Análise de Algoritmo

Giuseppe S. F. Agostini[1]

[1]ICEI – Pontifícia Universidade Católica de Minas Gerais (PUC-MG)

Quando vamos analizar os aspectos de complexidade de um algortimo, devemos levar em conta duas coisas: O tempo de execução e a memoria que vai ser ocupada. Entretando, o tempo de execução não depende só do algoritmo, quando medimos o tempo de execução, devemos levar em conta a qualidade do compilador, o hardware, sistema operacional, entre outros fatores. Logo, quando vamos medir o custo de um algoritmo devemos determinar as operações relevantes(comparações, operações aritméticas, movimento de dados) e o número de vezes que são executadas, lembrando sempre de desconsiderar sobrecargas de gerenciamento de memória ou E/S.

Para medir o custo de execução de um algoritmo é comum definir uma função de complexidade f. f(n) pode ser usado tanto para expressar uma função de complexidade de tempo, onde f(n) mede o tempo necessário para executar um algoritmo para um problema de tamanho n, quanto para expressar uma função de complexidade de espaço, onde f(n) mede a memoria necessária para executar um algoritmo para um problema de tamanho n. Depois que temos a função de complexidade, podemos achar a Ordem de complexidade, que vai ser representada pela letra 'O'. Quando temos $3n^2 + 5n + 1$ a ordem de complexidade vai ser $O(n^2)$, pois vamos ignorar os números que multiplicam o 'n' e considerar o maior 'n', que no caso é o n^2.

No calculo na complexidade nós temos três cenários possíveis:melhor caso(menor tempo de execução para todas entradas possíveis de tamanho) ,pior caso(maior tempo de execução) e o caso médio(média dos tempos de execução).

Vale lembrar que quando temos uma função de complexidade em que o valor de 'n' é muito baixo, qualquer algoritmo custa pouco para ser executado, logo , devemos analizar o comportamento assintótico das funções de complexidade de um programa. Uma função f(n) domina assintoticamente g(n), se existem duas constantes positivas 'c' e 'm' tais que, para $n >= m$, temos $|g(n)| <= cx|f(n)$

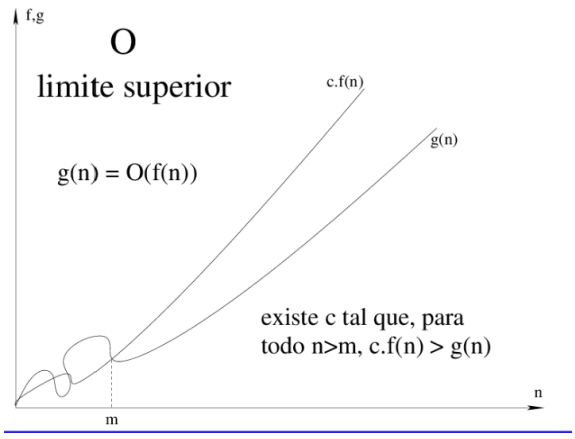

Figure 1. g(n) = O(f(n))

A classe P representa o conjunto de todos os problemas de decisão que podem ser resolvidos por um algoritmo determinístico(dado uma certa entrada, vai gerar sempre a mesma saída) em tempo polinomial. Logo a classe P possui as soluções para os problemas que são considerados trataveis.

A classe NP é o conjunto de todos os problemas que podem ser resolvidos por um algoritmo não determinístico. Um algoritmo não determinístico é aquele que possui diversas soluções para um mesmo problema e não é possível prever o resultado que será gerado pelo algoritmo.

Todos os problemas da classe NP-Completo podem ser reduzidos a um só, em outras palavras, se um algoritmo resolvesse em tempo polinomial um determinado problema NP-Completo, então esse algoritmo seria capaz de solucionar qualquer outro problema da classe NP-Completo.