GRAFOS

1

4 – GRAFOS NÃO-ORIENTADOS

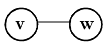
4.1 – Definições

- Um grafo não-orientado também é chamado de grafo **não-dirigido**, ou abreviadamente de **grafo**
- Num grafo não-orientado, os arcos são **linhas** ou **arestas não-orientadas** (deixam de ser setas)

2

- Um arco é definido por um par não-orientado de vértices **(v, w)**:

(v, w) = (w, v)



- Diz-se que o arco **(v, w)** é **incidente** sobre **v** e **w**
- **Caminho** em um grafo é uma seqüência de vértices

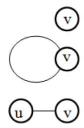
$\mathbf{v_1, v_2, \ldots, v_n}$, tais que

$(\mathbf{v_1, v_2}), (\mathbf{v_2, v_3}), \ldots, (\mathbf{v_{n-1}, v_n})$ são arcos

- **Comprimento** de um caminho é o número de arcos desse caminho

3

- O caminho $v_1, v_2, \ldots, v_n$ **conecta** v_1 a v_n
- **Ciclo:** caminho de um vértice a ele mesmo de comprimento ≥ 3
  - um arco não pode aparecer mais de 1 vez.
- Não são ciclos:



: comprimento zero

: comprimento um

: comprimento dois

Grafo cíclico: tem pelo menos um ciclo

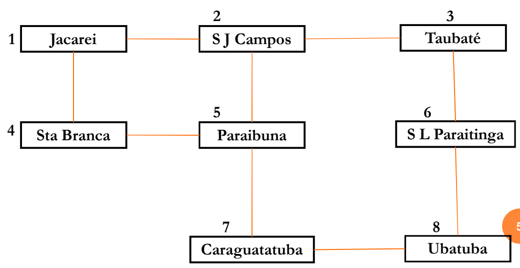
Caso contrário é **acíclico**

Ciclo aqui é diferente de ciclo para digrafos

4

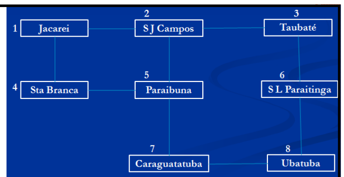
4.2 – Estruturas de dados para grafos não-orientados

- Seja um grafo representando o mapa rodoviário de uma região:



5

a) Matriz de adjacências

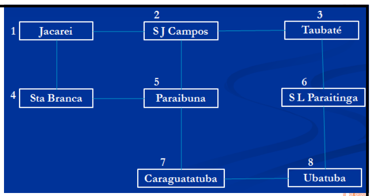


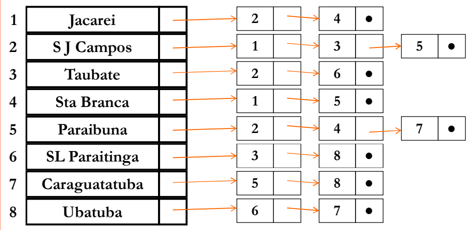
A matriz é simétrica em relação à diagonal principal

	Nome	1	2	3	4	5	6	7	8
1	**Jacarei**		V		V				
2	**S J Campos**	V		V		V			
3	**Taubate**		V				V		
4	**Sta Branca**	V				V			
5	**Paraibuna**		V		V			V	
6	**S L Paraitinga**			V					V
7	**Caraguatatuba**					V			V
8	**Ubatuba**						V	V	

6

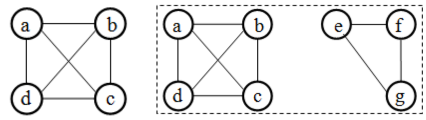
b) Listas de adjacências





4.3 – Componentes conexos e árvores livres

- Um grafo é **conexo** se todos os seus pares de vértices estiverem **conectados** (há um caminho entre dois vértices quaisquer desse grafo)
- Apesar de não haver caminho entre todos os pares de vértices, eles tem "conceitualmente" alguma ligação

