# Aula 14 – Árvore

**Notas de Aula de Teoria dos Grafos**

Prof[a.]: Patrícia D. L. Machado

UFCG – Unidade Acadêmica de Sistemas e Computação

## Sumário

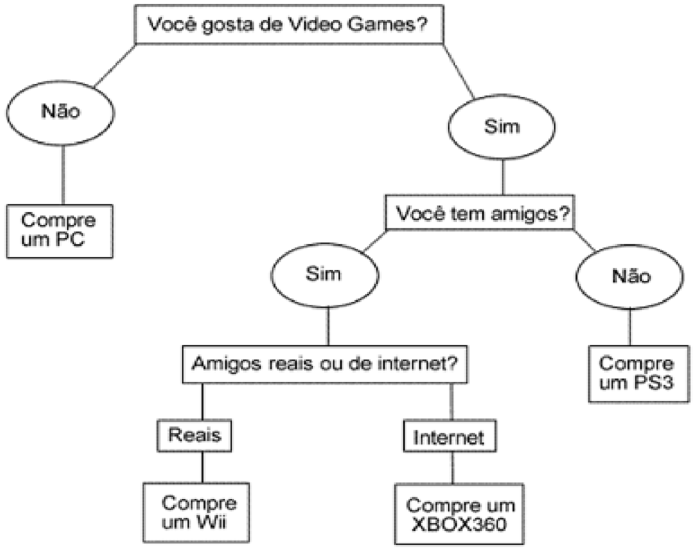


Nesta aula, o conceito de árvore é introduzido e ilustrado através de exemplos e aplicações.

## Conceito

Árvores são tipos especiais de grafos aplicados amplamente em ciência da computação para representar situações ou problemas onde o relacionamento entre os elementos envolvidos é determinado de forma não-ambígua.

Considere por exemplo o problema de implementar um programa que decida que dispositivo de vídeo game deve ser adquirido com base em informações sobre certas preferências do comprador. Abaixo apresentamos um grafo que pode representar a estrutura de decisão deste programa, uma árvore de decisão.



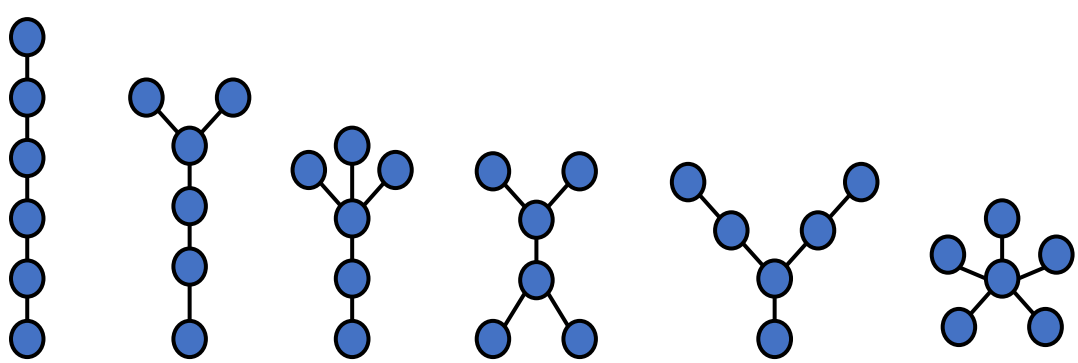
[1]

Observe os possíveis caminhos entre a pergunta central “*você gosta de vídeo games*?” até cada um dos vértices que representa um dispositivo. Note que existe exatamente um caminho entre eles. Desta forma, garantimos que não há ambiguidade na representação das escolhas. De forma geral, observamos que, neste grafo, entre quaisquer pares de vértices existe exatamente um caminho simples. Isto se deve ao fato de que o grafo não possui ciclos, o que daria a possibilidades de termos 2 ou mais caminhos. Além do mais, o grafo é conectado, garantido que sempre existe um caminho entre 2 pares de vértices. Como veremos a seguir, estas são as propriedades básicas de uma árvore.

## Definição

Formalmente, uma **árvore (*tree*)** é um *grafo acíclico* (sem ciclos incorporados) e *conectado*.

A seguir, vemos alguns exemplos de árvores. Note que, como toda árvore é um grafo conectado, então toda árvore possui um único componente.



