Aula 02 – Conceitos Básicos

Notas de Aula de Teoria dos Grafos

Prof$^{a.}$: Patrícia D. L. Machado

UFCG – Unidade Acadêmica de Sistemas e Computação

Grafo Não-Direcionado

Definição 01. um grafo é um par ordenado (*V(G), E(G)*), onde:

- *V(G)* é um conjunto de **vértices**;
- *E(G)* é um conjunto de **arestas**, juntamente com uma **função de incidência** ψ.
- Se *e* é uma aresta e *u* e *v* são vértices, $\psi(e) = \{u,v\} = uv = vu$
- *u e v* são **adjacentes** ou vizinhos e *e* é **incidente** a *u* e *v*.
- *u* e *v* são os **terminais** de *e.*

Observe que podemos representar o resultado da função de incidência usando a notação de conjunto de vértices *{u,v}* ou a concatenação dos nomes dos vértices em qualquer ordem: *uv* ou *vu.* Como este relacionamento não possui ordem, os nomes dos vértices podem aparecer em qualquer ordem. Por este motivo, G é denominado de grafo não-direcionado (*undirected graph*)

Por exemplo, considere o grafo G_1 abaixo com seu conjunto de vértices, arestas e função de incidência. A aresta *s*, segundo a função de incidência, relaciona os vértices *a* e *b*, indicando que são adjacentes através da mesma.

$G_1 = (V(G_1), E(G_1))$:

$V(G_1) = \{a, b, c, d, e\}$
$E(G_1) = \{s, t, u, v, w, x, y, z\}$

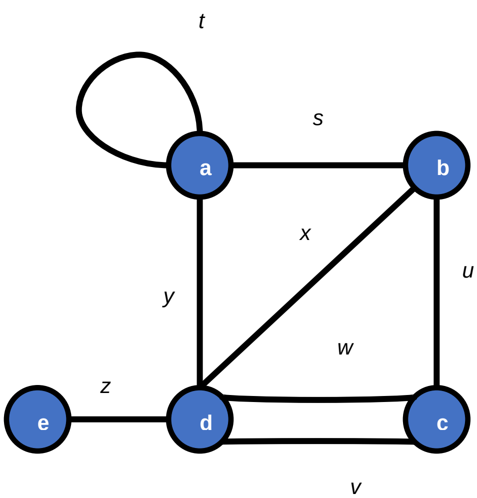
ψ_{G1} é definido como:
$\psi_{G1}(s)=ab$, $\psi_{G1}(t)=aa$, $\psi_{G1}(u)=bc$, $\psi_{G1}(v)=cd$,
$\psi_{G1}(x)=bd$, $\psi_{G1}(w)=cd$, $\psi_{G1}(y)=ad$, $\psi_{G1}(z)=de$

O número de vértices e arestas de um grafo *G* é dado por *v(G)* e *e(G)* respectivamente (**ordem** e **tamanho** de G). Em nosso exemplo, a ordem de G_1 é 5 e o tamanho de G_1 é 8.

Grafos podem ser representados por diagramas ou gráficos onde vértices são representados por pontos e arestas são representadas por linhas. No entanto, o diagrama serve unicamente para visualização humana, quando possível. Grafos reais podem ser imensos e desta forma é difícil ou impossível representá-los em uma forma gráfica detalhada. A fim de serem processados de forma automática, grafos precisam ser definidos usando uma notação formal, tal como a apresentada na Definição 01, ou uma estrutura de dados em uma linguagem de programação, dentre outras.

A partir de uma definição formal, diferentes representações gráficas podem ser construídas. A forma de representar graficamente é escolhida com base no tipo de informação que queremos destacar.

Abaixo, temos uma representação gráfica do grafo G_1 apresentado como exemplo. Utilizaremos este grafo para ilustrar demais conceitos apresentados nesta aula.



Como outro exemplo, considere o problema de representar as diferentes formas de viajar de um estado para outro na região nordeste atravessando um ou mais estados. Podemos definir um grafo onde os vértices são os estados e as arestas indicam quando 2 estados são vizinhos. Desta forma, podemos investigar, por exemplo, como viajar de uma cidade para outra cruzando o menor número possível de estados.

