01 - Conceitos e definições

Marco A L Barbosa malbarbo.pro.br

1. Usando a definição de caminho, justifique a afirmativa: sempre existe um caminho de comprimento 0 de um vértice u para u.
2. Liste todos os subcaminhos do caminho $\langle 1, 2, 2, 5\rangle$.
3. Dado um ciclo simples $c = \langle v_0, v_1, \ldots, v_{k-1}, v_0\rangle$, quantos caminhos distintos existem que formam o mesmo ciclo que c?
4. A definição que vimos em sala para ciclo em grafos orientados permite arestas repetidas enquanto a definição de ciclo para grafos não orientados não permite. Porque você acha que o autor fez esta distinção? Você acha mais adequado permitir ou não arestas repetidas? Veja como outros autores definem ciclo.
5. Construa um grafo orientado para representar as dependências dos conceitos relacionados com grafos orientados visto em sala (palavras em negrito no material). Cada vértice deve representar um conceito. Para cada par de conceitos a e b, deve existir uma aresta $a \to b$ se o conceito b depende do conceito a. Por exemplo, deve existir uma aresta do vértice que representa o conceito "caminho" para o vértice que representa o conceito "subcaminho", pois o conceito de subcaminho depende do conceito de caminho. Baseado neste grafo, responda:
   a. Pode existir ciclos neste grafo? Explique.
   b. Qual o conceito que tem mais dependências? Considere que se existem três conceitos a, b e c, e duas dependências $a \to b$ e $b \to c$, então a não tem dependência, b tem uma dependência, e c tem duas dependências.
6. Faça um programa que verifique se dois caminhos formam o mesmo ciclo. (Dica: veja a definição)

Soluções

1. Na definição é dito que um caminho de um vértice u até um vértice u' é uma sequência de vértices $\langle v_0, v_1, v_2, \ldots, v_k \rangle$, tal que, $u = v_0$ e $u' = v_k$ e para $i = 1, 2, \ldots, k$ existe a aresta (v_{i-1}, v_i) no grafo. Quando consideramos a sequência $\langle v_0 \rangle$, podemos notar que ela forma um caminho de v_0 a até v_0, pois o vértice inicial e o vértice final da sequência é v_0, e o conjunto de restrições de existência de arestas (v_{i-1}, v) no grafo para $i = 1, 2, \ldots, k$ é vazio (pois $k = 0$), e portanto todas as restrições são satisfeitas. Desta forma, para qualquer vértices u, sempre existe o caminho $\langle u \rangle$ de tamanho 0 de u para u.

2. Subcaminhos de tamanho 0: $\langle 1 \rangle$, $\langle 2 \rangle$, $\langle 5 \rangle$. Subcaminhos de tamanho 1: $\langle 1, 2 \rangle$, $\langle 2, 2 \rangle$, $\langle 2, 5 \rangle$. Subcaminhos de tamanho 2: $\langle 1, 2, 2 \rangle$, $\langle 2, 2, 5 \rangle$ Subcaminhos de tamanho 3: $\langle 1, 2, 2, 5 \rangle$.

6. Segundo a definição, dois caminhos $a = \langle a_0, a_1, \ldots, a_{k-1}, a_0 \rangle$ e $b = \langle b_0, b_1, \ldots, b_{k-1}, b_0 \rangle$ formam o mesmo ciclo se existe um inteiro j tal que $b_i = a_{(i+j) \mod k}$ para $i = 0, 1, \ldots, k-1$.

   A função a seguir escrita em Python implemente de forma direta a verificação se dois caminhos a e b formam o mesmo ciclo. O programa tenta valores para $j = 0, 1, \ldots, k-1$ e verifica se $a_i = b_{(i+j) \mod k}$ é verdadeiro para $i = 0, 1, \ldots, k-1$. Se a verificação for verdadeira para algum valor de j, a função devolve `True`, caso contrário, `False`.

```python
def mesmo_ciclo(a, b):
    '''
    Exemplos
    >>> mesmo_ciclo([2, 2], [2, 2])
    True
    >>> mesmo_ciclo([2, 2], [1, 1])
    False
    >>> mesmo_ciclo([2, 4, 2], [2, 4, 2])
    True
    >>> mesmo_ciclo([2, 4, 2], [2, 3, 2])
    False
    >>> mesmo_ciclo([4, 2, 4], [2, 4, 2])
    True
    >>> mesmo_ciclo([1, 2, 3, 1], [2, 3, 1, 2])
    True
    >>> mesmo_ciclo([1, 2, 3, 1], [3, 1, 2, 3])
    True
    >>> mesmo_ciclo([1, 2, 3, 1], [3, 4, 2, 3])
    False
    >>> mesmo_ciclo([1, 2, 3, 4, 1], [3, 4, 1, 2, 3])
    True
    >>> mesmo_ciclo([1, 2, 3, 4, 1], [3, 4, 1, 2, 1])
    False
    >>> mesmo_ciclo([1, 2, 3, 4, 1], [3, 4, 1, 2, 2])
    False
    '''
    # Se a ou b não formam um ciclo ou se tem tamanho diferente, devolve False
    if len(a) < 2 or len(b) < 2 or a[0] != a[-1] or b[0] != b[-1] or len(a) != len(b):
        return False
    k = len(a) - 1
    for j in range(0, k):
        if all(b[i] == a[(i + j) % k] for i in range(0, k)):
            return True
    return False
```