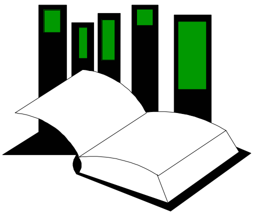
Grafos



- Teoria dos grafos

Estruturas de Dados

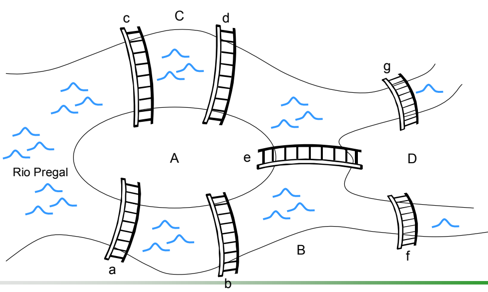
Prof. Me. Alexandre Gomes

Histórico

- A primeira evidência sobre **grafos** (*graphs*) remonta a 1736, quando Euler fez uso deles para solucionar o problema clássico das pontes de Koenigsberg
- Na cidade de Koenigsberg (na Prússia Oriental), o rio Pregal flui em torno da ilha de Kneiphof, dividindo-se em seguida em duas partes
- Assim sendo, existem quatro áreas de terra que ladeiam o rio: as áreas de terra (A-D) estão interligadas por sete pontes (a-g)
- O problema das pontes de Koenigsberg consiste em se determinar se, ao partir de alguma área de terra, é possível atravessar todos os pontos exatamente uma vez, para, em seguida, retornar à área de terra inicial

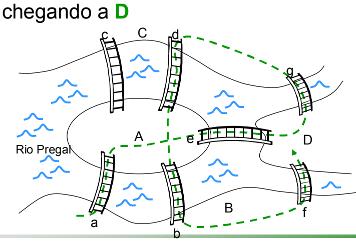
Histórico

- É possível caminhar sobre cada ponte exatamente uma única vez e retornar ao ponto de origem?



Histórico

- Um caminho possível consistiria em iniciar na área de terra **B**, atravessar a ponte **a** para a ilha **A**; pegar a ponte **e** para chegar à área **D**, atravessar a ponte **g**, chegando a **C**; cruzar a ponte **d** até **A**; cruzar a ponte **b** até **B** e a ponte **f**, chegando a **D**



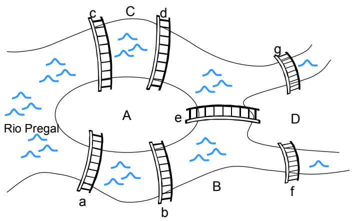
Histórico

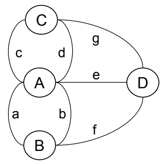
- Um caminho possível consistiria em iniciar na área de terra **B**, atravessar a ponte **a** para a ilha **A**; pegar a ponte **e** para chegar à área **D**, atravessar a ponte **g**, chegando a **C**; cruzar a ponte **d** até **A**; cruzar a ponte **b** até **B** e a ponte **f**, chegando a **D**
- Esse caminho não atravessa todas as pontes uma vez, nem tampouco retorna à área inicial de terra **B**
- Euler provou que não é possível o povo de Koenigsberg atravessar cada ponte exatamente uma vez, retornando ao ponto inicial
- Ele resolveu o problema, representando as áreas de terra como vértices e as pontes como arestas de um grafo (na realidade, um multigrafo)

Definição

- Um grafo G(**V**,**E**) é composto de
  - **V** é um conjunto não-vazio de **vértices**
    - vértice (***vertex***); vértices (***vertices, vertexes***)
  - **E** é um conjunto de **arestas** (***edges***), conectando os **vértices** em **V**
    - Uma **aresta** (**u**,**v**) é um par de vértices, ou seja, $u \in V$ e $v \in V$
- Usaremos a notação
  - **V** ou **V(G)** para representar o conjunto de vértices de G
  - **E** ou **E(G)** para representar o conjunto de arestas de G
  - G ou G=(**V**,**E**) ou G(**V**,**E**) para representar um grafo
  - **n** = |**V**| é o número de vértices
  - **e** = |**E**| é o número de arestas
- Em geral, **|A|** indica a cardinalidade (número de elementos) do conjunto **A**

Problema das Pontes de Koenigsberg como um Grafo



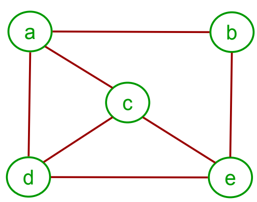


Problema das Pontes de Koenigsberg como um Grafo

- A solução é elegante e tem aplicação a todos os grafos
- Definindo o grau de um vértice como sendo o número de arestas que lhe são incidentes, Euler mostrou que existe um caminho com ponto de início em qualquer vértice, que passa através de cada aresta exatamente uma vez e termina no vértice inicial contanto que o grau de cada vértice seja par
- O caminho que cumprir com essas condições é denominado **Euleriano**
- Não existe nenhum caminho Euleriano nas pontes de Koenigsberg, uma vez que todos os quatro vértices têm grau ímpar

Exemplo

- **V**={a,b,c,d,e}
- **E**={(a,b),(a,c),(a,d), (b,e),(c,d),(c,e), (d,e)}
- Número de vértices, **n**
  - **n** = |**V**| = 5
- Número de arestas, e
  - **e** = |**E**| = 7



