Projeto de Algoritmo com Implementação nº 0

Lucas Valente Viegas de Oliveira Paes - RA 220958

31 de outubro de 2020

1 Demonstração de corretude

Vamos demonstrar que o algoritmo funciona por indução.

Hipótese de indução Dado um móbile, sabemos calcular seu peso em ambas as suas extremidades, bem como saber se ambas estão em equilíbrio.

Caso base Mobile sem sub-móbiles atrelados às suas extremidades. Nesse caso, seu peso é $P_e + P_d$ e ele está em equilíbrio somente se $P_e D_e = P_d D_d$.

Passo indutivo Pela *Hipótese de indução*, sabemos P_e e P_d, além de saber se as extremidades do móbile estão equilibradas. Assim, determinamos que o seu peso é $P_d + P_e$ e ele está em equilíbrio somente se $P_e D_e = P_d D_d$.

2 Análise de complexidade no pior caso

O pior caso ocorre quando temos um móbile equilibrado cujo número de folhas, ou seja, de móbiles que não têm um submóbile atrelado a eles, é mínimo. Neste caso, o número de chamadas recursivas é maximizado, enquanto as ocorrências do caso base são minimizadas.

Uma instância dele pode ser representada por uma árvore com $n - 1$ móbiles internos de um filho e apenas um móbile folha (sem nenhum filho). Para esse caso, a fórmula de recorrência é

$$\begin{cases} T(1) = \Theta(1), & \text{se } n = 1 \\ T(n) = T(n-1) + \Theta(1), & \text{se } n > 1 \end{cases} \tag{1}$$

Vamos supor que $T(m) = m$ para $m < n$. Considerando $m = n - 1$, $n > 1$ e substituindo em (1), temos

$$T(n) = T(m) + \Theta(1) = n - 1 + \Theta(1)$$

No caso base ($m = 1$),

$$T(1) = 1$$

Portanto, $T(n) \in \Theta(n)$.

3 Notas sobre a implementação

É sabido que a entrada do programa segue a ordem inversa de hierarquia da árvore, ou seja, que os nós pais são dados antes dos filhos. Sabe-se, ainda, que, para nós que têm filhos, estes aparecem logo em seguida, da esquerda para a direita. Em posse dessas informações, é possível desenhar uma uma abordagem não recursiva de travessia pela árvore, que começa a partir dos nós filhos e em seguida determina o balanceamento e peso do nó pai. Desse modo, evita-se erros de *stack overflow* além da travessia de volta por toda a pilha de chamadas recursivas (*call stack*) para retorno.

A ideia por trás dessa implementação é inserir os móbiles lidos em uma pilha P_m para acessá-los na ordem reversa, tendo à disposição uma pilha de pesos P_p. Ao desempilhar um móbile M de P_m, checamos se ele tem filho esquerdo. Se sim, o peso deste filho está no topo de P_p. Fazemos o mesmo com o filho direito. Após isso, verificamos se M está balanceado. Caso não, o móbile global está desbalanceado e podemos parar todo o processamento por aí. Caso sim, calculamos seu peso e adicionamos a P_p. Note que nunca consumiremos de P_p vazia, pois sempre iniciaremos o processamento pelas folhas (casos base), já que elas são dadas por último na entrada do programa.

Como a inserção e remoção nas pilhas ocorre em tempo constante ($O(1)$) e estamos calculando o peso e balanceamento de nós internos a partir de seus filhos, a demonstração de corretude e a análise de complexidade continuam valendo.