Representações Computacionais
5189-32

Rodrigo Calvo
rcalvo@uem.br

Departamento de Informática – DIN
Universidade Estadual de Maringá – UEM

1° semestre de 2016

Introdução

- Geralmente medimos o tamanho de um grafo $G = (V, A)$ em termos do número de vértices $|V|$ e do número de arestas $|A|$
  - Dentro da notação assintótica, o termo V representará $|V|$, e o termo A, representará $|A|$

- Um grafo $G = (V, A)$ é:
  - **Esparso** se $|A|$ é muito menor que $|V|^2$
  - **Denso** se $|A|$ está próximo de $|V|^2$

Representação de grafos

- Existe duas maneiras de representar um grafo $G = (V, A)$ em termos do número de vértices $|V|$ e do número de arestas $|A|$
  - Matriz de adjacências
  - Lista de adjacências

Matriz de Adjacências

- A **matriz de adjacência** de um grafo $G = (V, A)$ contendo $|V|$ vértices é uma matriz $|V| \times |V|$ de bits, onde $A[i, j]$ é 1 (ou verdadeiro) se e somente se existe um arco do vértice i para o vértice j.

- Para grafos ponderados, $A[i, j]$ contém o rótulo ou peso associado com a aresta e, neste caso, a matriz não é de bits.

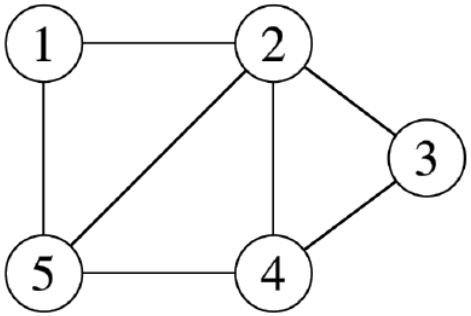
- Se não existir uma aresta de i para j então é necessário utilizar um valor que não possa ser usado como rótulo ou peso.

Matriz de Adjacências

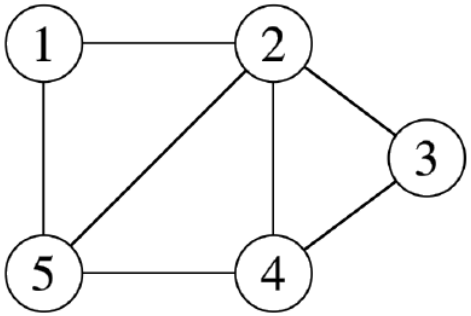
- Supondo que os vértices são numerados 1, 2, ..., $|V|$, a representação consiste em uma matriz $|V| \times |V| = (a_{ij})$

$$a_{ij} = \begin{cases} 1, & \text{se } (i,j) \in A \\ 0, & \text{caso contrário} \end{cases}$$

Matriz de Adjacências

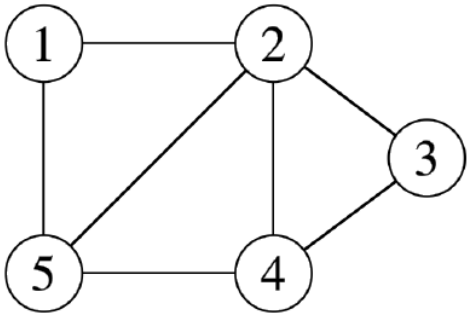


Matriz de Adjacências

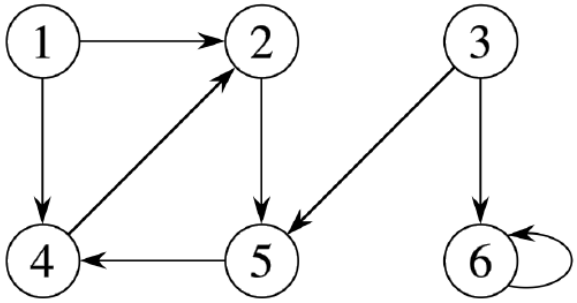


	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

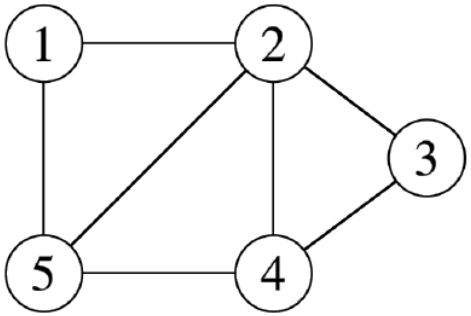
Matriz de Adjacências



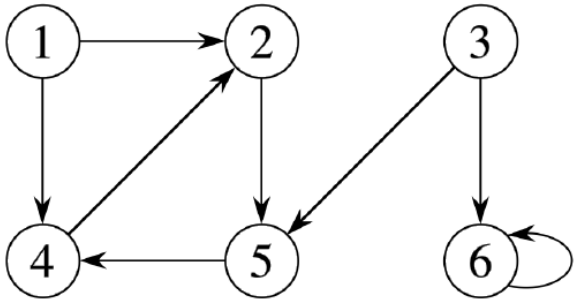
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0