[1] Figura de http://www.colmeia.tv/blog/2008/03/26/arvore-de-decisao-para-gamers/

Árvores são essencialmente grafos não-direcionados.

Nem todo grafo acíclico é uma árvore, porque alguns deles são desconectados. Mas todo componente de um grafo acíclico é uma árvore. Portanto, grafos acíclicos são usualmente chamados de **florestas (*forest*).**

Se considerarmos que todas as árvores de nosso exemplo acima juntas formam um único grafo, então este grafo é uma floresta formada por 6 componentes, onde cada um é uma árvore. Observe também que toda árvore é uma floresta formada por 1 componente.

## Propriedades

Vejamos agora algumas propriedades básicas sobre árvores.

***Teorema***. Em uma árvore, quaisquer dois vértices são conectados por exatamente um caminho.

Isto se deve ao fato de que toda árvore é um grafo conectado e acíclico.

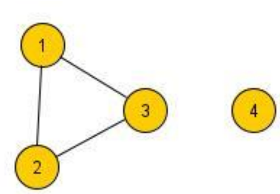
Conforme vimos anteriormente, todo grafo no qual todos os vértices possuem grau mínimo 2, contém um ciclo. Assim podemos concluir que *toda árvore possui um vértice com grau máximo 1*. Este vértice é chamado de **folha (*leaf*)**.

***Proposição***. Toda árvore não-trivial possui no mínimo 2 folhas.

Isto se deve ao fato de que árvores não possuem ciclos. Assim caminhos simples mais longos sempre iniciam e terminam em um vértice folha. Portanto, teremos no mínimo 2.

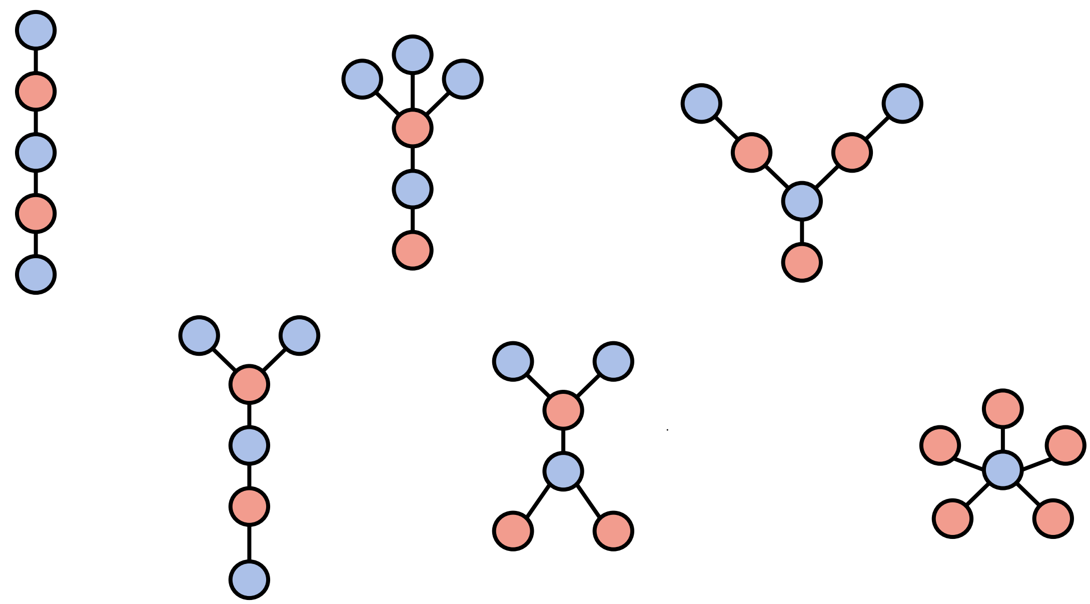
***Teorema***. Se *T* é uma árvore, então ***e(T) = v(T) -1*.**

Este teorema mostra uma propriedade fundamental de toda árvore. Porém, note que não podemos usar esta equação para determinar se um grafo é uma árvore. Observe o grafo abaixo. Este grafo atende a propriedade, mas não é uma árvore. Para provar que um grafo é uma árvore precisamos demonstrar que se trata de um grafo acíclico e conectado.




***Toda árvore é um grafo bipartido***. Já aprendemos em aulas anteriores que um grafo é bipartido ou possui um ciclo de tamanho ímpar. Como árvores são grafos acíclicos, não possuem ciclos. Portanto, são grafos bipartidos.