Aplicações

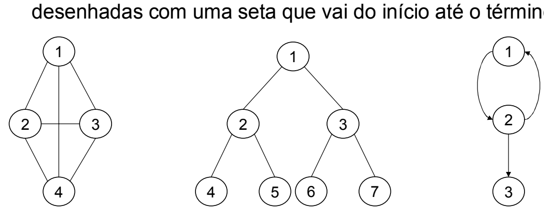
- Análise de circuitos elétricos
- Verificação de caminhos mais curtos
- Análise de planejamento de projetos (*scheduling*)
- Identificação de compostos químicos
- Genética
- Cibernética
- Lingüística
- Ciências Sociais, etc
- Pode-se afirmar que de todas as estruturas matemáticas, grafos são as que se encontram em uso mais amplo

Terminologia

- Em um **grafo não-orientado** (ou **não-dirigido**), não há ordenação especial no par de vértices que representam qualquer aresta
  - Assim sendo, os pares **(u,v)** e **(v,u)** representam a mesma aresta
- Num **grafo orientado** (ou **dirigido** *ou* **dígrafo**) cada aresta é representada por um par dirigido **(u,v)**, onde **u** é o início e **v** o término da aresta
  - Assim sendo **(u,v)** e **(v,u)** representam duas arestas distintas

Terminologia

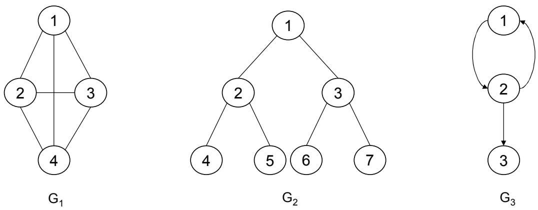
- Os grafos G_1 e G_2 são não-orientados
- O grafo G_3 é um grafo orientado (dígrafo)
- Observe que as arestas de um grafo orientado são desenhadas com uma seta que vai do início até o término



G_1 G_2 G_3

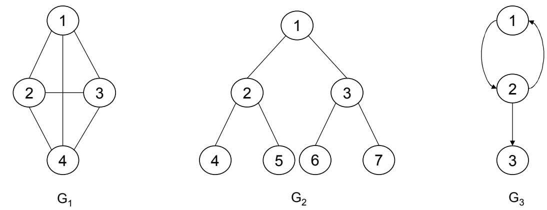
Terminologia

- $V(G_1)$={1,2,3,4}; $E(G_1)$={(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)}
- $V(G_2)$={1,2,3,4,5,6,7};
  $E(G_2)$={(1,2),(1,3),(2,4),(2,5),(3,6),(3,7)}
- $V(G_3)$={1,2,3}; $E(G_3)$={(1,2), (2,1), (2,3)}



Terminologia

- O grafo G_2 é também uma árvore, ao passo que os grafos G_1 e G_3 não são



Terminologia

- As árvores podem ser definidas como sendo casos especiais de grafos (veremos isso mais adiante)
- Vamos precisar que se (v_i,v_j) ou (v_j,v_i) é uma aresta em E(G), então $v_i \neq v_j$
- Uma vez que E(G) é um conjunto, um grafo não pode ter ocorrências múltiplas da mesma aresta
  - Quando há mais de uma ocorrência de uma mesma aresta, o objeto de dados é denominado **multigrafo**
- O mesmo é valido para V(G), ou seja, não há ocorrências múltiplas de um mesmo vértice

Número Máximo de Arestas

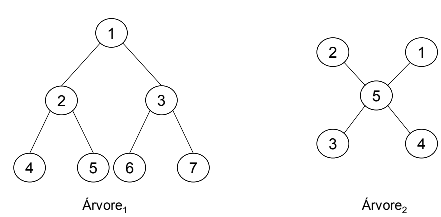
- O número de pares diferentes não-ordenados (v_i,v_j) com $v_i \neq v_j$ em um grafo com **n** vértices é **n***(**n**-1)/2
- Assim, o número máximo de arestas em qualquer grafo não-orientado com **n** vértices é **n***(**n**-1)/2
- Um grafo não-orientado com **n** vértices e com exatamente **n***(**n**-1)/2 arestas é denominado **completo**; caso contrário é denominado **incompleto** (ou **não-completo**)
- Para o caso de um grafo orientado com **n** vértices, o número máximo de arestas é **n***(**n**-1)

Número Máximo de Arestas

- Intuitivamente, em um grafo completo com **n** vértices, cada um dos **n** vértices é incidente a (**n**-1) arestas; assim, cada aresta é contada duas vezes, ou seja, resultando em **n***(**n**-1)/2

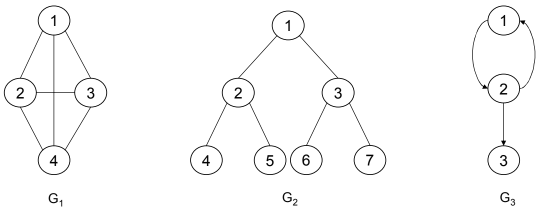
Número Máximo de Arestas em uma Árvore

- Uma árvore com **n** vértices possui exatamente (**n**-1) arestas



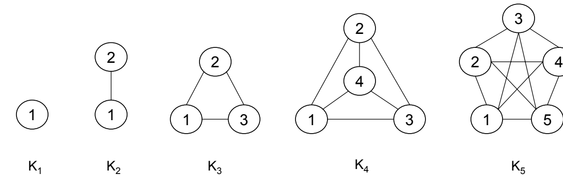
Grafos Completo e Incompleto

- G_1 é o grafo completo com 4 vértices, enquanto que G_2 e G_3 são grafos incompletos



