Digrafos - Representação Computacional

Prof. Andrei Braga

UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

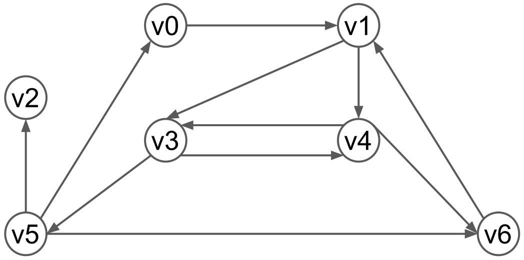
Conteúdo

- Representação computacional de digrafos
- Referências

Grafo dirigido – Digrafo (Revisão)

- Um **grafo dirigido** ou **digrafo** G é um par ordenado (V, E) composto por
  - um conjunto de **vértices** V e
  - um conjunto de **arestas** E, sendo cada aresta um par ordenado (v_i, v_j) de vértices de G
    - note que (v_i, v_j) ≠ (v_j, v_i);
    - denominamos v_i a **cauda** da aresta e v_j a **cabeça** da aresta

- Exemplo:
  - G = (V, E), onde
    - $V = \{ v_0, v_1, v_2, v_3, v_4, v_5, v_6 \}$ e
    - $E = \{ (v_0, v_1), (v_1, v_3), (v_1, v_4),$
      $(v_3, v_4), (v_3, v_5), (v_4, v_3),$
      $(v_4, v_6), (v_5, v_0), (v_5, v_2),$
      $(v_5, v_6), (v_6, v_1) \}$



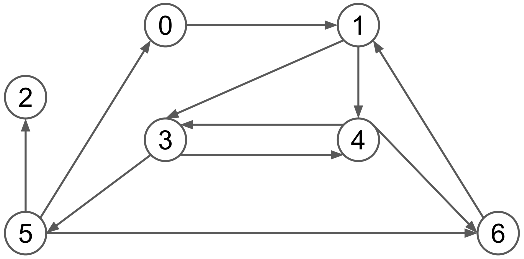
Representação computacional

- Anteriormente, vimos duas formas comuns de representar computacionalmente um grafo não-dirigido: matriz de adjacências e listas de adjacência
- A seguir, veremos formas equivalentes de representar computacionalmente um digrafo

Matriz de adjacências

- A representação de um digrafo *G* como uma **matriz de adjacências** consiste em uma matriz de |*V*(*G*)| linhas, com índices 0, 1, …, |*V*(*G*)| - 1, e de |*V*(*G*)| colunas, com índices 0, 1, …, |*V*(*G*)| - 1, tal que a célula (*i*, *j*) da matriz é igual a
    - 1 se i j é uma aresta de *G*
    - 0 caso contrário

G



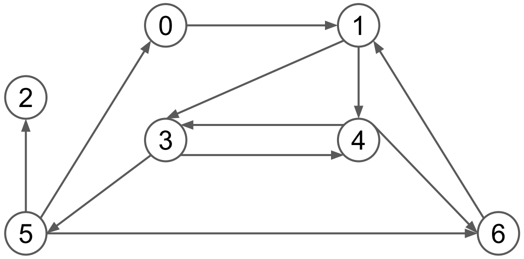
Matriz de adjacências de *G*

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

Matriz de adjacências

- Observações:
  - Não é possível representar arestas paralelas
  - Para digrafos simples, todas as células da diagonal principal da matriz são iguais a 0
  - Uma aresta *i j* é representada por **apenas uma** célula da matriz: (*i*, *j*) – a célula (*j*, *i*) representa uma aresta diferente, a aresta *j i*

G



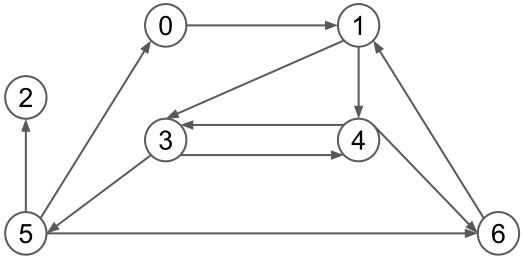
Matriz de adjacências de *G*

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

Matriz de adjacências

- Observações:
    - Em geral, a matriz não é simétrica em relação à diagonal principal