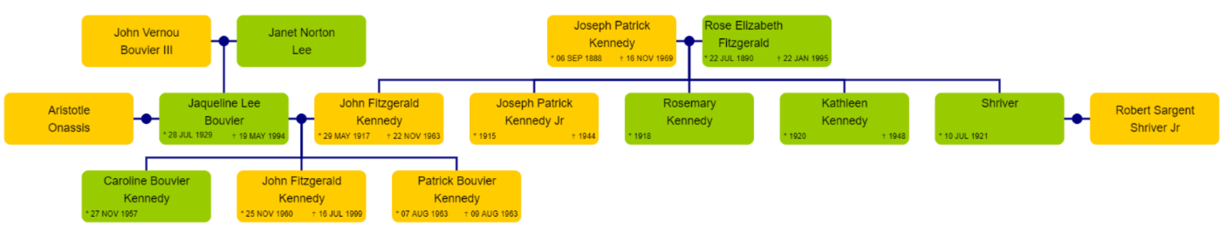
Podemos observar nos exemplos abaixo que isto é verdade pois para toda árvore, é possível definir duas partições do conjunto de vértices onde cada aresta possui um terminal em uma partição e um terminal na outra. Nestes exemplos, as partições estão representadas por cores.



## Aplicações

Abaixo ilustramos alguns exemplos de aplicações de árvores que não se limita a estes tipos. De fato, a aplicabilidade do conceito é muito ampla e não poderia ser ilustrada de forma completa neste documento.

