Grafos

Os primeiros slides relacionados ao histórico dos grafos foram baseados nos slides do professor José Augusto Baranauskas do Departamento de Física e Matemática da Universidade de São Paulo – FFCLRP-USP.

Grafos - Histórico

• A primeira evidência sobre grafos 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;

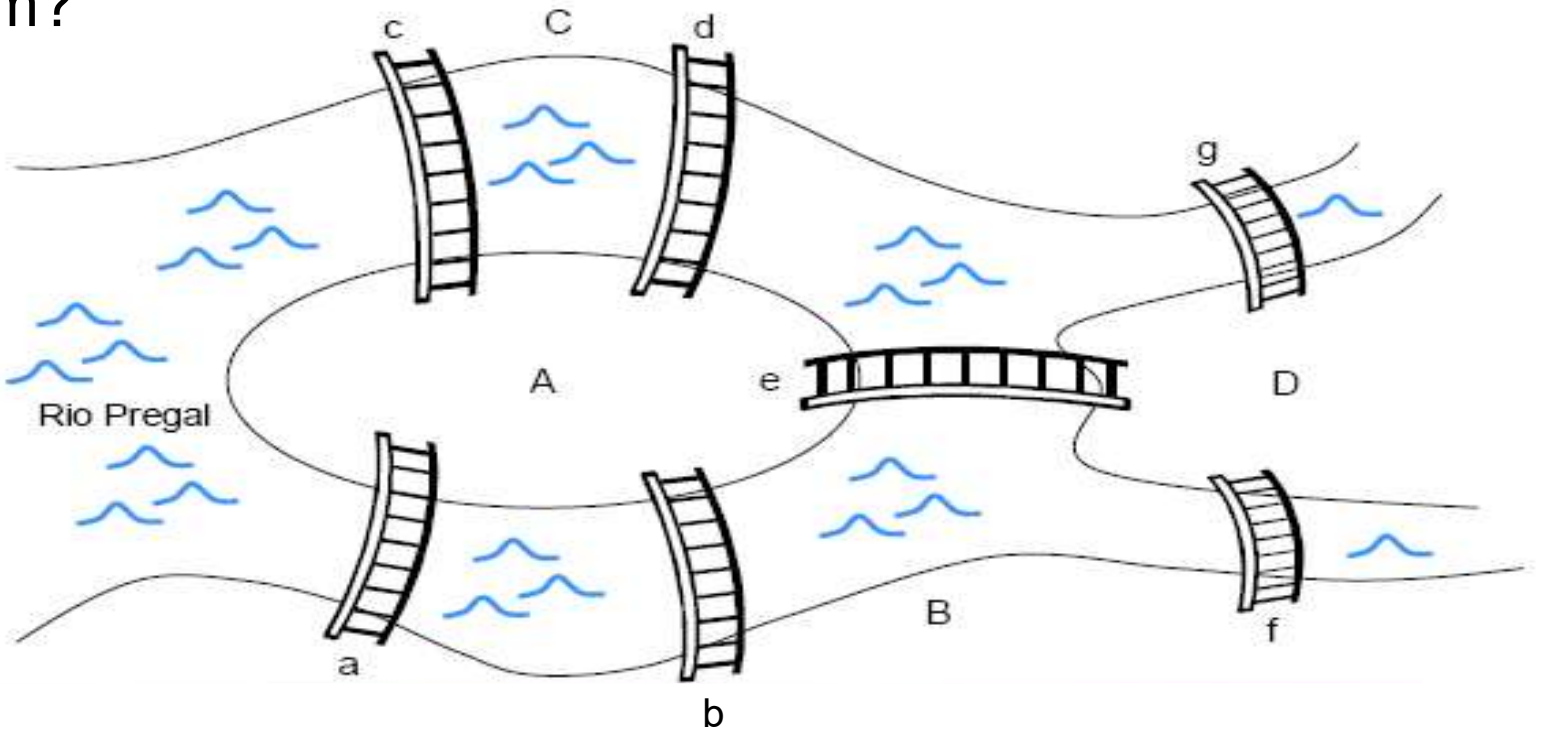
• Assim sendo, existem quatro áreas de terra que ladeiam o rio: as áreas de terra (A-D) estão interligadas por sete pontes (a-g);

• O problema das pontes de Koenigsberg consiste em determinar se, ao partir de alguma área de terra, é possível atravessar todos as pontes exatamente uma vez, para, em seguida, retornar à área de terra inicial.

