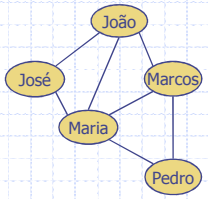
Algoritmos e Estruturas de Dados II - SCE-183

Grafos: Definições

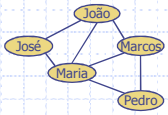
Gustavo Batista

Definição

- Um **grafo** pode ser definido como um par (V, A), onde:
  - V : conjunto de nós chamados vértices (ou nós).
  - A : conjunto de pares de vértices chamados arestas (ou arcos).
- Exemplo:
  - Considere um exemplo de um grafo onde cada vértice é uma pessoa e existe uma aresta entre duas pessoas se e somente se essas pessoas são amigas.

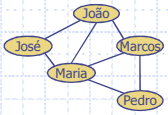


Grafo sobre Amizade



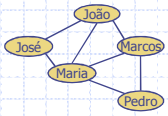
- Se eu sou seu amigo, isso significa que você é meu amigo?
  - Um grafo é **não-direcionado** se a aresta (x,y) sempre implica em (y,x). Caso contrário o grafo é **direcionado** (**dígrafo**).
  - Um grafo sobre "ouviu falar" é tipicamente direcionado.
- Eu sou amigo de mim mesmo?
  - Uma aresta (x,x) é dita ser um **laço** ou **self-loop**.
  - Se x é amigo de y diversas vezes, então essa relação pode ser modelada com **arestas múltiplas** ou **paralelas**.
  - Um grafo é dito ser **simples** se ele não possui laços nem arestas múltiplas.

Grafo sobre Amizade



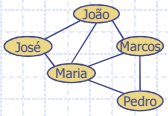
- Quão próximo você é meu amigo?
  - Um grafo é dito ser **ponderado** se as arestas possuem um valor numérico associado.
  - Um grafo é dito ser **não ponderado** se todas as arestas possuem um mesmo peso.
- Eu estou ligado a uma celebridade por uma cadeia de amigos?
  - Um **caminho** é uma seqüência de arestas que conectam dois vértices.

Grafo sobre Amizade



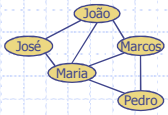
- Quão próxima é a minha ligação com essa celebridade?
  - Podem haver diversos caminhos que ligam dois nós.
  - O **caminho mais curto** é aquele com menor soma de pesos das arestas (ponderado) ou com menor número de arestas (não ponderado).
- Existe um caminho de amigos entre duas pessoas no mundo?
  - Teoria da separação por até "seis graus".
  - Um grafo é **conexo** ou **conectado** se existe um caminho entre quaisquer dois nós.
  - Um grafo direcionado é **fortemente conexo** (conectado) se sempre existir um caminho direcionado entre dois nós.
  - Se um grafo não é conexo, então cada parte conectada é chamada de **componente conexo**.

Grafo sobre Amizade



- Quem possui mais (ou menos) amigos?
  - O **grau** de um vértice é o número de vértices adjacentes a ele.
  - A pessoa mais popular tem o vértice de maior grau do grafo.
  - Ermitões remotos são vértices de grau zero.
  - Em **grafos densos**, a maior parte dos vértices de graus altos, o oposto de **grafos esparsos**.
  - Em **grafos regulares**, cada vértice tem o mesmo grau.
- Qual é o maior clique?
  - Um clique social é um conjunto de amigos que todos se conhecem.
  - Em grafos, um **clique** é um sub-grafo completo, com arestas entre todos os pares de vértices.
  - Cliques são os sub-grafos mais densos possíveis.

Grafo sobre Amizade



- Quanto tempo demora para que eu ouça uma fofoca que contei?
  - Um **ciclo** é um caminho no qual o último vértice é adjacente ao primeiro.
  - Um ciclo em que nenhum vértice se repete é dito ser **simples**.
  - Um ciclo que passa por todos os vértices de um grafo é dito ser **Hamiltoniano**.
  - Um grafo não direcionado acíclico é dito ser uma **árvore** (se conexo) ou uma **floresta** (caso contrário).
  - Um grafo direcionado acíclico é um **dígrafo** ou **DAG** (directed acyclic graph).

Motivação

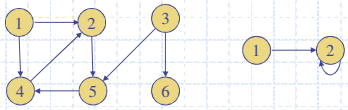
- Muitas aplicações em computação necessitam considerar conjunto de conexões entre pares de objetos:
  - Existe um caminho para ir de um objeto a outro seguindo as conexões?
  - Qual é a menor distância entre dois objetos?
  - Quantos outros objetos podem ser alcançados a partir de um determinado objeto?

Aplicações

- Alguns exemplos de problemas práticos que podem ser resolvidos através de uma modelagem em grafos:
  - Ajudar máquinas de busca a localizar informação relevante na Web.
  - Descobrir qual é o roteiro mais curto para visitar as principais cidades de uma região turística.

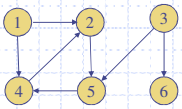
Grafos Direcionados (Dígrafos)

- Um **grafo direcionado** *G* é um par *(V,A)*, onde *V* é um conjunto finito de vértices e *A* é uma relação binária em *V*.
  - Uma aresta *(u,v)* sai do vértice *u* e entra no vértice *v*. O vértice *v* é **adjacente** ao vértice *u*.
  - Podem existir arestas de um vértice para ele mesmo, chamadas de ***self-loops***.