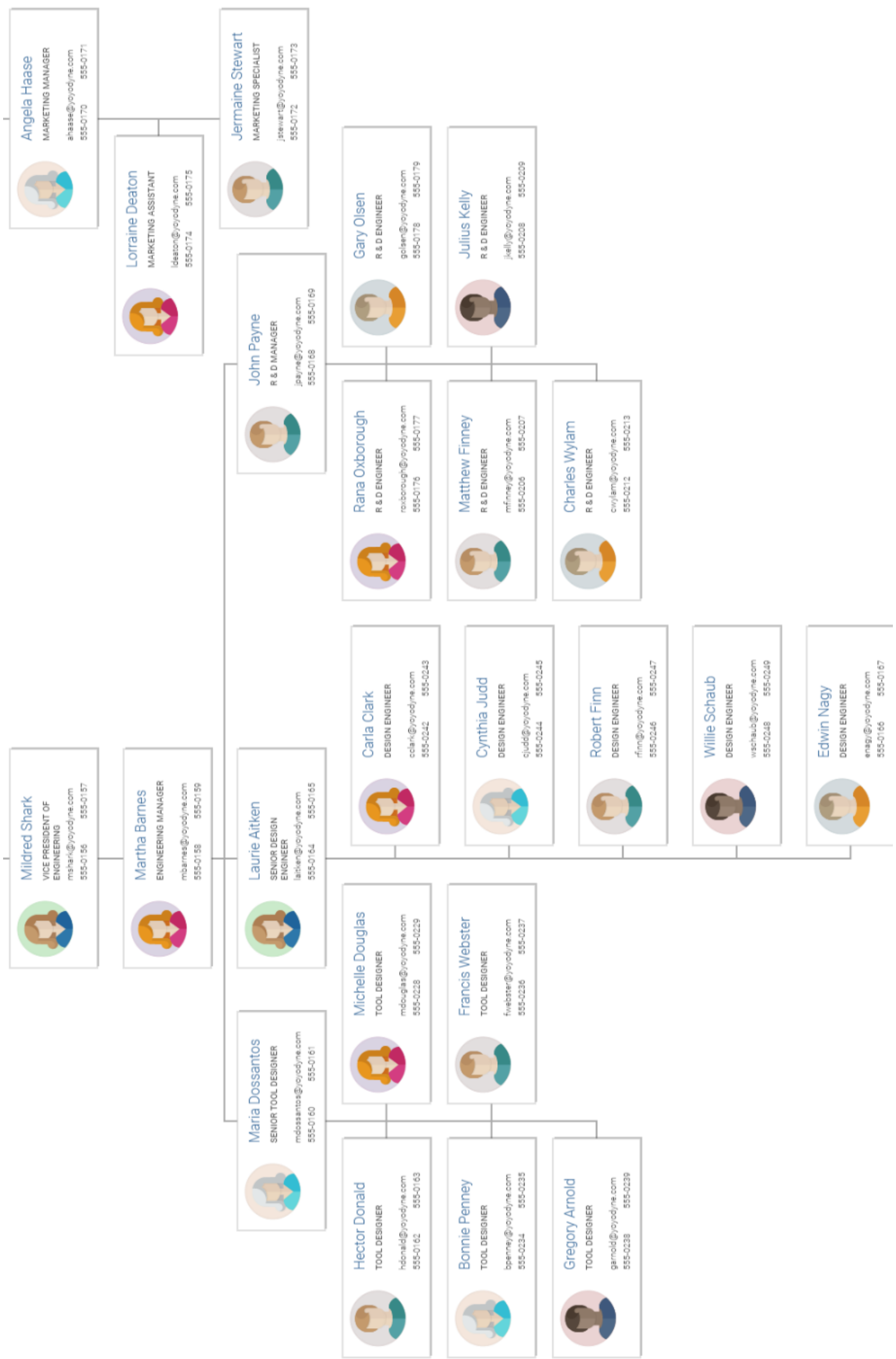
**Árvore Genealógica**



[2]

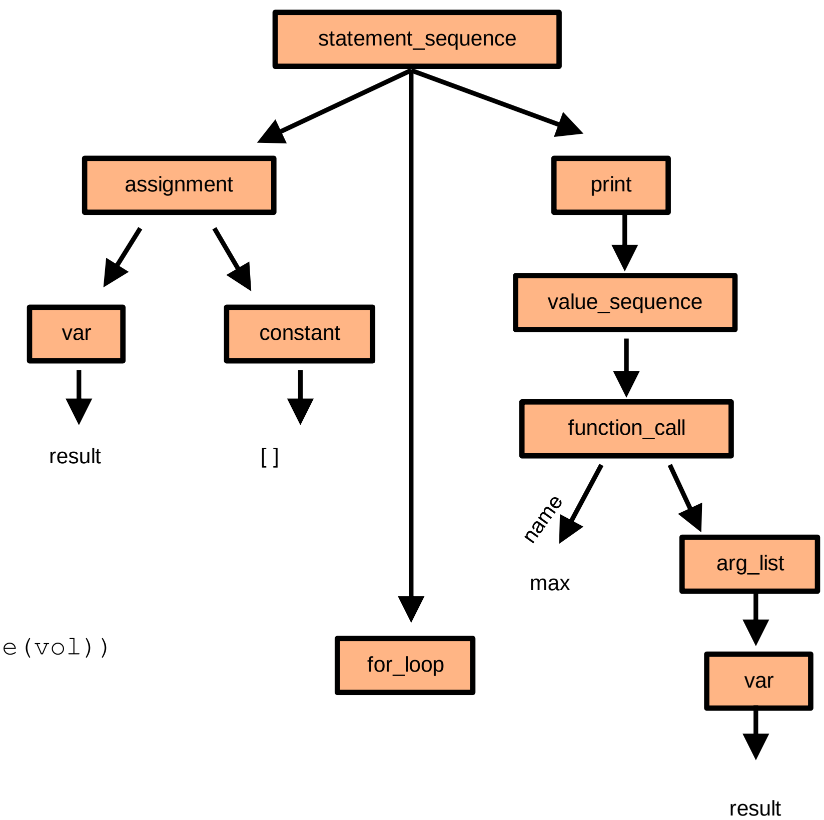
[2] Exemplo de: https://www.yworks.com/yed-live/

**Organograma**



## Árvore Sintática

Uma **árvore de análise sintática** é uma [estrutura de dados](#) que representa a estrutura [sintática](#) de um trecho de código de acordo com alguma [gramática formal](#) definida para uma linguagem de programação. Um programa que produz tal árvore é denominado um [analisador sintático](#) que é um módulo de um compilador responsável por determinar se o trecho de código está sintaticamente correto. A possibilidade de poder representar um trecho de código como uma árvore de acordo com a gramática definida pela linguagem é essencial para que o trecho seja considerado sintaticamente correto.[3]

