MO417 — Complexidade de Algoritmos I

Cid Carvalho de Souza Cândida Nunes da Silva
Orlando Lee

1 de outubro de 2008

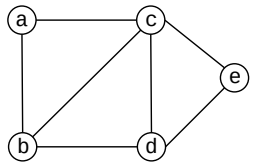
Grafos: Noções Básicas e Representação

Cid Carvalho de Souza, Cândida Nunes da Silva, Orlando Lee MO417 — Complexidade de Algoritmos – v. 2.1

Definição de Grafo

Um *grafo* é um par $G = (V, E)$ onde

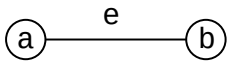
- V é um conjunto finito de elementos chamados *vértices* e
- E é um conjunto finito de pares não-ordenados de vértices chamados *arestas*.
- **Exemplo:**
  $V = \{a, b, c, d, e\}$
  $E = \{(a, b), (a, c), (b, c), (b, d), (c, d), (c, e), (d, e)\}$



Cid Carvalho de Souza, Cândida Nunes da Silva, Orlando Lee MO417 — Complexidade de Algoritmos – v. 2.1

Definição de Grafo

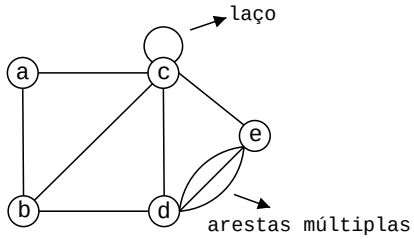
- Dada uma aresta $e = (a, b)$, dizemos que os vértices a e b são os *extremos* da aresta e e que a e b são vértices *adjacentes*.
- Dizemos também que a aresta e é *incidente* aos vértices a e b, e que os vértices a e b são incidentes à aresta e.



Cid Carvalho de Souza, Cândida Nunes da Silva, Orlando Lee MO417 — Complexidade de Algoritmos – v. 2.1

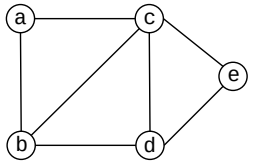
Grafo Simples

- Dizemos que um grafo é *simples* quando não possui laços ou arestas múltiplas.
- Um *laço* é uma aresta com extremos idêntico e *arestas múltiplas* são duas ou mais arestas com o mesmo par de vértices como extremos.
- **Exemplo:**



Tamanho do Grafo

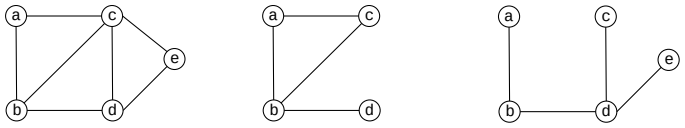
- Denotamos por $|V|$ e $|E|$ a cardinalidade dos conjuntos de vértices e arestas de um grafo G, respectivamente.
- No exemplo abaixo temos $|V| = 5$ e $|E| = 7$.



O *tamanho* do grafo G é dado por $|V| + |E|$.

Subgrafo e Subgrafo Gerador

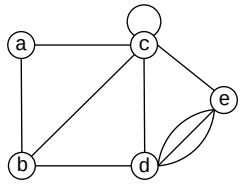
- Um *subgrafo* $H = (V', E')$ de um grafo $G = (V, E)$ é um grafo tal que $V' \subseteq V$, $E' \subseteq E$.
- Um *subgrafo gerador* de G é um subgrafo H com $V' = V$.
- **Exemplo:**



Grafo G | Subgrafo não gerador | Subgrafo gerador

Grau de um vértice

- O *grau* (*degree*) de um vértice v, denotado por $d(v)$ é o número de arestas incidentes a v, com laços contados duas vezes.
- **Exemplo:**




```
d(a)=2
d(b)=3
d(c)=6
d(d)=5
d(e)=4
```

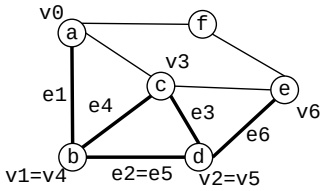
Teorema (*Handshaking lemma*)

Para todo grafo $G = (V, E)$ temos:

$$\sum_{v \in V} d(v) = 2|E|.$$

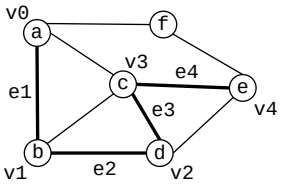
Caminhos em Grafos

- Um *caminho* P de v_0 a v_n no grafo G é uma seqüência finita e não vazia $(v_0, e_1, v_1, \ldots, e_n, v_n)$ cujos elementos são alternadamente vértices e arestas e tal que, para todo $1 \leq i \leq n$, v_{i-1} e v_i são os extremos de e_i.
- O *comprimento* do caminho P é dado pelo seu número de arestas, ou seja, n.
- **Exemplo:**