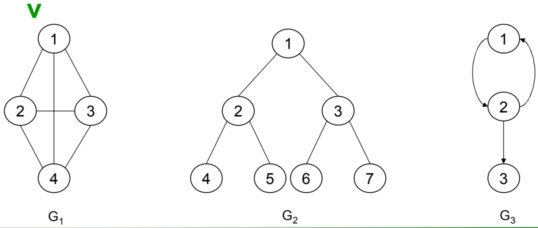
Grafos Completos

- Em geral, **K_n** denota o grafo completo com **n** vértices



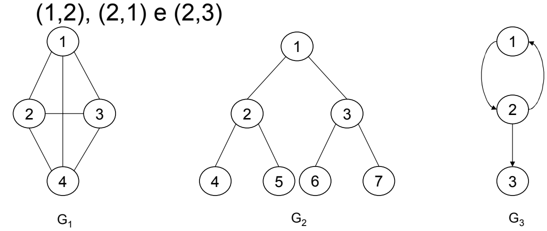
Vértices Adjacentes

- Sendo **(u,v)** uma aresta em E(G), dizemos que os vértices **u** e **v** são **adjacentes** e que a aresta **(u,v)** é **incidente** nos vértices **u** e **v**



Vértices Adjacentes

- Os vértices adjacentes ao vértice 3 em G_2 são 1, 6 e 7
- Em G_3, as arestas incidentes ao vértice 2 são (1,2), (2,1) e (2,3)



Grau

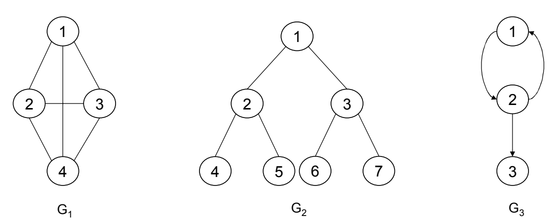
- O **grau** de um vértice **v,** escrito como **grau(v)**, é o número de arestas incidentes no vértice **v**
- Caso G seja um grafo orientado:
  - o **grau de entrada** de um vértice **v** é definido como sendo o número de arestas para as quais **v** seja o término
  - o **grau de saída** é o número de arestas para as quais **v** é o início
- Em grafo G com **n** vértices $\{v_1, v_2, ..., v_n\}$ e **e** arestas, é fácil perceber que

$$e = \frac{1}{2}\sum_{i=1}^{n} grau(v_i)$$

- No restante desta apresentação vamos nos referir a um grafo orientado como dígrafo; um grafo não-orientado será, por vezes, chamado simplesmente de um grafo

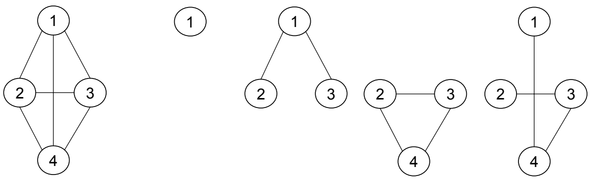
Grau

- O grau de vértice 1 em G_1 é 3
- O vértice 2 de G_3 tem grau de entrada igual a 1, grau de saída igual a 2 e grau igual a 3



Subgrafo

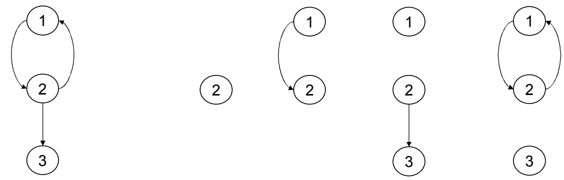
- Um **subgrafo** de G é um grafo G' tal que V(G') ⊆ V(G) e E(G') ⊆ E(G)



G_1 — Subgrafos de G_1

Subgrafo

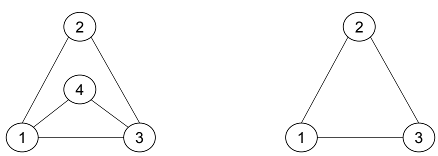
- Um **subgrafo** de G(V,E) é um grafo G'(V',E') tal que V(G') ⊆ V(G) e E(G') ⊆ E(G)



G_3 — Subgrafos de G_3

Clique

- O **clique** de um grafo G é um subgrafo de G que seja completo

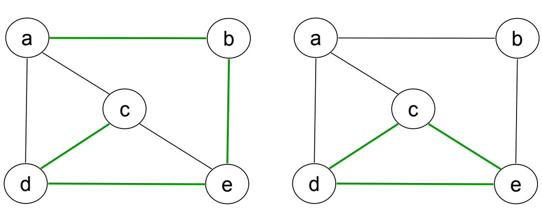


G — clique de G

Caminho

- Um **caminho** (*path*) do vértice v_p para o vértice v_q no grafo G é uma seqüência de vértices v_p, v_{i1}, v_{i2}, ...,v_{in}, v_q de tal maneira que (v_p,v_{i1}), (v_{i1},v_{i2}), ...,(v_{in},v_q) são arestas em E(G)
- O **comprimento** (ou **tamanho**) de um caminho é o número de arestas que ele contém
- Um **caminho simples** é um caminho em que são diferentes todos os vértices, com a possível exceção do primeiro e o do último
- Um **ciclo** é um caminho simples em que o primeiro e o último vértices são iguais
- Um **trajeto** é um caminho no qual todas as arestas são distintas