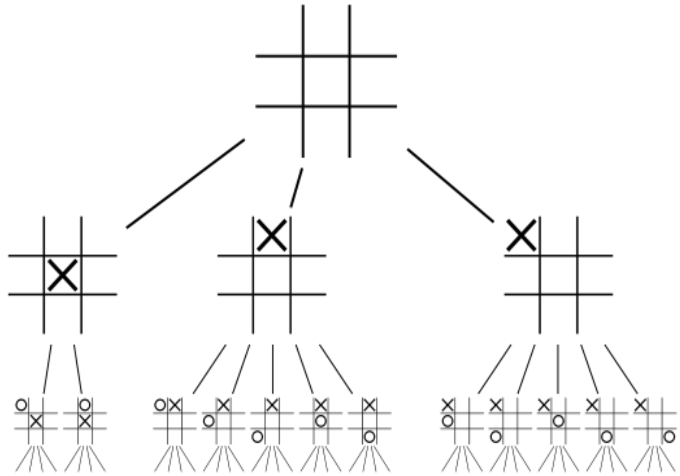



```
result = []
for val in seq:
    result.append(analyze(vol))
print max (result)
```

## Árvore de Jogos

Uma árvore de jogos é um grafo orientado cujos vértices são posições em um jogo e arestas são movimentos. A árvore completa possui como folhas as posições terminais de um jogo (com derrota ou vitória) e cada caminho ilustra uma sequência de jogadas que partem da posição inicial até um destes terminais.

[3]https://en.wikipedia.org/wiki/Abstract_syntax_tree

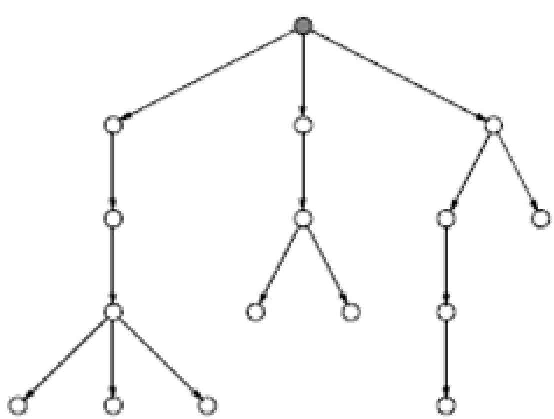
 [4]

## Árvore Enraizada

No conceito mais geral de árvores, não existe distinção entre os vértices do grafo. Árvores são basicamente grafos acíclicos e conectados quaisquer. Porém, quando utilizamos árvores para representar relações hierárquicas ou ordenadas, como no caso da árvore de jogos, é comum distinguirmos um vértice que representa para o problema em questão o vértice inicial. Chamamos este vértice de **raiz**.

Uma **árvore com raiz** (*rooted tree*) *T(x)* é uma árvore *T* com um vértice *x* chamado de raiz de *T*.

Apesar de árvores não serem grafos dirigidos, o fato de escolhermos um vértice como raiz, faz com uma orientação seja definida, como podemos observar no exemplo abaixo.



[4] http://en.wikipedia.org/wiki/Game_tree

Uma orientação de $T(x)$ na qual todo vértice possui grau de entrada igual a *1*, exceto a raiz, é chamada de **árvore enraizada** (*branching*).

## Exercícios Resolvidos

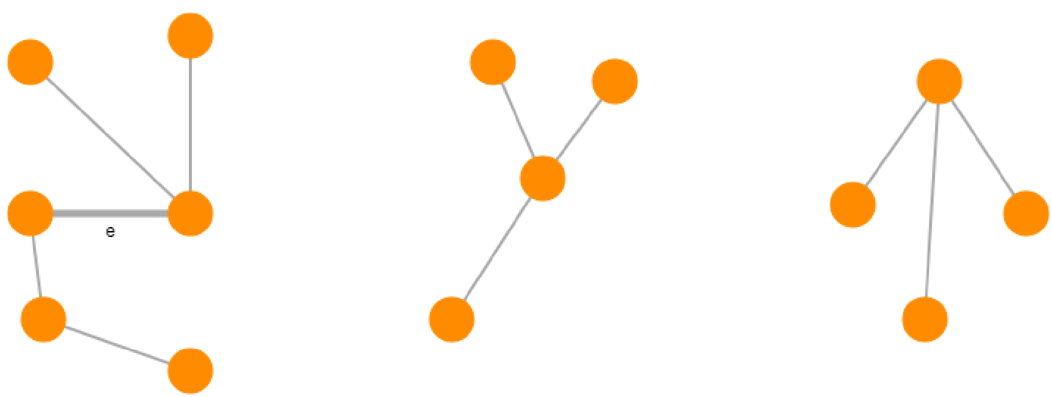
Seja *e* uma das arestas de uma floresta *F*. Mostre que $c(F\backslash e) = c(F) + 1$ (onde c(G) representa a quantidade de componente de um grafo G).

