IDENTIFICAÇÃO DOS VÉRTICES DE ARTICULAÇÃO

Os vértices de articulação são identificados a partir do algoritmo de busca em profundidade, iniciando a busca pelo vértice de menor índice. Os índices são percorridos até que se chegue ao último nó da lista de adjacência, onde cada um dos nós são marcados segundo as visitas, sendo BRANCO = 0 quando o nó ainda não tenha sido visitado, CINZA = 1 na primeira visita e PRETO = 2 quando além do nó ter sido visitado, todos os seus adjacentes também o foram. E para cada nó da lista de adjacência, seus adjacentes também são percorridos, enquanto existir.

Um nó folha nunca pode ser um vértice de articulação, já que o nó folha é o último vértice a ser acessado em uma árvore de busca, i.e., não há um próximo destino a partir do nó folha. Tomando como exemplo a Figura 1, o vértice **3** não poderia ser um nó folha, já que existem outros caminhos para percorrer a partir deste e um nó folha não teria. Portanto não haveria como remover um nó folha e gerar mais um componente conexo maximal. Já um nó raiz pode ser um vértice de articulação quando este possuir dois descendentes diretos ou mais. Portanto, foi criado um contador de descendentes diretos de um dado vértice **vAtual**. Como esta variável é declarada e inicializada nas chamadas recursivas, é apenas contabilizado os descendentes diretos de cada vértice vAtual da chamada recursiva. O laço da função visitaBP tem a finalidade de fazer o percurso da busca em profundidade do primeiro índice da listaAdj[]. Após o laço é verificado se o nó é raiz, caso o antecessor deste seja igual a -1, i.e., não existe antecessor, significa que este é raiz. E se o nº de descendentes diretos for de dois ou mais, significa que este é um vértice de articulação e é adicionado ao vetor **verticesArticulacao[*iPA]**, onde iPA é o índice ponto de articulação deste vetor.

E para os nós que não são nem raiz nem folha (quando antecessor[vAtual] != -1, isto é, vAtual possui um antecessor), tal nó **vAtual** só será um vértice de articulação caso este seja o único caminho (ponto de acesso) entre os nós acima deste (chamarei de **a**) e os abaixo dele. E para que este nó, que esteja entre **a** (vértice acima de vAtual) e um ou mais vértices abaixo seja o único caminho, não podem existir arestas que conectem **a** com os descendentes de **vAtual**, ou seja, uma aresta de retorno (ilustrado na Figura 2, onde vAtual é 3). Então foi criado o **arestaRet[]** que conterá o passo em que um vértice foi descoberto e que possua uma aresta de retorno entre um descendente de **vAtual** com seu vértice antecessor **a**. Caso não exista uma aresta de retorno que conecte estes vértices, i.e., **a** não seja alcançável por um descendente de **vAtual**, então **vAtual** é um vértice de articulação. Então se a cor do próximo vértice de vAtual seja BRANCO, o antecessor de

vProx será vAtual e a arestaRet recebe o vértice descoberto de menor passo de visita entre a arestaRet[vAtual] e a arestaRet[vProx], e se ainda assim arestaRet[vProx] possuir um valor do passo em que este foi descoberto maior que o tempo de descoberta do vértice vAtual (tdesc[vAtual]), então não existe conexão entre um vértice **a** e um descendente de **vAtual**, ou seja, **vAtual** é um vértice de articulação. Por fim é incrementado a quantidade de descendentes, o ponteiro responsável por percorrer os adjacentes do nó listaAdj[vAtual] passa a apontar para o próximo e verificado se este nó é raiz, caso este não satisfaça a condição anterior onde antecessor[vAtual] != -1. E para os dois casos onde o vértice é raiz ou não, se este for um vértice de articulação, é incrementado em 1 o índice iPA, assim guardando os próximos vértices de articulação, se existirem.

Figura 1 : Grafo

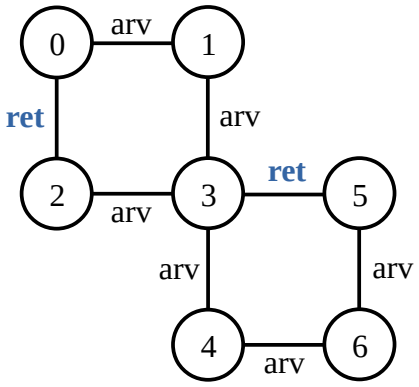

Figura 2 : Árvore da busca em profundidade