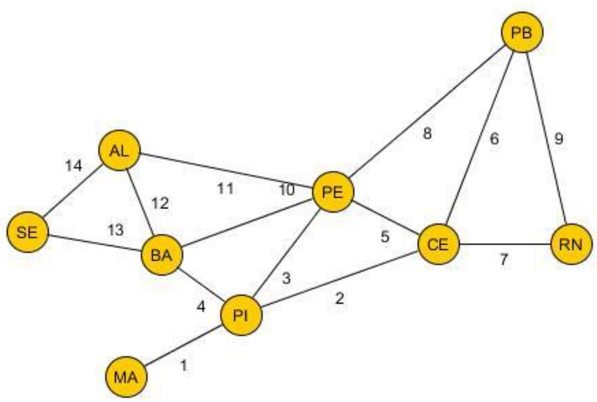
Abaixo apresentamos um grafo F que modela este problema. Note que F possui 14 arestas representado a existência de 14 vizinhanças diferentes. As vizinhanças são descritas através da função de incidência.

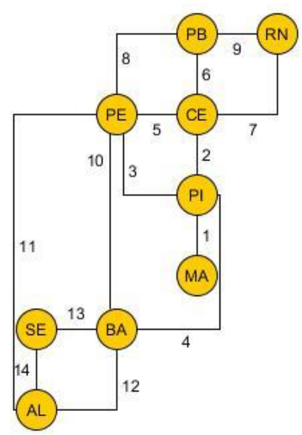
F = (V(F), E(F)):

V(F) = {MA, PI, CE, RN, PB, PE, AL, SE, BA}
E(F) = {1,2,3,4,5,6,7,8,9,10,11,12,13,14}

$\Psi_F(1)$= {MA,PI}; $\psi_F(2)$ = {PI,CE}; $\psi_F(3)$ = {PI,PE}; $\psi_F(4)$ = {PI,BA};
$\psi_F(5)$ = {CE,PE}; $\psi_F(6)$ = {CE,PB}; $\psi_F(7)$ = {CE,RN}; $\psi_F(8)$ = {PE,PB};
$\psi_F(9)$ = {RN,PB}; $\psi_F(10)$ = {PE,BA}; $\psi_F(11)$ = {PE,AL}; $\psi_F(12)$ = {BA,AL};
$\psi_F(13)$ = {BA,SE}; $\psi_F(14)$ = {AL,SE};

Abaixo, temos duas visualizações gráficas diferentes para o grafo F. A visualização a esquerda mostra de forma mais clara os estados que possuem uma posição mais ou menos central com relação aos demais. Por exemplo, é possível perceber através deste grafo que PE é o estado que possui mais vizinhos enquanto MA é o estado que possui menos vizinhos. Isto pode ser visto através da contagem do número de vezes em que PE e MA é terminal de uma aresta. A representação gráfica da esquerda permite que esta informação seja compreendida de forma mais imediata. Outro fato importante que podemos observar neste grafo é que todo caminho para o MA passa obrigatoriamente por PI. Assim o PI é um estado de grande influência na região.





Em aulas posteriores, estudaremos mais sobre como analisar e obter informações importantes em grafos e suas visualizações.

Definições Básicas

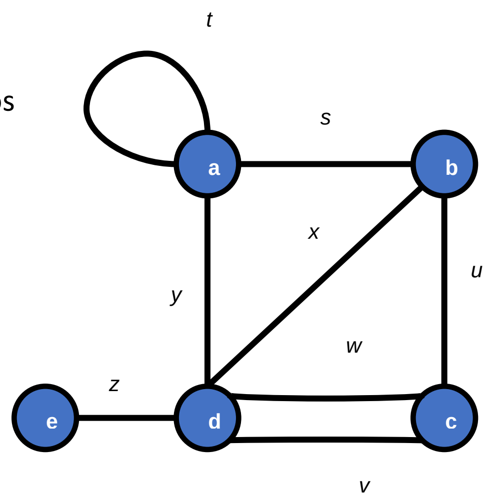
A seguir, apresentamos alguns conceitos básicos de grafos. Para exemplificar estes conceitos, utilizamos o grafo G_1 definido anteriormente.

