Capítulo 1

O algoritmo *mergesort*

Algoritmos recursivos desempenham um papel fundamental em Computação. O algoritmo de ordenação *mergesort* é um exemplo de algoritmo recursivo, que se caracteriza por dividir o problema original em subproblemas que, por sua vez, são resolvidos recursivamente. As soluções dos subproblema são então combinadas para gerar uma solução para o problema original. Este paradigma de projeto de algoritmo é conhecido com *divisão e conquista*. Este algoritmo foi inventado por J. von Newmann em 1945.

O algoritmo *mergesort* é um algoritmo de ordenação que utiliza a técnica de divisão e conquista, que consiste das seguintes etapas:

1. **Divisão**: O algoritmo divide a lista (ou vetor) l recebida como argumento ao meio, obtendo as listas l_1 e l_2;

2. **Conquista**: O algoritmo é aplicado recursivamente às listas l_1 e l_2 gerando, respectivamente, as listas ordenadas l'_1 e l'_2;

3. **Combinação**: O algoritmo combina as listas l'_1 e l'_2 através da função *merge* que então gera a saída do algoritmo.

Por exemplo, ao receber a lista $(4 :: 2 :: 1 :: 3 :: nil)$, este algoritmo inicialmente divide esta lista em duas sublistas, a saber $(4 :: 2 :: nil)$ e $(1 :: 3 :: nil)$. O algoritmo é aplicado recursivamente às duas sublistas para ordená-las, e ao final deste processo, teremos duas listas ordenadas $(2 :: 4 :: nil)$ e $(1 :: 3 :: nil)$. Estas listas são, então, combinadas para gerar a lista de saída $(1 :: 2 :: 3 :: 4 :: nil)$.

```
1 if p < r then
2 |   q = ⌊(p+r)/2⌋;
3 |   mergesort(A, p, q);
4 |   mergesort(A, q + 1, r);
5 |   merge(A, p, q, r);
6 end
```

Algoritmo 1: mergesort(A, p, r)

A etapa de combinar dois vetores ordenados (algoritmo *merge*) é a etapa principal do algoritmo *mergesort*. O procedimento $merge(A, p, q, r)$ descrito a seguir recebe como argumentos o vetor A, e os índices p, q e r tais que $p \leq q < r$. O procedimento assume que os subvetores $A[p..q]$ e $A[q+1..r]$ estão ordenados.

$$= 2^2 \cdot T(2^{K-2}) + 2 \cdot c \cdot 2^K$$

$$= 2^2 \cdot (2 \cdot T(2^{K-3}) + c \cdot 2^{K-2}) + 2 \cdot c \cdot 2^K$$

$$= 2^3 \cdot T(2^{K-3}) + 3 \cdot c \cdot 2^K = \ldots$$

$$= 2^K \cdot T(2^{K-K}) + K \cdot c \cdot 2^K = \underbrace{c \cdot K \cdot 2^K}_{\text{chute.}}$$

Verificação do chute por indução: $\boxed{T(2^K) = c \cdot K \cdot 2^K}$

base: $K = 0$: $T(2^0) = T(1) = c \cdot 0 \cdot 2^0 = 0$ ✓

passo: $K > 0$: $T(2^K) = 2 \cdot T(2^{K-1}) + c \cdot 2^K$

$$\overset{\text{h.i.}}{=} 2 \cdot (c \cdot (K-1) \cdot 2^{K-1}) + c \cdot 2^K$$

$$= 2 \cdot c \cdot K \cdot 2^{K-1} - 2 \cdot c \cdot 2^{K-1} + c \cdot 2^K$$

$$= 2 \cdot c \cdot K \cdot 2^{K-1}$$

$$= c \cdot K \cdot 2^K. \quad ✓$$

$2 \cdot c \cdot 2^{K-1}$

□

$n = 2^K \Rightarrow K = \lg n$

$$T(n) = c \cdot (\lg n) \cdot n = \underline{c \cdot n \cdot \lg n}.$$

$$T(n) = \Theta(n \cdot \lg n).$$

Exercício: Mostre que $n \cdot \lg n$ é uma função suave.

```
1  n_1 = q − p + 1 ;                          // Qtd. de elementos em A[p..q]
2  n_2 = r − q ;                              // Qtd. de elementos em A[q+1..r]
3  let L[1..n_1 + 1] and R[1..n_2 + 1] be new arrays;
4  for i = 1 to n_1 do
5  |  L[i] = A[p + i − 1];
6  end
7  for j = 1 to n_2 do
8  |  R[j] = A[q + j];
9  end
10 L[n_1 + 1] = ∞;
11 R[n_2 + 1] = ∞;
12 i = 1;
13 j = 1;
14 for k = p to r do
15 |  if L[i] ≤ R[j] then
16 |  |  A[k] = L[i];
17 |  |  i = i + 1;
18 |  end
19 |  else
20 |  |  A[k] = R[j];
21 |  |  j = j + 1;
22 |  end
23 end
```

Algoritmo 2: merge(A, p, q, r)

Exercício 1. *Prove que o algoritmo merge é correto.*

Exercício 2. *Prove que o algoritmo mergesort é correto.*

Exercício 3. *Faça a análise assintótica do algoritmo merge.*

Exercício 4. *Faça a análise assintótica do algoritmo mergesort.*

1.0.1 Equações de recorrência

Nesta seção estudaremos as equações de recorrência utilizadas no paradigma de divisão de conquista [2]:

Definição 1. *Seja $f(n)$ uma função não-negativa definida no conjunto dos números naturais. Dizemos que $f(n)$ é* eventualmente não-decrescente *se existir um número inteiro n_0 tal que $f(n)$ é não-decrescente no intervalo $[n_0, \infty)$, ou seja,*

$$f(n_1) \leq f(n_2), \forall n_2 > n_1 \geq n_0.$$

Definição 2. *Seja $f(n)$ uma função não-negativa definida no conjunto dos números naturais. Dizemos que $f(n)$ é* suave *se for eventualmente não-decrescente e*

$$f(2.n) = \Theta(f(n))$$