G



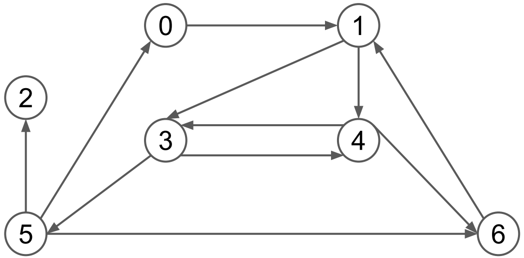
Matriz de adjacências de *G*

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

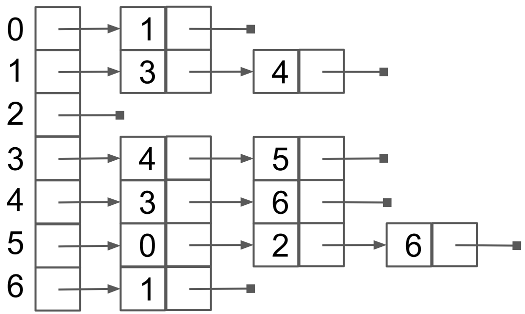
Listas de adjacência

- A representação de um digrafo G como **listas de adjacência** consiste em um vetor de $|V(G)|$ elementos, com índices 0, 1, …, $|V(G)|$ - 1, tal que o elemento i do vetor armazena uma lista com os vizinhos de saída do vértice i em G

G



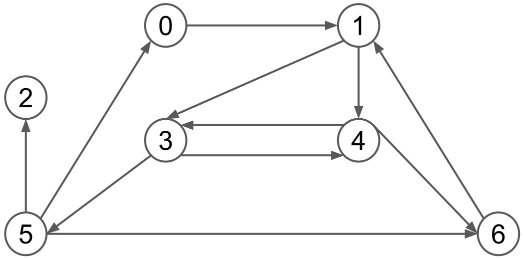
Listas de adjacência de *G*



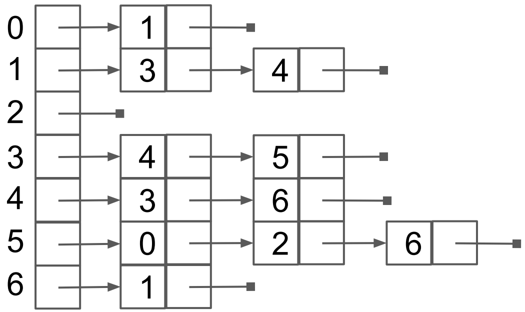
Listas de adjacência

- Observações:
  - Uma aresta *i j* é representada em **apenas uma** lista de adjacência: o vértice *j* está na lista do vértice *i*

G



Listas de adjacência de *G*



Matriz de adjacências vs. listas de adjacência

- Dado um digrafo $G = (V, E)$, a quantidade de memória utilizada para representar G
  - como uma matriz de adjacências é proporcional a $|V|^2$ e
  - como listas de adjacência é proporcional a $|V| + |E|$
- Se G é um digrafo **esparso**, isto é, $|E|$ é bem menor que $|V|^2$, então é usualmente mais interessante representar G como listas de adjacência
- Se G é um digrafo **denso**, isto é, $|E|$ é um número próximo a $|V|^2$, então é usualmente mais interessante representar G como uma matriz de adjacências

Matriz de adjacências vs. listas de adjacência

- Dado um digrafo $G = (V, E)$:

	Matriz de Adjacências	Listas de Adjacência
Memória utilizada	$\lvert V\rvert^2$	$\lvert V\rvert + \lvert E\rvert$
Tempo para inserir aresta	constante	constante
Tempo para verificar aresta	constante	pior caso: $\lvert V\rvert$
Tempo para remover aresta	constante	pior caso: $\lvert V\rvert$

Valores proporcionais a

Referências

- Esta apresentação é baseada nos seguintes materiais:
  - Capítulo 22 do livro
    Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. Introduction to Algorithms. 3rd. ed. MIT Press, 2009.
  - Capítulo 19 do livro
    Sedgewick, R. Algorithms in C++ – Part 5. Graph Algorithms. 3rd. ed. Addison-Wesley, 2002.