Análise de Algoritmo

Todo algoritmo é projetado para executar uma determinada função e, para isso, ele utiliza uma quantidade de memória e gasta um determinado tempo.

Algoritmo

- Qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como **entrada** e produz algum valor ou conjunto de valores como **saída** (Cormen, 2002).

Tipo Abstrato de Dados

- Quando os dados obtidos na entrada do algoritmo são dispostos e manipulados de **forma homogênea** no processo de computação de sua saída.
- É formado por um conjunto de **valores** e uma série de **funções** que podem ser aplicadas sobre esses valores.
- O conjunto função e valores constituem um **modelo matemático** que pode ser empregado para modelar e solucionar problemas do **mundo real**

Estrutura de Dados

- Para implementar um algoritmo é necessário representa-lo de alguma forma em uma linguagem utilizando tipos e operações.
- Na representação de TAD, emprega-se uma **estrutura de dados**
- Uma estrutura de dados é um meio para armazenar e organizar dados com o objetivo de facilitar o acesso e as modificações (Cormen, 2002).

Estrutura de Dados

- As estruturas **diferem** uma das outras pela disposição ou manipulação de seus dados.
- Não se pode **separar** as estruturas de dados e os algoritmos associados a elas.
- Todos os problemas a serem resolvidos por algoritmos possuem **dados**. Estes são armazenados em **estruturas**, escolhidas de acordo com as operações que podem ser realizadas sobre elas e com o custo de cada uma dessas operações.

Análise de Algoritmos

- É a previsão dos recursos de que o algoritmo necessitará. (Cormen , 2002)
  - Memória.
  - Largura de banda de comunicação.
  - *Hardware* de computação.
  - Tempo de computação.

Análise de Algoritmos

- Envolve dois tipos de problemas distintos (Knusth, *apud* Ziviani, 1971):

- análise de um ***algoritmo particular***: calcular o custo de um determinado algoritmo na resolução de um problema.

- análise de uma ***classe de algoritmos***: determinar o algoritmo de menor custo possível para resolver um problema.

Modelo RAM

- *Random Access Machine* (RAM) – Máquina de Acesso Aleatório
- Forma de medir o custo através de um modelo de computação genérico (modelo matemático) com um único processador
- Instruções são executadas uma após a outra, sem operações concorrentes.

Modelo RAM

- Instruções encontradas em computadores reais:
  - instruções **aritméticas** (soma, subtração, multiplicação, divisão, resto)
  - instruções de **movimentação** de dados (carregar, armazenar, copiar)
  - instruções de **controle** (desvio condicional, chamada e retorno de funções)

Complexidade Algoritmica

- O **tempo de execução** de um algoritmo: função de custo *T*, onde *T*(*n*) é a medida do **tempo necessário para executar** um algoritmo para um problema de tamanho *n*.
- *T* = função **complexidade de tempo**.
- Se *T*(*n*) é a medida de memória necessária para a execução de um algoritmo, então
- *T* = função de **complexidade de espaço**.

Aplicações

```
int calculaMenor(int A[ ], int n){
     int i, menor;
     menor = A[0];
     for (i = 1; i <= ; i++){
          if (A[i] < menor){
               menor = A[i];
          }
     }
     return menor;
}
```

Complexidade Algoritmica

- T(n) não representa diretamente o tempo de execução, mas o **número de vezes** que certa operação relevante é executada.
- No exemplo anterior, a função de complexidade de tempo é o número de elementos de A, visto que para encontrar o menor é preciso mostrar que cada um dos n – 1 elementos é menor do que algum outro.
- $T(n) = n - 1$.

Tempo de Execução

- Depende principalmente do tamanho da entrada de dados
- No exemplo anterior, o tempo de execução é uniforme para qualquer tamanho de entrada n
- Para **algoritmos de ordenação**, o tempo não é uniforme e depende se a entrada estiver ordenada

Tempo de Execução

- Busca em arquivo contendo registro de dados (sequencial)
- **Pior** *caso*: maior tempo de execução sobre todas as entradas de tamanho *n*. T(n) = **n**
- **Melhor** *caso*: menor tempo de execução sobre todas as entradas de tamanho *n*. T(n) = **1**
- **Caso médio**: média dos tempos de execução sobre todas as entradas de tamanho *n*. T(n) = **(n + 1) / 2**

Tempo de Caso Médio

- Considere p_i a probabilidade de procurar o i-ésimo registro. Para encontrar o i-ésimo registro são necessárias i comparações:

$T(n) = 1 . p_1 + 2 . p_2 + 3 . p_3 + \ldots + n . p_n$

- Considerando que a probabilidade de cada um dos registros ser encontrado é 1/n:

$T(n) = 1 . 1 / n + 2 . 1 / n + 3 . 1 / n + \ldots + n . 1 / n$

$T(n) = 1 / n . (1 + 2 + 3 + \ldots + n)$ (PA)

$T(n) = 1 / n . ((n . (n + 1)) / 2)$

$T(n) = (n + 1) / 2$

Tempo de Execução

- O tempo de execução de um algoritmo aumenta proporcionalmente ao tamanho da entrada **n** do problema.
- Escolha do algoritmo pelo desempenho em **entradas grandes**.
- Estudo da eficiência **assintótica**: observa-se a maneira como o tempo de execução de um algoritmo aumenta

Notação Assintótica

- Utilizada para **representar o comportamento** assintótico das funções de **complexidade de tempo dos algoritmos**, bem como **relacionar** o comportamento das funções de complexidade de dois algoritmos.

- Uma função g(n) **domina assintoticamente** outra função f(n) se existem duas constantes positivas c e n_0 tais que, para $\mathbf{n \geq n_0}$, temos que $\mathbf{|f(n)| \leq c \, . \, |g(n)|}$.

Análise Assintótica

- Considere $f(n) = n$ e $g(n) = -n^2$.
- É fácil observar que : $|n| \leq c \, . \, |-n^2|$
- Considerando $c = 1$ e $n_0 = 1$ $|n| \leq 1 \, . \, |-n^2|$

n	**$\|n\| \leq c \, . \, \|-n^2\|$ ($c = 1$)**
1	$1 \leq 1$
2	$2 \leq 4$
3	$3 \leq 9$
4	$4 \leq 16$

- Portanto g(n) domina assintoticamente f(n). No entanto, f(n) não domina g(n), pois $|-n^2| > c \, . \, |n|$

Exercício

- Considere $f(n) = (n + 1)^3$ e $g(n) = n^3$.
- Considere $c = 3$ e $n_0 = 3$.
- Prove que f(n) domina assintoticamente g(n).
- O inverso também é verdadeiro?