Caminho

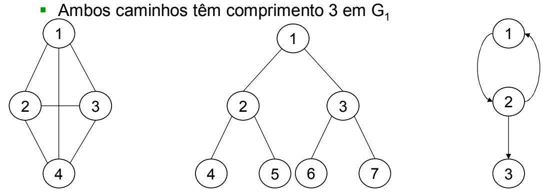


Caminho: a,b,e,d,c
Tamanho: 4

Ciclo: c,e,d,c
Tamanho: 3

Caminho

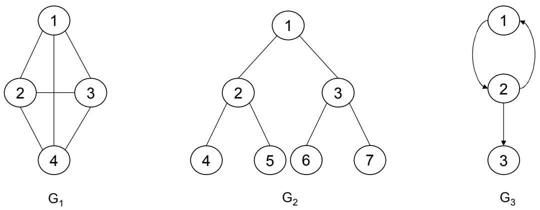
- O caminho (1,2),(2,4),(4,3) em G_1, também escrito como 1,2,4,3, é caminho simples ao passo que (1,2),(2,4),(4,2) escrito como 1,2,4,2 também é um caminho em G_1, mas não um caminho simples
  - Ambos caminhos têm comprimento 3 em G_1



G_1 — G_2 — G_3

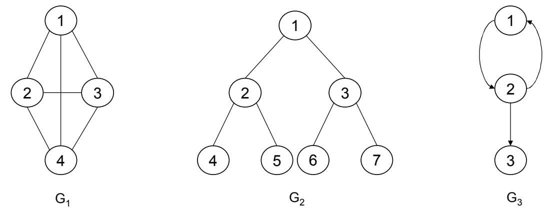
Caminho

- 1,2,3 é um caminho simples orientado em G_3
- 1,2,3,2 não é um caminho em G_3, uma vez que a aresta (3,2) não se encontra em $E(G_3)$



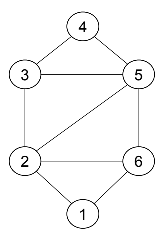
Ciclo

- 1,2,3,1 é um ciclo em G_1
- 1,2,1 é um ciclo orientado em G_3
- Normalmente, para grafos orientados, acrescentamos o termo "orientado" aos termos ciclo e caminho



Caminhos Hamiltoniano e Euleriano

- Caminho ou Ciclo Hamiltoniano
  - É um caminho que contém cada vértice do grafo exatamente uma vez.
  - Um ciclo $v_1, ..., v_k, v_{k+1}$ é hamiltoniano quando o caminho $v_1, ..., v_k$ o for
- Caminho ou Ciclo Euleriano
  - É um caminho que contém cada aresta do grafo exatamente uma vez
- caminho hamiltoniano 3, 4, 5, 2, 1, 6
- caminho euleriano 3, 4, 5, 3, 2, 5, 6, 2, 1, 6

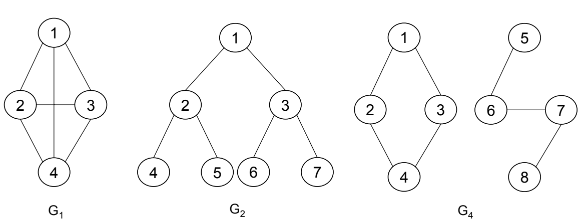


Vértices e Grafos Interligados

- Em um grafo não-orientado G dois **vértices** v_p e v_q se dizem **interligados** se houver um caminho em G desde v_p até v_q (uma vez que G não é orientado, isto significa que há também um caminho desde v_q até v_p)
- Um **grafo** não-orientado se diz **interligado** (**conexo**) se para cada par de vértices individuais v_i e v_j em V(G) existe um caminho desde v_i até v_j em G; caso contrário G é **não-interligado** (**desconexo**)
- Assim, se **e**<(**n**-1) então G é desconexo, onde **e**=|E(G)|, **n**=|V(G)|

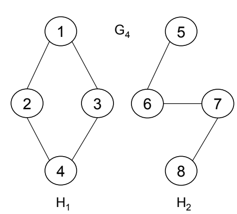
Vértices e Grafos Interligados

- Os grafos G_1 e G_2 são conexos (interligados) ao passo que G_4 não o é



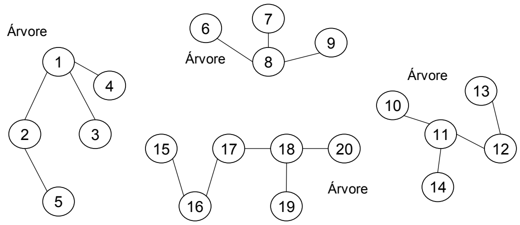
Componente Conexo

- Um **componente conexo**, ou simplesmente um **componente** de grafo não-orientado, é um subgrafo interligado ao máximo
- No exemplo, G_4 tem dois **componentes** H_1 e H_2



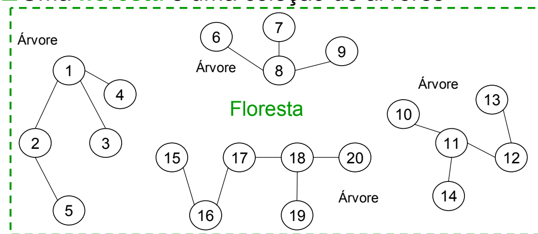
Componente Conexo

- Uma **árvore** é um grafo conexo sem ciclos (acíclico)
- Uma **floresta** é uma coleção de árvores



Componente Conexo

- Uma **árvore** é um grafo conexo sem ciclos (acíclico)
- Uma **floresta** é uma coleção de árvores



