

# Fundamentos de Programação

Aula 03 - Noções de Complexidade

Profa. Elisa de Cássia Silva Rodrigues

#### Sumário

- Introdução.
- Análise de algoritmos.
- Complexidade de algoritmos.
- Atividade prática.
- Atividade para casa.

#### Introdução

- O que é um algoritmo?
  - Qualquer procedimento computacional bem definido que recebe valores de entrada e produz valores de saída.
- Quando um algoritmo resolve um problema?
  - Quando ele gera uma saída correta, para qualquer entrada, se forem concedidos tempo e memória suficientes para sua execução.
- Quais os critérios para avaliar um algoritmo?
  - Quantidade de tempo gasto.
  - Quantidade de memória utilizada.
  - Simplicidade do algoritmo.
  - Exatidão de resposta.

### Análise de Algoritmos

- O que significa análise de algoritmos?
  - Prever os recursos que o algoritmo irá necessitar.
    - \* Quantidade de memória.
    - ★ Largura de banda de comunicação.
    - \* Hardware de computação.
    - ★ Tempo de execução.
- Existem dois tipos de problemas na análise de algoritmos:
  - Análise de um algoritmo particular.
    - Calcular o custo de um determinado algoritmo para resolver um problema específico.
    - Análise do número de vezes que cada parte do algoritmo deve ser executada, seguida pelo estudo da quantidade de memória necessária.
  - Análise de uma classe de algoritmos.
    - Determinar o algoritmo de menor custo possível para resolver um problema específico.
    - \* Todos os algoritmos são investigados e o resultado é comparado para identificar o melhor possível.

- O que significa complexidade de algoritmos?
  - Mede a dificuldade (custo) para resolver um problema computacional.
  - ► Tempo (ou espaço de memória) gasto na execução de um algoritmo.
    - \* Chamamos, complexidade temporal e complexidade espacial.
- Qual é o objetivo de se calcular a complexidade?
  - Projetar algoritmos eficientes.
  - ▶ Avaliar o desempenho de um algoritmo e se possível melhorá-lo.

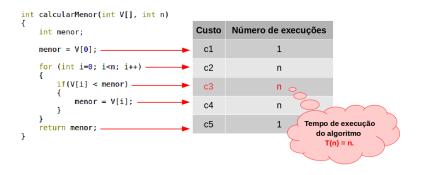
Quanto maior a complexidade, menor a eficiência do algoritmo!

- Como medir o tempo de execução de um algoritmo?
  - Por meio da sua execução em um computador real.
    - **★ Desvantagem:** resultados dependem do computador.
    - \* Ex: compilador, hardware e quantidade de memória.
  - Por meio do uso de um modelo matemático.
    - ★ Contagem do número de instruções executadas.
    - ★ Cada operação fundamental demora um período de tempo constante.
    - ★ Operações mais relevantes contribuem para o tempo de execução.
  - Quais são as operações fundamentais?
    - \* Aritméticas: soma, subtração, multiplicação, divisão, resto, piso, teto.
    - ★ Movimentação de dados: carregar, armazenar, copiar e atribuições.
    - ★ Controle: desvio condicional, chamada e retorno de sub-rotinas.

• **Exemplo 1:** Encontre o menor valor em um vetor de tamanho *n*.

Qual é o tempo de execução deste algoritmo?

• **Exemplo 1:** Encontre o menor valor em um vetor de tamanho *n*.



Tempo de execução T(n) de um algoritmo depende principalmente do tamanho da entrada de dados n!

- No exemplo anterior, o algoritmo precisa percorrer todo o vetor.
- Agora, considere um problema de busca sequencial.
  - ► Encontrar um valor específico dentre um conjunto de valores aleatórios.
  - O tempo de execução depende da entrada.
- Em problemas deste tipo, identificam-se três casos:
  - ▶ Pior caso:
    - ★ Maior tempo de execução sobre todas as entradas de tamanho n.
  - ► Melhor caso:
    - ★ Menor tempo de execução sobre todas as entradas de tamanho n.
  - ► Caso médio:
    - ★ Média dos tempos de execução sobre todas as entradas de tamanho n.