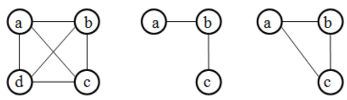


Um grafo conexo

Um grafo não conexo

- Seja um grafo **G = (V, A)**; **sub-grafo** de **G** é um grafo **G'= (V',A')** onde
  - **V'** é um subconjunto de **V**
  - **A'** consiste de arcos **(v, w)** em **A** tais que **v** e **w** estão em **V'**
- **Sub-grafo induzido** de **G** é aquele em que **A'** consiste de **todos** os arcos **(v, w)** em **A**, tais que, tanto **v** como **w** estão em **V'**

Exemplo:

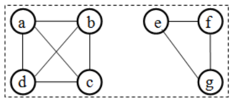


Um grafo G conexo

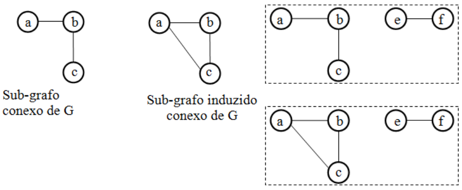
Sub-grafo de G

Sub-grafo induzido de G

Exemplo:



Um grafo G não conexo



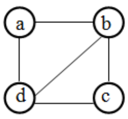
Sub-grafo conexo de G

Sub-grafo induzido conexo de G

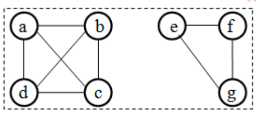
Sub-grafo não conexo de G

Sub-grafo induzido não conexo de G

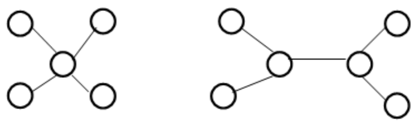
- **Componente conexo** de um grafo **G** é um **sub-grafo induzido conexo máximo** de **G**,
  - isto é, um sub-grafo induzido conexo que não é sub-grafo próprio de nenhum outro sub-grafo conexo de **G**.
- **Exemplo:** grafo com apenas um componente conexo:



- **Exemplo:** grafo com dois componentes conexos:

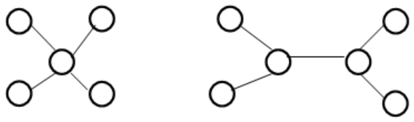


- **Árvore livre:** é um grafo acíclico conexo.
  - Qualquer vértice de uma árvore livre pode ser sua raiz
- **Exemplo:** Grafo não conexo cujos componentes conexos são 2 árvores livres:

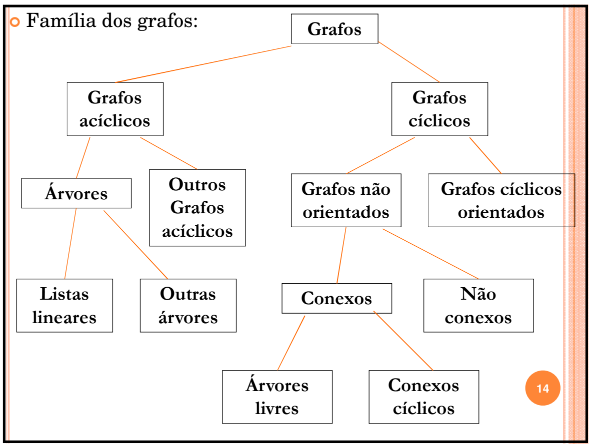


- Propriedades das árvores livres:

1. Toda árvore livre tem pelo menos 1 vértice com somente 1 arco incidente sobre ele.
2. Toda árvore livre com n≥1 vértices tem exatamente n-1 arcos.
3. Acrescentando qualquer arco a uma árvore livre, então um e somente um ciclo é formado.



