



Figura 5.16 Grafo acíclico com $M = 15$ vértices e $N = 12$ arestas e sua representação usando listas de arestas incidentes a cada vértice.

$\dots + g[v_{r-1}]) \bmod N$ seja igual ao rótulo da aresta a . Na representação de um hipergrafo apresentada na Seção 7.10, o conjunto de arestas é implementado por um arranjo de N arestas e cada aresta é, portanto, indexada de $0 \leq i_a < N$. No exemplo aqui descrito os índices das arestas correspondem a seus rótulos ou pesos. Ao verificar se o grafo da Figura 5.16 é acíclico o Programa 7.10 retorna os índices das arestas retiradas no arranjo $\mathcal{L} = (2, 1, 10, 11, 5, 9, 7, 6, 0, 3, 4, 8)$.

O arranjo \mathcal{L} indica a ordem de retirada das arestas, isto é, a primeira aresta retirada foi $a = (0, 10)$ de índice $i_a = 2$; a segunda, $a = (2, 14)$ de índice $i_a = 1$; e assim sucessivamente. As arestas do arranjo \mathcal{L} devem ser consideradas da direita para a esquerda, na ordem contrária em que foram retiradas no procedimento que verifica se o grafo é acíclico. Essa é uma condição suficiente para ter sucesso na criação do arranjo g . Para o grafo da Figura 5.16, a aresta $a = (4, 11)$ de índice $i_a = 8$ é a primeira a ser processada. Como inicialmente $g[4] = g[11] = -1$, fazemos $g[11] = N$ e $g[4] = i_a - g[11] \bmod N = 8 - 12 \bmod 12 = 8$. Para a próxima aresta $a = (4, 12)$ de índice $i_a = 4$, como $g[4] = 8$, temos que $g[12] = i_a - g[4] \bmod N = 4 - 8 \bmod 12 = 8$, e assim sucessivamente até a última aresta de \mathcal{L} .

O Programa 5.30 mostra o procedimento para obter o arranjo g a partir de um hipergrafo. O procedimento foi proposto por Czech, Havas e Majewski (1997). Inicialmente todas as entradas do arranjo g são feitas igual a *Indefinido* $= -1$. Dada uma aresta a com r vértices v_0, v_1, v_{r-1} e rótulo i_a , seja $u = v_j$ tal que $g[v_j] = \textit{Indefinido}$ e $j = \min\{0, \dots, r - 1\}$. Isto é, v_j é o primeiro vértice de a ainda não atribuído. Atribua o valor N para $g[v_{j+1}], \dots, g[v_{r-1}]$ que ainda estão indefinidos e faça $g[v_j] = (i_a - \sum_{v_i \in a \wedge g[v_i] \neq -1} g[v_i]) \bmod N$.

Programa 5.30 Rotula grafo e atribui valores para o arranjo g

```
Procedure Atribui (var Grafo: TipoGrafo;
var L : TipoArranjoArestas;
var g : Tipog);
var
i, u, Soma: integer;
v: TipoValorVertice; a: TipoAresta;
begin
for i := Grafo.NumVertices - 1 downto 0 do g[i] := INDEFINIDO;
for i := Grafo.NumArestas - 1 downto 0 do
begin
a := L[i]; Soma := 0;
for v := Grafo.r - 1 downto 0 do
if g[a.Vertices[v]] = INDEFINIDO
then begin
u := a.Vertices[v];
g[u] := Grafo.NumArestas;
end
else Soma := Soma + g[a.Vertices[v]];
g[u] := a.Peso - Soma;
if g[u] < 0 then g[u] := g[u] + (Grafo.r - 1) * Grafo.NumArestas;
end;
end; { -Fim Atribui - }
```

O Programa 5.31 mostra os principais passos para obter uma função de transformação perfeita. O programa gera hipergrafos randômicos iterativamente e testa se o grafo gerado é acíclico. Cada iteração gera novas funções h_0, h_1, \dots, h_{r-1} até que um grafo acíclico seja obtido. A função de transformação perfeita passa a ser determinada pelos pesos p_0, p_1, \dots, p_{r-1} , e pelo arranjo g .

Programa 5.31 Programa para obter função de transformação perfeita

```
Program ObtemHashingPerfeito;
begin
Ler conjunto de N chaves;
Ler o valor de M;
Ler o valor de r;
repeat
Gera os pesos  $p_0[i], p_1[i], \dots, p_{r-1}[i]$  para  $1 \leq i \leq \text{MAXTAMCHAVE}$ ;
Gera o hipergrafo  $G_r = (V, A)$ ;
GrafoAciclico (G_r, L, GAiclico)
until GAiclico;
Atribui (G, L, g);
Retorna  $p_0[i], p_1[i], \dots, p_{r-1}[i]$  e  $g$ ;
end.
```

O Programa 5.32 apresenta as estruturas de dados usadas pelo programa que obtém a função de transformação perfeita.