Árvore é um Grafo Conexo Acíclico

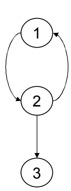
- Teorema:
  - Um grafo G é uma árvore se e somente se existir um único caminho entre cada par de vértices G
- Prova:
  - Se G uma árvore então G é conexo
  - Portanto existe pelo menos um caminho entre cada par de vértices **v** e **w** de G
  - Suponha que existem dois caminhos distintos (**v**,P_1,**w**) e (**v**,P_2,**w**) entre **v** e **w**; então o caminho (**v**,P_1,**w**,P_2,**v**) forma um ciclo, o que contradiz G ser acíclico
  - Reciprocamente, se existe exatamente um caminho entre cada par de vértices de G, então o grafo é obviamente conexo, e além disso não pode conter ciclos
  - Portanto, G é uma árvore

Componente Conexo

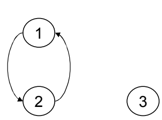
- Um **grafo** orientado G se diz **fortemente conexo** (**fortemente interligado**) se para cada par de vértices diferentes v_i e v_j em V(G) existe um caminho orientado desde v_i até v_j e também de v_j para v_i
- Um **componente fortemente interligado** é um subgrafo máximo fortemente interligado

Componente Conexo

- O grafo G_3 não está fortemente interligado, uma vez que não existe nenhum caminho de v_3 até v_2
- G_3 tem dois componentes fortemente interligados



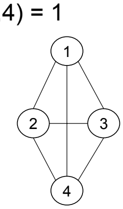
G_3



Componentes de G_3 fortemente interligados

Distância

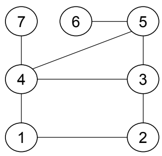
- A **distância d(v,w)** entre dois vértices **v** e **w** de um grafo é o tamanho do menor caminho entre **v** e **w**
- No exemplo, d(1,4) = 1



G_1

Excentricidade

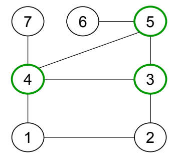
- A excentricidade de um vértice **v** é a distância máxima entre **v** e **w**, para todo **w** de V(G)



Vértice	Excentricidade
1	3
2	3
3	2
4	2
5	2
6	3
7	3

Centro

- O **centro** de um grafo G é o subconjunto de vértices de excentricidade mínima
- No exemplo, centro(G)={3,4,5}



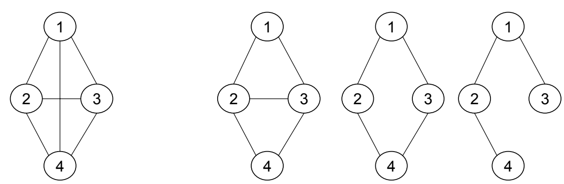
Vértice	Excentricidade
1	3
2	3
3	2
4	2
5	2
6	3
7	3

Subgrafo e Árvore de Cobertura

- O **subgrafo de espalhamento** (*spanning graph*), ou **subgrafo gerador** ou **subgrado estendido**, de um grafo $G_1(V_1,E_1)$ é um subgrafo $G_2(V_2,E_2)$ de G_1, onde $V_1 = V_2$
- Quando o subgrafo de espalhamento é uma árvore, ele recebe o nome de **árvore de espalhamento**, ou **árvore geradora** ou **árvore estendida** (*spanning tree*)
- Dessa forma, a **árvore de geradora** de um grafo G é uma árvore contendo todos os vértices de G
- Um grafo G pode possuir várias árvores de cobertura

Subgrafo e Árvore de Cobertura

- G_2, G_3 e G_4 são subgrafos de cobertura de G_1
- G_3 e G_4 são subgrafos de cobertura de G_2
- G_4 é subgrafo de cobertura de G_3
- G_4 é árvore de cobertura de G_1, G_2 e G_3



G_1 G_2 G_3 G_4

Árvore de Cobertura

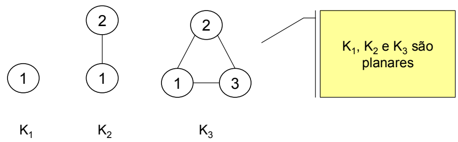
- Todo grafo conexo G(V,E) possui uma árvore de cobertura que pode ser obtida da seguinte forma
  - Para cada aresta (u,v) ∈ E(G)
    - Se G(V,E-{(u,v)}) for conexo então remova (u,v) de G
- Quando todas as arestas que permanecerem tiverem sido consideradas, o grafo resultante é uma árvore de cobertura de G

Planaridade

- Seja G um grafo e uma representação geométrica de G em um plano (duas dimensões)
- A representação é denominada **plana** quando não houver cruzamento de linhas (exceto nos vértices)
- Um grafo é denominado **planar** se possuir pelo menos uma representação plana
- As linhas de representação dividem o plano em regiões, denominadas **faces**
  - Existe exatamente uma região ilimitada (chamada face externa)
  - Duas representações planas de um mesmo grafo possuem sempre o mesmo número de faces
- Todo grafo planar admite uma representação plana em que todas as linhas são retas

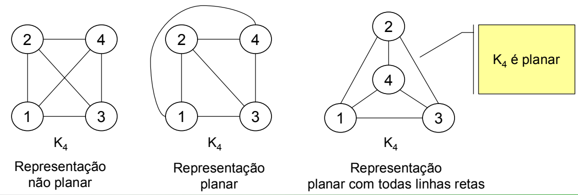
Planaridade

- Se G é um grafo planar então **n**+**f** = **e**+2
  - **n**=|V|, **e**=|E|, **f** é o número de faces
- Se G é um grafo planar então **e** ≤ 3*__n__-6
- Assim, quanto maior o número de arestas em relação ao número de vértices, mais difícil se torna obter representações planas



