QuickSort

Partição de Vetor

Resolver o problema da Partição de Vetor serve para projetar o Quicksort.

Partição de Vetor

Entrada: (v, a, b) onde v é um vetor indexado por $[a..b]$.
Saída : Seja $x = v[b]$. Modifica o vetor v de forma a garantir a existência de um índice $m \in [a-1..b]$ tal que

$$v[i] \leq x, \forall i \in [a..m],$$

e

$$x < v[i], \forall i \in [m+1..b].$$

Devolve este índice m.

$\leq x$	**x**	$> x$

- x é chamado de *pivô*.

Particiona(v, a, b)

$m \leftarrow a - 1$
$i \leftarrow a$
Para $i \leftarrow a$ to b
    Se $v[i] \leq x$
        $m \leftarrow m + 1$
        Troca(v, m, i)
Devolva m

Executar Particiona$(v, 1, 6)$

i	1	2	3	4	5	6
$v[i]$	3	10	13	5	8	6

Importante: após a execução do Particiona, x fica na posição que deveria estar se o vetor estivesse ordenado.

Teorema 1. *Particiona executa em tempo $\Theta(n)$, onde n é o tamanho do vetor.*

Quicksort

É um algoritmo projetado usando a técnica "divisão e conquista".

Quicksort(v, a, b)

```
Se a < b
    m ← Particiona(v, a, b)
    Quicksort(v, a, m − 1)
    Quicksort(v, m + 1, b)
```

Análise

$T(n)$: tempo do Quicksort em vetores de n elementos.

$$T(n) = \begin{cases} \Theta(1), & Se\ n \leq 1 \\ T(k) + T(n-k-1) + \Theta(n), & Se\ n > 1 \end{cases}$$

onde $\Theta(n)$ vem do tempo do Particiona.

Um Caso Ruim

Ocorre quando a Particiona produz as duas partições com tamanho 0 e $n-1$.

Isso ocorre, **por exemplo**, quando vetor JÁ está ordenado. Mas existem outros casos ruins também.

Assim, a relação de recorrência fica:

$$T(n) = \begin{cases} \Theta(1), & Se\ n \leq 1 \\ T(0) + T(n-1) + \Theta(n), & Se\ n > 1 \end{cases}$$

ou

$$T(n) = \begin{cases} \Theta(1), & Se\ n \leq 1 \\ T(n-1) + \Theta(n), & Se\ n > 1 \end{cases}$$

É possível mostrar que a solução dessa recorrência é $T(n) = \Theta(n^2)$.

Pior Caso

Como saber que o caso ruim é o pior caso?

Teorema 2. *A recorrência*

$$T(n) = \begin{cases} \Theta(1), & Se\ n \leq 1 \\ \max\limits_{0 \leq k \leq n-1} \{T(k) + T(n-k-1)\} + \Theta(n), & Se\ n > 1 \end{cases}$$

tem como solução $T(n) = O(n^2)$.

Demonstração. (indução em n)
base: (Exercício)
hipótese: $T(q) \leq cq^2, \forall 1 \leq q < n$
passo: Vamos provar que $T(n) \leq cn^2$.

$$\begin{aligned} T(n) &= \max_{0 \leq k \leq n-1} \left\{ T(k) + T(n-k-1) \right\} + bn \\ &\leq \max_{0 \leq k \leq n-1} \left\{ ck^2 + c(n-k-1)^2 \right\} + bn \\ &= c \max_{0 \leq k \leq n-1} \left\{ k^2 + (n-k-1)^2 \right\} + bn \end{aligned}$$

A expressão $k^2 + (n-k-1)^2$ atinge valor máximo quando $k = 0$ ou $k = n-1$ [Exercício CLRS]. Assim,

$$\begin{aligned} T(n) &\leq c \max_{0 \leq k \leq n-1} \left\{ k^2 + (n-k-1)^2 \right\} + bn \\ &= c(n-1)^2 + bn \\ &\leq cn^2, \end{aligned}$$

onde a última desigualdade é válida se $c \geq b$ e $n \geq 1$. $\square$

A prova mostra que quando Particiona produz as partições com tamanho 0 e $n-1$ é de fato o **pior caso**.

Algumas Conclusões Até Agora...

A complexidade de tempo do Quicksort no **pior caso** é $\Theta(n^2)$.

A complexidade de tempo do Quicksort é $O(n^2)$.

Melhor caso

Não seremos rigorosos na análise de melhor caso (só daremos a ideia).

Ocorre quando a Particiona produz as duas partições com tamanho "igual".

$\lfloor n/2 \rfloor$	**x**	$\lceil n/2 \rceil - 1$

$$T(n) = \begin{cases} \Theta(1), & Se\ n \leq 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil - 1) + \Theta(n), & Se\ n > 1 \end{cases}$$

Teorema 3. *A relação de recorrência:*

$$T(n) = \begin{cases} \Theta(1), & Se\ n \leq 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil - 1) + \Theta(n), & Se\ n > 1 \end{cases}$$

tem como solução $T(n) = \Theta(n \log n)$.

Demonstração. (Exercício) □

Outra Conclusões...

A complexidade de tempo do Quicksort no **melhor caso** é $\Theta(n \log n)$.

A complexidade de tempo do Quicksort é $\Omega(n \log n)$.

Na prática: O Quicksort é um dos mais eficientes algoritmos de ordenação.

Na teoria: Seu pior caso é aproximadamente igual aos dos algoritmos mais simples de ordenação.