Epaminondas Sort

Daniel Saad

30 de março de 2026

Hoje, na minha prova de Análise de Algoritmos, propus um problema que era uma variação do MERGESORT. Segue a descrição do problema, o qual, carinhosamente, apelidei de EPAMINONDAS-SORT:

> Epaminondas resolveu implementar o algoritmo MERGESORT para ordenar um vetor de n elementos. Contudo, como ele aprendeu a programar com o *Chato GPT*, quando foi implementar o procedimento de MERGE, ele fez a seguinte implementação:
>
> **Algorithm 1:** MERGE$(V[0, n-1], V_1[0, \lceil n/2 \rceil - 1], V_2[0, n - \lceil n/2 \rceil - 1])$
>
> ```
> 1 for( i ← 0 to ⌈n/2⌉ − 1 )
> 2     V[i] ← V1[i]
> 3 for( i ← ⌈n/2⌉ to n − 1 )
> 4     V[i] ← V2[i − ⌈n/2⌉]
> 5 BUBBLESORT(V)
> ```

O desafio era analisar o tempo de execução, no pior caso, do EPAMINONDAS-SORT.

Sabemos que o MERGESORT tem tempo de execução dado pela seguinte equação de recorrência:

$$T(n) = \begin{cases} \Theta(1), & n = 1 \\ 2T\left(\frac{n}{2}\right) + f(n), & n > 1 \end{cases} \tag{1}$$

No caso, $f(n)$ é o custo do procedimento de MERGE. Quando bem implementado, o MERGE tem custo $\Theta(n)$. Contudo, não é o caso do MERGE do EPAMINONDAS-SORT, que invoca o BUBBLESORT, algoritmo com custo $\Theta(n^2)$. Isto é, no EPAMINONDAS-SORT, temos $f(n) \in \Theta(n^2)$.

Ao aplicar o Teorema Mestre, temos que $a = 2$, $b = 2$ e $f(n) = \Theta(n^2)$. Assim, $n^{\log_b a} = n^{\log_2 2} = n$. Como $f(n)$ é assintoticamente maior do que $n^{\log_b a}$, temos que o tempo de execução do EPAMINONDAS-SORT é dado por $T(n) \in \Theta(f(n)) = \Theta(n^2)$. Falta verificar a condição de regularidade, mas deixo de exercício para o leitor.