UNIVASF

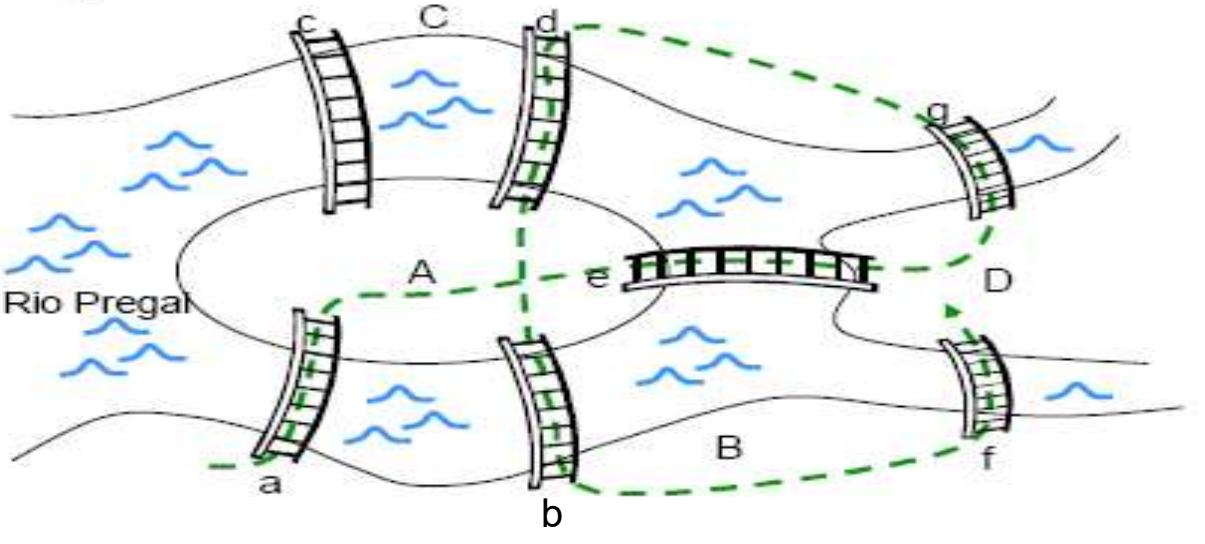
Grafos - Histórico

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



Grafos - 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.**



Grafos - Histórico

• Esse caminho não atravessa todas as pontes uma vez, nem tampouco retorna à área inicial de terra B.

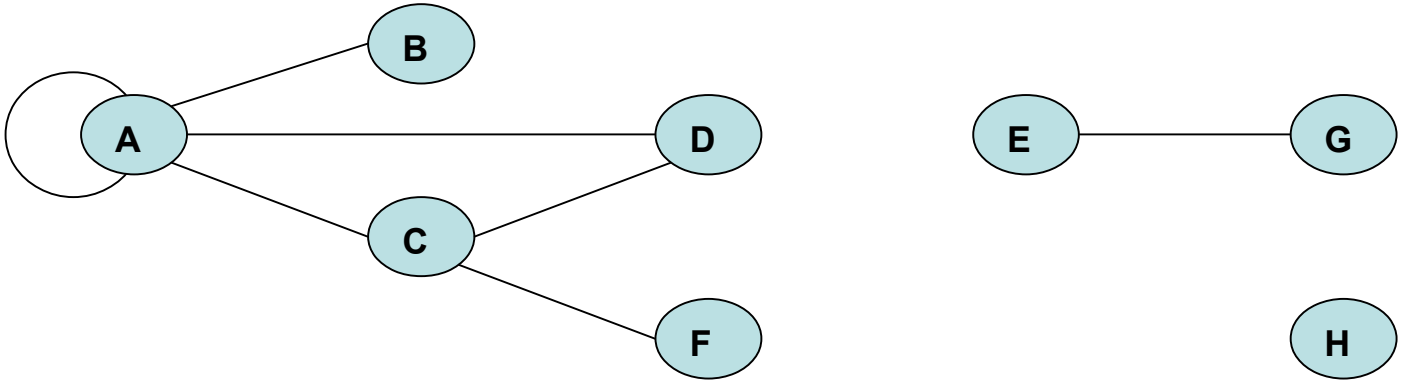
• Euler provou que não é possível o povo de Koenigsberg atravessar cada ponte exatamente uma vez, retornando ao ponto inicial.