O **conjunto de vizinhos** de um vértice *v* em um grafo *G* é definido por $N_G(v)$. Em G_1, o conjunto de vizinhos de *d* é {*e, a ,c, b*} e o conjunto de vizinhos de *a* é {*a, b, d*}.

Uma aresta com terminais:

- Idênticos é chamada de **loop**
- Distintos é chamada de **link**.

Em G_1, *t* é um loop e *s* é um link.



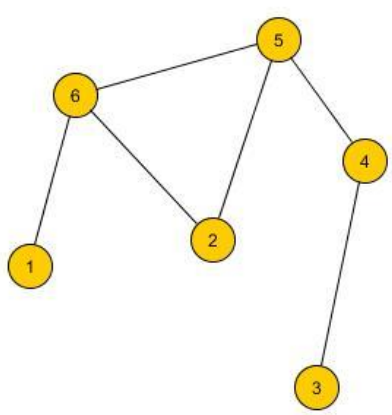
Quando 2 ou mais links possuem o mesmo par de terminais estes são chamados de **arestas paralelas**. Um grafo com arestas paralelas é chamado de **multigrafo.** Alguns autores admitem que multigrafos podem conter loops. Outros denominam um grafo com loops de **pseudografo**. Pseudografos podem conter arestas paralelas. O grafo exemplo G_1 é um pseudografo e pode ser considerado um multigrafo por alguns autores.

Um grafo é **finito** se seus conjuntos de vértices e arestas são finitos.

O grafo sem vértices ($V(G) = \emptyset$) é chamado de **grafo nulo**.

Qualquer grafo com apenas um vértice é chamado de **trivial**. Todos os outros grafos são grafos **não-triviais.**

Um grafo é **simples** se não possui loops ou arestas paralelas. Neste caso, a função de incidência pode ser ignorada porque podemos representar cada aresta por seus terminais, sem ambiguidade. Abaixo temos um exemplo de um grafo simples e sua definição formal, onde a função de incidência pode ser omitida.



$G_2 = (V(G_2), E(G_2))$, onde:

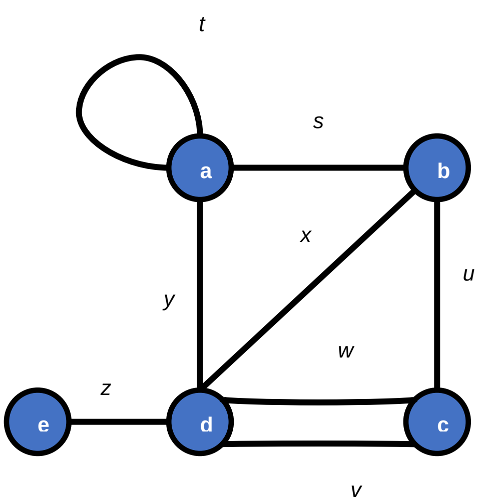
$V(G_2) = \{1,2,3,4,5,6\}$
$E(G_2) = \{16, 65, 62, 52, 54, 34\}$

Representações Matriciais

Grafos podem ser representados através de matrizes. Seja G um grafo, onde *V(G)* é o conjunto de vértices e *E(G)* é o conjunto de arestas. Seja n o número de vértices de G e m o número de arestas de G.

A **matriz de adjacência** de G é uma matriz $n \times n$ onde cada elemento i_{uv} representa o número de arestas que unem os vértices u e v, onde cada loop conta como duas arestas (o vértice associado é terminal 2 vezes do loop). Observe que, para grafos não-direcionados, a matriz de adjacências é simétrica, ou seja, o valor do elemento i_{uv} é igual ao valor do elemento i_{vu}.

