Estrutura de dados II – Exercício de programação 1: Vértices de articulação

Um grafo pode conter diversos grupos internos – também conhecidos como componentes conectados – que possuem a seguinte propriedade: dado um componente conectado, cada vértice do componente pode alcançar o outro e não alcançam vértices de outros componentes. Porém, ocasionalmente, vértices especiais denominados vértices de articulação podem nascer. Tal vértice , quando apagado, aumenta o número de componentes conectados do grafo. Portanto, esse vértice funciona como "ponte" entre os componentes.

A propriedade desse vértice, mais formalmente falando, consiste no fato de que, dado uma árvore de busca binária, nenhum descendente do vértice possui uma *back edge* para um vértice ancestral. Em outras palavras, é necessário detectar se o tempo de descobrimento dos vértices descendentes ou os vértices que eles atingem com *back edges* (o que for menor) são sempre maiores que o tempo de descobrimento do vértice candidato. Se isso for o caso, então o vértice é um ponto de articulação.

O algoritmo implementado faz exatamente isso. Se o vértice candidato não possui ancestrais, então ele automaticamente não pode ser um vértice de articulação. Tirando esse caso específico, o algoritmo varre os vértices da árvore de busca verificando se seu menor tempo é superior ao tempo de descoberta do vértice candidato. O **menor tempo** é, inicialmente, o tempo de descoberta do vértice. Porém, se ele possuir uma *back edge* para um vértice *v* e o tempo de descoberta de *v* for menor, esse valor é armazenado no lugar.

Dessa forma, se não existir nenhum vértice com menor tempo inferior ao do vértice candidato, então um vértice de articulação foi encontrado. Caso exista um vértice que possui uma *back edge* para um vértice de tempo inferior ao do vértice candidato, então a condição é violada e o vértice candidato pode ser descartado.