Análisis de un algoritmo en el caso promedio

IIC2283 – Diseño y Análisis de Algoritmos

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

2025-2

Analizando la complejidad de un algoritmo

Hasta ahora hemos analizado la complejidad de un algoritmo considerando el peor caso.

También tiene sentido estudiar la complejidad del algoritmo $\mathcal{A}$ en el caso promedio.

El tiempo promedio para entradas de tamaño n está dado por la siguiente expresión:

$$\bar{t}(n) = \frac{1}{|\mathcal{I}_{\mathcal{P},n}|} \cdot \left(\sum_{w \in \mathcal{I}_{\mathcal{P},n}} \text{tiempo}_{\mathcal{A}}(w) \right)$$

donde $\mathcal{I}_{\mathcal{P},n} = \{w \in \mathcal{I}_{\mathcal{P}} \mid |w| = n\}$ **[Las instancias de tamaño n]**

El caso promedio de un algoritmo

La definición anterior asume que todas las entradas son igualmente probables.

- Esto podría no reflejar la distribución de las entradas en la práctica

Para solucionar este problema podemos usar otras distribuciones de probabilidad.

El caso promedio de un algoritmo

Suponemos que para cada $n \in \mathbb{N}$ hay una distribución de probabilidades:

$\Pr_n(w)$ es la probabilidad de que $w \in \mathcal{I}_{\mathcal{P},n}$ aparezca como entrada de $\mathcal{A}$

Nótese que $\sum_{w \in \mathcal{I}_{\mathcal{P},n}} \Pr_n(w) = 1$

Ejemplo

Para la definición de caso promedio en las transparencias anteriores tenemos que $\Pr_n(w) = \frac{1}{|\mathcal{I}_{\mathcal{P},n}|}$

El caso promedio de un algoritmo

Para definir el caso promedio, para cada $n \in \mathbb{N}$ usamos una variable aleatoria X_n

- Para cada $w \in \mathcal{I}_{\mathcal{P},n}$ se tiene que $X_n(w) = \mathit{tiempo}_{\mathcal{A}}(w)$

Para las entradas de largo n, el número de pasos de $\mathcal{A}$ en el caso promedio es el valor esperado de la variable aleatoria X_n:

$$\mathrm{E}(X_n) \quad = \quad \sum_{w \in \mathcal{I}_{\mathcal{P},n}} X_n(w) \cdot \mathrm{Pr}_n(w)$$

Definición (Complejidad en el caso promedio)

Decimos que $\mathcal{A}$ en el caso promedio es $O(f(n))$ si $E(X_n) \in O(f(n))$

Sobre la definición en el caso promedio

Notación

La definición del caso promedio puede ser modificada para considerar las notaciones Θ y Ω

- Simplemente reemplazando $O(f(n))$ por $\Theta(f(n))$ u $\Omega(f(n))$, respectivamente

Por ejemplo, decimos que $\mathcal{A}$ en el caso promedio es $\Theta(f(n))$ si $\mathrm{E}(X_n) \in \Theta(f(n))$

Un ejemplo: el algoritmo de ordenación Quicksort

Quicksort es un algoritmo de ordenación muy utilizado en la práctica

Conceptualmente, sobre un arreglo $L[1..n]$ el algoritmo funciona así

- Si $n \leq 1$, L está trivialmente ordenado.
- En otro caso:
  - Sea p un elemento arbitrario del arreglo L, el *pivote*
  - Sea $L_{\min} = \{x \in L \mid x < p\}$
  - Sea $L_{\max} = \{x \in L \mid x > p\}$
  - Se aplica $\mathit{Quicksort}(L_{\min})$ y $\mathit{Quicksort}(L_{\max})$ recursivamente para ordenar $L_{\min}$ y $L_{\max}$
  - Luego de ordenar las partes, se produce el arreglo ordenado concatenando :

$$L_{\min} + \langle p \rangle + L_{\max}$$

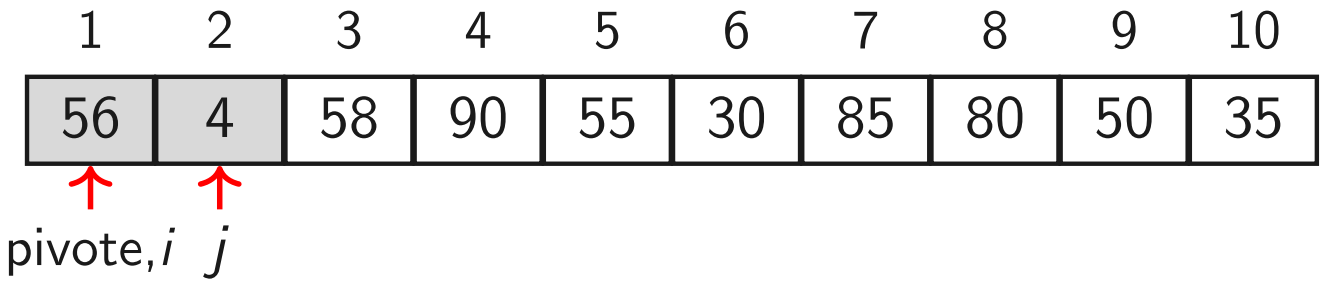
Un ejemplo: el algoritmo de ordenación Quicksort