- Família dos grafos:



Grafos

1 – Aspectos gerais

2 – Grafos orientados

3 – Problemas clássicos sobre grafos orientados

4 – Grafos não-orientados

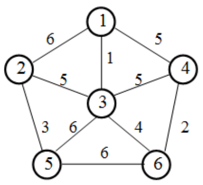
5 – Problemas clássicos sobre grafos não-orientados

5 – Problemas Clássicos sobre Grafos Não-Orientados

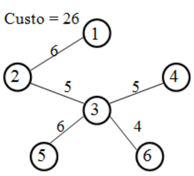
5.1 – Árvore de cobertura de custo mínimo

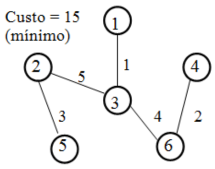
- Seja G = (V, A) um grafo não orientado conexo, com custos associados aos arcos
- **Árvore de cobertura de G:** árvore livre (sub-grafo) de **G** contendo todos os seus vértices
- **Custo de uma árvore de cobertura:** somatória dos custos associados a todos os arcos da árvore

Exemplo: seja o grafo:



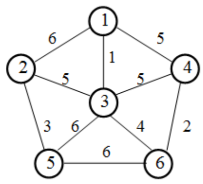
Árvores de cobertura:

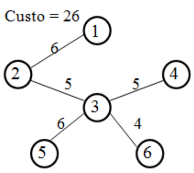


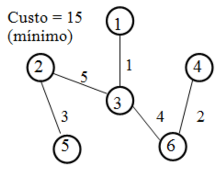


Aplicação: cidades ligadas por uma rede de comunicação:

A árvore de cobertura de custo mínimo representa uma rede que conecta todas as cidades, por um custo mínimo





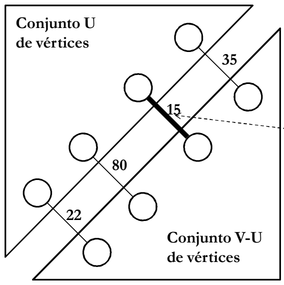


Problema: achar uma árvore de cobertura de um grafo G = (V, A) que seja de custo mínimo (pode haver mais de uma)

Solução: propriedade fundamental dessa árvore:

- Seja **U** um subconjunto próprio de **V**;
- Seja **(u, v)** um dos arcos de menor custo tal que **u ∈ U** e **v ∈ V - U;**
- Então há uma árvore de cobertura de custo mínimo de **G** que inclui o arco **(u, v)**.

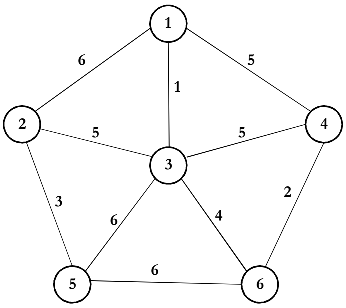
Visualização: um conjunto **V** de vértices

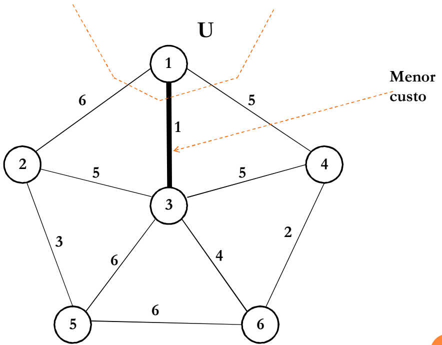


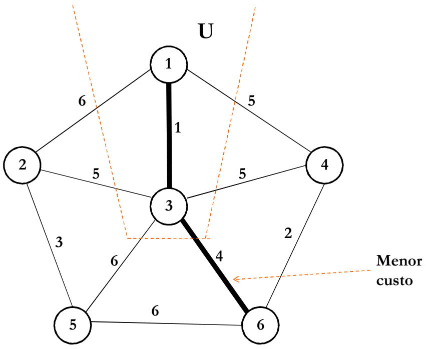
Arco pertencente a uma árvore de cobertura de custo mínimo

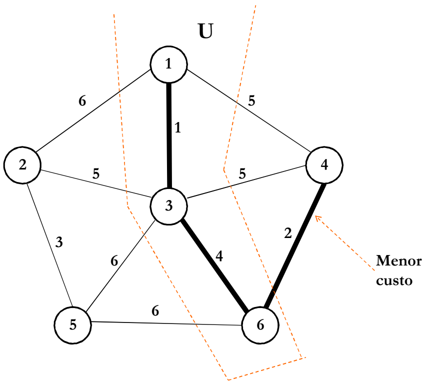
Pode haver mais de um arco de **U** a **V-U** de menor custo

