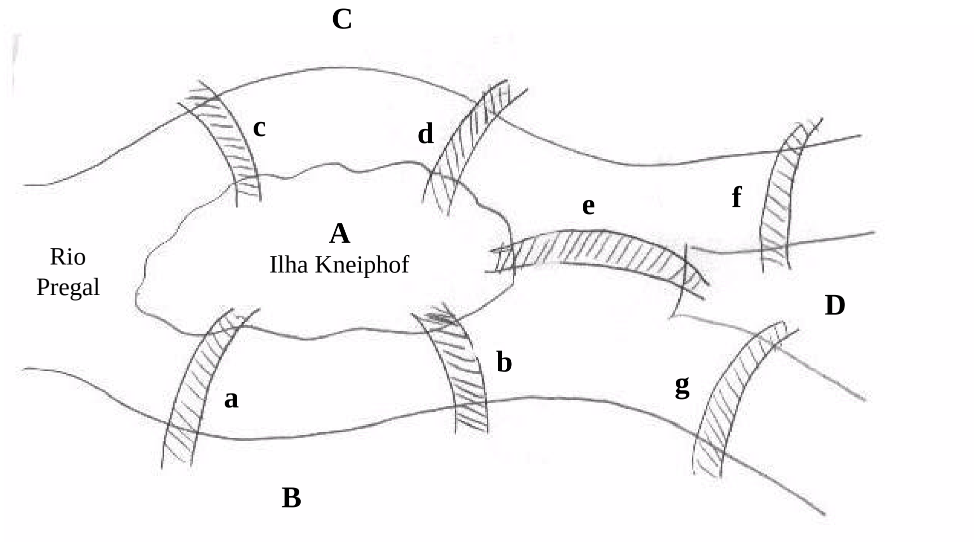
Aula 09 - Conteúdo

1) Grafos
2) Representação de Grafos
3) Algoritmos
4) Exercícios

Grafos

A primeira evidência de que se tem notícia quanto a aplicação de grafos remonta a 1736, o ano em que Euler fez uso deles para solucionar o agora clássico problema da ponte de Koenigsberg (Prussia Oriental).



Problema: ao partir de alguma área de terra, é possível atravessar todas as partes exatamente uma vez para, em seguida, retornar a área de terra inicial.

Euler mostrou que existe um caminho com ponto de início em qualquer vértice que passa através de cada borda exatamente uma vez e termina no vertice inicial contanto que seja de valor par o grau de cada vértice. O caminho que não cumprir com essas condições não é Euleriano. No exemplo da ponte, todos os quatro vértices têm grau ímpar.

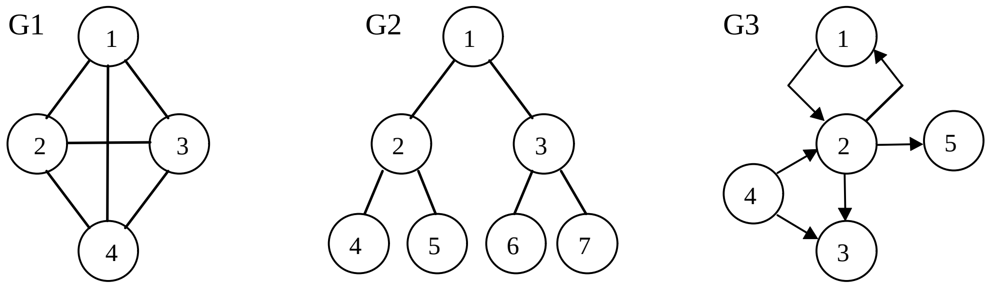
Aplicações:

- análise de circuitos elétricos;
- percurso em grafos - caminho de custo mínimo;
- análise de planejamento de projetos,
- identificação de compostos químicos,
- genética e cibernética.

Definição: Um grafo G consiste de dois conjuntos V e E. V é um conjunto de vértices não vazio, e E é um conjunto de pares de vértices denominadas bordas. V(G) e E(G) representarão o conjunto de vértices e bordas do grafo G. Vamos escrever G = (V,E) para representar um grafo.