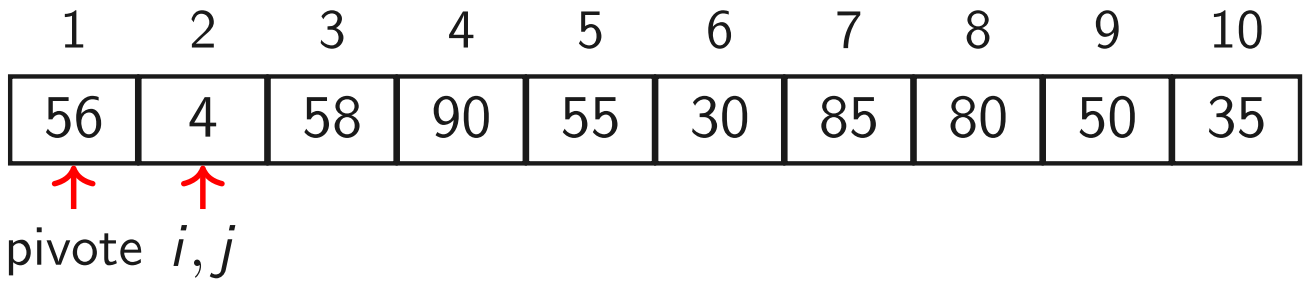
En una implementación eficiente de Quicksort, la función clave es la siguiente:

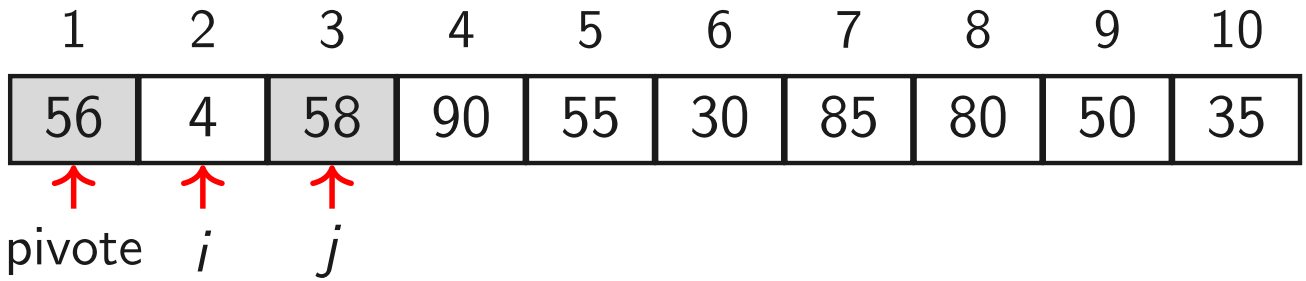
Partición(L, m, n)
    $pivote := L[m]$
    $i := m$
    for $j := m + 1$ **to** n **do**
        if $L[j] \leq pivote$ **then**
            $i := i + 1$
            intercambiar $L[i]$ con $L[j]$
    intercambiar $L[m]$ con $L[i]$
    return i

Nótese que en la definición de **Partición** la lista L es pasado por referencia

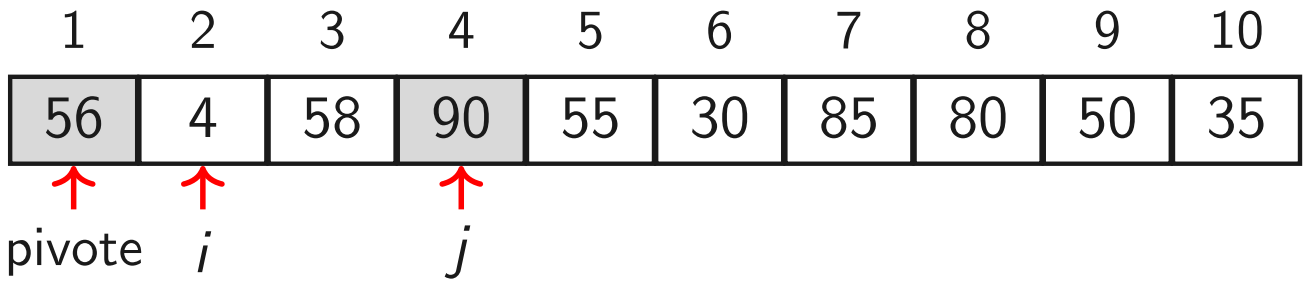
Un ejemplo: el algoritmo de ordenación Quicksort

