Ordenamiento en tiempo lineal

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y T.I.

CI-2612: Algoritmos y Estructuras de Datos II

Plan

Tiempo de los algoritmos de ordenamientos por comparaciones

- InsertionSort $O(n^2)$
- Mergesort $\Theta(n \log n)$
- Heapsort $\Theta(n \log n)$
- Quicksort $\Theta(n \log n)$ en promedio

Algoritmos de ordenamientos por comparaciones

- Se comparan los elementos de la secuencia de entrada $(a1, a2, \ldots, an)$
- Para determinar el orden de ejecuta $ai > aj$, $ai \geq aj$, $ai < aj$ y $ai \leq aj$
- Se asume que todos los elementos son distintos

Cota inferior para los algoritmos de ordenamiento por comparaciones

Teorema

Para ordenar n elementos los algoritmos basados en comparaciones deben hacer $\Omega(n \log n)$ comparaciones en el peor caso.

Modelo de árbol de decisión

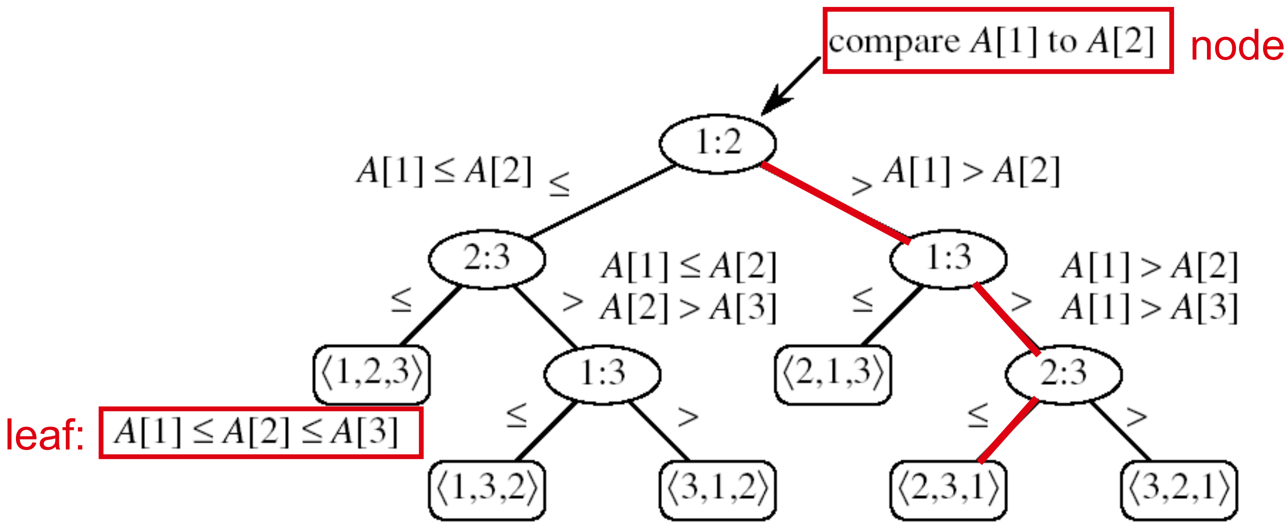

Figura: Árbol de decisión de la secuencia $\langle a_1 = 8, a_2 = 5, a_3 = 6\rangle$. Fuente [1]

Peor caso de comparaciones

El peor caso del número de comparaciones sucede cuando se recorre el camino más largo desde la raíz hasta un nodo hoja. Es decir, la altura del árbol.