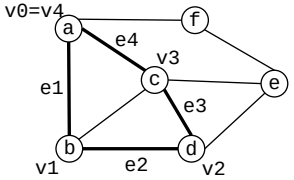


Caminhos Simples e Ciclos

- Um *caminho simples* é um caminho em que não há repetição de vértices e nem de arestas na sequência.
- Um *ciclo* ou *caminho fechado* é uma caminho em que $v_0 = v_n$.
- **Exemplo:**



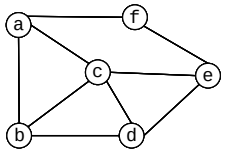
Caminho Simples



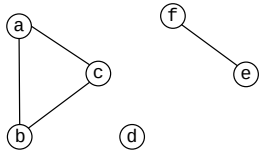
Ciclo

Grafo Conexo

- Dizemos que um grafo é *conexo* se, para qualquer par de vértices u e v de G, existe um caminho de u a v em G.
- Quando o grafo G não é conexo, podemos particionar em *componentes conexos*. Dois vértices u e v de G estão no mesmo componente conexo de G se há caminho de u a v em G.
- **Exemplo:**



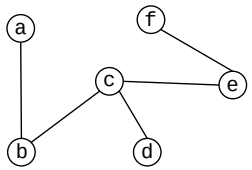
Conexo



Não-conexo com
3 componentes conexos

Árvore

- Um grafo G é uma *árvore* se é conexo e não possui ciclos (acíclico). As seguintes afirmações são equivalentes:
  - G é uma árvore.
  - G é conexo e possui exatamente $|V| - 1$ arestas.
  - G é conexo e a remoção de qualquer aresta desconecta o grafo (minimal conexo).
  - Para todo par de vértices u, v de G, existe um único caminho de u a v em G.
- **Exemplo:**

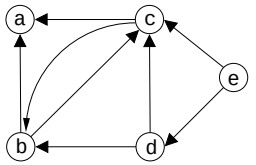


Alguns exemplos de grafos

- Floresta: grafo acíclico (não precisa ser conexo). Cada componente é uma árvore!
- Grafo completo: para todo par de vértices u, v a aresta (u, v) pertence ao grafo.
- Grafo bipartido: possui uma bipartição (A, B) do conjunto de vértices tal que toda aresta tem um extremo em A e outro em B.
- Grafo planar: pode ser desenhado no plano de modo que arestas se interceptam apenas nos extremos.

Grafo Orientado

- As definições que vimos até agora são para grafos *não orientados*.
- Um *grafo orientado* é definido de forma semelhante, com a diferença que as arestas (às vezes chamadas de *arcos*) consistem de pares ordenados de vértices.
- **Exemplo:**



- Às vezes, para enfatizar, dizemos grafo não-orientado em vez de simplesmente grafo.

Grafo orientado

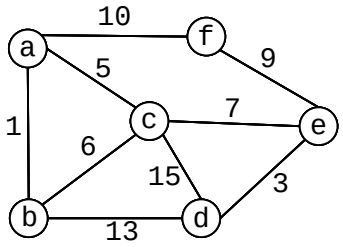
- Se $e = (u, v)$ é uma aresta de um grafo orientado G, então dizemos que e sai e entra em v.
- O *grau de saída* $d^+(v)$ de um vértice v é o número de arestas que saem de v. O *grau de entrada* $d^-(v)$ de v é o número de arestas que entram em v.
  Teorema. *Para todo grafo orientado $G = (V, E)$ temos:*

$$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^+(v) = |E|.$$

- Em geral considera-se que em caminhos e ciclos em grafos orientados são todas as arestas "vão na mesma direção".
- Há um conceito de conexidade para grafos orientados que veremos mais tarde.

Grafo Ponderado

- Um grafo (orientado ou não) é *ponderado* se a cada aresta e do grafo está associado um valor real $c(e)$, o qual denominamos *custo (ou peso)* da aresta.
- **Exemplo:**



Algoritmos em Grafos - Motivação

- Grafos são estruturas abstratas que podem modelar diversos problemas do mundo real.
- Por exemplo, um grafo pode representar conexões entre cidades por estradas ou uma rede de computadores.
- O interesse em estudar algoritmos para problemas em grafos é que conhecer um algoritmo para um determinado problema em grafos pode significar conhecer algoritmos para diversos problemas reais.

Aplicações

- Caminho mínimo: dado um conjunto de cidades, as distâncias entre elas e duas cidades A e B, determinar um caminho (trajeto) mais curto de A até B.
- Árvore Geradora de Peso Mínimo: dado um conjunto de computadores, onde cada par de computadores pode ser ligado usando uma quantidade de fibra ótica, encontrar uma rede interconectando-os que use a menor quantidade de fibra ótica possível.
- Emparelhamento máximo: dado um conjunto de pessoas e um conjunto de vagas para diferentes empregos, onde cada pessoa é qualificada para certos empregos e cada vaga pode ser ocupada por uma pessoa, encontrar um modo de empregar o maior número possível de pessoas.

