Análise de Algoritmos

Em projetos de software, é importante analisar a modularidade, correção, manutenção, funcionalidade, robustez, tempo de execução, user friendliness, tempo de desenvolvimento, simplicidade de uso, extensibilidade, confiabilidade e uso de memória.

Algoritmos nos permite entender escalabilidade. O desempenho muitas vezes estabelece a fronteira entre o que é e o que não é possível. A matemática dos algoritmos nos fornece uma linguagem para falar sobre o comportamento de um programa. Medimos o desempenho de um algoritmo a partir do seu uso de memória e tempo de execução.

Essa análise pode ser feita de **forma experimental**, implementando em alguma linguagem e medindo a memória e tempo gastos, mas os resultads variam com a máquina e linguagem, ou, de forma teórica, a partir da **análise assintótica**.

Na forma experimental, modelo RAM, cada operação simples demora 1 unidade de tempo e cada acesso de memória demora 1 unidade de tempo. O problema com esse método é que é necessário analisar todos os valores possíveis de elementos no vetor para cada valor de n e para cada um dos casos acima deve-se analisar todos os valores possíveis para o valor da chave.

Na análise assintótica, é estabelecido um limitante, superior ou inferior, para o desempenho de um algoritmo. Ela se baseia apenas na lógica do algoritmo, sem se importar com os dados reais que serão processados.

Análise assintótica:

Quando há mais de uma estrutura de repetição aninhadas, a complexidade do trecho inteiro será a multiplicação das complexidades de cada estrutura de repetição.
Quando há mais de uma estrutura de repetição em sequência, a complexidade total é obtida a partir da soma das complexidades de cada estrutura de repetição.

1. Complexidade linear (soma ou subtração)
   - Incremento de 1 por vez: $T(n) = n$;
   - Incremento de 2 por vez: $T(n) = n/2$;

2. Complexidade logarítmica (multiplicação ou divisão)
   - $i = i*2$: $T(n) = \log_2(n)$;
   - $i = i/2$: $T(n) = \log_2(n)$;

3. Dependência linear
   - $T(n) = n(n+1)/2$;

4. Complexidade quadrática
   - Duas estruturas de repetição de complexidade linear aninhadas: $T(n) = n^2$;

Notação O

Para cada um dos termos da função T(n), deixamos os coeficientes iguais a 1 e mantemos o maior termo da função, descartando os demais. A classificação de importância dos termos, da mais baixa para a mais alta, é a seguinte:

log(n), n, nlog(n), n^2, n^3, ... , n^k, 2^n, n!