3 -REPPRESENTAÇÃO E IMPLEMENTAÇÃO

Última revisão: 29/3/2000

Uma maneira natural de representar um grafo no computador é utilizar uma matriz, aproveitando assim de todas as manipulações permitidas pela álgebra linear. Basicamente, existem dois tipos de matri zes para representar de maneira não equívoca um grafo: matrizes de incidência e matrizes de adjacência. Vamos discutir a duas, e em seguida representações mais compactas.

Matriz de incidência

Seja G um grafo de *n* vértices v_1, v_2... v_n, e *m* arestas a_1, a_2...a_m, e nenhum laço. A matriz de incidência é uma matriz *n x m*, onde o valor de cada elemento e_{jk} da matriz é determinado da seguinte maneira:

e_{jk} = 1, se a aresta a_k é incidente ao vértice a_j
= 0 senão

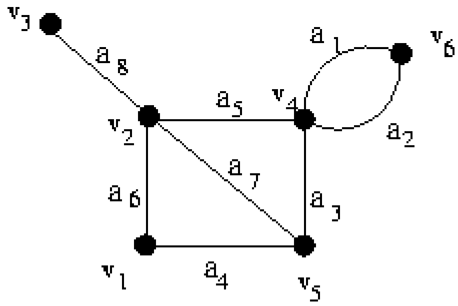

Figura 1

A matriz de incidência do grafo da figura 1 é a seguinte:

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
v_1	0	0	0	1	0	1	0	0
v_2	0	0	0	0	1	1	1	1
v_3	0	0	0	0	0	0	0	1
v_4	1	1	1	0	1	0	0	0
v_5	0	0	1	1	0	0	1	0
v_6	1	1	0	0	0	0	0	0

Eis as propriedades interessantes da matriz de incidência:

1. Como cada aresta é incidente a exatamente dois vértices, cada coluna contém exatamente dois 1.
2. O número de 1 em cada linha é igual ao grau do vértice correspondente.
3. Uma linha que contém somente 0 representa um vértice isolado.
4. Arestas paralelas resultam em duas colunas idênticas.
5. Se um grafo G é desconexo e contém dois componentes g_1 e g_2, a matriz de incidência A(G) pode ser escrita da seguinte maneira:

$$\begin{matrix} A(g_1) & 0 \\ 0 & A(g_2) \end{matrix}$$

Para entender melhor, considere o grafo da figura 2:

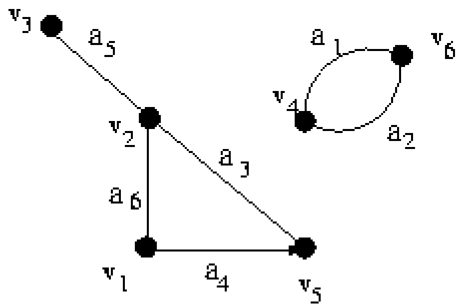

Figura 2

Eis uma matriz de incidência para esse grafo:

	a_1	a_2	a_3	a_4	a_5	a_6
v_1	0	0	0	1	0	1
v_2	0	0	1	0	1	1
v_3	0	0	0	0	1	0
v_4	1	1	0	0	0	0
v_5	0	0	1	1	0	0
v_6	1	1	0	0	0	0

Substituindo a ordem v_1, v_2, v_3, v_4, v_5, v_6 dos vértices pela ordem v_4, v_6, v_1, v_2, v_3, v_5, obtemos a seguinte matriz:

	a_1	a_2	a_3	a_4	a_5	a_6
v_4	1	1	0	0	0	0
v_6	1	1	0	0	0	0
v_1	0	0	0	1	0	1
v_2	0	0	1	0	1	1
v_3	0	0	0	0	1	0
v_5	0	0	1	1	0	0

6. A permutação de duas linhas ou duas colunas resulta em um grafo isomorfo.

Multi-grafo

A representação de um multi-grafo não exige nenhum tratamento especial. Simplemente, a matriz conterá algumas colunas idênticas que correspondem a arestas paralelas.