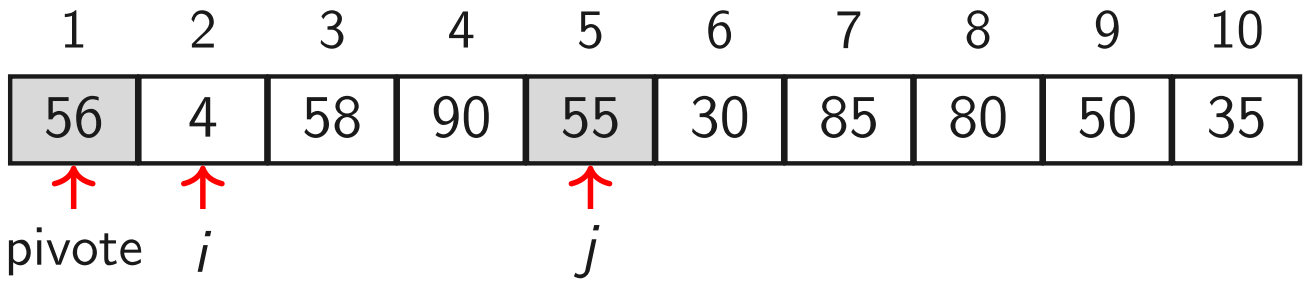


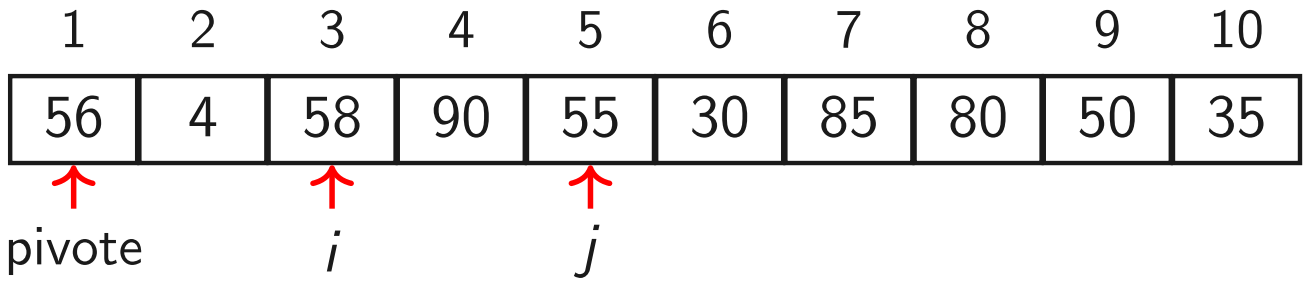




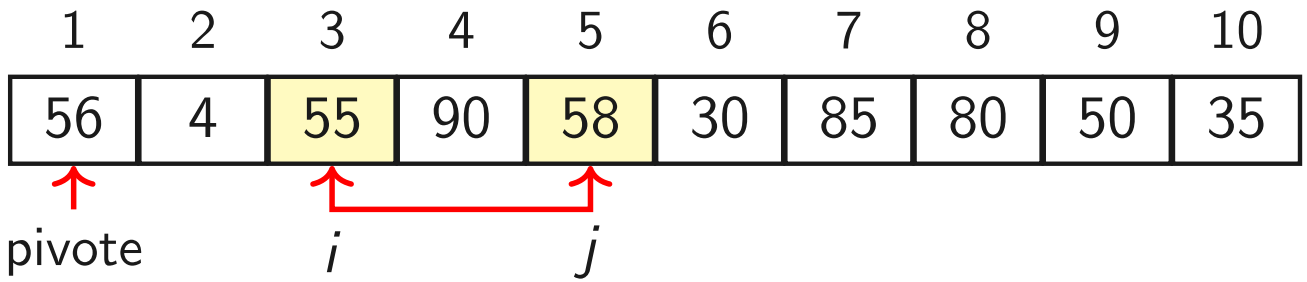
Un ejemplo: el algoritmo de ordenación Quicksort