• Ele resolveu o problema, representando as áreas de terra como vértices e as pontes como arestas de um grafo (na realidade, um multigrafo).

UNIVASF

Grafos - Definições

- Um grafo consiste num conjunto de nós (ou vértices) e num conjunto de arcos (ou arestas).
- Cada arco num grafo é especificado por um par de nós.

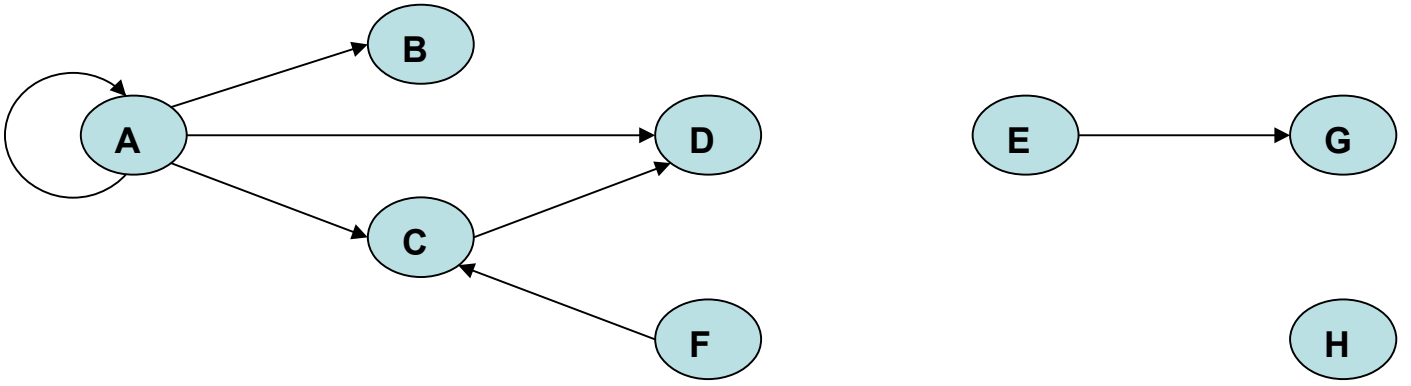


- A sequência de nós é {A,B,C,D,E,F,G,H},e o conjunto de arcos é {(A,B), (A,D), (A,C), (C,D), (C,F), (E,G), (A,A)}.

UNIVASF

Grafos - Definições

• Se os pares de nós que formam os arcos forem pares ordenados, diz-se que o grafo é um grafo orientado (ou dígrafo). Exemplos:



• As setas entre os nós representam arcos. A ponta de cada seta representa o segundo nó no par ordenado de nós que forma um arco, e o final de cada seta representa o primeiro nó no par. O conjunto de arcos do grafo acima é {<A,B>, <A,C>, <A,D>, <C,D>, <F,C>, <E,G>, <A,A>}.

Grafos - Definições

• Note que foram usados parênteses para indicar um par não-ordenado (desordenado) e chaves angulares para indicar um par ordenado.

• Concentraremos inicialmente nosso estudo nos dígrafos.

• Observe que, como vimos, um grafo não precisa ser uma árvore, mas uma árvore tem de ser um grafo. Note também que um nó não precisa ter arcos associados a ele (por exemplo, nó *H* no grafo anterior).

