7.3 Busca em Profundidade

A **busca em profundidade** (do inglês ***depth-first search***) é um algoritmo para caminhar no grafo. A estratégia seguida pelo algoritmo é a de buscar, sempre que possível, o mais profundo no grafo. Na busca em profundidade, as arestas são exploradas a partir do vértice v mais recentemente descoberto que ainda possui arestas não exploradas saindo dele. Quando todas as arestas adjacentes a v tiverem sido exploradas, a busca anda para trás (do inglês *backtrack*) para explorar vértices que saem do vértice do qual v foi descoberto. O processo continua até que sejam descobertos todos os vértices alcançáveis a partir do vértice original. O algoritmo é a base para muitos outros algoritmos importantes, tais como verificação de grafos acíclicos (Seção 7.4), ordenação topológica (Seção 7.6) e componentes fortemente conectados (Seção 7.7).

Sempre que um vértice v é descoberto durante a leitura da lista de adjacentes de um vértice u já descoberto, a busca em profundidade registra esse evento atribuindo u a Antecessor[v]. Para acompanhar o progresso do algoritmo, cada vértice é colorido de branco, cinza ou preto. Todos os vértices são inicializados brancos e podem posteriormente se tornar cinza e, finalmente pretos. Quando um vértice é descoberto pela primeira vez durante a busca, ele torna-se cinza, e muda para preto depois que sua lista de adjacentes é completamente examinada.

A busca em profundidade registra em $d[v]$ o tempo (ou momento) em que o vértice é descoberto (e tornado cinza), e em $t[v]$ o tempo em que a busca termina o exame da lista de adjacentes de v (e tornado preto). A razão de usar os tempos de descoberta e de término é que eles são empregados em muitos algoritmos para grafos, além de serem úteis para acompanhar o comportamento da busca em profundidade. Esses registros são inteiros entre 1 e $2|V|$, pois existe um evento de descoberta e um evento de término para cada um dos $|V|$ vértices. O Programa 7.9 implementa a busca em profundidade. O grafo $G(V, A)$ pode ser direcionado ou não direcionado. A variável Tempo é usada para marcar o tempo de descoberta e de término.

O procedimento BuscaEmProfundidade funciona como se segue. Na primeira linha, a variável global Tempo, usada para registrar os tempos de descoberta e de término, é inicializada com zero. O primeiro anel logo a seguir colore todos os vértices de branco e inicializa os seus antecessores para -1 na variável Antecessor. O anel seguinte verifica cada vértice em V e, quando um vértice branco é encontrado, visita-o usando VisitaDfs. Nesse caso, toda vez que VisitaDfs(u) é chamado, o vértice u torna-se a raiz de uma nova **árvore de busca em profundidade**, e o conjunto de árvores forma uma **floresta** de árvores de busca.

Em cada chamada VisitaDfs(u), o vértice u é inicialmente branco. Na primeira linha, u é tornado cinza, a variável Tempo é incrementada, e o novo valor de Tempo é registrado como o tempo de descoberta $d[u]$. O comando **if** seguinte examina a lista de vértices v adjacentes a u e visita recursivamente v se ele for

branco. Quando VisitaDfs retorna, cada vértice u possui um tempo de descoberta $d[u]$ e um tempo de término $t[u]$.

Programa 7.9 *Busca em profundidade*

```
procedure BuscaEmProfundidade (var Grafo: TipoGrafo);
var
  Tempo      : TipoValorTempo;
  x          : TipoValorVertice;
  d, t       : array[TipoValorVertice] of TipoValorTempo;
  Cor        : array[TipoValorVertice] of TipoCor;
  Antecessor : array[TipoValorVertice] of integer;

  procedure VisitaDfs (u:TipoValorVertice);
  var FimListaAdj: boolean;
      Peso       : TipoValorAresta;
      Aux        : TipoApontador;
      v          : TipoValorVertice;
  begin
    Cor[u] := cinza;
    Tempo := Tempo + 1;
    d[u] := Tempo;
    writeln('Visita',u:2,' Tempo descoberta:',d[u]:2,' cinza'); readln;
    if not ListaAdjVazia (u, Grafo)
    then begin
          Aux := PrimeiroListaAdj (u, Grafo);
          FimListaAdj := false;
          while not FimListaAdj do
            begin
            ProxAdj (u, v, Peso, Aux, FimListaAdj);
            if Cor[v] = branco
            then begin
                  Antecessor[v] := u;  VisitaDfs (v);
                 end;
            end;
         end;
    Cor[u] := preto;
    Tempo := Tempo + 1;  t[u] := Tempo;
    writeln ('Visita',u:2,' Tempo termino:',t[u]:2,' preto'); readln;
  end; { VisitaDfs }

begin
  Tempo := 0;
  for x := 0 to Grafo.NumVertices-1 do
    begin  Cor[x] := branco;  Antecessor[x] := -1;  end;
  for x := 0 to Grafo.Numvertices-1 do
    if Cor[x] = branco then VisitaDfs (x);
end; { BuscaEmProfundidade }
```