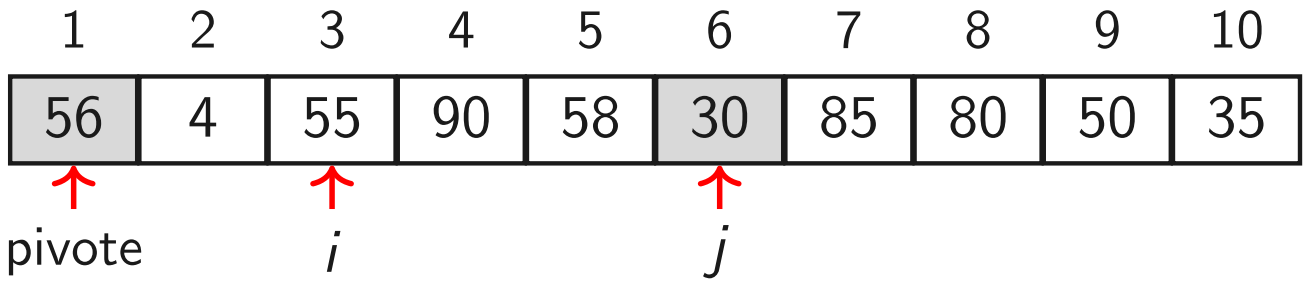


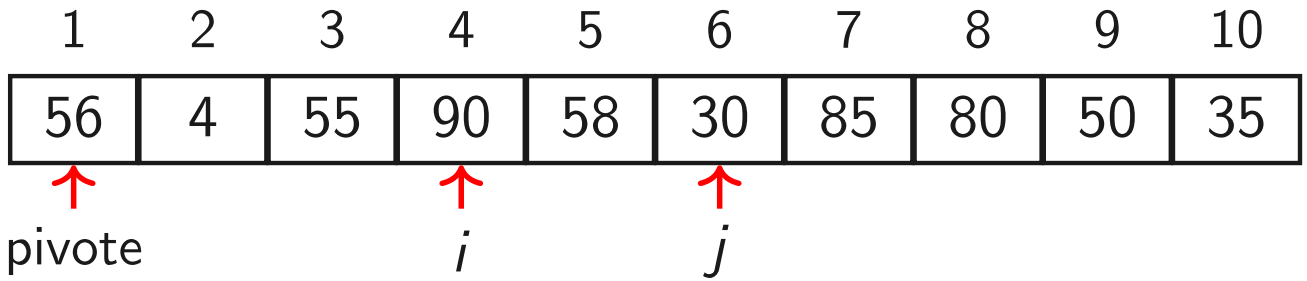




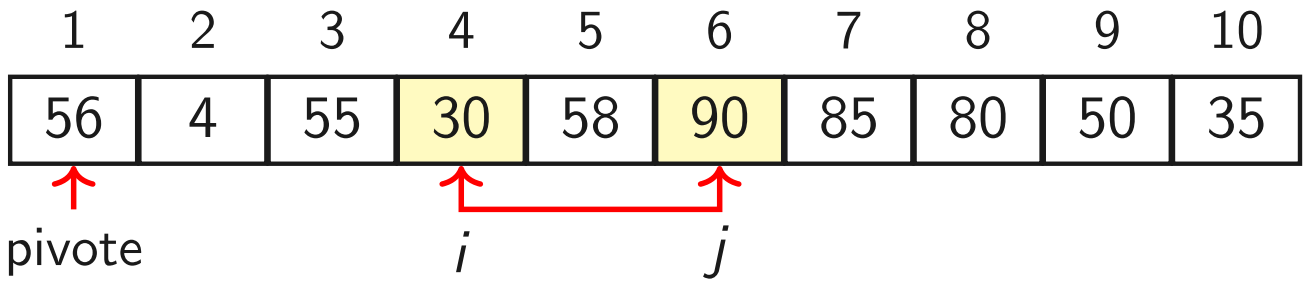
Un ejemplo: el algoritmo de ordenación Quicksort

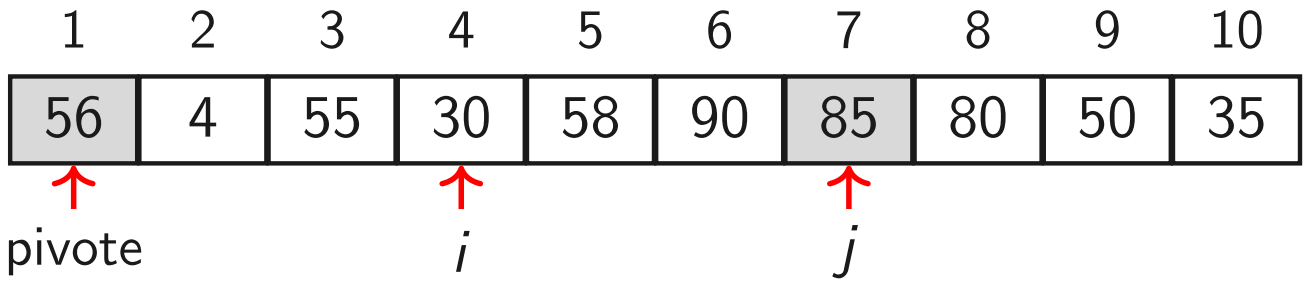


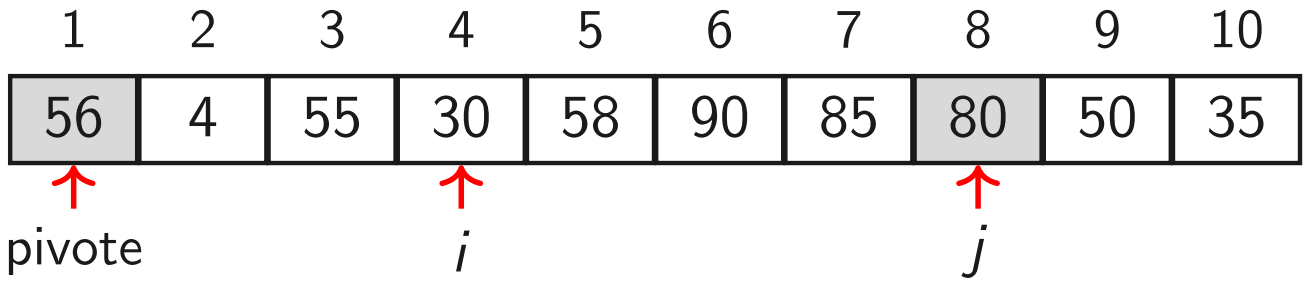




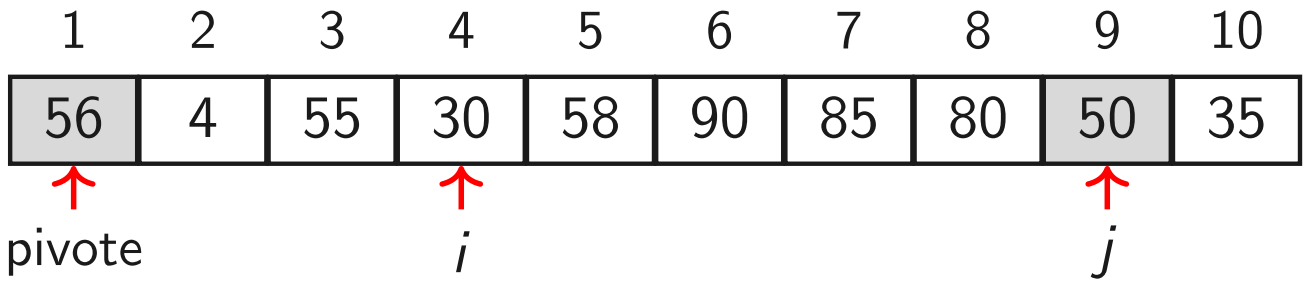
Un ejemplo: el algoritmo de ordenación Quicksort