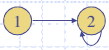


Grafos Direcionados (Dígrafos)

- *G = (V,A)*
  - *V = {1,2,3,4,5,6}* e
  - *A = {(1,2),(1,4),(2,5),(4,2), (5,4),(3,5),(3,6)}*

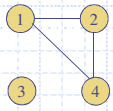


- *G = (V,A)*
  - *V = (1,2)* e
  - *A = {(1,2),(2,2)}*



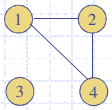
Grafos Não-Direcionados

- Um **grafo não direcionado** *G* é um par *(V,A)*, onde o conjunto de arestas A é constituído de pares de vértices não ordenados.
  - As arestas *(u,v)* e *(v,u)* são consideradas como uma única aresta. A relação de adjacência é simétrica.
  - Self-loops não são permitidos.



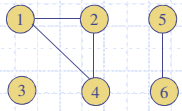
Grafos Não-Direcionados

- *G = (V,A)*
  - V = {1, 2, 3, 4}
  - A = {(1,2),(1,4),(2,4)}



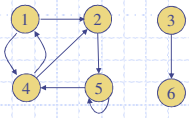
Definições

- Em grafos não direcionados:
  - O **grau** de um vértice é o número de arestas que incidem nele.
  - Um vértice de grau zero é dito **isolado** ou **não conectado**.
  - Ex. O vértice 1 tem grau 2 e o vértice 3 é isolado.



Definições

- Em grafos direcionados
  - O **grau** de um vértice é o número de arestas que saem dele (***grau de saída ou out-degree***) mais o número de arestas que chegam nele (***grau de entrada ou in-degree***).
  - Ex. Vértice 5 tem *in-degree* 2, *out-degree* 2 e grau 4.

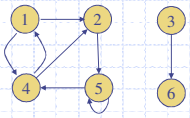


Definições

- Um **caminho de comprimento *k*** de um vértice *x* a um vértice *y* em um grafo *G = (V,A)* é uma seqüência de vértices $(v_0, v_1, v_2, \ldots, v_k)$ tal que $x = v_0$ e $y = v_k$, e $(v_{i-1}, v_i) \in A$ para $i = 1,2,\ldots,k$.
- O comprimento de um caminho é o número de arestas nele, isto é, o caminho contém os vértices $v_0, v_1, v_2, \ldots, v_k$ e as arestas $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$.

Definições

- Se existir um caminho *c* de *x* a *y* então *y* é **alcançável** a partir de *x* via *c*.
- Um caminho é **simples** se todos os vértices do caminho são distintos.
- Ex.: O caminho *(1,2,5,4)* é simples e tem comprimento 3. O caminho *(1,4,1,2)* não é simples.

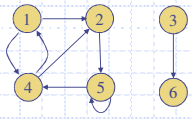


Definições

- Em um grafo direcionado:
  - Um caminho $(v_0, v_1, \ldots, v_k)$ forma um **ciclo** se $v_0 = v_k$ e o caminho contém pelo menos uma aresta.
  - O ciclo é **simples** se os vértices $v_1, v_2, \ldots, v_k$ são distintos.
  - O *self-loop* é um ciclo de tamanho 1.

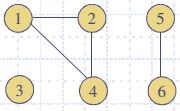
Definições

- Em um grafo direcionado:
  - Dois caminhos $(v_0, v_1, \ldots, v_k)$ e $(v'_0, v'_1, \ldots, v'_k)$ formam o **mesmo ciclo** se existir um inteiro j tal que $v'_i = v_{(i+j) \bmod k}$ para $i = 0,1,\ldots,k-1$.
  - Ex.: O caminho (1,2,5,4,1) forma um ciclo simples. O caminho (2,5,4,2) é o mesmo ciclo que os caminhos (5,4,2,5) e (4,2,5,4).



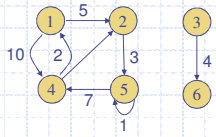
Definições

- Em um grafo não direcionado:
  - Um caminho $(v_0, v_1, \ldots, v_k)$ forma um **ciclo** se $v_0 = v_k$ e o caminho contém pelo menos três arestas.
  - O ciclo é **simples** se os vértices $v_1, v_2, \ldots, v_k$ são distintos.
  - Ex.: O caminho (1,2,4,1) é um caminho simples.



Definições

- Um **grafo ponderado** possui pesos associados às arestas.



Representação

- Existem diversas representações que podem ser utilizadas.
- Importante considerar os algoritmos em grafos como tipos abstratos de dados (TAD).
- É de fundamental importância a independência de implementação para as operações. Dessa forma, pode-se alterar a implementação do TAD sem ter que alterar a implementação do programa que utiliza o TAD.

Operações do TAD Grafo

- Inicializa(Grafo): Cria um grafo vazio.
- InsereAresta(v,u,Peso, Grafo): Insere a aresta *(v,u)* no grafo com peso.
- ExisteAresta(v,u,Grafo): Verifica se existe a aresta *(v,u)*.
- RetiraAresta(v,u,Peso, Grafo): Retira a aresta *(v,u)* do grafo e retorna seu peso.
- LiberaGrafo(Grafo): Liberar o espaço ocupado por um grafo.

Operações do TAD Grafo

- ExisteAdj(v, Grafo): retorna **verdade** se existe algum vértice adjacente à *v*.
- PrimeiroAdj(v, Grafo): retorna o endereço do primeiro vértice adjacente à *v*.
- ProxAdj(v, Grafo, u, Peso, Aux, FimAdj): retorna o vértice *u* (apontado por Aux) adjacente a *v*, bem como o peso da aresta *(v,u)*. FimAdj retorna verdade se não há mais vértices adjacentes a *v*. Ao retornar, Aux aponta para o próximo vértice adjacente a *v*.