Aplicações

- Problema do Caixeiro Viajante: dado um conjunto de cidades, encontrar um passeio que sai de uma cidade, passa por todas as cidades e volta para a cidade inicial tal que a distância total a ser percorrida seja menor possível.
- Problema Chinês do Correio: dado o conjunto das ruas de um bairro, encontrar um passeio que passa por todas as ruas voltando ao ponto inicial tal que a distância total a ser percorrida seja menor possível.

Representação Interna de Grafos

- A complexidade dos algoritmos para solução de problemas modelados por grafos depende fortemente da sua representação interna.
- Existem duas representações canônicas: matriz de adjacência e listas de adjacência.
- O uso de uma ou outra num determinado algoritmo depende da natureza das operações que ditam a complexidade do algoritmo.

Matriz de adjacência

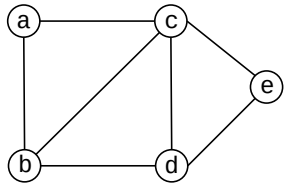
- Seja $G = (V, E)$ um grafo simples (orientado ou não).
- A *matriz de adjacência* de G é uma matriz quadrada A de ordem $|V|$, cujas linhas e colunas são indexadas pelos vértices em V, e tal que:

$$A[i,j] = \begin{cases} 1 & \text{se } (i,j) \in E, \\ 0 & \text{caso contrário.} \end{cases}$$

- Note que se G é não-orientado, então a matriz A correspondente é simétrica.

Matriz de adjacência

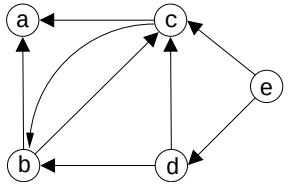
- Exemplo de um grafo e a matriz de adjacência correspondente.



	a	b	c	d	e
a	0	1	1	0	0
b	1	0	1	1	0
c	1	1	0	1	1
d	0	1	1	0	1
e	0	0	1	1	0

Matriz de adjacência

- Exemplo de um grafo orientado e a matriz de adjacência correspondente.



	a	b	c	d	e
a	0	0	0	0	0
b	1	0	1	0	0
c	1	1	0	0	0
d	0	1	1	0	0
e	0	0	1	1	0

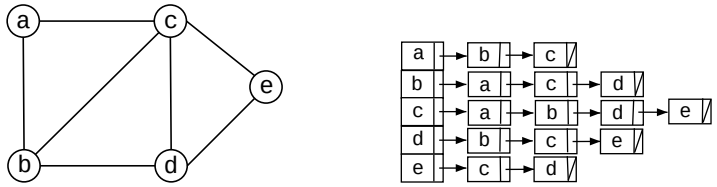
Listas de adjacência

- Seja $G = (V, E)$ um grafo simples (orientado ou não).
- A representação de G por uma *lista de adjacências* consiste no seguinte.

  Para cada vértice v, temos uma lista ligada $\mathrm{Adj}[v]$ dos vértices adjacentes a v, ou seja, w aparece em $\mathrm{Adj}[v]$ se (v, w) é uma aresta de G. Os vértices podem estar em qualquer ordem em uma lista.

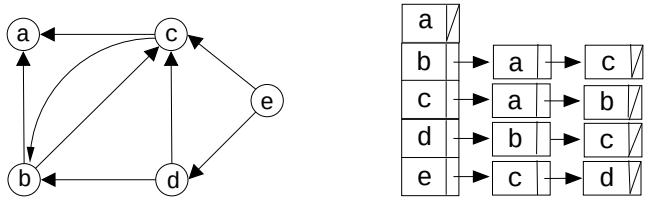
Listas de adjacência

- Exemplo de um grafo não-orientado e a listas de adjacência correspondente.



Lista de adjacências

Exemplo de um grafo orientado e a lista de adjacências correspondente.



Matriz $\times$ Lista de adjacência

- Matriz de adjacência: é fácil verificar se (i, j) é uma aresta de G.
- Lista de adjacência: é fácil descobrir os vértices adjacentes a um dado vértice v (ou seja, listar $\mathrm{Adj}[v]$).
- Matriz de adjacência: espaço $\Theta(|V|^2)$.
  Adequada a grafos densos ($|E| = \Theta(|V|^2)$).
- Lista de adjacência: espaço $\Theta(|V| + |E|)$.
  Adequada a grafos esparsos ($|E| = \Theta(|V|)$).

Extensões

- Há outras alternativas para representar grafos, mas matrizes e listas de adjacência são as mais usadas.
- Elas podem ser adaptadas para representar grafos ponderados, grafos com laços e arestas múltiplas, grafos com pesos nos vértices etc.
- Para determinados problemas é essencial ter estruturas de dados adicionais para melhorar a eficiência dos algoritmos.