Capítulo 4

Divisão e Conquista

Nesta seção estudaremos o paradigma de programação conhecido com *divisão e conquista*. Este paradigma é caracterizado pelos seguintes passos:

1. **Divisão**: O problema original é dividido em subproblemas que são instâncias menores do problema original.

2. **Conquista**: Os subproblemas são resolvidos recursivamente, e em seguida, estas soluções são combinadas para gerar uma solução do problema original.

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 algoritmo foi inventado por J. von Newmann em 1945.

Algorithm 3: mergesort(A, p, r)

```
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
```

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.

Algorithm 4: merge(A, p, q, r)

```
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
```

4.1 A complexidade do algoritmo *mergesort*.

A estratégia de divisão e conquista anda de mãos dadas com as chamadas equações de recorrência. No caso de *mergesort*, o tempo de execução para uma entrada de tamanho n é dado pela recorrência:

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1; \\ 2.T(n/2) + \Theta(n) & \text{se } n > 1. \end{cases} \tag{4.1}$$

Esta recorrência nos diz que se o tamanho da entrada é 1 então o problema é resolvido em tempo constante. De fato, para vetores vazios ou unitários não há nada a ser feito. Para vetores com pelo menos dois elementos, o problema é dividido em dois subproblemas de mesmo tamanho que depois têm suas soluções juntadas pelo algoritmo *merge*.

Exercício: Qual a complexidade assintótica do algoritmo *mergesort* em função do comprimento n do vetor A?