Num grafo não dirigido, o par de vértices não tem ordenação especial, assim sendo os pares (v_1,v_2) e (v_2,v_1) representam a mesma borda. Num grafo dirigido, cada borda é representada por uma par dirigido (v_1,v_2) onde v_1 é o início e v_2 é o fim da borda. Dessa maneira, (v_1,v_2) e (v_2,v_1) representam bordas diferentes.

Exemplo:



V(G1) = {1,2,3,4 E(G1) = {(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)}
V(G2) = {1,2,3,4,5,6,7} E(G2) = {(1,2), (1,3), (2,4), (2,5), (3,6), (3,7)}
V(G3) = {1,2,3,4,5 E(G3) = {(1,2), (2,1), (2,3), (2,5), (4,2), (4,3)}

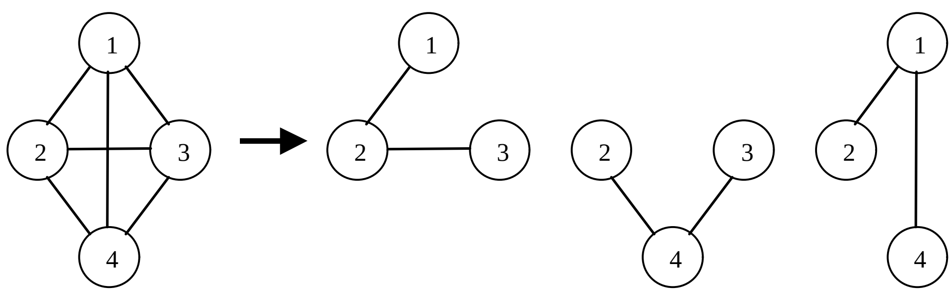
O número de pares diferentes não ordenados (v_i , v_j) com $v_i \neq v_j$ num grafo com **n** vértices é n (n-1)/2. Um grafo não dirigido com **n** vértices e com exatamente n(n-1)/2 bordas se denomina **completo**. No caso de um grafo dirigido com **n** vértices o número máximo de bordas passa a ser n(n-1).

Sendo (v_1 , v_2) uma borda em E(G), dizemos que os vértices v_1 e v_2 são adjacentes e que a borda (v_1 , v_2) é incidente nos vértices v_1 e v_2 .

Um subgrafo de G vem a ser um grafo G` de tal natureza que V(G`) $\subseteq$ V(G) e E(G`) $\subseteq$ E(G).

Exemplo:

Grafo G1 Subgrafos de G1



Um caminho 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) tem bordas em E(G). O comprimento de um caminho vem a ser o número de bordas que o caminho contém. Um caminho simples é um caminho em que são diferentes todos os vértices com a possível exceção do primeiro e do último.

Considere o grafo G1 acima. O caminho (1,2), (2,4) e (4,3) é um caminho simples e o caminho (1,2), (2,4) e (4,2) é um caminho não simples.

Ciclo é um caminho simples onde o primeiro e o último vértices são os mesmos. Por exemplo: (1,2), (2,3) e (3,1) é um ciclo em G1.

O **grau** de um vértice é o número de bordas incidentes nesse vértice. Num grafo dirigido definimos **grau de entrada** como as bordas que *chegam* num vértice e **grau de saída** como sendo as bordas que *saem* do vértice.

Representação de Grafos.

A escolha de uma representação dependerá da aplicação que se tem em vista e das funções que se espera realizar no grafo. Veremos três maneiras de representar grafos: a matriz de adjacências e a lista de adjacências.

Matriz de Adjacências

Seja G = (V, E) um grafo com **n** vértices , $n \geq 1$. A matriz de adjacências de G é um matriz A(nxn) com a propriedade de que A(i,j) = 1 se a borda (v_i, v_j) fica em E(G) e A(i,j) = 0 se a borda (v_i, v_j) não existe. Observe que a matriz de adjacências é simétrica para grafos não dirigidos.

