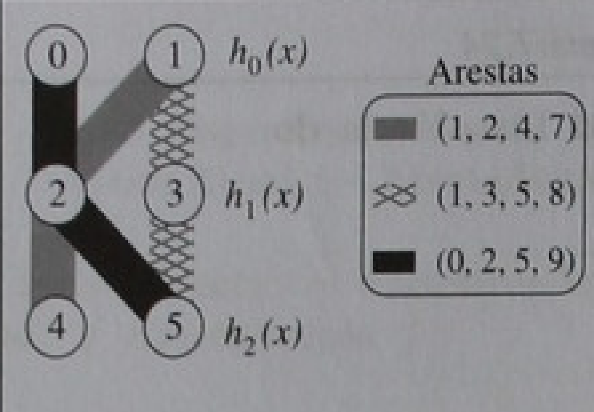

(a)

Arestas $|A|=3$

0	1	2
(1,2,4)	(1,3,5)	(0,2,5)

Prim $|V|=3$

0	1	2	3	4	5
2	1	5	4	6	8

Prox $r|A|=3x3=9$

0	1	2	3	4	5	6	7	8
−1	0	−1	−1	−1	3	−1	−1	7

(b)

***Figura 7.23** (a) 3-grafo com 6 vértices e 3 arestas; (b) Representação do 3-grafo.*

assim por diante definem as arestas subsequentes que contêm v. O arranjo $Prim$ deve possuir $|V|$ entradas, uma para cada vértice. O arranjo $Prox$ deve possuir $r|A|$ entradas, pois cada aresta a é armazenada na lista de arestas incidentes a cada um de seus r vértices. Assim, a representação por listas de incidências possui uma complexidade de espaço $O(|V| + |A|)$, pois $Arestas$ tem tamanho $O(|A|)$, $Prim$ tem tamanho $O(|V|)$ e $Prox$ tem tamanho $r \times |A|$, o que pode ser considerado $O(|A|)$ porque r é geralmente uma constante pequena.

Para descobrir quais são as arestas que contêm determinado vértice v, é preciso percorrer a lista de arestas que inicia em $Prim[v]$ e termina quando $Prox[\ldots Prim[v] \ldots] = -1$. Assim, para se ter acesso a uma aresta a armazenada em $Arestas[a]$, é preciso tomar os valores armazenados nos arranjos $Prim$ e $Prox$ módulo $|A|$. O valor -1 é utilizado para finalizar a lista. Por exemplo, ao se percorrer a lista das arestas do vértice 2, os valores $\{5, 3\}$ são obtidos, os quais representam as arestas que contêm o vértice 2 (arestas 2 e 0), ou seja, $\{5 \bmod 3 = 2, 3 \bmod 3 = 0\}$.

Os valores armazenados nos arranjos $Prim$ e $Prox$ são obtidos pela equação $i = a + j|A|$, sendo $0 \leq j \leq r-1$ e a um índice de uma aresta no arranjo $Arestas$. Inicialmente as entradas do arranjo $Prim$ são iniciadas com -1. Ao inserir a aresta $a = 0$ contendo os vértices $(1, 2, 4)$, temos que: $i = 0 + 0 \times 3 = 0$, $Prox[i = 0] = Prim[1] = -1$ e $Prim[1] = i = 0$, $i = 0 + 1 \times 3 = 3$, $Prox[i = 3] = Prim[2] = -1$ e $Prim[2] = i = 3$, $i = 0 + 2 \times 3 = 6$, $Prox[i = 6] = Prim[4] = -1$ e $Prim[4] = i = 6$. Ao inserir a aresta $a = 1$ contendo os vértices $(1, 3, 5)$ temos que: $i = 1 + 0 \times 3 = 1$, $Prox[i = 1] = Prim[1] = 0$ e $Prim[1] = i = 1$, $i = 1 + 1 \times 3 = 4$, $Prox[i = 4] = Prim[3] = -1$ e $Prim[3] = i = 4$, $i = 1 + 2 \times 3 = 7$, $Prox[i = 7] = Prim[5] = -1$ e $Prim[5] = i = 7$. Finalmente, ao inserir a aresta $a = 2$ contendo os vértices $(0, 2, 5)$ temos que: $i = 2 + 0 \times 3 = 2$, $Prox[i = 2] = Prim[0] = -1$ e $Prim[0] = i = 2$, $i = 2 + 1 \times 3 = 5$, $Prox[i = 5] = Prim[2] = 3$ e $Prim[2] = i = 5$, $i = 2 + 2 \times 3 = 8$, $Prox[i = 8] = Prim[5] = 7$ e $Prim[5] = i = 8$.

O Programa 7.25 apresenta a estrutura de dados do **tipo abstrato de dados hipergrafo** utilizando listas de incidência implementadas por meio de arranjos. A estrutura de dados contém os três arranjos necessários para representar um hipergrafo, como ilustrado na Figura 7.23. A variável r é utilizada para armazenar a ordem do hipergrafo, a variável NumVertices contém o número de vértices do hipergrafo, a variável NumArestas contém o número de arestas do hipergrafo e

a variável ProxDisponivel contém a próxima posição disponível para inserção de uma nova aresta.

***Programa 7.25** Estrutura do tipo hipergrafo implementado como listas de adjacência usando arranjos*

```
const
  MAXNUMVERTICES = 100;
  MAXNUMARESTAS  = 4500;
  MAXR           = 5;
  MAXTAMPROX     = MAXR * MAXNUMARESTAS;
  INDEFINIDO     = -1;
type
  TipoValorVertice      = -1..MAXNUMVERTICES;
  TipoValorAresta       = 0..MAXNUMARESTAS;
  Tipor                 = 0..MAXR;
  TipoMaxTamProx        = -1..MAXTAMPROX;
  TipoPesoAresta        = integer;
  TipoArranjoVertices   = array[Tipor] of TipoValorVertice;
  TipoAresta            = record
                            Vertices: TipoArranjoVertices;
                            Peso    : TipoPesoAresta;
                          end;
  TipoArranjoArestas    = array[TipoValorAresta] of TipoAresta;
  TipoGrafo =
    record
      Arestas        : TipoArranjoArestas;
      Prim           : array[TipoValorVertice] of TipoMaxTamProx;
      Prox           : array[TipoMaxTamProx] of TipoMaxTamProx;
      ProxDisponivel : TipoMaxTamProx;
      NumVertices    : TipoValorVertice;
      NumArestas     : TipoValorAresta;
      r              : Tipor;
    end;
  TipoApontador = integer;
```

Uma possível implementação para as primeiras seis operações definidas anteriormente é mostrada no Programa 7.26. O procedimento ArestasIguais permite a comparação de duas arestas cujos vértices podem estar em qualquer ordem. Assim, o custo do procedimento ArestasIguais é $O(r^2)$. O procedimento InsereAresta insere uma aresta no grafo a um custo $O(r)$. O procedimento ExisteAresta verifica se uma aresta está presente no grafo, a um custo equivalente ao **grau** de cada vértice da aresta no grafo. O procedimento RetiraAresta primeiro localiza a aresta no grafo, retira a mesma da lista de arestas incidentes a cada vértice em Prim e Prox e marca a aresta como removida no arranjo Arestas. Cabe ressaltar que a aresta marcada no arranjo Arestas não é reutilizada, pois não estamos mantendo uma lista de posições vazias.