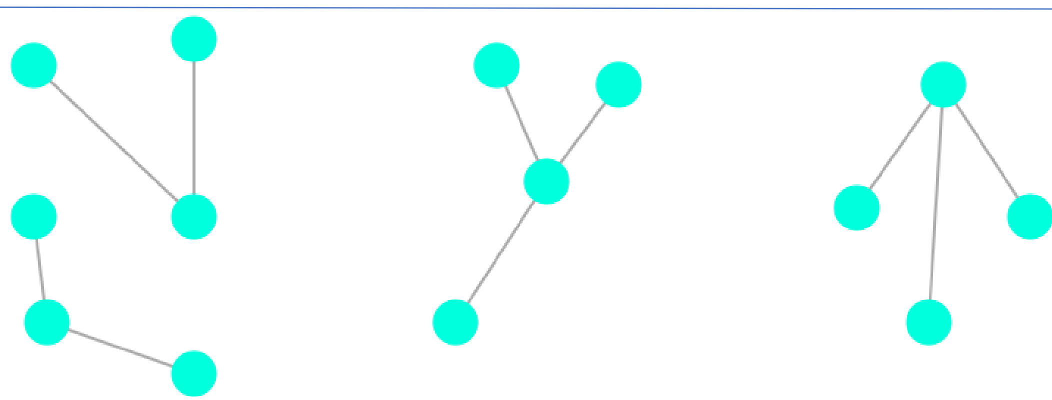
Uma floresta formada por *n* árvores, possui *n* componentes, visto que cada árvore é um grafo conectado.

Como árvores são grafos acíclicos, então todas as arestas de uma árvore são arestas de corte.

Seja *e* uma aresta de uma árvore *A* qualquer da floresta *F*. O resultado da remoção de *e, A\e,* é duas novas árvores A' e A'', substituindo a original. Assim, a quantidade de componentes de *F\e* é a quantidade de componentes de *F* mais *1*.

Considere o exemplo abaixo:



Mostre que toda árvore F tem pelo menos Δ(F) folhas (onde Δ representa o grau máximo de F).

Seja *F* o grafo trivial. Este grafo tem grau máximo 0 e zero folhas.

Seja *F* a árvore com 2 vértices. Este grafo tem grau máximo 1 e 2 folhas.

Seja *F* uma árvore com 3 ou mais vértices. Seja *v* um vértice de *F* que possui grau Δ*(F)*.

Como *v* não é um vértice folha, então deve existir pelo menos Δ(*F*) caminhos disjuntos (que não repetem vértices além de *v*) de *v* até um vértice folha, visto que *F* é um grafo acíclico.

Ou seja, a única forma de não existirem estes caminhos disjuntos é se o grafo tiver ciclos.

## Exercícios Propostos[5]

1. Seja *v* uma folha de uma árvore *T*. Mostre que *T - v* é uma árvore.

2. Mostre que todo grafo caminho (simples) é uma árvore.

3. Seja *G* um grafo conexo tal que *e(G) = v(G) - 1*. Prove que *G* é uma árvore, onde *e(G)* determina a quantidade de arestas e *v(G)* a quantidade de vértices de G.

4. Seja *T* uma árvore com *p + q* vértices. Suponha que *p* dos vértices têm grau 4 e *q* são folhas. Mostre que *q = 2p + 2*.

5. Seja *T* uma árvore com pelo menos três vértices. É verdade que o complemento de *T* é conexo a menos que *T* seja uma estrela?

6. Se *G* é um grafo simples, mostre que se *e(G) >= v(G)*, então *G* possui um ciclo.

## Referências

J. A. Bondy and U. S. R. Murty. Graph Theory. Springer, 2008,2010.

- 4.1, 4.2 (excluindo Cayley's Formula), 4.3 (apenas a definição de co-árvore)
- 4.1 (excluindo Binary Tree), 4.2, 4.3

[5] Algumas questões foram adaptadas de: http://www.ime.usp.br/~pf/grafos-exercicios/