O espaço de armazenamento da matris de adjacências é n^2 bits, sendo que metade deste espaço pode ser poupado no caso de um grafo não dirigido, pois podemos armazenar somente o triângulo superior ou inferior da matriz (a matriz é simétrica).

Exemplo:

a) Matriz de adjacências do grafo G1.

$$\begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \left\| \begin{array}{cccc} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{array} \right\| \end{array}$$

b) Matriz de adjacências do grafo G2.

$$\begin{array}{c c} & \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} & \left\| \begin{array}{ccccccc} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array} \right\| \end{array}$$

c) Matriz de adjacências do grafo G3.

$$\begin{array}{c c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \left\| \begin{array}{ccccc} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right\| \end{array}$$

Vantagens:

- pode-se determinar facilmente se existe uma borda ligando quaisquer dois vértices.
- para um grafo não dirigido, o grau de qualquer vértice **k** pode ser calculado com a soma de sua linha $\sum_{j=1}^{n} A(k, j)$.
- para um grafo dirigido, a soma da linha é o grau de saída e a soma da coluna é o grau de entrada.

Desvantagens:

- espaço de armazenamento;
- tempo de execução de algorítmos : Ordem (n^2)

Matriz de Incidência

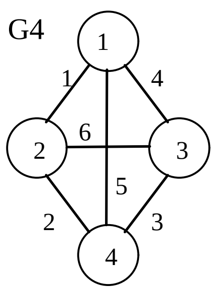
Seja G = (V, E) um grafo com **n** vértices e **k** arestas (**k** ≥ 1 e n ≥ 1). A matriz de incidência de G é um matriz A(n x k) com a propriedade de que o conteúdo de A(i,j) indica a existência de uma aresta **j** incidente no vértice **i**.

Podemos utilizar a seguinte convensão para representar uma matriz de incidência:

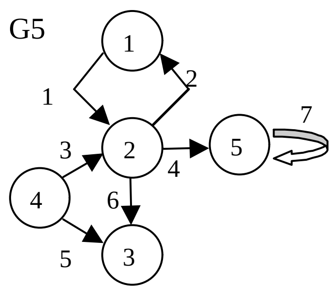
- A(i,j) = 0 indica a não incidência da aresta j no vértice i.
- A(i,j) = 1 indica uma aresta j “saindo” do vértice i.
- A(i,j) = -1 indica uma aresta j “chegando” no vértice i.
- A(i,j) = 2 indica uma aresta j “saindo e chegando” no vértice i.

Exemplo:

G4



G5



a) Matriz de incidência do grafo G4.

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

b) Matriz de incidência do grafo G5.

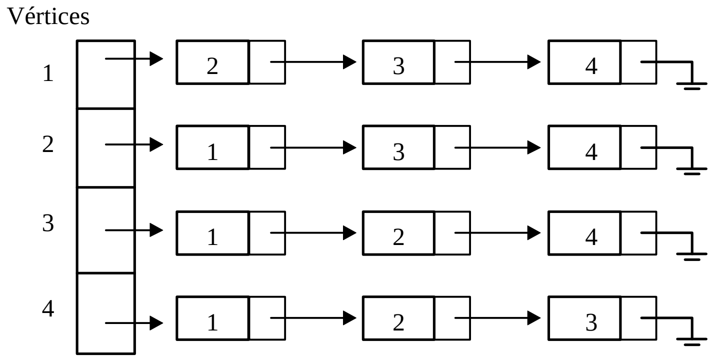
	1	2	3	4	5	6	7
1	1	-1	0	0	0	0	0
2	-1	1	-1	1	0	1	0
3	0	0	0	0	-1	-1	0
4	0	0	1	0	1	0	0
5	0	0	0	-1	0	0	2