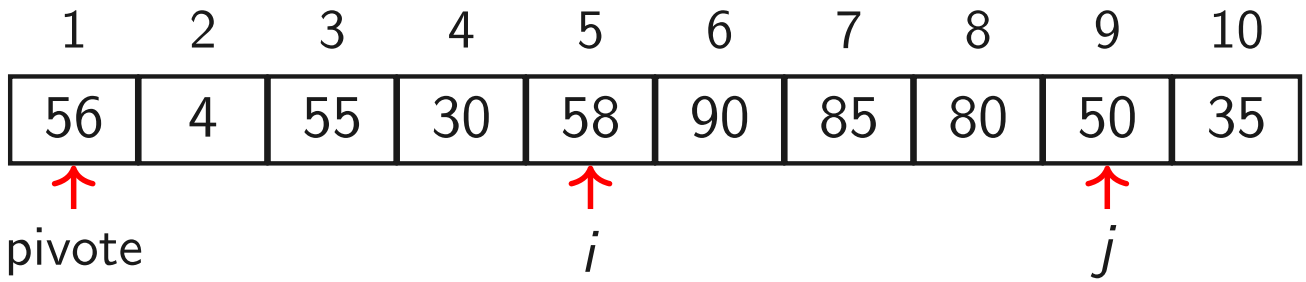


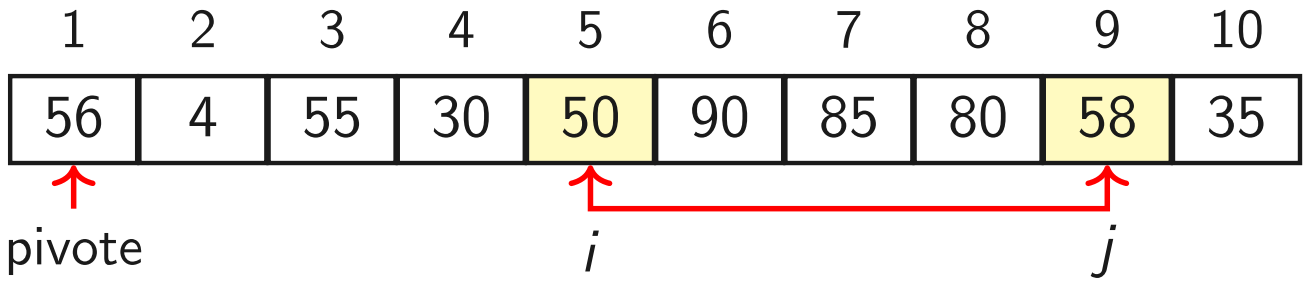




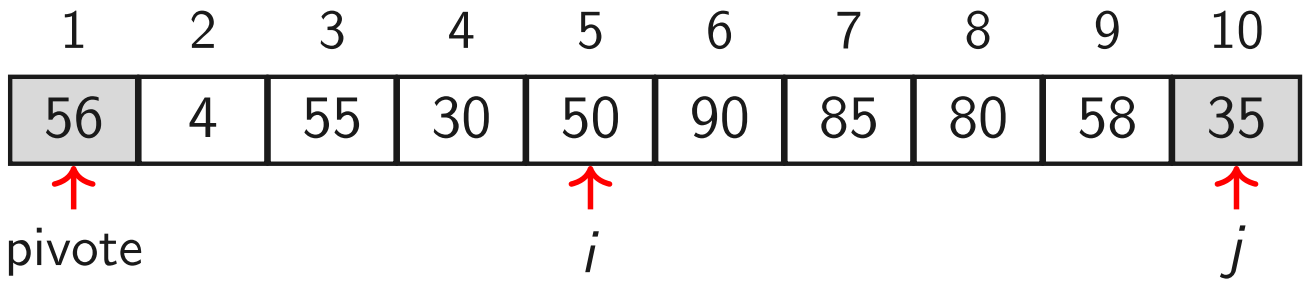
Un ejemplo: el algoritmo de ordenación Quicksort

