Quick Sort

18-Sort

Algoritmo

Algoritmo

```
Partition (A, i, f):
    x ← índice aleatorio en {i, . . . , f}
    p ← A[x]
    A[x] ⇄ A[f]
    j ← i
    for k = i . . . f − 1 :
        if A[k] < p :
            A[j] ⇄ A[k]
            j ← j + 1
    A[j] ⇄ A[f]
    return j
```

Elección del pivote

```
Partition (A, i, f):
    x ← índice aleatorio en {i, . . . , f}
    p ← A[x]
    A[x] ⇄ A[f]
    j ← i
    for k = i . . . f − 1 :
        if A[k] < p :
            A[j] ⇄ A[k]
            j ← j + 1
    A[j] ⇄ A[f]
    return j
```

Elección del pivote

- Aleatorio

`Partition` (A, i, f):

$x \leftarrow$ índice aleatorio en $\{i, \ldots, f\}$
$p \leftarrow A[x]$
$A[x] \rightleftarrows A[f]$
$j \leftarrow i$
for $k = i \ldots f - 1$:
    if $A[k] < p$:
        $A[j] \rightleftarrows A[k]$
        $j \leftarrow j + 1$
$A[j] \rightleftarrows A[f]$
return j

Elección del pivote

```
Partition (A, i, f):
    x ← índice aleatorio en {i, ..., f}
    p ← A[x]
    A[x] ⇄ A[f]
    j ← i
    for k = i ... f − 1 :
        if A[k] < p :
            A[j] ⇄ A[k]
            j ← j + 1
    A[j] ⇄ A[f]
    return j
```

- Aleatorio
- El primer elemento

Elección del pivote

Partition (A, i, f):

$x \leftarrow$ índice aleatorio en $\{i, \ldots, f\}$

$p \leftarrow A[x]$

$A[x] \rightleftarrows A[f]$

$j \leftarrow i$

for $k = i \ldots f - 1$:

    if $A[k] < p$:

        $A[j] \rightleftarrows A[k]$

        $j \leftarrow j + 1$

$A[j] \rightleftarrows A[f]$

return j

- Aleatorio
- El primer elemento
- El último elemento

Elección del pivote

```
Partition (A, i, f):
    x ← índice aleatorio en {i, . . . , f}
    p ← A[x]
    A[x] ⇄ A[f]
    j ← i
    for k = i . . . f − 1 :
        if A[k] < p :
            A[j] ⇄ A[k]
            j ← j + 1
    A[j] ⇄ A[f]
    return j
```

- Aleatorio
- El primer elemento
- El último elemento
- El elemento central

Elección del pivote

Partition (A, i, f):

$x \leftarrow$ índice aleatorio en $\{i, \ldots, f\}$
$p \leftarrow A[x]$
$A[x] \rightleftarrows A[f]$
$j \leftarrow i$
for $k = i \ldots f - 1$:
  if $A[k] < p$:
    $A[j] \rightleftarrows A[k]$
    $j \leftarrow j + 1$
$A[j] \rightleftarrows A[f]$
return j

- Aleatorio
- El primer elemento
- El último elemento
- El elemento central
- Mediana entre los 3 anteriores

Elección del pivote

Partition (A, i, f):

$x \leftarrow$ índice aleatorio en $\{i, \dots, f\}$

$p \leftarrow A[x]$

$A[x] \rightleftarrows A[f]$

$j \leftarrow i$

for $k = i \dots f - 1$:

  if $A[k] < p$:

    $A[j] \rightleftarrows A[k]$

    $j \leftarrow j + 1$

$A[j] \rightleftarrows A[f]$

return j

- Aleatorio
- El primer elemento
- El último elemento
- El elemento central
- Mediana entre los 3 anteriores
- El que se les pueda ocurrir bajo criterios buenos

Algoritmo

Partition (A, i, f):

$x \leftarrow$ índice aleatorio en $\{i, \ldots, f\}$
$p \leftarrow A[x]$
$A[x] \rightleftarrows A[f]$
$j \leftarrow i$
for $k = i \ldots f - 1$:
    if $A[k] < p$:
        $A[j] \rightleftarrows A[k]$
        $j \leftarrow j + 1$
$A[j] \rightleftarrows A[f]$
return j

