Analisis y Diseño de Algoritmos

Juan Gutiérrez

September 20, 2021

El Mergesort


```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

Figure: Tomada del libro Cormen, Introduction to Algorithms

El Mergesort

```
MERGE(A, p, q, r)
n1 = q − p + 1
n2 = r − q
let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i − 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

Figure: Tomada del libro Cormen, Introduction to Algorithms

El Mergesort

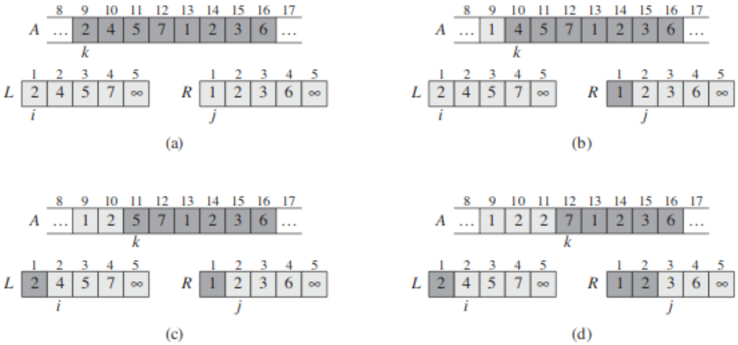
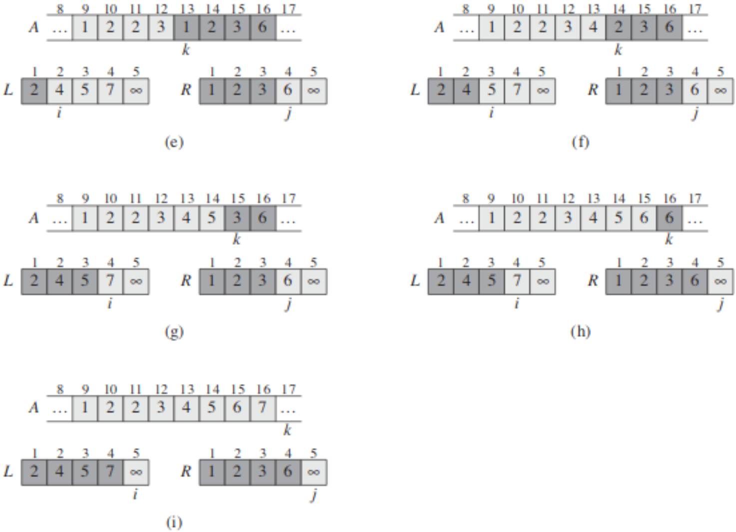


Figure: Tomada del libro Cormen, Introduction to Algorithms

El Mergesort



El Mergesort

Invariante: Al inicio de cada iteración del bucle **for** (líneas 12–17), el subvector $A[p..k-1]$ contiene a los $k-p$ elementos más pequeños entre $L[1..n_1+1]$ y $R[1..n_2+1]$ ordenados. También, $L[i]$ y $R[j]$ son los elementos más pequeños que no han sido copiados.

El Mergesort

- **Inicialización** $k = p$, luego $A[p..k-1] = \emptyset$
- **Manuntención**

  Caso 1: $L[i] \leq R[j]$. Entonces se ejecuta la línea 14. Como $A[p..k-1]$ estaba ordenado con los menores elementos, entonces $A[p..k-1]$ tendrá los $k-p+1$ elementos menores. Caso 2: $L[i] > R[j]$: similar.
- **Terminación**

  $k = r+1$. Luego $A[p..k-1] = A[p..r]$ contiene los $k-p = r-p+1 = n_1 + n_2 - 2$ elementos más pequeños de $L[1..n_1+1]$ y $R[1..n_2+1]$. Osea todos excepto los sentinelas.

El Método de división y conquista

- **Dividir** el problema en subproblemas que son instancias del mismo problema.

  (En MergeSort: dividir el arreglo de tamaño n en dos subsecuencias de tamaño $n/2$.)

- **Conquistar**: Resolver los subproblemas recursivamente. Si el tamaño es pequeño, resolverlos directamente.

  (En MergeSort: ordenar las dos subsecuencias usando MergeSort.)

- **Combinar** las soluciones de los subproblemas en una solución para el problema original.

  (En MergeSort: Mezclar las dos subsecuencias ordenadas.)

