Representações computacionais

Marco A L Barbosa

malbarbo.pro.br

Departamento de Informática
Universidade Estadual de Maringá

Introdução

- Geralmente medimos o tamanho de um grafo $G = (V, E)$ em termos do número de vértices $|V|$ e do número de arestas $|E|$
  - Dentro da notação assintótica, e apenas neste caso, o termo V representará $|V|$, e o termo E, representará $|E|$
  - Por exemplo, quando dizermos que o tempo de execução de um algoritmo é $O(VE)$, significa que o tempo de execução é $O(VE)$

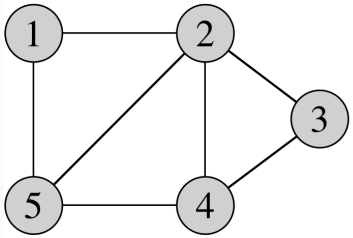
Introdução

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

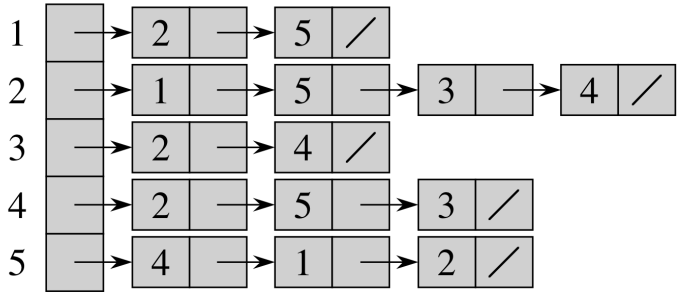
Representação de grafos

- Existem duas maneiras comuns para representar um grafo $G = (V, E)$
  - Lista de adjacências
  - Matriz de adjacências

Exemplos



(a)

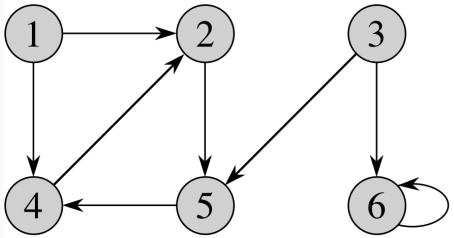


(b)

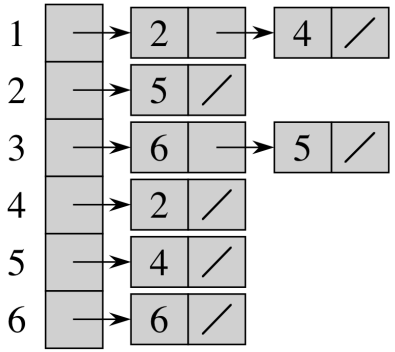
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

22-1



(a)



(b)

	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

(c)

22-2

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 todos os vértices (ou referências para os vértices) v tal que $(u, v) \in E$
- No pseudo do código vamos tratar o arranjo Adj como um atributo do grafo, assim como V e E
  - $G.V$ - conjunto de vértices
  - $G.E$ - conjunto de arestas
  - $G.Adj$ - arranjo com as listas de adjacências

Lista de adjacências

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

Lista de adjacências

- Vantagens
  - Flexível, é possível estender a representação para multigrafos
  - A quantidade de memória é assintoticamente ótima (adequada para grafos esparsos)
- Desvantagem
  - Não existe nenhum modo rápido para determinar se uma dada aresta (u, v) está presente no grafo

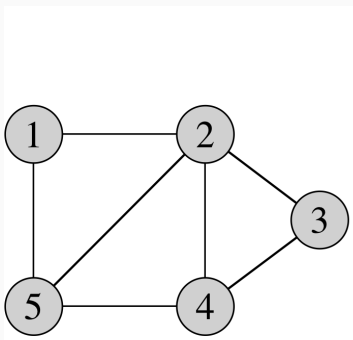
Matriz de adjacências

- Na **representação de matriz de adjacências**, usamos uma matriz $A = (a_{ij})$, de tamanho $|V| \times |V|$, tal que