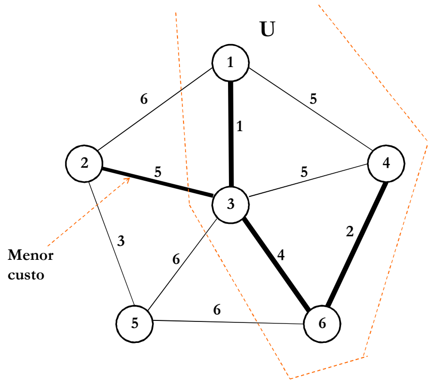
Exemplo: seja o grafo:

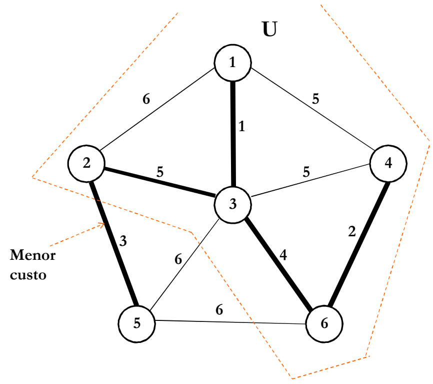






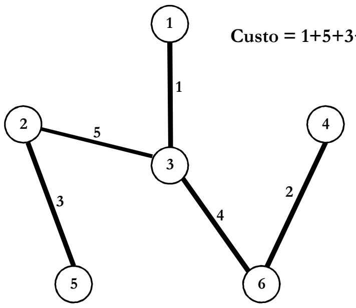






Árvore de cobertura de custo mínimo

Custo = 1+5+3+4+2 = 15



Algoritmo de Prim: determina o conjunto de arcos de uma árvore de cobertura de custo mínimo para o grafo **G**

```
conjuntoarcos  Prim  (grafo  G) {
  conjuntovértices U;  vertice u, v;  conjuntoarcos  T;

  T = ϕ;  U = {1};
  while(U ≠ G.V)  {
       (u, v) = arco de custo mínimo | u ∈ U  e  v ∈ G.V - U;
       T = T ∪ {( u, v )};
       U = U ∪ {v};
  }
  return  T;
}
```


Algoritmo de Prim

```
conjuntoarcos  Prim  (grafo  G) {
    conjuntovertices U,V;  vertice u, v;  conjuntoarcos  T;
     arco  aMinimo;               T = ϕ;  U = {1};
    while(!verticesIguais(U, G.V))  { V = subtrair(G.V,U);
            aMinimo= arcoCustoMinimo(U,V)
            adicionarArco(aMinimo,&T);
            adicionarVertice(v,U);
    }
    return  T;
}
arco arcoCustoMinimo(conjuntovertices U, conjuntovertices V) {
   arco aM;    int custo=infinito;
   Para cada u em U {
        Para cada v em V {
             Se (CustoArco(u,v) <custo ) {
                     aM.inicio=u; aM.fim=v; aM.custo=CustoArco(u,v)
             }
        }
 }}
```


5.2 – Travessia de grafos não orientados

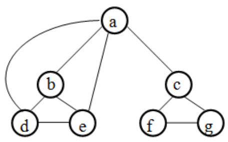
- Visitar todos os nós de um grafo não orientado, de uma maneira sistemática.

a) Método da busca em profundidade

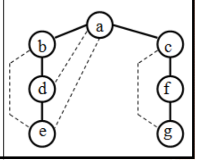
- Usa o mesmo algoritmo da busca em profundidade de digrafos

- **Arcos de árvores** são os mesmos; alguns **arcos de volta** coincidem com eles e são só **arcos de árvore**
- **Arcos para frente** coincidem com **arcos de volta** e são só **arcos de volta**
- **Arcos cruzantes** não existem
- **Exemplo:**