El Método de división y conquista

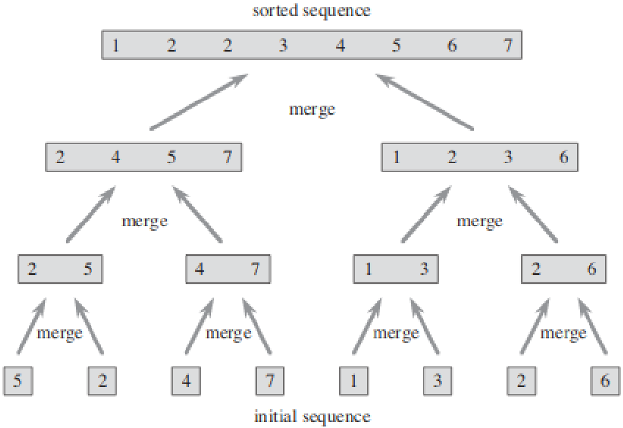


Figure: Tomada del libro Cormen, Introduction to Algorithms.

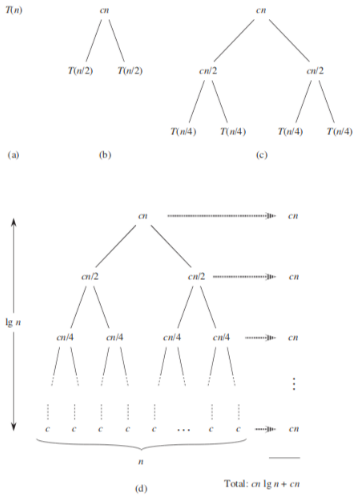
Análisis del tiempo de ejecución

$$T(n) = aT(n/b) + D(n) + C(n),$$

Análisis del tiempo de ejecución

$$T(n) = 2T(n/2) + k_2 n + k_1 = 2T(n/2) + cn.$$

Análisis del tiempo de ejecución



Recurrencias

Ejemplo 3.1. *Sea* $F : \mathbb{N} \to \mathbb{R}^{>}$ *definido por*

$$F(n) = \begin{cases} 1 & n = 1 \\ 2F(n-1) + 1 & \textit{caso contrário} \end{cases}$$

Por ejemplo $F(2) = 2F(1) + 1 = 3, F(3) = 2F(2) + 1 = 7, F(4) = 15$. *¿Cuanto vale* $F(n)$*?*

Recurrencias

Ejemplo 3.2. *Sea* $F : \mathbb{N} \to \mathbb{R}^{>}$ *definido por*

$$F(n) = \begin{cases} 1 & n = 1 \\ F(n-1) + n & \textit{caso contrário} \end{cases}$$

Por ejemplo $F(2) = F(1) + 2 = 3, F(3) = F(2) + 3 = 6$. *Cuanto vale* $F(n)$*?*

Recurrencias

Ejemplo 3.3. *Sea* $F : \mathbb{N} \to \mathbb{R}^{>}$ *definido por*

$$F(n) = \begin{cases} 1 & n = 1 \\ 2F(\lfloor n/2 \rfloor) + n & \textit{caso contrário} \end{cases}$$

Por ejemplo $F(2) = 2F(1) + 2 = 4, F(3) = 2F(1) + 3 = 5$. *Como podemos acotar* $F(n)$*?*

Recurrencias: prueba por inducción

Ejemplo 3.4. *Sea* $F : \mathbb{N} \to \mathbb{R}^{>}$ *definido por*

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(\lfloor \frac{n}{2} \rfloor) + n & \textit{caso contrário} \end{cases}$$

Compruebe por inducción que $T(n) = O(n^2)$. *Compruebe por inducción que* $T(n) = \Omega(n \lg n)$.

Recurrencias: prueba por inducción

Ejemplo 3.5. *Sea* $F : \mathbb{N} \to \mathbb{R}^{>}$ *definido por*

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(\lfloor \frac{n}{2} \rfloor) + 1 & \textit{caso contrário} \end{cases}$$

Compruebe por inducción que $T(n) = O(n)$.

Teorema Maestro

Sean $a \geq 1$, $b \geq 2$, $k \geq 0$, $n_0 \geq 1 \in \mathbb{N}$. Sea $c \in \mathbb{R}^{>}$. Sea $F : \mathbb{N} \to \mathbb{R}^{>}$ una función no decreciente tal que

$$F(n) = aF(n/b) + cn^k$$

para $n = n_0 b^1, n_0 b^2, n_0 b^3, \ldots$. Se cumple que

- Si $\lg a / \lg b > k$ entonces $F(n) = \Theta(n^{\lg a / \lg b})$
- Si $\lg a / \lg b = k$ entonces $F(n) = \Theta(n^k \lg n)$
- Si $\lg a / \lg b < k$ entonces $F(n) = \Theta(n^k)$

Cuando $b = 2$ tenemos que

- Si $\lg a > k$ entonces $F(n) = \Theta(n^{\lg a})$
- Si $\lg a = k$ entonces $F(n) = \Theta(n^k \lg n)$
- Si $\lg a < k$ entonces $F(n) = \Theta(n^k)$

Gracias