

Análise dos Algoritmos:

a) Maior elemento:

a) Se vários índices possuírem o maior valor, o algoritmo retornará o índice do primeiro valor máximo encontrado durante a execução do algoritmo.

b) Relação de Recorrência:

$$T(n) = 2T(n/2) + 1$$

$$T(1) = 1$$

Resolvendo pelo Teorema Mestre:

$$a = 2 \quad b = 2 \quad f(n) = 1$$

$$f(n) = O(n^{\log_2 2 - 1})$$

$$f(n) = O(n^{\log_2 2 - 1})$$

$$= O(n^{1-1}) = O(n^0) = 1 \quad \checkmark$$

Logo:

$$T(n) = \Theta(n^{\log_2 2}) = \Theta(n)$$

c) O algoritmo de força bruta faz menos comparações que o recursivo, apesar de ambos possuírem complexidade $O(n)$.



2) merge Sort:

Complexidade do Merge: $O(n)$

EQUAÇÃO DE RECORRÊNCIA

$$T(1) = 1$$

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

Resolvendo pelo Teorema Mestre:

$$a=2 \quad b=2 \quad f(n)=n$$

$$f(n) = \Theta(n^{\log_b a})$$

$$f(n) = \Theta(n^{\log_2 2}) = \Theta(n) \quad \checkmark$$

Logo:

$$T(n) = \Theta(n^{\log_2 2} \log n)$$

$$= \Theta(n \log n) //$$

É um algoritmo estável?

Sim, pois a ordem relativa dos elementos com chaves iguais não é afetada. Na etapa do merge os elementos são comparados um a um e colocados na ordem correta.



3) Quicksort:

Complexidade do partition: $O(n)$

* Nesse caso a complexidade do swap é $O(1)$ pois ele troca dois valores.

Pior Caso: Ocorre quando o pivô divide o vetor de forma desbalanceada. no caso, em duas sublistas, uma de tamanho 0 e outra de tamanho $n-1$.

Pode ocorrer quando o elemento pivô é o maior ou menor elemento do vetor.

EQUAÇÃO DE RECORRÊNCIA:

$$\begin{cases} T(0) = 0 \\ T(1) = 1 \end{cases}$$

$$T(n) = T(n-1) + T(0) + O(n)$$

→ Resolução:

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

$$(T(n-2) + n) + n$$

$$((T(n-3) + n) + n) + n$$

:

$$T(n-i) + \sum_{j=1}^i n$$

$$\begin{cases} n-i=0 \\ i=n \end{cases}$$

$$\frac{n(1+i)}{2}$$

$$T(1) + \frac{n^2 + n}{2}$$

$$\frac{n(1+n)}{2}$$

$$\frac{n^2 + n + 2}{2} // \Theta(n^2)$$

$$\frac{n^2 + n}{2} //$$



Melhor caso: Quando as sublistas possuem tamanho $n/2$:

EQUAÇÃO DE RECORRÊNCIA:

$$T(0) = 0$$

$$T(1) = 1$$

$$T(n) = 2 T\left(\frac{n}{2}\right) + O(n)$$

Resolução pelo Teorema mestre:

$$f(n) = n, \quad a = 2, \quad b = 2$$

$$f(n) = \Theta(n^{\log_2 2}) = \Theta(n) \quad \checkmark$$

Logo:

$$T(n) = n^{\log_2 2} \log n$$

$$\boxed{\Theta(n \log n)}$$

É um algoritmo estável?

Não, pois a ordem relativa dos elementos iguais não é necessariamente preservada durante a ordenação, já que os elementos iguais podem ser trocados de posição com outros elementos durante a troca de pivôs.



4) Árvore Binária:

4.1) Tamanho da Árvore:

EQUAÇÃO DE RECORRÊNCIA:

$$T(1) = 1 \quad (\text{no melhor caso})$$

$$\begin{aligned} T(n) &= T(n/2) + T(n/2) + 1 \\ &= 2T(n/2) + 1 \end{aligned}$$

Conforme resolvendo o algoritmo 1:

$$T(n) = \Theta(n)$$

$$T(1) = 1 \quad (\text{no pior caso})$$

$$\begin{aligned} T(n) &= T(0) + T(n-1) + 1 \\ &= T(n-1) + 1 \end{aligned}$$

Resolvendo:

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

$$(T(n-2) + 1) + 1$$

$$((T(n-3) + 1) + 1) + 1 \quad n-i=1$$

⋮

$i=n-1$

$$T(n-i) + i$$

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

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

$$= n$$

$$= \boxed{\Theta(n)}$$



4.2) inOrder:

→ Visita o filho da esquerda, a raiz e o filho da direita.

EQUAÇÃO DE RECORRÊNCIA:

$$T(1) = 1$$

(caso árvore balanceada)

$$T(n) = 2T(n/2) + 1$$

Conforme já foi resolvido:

$$T(n) = \Theta(n)$$

$$T(1) = 1$$

(caso árvore desba-

$$T(n) = T(0) + T(n-1) + 1$$

lanceada)

Conforme resolvido antes:

$$T(n) = \Theta(n)$$

4.3) preOrder:

→ Visita a raiz, o filho da esquerda e o filho da direita.

EQUAÇÃO DE RECORRÊNCIA:

$$T(1) = 1$$

(árvore balanceada)

$$T(n) = 2T(n/2) + 1$$

$$\Rightarrow T(n) = \Theta(n)$$

$$T(1) = 1$$

(árvore desbalanceada)

$$T(n) = T(0) + T(n-1) + 1$$

$$\Rightarrow T(n) = \Theta(n)$$



4.4) Post Order:

→ Visita o filho da esquerda, o filho da direita e a raiz.

Equação de Recorrência:

$$T(1) = 1$$

$$T(n) = 2T(n/2) + 1 \quad (\text{Árvore balanceada})$$

$$\Rightarrow T(n) = \Theta(n)$$

$$T(1) = 1$$

$$T(n) = T(0) + T(n-1) + 1 \quad (\text{Árvore desbalanceada})$$

$$\Rightarrow T(n) = \Theta(n)$$

———— // —————