Algoritmos em Grafos

Conectividade

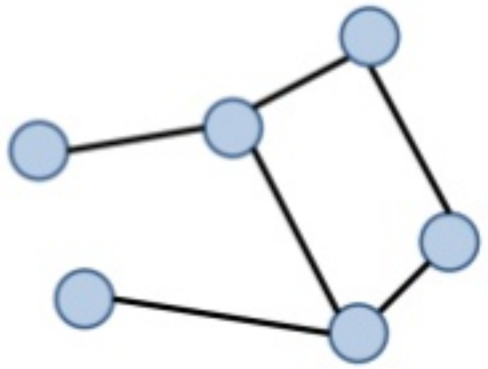
Professora Michelle Nery Nascimento

PUC Minas

Sistemas de Informação

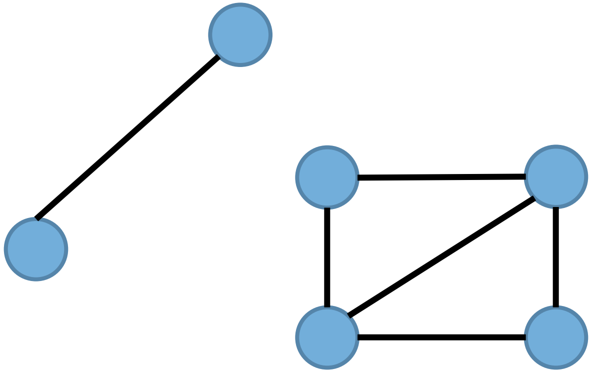
Grafos conexos

- Um grafo é conexo quando existe pelo menos um caminho entre quaisquer pares de vértices



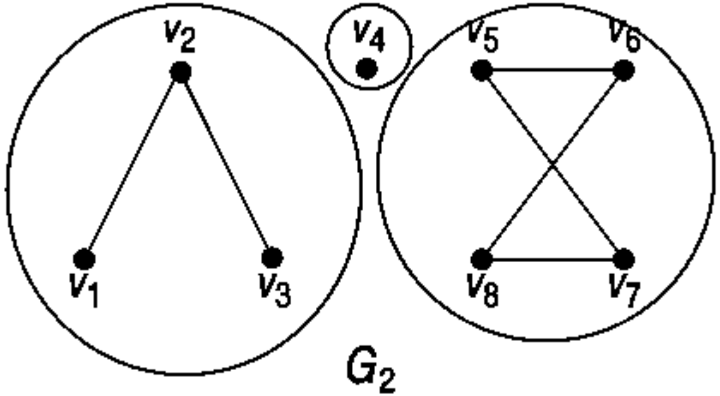
Grafos desconexos e componentes conexos

- Cada componente de um grafo desconectado é chamado de componente conexo



Componentes conexos

- Um grafo pode ser visto como a **união** de seus **componentes conexos**



G_2 apresenta três componentes conexos.

Componentes conexos

- Como saber se um grafo é conexo? (ou, como saber quantos componentes conexos há em um grafo?)

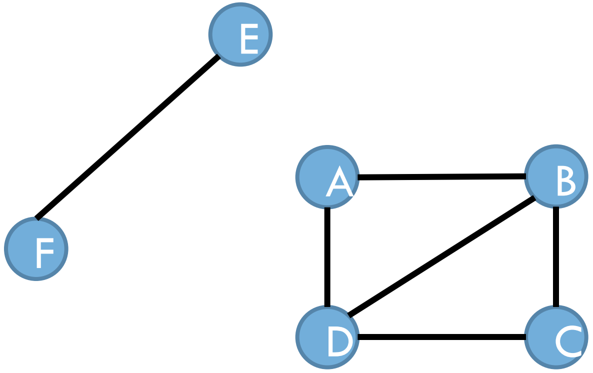
Componentes conexos

- Como saber se um grafo é conexo? (ou, como saber quantos componentes conexos há em um grafo?)

- A busca em profundidade forma árvores. Esta informação pode ser utilizada para contarmos os componentes de um grafo

Algoritmo DFS - inicialização

```
Para cada vértice u faça
        u.cor = branco;
        u.pai = null;
Fim para
componentes=1;
timestamp = 0
Para cada vértice u faça
        se u.cor == branco
                Visitar(u);
                componentes++;
        Fim se
Fim Para
```