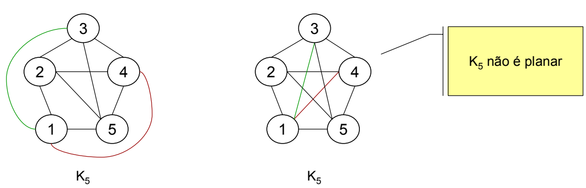
Planaridade

- Se G é um grafo planar então **n**+**f** = **e**+2
  - **n**=|V|, **e**=|E|, **f** é o número de faces
- Se G é um grafo planar então **e** ≤ 3*__n__-6



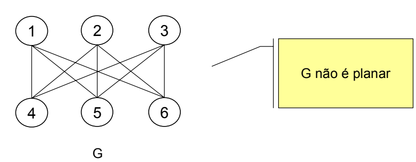
Planaridade

- Se G é um grafo planar então **n**+**f** = **e**+2
  - **n**=|V|, **e**=|E|, **f** é o número de faces
- Se G é um grafo planar então **e** ≤ 3*__n__-6



Planaridade

- Se G é um grafo planar então **n**+**f** = **e**+2
  - **n**=|V|, **e**=|E|, **f** é o número de faces
- Se G é um grafo planar então **e** ≤ 3*__n__-6

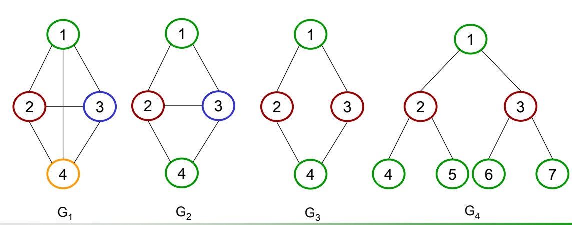


Coloração

- Uma **coloração** de G(V,E) é uma atribuição de alguma cor para cada vértice de V, de forma que dois vértices adjacentes possuam cores distintas
- Uma **k-coloração** é uma coloração que utiliza um máximo de **k** cores
- O **número cromático** de um grafo G é número mínimo de cores **k** para o qual existe uma **k**-coloração de G
- Colorir um grafo é simples, entretanto não é trivial encontrar um algoritmo eficiente para obtenção do número cromático; na realidade ainda é desconhecido um algoritmo eficiente para isso

Coloração

- Numero cromático de G_1 = 4; G_2 = 3, G_3 = 2; G_4 = 2

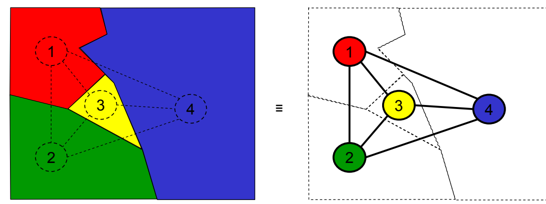


Coloração

- O **problema das 4 cores**, ou **coloração de mapas usando quatro cores**, data de 1852 quando o inglês Francis Guthrie observou que apenas quatro cores eram suficientes para colorir o mapa dos condados da Inglaterra
- O **problema das 4 cores** consiste em colorir os países de um mapa arbitrário plano, cada país com uma cor, de tal forma que países fronteiriços possuam cores diferentes, usando no máximo 4 cores, teorema provado em 1977 por Appel e Haken
- Desde a prova do teorema, os algoritmos mais eficientes encontrados para 4-coloração de mapas quererem O(**n**^{2}), onde **n** é o número de vértices

Coloração

- Cada região do mapa é substituída por um vértice
- Dois vértices são conectados por uma aresta se e somente se as duas regiões são fronteiriças (compartilham um segmento de borda)



Especificação

- bool Graph::Empty();
  - retorna **true** se o grafo está vazio; **false** caso contrário
- int Graph::numVertices();
  - Retorna o número de vértices
- int Graph::numEdges();
  - Retorna o número de arestas
- int Graph::Size();
  - Retorna o número de vértices mais arestas
- vertex Graph::Vertex(int i)
  - Retorna o **i**-ésimo vértice, 1 <= **i** <= numVertices()
- edge Graph::Edge(int j)
  - Retorna a **j**-ésima aresta , 1 <= **j** <= numEdges()

Especificação

- float Graph::distance(vertex v, vertex w)
  - Retorna a distância entre os vértices **v** e **w**
- int Graph::degree(vertex v)
  - Retorna o grau de **v**
- int Graph::inDegree(vertex v)
  - Retorna o grau de entrada de **v**
- int Graph::outDegree(vertex v)
  - Retorna o grau de saída de **v**
- edges Graph::incidentEdges(vertex v)
  - Retorna uma enumeração de todas as arestas incidentes ao vértice **v**
- edges Graph::inIncidentEdges(vertex v)
  - Retorna uma enumeração de todas as arestas que chegam no vértice **v**
- edges Graph::outIncidentEdges(vertex v)
  - Retorna uma enumeração de todas as arestas que partem do vértice **v**

Especificação

- vertices Graph::adjacentVertices(vertex v)
  - Retorna uma enumeração dos vértices adjacentes a **v**
- vertices Graph::inAdjacentVertices(vertex v)
  - Retorna uma enumeração dos vértices adjacentes a **v** considerando arestas que chegam a **v**
- vertices Graph::outAdjacentVertices(vertex v)
  - Retorna uma enumeração dos vértices adjacentes a **v** considerando arestas que partem de **v**
- bool Graph::areAdjacent(vertex v, vertex w)
  - Retorna **true** se os vértices **v** e **w** são adjacentes, **false** caso contrário
- vertex Graph::insertVertex(object o)
  - Insere e retorna um novo (isolado) vértice, armazenando **o** na sua posição
- edge Graph::insertEdge(vertex v, vertex w, object o)
  - Insere e retorna uma aresta não orientada entre v e w, armazenando **o** na sua posição
- edge Graph::insertDirectedEdge(vertex v, vertex w, object o)
  - Insere e retorna uma aresta orientada entre **v** e **w**, armazenando **o** na sua posição
- void Graph::removeEdge(edge e)
  - Remove aresta **e**

