Continuação do Programa 7.11

```
      if not ListaAdjVazia (u, Grafo)
      then begin
           Aux := PrimeiroListaAdj (u,Grafo); FimListaAdj := false;
           while FimListaAdj = false do
             begin
             ProxAdj (u, v, Peso, Aux, FimListaAdj);
             if Cor[v] = branco
             then begin
                  Cor[v] := cinza;  Dist[v] := Dist[u] + 1;
                  Antecessor[v] := u;
                  Item.Vertice := v;  Item.Peso := Peso;
                  Enfileira (Item, Fila);
                  end;
             end;
           end;
      Cor[u] := preto;
      write ('Visita', u:2,' Dist',Dist[u]:2,' cor: preto F:');
      ImprimeFila (Fila); readln;
      end;
  end; { VisitaBfs }

begin { BuscaEmLargura }
  for x := 0 to Grafo.NumVertices-1 do
    begin
    Cor[x] := branco;  Dist[x] := INFINITO;  Antecessor[x] := -1;
    end;
  for x := 0 to Grafo.NumVertices-1 do
    if Cor[x] = branco then VisitaBfs (x);
end; { BuscaEmLargura }
```

O procedimento BuscaEmLargura funciona como se segue. O primeiro anel **for** colore todos os vértices de branco, inicializa a distância $Dist[u]$ do vértice origem até o vértice u para o valor infinito, e inicializa os seus antecessores para -1 (*nil*) na variável Antecessor. Se u não tem antecessor, como nos casos em que u corresponde ao vértice origem ou u ainda não foi descoberto, então a variável Antecessor $= -1$. O anel seguinte **for** verifica cada vértice em V e, quando um vértice branco for encontrado, visita-o usando VisitaBfs. Nesse caso, toda vez que VisitaBfs(u) é chamado, o vértice u se torna a raiz de uma nova **árvore de busca em largura**, e o conjunto de árvores forma uma **floresta** de árvores de busca.

Dentro do procedimento VisitaBfs, o vértice u é considerado como origem da nova árvore e é pintado de cinza, uma vez que ele é considerado descoberto quando o procedimento inicia. O algoritmo usa uma **fila** do tipo "primeiro-que-chega, primeiro-atendido" (veja Seção 3.3) para gerenciar o conjunto de vértices cinza. A seguir $Dist[u]$ é inicializado como 0 e a Fila é inicializada com o vértice

origem u. O primeiro anel **while** é executado enquanto houver vértices cinza, que formam o conjunto de vértices descobertos que ainda não tiveram suas listas de adjacentes totalmente examinadas. Assim, no teste do **while**, a Fila contém o conjunto de vértices cinza. Antes da primeira iteração do anel, o único vértice na Fila é o vértice origem u. No primeiro comando dentro do anel, o vértice cinza u que está no início da Fila é desenfileirado. O comando **if** seguinte examina a lista de vértices v adjacentes a u e visita v se ele for branco. Nesse momento, v é tornado cinza, a distância de v a u é registrada fazendo $Dist[v] = Dist[u]+1$, u é atribuído a Antecessor[v], e o novo vértice cinza é enfileirado em Fila. Finalmente, depois que toda a lista de adjacentes de u é percorrida, o vértice u é pintado de preto.

A Figura 7.12 mostra o progresso da busca em largura no grafo não direcionado da Figura 7.5. Arestas de árvore são mostradas em negrito na medida em que são criadas pela busca em largura. Ao lado de cada vértice é mostrada a cor branca, cinza ou preta (b, c ou p), e entre parênteses a distância $Dist[u]$. A fila F é mostrada no final de cada iteração do anel **while** do procedimento VisitaBfs no Programa 7.11.

Análise A análise é válida para quando a implementação utilizar listas de adjacência para representar o grafo (veja Seção 7.2.2). O custo de inicialização do primeiro anel em BuscaEmLargura é $O(|V|)$ cada um. Para o segundo anel, o custo é também $O(|V|)$, a menos da chamada do procedimento VisitaBfs(u). Dentro do procedimento VisitaBfs, nenhum vértice é tornado branco, o que garante que cada vértice é tornado cinza no máximo uma vez, enfileirado também no máximo uma vez, e assim desenfileirado também no máximo uma vez. Como as operações de enfileirar e desenfileirar têm custo $O(1)$ cada uma, as operações com a fila têm custo total $O(|V|)$. A lista de adjacentes de cada vértice é percorrida apenas quando o vértice é desenfileirado, logo, cada lista de adjacentes é percorrida no máximo uma vez. Já que a soma de todas as listas de adjacentes é $O(|A|)$, o tempo total gasto com as listas de adjacentes é $O(|A|)$. Logo, o procedimento BuscaEmLargura possui complexidade de tempo total igual a $O(|V| + |A|)$.

Caminhos mais Curtos

O procedimento VisitaBfs do Programa 7.11 obtém a distância do vértice origem $u \in V$ para cada vértice alcançável $v \in V$ em um grafo $G = (V, A)$. De fato, a busca em largura obtém o **caminho mais curto** de u até v. Como mostra a Figura 7.12, o procedimento VisitaBfs constrói uma árvore de busca em largura durante a busca no grafo armazenada na variável Antecessor.

O Programa 7.12 imprime os vértices do caminho mais curto entre o vértice origem e outro vértice qualquer do grafo, a partir do vetor Antecessor obtido após a execução do Programa 7.11. O procedimento ImprimeCaminho tem custo linear no número de vértices do caminho impresso, uma vez que cada chamada recursiva ocorre para um caminho que tem um vértice a menos que a chamada anterior.