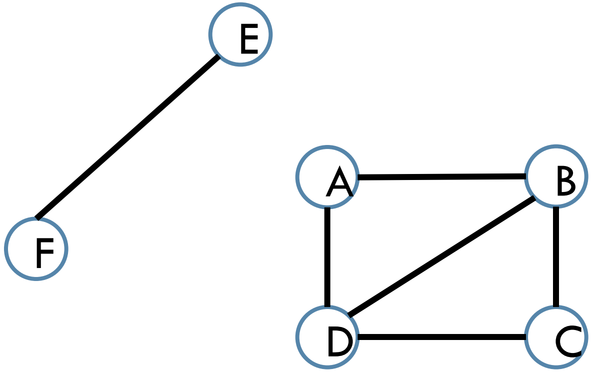


Componente: 1

A	B	C	D	E	F

Algoritmo DFS – principal (visita)

```
timestamp = timestamp + 1;
u.descoberta = timestamp;
u.cor = cinza;
u.componente = componentes;
Para cada vértice v vizinho de u faça
        se v.cor == branco
                v.pai = u;
                Visitar(v);
        Fim se
Fim Para
u.cor = preto;
timestamp = timestamp+1;
u.término = timestamp;
```

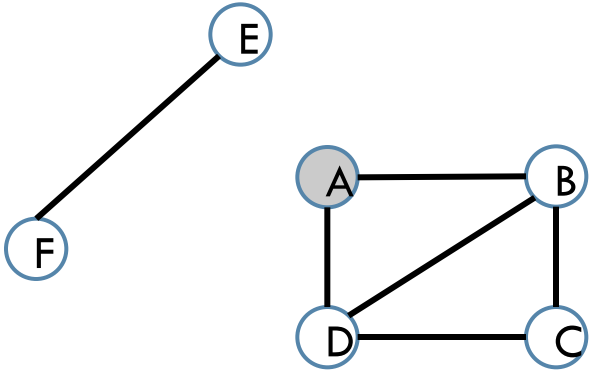


Componente: 1

A	B	C	D	E	F

Algoritmo DFS – principal (visita)

```
timestamp = timestamp + 1;
u.descoberta = timestamp;
u.cor = cinza;
u.componente = componentes;
Para cada vértice v vizinho de u faça
        se v.cor == branco
                v.pai = u;
                Visitar(v);
        Fim se
Fim Para
u.cor = preto;
timestamp = timestamp+1;
u.término = timestamp;
```

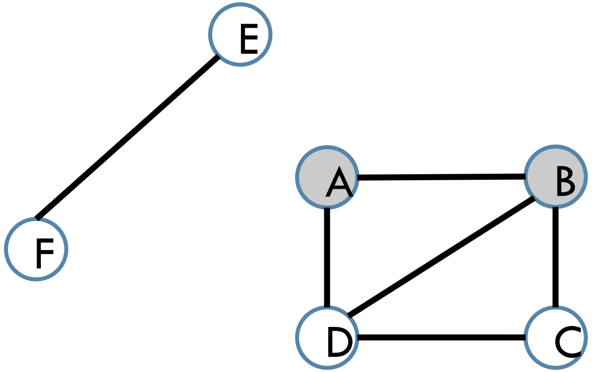


Componente: 1

A	B	C	D	E	F
1					

Algoritmo DFS – principal (visita)

```
timestamp = timestamp + 1;
u.descoberta = timestamp;
u.cor = cinza;
u.componente = componentes;
Para cada vértice v vizinho de u faça
        se v.cor == branco
                v.pai = u;
                Visitar(v);
        Fim se
Fim Para
u.cor = preto;
timestamp = timestamp+1;
u.término = timestamp;
```

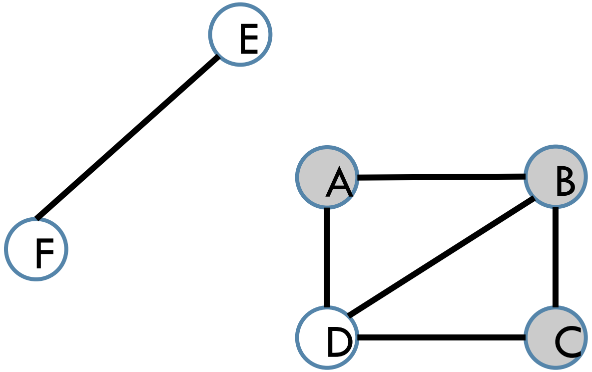


Componente: 1

A	B	C	D	E	F
1	1				

Algoritmo DFS – principal (visita)

```
timestamp = timestamp + 1;
u.descoberta = timestamp;
u.cor = cinza;
u.componente = componentes;
Para cada vértice v vizinho de u faça
        se v.cor == branco
                v.pai = u;
                Visitar(v);
        Fim se
Fim Para
u.cor = preto;
timestamp = timestamp+1;
u.término = timestamp;
```

