ESTUDO DA COMPLEXIDADE DE ALGORITMOS

# CONTAGEM DE INSTRUÇÕES

A complexidade de tempo de um programa é a quantidade de tempo que o computador precisa para concluir a execução de determinado programa. O tempo, T(p), tomado por um programa p, é a soma do tempo de compilação e do tempo de execução das instruções. O tempo de execução de um algoritmo é obtido contando o número de instruções que ele realiza, deste modo entrará em nossa contagem:

* Atribuição de valores a variáveis;
* Chamadas de métodos;
* Operações lógicas e aritméticas;
* Comparação de dois números;
* Acesso a elemento de um array;
* Seguir uma referência de objeto (acesso a objeto);
* Retorno de um método.

Neste estudo assumiremos que as instruções possuem o mesmo custo, o que chamaremos de 1UT ou 1 Unidade de Tempo. Por outro lado, algumas instruções terão tempo 0, como as instruções de seleção.

**Tabela : código com quatro instruções**

Padrão do plano de fundo

Descrição gerada automaticamente

T(p) = 1 + 1 + 1 + = 4.

**Tabela : contagem das instruções do código com condicional (versão 2)**

**Tela de celular

Descrição gerada automaticamente com confiança média**

T(p) = 1 \* c1 \* c2 + 1 \* c3 + 1 \* c4 + 0 \* c5 + 0 + 1 \* c6

**Tabela : código com estrutura de repetição (versão 3)**

****

T (n) = 3n + 2

# CONSUMO DE TEMPO ASSINTÓTICO

A complexidade de tempo é o número de operações que um algoritmo executa para completar sua tarefa em relação ao tamanho da entrada, por isso o algoritmo que executa a tarefa no menor número de operações é considerado o mais eficiente. Usamos a notação assintótica para analisar qualquer algoritmo e, com base nisso, encontramos o algoritmo mais eficiente.

Em notação assintótica não consideramos a configuração do sistema, mas sim a ordem de crescimento da entrada. Tentamos descobrir como o tempo ou o espaço ocupado pelo algoritmo aumenta/diminui após aumentar/diminuir o tamanho da entrada.

**Melhor caso:** Esse é o limite inferior no tempo de execução de um algoritmo.

**Caso médio:** Calculamos o tempo de execução para todas as entradas possíveis, somamos todos os valores calculados e dividimos a soma pelo número total de entradas. Devemos conhecer (ou prever) a distribuição dos casos.

**Pior caso:** Esse é o limite superior no tempo de execução de um algoritmo.

Existem três principais notações assintóticas que são usadas para representar a complexidade de tempo de um algoritmo. São elas:

* Notação Θ = define um limite superior e um limite inferior, e seu algoritmo ficará entre esses níveis;
* Notação O (lê-se: Big-O) = denota o tempo máximo gasto por um algoritmo ou a complexidade de tempo do pior caso de um algoritmo;
* Notação Ω = determina que o tempo gasto pelo algoritmo não pode ser inferior ao que foi determinado.

# HABILIDADES MATEMÁTICAS NECESSÁRIAS]

Os logaritmos são realmente úteis para nos permitir trabalhar com números muito grandes enquanto manipulamos números de tamanho muito mais gerenciável.

A função logarítmica é definida como

para x>0,a>0 e a≠1,



Tabela

Descrição gerada automaticamente

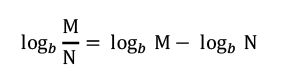
As funções logarítmicas possuem algumas das propriedades que permitem simplificar os logaritmos quando a entrada está na forma de produto, quociente ou o valor elevado à potência. Algumas das propriedades estão listadas abaixo.

* **Regra do produto:**

logb MN = logb M + logb N

A multiplicação de dois logaritmos de mesma base é igual à soma desses logaritmos de base igual.

* **Regra do quociente:**



A divisão de dois logaritmos de mesma base é igual à subtração desses logaritmos de base igual.