Pseudo-grafo

Para representar um grafo que contém um laço, simplesmente teremos, na coluna que corresponde à aresta, um 1 só que identifica o vértice que contém esse laço. Nessa representação, perdemos a propriedade de ter dois 1 em cada coluna. Uma consequência disso é que será mais difícil detectar erros na construção da matriz (ou na transmissão).

Grafo direcionado

Nesse caso, é preciso distinguir as arestas convergentes das arestas divergentes. Em outras palavras, para cada aresta, temos que especificar de qual vértice ela vem e no qual ela chega. Podemos simplesmente utilizar 1 no primeiro caso e -1 no segundo. Veja a representação do grafo da figura 3:

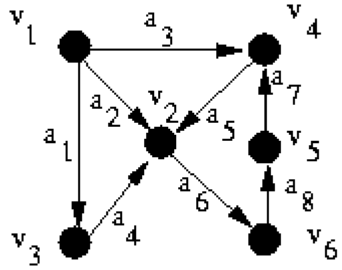

Figura 3

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
v_1	1	1	1	0	0	0	0	0
v_2	0	-1	0	-1	-1	1	0	0
v_3	-1	0	0	1	0	0	0	0
v_4	0	0	-1	0	1	0	-1	0
v_5	0	0	0	0	0	0	1	-1
v_6	0	0	0	0	0	-1	0	1

Grafo rotulado em arestas

Quando as arestas tém um valor associado (ou qualquer outro rótulo), uma modificaçao possível é colocar esse valor ao invéz de 1. Mas isso tem a desvantagem de ser redundante, pois o valor seria colocado duas vezes para cada arestas. Para usar menos espaço de memória seria melhor associar em uma tabela separada cada aresta com o seu rótulo.

Matriz de adjacência

Seja G um grafo simples de *n* vértices $v_1, v_2 ... v_n$. A matriz de adjacência é uma matriz *n x n*, onde o valor de cada elemento e_{jk} da matriz é determinado da seguinte maneira:

> e_{jk} = 1, se os vértices v_j e v_k são ligados por uma aresta
> = 0 senão

Eis a matriz de adjacência do grafo da figura 4.

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	1	0
v_2	1	0	1	1	1	0
v_3	0	1	0	0	0	0
v_4	0	1	0	0	1	1
v_5	1	1	0	1	0	0

v_6	0	0	0	1	0	0

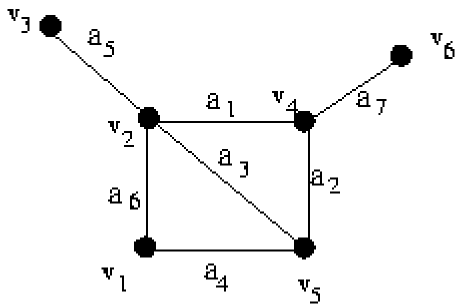

Figura 4

Eis as propriedades interessantes da matriz de adjacência:

1. Os valores da diagonal principal da matriz são 0.
2. O grau de um vértice é igual ao número de 1 na linha ou coluna correspondente ao vértice.
3. Permutações de colunas e das linhas correspondentes resultam em uma representação de um grafo isomorfo.
4. Seja um grafo G desconexo que contém dois componentes g_1 e g_2. A matriz de adjacêncica X(G) pode ser escrita em função das duas matrizes de $X(g_1)$ e $X(g_2)$:

$$\begin{matrix} X(g_1) & 0 \\ 0 & X(g_2) \end{matrix}$$

5. Seja qualquer matriz M simétrica, de tamanho *n x n*, é possível construir um grafo tal que M é a sua matriz de adjacência.

Multi-grafo

Para representar um multi-grafo, é só permitir que um elemento da matriz possa ter um valor superior a 1. Nesse caso, o valor do elemento e_{jk} representa o número de arestas entre v_j e v_k.

Peudo-grafo

Não tem dificuldade a representar laços. Simplesmente, um elemento da diagonal principal poderá ter um valor diferente de 0.