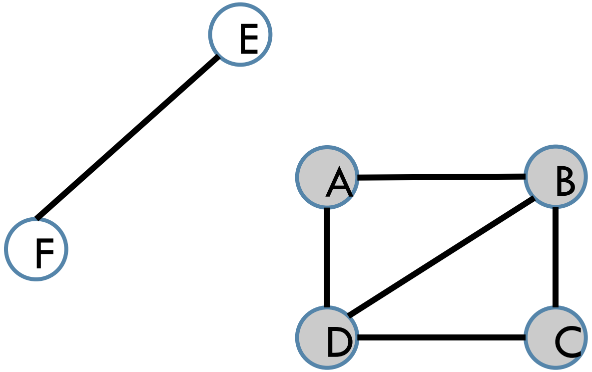


Componente: 1

A	B	C	D	E	F
1	1	1			

Algoritmo DFS – principal (visita)

```
timestamp = timestamp + 1;
u.descoberta = timestamp;
u.cor = cinza;
u.componente = componentes;
Para cada vértice v vizinho de u faça
        se v.cor == branco
                v.pai = u;
                Visitar(v);
        Fim se
Fim Para
u.cor = preto;
timestamp = timestamp+1;
u.término = timestamp;
```

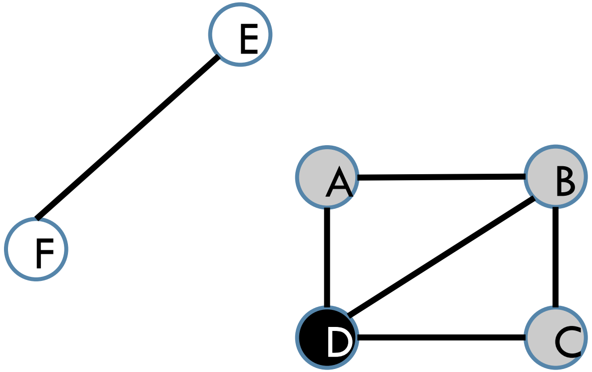


Componente: 1

A	B	C	D	E	F
1	1	1	1		

Algoritmo DFS – principal (visita)

```
timestamp = timestamp + 1;
u.descoberta = timestamp;
u.cor = cinza;
u.componente = componentes;
Para cada vértice v vizinho de u faça
        se v.cor == branco
                v.pai = u;
                Visitar(v);
        Fim se
Fim Para
u.cor = preto;
timestamp = timestamp+1;
u.término = timestamp;
```

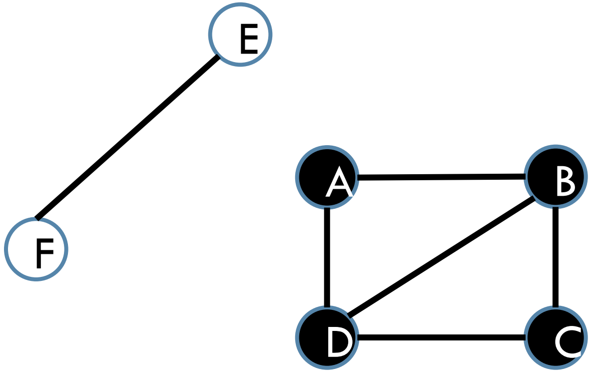


Componente: 1

A	B	C	D	E	F
1	1	1	1		

Algoritmo DFS – principal (visita)

```
timestamp = timestamp + 1;
u.descoberta = timestamp;
u.cor = cinza;
u.componente = componentes;
Para cada vértice v vizinho de u faça
        se v.cor == branco
                v.pai = u;
                Visitar(v);
        Fim se
Fim Para
u.cor = preto;
timestamp = timestamp+1;
u.término = timestamp;
```

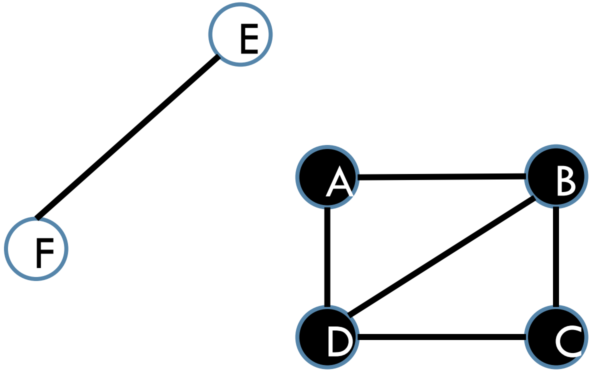


Componente: 1

A	B	C	D	E	F
1	1	1	1		

Algoritmo DFS - inicialização

```
Para cada vértice u faça
        u.cor = branco;
        u.pai = null;
Fim para
componentes=1;
timestamp = 0
Para cada vértice u faça
        se u.cor == branco
                Visitar(u);
                componentes++;
        Fim se
Fim Para
```