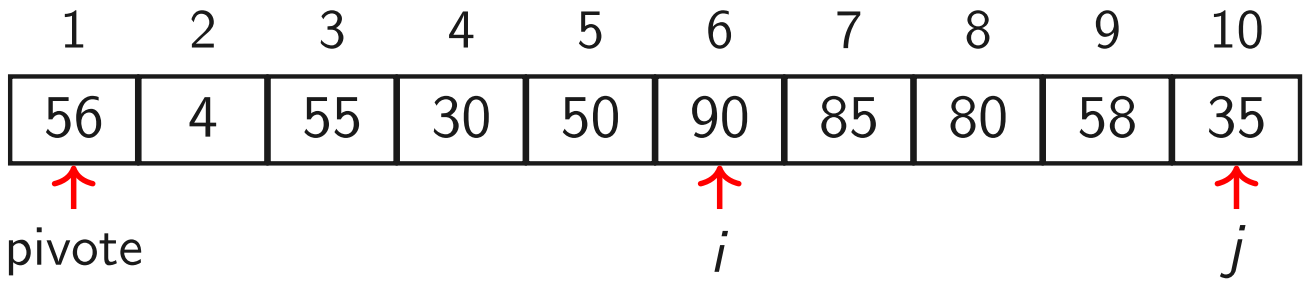


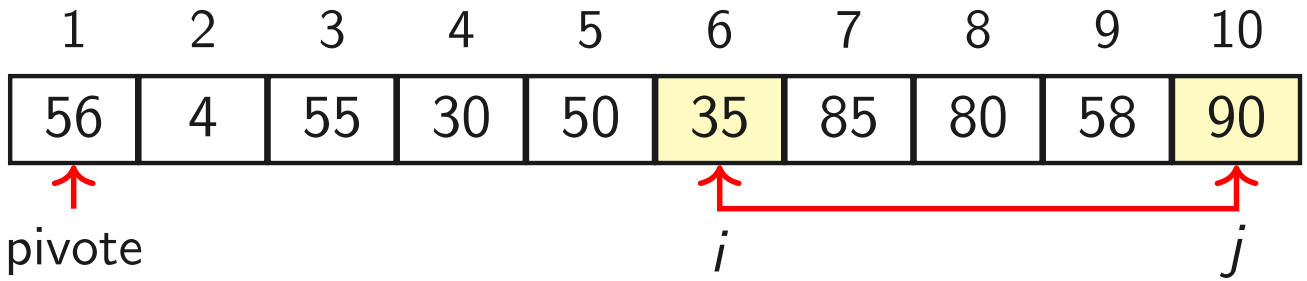




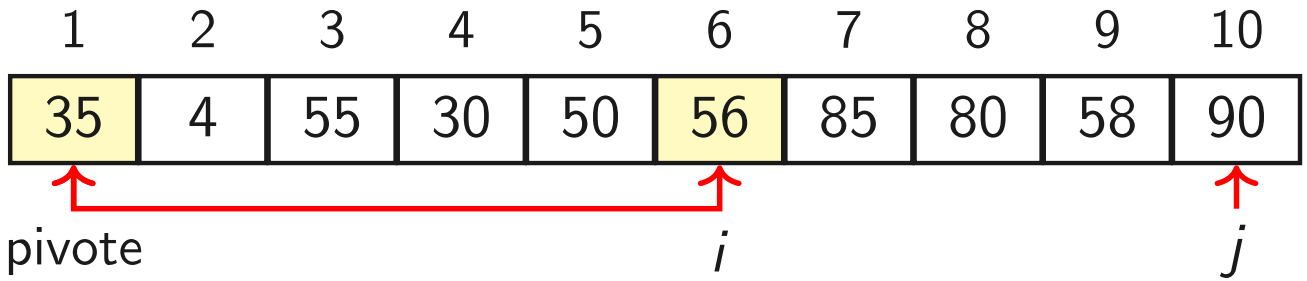
Un ejemplo: el algoritmo de ordenación Quicksort