Lista de Adjacências

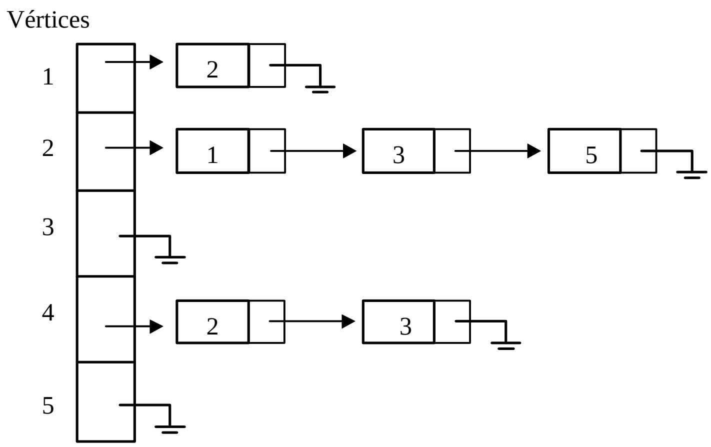
A lista de adjacências armazena somente as arestas existentes em relação a um determinado vértice. Ao contrário da matriz de adjacência, as arestas inexistentes não precisam ser indicadas.

Exemplo:

a) Lista de adjacências do grafo G1.



b) Lista de adjacências do grafo G3.



Algoritmos de Pesquisa em Grafos.

No estudo de árvores binárias desenvolvemos algoritmos que percorriam a mesma a partir de sua raiz (in order, pre order e pos order) com o intuito de acessar os vários nós. Um problema análogo ocorre com os grafos: dado um grafo G não dirigido e um vértice v_0, como visitar todos os vértices em G que são acessíveis a partir de v_0.

Pesquisa em Profundidade.

Visita-se o vértice **v**. Em seguida um vértice NÃO visitado **w**, adjacente a **v**, é selecionado iniciando-se uma nova pesquisa em profundidade a partir de **w**. Atingindo-se um vértice **u** tal que tenham sido visitados todos seus vértices adjacentes, voltamos para o último vértice visitado que tem vértice adjacente **z** NÃO visitado e iniciamos uma nova pesquisa em profundidade a partir de z.

Seja G = (V,E) um grafo não dirigido com **n** vértices, VISITADOS(n) um vetor inicialmente zerado e v_0 um vértice inicial.

Agoritmo de pesquisa em profundidade

```
Algoritmo PesqProf(v)
  VISITADOS(v) = 1
  Para cada vértice w adjacente a v faça
    Se VISITADOS(w) = 0 então  Pesq Prof(w)
  fimPara
```

Pesquisa em Largura

Começando no vértice v_0 e marcando-o ao ser visitado, a pesquisa em largura visita em seguida todos os vértices não visitados adjacentes a v_0. Em seguida, os vértices não visitados adjacentes a estes vértices são visitados e assim sucessivamente.

Algoritmo para pesquisa em largura

```
Algoritmo PesqLarg(v)
  VISITADOS(v) = 1
  InicFila(Q)         /* Q é uma fila */
  Enquanto verdadeira
    Para cada vértice w adjacente a v faça
      Se VISITADOS(w) = 0 então
        InsFila(Q,w)
        VISITADOS(w) = 1
      fimSe
    fimPara
    Se FilaVazia(Q) então Sair Enquanto
    DelFila(Q,v)
  FimEnq
```

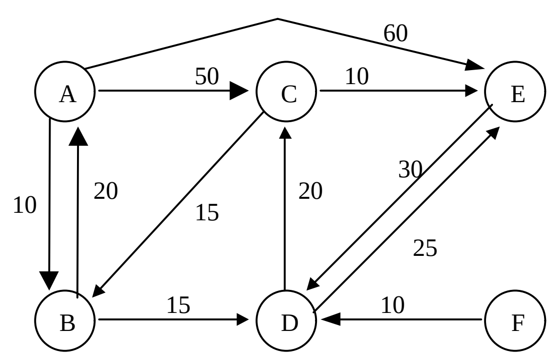
Caminho de Custo Mínimo.