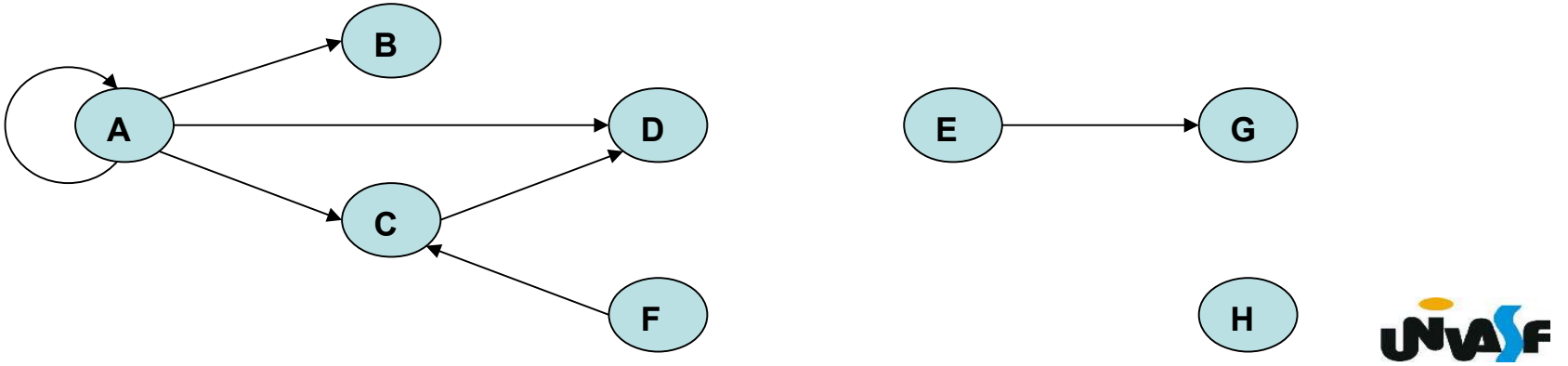
• Um nó *n* ***incide*** em um arco *x* se *n* for um de seus dois nós no par ordenado de nós que constituem *x*. (Dizemos também que *x* incide em *n*.)

Grafos - Definições

• O ***grau*** de um nó é o número de arcos incidentes nesse nó.

• O ***grau de entrada*** de um nó *n* é o número de arcos que têm *n* como cabeça, e o ***grau de saída*** de *n* é o número de arcos que têm *n* como terminação da seta.

Por exemplo, o nó C no grafo abaixo tem grau de entrada 2, grau de saída 1 e grau 3.



Grafos - Definições

• Um nó *n* será ***adjacente*** a um nó *m* se existir um arco de *m até n*. Se *n* for adjacente a *m*, *n* será chamado ***sucessor*** de *m* e *m* será um ***predecessor*** de *n*.

• Uma ***relação*** *R* num conjunto A é uma sequência de pares ordenados de elementos de A.

Por exemplo, se *A* = {3, 5, 6, 8, 10, 17}, o conjunto *R* = {<3,10>, <5,6>, <5,8>, <6,17>, <8,17>, <10,17>} será uma relação. Se <*x, y*> for um membro de uma relação *R*, diz-se que *x* está relacionado a *y* em *R*.

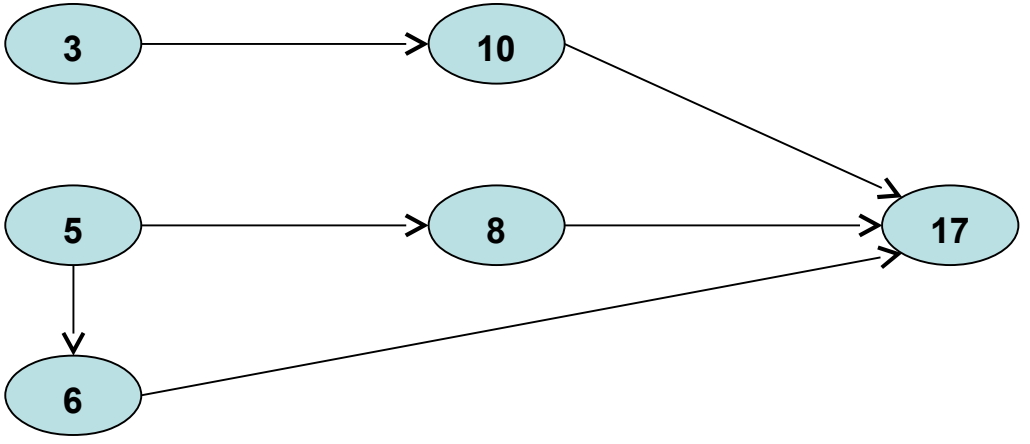
A relação *R* anterior pode ser descrita dizendo-se que *x* está relacionado com y se x for menor que *y e o* resto obtido a partir da divisão de *y* por *x* for ímpar.