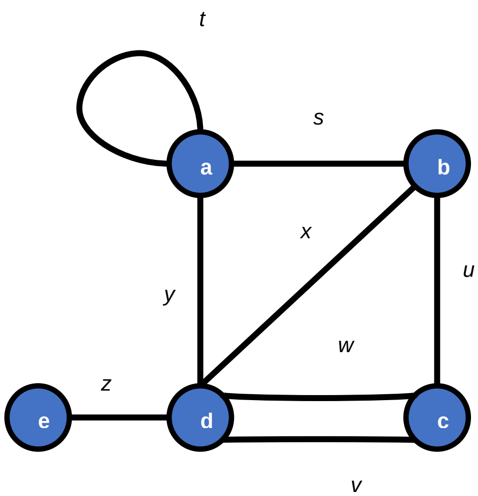
Como exemplo, abaixo temos a matriz de adjacência para o grafo G_1, onde o valor de i_{aa} é 2 visto que o vértice a é terminal do loop t. Já os elementos i_{ab} e i_{ba} possuem o valor 1, indicando a existência de um link entre os vértices a e b. Os elementos i_{ac} e i_{ca} possuem valor 0, indicando a inexistência de qualquer aresta entre os vértices a e c. No caso dos vértices c e d, o valor associado na matriz é 2, tanto para o elemento i_{cd} quanto i_{dc}. Este valor indica a existência de duas arestas paralelas entre estes vértices.



	a	**b**	**c**	**d**	**e**
a	2	1	0	1	0
b	1	0	1	1	0
c	0	1	0	2	0
d	1	1	2	0	1
e	0	0	0	1	0

Uma outra forma de representação matricial é a **matriz de incidência**. Diferentemente da matriz de adjacência que relaciona vértices com vértices através do número de arestas comuns em que são terminais, a matriz de incidência relaciona vértices com arestas. Formalmente, a **matriz de incidência** é uma matriz $n \times m$ onde cada elemento $\boldsymbol{i_{ve}}$ representa o número de vezes que cada vértice v e aresta e são incidentes. Este número pode ser 0 no caso de não serem incidentes, 1 no caso de serem incidentes e a aresta ser um link e 2 no caso de serem incidentes e a aresta ser um loop.

Como exemplo, temos abaixo a matriz de incidência para o grafo G_1. Observe que o valor de i_{at} é 2, visto que este vértice é terminal do loop t. Já o elemento i_{as} possui o valor 1, indicando que *a* é um dos terminais de *s*. O elemento i_{au} possui valor 0, indicando que *a* não é terminal de *u*.



	s	t	u	v	w	x	y	z
a	1	2	0	0	0	0	1	0
b	1	0	1	0	0	1	0	0
c	0	0	1	1	1	0	0	0
d	0	0	0	1	1	1	1	1
e	0	0	0	0	0	0	0	1

Grau

O **grau** de um vértice *v* em um grafo *G*, representado por $d_G(v)$, é o número de arestas de *G* que são incidentes ao vértice *v*, onde cada loop conta como duas arestas.

Se *G* é um grafo simples, então o grau de *v* pode ser determinado contando o número de vizinhos de *v* em *G*. Neste grafo, não temos loops nem arestas paralelas.

$\Delta(G)$ e $\delta(G)$ representam o grau **máximo** e **mínimo** de vértices de *G* respectivamente. Ou seja, dentre os graus de todos os vértices de G, $\Delta(G)$ é o valor do maior deles *e* $\delta(G)$ é o valor do menor deles.

Por fim, um vértice de grau 0 é chamado de **vértice isolado**. Este vértice não se relaciona com qualquer outro vértice do grafo.

No grafo G_1, vemos que o grau do vértice *b* é 3, porque 3 arestas são incidentes a ele. Já o grau do vértice *a* é 4, já que 3 arestas são incidentes a ele, mas uma destas arestas, a aresta *t,* é uma aresta loop, portanto conta 2. G_1 não possui vértices isolados. Observe também que $\Delta(G_1) = 5$ e $\delta(G_1) = 1$

O **grau médio** de um grafo G, representando por $d(G) = \frac{1}{n}\sum_{v \in V(G)} d(v)$, é a média aritmética dos graus de todos os vértices do grafo.

Teorema 1. *(Handshaking Lemma)*. Para todo grafo *G*, $\sum_{v \in V(G)} d(v) = 2m$, onde *m* é a quantidade de arestas de G.