Um grafo



Um seu esquema de visitação



b) Método da busca em largura

- Generalização do caminhamento por **ordem de nível** em árvores
- Ao invés de caminhar na direção dos **filhos**, caminha na direção dos **irmãos**

32

```
void BuscaLargura (Grafo *G) {
  filavertices  F; vertice x, y;
  Assinalar todos os vértices de G como não visitados;
  enquanto (há vértices não visitados) {
       F = ϕ;
       Seja v um vértice qualquer, não visitado;
       Marcar v como visitado;
       EntrarFilaVertices (v, F);
       enquanto  (Vazia (F) == FALSE)   {
              x = FrenteFila (F);  DeletarFila (F);
              para (cada vértice y adjacente a x)
                     se (y não está visitado ) {
                            Marcar y como visitado;
                            EntrarFila (y, F);
                     }
       }
  }
}
```

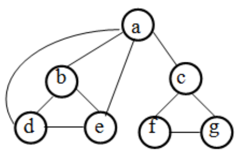
33

5.3 – Pontos de articulação e componentes bi-conexos

- **Ponto de articulação (p-artic):** Vértice **v** de um grafo **G** tal que, se for removido de **G** junto com todos os arcos incidentes sobre ele, um componente conexo de **G** é particionado em dois ou mais componentes conexos
- **Componente bi-conexo:** componente conexo de um grafo **G**, sem p-artic's

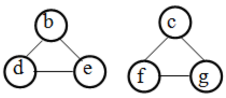
34

- **Exemplo:** **G:**



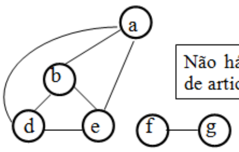
a é ponto de articulação:

G – a:



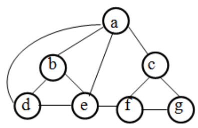
c é ponto de articulação:

G – c:



Não há outros pontos de articulação em **G**

Exemplo: um grafo com um componente bi-conexo



- **Aplicação:** em redes de comunicação, um p-artic é um elemento da rede que não pode falhar:
  - Pares de elementos podem ficar incomunicáveis
- **Grafo bi-conexo:** rede protegida contra a falha de, no máximo, um elemento

36

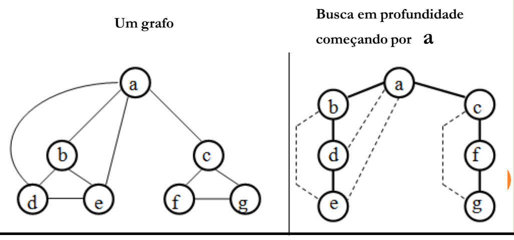
Determinação dos p-artic's de um grafo

- Um método **simples**, porém **ineficiente**:
  - Percorrer o grafo em profundidade tantas vezes quanto for o **número de seus vértices**
  - Em cada uma desses percursos, **começar por um vértice diferente**
  - A raiz da árvore de uma busca é um **p-artic** se tiver **mais de um filho**

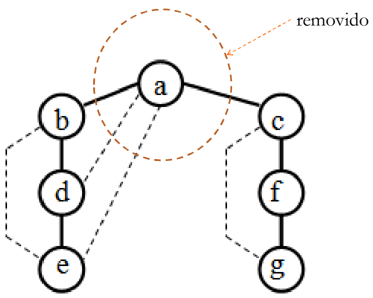
37

Exemplo: busca em profundidade começando pelo vértice **a** do grafo a seguir

- O vértice **a** é p-artic, pois tem dois filhos na árvore de busca



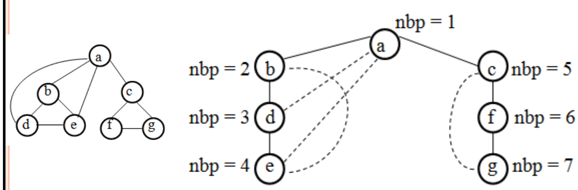
- Como não há arcos cruzantes, com a remoção de **a**, suas duas sub-árvores ficam sem comunicação



