Lista 5

Caio Ian R. S. Melo

February 26, 2021

1

Determinar qual o algorítmo ótimo para a resolução de um problema. Medimos a complexidade frente a parte que demanda mais recursos computacionais em um software como processamento ou uso de memória. Por exemplo em uma busca, verifica-se que percorrer a estrutura de dados é muito mais custoso que uma operação algébrica como somar e multiplicar, então esse é o ponto chave que demanda uma maior atenção do programador na hora do desenvolvimento.

2

Considerando que hajam "n" elementos nas estruturas de dados a seguir:

- $\mathcal{O}(n)$, isso acontece caso a inserção seja feita sempre percorrento toda a lista para a ao invés de $\mathcal{O}(1)$ que seria no inicio dela.
- $\mathcal{O}(n)$, ocorrerá quando o elemento distar toda a lista de onde se começa a percorrê-la. Pode-se diminuir esse problema subdividido-a e comparando a lista paralelamente em outras threads para amenizar a situação, mas, há sempre a chance de se "cair" no pior caso citado anteriormente, por isso outras estruturas dados são recomendadas quanto é necessário uma maior velocidade de busca.
- Para o caso de uma árvore binária cheia $\mathcal{O}(\log_2 n)$, já para uma árvore genérica com n "galhos", $\mathcal{O}(\log_k n)$, onde k seria a quantidade de "galhos" que a árvore possui. As complexidades citadas anteriormente seriam para árvores "balanceadas", pois, no pior caso uma árvore seria apenas uma lista encadeada e portando a complexidade de busca seria $\mathcal{O}(n)$, para resolver esse problema utiliza-se o "balanceamento" frequentemente após um certo número de inserções ou remoções.

3

- $\mathcal{O}(\frac{n}{2}\log_2 n)$.
- $\mathcal{O}(n\log_2 n)$.