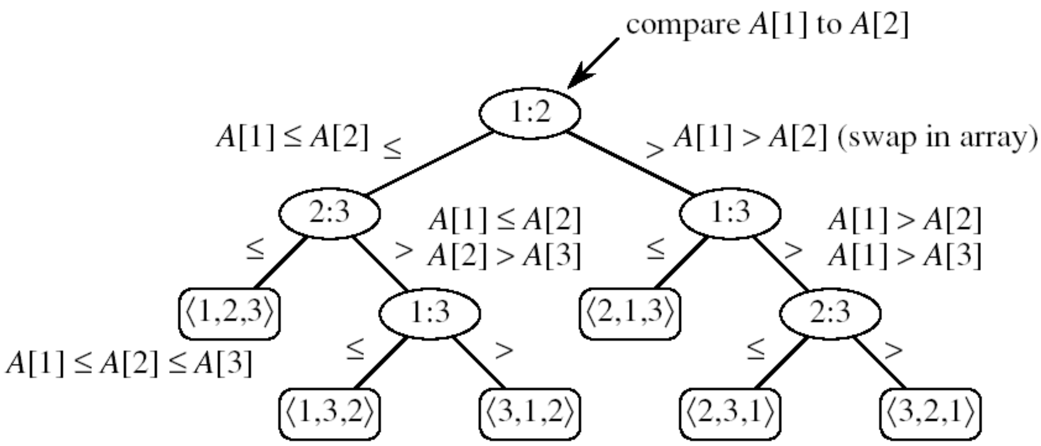

Figura: Árbol de decisión de la secuencia $\langle a_1 = 8, a_2 = 5, a_3 = 6 \rangle$. Fuente [1]

Peor caso de comparaciones

Todas las permutaciones de n elementos deben estar en las hojas del árbol por se posible resultado del algoritmo de ordenamiento. Por lo tanto hay $n!$ hojas.

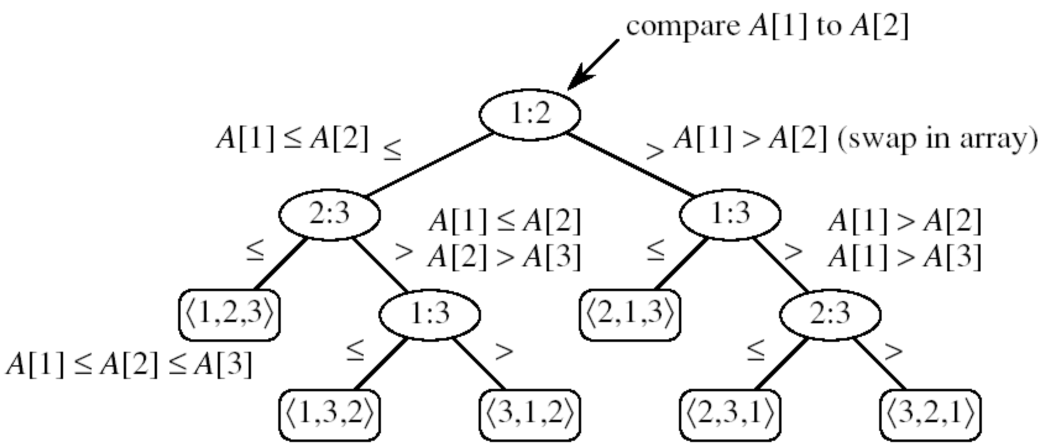

Figura: Árbol de decisión de la secuencia $\langle a_1 = 8, a_2 = 5, a_3 = 6 \rangle$. Fuente [1]

Cota inferior para los algoritmos de ordenamiento por comparaciones

Teorema

Para ordenar n elementos los algoritmos basados en comparaciones deben hacer $\Omega(n \log n)$ comparaciones en el peor caso.

Prueba:

- El árbol tiene al menos $n!$ hojas de las $n!$ permutaciones de la secuencia
- Se h la altura del árbol, entonces hay a lo sumo 2^h hojas
- Por lo tanto $n! \leq 2^h$
- $h \geq \log n! = \Omega(n \log n)$

Procedimiento `Counting-Sort`

Procedimiento Counting-Sort(A, B, k)

inicio
- $C[0..k] \leftarrow$ Crear un arreglo nuevo ;
- **para** $i \leftarrow 0$ **a** k **hacer**
  - $C[i] \leftarrow 0$
- **para** $j \leftarrow 1$ **a** $A.length$ **hacer**
  - $C[A[j]] \leftarrow C[A[j]] + 1$;
- **para** $i \leftarrow 1$ **a** k **hacer**
  - $C[i] \leftarrow C[i] + C[i-1]$;
- **para** $j \leftarrow A.length$ ***decrementando hasta*** *1* **hacer**
  - $B[C[A[j]]] \leftarrow A[j]$;
  - $C[A[j]] \leftarrow C[A[j]] - 1$;