Matriz de Adjacências



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



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

Matriz de Adjacências

- Qual e quantidade de memória requerida?

Matriz de Adjacências

- Qual e quantidade de memória requerida? $|V|^2$
    - A quantidade de memória independe de A

Matriz de Adjacências

- Qual e quantidade de memória requerida? $|V|^2$
  - A quantidade de memória independe de A
- Em um grafo não direcionado, a matriz é igual a sua transposta, desta forma é possível usar apenas os elementos abaixo (ou acima) da diagonal principal

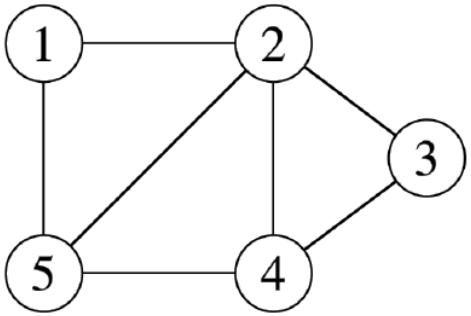
Matriz de Adjacências

- Qual e quantidade de memória requerida? $|V|^2$
  - A quantidade de memória independe de A
- Em um grafo não direcionado, a matriz é igual a sua transposta, desta forma e possível usar apenas os elementos abaixo (ou acima) da diagonal principal
- Vantagens
  - Simplicidade
  - Permite consultar se uma aresta faz parte do grafo em tempo constante
- Desvantagens
  - Memória: a matriz necessita de $\Omega(|V|^2)$ de espaço
  - Ler ou examinar a matriz tem complexidade de tempo $O(|V|^2)$
- Adequada para **grafos densos** → |A| é próximo de $|V|^2$

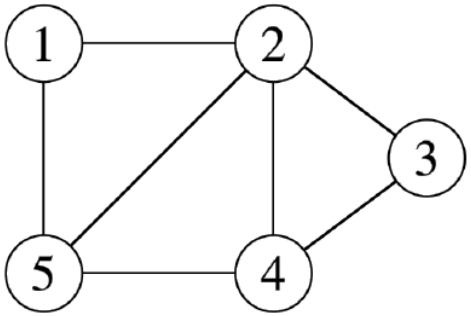
Lista de Adjacências

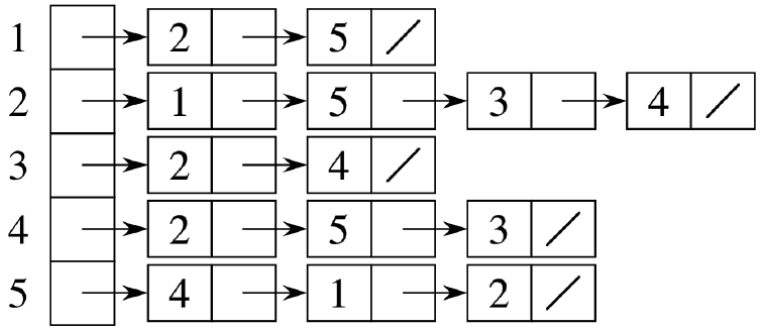
- A representação de **lista de adjacências** consiste de um arranjo *Adj* de $|V|$ listas, uma para cada vértice
- Para cada $u \in V$, a lista de adjacências *Adj[u]* contém (ponteiros para) todos os vértices *v* tal que $(u, v) \in E$
- Atributos para algoritmos em pseudo-código
  - G.V - conjunto de vértices
  - G.E - conjunto de arestas
  - G.Adj - arranjo com as listas de adjacências

