Discente: Ronaldo Ribeiro Porto Filho – 202410131
Data: 21/09/2025

Exercício tamanho da pilha ajustada do algoritmo Quicksort

Vamos analisar o tamanho da pilha do algoritmo Quicksort abaixo, e então, provar que ele é $O(\log_2 n)$:

```
QuickSort(min, max){
   while(min < max){
        p = partition(min, max);
        if(p – min < max – p){
          QuickSort(min, p-1);
          min = p + 1
        }else{
          QuickSort(p+1, max);
          max = p – 1
         }
    }
}
```

Esse algoritmo funciona da seguinte forma, ao invés de chamar recursivamente dois subvetores, ele chama somente o menor, sendo que o maior é resolvido no próprio laço while, otimizando as pilhas de recursão.

Com isso, um vetor de tamanho n, após a função particiona, verifica qual subvetor é maior, este será tratado no próprio while, e o subvetor menor, que logicamente possui no máximo n/2 elementos, é usado como parâmetro para a próxima recursão. Ou seja, no pior caso, a próxima chamada terá tamanho n/2, e a próxima n/4, e a próxima n/8, n/16, etc.

Portanto, o tamanho do subvetor recursivo decresce geometricamente, e termina no caso base, quando o vetor é de tamanho unitário, e dessa forma:

Prova:

Caso Base:

$T(1) = 1$

Recorrência:

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

$T(n) = (T(n/4) + 1) + 1 , n > 1$

$T(n) = ((T(n/8) + 1) + 1) + 1 , n > 1$

$T(n) = (((T(n/16) + 1) + 1) + 1) + 1 , n > 1$

$T(n) = T(n/16) + 4 , n > 1$

$T(n) = T(n/2^k) + k , n > 1$

$n/2^k = 1$

$n = 2^k$

$\log_2 n = \log_2 2^k$

$\log_2 n = k * \log_2 2$

$\log_2 n = k$

$T(n) = T(n/2^{\log_2 n}) + \log_2 n , n > 1$

$T(n) = T(n/n) + \log_2 n , n > 1$

$T(n) = T(1) + \log_2 n , n > 1$

$T(n) = \log_2 n + 1 , n > 1$

Termo de maior ordem: $\log_2 n$

$T(n) \in O(\log_2 n)$