Ejemplo de Counting Sort

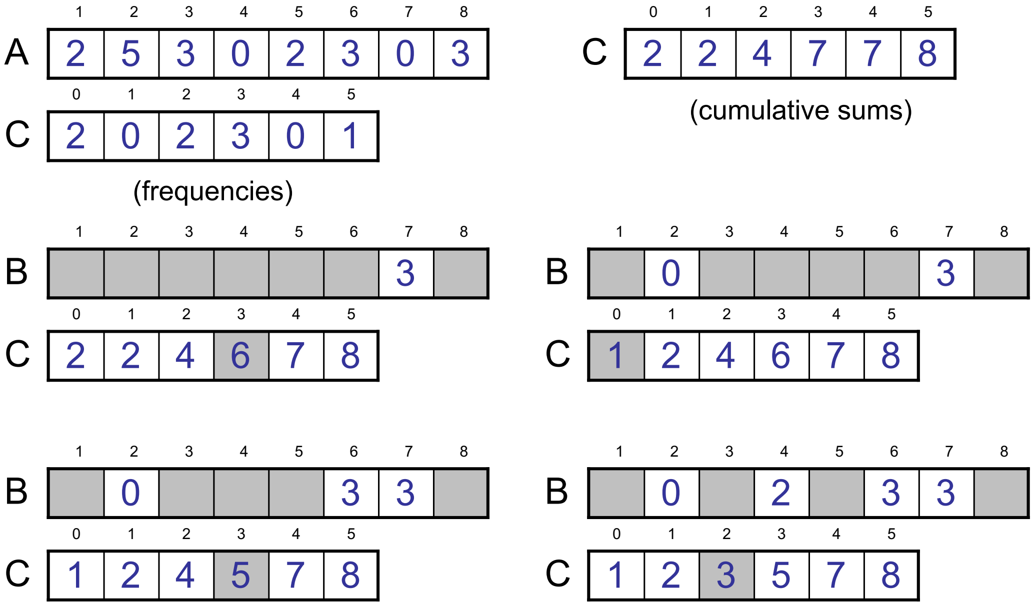

Figura: Operación de Counting sort en una secuencia $A[1..8]$. Fuente [1]

Ejemplo de Counting Sort cont.

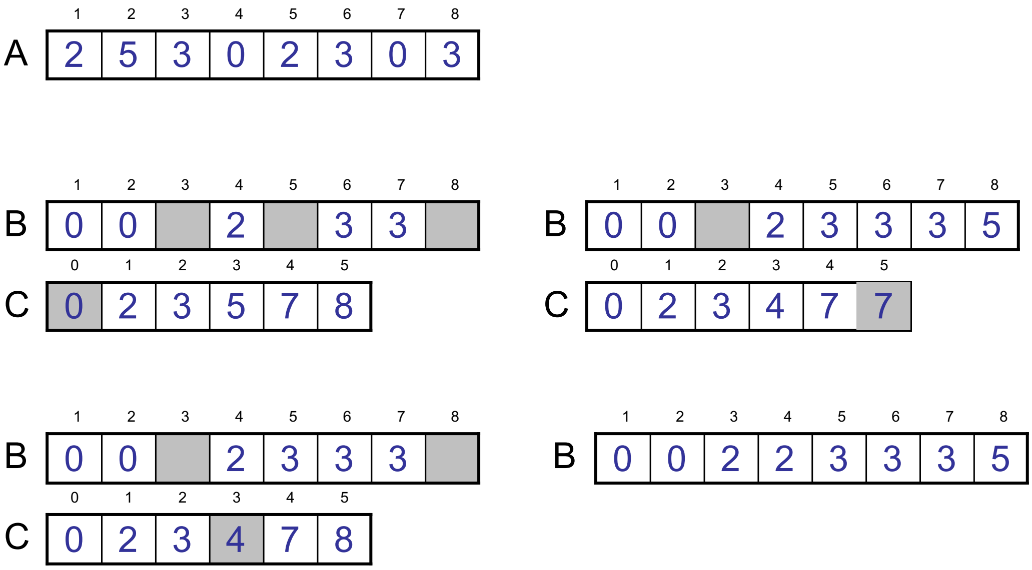

Figura: Operación de Counting sort en una secuencia $A[1..8]$. Fuente [1]

Tiempo de Counting-Sort

- Tiempo peor caso $\Theta(n + k)$
- Si $k = O(n)$ entonces es $\Theta(n)$ en el peor caso
- Counting sort es un algoritmo de ordenamiento estable