Algoritmo

`Partition` (A, i, f):

$x \leftarrow$ índice aleatorio en $\{i, \ldots, f\}$
$p \leftarrow A[x]$
$A[x] \rightleftarrows A[f]$
$j \leftarrow i$
for $k = i \ldots f - 1$:
    if $A[k] < p$:
        $A[j] \rightleftarrows A[k]$
        $j \leftarrow j + 1$
$A[j] \rightleftarrows A[f]$
return j

`QuickSort` (A, i, f):

if $i \leq f$:
    $p \leftarrow$ `Partition`(A, i, f)
    `Quicksort`$(A, i, p - 1)$
    `Quicksort`$(A, p + 1, f)$

Visualicemos el algoritmo

- https://visualgo.net/en/sorting

- https://www.youtube.com/watch?v=8hEyhs3OV1w

Complejidad Asintótica

Complejidad Asintótica

- Mejor caso → **O(n log n)**

Complejidad Asintótica

- Mejor caso → **O(n log n)**

- Caso promedio → **O(n log n)**

Complejidad Asintótica

- Mejor caso → **O(n log n)**
- Caso promedio → **O(n log n)**

Complejidad Asintótica

- Mejor caso → **O(n log n)**
- Caso promedio → **O(n log n)**
- Peor caso → **O(n^2)**

Complejidad Asintótica

- Mejor caso → **O(n log n)**

- Caso promedio → **O(n log n)**

- Peor caso → **O(n^2)**

Mejor Caso

Mejor Caso

- Ocurre cuando los pivotes elegidos dividen los arreglos y sub-arreglos en 2 mitades del mismo tamaño.

  T(1) = 1

  T(n) = 2 T(n/2) + n

Mejor Caso

- Ocurre cuando los pivotes elegidos dividen los arreglos y sub-arreglos en 2 mitades del mismo tamaño.

  $T(1) = 1$

  $T(n) = 2\ T(n/2) + n$

- Es la misma ecuación que MergeSort, por ende la misma demostración

Peor Caso

Peor Caso

- Ocurre cuando los pivotes elegidos corresponden a los mínimos o máximos de los arreglos y sub-arreglos.

  $T(1) = 1$

  $T(n) = T(n-1) + n$

Peor Caso

- Ocurre cuando los pivotes elegidos corresponden a los mínimos o máximos de los arreglos y sub-arreglos.

  $T(1) = 1$

  $T(n) = T(n-1) + n$

- Vemos que los llamados solo van eliminando de a 1 elemento con pasadas lineales.

Caso Promedio

Caso Promedio

Caso Promedio

Caso Promedio

- Consideremos 2 elementos cualquiera i y j dentro de nuestro arreglo. Definamos Y_{ij} como la cantidad de veces que se comparan estos 2 elementos.

Caso Promedio

- Consideremos 2 elementos cualquiera i y j dentro de nuestro arreglo. Definamos Y_{ij} como la cantidad de veces que se comparan estos 2 elementos.
- Si pensamos en todos los posibles pares, tenemos:

$$\sum_{i=1}^{n-1} \sum_{i+1}^{n} Y_{ij}$$

Caso Promedio

- Consideremos 2 elementos cualquiera i y j dentro de nuestro arreglo. Definamos Y_{ij} como la cantidad de veces que se comparan estos 2 elementos.
- Si pensamos en todos los posibles pares, tenemos:

$$\sum_{i=1}^{n-1} \sum_{i+1}^{n} Y_{ij}$$

- ¿A qué se deben esos subíndices?

Caso Promedio

- Consideremos 2 elementos cualquiera i y j dentro de nuestro arreglo. Definamos Y_{ij} como la cantidad de veces que se comparan estos 2 elementos.
- Si pensamos en todos los posibles pares, tenemos:

$$\sum_{i=1}^{n-1} \sum_{i+1}^{n} Y_{ij}$$

- ¿A qué se deben esos subíndices?
- Nos interesa saber la esperanza de esta sumatoria. ¿Por qué?

$$\sum_{i=1}^{n-1} \sum_{i+1}^{n} E(Y_{ij})$$

- Tengamos en cuenta a p, nuestro pivote tras 1 aplicación del algoritmo de partición.