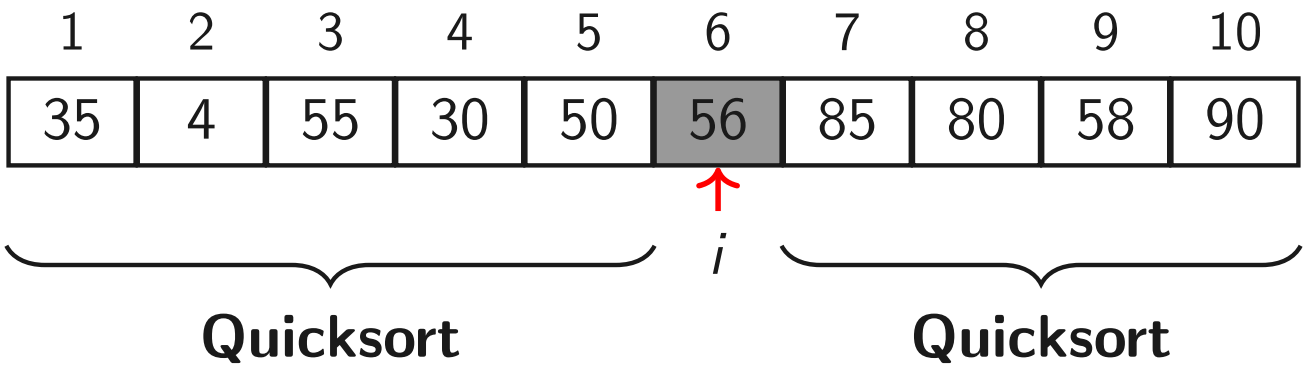




Un ejemplo: el algoritmo de ordenación Quicksort



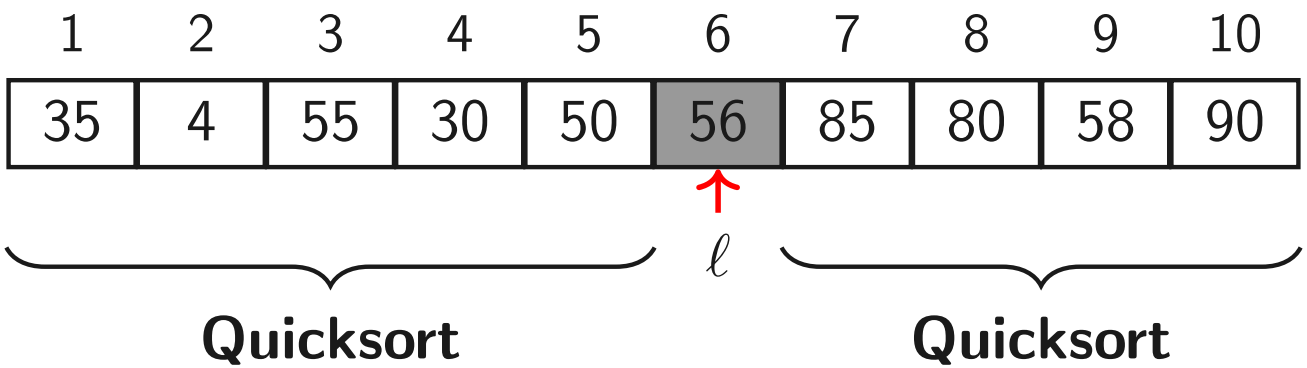
Luego de particionar el arreglo en base al pivote, se continua recursivamente en cada una de las partes:



Un ejemplo: el algoritmo de ordenación Quicksort

La definición de Quicksort:

Quicksort$(L,\ m,\ n)$
    if $m < n$ **then**
        $\ell :=$ **Partición**$(L,\ m,\ n)$
        Quicksort$(L,\ m,\ \ell - 1)$
        Quicksort$(L,\ \ell + 1,\ n)$



¿Cuál es la complejidad de Quicksort?

Vamos a contar el número de comparaciones al medir la complejidad de **Quicksort**.

- ¿Es ésta una buen medida de complejidad?

Sí **Partición** siempre divide a una lista en dos listas del mismo tamaño, entonces la complejidad de **Quicksort** estaría dada por la siguiente ecuación de recurrencia:

$$T(n) = \begin{cases} 1 & n = 0 \\ 2 \cdot T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + c \cdot n & n \geq 1 \end{cases}$$

¿Cuál es la complejidad de Quicksort?

Dado que $c \cdot n \in \Theta(n^{\log_2(2)})$, concluimos usando el Teorema Maestro que $T(n) \in \Theta(n \cdot \log_2(n))$

¿Pero qué nos asegura que **Partición** divide a una lista en dos listas del mismo tamaño?

Ejercicio

1. Dada una lista con n elementos, ¿cuál es el peor caso para **Quicksort**?
2. Demuestre que **Quicksort** en el peor caso es $\Theta(n^2)$

¿Por qué usamos Quicksort entonces?

Vamos a demostrar que **Quicksort** en el caso promedio es $\Theta(n \cdot \log_2(n))$

- Suponiendo una distribución uniforme en las entradas

Vamos a considerar listas sin elementos repetidos.

- Usted va a tener que pensar cómo obtener la misma complejidad para listas con elementos repetidos

La complejidad de Quicksort en el caso promedio

Sea $n \in \mathbb{N}$ tal que $n \geq 2$, y sea $\mathcal{E}_n$ el conjunto de listas L con n elementos distintos sacados desde el conjunto $\{1, \ldots, n\}$

- Tenemos que $|\mathcal{E}_n| = n!$

Para cada $L \in \mathcal{E}_n$ asumimos lo siguiente:

- $\Pr_n(L) = \frac{1}{n!}$
  [Es decir, asumimos que todas las permutaciones tienen la misma probabilidad de ocurrir]
- $X_n(L)$ es el número de comparaciones realizadas por la llamada **Quicksort**$(L, 1, n)$

La complejidad de Quicksort en el caso promedio

La complejidad de **Quicksort** en el caso promedio está dada por $\mathrm{E}(X_n)$

¿Por qué nos restringimos a las listas con elementos sacados desde el conjunto $\{1, \ldots, n\}$?

- ¿En qué sentido estamos considerando todas las listas posibles con n elementos (sin repeticiones)?
- ¿Cómo refleja esto el hecho de que estamos considerando las entradas de un cierto largo?