Procedimiento `Radix-Sort`

Procedimiento Radix-Sort(A, d)

inicio
  para $i \leftarrow 1$ **a** d **hacer**
    Ordenar A en el dígito i usando un algoritmo de ordenamiento estable (Ej. Counting Sort) ;

Ejemplo de Radix sort

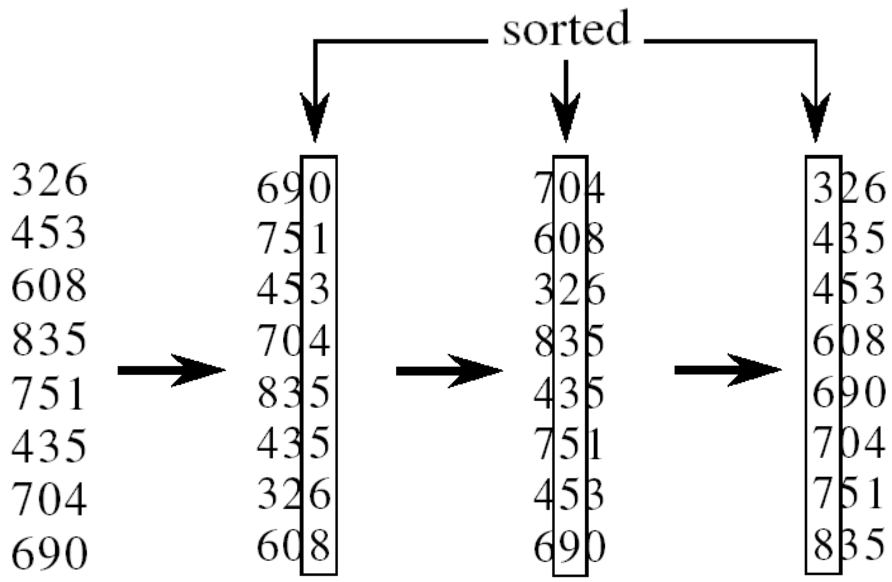

Figura: Operación de Radix sort en una secuencia con máximo tres dígitos d. Fuente [1]

Tiempo de Radix sort

- Se tienen d dígitos
- Para cada dígito se usa Counting sort que es $\Theta(n+k)$
- Como se ordena por cada d dígito el tiempo es $\Theta(d(n+k))$

Procedimiento `Bucket-Sort`

Procedimiento Bucket-Sort(A)

inicio

$n \leftarrow A$.length ;

$B[0..n-1] \leftarrow$ Crear un arreglo nuevo ;

para $i \leftarrow 0$ **a** $n-1$ **hacer**

  $B[i] \leftarrow$ Crear una lista vacía ;

para $i \leftarrow 1$ **a** n **hacer**

  Insertar $A[i]$ en la lista $B[\lfloor nA[i] \rfloor]$;

para $i \leftarrow 0$ **a** $n-1$ **hacer**

  Ordenar $B[i]$ usando InsertionSort ;

Concatenar las listas $B[0], B[1], \ldots, B[n-1]$ juntas en orden ;