Os grafos podem ser utilizados para representar a estrutura rodoviária de uma região, onde os vértices representam as cidades e as arestas os trechos de rodovia. As arestas podem ter ponderações indicando por exemplo a distância entre as duas cidades interligadas pelas mesmas.

Um motorista que deseja sair da cidade A e chegar a cidade B teria duas perguntas básicas:

- existe um caminho de A para B?
- havendo mais de um caminho de A para B, qual o caminho mais curto(mais barato)?

Exemplo:



Caminho	Custo
A, B	10
A, B, D	25
A, B, D, C	45
A, B, D, E	50

O Caminho mais curto de A para C é A, B, D, C com custo de 45 (10 + 15 + 20) e não A, C que tem custo 50. Não existe um caminho de A para F.

Matriz de Adjacências ponderada

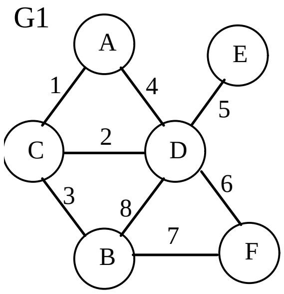
$$
\begin{array}{c|cccccc|}
 & A & B & C & D & E & F \\
A & 0 & 10 & 50 & 0 & 60 & 0 \\
B & 20 & 0 & 0 & 15 & 0 & 0 \\
C & 0 & 15 & 0 & 0 & 10 & 0 \\
D & 0 & 0 & 20 & 0 & 25 & 0 \\
E & 0 & 0 & 0 & 30 & 0 & 0 \\
F & 0 & 0 & 0 & 10 & 0 & 0
\end{array}
$$

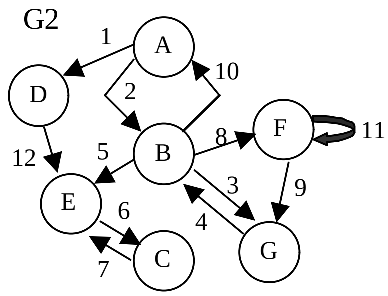
Algoritmo para determinar o caminho de custo mínimo (menor caminho)

```
Algoritmo MenorCaminho(v, CUSTO, DIST, n)
  declarar  S(n)
  Para  i = 1  até  n   faça
    S(i) = 0
    DIST(i) = CUSTO(v,i)
  fimPara
  S(v) = 1
  DIST(v) = 0
  num = 2
  Enquanto num <  n  faça
    escolha vértice u
    DIST(u)  =  min {DIST(w)}
    S(u) = 1
    num = num  +  1
    Para todo  w  com  S(w) = 0 faça
      DIST(w) = min {DIST(u), DIST(w) + CUSTO( u, w)}
    fimPara
  fimEnquanto
```

Exercícios

9.01) Represente os grafos G1 e G2 abaixo utilizando matriz de adjacências e matriz de incidências. Utilize a seguinte convenção para matriz de incidências: 1 p/ aresta saindo de um nó, -1 p/ aresta chegando ao nó e 2 p/ aresta saindo e chegando num nó.





9.02) Elabore uma função em Pascal que, tendo como parâmetros uma matriz de adjacências A, o números de nós **n** e um nó origem qualquer **na**, imprima todos os caminhos de tamanho 2 a partir de **na**. Considere um grafo não dirigido. Protótipo da função:

```
Procedure Caminho2(A:matrizNN; n,no: integer);
```

9.03) Refaça o exercício anterior utilizando matriz de incidências.

9.04) Elabore uma função em Pascal que, tendo como parâmetros uma matriz de adjacências **A** de um grafo dirigido e o número de nós **n** deste grafo, imprima o grau de entrada e o grau de saída de cada nó. Protótipo da função:

```
Procedure ImpGrausGD(A:matrizNN; n: integer);
```

9.05) Refaça o exercício anterior utilizando matriz de incidências.