O Teorema 1 estabelece que para todo grafo *G* a soma dos graus de todos os vértices de G é igual a 2 vezes a quantidade de arestas representada por *m*. A prova deste teorema se baseia no fato de que, na matriz de incidência, a soma dos elementos na linha correspondente ao vértice *v* é exatamente *d(v),* porque cada elemento representa a quantidade de vezes em que um vértice é terminal de uma aresta. Assim, a soma de todos os graus dos vértices de *G* é exatamente a soma de todos os elementos da matriz. Note que esta soma é também exatamente igual a *2* vezes a quantidade de arestas, pois a soma de cada coluna da matriz é exatamente igual a 2, já que cada aresta tem exatamente 2 terminais.

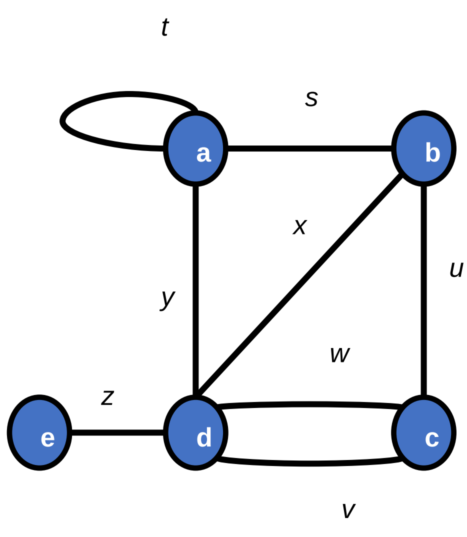
Como exemplo, observe a matriz de incidência para o grafo G_1 apresentada anteriormente. A soma dos valores na linha a é 4 e d(a) = 4. A soma dos valores em cada coluna é 2.

Corolário 1. Em qualquer grafo, o número de vértices de grau ímpar é par.

De maneira informal, podemos observar que esta propriedade expressa no Corolário 1 é válida visto que cada aresta contribui com 2 no somatório geral de graus de vértices. Nos exemplos abaixo, observe que os grafos possuem um número par de vértices de grau ímpar. Para qualquer grafo com 2 ou mais vértices de grau ímpar, se adicionamos uma aresta entre dois vértices de grau ímpar, ambos passam a ter grau par e desta forma mantemos a propriedade. Em nosso exemplo da esquerda, considere adicionar uma aresta entre os vértices *b* e *e*.

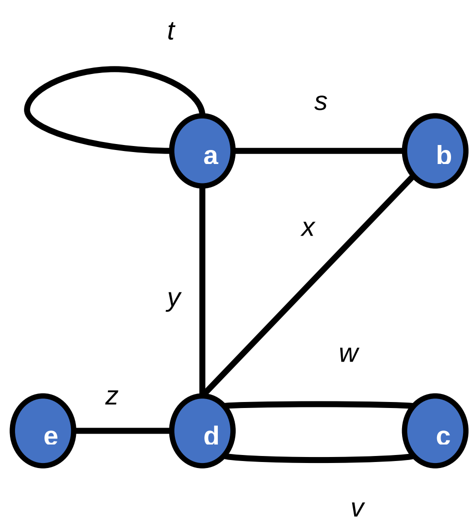
Para qualquer grafo com 2 ou mais vértices de grau par, se adicionarmos uma aresta entre vértices de grau par, ambas passam a ter grau ímpar, mantendo a propriedade. Em nosso exemplo da direita, considere adicionar uma aresta entre os vértices *b* e *c*.

Já, se adicionamos uma aresta entre um vértice de grau par e outro de grau ímpar, o vértice de grau par passará a ter grau ímpar e o de grau ímpar passará a ter grau par, mantendo a propriedade. Em nosso exemplo da direita, considere adicionar uma aresta entre os vértices *e* e *c*.