Ejemplo de Bucket sort

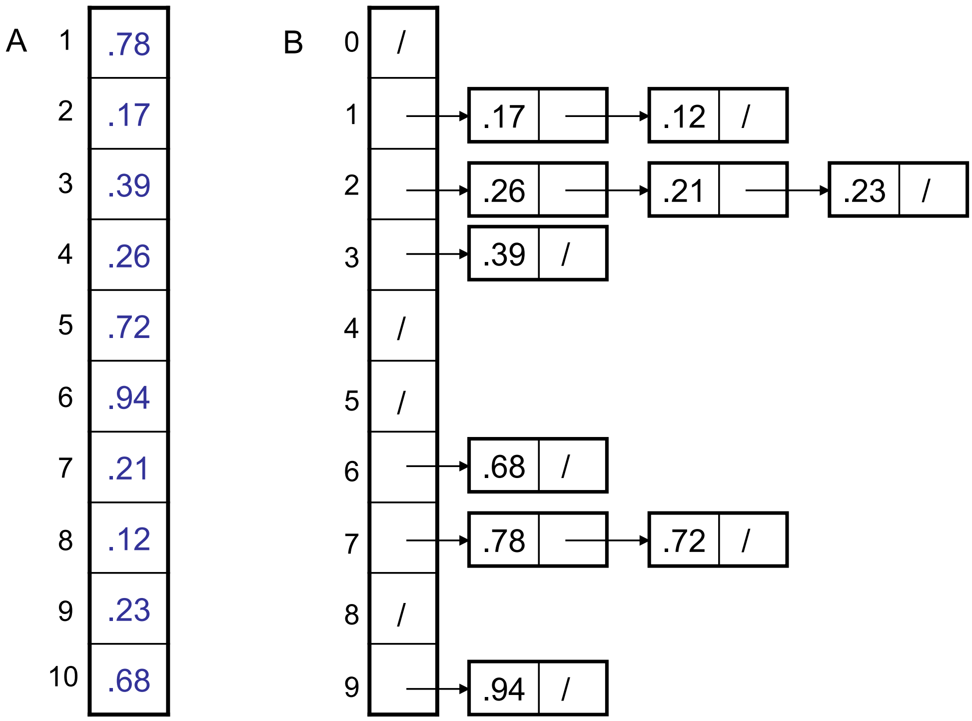

Figura: Operación de Bucket sort con $n = 10$. Fuente [1]

Ejemplo de Bucket sort cont.

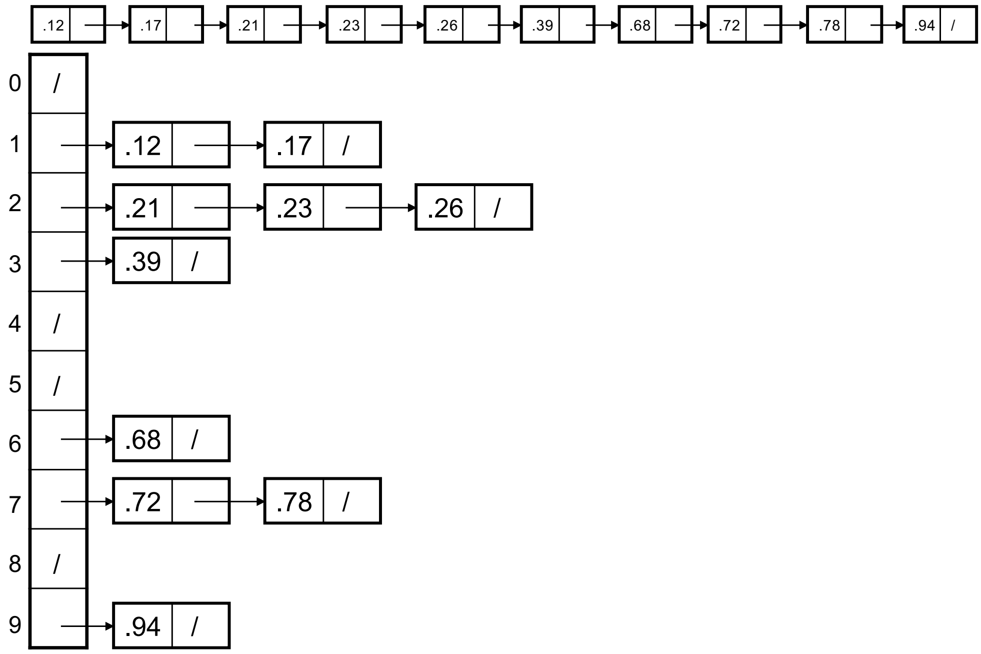

Figura: Operación de Bucket sort con $n = 10$. Fuente [1]

Tiempo de Bucket sort

- Inserción de los elementos en los buckets $O(n)$
- Ordenar las n listas con Insertionsort $\Theta(n)$
- Concatenar los elementos de las listas $O(n)$
- Por lo tanto es $\Theta(n)$

Criterio para la selección de un algoritmo de ordenamiento

Criteria	Sorting algorithm
Only a few items	INSERTION SORT
Items are mostly sorted already	INSERTION SORT
Concerned about worst-case scenarios	HEAP SORT
Interested in a good average-case result	QUICKSORT
Items are drawn from a dense universe	BUCKET SORT
Desire to write as little code as possible	INSERTION SORT

Figura: Criterio de selección basado en experimentos presentados en [2]

Referencias

T. Cormen, C. Leirserson, R. Rivest, and C. Stein.
Introduction to Algorithms.
McGraw Hill, 3ra edition, 2009.

G. Pollice, S. Selkow, and G. Heineman.
Algorithms in a Nutshell.
O'Reilly, 2009.