Calculando el valor esperado de X_n

Sea $L \in \mathcal{E}_n$

Para cada $i, j \in \{1, \ldots, n\}$ con $i \leq j$ defina la siguiente variable aleatoria:

$Y_{i,j}(L)$: número de veces que i es comparado con j en la llamada **Quicksort**$(L, 1, n)$

Entonces tenemos lo siguiente:

$$X_n(L) = \sum_{i=1}^{n} \sum_{j=i}^{n} Y_{i,j}(L)$$

Calculando el valor esperado de X_n

Dado que el valor esperado de una variable aleatoria es una función lineal, concluimos que:

$$\mathrm{E}(X_n) \quad = \quad \sum_{i=1}^{n} \sum_{j=i}^{n} \mathrm{E}(Y_{i,j})$$

Para calcular $\mathrm{E}(X_n)$ basta entonces calcular $\mathrm{E}(Y_{i,j})$ para cada $i, j \in \{1, \ldots, n\}$ con $i \leq j$

Calculando el valor esperado de $Y_{i,j}$

Para determinar $\mathrm{E}(Y_{i,j})$ para cada $i, j \in \{1, \ldots, n\}$ con $i \leq j$, analizamos el proceso de **Partición**

En nuestro análisis, sea p el elemento elegido como pivote

Supongamos que i y j se encuentran en la misma partición del arreglo

Hay que considerar los siguientes casos

Calculando el valor esperado de $Y_{i,j}$

Caso i) El pivote es tal que

$$i < p < j,$$

por lo que i y j van a quedar en particiones distintas y **Quicksort** nunca va a compararlos a lo largo de esta ejecución.

En este caso tenemos $Y_{i,j}(L) = 0$

Caso ii) El pivote es tal que

$$p < i < j \quad \text{o} \quad i < j < p,$$

por lo que i y j no se comparan y quedan en la misma partición para la llamada recursiva (podrían compararse más adelante)

Caso iii) El pivote es tal que

$$p = i < j \quad \textbf{o} \quad i < j = p,$$

por lo que i y j son comparados. El algoritmo no vuelve a compararlos

En este caso tenemos $Y_{i,j}(L) = 1$

Calculando el valor esperado de $Y_{i,j}$

Podemos concluir que $Y_{i,j}(L)$ vale 0 o 1

Es una variable aleatoria indicatriz (indicator random variable)

Notar que

$$\begin{aligned} \mathrm{E}(Y_{i,j}) &= 0 \cdot \Pr(Y_{i,j} = 0) + 1 \cdot \Pr(Y_{i,j} = 1) \\ &= \Pr(Y_{i,j} = 1). \end{aligned}$$

Por lo tanto, nos resta calcular $\Pr(Y_{i,j} = 1)$

Calculando el valor esperado de $Y_{i,j}$

Para determinar $\Pr(Y_{i,j} = 1)$, pensemos en el conjunto $Z_{i,j} = \{i, i+1, \ldots, j\}$, de $j - i + 1$ elementos

- Todos los elementos de $Z_{i,j}$ permanecen en la misma partición del arreglo hasta la primera vez que uno de los elementos de $Z_{i,j}$ actúa como pivote.
- En ese caso, $Z_{i,j}$ se particiona y ya no todos sus elementos quedan en la misma partición.
- Como hemos asumido que todas las permutaciones de $\mathcal{E}_n$ son igualmente probables, tenemos que cuando el pivote pertenece a $Z_{i.j}$, cada elemento tiene probabilidad $1/(j - i + 1)$ de ser el pivote.
- $Y_{i,j} = 1$ cuando i o j son elegidos como pivotes (recuerde el **Caso iii**), por lo que

$$\Pr(Y_{i,j} = 1) = \frac{1}{j - i + 1} + \frac{1}{j - i + 1} = \frac{2}{j - i + 1}$$

- Concluimos que

$$\mathrm{E}(Y_{i,j}) = \frac{2}{j - i + 1}$$

El cálculo final

Concluimos que:

$$
\begin{aligned}
\mathrm{E}(X_n) &= \sum_{i=1}^{n}\sum_{j=i}^{n}\mathrm{E}(Y_{i,j}) \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mathrm{E}(Y_{i,j}) \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1}\sum_{k=2}^{n-i+1}\frac{2}{k} \\
&= \sum_{k=2}^{n}(n+1-k)\cdot\frac{2}{k} \\
&= 2\cdot(n+1)\cdot\left(\sum_{k=2}^{n}\frac{1}{k}\right)-2\cdot(n-1) \\
&= 2\cdot(n+1)\cdot\left(\sum_{k=1}^{n}\frac{1}{k}\right)-4\cdot n
\end{aligned}
$$

La sumatoria armónica

Para terminar el calculo de $\mathrm{E}(X_n)$ tenemos que acotar la sumatoria armónica $\sum_{k=1}^{n} \frac{1}{k}$

Tenemos que:

$$\sum_{k=2}^{n} \frac{1}{k} \leq \int_{1}^{n} \frac{1}{x}\, dx \leq \sum_{k=1}^{n} \frac{1}{k}$$

Dado que $\int_{1}^{n} \frac{1}{x}\, dx = \ln(n) - \ln(1) = \ln(n)$, concluimos que:

$$\ln(n) \leq \sum_{k=1}^{n} \frac{1}{k} \leq \ln(n) + 1$$

La sumatoria armónica

Por lo tanto:

$$2 \cdot (n+1) \cdot \ln(n) - 4 \cdot n \;\leq\; \mathrm{E}(X_n) \;\leq\; 2 \cdot (n+1) \cdot (\ln(n)+1) - 4 \cdot n$$

De lo cual concluimos que $\mathrm{E}(X_n) \in \Theta(n \cdot \log_2(n))$

- Vale decir, **Quicksort** en el caso promedio es $\Theta(n \cdot \log_2(n))$