Grafo direcionado

Nesse caso, podemos utilizar uma matriz não necessariamente simétrica. Um elemento e_{jk} da matriz terá o valor 1 se existe uma aresta de v_j até v_k. Eis a representação do grafo da figura 3:

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	1	1	0	0
v_2	0	0	0	0	0	1
v_3	0	1	0	0	0	0
v_4	0	1	0	0	0	0
v_5	0	0	0	1	0	0
v_6	0	0	0	0	1	0

Grafo rotulado em arestas

Para representar um grafo rotulado, podemos simplesmente colocar, ao invéz do valor 1 quando existe uma aresta entre dois vértices, o rótulo associado a essa aresta. Note que essa opção entra em conflito com a representação de multi-grafo. Se for o caso, o valor de cada elemento da matriz deverá ser uma lista.

Codificação da matriz

Numa representação com matriz de adjacência, geralmente terá muitas posições que contêm o valor 0. Uma solução para ocupar menos espaço de memória consiste em uma codificação da matriz com um valor único. Eis uma maneira de codificar:

Para qualquer posição localizada acima da diagonal principal, vamos multiplicar o valor dessa posição com um valor 2^i, e somar esses valores. Mais especificamente, sendo A a matriz de ajacência, o código da matriz sera o seguinte valor:

$$a_{12} \times 2^0 + a_{13} \times 2^1 + \ldots + a_{1n} \times 2^{n-1} + a_{23} \times 2^n + \ldots + a_{2n} \times 2^{2n-3} + \ldots + a_{n-1,n} \times 2^{k-1}$$
$$\text{onde } k = n(n-1)/2$$

Mais intuitivamente, isso é um valor binário obtido justapondo, em ordem inverso, as partes das linhas da matriz que contém somente elementos acima da diagonal principal. Consideremos por exemplo a matriz de adjacência do grafo da figura 4:

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	0	1	0
v_2	1	0	1	1	1	0
v_3	0	1	0	0	0	0
v_4	0	1	0	0	1	1
v_5	1	1	0	1	0	0
v_6	0	0	0	1	0	0

Consideremos agora somente os valores acima da diagonal principal:

	v_1	v_2	v_3	v_4	v_5	v_6
v_1		1	0	0	1	0
v_2			1	1	1	0
v_3				0	0	0
v_4					1	1
v_5						0
v_6						

Juntando o que está sobrando nas linhas, obtemos: 100101110000110

Invertindo, obtemos o valor 011000011101001 = 12521

Estrutura de adjacência

O fato de que as linhas da matriz de adjacência contêm normalmente muitos 0 sugere outras implementações mais compactas. Uma delas é é a estrutura de adjacência. Seja G=(V,E) um grafo. A estrutura de adjacência A de G é um conjunto de *n* listas A(v), uma para cada vértice *v* de V. Cada lista A(v) é denominada *lista de adjacências* de *v*, e contém todos os vértices que são adjacentes a *v*.

Eis um exemplo de estrutura de adjacência:

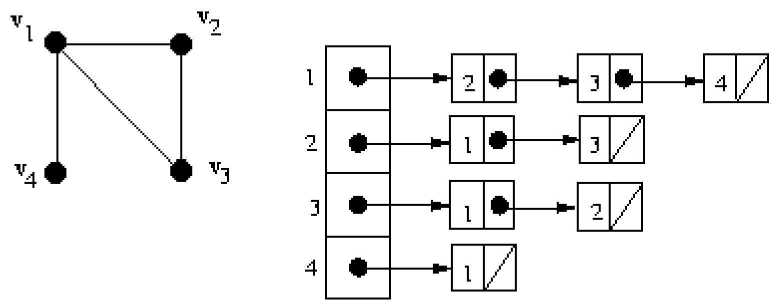

Figura 5

Multi-grafo e Pseudo-grafo

A representação de um multi-grafo ou pseudo-grafo não exige nenhum tratamento especial. A lista de adjacência de um vértice v_i conterá dois elementos idênticos no primeiro caso, e v_i no segundo caso.

Grafo direcionado

Na definição dada acima, a representação de cada aresta exige dois elementos, um em cada lista de adjacência que corresponde a um vértice ligado pela aresta. Em um grafo direcionado, teria somente um elemento por cada aresta. Poderíamos, por exemplo, dar para cada vértice v_i a lista dos vértices ligados por uma aresta que a partir de v_i. Poderíamos também, para cada vértice, utilizar duas listas: uma para as arestas divergentes e outra para as arestas convergentes.

