad de llamada recursiva
- Prácticas en computadora: peor caso, mala elección del pivote y disposición de los datos inconveniente

Algoritmos y Estructuras de Datos

62

62

---

---

---

---

---

---

---

---

## Quicksort - Ejercicios



- Pivotes:
  - ¿media?,
  - ¿Elemento central del conjunto?
  - ¿mayor de dos primeros?
  - ¿órdenes de ejecución? ¿Probabilidades de ocurrencia de las claves?
- Mediana
  - Desarrollar algoritmo para encontrar la mediana
  - ¿Orden?

Algoritmos y Estructuras de Datos

63

63

---

---

---

---

---

---

---

---

## Preguntas y Ejercicios



- ¿Qué es la clasificación interna y cuáles son sus objetivos? ¿Cómo se pueden categorizar los distintos métodos de ordenación existentes? Explique las características de cada grupo. ¿a qué grupo pertenece el algoritmo de SHELLSORT?
- ¿Cuál es el orden de ejecución del MEJOR CASO del algoritmo de QUICKSORT? Indique cuál es el caso.
- Dado el siguiente conjunto de datos, proceda a clasificarlo utilizando el algoritmo de QUICKSORT, mostrando la evolución del vector en cada iteración.

22- 11- 44- 55- 88- 77- 33- 01

Algoritmos y Estructuras de Datos

64

64

---

---

---

---

---

---

---

---

## Preguntas y Ejercicios



- La inserción ordenada de un elemento en una lista encadenada tiene un orden de ejecución:
  1.  $O(N)$
  2.  $O(N^3)$
  3.  $O(N^2)$
  4.  $O(N \log N)$
  5.  $O(\log N)$
- ¿Cuál es el orden de ejecución en el peor caso del algoritmo de clasificación QuickSort?
  1.  $O(\log_2(n))$ .
  2.  $O(n)$ .
  3.  $O(n * \log_2(n))$ .
  4.  $O(n^2)$ .

Algoritmos y Estructuras de Datos

65

65

---

---

---

---

---

---

---

---

## Preguntas y Ejercicios



- ¿Cuál de los siguientes algoritmos se comporta mejor si tenemos un conjunto de datos que ya está ordenado?
  1. Inserción Directa.
  2. QuickSort.
  3. HeapSort.
  4. Burbuja
- ¿A cuál método de clasificación corresponde el siguiente código fuente?
 

```
public void metodo(int[] vector) {
    for (int i=0; i <= vector.length-2; i++) {
        for (int j=vector.length-1; j >= i+1; j++) {
            if (vector[j] < vector[j-1])
                intercambia(vector, j, j-1);
        }
    }
}
```

Algoritmos y Estructuras de Datos

66

66

---

---

---

---

---

---

---

---

## Preguntas y Ejercicios



- Escriba un algoritmo en pseudocódigo que implemente el método de clasificación Quicksort sobre un vector de claves, muestre su orden del tiempo de ejecución y aplíquelo sobre el siguiente conjunto de datos mostrando cómo queda el vector en cada paso. Para este caso, utilizar como criterio de selección del pivote, al mayor de los primeros dos elementos diferentes del conjunto

**223 784 376 285 015 440 666 007**

- Muestre el funcionamiento de los métodos de clasificación intercambio directo ("burbuja"), selección directa e inserción directa aplicándolos sobre el vector de claves del ejercicio anterior, mostrando el estado del vector luego de cada bucle más externo (para cada valor de "i").

Algoritmos y Estructuras de Datos

67

67

---

---

---

---

---

---

---

---

## Algunos links útiles:



- **Algoritmos de ordenación :**
  - <http://www.bogotobogo.com/Algorithms/bubblesort.php>
- **Inserción**
  - [http://en.wikipedia.org/wiki/Insertion\\_sort](http://en.wikipedia.org/wiki/Insertion_sort)
  - [http://es.wikipedia.org/wiki/Ordenamiento\\_por\\_inserci%C3%B3n](http://es.wikipedia.org/wiki/Ordenamiento_por_inserci%C3%B3n)
  - <http://www.youtube.com/watch?v=Fr0SmtN0IJM>
  - [http://www.algolist.net/Algorithms/Sorting/Insertion\\_sort](http://www.algolist.net/Algorithms/Sorting/Insertion_sort)
- **Shellsort:**
  - <http://en.wikipedia.org/wiki/Shellsort>
  - [http://es.wikipedia.org/wiki/Ordenamiento\\_Shell](http://es.wikipedia.org/wiki/Ordenamiento_Shell)
  - <http://www.youtube.com/watch?v=cV6UxwdkLuc&feature=related>

Algoritmos y Estructuras de Datos

68

68

---

---

---

---

---

---

---

---

## Algunos links útiles:



- **Burbuja:**
  - [http://es.wikipedia.org/wiki/Ordenamiento\\_de\\_burbuja](http://es.wikipedia.org/wiki/Ordenamiento_de_burbuja)
  - [http://en.wikipedia.org/wiki/Bubble\\_sort](http://en.wikipedia.org/wiki/Bubble_sort)
  - <http://www.youtube.com/watch?v=Unk5ueUgc88&feature=relmfu>
  - [http://www.algolist.net/Algorithms/Sorting/Bubble\\_sort](http://www.algolist.net/Algorithms/Sorting/Bubble_sort)
- **Quicksort:**
  - <http://es.wikipedia.org/wiki/Quicksort>
  - <http://www.csanimated.com/animation.php?t=Quicksort>
  - <http://www.cse.iitk.ac.in/users/dsrkg/cs210/applets/sortin gl/quickSort/quickSort.html>
  - <http://www.youtube.com/watch?v=Z5nSXTnD1I4&feature=related>
  - <http://www.algolist.net/Algorithms/Sorting/Quicksort>

Algoritmos y Estructuras de Datos

69

69

---

---

---

---

---

---

---

---