UNIVASF

Grafos - Definições

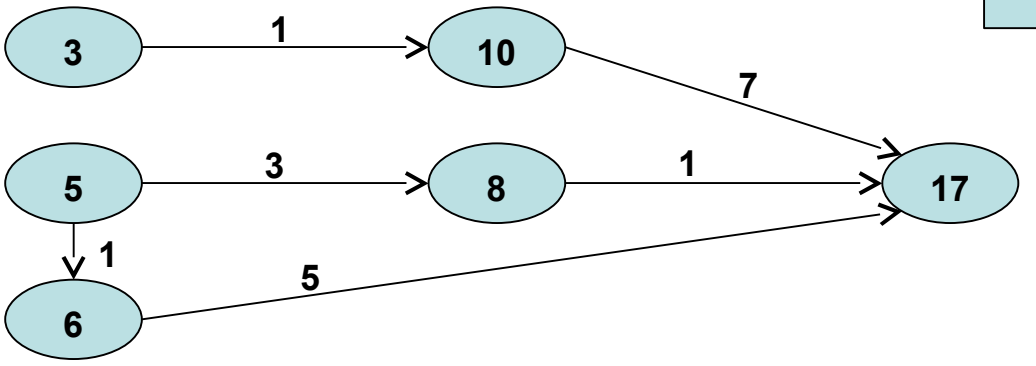
Uma relação pode ser representada por um grafo no qual os nós representam o conjunto básico e os arcos representam os pares ordenados da relação.

O grafo abaixo representa a relação anterior.



Grafos - Definições

Um número pode ser associado a cada arco de um grafo, como no grafo abaixo.



Neste grafo, o número associado a cada arco é o resto obtido da divisão do inteiro posicionado na cabeça do arco pelo inteiro posicionado em sua terminação. Um grafo desse tipo, no qual existe um número associado a cada arco, é chamado ***grafo ponderado*** ou ***rede.*** O número associado a um arco é chamado ***peso.***

UNIVASF

Grafos - Definições

Identificaremos várias operações primitivas que serão úteis ao lidar com grafos:

- A operação *ligar(a,b)* introduz um arco do nó *a* até o nó *b* se ainda não existir *um;*
- *ligarComPeso(a,b,x)* insere um arco de *a* até *b* com peso *x* num grafo ponderado;
- *remover(a,b)* e *removerComPeso (a,b,x)* eliminam um arco de *a* até *b*, caso exista *(removerComPeso* define também *x* com seu peso).

Embora possamos também acrescentar ou eliminar nós de um grafo, discutiremos essas possibilidades posteriormente.

- A função *adjacente(a,b)* retorna *true* se *b* for adjacente a *a*, e *false,* caso contrário.

UNIVASF

Grafos - Definições

Um ***caminho de comprimento k*** do nó *a* ao nó *b* é definido como uma sequência de *k + 1* nós *n1, n2,..., nk + 1,* tal que *n1 = a, nk + 1 = b* e *adjacente(ni, ni + 1)* é *true* para todo *i* entre *1* e *k*. Se, para algum inteiro *k*, existir um caminho de comprimento *k* entre *a* e *b,* existirá um ***caminho*** de *a* até *b.*

Um caminho de um nó para si mesmo é chamado ***ciclo***.