Grau par: a

Grau ímpar: b,c,d,e



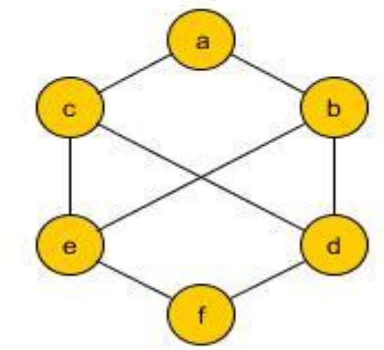
Grau par: a,b,c

Grau ímpar: d,e

Representações Computacionais

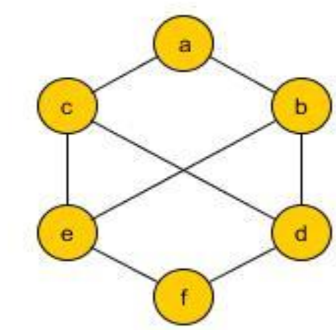
Uma forma de representação computacional de grafos simples bastante utilizada é a **lista de adjacência**. Nela, cada vértice está associado a uma coleção contendo todos os seus vizinhos, conforme podemos observar em nosso exemplo abaixo. O conjunto de arestas do grafo pode ser obtido a partir desta lista, criando uma aresta para cada par de um vértice com um dos seus vizinhos. Esta forma de representação é adequada a grafos simples e pode ser utilizada para representar grafos com *loops*, mas não é adequada para representar arestas paralelas, visto que não é possível nomear as arestas.

a → b,c;
b → a,d,e;
c → a,d,e;
d → c,b,f;
e → b,c,f;
f → d,e



Outra forma de representação computacional para grafos simples é a **lista de arestas**, onde o grafo é representado unicamente através da listagem de todas as suas arestas, como podemos observar no exemplo abaixo. Usando esta representação, o conjunto de vértices pode ser obtido, observando-se todos os vértices que são terminais de alguma aresta. Da mesma forma que para a lista de adjacência, esta forma também pode ser utilizada para representar grafos com *loops*, mas não é adequada para representar arestas paralelas, visto que não é possível nomear as arestas.

b,a
a,c
c,d
c,e
d,b
d,f
f,e
e,b

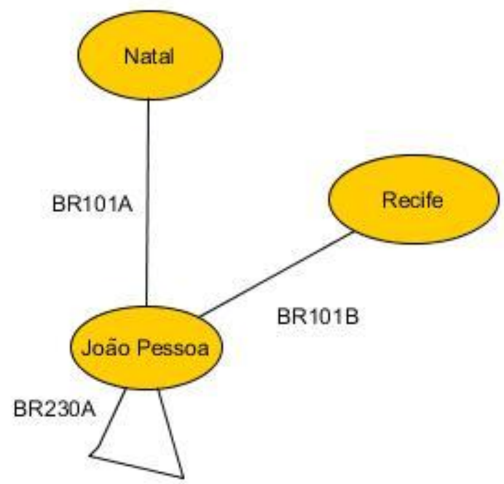


Trivial Graph Format ou **TGF** é um formato computacional que pode ser utilizado para grafos com vértices e arestas rotulados. Neste formato, os vértices são definidos em cada linha de um arquivo texto através de dois campos separados por espaço em branco como podemos observar no exemplo abaixo. O primeiro é um identificador do vértice e o segundo é um *label* ou rótulo que normalmente é utilizado para ser apresentado na representação gráfica ou possui uma descrição mais detalhada do elemento representado pelo vértice. O identificador é utilizado para

referenciar os vértices de maneira mais simples do que o rótulo que pode conter informação mais detalhada ou complexa.
Após os vértices, o caracter “#” é adicionado e, logo em seguida, as arestas são listadas, uma em cada linha. Cada aresta é definida por 3 campos onde os dois primeiros representam os identificadores dos vértices e o terceiro é um *label* ou rótulo para a aresta.

No exemplo abaixo, temos um código TGF para o grafo que está representado graficamente a direita. Note que os identificadores não são exibidos nesta representação gráfica visto que os rótulos apresentam informações mais significativas sobre o que queremos apresentar.

```
1 Natal
2 João Pessoa
3 Recife
#
2 3 BR101B
1 2 BR101A
2 2 BR230A
```