### Atividade prática

- Escreva um algoritmo que resolva o problema da busca sequencial de um valor chave inteiro em um vetor de tamanho n, sem repetição de valores. E devolva a quantidade de números comparados até encontrar o valor chave.
- Analise o algoritmo desenvolvido e identifique:
  - Melhor caso. Qual o tempo de execução do algoritmo?
  - Pior caso. Qual o tempo de execução do algoritmo?
- Faça a análise contando as instruções fundamentais do algoritmo e descreva como deve ser o vetor de entrada para cada caso.

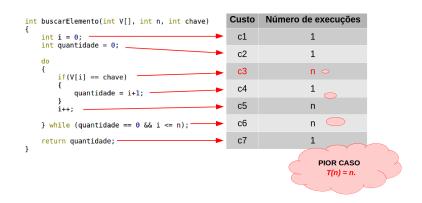
Vamos usar a linguagem C para implementar o algoritmo!

# Atividade prática



O elemento chave se encontra na primeira posição do vetor!

# Atividade prática



O elemento chave está na última posição do vetor ou não existe!

- Tempos mínimos são de pouca utilidade.
- Tempos médios são difíceis de calcular.
  - Depende da probabilidade de ocorrência das diferentes entradas.

Quando calculamos a complexidade de um algoritmo considera-se o pior caso de desempenho!

- Como é feita a análise do tempo no caso médio?
  - ▶ Considera-se uma distribuição de probabilidades sobre a entrada *n*.
  - ▶ Em geral, supõe-se que todas as entradas são igualmente possíveis.
- Qual seria a complexidade de tempo T(n) para a busca sequencial?
  - Considera-se a probabilidade p<sub>i</sub> de procurar i-ésimo registro, sendo necessárias i comparações.
  - ▶ Supondo que  $p_i = 1/n$ , temos:

$$T(n) = 1p_1 + 2p_2 + 3p_3 + \dots + np_n$$

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

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

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

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

### Complexidade Assintótica

- Note que, o tempo de execução de uma algoritmo aumenta à medida que o tamanho da entrada n do problema aumenta.
- Para valores pequenos de *n*, qualquer algoritmo gastará pouco tempo.

A análise de tempo deve ser feita sobre tamanhos grandes de entrada!

- Essa análise chama-se complexidade assintótica.
  - Preocupa-se em medir o crescimento do tempo de execução conforme o tamanho da entrada n aumenta indefinidamente.
  - A notação assintótica representa o comportamento assintótico das funções de complexidade de tempo dos algoritmos.
  - ► Classes de complexidade assintótica são definidas de acordo com o limite assintótico superior (Notação O).

# Classes de Complexidade Assintótica

- O(1): ordem constante.
  - Instruções são executadas um número fixo de vezes.
  - Não depende do tamanho dos dados de entrada.
- O(log n): ordem logarítmica.
  - Resolve um problema transformando-o em problemas menores.
- O(n): ordem linear.
  - Operações são realizadas sobre cada um dos elementos de entrada.
- O(n log n): ordem log linear.
  - Algoritmos que trabalham com particionamento dos dados.
  - Transformam um problema em vários problemas.
  - Cada problema é resolvido de forma independente e depois unidos.

# Classes de Complexidade Assintótica

- O(n2): ordem quadrática.
  - Ocorre quando os dados são processados aos pares.
  - Caracterizado pela presença de dois comandos de repetição aninhadas.
- O(n3): ordem cúbica.
  - Caracterizado pela presença de três estruturas de repetição aninhadas.
- O(2n): ordem exponencial.
  - Ocorre quando se usa uma solução de força bruta.
  - Não são úteis do ponto de vista prático.
- O(n!): ordem fatorial.
  - Possuem comportamento muito pior que o exponencial.

#### Atividade para Casa...

- Faça uma resenha do seguinte tópico do livro texto (Cormen, 2012):
  - ► Seção 2.2 Análise de Algoritmos.
- Escolha e faça dois exercícios da Seção 2.2.

#### Observação

• Prazo para entrega: início da próxima aula (valendo ponto extra).

#### Próxima aula...

- Complexidade assintótica.
  - ► Continuação...

#### Sugestão de leitura:

Capítulo 3 - Crescimento de Funções.

# Bibliografia

- ORMEN, Thomas H. et al. Algoritmos: Teoria e Prática. 2ª ed. 2012.
- ASCÊNCIO, A. F. G.; ARAÚJO, Graziela S. Estrutura de Dados: Algoritmos, Análise de Complexidade, Implementações em Java e C/C++. 2010.

Obrigada pela atenção!