Grafo rotulado em arestas

Nesse caso, simplesmente acrescentamos mais uma informação na estrutura que representa um elemento de lista de adjacência.

Conjuntos

Em certos casos, pode ser vantajoso de representar os grafos de maneira semelhante à definição, isto é, usando dois conjuntos, um para representar os vértices, outro para representar as arestas. Uma maneira de representar um conjunto é o uso de listas. Para representar o conjunto das arestas, os elementos do conjunto serão pares, conforme a definição.

Eis um exemplo:

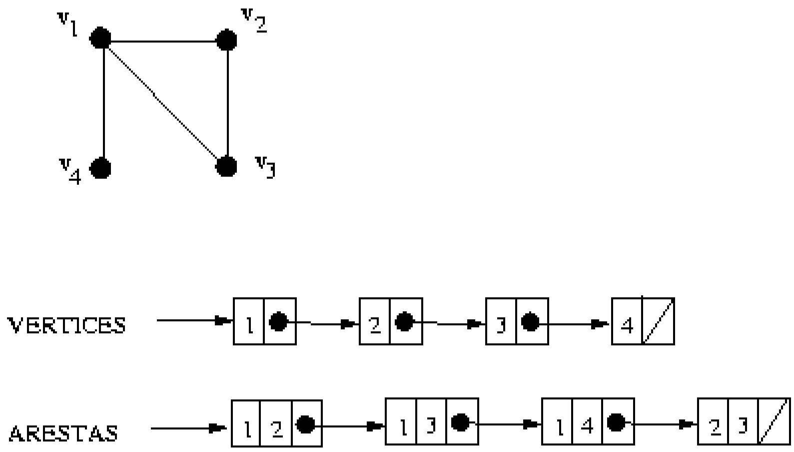

Figura 6

Multi-grafo

Nesse caso, não podemos representar uma aresta especificando somente os dois vértices que ela liga. É preciso também rotular as arestas.

Pseudo-grafo

A representação de um laço não apresenta dificuldade adicional.

Grafo direcionado

Nesse caso, só precisamos tomar cuidado com a ordem dos elementos que constituem os pares do conjunto de arestas.

Grafo rotulado em arestas

Nesse caso, simplesmente acrescentamos mais uma informação na estrutura que representa uma aresta no conjunto de arestas.

Complexidade:

Eis uma tabela que resume a complexidade das implementações descritas acima (Supondo *n* o número de vértices e *m* o número de arestas (é colocado em negrito, para cada caso, a representação que geralmente é a melhor):

	Mat. Inc.	Mat. Adj.	Estr. Adj.	Conjuntos
Memória	O(mn)	$O(n^2)$	**O(m+n)**	**O(m+n)**
Buscar todos os vizinhos de v_i	O(mn)	O(n)	**O(n)**	O(m)
Conferir adjacência de v_i e v_j	O(m)	**O(1)**	O(n)	O(m)
Visitar todas as arestas	O(mn)	$O(n^2)$	**O(m)***	**O(m)**
Calcular grau de um vértice	O(m)	O(n)	**O(n)**	O(m)

*Nesse caso, deve fazer um teste para não considerar duas vezes a mesma arestas. Suponhamos que cada vértice é representado por um valor de 1 até *n*. Na lista de adjacência de um vértice *i*, só consideramos os vértices *j* tal que $j > i$

Os dados dessa tabela mostram a vantagem da estrutura de adjacência em todos os casos com a exceção de um. Para conferir a adjacência de dois vértices, é difícil fazer melhor que a matriz de adjacência.