Lista de Adjacências

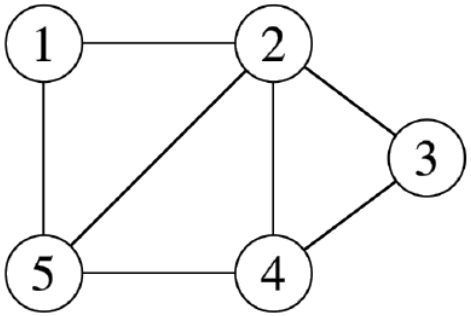


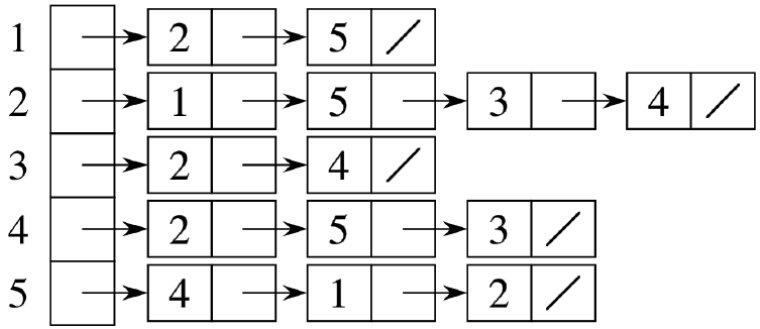
Lista de Adjacências

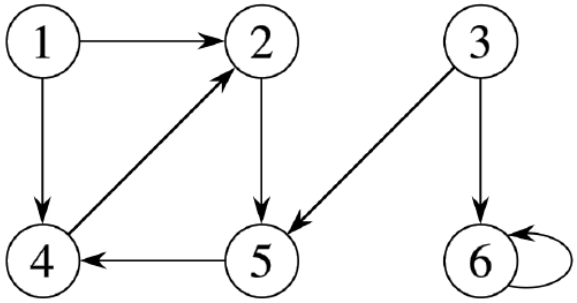




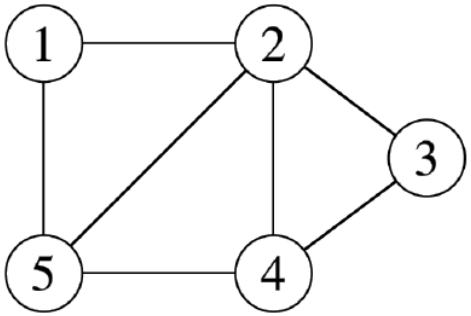
Lista de Adjacências

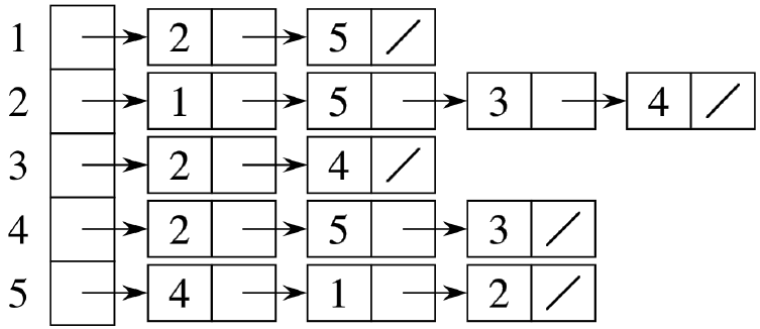


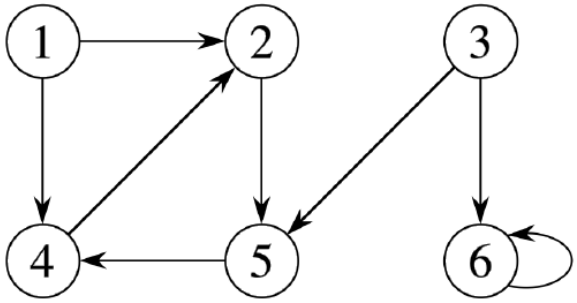


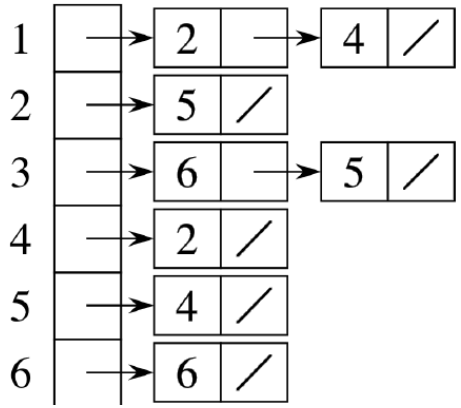


Lista de Adjacências









Lista de Adjacências

- Qual é a soma dos comprimentos de todas as listas de adjacências?