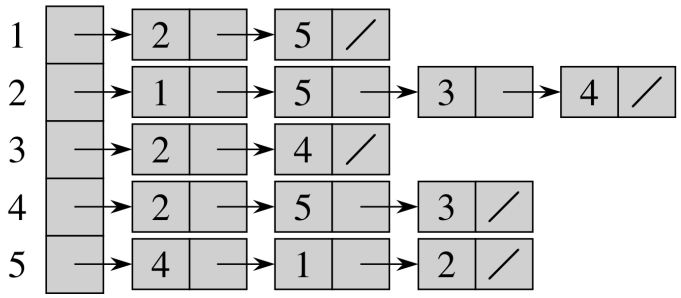
$$a_{ij} = \begin{cases} 1 \text{ se } (i, j) \in E \\ 0 \text{ caso contrário} \end{cases}$$

- Neste caso, supomos que os vértices são numerados $1, 2, \ldots, |V|$

Matriz de adjacências



(a)

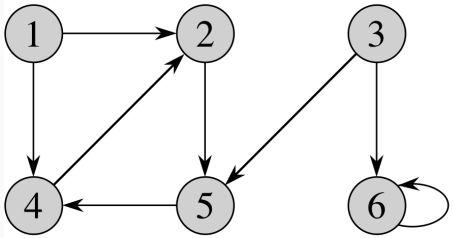


(b)

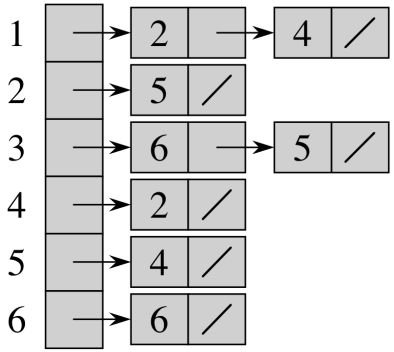
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

22-1



(a)



(b)

	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

(c)

22-2

Matriz de adjacências

- Qual é quantidade de memória requerida?
  $\Theta(V^2)$ (independe de $|E|$)
- Adequada para grafos densos
- Em um grafo não orientado, a matriz é igual a sua transposta, desta forma é possível usar apenas os elementos abaixo (ou acima) da diagonal principal

Matriz de adjacências

- Vantagens
  - Simplicidade
  - Permite consultar se uma aresta faz parte do grafo em tempo constante
- Desvantagem
  - Uso excessivo da memória

Atributos

- Muitos algoritmos que operam em grafos precisam manter atributos para vértices e/ou arestas
- Nos códigos, indicamos os atributos como
  - $v.d$, atributo d do vértice v
  - $(u, v).f$, atributo f da aresta (u, v)

Atributos

- Como estes atributos podem ser implementados?
  - Depende da linguagem de programação, algoritmo, etc
  - Os atributos da arestas podem ser armazenados diretamente na lista ou matriz de adjacência
  - 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

Exemplos de implementação

Veja na página da disciplina.

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? Qual o tempo necessário para computar os graus de entrada?

Resolução 22.1-1

Antes de fazer a análise do tempo de execução é necessário escrever o pseudo código do algoritmo

Resolução 22.1-1

GRAUS-DE-SAIDA(G)

```
1  for v ∈ G.V
2      v.grau-de-saida = 0
3  for u ∈ G.V
4      for v ∈ G.Adj[u]
5          u.grau-de-saida = u.grau-de-saida + 1
```

Análise do tempo de execução

- O laço das linhas 1 a 2 tem tempo de execução $\Theta(V)$
- A cada interação do laço da linha 3, o laço das linha 4 a 5 é executado $|G.Adj[u]|$ vezes, como o laço das linha 4 a 5 é executado uma vez para cada vértice, temos que seu tempo de execução é $\sum_{u \in G.V} |G.Adj[u]| = |E|$. Ou seja, o tempo de execução das linhas 3 a 5 é $\Theta(V + E)$
- Portanto, o tempo de execução do procedimento GRAUS-DE-SAIDA é $\Theta(V + E)$

Resolução 22.1-1

GRAUS-DE-ENTRADA(G)

```
1  for u ∈ G.V
2      u.grau-de-entrada = 0
3  for u ∈ G.V
4      for v ∈ G.Adj[u]
5          v.grau-de-entrada = v.grau-de-entrada + 1
```

Análise do tempo de execução

- Mesmo do procedimento GRAUS-DE-SAIDA

Referências

- Thomas H. Cormen et al. Introduction to Algorithms. 3rd edition. Capítulo 22.1.
- `networkx`, um exemplo de uma biblioteca de grafos em Python