Implementação de Classe para Análise de Grafos Direcionados

Arthur Morgado Teixeira, Arthur Pelisson, Mateus Melo 1, Mateus Melo 2,

Abstract Exemplo de resumo interessante e informativo.

Keywords grafos direcionados, grafos, operações com gra

1 Introdução

A disciplina de Teoria dos Grafos, ministrada pelo professor Dr.Jairo Lucas de Moraes, propôs um projeto prático com o objeto de desenvolvermos um programa, na linguagem Python ou C, capaz de analisar um grafo direcionado qualquer e retornar algumas informações sobre o mesmo. O presente artigo tem como finalidade descrever e detalhar a abordagem utilizada pelo grupo para a implementação das funcionalidades solicitadas, bem como discutir sobre as estruturas de dados empregadas. Como proposta do projeto, o programa deveria cumprir as seguintes exigências:

- Receber um grafo direcionado qualquer e:
- representá-lo nos formatos de matriz de adjacência e lista de adjacência;
- verificar a existência de laços;
- identificar suas características (simples, completo);
- analisar se o mesmo é uma árvore e caso seja, identificar seu tipo.

Com os requisitos definidos, é possível partir para a primeira etapa da implementação do programa. Quando se pretende implementar algoritmos computacionais para resolução de um problema, especialmente para problemas matemáticos, é de extrema importância estar bem fundamentado com as bases relacionadas ao conteúdo do problema. Muitas soluções algorítmicas podem ser simplificadas utilizando fundamentos matemáticos. Um exemplo de problema da qual a abordagem matemática simplifica a complexidade, é o problema de encontrar a soma dos n primeiros números $\mathbb{N}$. A abordagem mais intuitiva, que utiliza uma estrutura de repetição iterando sob o intervalo e somando os números percorridos, possui complexidade $\mathbf{O(N)}$, enquanto a abordagem utilizando a fórmula matemática para progressões aritméticas, resolve com complexidade $\mathbf{O(1)}$.

Código 1: Função soma com complexidade linear

```python
def soma(n: int) -> int:
    """
    Calcula a soma dos n primeiros números naturais, incluindo o 0.
    Implementado com estruturas de repetição.

    return:
        Número inteiro referente a soma dos números do intervalo.
    """
    total = 0

    for num in range(n):
        total += num

    return total
```

Código 2: Função soma com complexidade constante

```python
def soma(n: int) -> int:
    """
    Calcula a soma dos primeiros n números naturais, incluindo o 0.
    Implementado com a fórmula matemática de progressão aritmética.

    return:
        Número inteiro referente a soma dos n primeiros números.
    """
    return (n * (n - 1)) / 2
```

Como demostrado acima, ter conhecimento dos fundamentos matemáticos que envolvem o problema, pode gerar soluções com complexidade menor. Por conta disso, antes de descrever sobre a implementação do código, é essencial estabelecer alguns conceitos fundamentais que orientaram na tomada de decisões relacionadas aos algoritmos e estruturas adotadas.