Lista de Adjacências

- Qual é a soma dos comprimentos de todas as listas de adjacências?
  - Se G e um grafo direcionado, a soma é $|A|$

Lista de Adjacências

- Qual é a soma dos comprimentos de todas as listas de adjacências?
  - Se G e um grafo direcionado, a soma é $|A|$
  - Se G e um grafo não direcionado, a soma é $2|A|$

Lista de Adjacências

- Qual é a soma dos comprimentos de todas as listas de adjacências?
  - Se G e um grafo direcionado, a soma é $|A|$
  - Se G e um grafo não direcionado, a soma é $2|A|$
- Qual é a quantidade de memória requerida?

Lista de Adjacências

- Qual é a soma dos comprimentos de todas as listas de adjacências?
  - Se G e um grafo direcionado, a soma é $|A|$
  - Se G e um grafo não direcionado, a soma é $2|A|$
- Qual é a quantidade de memória requerida? $\Theta(V + A)$

Lista de Adjacências

- Qual é a soma dos comprimentos de todas as listas de adjacências?
  - Se G e um grafo direcionado, a soma é $|A|$
  - Se G e um grafo não direcionado, a soma é $2|A|$
- Qual é a quantidade de memória requerida? $\Theta(V + A)$

- Vantagens
  - Flexível, é possível adaptar a representação para variantes de grafos;
  - A quantidade de memória e assintoticamente ótima.
- Desvantagem
  - Não existe nenhum modo rápido para determinar se uma dada aresta (u, v) esta presente no grafo.

- Adequada para **grafos esparsos** → $|A|$ é muito menor que $|V|^2$

Atributos

- Muitos algoritmos que operam em grafos precisam manter atributos para vértices e/ou arestas
- Em alguns exemplos de código, indicaremos os atributos como
  - *v.d*, atributo *d* do vértice *v*
  - *(u, v).f* , atributo *f* da aresta (*u, v*)

- Como estes atributos podem ser implementados?
  - Depende da linguagem de programação, algoritmo, etc
  - Os atributos podem ser armazenados diretamente na lista ou matriz de adjacências
  - Se os vértices são enumerados de 1, ..., $|V|$ os atributos podem ser representados em arranjos, tais como d[1, ..., $|V|$]
  - Atributos de vértices podem ficar nos registros que representam os vértices
  - Atributos de arestas podem ficar nos registros que representam as arestas

Exercício

[22.1-1] Dada uma representação de lista de adjacências de um grafo orientado, qual o tempo necessário para computar o grau de saída de todo o vértice?

Exercício

[22.1-1] Dada uma representação de lista de adjacências de um grafo orientado, qual o tempo necessário para computar o grau de saída de todo o vértice?

- Antes de fazer a análise do tempo de execução e necessário, escrever o pseudo-código do algoritmo

Exercício

```
computa-graus-de-saida(G)
1 for u in G.V
2    u.grau-de-saida = 0
3 for u in G.V
4    for v in G.Adj[u]
5       u.grau-de-saida += 1
```

Exercício

```
computa-graus-de-saida(G)
1 for u in G.V
2   u.grau-de-saida = 0
3 for u in G.V
4   for v in G.Adj[u]
5     u.grau-de-saida += 1
```

Análise do tempo de execução

- Os laços das linhas 1 e 3 demoram $\Theta(V)$
- O laço da linha 4 (sem contar a linha 5) demora $\Theta(V)$
- A cada interação do laço da linha 4, a linha 5 é executada $|G.Adj[u]|$ vezes, como o laço da linha 4 é executado uma vez para cada vértice, temos que a linha 5 é executada:

$$\sum_{u \in G.V} |G.Adj[u]| = |A|$$

- Ou seja, o tempo de execução da linha 5 é $\Theta(A)$
- Portanto, o tempo de execução do procedimento computa-graus-saida é $\Theta(V + A)$

Exercício

```
computa-graus-de-entrada(G)
1 for u in G.V
2    u.grau-de-entrada = 0
3 for u in G.V
4    for v in G.Adj[u]
5       v.grau-de-entrada += 1
```

Análise do tempo de execução

- Mesmo do procedimento **computa-graus-saida**

Bibliografia

- Thomas H. Cormen et al. Introduction to Algorithms. 3rd edition. Capítulo 22.1.

- Nivio Ziviani. Projeto de Algoritmos com Implementações em Pascal e C. 3a Edição Revista e Ampliada, Cengage Learning, 2010. Capítulo 7. Seção 7.2.1