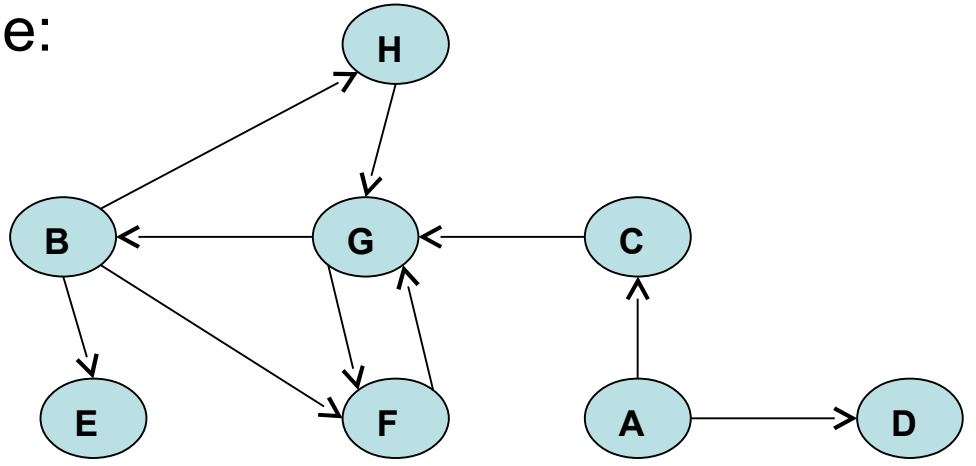
Se um grafo contiver um ciclo, ele será ***cíclico***; caso contrário, será ***acíclico***.

Um grafo acíclico orientado é chamado ***dag***, uma aglutinação das iniciais de *directed acyclic graph.*

UNIVASF

Grafos - Definições

Uma análise do grafo abaixo possibilita perceber a existência de:



- um caminho de comprimento 1 de A até C;
- dois caminhos de comprimento 2 de *B até G;*
- *e um caminho de* comprimento 3 de A até F.

Note que não existe um caminho de B até C.

Existem ciclos de B para B, de F para F e de H para H.

Grafos - Representação

Com base no que foi exposto e considerando que o número de nós no grafo seja constante, ou seja, os arcos podem ser acrescentados ou eliminados, mas os nós não.

Que estrutura poderia ser utilizada para sua representação?

Um vetor bidimensional.

Se considerarmos que os nós de um grafo serão numerados de 0 a *MAXNODES - 1* e nenhuma informação é atribuída (associada) a eles. Além disso, também consideraremos a existência de arcos sem a associação de pesos ou outras informações aos mesmos.

UNIVASF

Grafos - Representação

Exercício:

Nesse caso, defina como um grafo poderia ser declarado.

#define MAXNODES valorInteiroPositivo

int adj[MAXNODES][MAXNODES];

O valor de *adj[i][j]* será *TRUE* ou *FALSE,* dependendo de o nó *j* ser ou não adjacente ao nó *i.* O vetor bidimensional *adj[ ][ ]* é chamado ***matriz de adjacência.***

UNIVASF

Grafos - Representação

Exercício: Seguindo esta linha de raciocínio, defina uma representação mais completa para um grafo.

```
#define MAXNODES valorInteiroPositivo
struct node {
   /* informacao associada a cada noh */
};
struct arc {
   int adj;
   /* informacao associada a cada arco */
};
struct graph {
   struct node nodes[MAXNODES];
   struct arc arcs[MAXNODES][MAXNODES];
};
struct graph g;
```

UNIVASF

Grafos - Representação

Onde, assim como na representação anterior, cada nó do grafo é representado por um inteiro entre 0 e *MAXNODES* - 1, e o campo vetor *nodes representa as informações associadas a* cada nó. O campo vetor *arcs* é um vetor bidimensional representando todo possível par ordenado de nós. O valor de *g.arcs[i][j].adj é TRUE* ou *FALSE,* dependendo de o nó *j* ser ou não adjacente ao nó *i.* Neste caso, o vetor bidimensional *g.arcs[][].adj* é a matriz de adjacência. No caso de um grafo ponderado, cada arco poderá também receber a atribuição de informações.