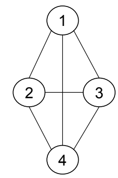
Representação

- Embora sejam possíveis diversas representações dos grafos, vamos estudar duas mais utilizadas comumente
  - matriz de adjacência e
  - lista de adjacências
- A escolha de determinada representação dependerá da aplicação que se tem em vista e das funções que se espera realizar no grafo

Matriz de Adjacências

- Seja G=(V,E) um grafo com **n** vértices, n≥1
- A **matriz de adjacências** A de G é um arranjo bidimensional **n** x **n** com a propriedade de que
  - A[i,j] = 1 se a aresta (v_i,v_j) pertence a E(G)
  - A[i,j] = 0 caso contrário
- A matriz de adjacências para um grafo não-orientado é simétrica, pois a aresta (v_i,v_j) está em E(G), se a aresta (v_j,v_i) também está em E(G)
- A matriz de adjacências de um grafo orientado não é necessariamente simétrica
- O espaço necessário para representar um grafo usando a matriz de adjacências é de **n**^{2} bits
  - Aproximadamente metade desse espaço pode ser poupado no caso dos grafos não-orientados, armazenando apenas o triângulo superior ou inferior da matriz

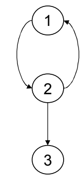
Matriz de Adjacências



G_1

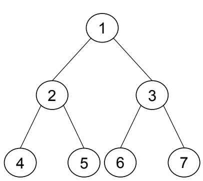
$$\begin{array}{cc} & \begin{array}{cccc}1&2&3&4\end{array} \\ \begin{array}{c}1\\2\\3\\4\end{array} & \begin{bmatrix}0&1&1&1\\1&0&1&1\\1&1&0&1\\1&1&1&0\end{bmatrix}\end{array}$$

$$\begin{array}{cc} & \begin{array}{ccc}1&2&3\end{array} \\ \begin{array}{c}1\\2\\3\end{array} & \begin{bmatrix}0&1&0\\1&0&1\\0&0&0\end{bmatrix}\end{array}$$



G_3

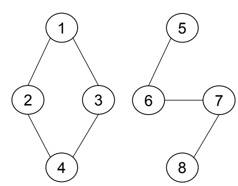
Matriz de Adjacências



G_2

$$\begin{array}{cc} & \begin{array}{ccccccc}1&2&3&4&5&6&7\end{array} \\ \begin{array}{c}1\\2\\3\\4\\5\\6\\7\end{array} & \begin{bmatrix}0&1&1&0&0&0&0\\1&0&0&1&1&0&0\\1&0&0&0&0&1&1\\0&1&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&1&0&0&0&0\end{bmatrix}\end{array}$$

Matriz de Adjacências



G_4

$$\begin{array}{cc} & \begin{array}{cccccccc}1&2&3&4&5&6&7&8\end{array} \\ \begin{array}{c}1\\2\\3\\4\\5\\6\\7\\8\end{array} & \begin{bmatrix}0&1&1&0&0&0&0&0\\1&0&0&1&0&0&0&0\\1&0&0&1&0&0&0&0\\0&1&1&0&0&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&0&1&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}\end{array}$$

Matriz de Adjacências

- A partir da matriz de adjacências é possível determinar se existe uma aresta que liga quaisquer dois vértices v_i e v_j
- Para um grafo não-orientado, o grau de qualquer vértice v_i vem a ser a soma de sua linha:

$$\sum_{j=1}^{n} A[i,j]$$

- Para um grafo orientado, a soma da linha é o grau de saída ao passo que a soma da coluna vem a ser o valor grau de entrada

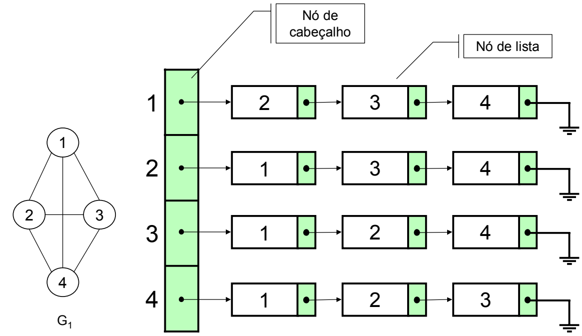
Matriz de Adjacências

- Suponha que queremos responder perguntas sobre grafos tais como:
  - Quantas arestas existem em G?
  - Será que G está interligado?
- Usando as matrizes de adjacências, todos os algoritmos vão exigir, pelo menos, O(**n**^{2}) de tempo uma vez que (**n**^{2}-**n**) entradas da matriz (a diagonal principal possui somente zeros) têm de ser examinadas
- Se o grafo for esparso, isto é, quando a maioria dos termos na matriz de adjacências for zero, é possível responder as perguntas acima em tempo O(**n**+**e**), onde **e** é o número de arestas em G e **e** << **n**^{2}/2
- Esse aceleramento pode ser possibilitado mediante a utilização de listas ligadas, nas quais são representadas apenas as arestas existentes em G, o que nos leva à próxima representação para grafos

