Análise Assintótica de Algoritmos Recursivos

Na postagem passada, vimos como realizar a análise assintótica de algoritmos com laços e condicionais. Agora, veremos como realizar a análise de algoritmos recursivos. Serão apresentados quatro métodos para a análise de algoritmos recursivos: o **Método Mestre**, o **Método da Árvore de Recursão**, o **Método da Substituição Iterativo**, e o **Método da Substituição Indutiva**. Cada método é apresentado em uma postagem específica.

Antes disso, um conceito importante é a **recorrência**. A recorrência é uma função que descreve o tempo de execução de um algoritmo recursivo. Para ilustrar esse conceito, vamos utilizar como exemplos o **Merge Sort**, um algoritmo para resolver o **problema da ordenação**, e o problema da **Torre de Hanói**.

Merge Sort

Nesta postagem, o Merge Sort será apresentado apenas em alto nível, pois o objetivo é apenas demonstrar o conceito de recorrência. Os detalhes do mesmo são apresentados em outra postagem.

O problema da ordenação consiste em:

Entrada: uma lista de elementos <a1, a2, …, an>

Saída: <a1, a2, …, an>, sendo ai <= aj, para todo i < j.

O pseudocódigo do Merge Sort é apresentado abaixo:

```
1 proc mergesort(lista l)
2     #dividir a lista ao meio e distribuí-la em duas variáveis
3     esquerda, direita = dividir(l)
4     #recursivamente, resolver cada subproblema menor
5     e = mergesort(esquerda)
6     d = mergesort(direita)
7     #intercalar resultados
8     merge(e,d)
```

No pseudocódigo acima, na linha 3, a lista é dividida ao meio e cada metade é alocada para as variáveis **esquerda** e **direita**. Posteriormente, nas linhas 5 e 6, cada metade da lista original (subproblema) é ordenada recursivamente. Ao final, na linha 8, o procedimento **merge()** recebe as listas **e** e **d**, que estão ordenados, e as rearranja de forma que a lista resultante esteja ordenada.

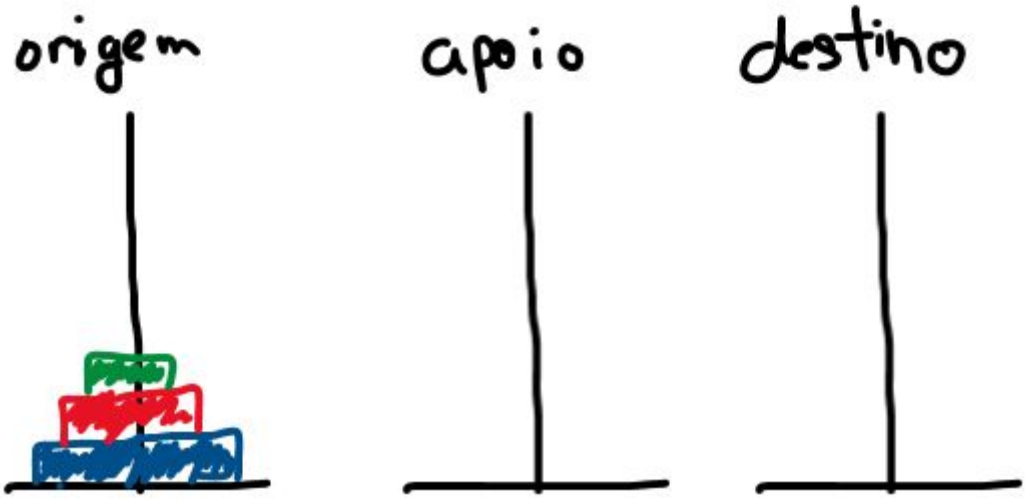
Como dito anteriormente, os detalhes do Merge Sort são apresentados em outra postagem. Então, por hora, vamos assumir que o procedimento **merge()** é **O(n)** (acreditem em mim =)). Dessa forma, temos que o tempo necessário para executar o **Merge Sort**, **T(n)**, depende do tempo necessário para resolver as duas metadas (linhas 5 e 6) somado com o tempo necessário para resolver a intercalação na linha 8 (**O(n)**). Dado isso, podemos escrever que:

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

$$T(n) = 2T(n/2) + \Theta(n)$$

Torre de Hanói

Outro exemplo clássico de recorrência é o algoritmo recursivo para o **problema da Torre de Hanói**.



O problema da Torre de Hanói consiste em uma base com três pinos: **origem**, **apoio** e **destino**. O pino **origem** contém inicialmente **n** discos (na imagem acima, contém 3 discos) em ordem decrescente de raio (ou diâmetro). O objetivo é mover todos os discos do pino **origem** para o pino **destino** de forma que um **disco maior nunca fique em cima de outro menor**.

O pseudocódigo para o algoritmo é apresentado abaixo:

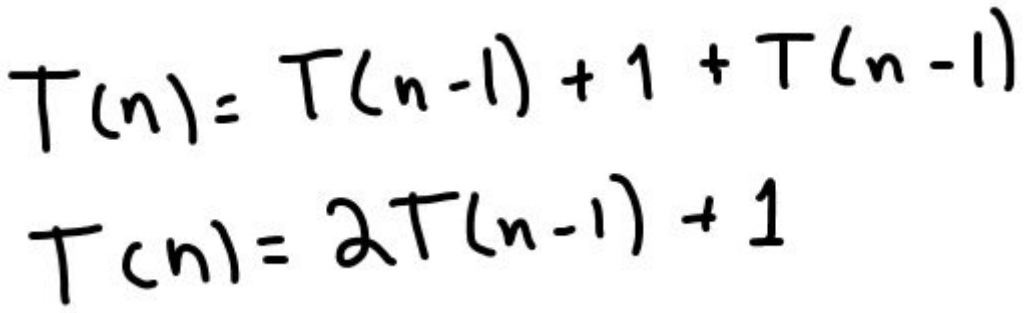
```
1 proc torreDeHanoi(discos, origem, destino, apoio)
2     se discos é igual a 1 então
3         moverDisco(1, origem, destino)
4     torreDeHanoi(discos - 1, origem, apoio, destino)
5     moverDisco(discos, origem, destino)
6     torreDeHanoi(discos - 1, apoio, destino, origem)
```

Um ótimo vídeo do GeeksforGeeks que explica o funcionamento do algoritmo é apresentado abaixo:

https://www.youtube.com/watch?v=YstLjLCGmgg

Analisando o pseudocódigo, podemos observar que há duas chamadas recursivas nas linhas 4 e 6, cada uma reduzindo o problema (número de discos) em 1, ou seja,

com custo **T(n-1)**. O custo da linha 5 é O(1). Dado isso, temos queo custo total, **T(n)**, do algoritmo da Torre de Hanói é:



Nas próximas postagens, serão apresentados quatro métodos para a análise de algoritmos recursivos: o **Método Mestre**, o **Método da Árvore de Recursão**, o **Método da Substituição Iterativo**, e o **Método da Substituição Indutiva**.

Sumário

Nesta postagem, você aprendeu sobre:

- Relações de recorrência como um meio para representar algoritmos recursivos.
- Uma introdução ao **problema da ordenação** e ao **Merge Sort**, incluindo a sua recorrência.
- O algoritmo recursivo para resolver o problema da **Torre de Hanói**, incluindo a sua recorrência.