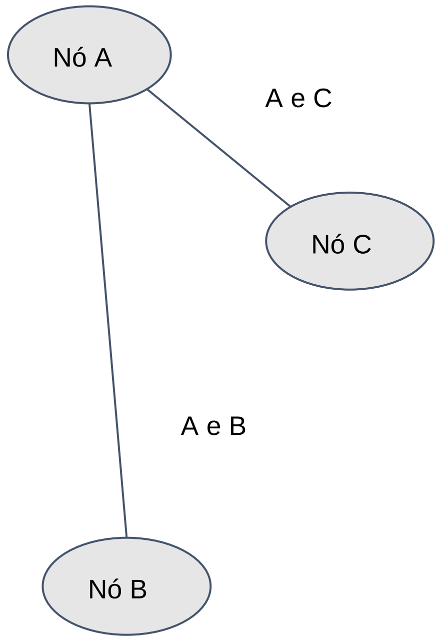
Grafos reais possuem mais informações associadas aos vértices e arestas do que um identificador e um rótulo. Assim, formatos mais avançados foram propostos a fim de que possamos representar conjuntos de atributos para vértices e arestas, incluindo atributos relacionados ao formato de visualização, tal como as coordenadas na tela, a cor, dentre outros.

O formato ***Graph Modelling Language*** ou **GML**, também conhecido como *Graph Meta Language* é um formato hierárquico, onde vértices são definidos através de blocos denominados de *node*. Estes blocos são delimitados por []. No exemplo abaixo, podemos observar o primeiro *node* que define o vértice com identificador A cujo label é “Nó A”. Arestas são representadas através dos blocos *edge*. No primeiro, temos uma aresta entre os vértices B e A com *label* “A e B”.

Para grafos não-direcionados, os terminais de uma aresta podem ser definidos tanto na *tag source* quanto *target*. Para grafos direcionados, os quais estudaremos em aulas mais adiante, a aresta representa um relacionamento que vai de *source* para *target*, ou seja, em uma única direção. Em nosso exemplo, a *tag directed*, destacada em laranja, está definida com o valor 0, indicando que este grafo é não-direcionado.

O formato GML é um formato usualmente utilizado por ferramentas para exportar um grafo com atributos em vértices e arestas e informações sobre uma certa visualização gráfica definida. Os formatos que vimos anteriormente não registram informações sobre uma representação gráfica, nem tão pouco atributos com informações detalhadas sobre vértices e arestas.

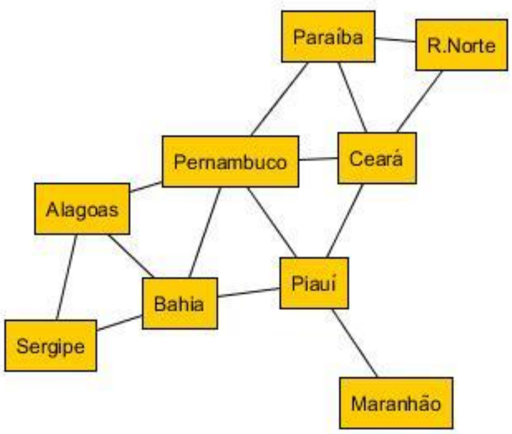
```
graph [
  directed 0
  node [
        id A
        label "Nó A" ]
  node [
        id B
        label "Nó B" ]
  node [
        id C
        label "Nó C" ]
   edge [
        source B
        target A
        label “A e B" ]
  edge [
        source C
        target A
        label “A e C" ]
]
```