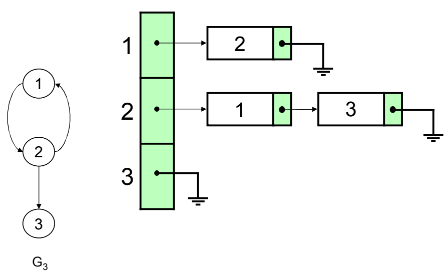
Lista de Adjacências

- Nesta representação as **n** linhas da matriz de adjacências são representadas como **n** listas encadeadas, ou seja, existe uma lista para cada vértice em G
- Os nós na lista **i** representam os vértices que são adjacentes ao vértice v_i
- Cada nó possui, pelo menos, dois campos
  - Um campo que contém o índice do vértice adjacente ao vértice v_i
  - Um campo de ligação com o próximo vértice adjacente ao vértice v_i

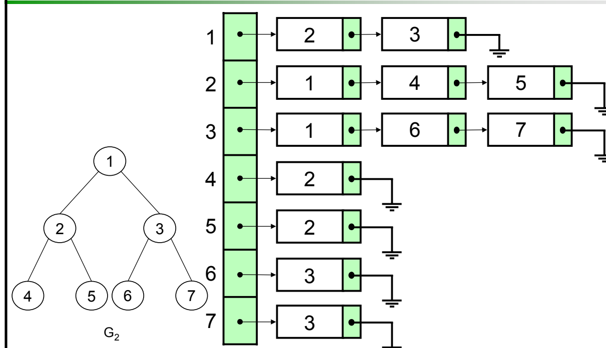
Lista de Adjacências



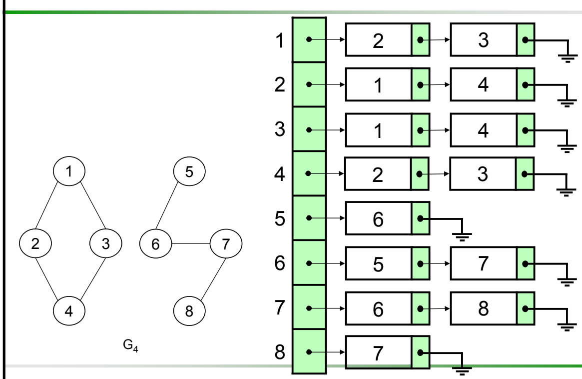
Lista de Adjacências



Lista de Adjacências



Lista de Adjacências



Lista de Adjacências

- Cada lista possui um nó de cabeçalho (início ou *head* da lista)
  - Os nós de cabeçalho são seqüenciais, permitindo fácil acesso aleatório à lista de adjacências de determinado vértice
- No caso de um grafo não-orientado com **n** vértices e **a** arestas, essa representação requer **n** nós de cabeçalho e 2***a** nós de lista, sendo que cada nó de lista possui 2 campos (vértice adjacente + ligação)
- Em termos de quantidade de bits de memória necessária, essa contagem deve ser multiplicada por $\log_2(n)$ para os nós de cabeçalho e por log2(**n**)+log2(**a**) para os nós de lista, uma vez que são necessários $\log_2(x)$ bits para representar um número de valor **x**
- Em alguns casos os nós podem ser condensados seqüencialmente nas listas de adjacências eliminando-se os campos de ligação

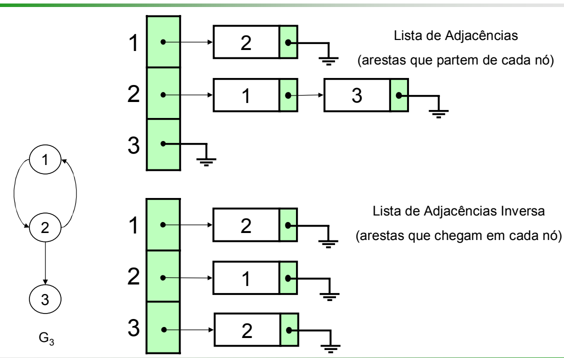
Lista de Adjacências

- O grau de qualquer vértice em um grafo não-orientado pode ser determinado simplesmente contando o número de nós na respectiva lista de adjacências
- Portanto, o número total de arestas pode ser determinado em tempo O(**n**+**e**)
- No caso de um dígrafo, o número total de arestas corresponde ao número de nós de lista
- O grau-de-saída de qualquer vértice pode ser determinado contando o número de nós na lista de adjacências
- O número total de arestas em G pode, portanto, ser determinado em O(**n**+**e**)

Lista de Adjacências

- Determinar o grau de entrada de um vértice é uma tarefa um pouco mais complexa, havendo necessidade de ter acesso repetidamente a todos os vértices adjacentes a outro vértice
- Assim, talvez seja conveniente o esforço de manter outro conjunto de listas, além das listas de adjacências
- Esse conjunto de listas, chamado de **listas de adjacência inversa**, conterá uma lista para cada vértice
  - Cada vértice conterá um nó para cada vértice adjacente ao vértice que representa
- Alternativamente, pode-se adotar Listas Cruzadas, estrutura também utilizada para a representação de matrizes esparsas

Lista de Adjacências Inversa



Representação

- Em algumas situações são atribuídos pesos às arestas de um grafo
- Esses pesos podem representar a distância de um vértice para outro ou o custo de se passar de um vértice para outro adjacente
- Neste caso as entradas da matriz de adjacências A[i,j] mantêm também essas informações
- No caso das listas de adjacências, essa informação sobre pesos pode ser conservada nos nós de listas, incluindo um campo adicional
- Um grafo com arestas dotadas de pesos é denominado **rede** ou **grafo ponderado**