UNIVASF

Grafos - Representação

Exercício: Considerando a representação de um grafo apenas pela matriz de adjacência, ou seja, onde não serão atribuídos pesos e outras informação à arestas e nem informações ao nós. Implemente a operação *ligar(a,b)*.

```
void ligar (int adj[][MAXNODES], int node1, int node2)
{
   adj[node1][node2] = 1;
}
```

Grafos - Representação

Exercício: Considerando a representação de um grafo em questão. Implemente a operação *remover (a,b)*.

```
void remover (int adj[][MAXNODES], int node1, int node2)
{
   adj[node1][node2] = 0;
}
```

Grafos - Representação

Exercício: Considerando a representação de um grafo em questão. Implemente a operação *adjacente (a,b)*.

```
int adjacente (int adj[][MAXNODES], int node1, int node2)
{
   return adj[node1][node2];
}
```

Grafos - Representação

Exercício: Proponha uma estrutura capaz de representar um grafo ponderado com um número fixo de nós.

```
#define MAXNODES valorInteiroPositivo
struct arc {
   int adj;
   int peso;
};
struct arc g[MAXNODES][MAXNODES];
```

Grafos - Representação

Exercício: Considerando a representação para um grafo ponderado apresentada, implemente a operação *ligarP(...)*.

```
void ligarP (struct arc adj[][MAXNODES],
int node1, int node2, int peso)
{
   adj[node1][node2].adj = 1;
   adj[node1][node2].peso = peso;
}
```

UNIVASF

Grafos - Representação

Exercício: Considerando a representação para um grafo ponderado apresentada, implemente a operação *removerP(...)*.

```
void removerP (struct arc adj[][MAXNODES],
int node1, int node2, int peso)
{
   adj[node1][node2].adj = 0;
   adj[node1][node2].peso = peso;
}
```

UNIVASF

Grafos - Representação

Exercício: Considerando a representação para um grafo ponderado apresentada, implemente a operação *adjacenteP(...)*.

```
int adjacenteP (struct arc adj[][MAXNODES],
int node1, int node2)
{
   return adj[node1][node2].adj;
}
```

Grafos - Aplicação

Examinaremos agora um exemplo de aplicação de um grafo. Vamos supor o seguinte problema: Considerando a existência de **n** cidades. Considerando que alguns pares destas cidades possuem estradas que as ligam. Determine se é possível sair de uma cidade **A** e chegar em uma cidade **B** utilizando exatamente **nr** estradas.

Determine uma estratégia para solucionar o problema apresentado.

UNIVASF

Grafos - Aplicação

Uma estratégia para a solução é a seguinte: crie um grafo com as cidades como nós e as estradas como arcos. Para achar um caminho de comprimento *nr* do nó *A* ao nó *B,* procure um nó *C* de modo que exista um arco de *A* até *C* e um caminho de comprimento *nr - 1* de *C* até *B.* Se essas condições forem atendidas para um nó *C*, o caminho desejado existirá; se elas não forem atendidas para qualquer nó *C*, o caminho não existirá.

O algoritmo usará uma função recursiva auxiliar, *procurarCaminho(k,a,b)*. Essa função retornará *true* se existir um segmento de comprimento *k* de *A* até *B,* e *false*, caso contrário.

UNIVASF

Grafos - Aplicação

Implemente na linguagem C a função procurarCaminho(...).

Grafos - Aplicação

Visando possibilitar o teste da função apresentada implemente, na linguagem C, uma função *main()*, que receba uma linha de entrada contendo quatro inteiros seguidos por um número qualquer de linhas de entrada com dois inteiros cada uma. O primeiro inteiro na primeira linha, *n,* representa um número de cidades, que, para simplificar, serão numeradas de 0 a *n* - 1*. O* segundo e o terceiro inteiro nessa linha estão entre 0 e *n* - 1 e representam duas cidades. Queremos sair da primeira cidade para a segunda usando exatamente *nr* estradas, onde *nr* é o quarto inteiro na primeira linha de entrada. Cada linha de entrada subsequente contém dois inteiros representando duas cidades, indicando que existe uma estrada da primeira cidade até a segunda. A última linha na sequência conterá dois valores inteiros negativos. O problema é determinar se existe um percurso do tamanho solicitado pelo qual se possa viajar da primeira cidade para a segunda.

UNIVASF