GraphML é um formato análogo ao GML. Mas este é baseado no padrão XML e tem sido mais largamente utilizado na comunidade, podendo conter atributos *default* e definir uma ampla variedade de grafos incluindo grafos hierárquicos. O formato GML é um predecessor do GraphML. O formato **graphmlz** é o formato graphml compactado (zip). Abaixo temos um exemplo de um grafo no formato GraphML.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="undirected">
    <node id="n0"/>
    <node id="n1"/>
    <edge id="e1" source="n0" target="n1"/>
  </graph>
</graphml>
```

Por fim, grafos também são comumente representados através de **tabelas** criadas em base de dados ou em planilhas eletrônicas. Neste caso, usualmente temos uma tabela para a lista de vértices e uma tabela para a lista de arestas como podemos observar no exemplo abaixo. Cada coluna de cada tabela representa os atributos de um vértice ou aresta específico. Alguns atributos podem ser padrão, tais como o ID e *label* para vértices e *o id, source* e *target* para arestas. Mas, no geral, qualquer tabela de dados pode ser utilizada para representação de grafos desde que as informações essenciais sejam identificadas.



Edge List				Node List	
ID	**Source**	**Target**		**ID**	**Label**
1	MA	PI		MA	Maranhão
2	PI	CE		PI	Piauí
3	PI	PE		CE	Ceará
4	PI	BA		RN	R.Norte
5	CE	PE		PB	Paraíba
6	CE	PB		PE	Pernambuco
7	CE	RN		AL	Alagoas
8	PE	PB		SE	Sergipe
9	RN	PB		BA	Bahia
10	PE	BA			
11	PE	AL			
12	BA	AL			
13	BA	SE			
14	AL	SE			

As tabelas podem ser criadas em ferramentas como *Excel* ou *Google Sheets* e armazenadas ou exportadas no formato .xlsx que é usualmente lido por ferramentas que manipulam grafos. Mas também são usualmente armazenados no formato .csv (onde os campos são separados por vírgulas). Em nosso exemplo, o conjunto de vértices pode ser armazenado em um arquivo .csv com este conteúdo, onde na primeira linha temos os identificadores dos atributos dos vértices.

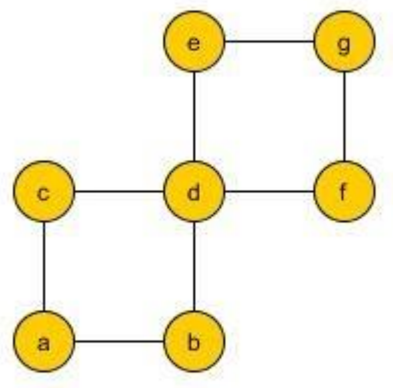
```
ID,Label
MA,Maranhão
PI,Piauí
CE,Ceará
RN,R.Norte
PB,Paraíba
PE,Pernambuco
AL,Alagoas
SE,Sergipe
BA,Bahia
```

Exercícios Propostos

1) Construa a lista completa de todos os grafos simples que tenham {a,b,c} por seu conjunto de vértices. (Original: https://www.ime.usp.br/~pf/grafos-exercicios/texto/ETG.pdf)

2) Para o grafo abaixo, construa representações matriciais e computacionais no formato:

a. Matriz de adjacência
b. Matriz de incidência
c. Lista de adjacência
d. Lista de arestas
e. TGF
f. Tabela



3) Dados números inteiros p e q, seja V o conjunto {1,2,3,...,pq-2,pq-1,pq}. Digamos que 2 elementos k e k' de V, com k < k', são adjacentes se k'=k+q ou (k mod q <> 0 e k' = k+1). Esta relação de adjacência define um grafo com conjunto de vértices V. Apresente a definição formal deste grafo considerando que p = 3 e q = 4. (mod denota o resto da divisão e <> a diferença entre 2 números)
(Original: https://www.ime.usp.br/~pf/grafos-exercicios/texto/ETG.pdf)

4) Um grafo de palavras é definido assim: cada vértice é uma palavra e dois vértices são adjacentes se as palavras diferem unicamente na letra de uma posição. Por exemplo, rato e ralo são adjacentes, enquanto que ralo e rota não são. Se o conjunto de vértices é definido pelas palavras a seguir, qual o conjunto de arestas? (caiado, cavado, cavalo, girafa, ralo, ramo, rata, rato, remo, reta, reto, rota, vaiado, varado, virada, virado, virava)
(Original: https://www.ime.usp.br/~pf/grafos-exercicios/texto/ETG.pdf)

5) Seja G um grafo tal que e(G) > v(G). Mostre que Δ(G) ≥ 3, através de um exemplo. (e(G) é a quantidade de arestas e v(G) é a quantidade de vértices).

6) Suponha que um grafo G tem menos arestas que vértices, ou seja, que e(G) < v(G). Mostre que G tem (pelo menos) um vértice de grau 0 ou (pelo menos) dois vértices de grau 1. Pode usar exemplos.

Referências

- J. A. Bondy and U. S. R. Murty. Graph Theory. Springer, 2008,2010. (Seção 1.1, exceto Special Families of Graphs)
- https://en.wikipedia.org/wiki/Trivial_Graph_Format
- https://en.wikipedia.org/wiki/Graph_Modelling_Language
- http://graphml.graphdrawing.org/index.html