- Tengamos en cuenta a p, nuestro pivote tras 1 aplicación del algoritmo de partición.

- Si **($i < p < j$)** → el pivote separo para siempre i y j, por ende $Y_{ij} = 0$

- Tengamos en cuenta a p, nuestro pivote tras 1 aplicación del algoritmo de partición.

- Si **(i < p < j)** → el pivote separo para siempre i y j, por ende $Y_{ij} = 0$

- Si **(p < i < j)** o **(i < j < p)** → el pivote dejo a i y j en la misma partición, con posibilidad de que se comparen a futuro.

- Tengamos en cuenta a p, nuestro pivote tras 1 aplicación del algoritmo de partición.

- Si **($i < p < j$)** → el pivote separo para siempre i y j, por ende $Y_{ij} = 0$

- Si **($p < i < j$)** o **($i < j < p$)** → el pivote dejo a i y j en la misma partición, con posibilidad de que se comparen a futuro.

- Si **($p = i < j$)** o **($i < j = p$)** → los elementos fueron comparados, por ende no vuelven a ser comparados. De esta forma $Y_{ij} = 1$

- Ya que Y_{ij} solo toma los valores 0 y 1:

$$E(Y_{ij}) = 0 \cdot Pr(Y_{ij} = 0) + 1 \cdot Pr(Y_{ij} = 1)$$

$$= \mathbf{Pr(Y_{ij} = 1)}$$

- Ya que Y_{ij} solo toma los valores 0 y 1:

$$E(Y_{ij}) = 0 \cdot Pr(Y_{ij} = 0) + 1 \cdot Pr(Y_{ij} = 1)$$

$$\mathbf{= Pr(Y_{ij} = 1)}$$

- Para calcular $Pr(Y_{ij} = 1)$ pensemos en el siguiente conjunto

$$\mathbf{\{i, i+1, i+2, \cdot \cdot \cdot, j-2, j-1, j\}}$$

- Ya que Y_{ij} solo toma los valores 0 y 1:

$$E(Y_{ij}) = 0 \cdot Pr(Y_{ij} = 0) + 1 \cdot Pr(Y_{ij} = 1)$$

$$\mathbf{= Pr(Y_{ij} = 1)}$$

- Para calcular $Pr(Y_{ij} = 1)$ pensemos en el siguiente conjunto

$$\mathbf{\{i, i+1, i+2, \cdot\cdot\cdot, j-2, j-1, j\}}$$

- Este arreglo posee j-i+1 elementos, si asumimos una distribución uniforme sobre la elección del pivote, cada elemento tiene una posibilidad de ser el pivote de 1 / (j-i+1). Recordemos cuando $Y_{ij} = 1$.

- Ya que Y_{ij} solo toma los valores 0 y 1:

$$E(Y_{ij}) = 0 \cdot Pr(Y_{ij} = 0) + 1 \cdot Pr(Y_{ij} = 1)$$

$$\mathbf{= Pr(Y_{ij} = 1)}$$

- Para calcular $Pr(Y_{ij} = 1)$ pensemos en el siguiente conjunto

$$\mathbf{\{i, i+1, i+2, \cdot \cdot \cdot, j-2, j-1, j\}}$$

- Este arreglo posee j-i+1 elementos, si asumimos una distribución uniforme sobre la elección del pivote, cada elemento tiene una posibilidad de ser el pivote de 1 / (j-i+1). Recordemos cuando $Y_{ij} = 1$.

- Esto nos deja un valor muy claro, ya que solo hay 2 casos, cuando i es el pivote o cuando j es el pivote, así:

$$\mathbf{Pr(Y_{ij} = 1) = 2 / (j-i+1)}$$

(casi) Finalmente vemos que

(casi) Finalmente vemos que

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

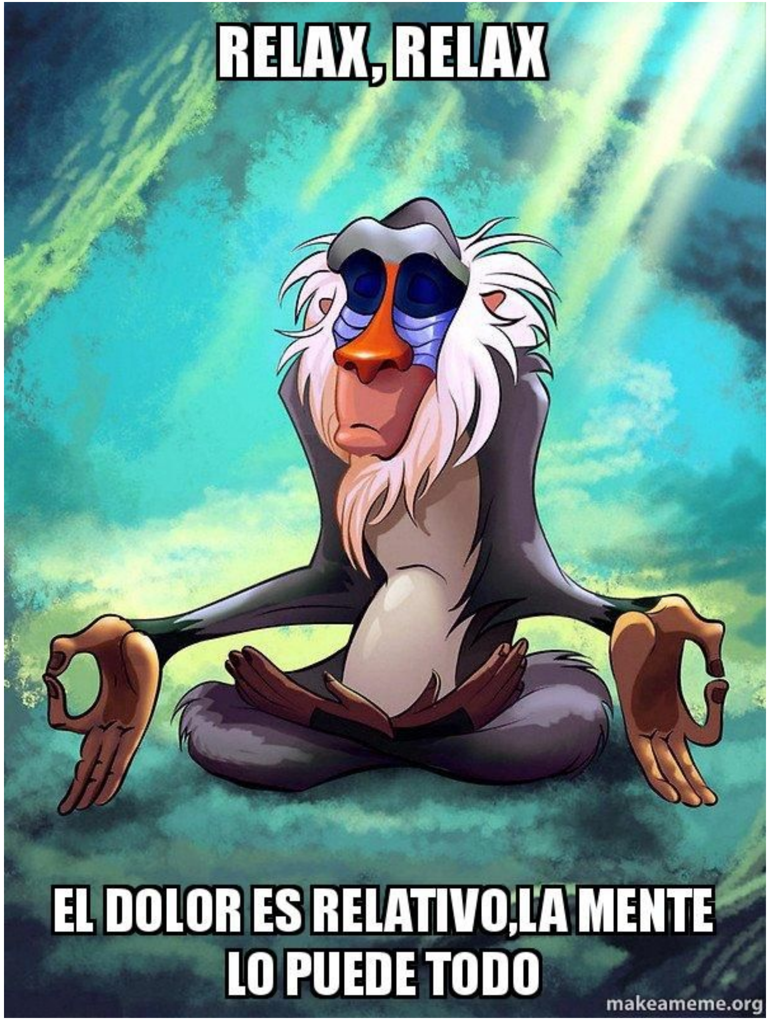
$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

$$\sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k}$$

$$\sum_{k=2}^{n} (n+1-k) \cdot \frac{2}{k}$$

$$2 \cdot (n+1) \cdot \left(\sum_{k=2}^{n} \frac{1}{k} \right) - 2 \cdot (n-1)$$

$$2 \cdot (n+1) \cdot \left(\sum_{k=1}^{n} \frac{1}{k} \right) - 4 \cdot n$$


RELAX, RELAX
EL DOLOR ES RELATIVO,LA MENTE
LO PUEDE TODO
makeameme.org

$$2 \cdot (n+1) \cdot \left(\sum_{k=1}^{n} \frac{1}{k} \right) - 4 \cdot n$$

$$2 \cdot (n+1) \cdot \left(\sum_{k=1}^{n} \frac{1}{k} \right) - 4 \cdot n$$

Lineal

$$2 \cdot (n+1) \cdot \left(\sum_{k=1}^{n} \frac{1}{k} \right) - 4 \cdot n$$

Lineal

Lineal

$$2 \cdot (n+1) \cdot \left(\sum_{k=1}^{n} \frac{1}{k} \right) - 4 \cdot n$$

Lineal

Lineal

¡ES LA SUMA ARMÓNICA!

¡Es la suma armónica!

$$\sum_{k=1}^{n} \frac{1}{k}$$

¡Es la suma armónica!

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

$$2 \cdot (n+1) \cdot \left(\sum_{k=1}^{n} \frac{1}{k} \right) - 4 \cdot n$$

Lineal

Lineal

$$2 \cdot (n+1) \cdot \left(\sum_{k=1}^{n} \frac{1}{k} \right) - 4 \cdot n$$

Lineal

Logarítmico

Lineal

$$2 \cdot (n+1) \cdot \left(\sum_{k=1}^{n} \frac{1}{k} \right) - 4 \cdot n$$

Lineal

Logarítmico

Lineal

$$\mathcal{O}(n \log(n))$$