39

Um método **mais eficiente**:

1. Percorrer o grafo em profundidade, numerando os vértices na ordem em que forem visitados (**nvis**)



- Conclui-se inicialmente que a raiz **a** é p-artic

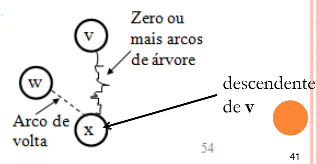
40

2. Visitar os vértices dessa árvore em pós-ordem; para cada vértice **v** visitado, computar **menor[v]**

- Definição de **menor[v]**

  menor[v] = min (nvis[v], nvis[w's])

- **w's** são todos os vértices que se ligam com **v** ou com os descendentes próprios de **v** por arcos de volta



41

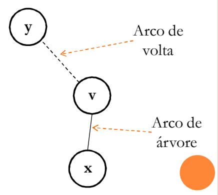
Desenvolvendo a fórmula anterior:

menor[v] = min (nvis[v], nvis[y's], menor[x's])

y's: todos os vértices que se ligam com **v** por arcos de volta

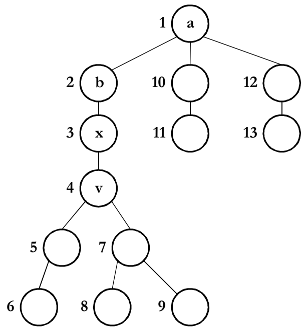
x's: todos os filhos de **v**

A seguir, um estudo da utilidade do conceito de **menor** de um vértice



42

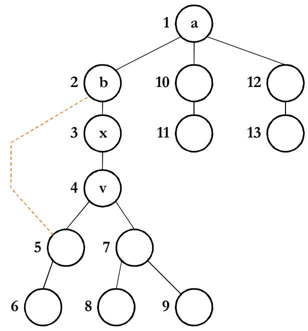
Seja a seguinte árvore de busca em profundidade (nº de busca ao lado de cada vértice):



Sem arcos de volta, o grafo é uma árvore livre

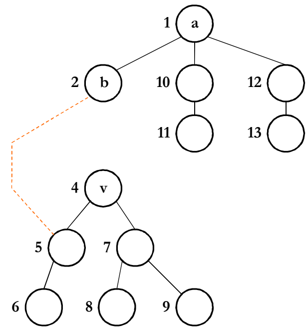
Com a exceção das folhas e de raízes com um só filho, todos os vértices são p-artic

Se algum descendente de **v** (pode ser o próprio **v**) se ligar a **b** ou **a** por arco de volta:



Seja **x** removido

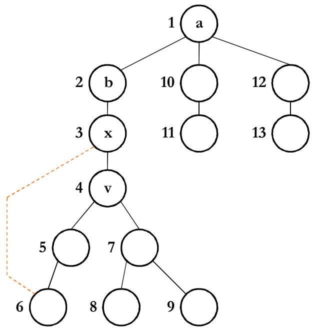
Se algum descendente de **v** (pode ser o próprio **v**) se ligar a **b** ou **a** por arco de volta:



v e seus descendentes não ficam sem comunicação com o resto do grafo

x não é p-artic

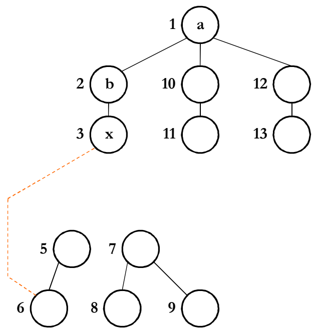
Se de algum filho de **v** não se consegue voltar a **x**, **b** ou **a** sem passar por **v**:



É o caso do nó **7**

Seja **v** removido

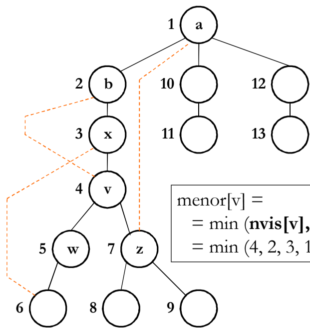
Se de algum filho de **v** não se consegue voltar a **x**, **b** ou **a** sem passar por **v**:



O nó **7** e seus descendentes ficam sem comunicação com o resto do grafo

v é p-artic

Menor[v]: ponto mais alto da árvore (nvis) que se chega de **v**, sem voltar por seus ancestrais

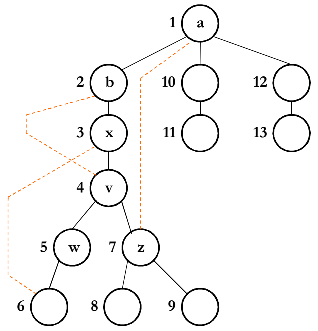


Neste esquema, **menor[v] = 1**

De **v** se chega a **b, x, a**

menor[v] =
= min (**nvis[v], nvis[b], menor[w], menor[z]**)
= min (4, 2, 3, 1) = 1

Se menor [algum filho de v] ≥ nvis(v), v é p-artic



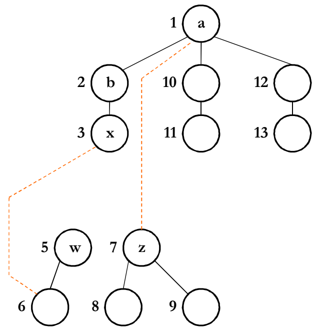
Neste esquema,
menor[w] = 3
menor[z] = 1
nvis[v] = 4

v não é p-artic

Seja **v** removido

49

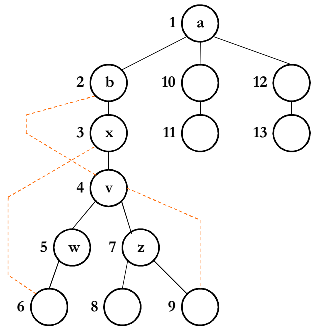
Se menor [algum filho de v] ≥ nvis(v), v é p-artic



w, z e seus descendentes se comunicam com o resto do grafo

50

Se menor [algum filho de v] ≥ nvis(v), v é p-artic



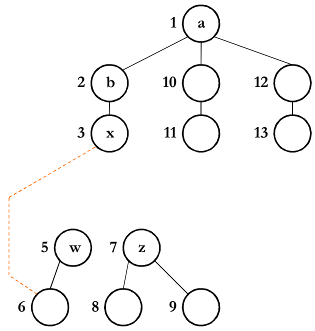
Neste esquema,
menor[w] = 3
menor[z] = 4
nvis[v] = 4

v é p-artic

Seja **v** removido

51

Se menor [algum filho de v] ≥ nvis(v), v é p-artic



z e seus descendentes ficam sem comunicação com o resto do grafo

52

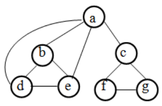
Conclusões sobre a detecção de pontos de articulação:

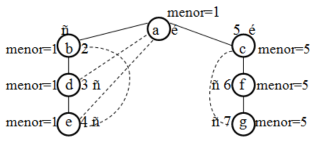
- O vértice **raiz** é p-artic se e somente se tiver **2 ou mais filhos**
- Um vértice **v** ≠ raiz é p-artic se e somente se **∃x** filho de **v** tal que

menor [x] ≥ nvis [v]

53

Exemplo: no grafo ilustrativo





menor [e] = min (nvis [e],
menor [d] = min (nvis [d], nvis [a], menor [e]) = min (3, 1, 1) = 1
menor [b] = min (nvis [b], nvis [e], menor [d]) = min (2, 4, 1)
= 1
menor [g] = min (nvis [g], nvis [c]) = min (7, 5) = 5
menor [f] = min (nvis [f], menor [g]) = min (6, 5) = 5
menor [c] = min (nvis [c], nvis [g], menor [f]) = min (5, 7, 5) = 5
a : é a raiz e tem 2 filhos: é ponto de articulação
c : menor [f] ≥ nvis [c] : é ponto de articulação.

54