Além do tipo de processamento que queremos efetuar com o grafo, a escolha da implementação dependerá também das características dos grafos implementados. Vamos agora considerar alguns casos interessantes:

Grafo completo

O grafo completo contém n(n-1)/2 arestas. Portanto, o valor m é proporcional a n^2. A complexidade, nesse caso, é ilustrada pela seguinte tabela:

	Mat. Inc.	Mat. Adj.	Estr. Adj.	Conjuntos
Memória	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
Buscar todos os vizinhos de v_i	$O(n^3)$	$O(n)$	$O(n)$	$O(n^2)$
Conferir adjacência de v_i e v_j	$O(n^2)$	$O(1)$	$O(n)$	$O(n^2)$
Visitar todas as arestas	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n^2)$
Calcular grau de um vértice	$O(n^2)$	$O(n)$	$O(n)$	$O(n^2)$

Podemos ver que a representação por matriz de incidência, que já não se destacava na tabela dada acima, parece pior ainda nesta tabela. Dá para concluir também que quando o número de arestas se aproxima do grafo completo, a estrutura de adjacência não aparece muito melhor que a matriz de adjacência.

Grafo esparso

Suponhamos agora um grafo onde o número de vértices é muito grande, relativamente ao número de arestas. Podemos então considerar desprezível o número de arestas. Em outras palavara, o número *m* poderia ser considerado constante.

A complexidade nesse caso, é semelhante a isso:

	Mat. Inc.	Mat. Adj.	Estr. Adj.	Conjuntos
Memória	$O(n)$	$O(n^2)$	$O(n)$	$O(n)$
Buscar todos os vizinhos de v_i	$O(n)$	$O(n)$	$O(n)$	$O(1)$
Conferir adjacência de v_i e v_j	$O(1)$	$O(1)$	$O(n)$	$O(1)$
Visitar todas as arestas	$O(n)$	$O(n^2)$	$O(n)$	$O(1)$
Calcular grau de um vértice	$O(1)$	$O(n)$	$O(n)$	$O(1)$

Nesse caso, as implementações em matriz de incidência et de conjuntos aparecem melhores.

Comentários

A atualização de lista dinâmica, como é preciso nas implementações de listas de adjacência e de conjuntos, exige mais tempo que a manipulação de uma tabela. Se uma lista é ordenada, a adição de novos elementos vai demorar mais, mas a verificação da existência de um elemento vai ser mais rápida.

A estrutura de adjacência e a representação usando os conjuntos têm a mesma complexidade em memória, mas na verdade no segundo caso menos espaço de memória é ocupado, pois não tem a redundância que ocorre nas listas de adjacência. Entretanto, podemos ver facilmente que quando o número de arestas é grande (o que freqüentemente é o caso), a estrutura de adjacência oferecem um melhor tempo de acesso.

Exercícios

Exercício 3-1: Escreva a matriz de incidência dos grafos ilustrados na figura 7. Escreva a matriz do grafo da figura 7b, usando a submatrizes que correspondem a cada componente do grafo.

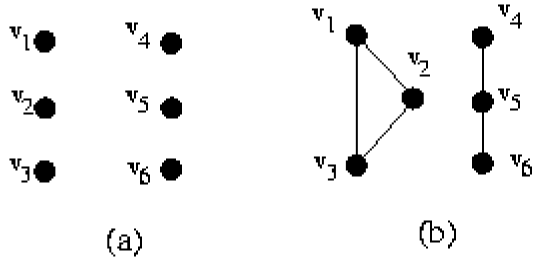

Figura 7

Exercício 3-2: Desenhe o grafo que corresponde à seguinte matriz de adjacência:

0 0 0 1 0 1 1
0 1 0 0 1 1 1
0 0 0 0 0 0 0
1 1 1 0 1 0 0
0 0 1 1 1 0 1
1 1 0 0 1 0 0
1 1 0 0 1 1 1

Exercício 3-3: Seja uma matriz de adjacência que representa um grafo direcionado. Se trocamos duas linhas e as duas colunas correspondentes, obtemos um grafo isomorfo?

Exercício 3-4: Analise a complexidade das implementações com um grafo regular de *k* vértices.

Exercício 3-5: Analise a complexidade das implementações no caso onde queremos determinar se um grafo é um multi-grafo ou um pseudo-grafo.

Exercício 3-6: Escreva um algoritmo para realizar cada uma das seguintes transformações de representações:

1. Matriz de adjacência para matriz de incidência.
2. Matriz de adjacência para estrutura de adjacência.
3. Estrutura dos dois cunjuntos para estrutura de adjacência.
4. Estrutura dos dois cunjuntos para lista de adjacência.