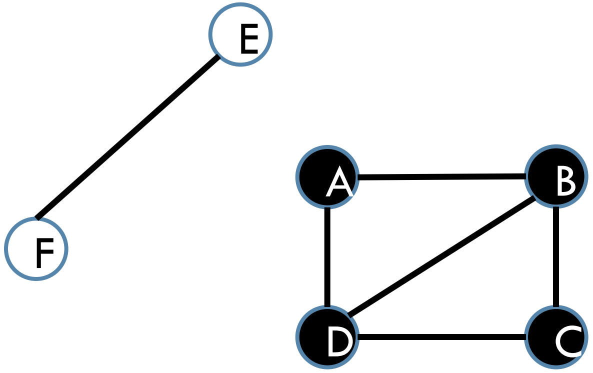


Componente: 1

A	B	C	D	E	F
1	1	1	1		

Algoritmo DFS - inicialização

```
Para cada vértice u faça
        u.cor = branco;
        u.pai = null;
Fim para
componentes=1;
timestamp = 0
Para cada vértice u faça
        se u.cor == branco
                Visitar(u);
                componentes++;
        Fim se
Fim Para
```

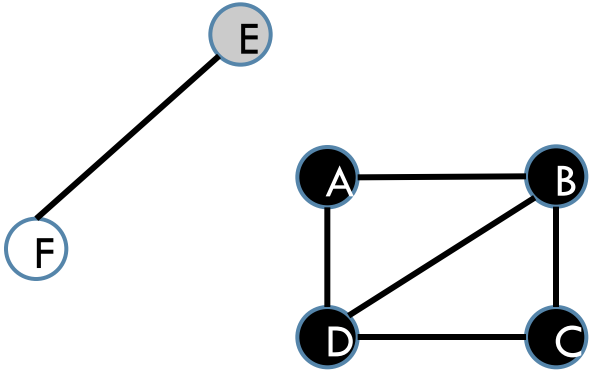


Componente: 2

A	B	C	D	E	F
1	1	1	1		

Algoritmo DFS – principal (visita)

```
timestamp = timestamp + 1;
u.descoberta = timestamp;
u.cor = cinza;
u.componente = componentes;
Para cada vértice v vizinho de u faça
        se v.cor == branco
                v.pai = u;
                Visitar(v);
        Fim se
Fim Para
u.cor = preto;
timestamp = timestamp+1;
u.término = timestamp;
```

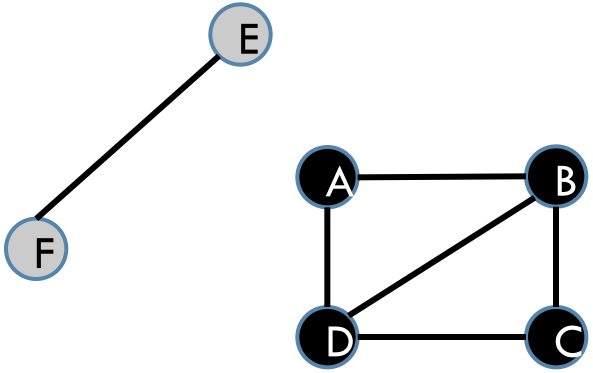


Componente: 2

A	B	C	D	E	F
1	1	1	1	2	

Algoritmo DFS – principal (visita)

```
timestamp = timestamp + 1;
u.descoberta = timestamp;
u.cor = cinza;
u.componente = componentes;
Para cada vértice v vizinho de u faça
        se v.cor == branco
                v.pai = u;
                Visitar(v);
        Fim se
Fim Para
u.cor = preto;
timestamp = timestamp+1;
u.término = timestamp;
```

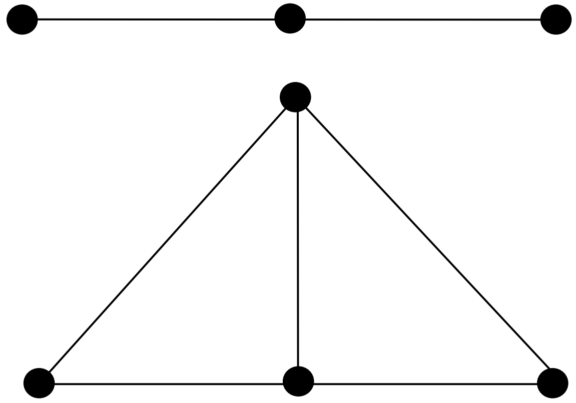


Componente: 2

A	B	C	D	E	F
1	1	1	1	2	2

Rank ou posto

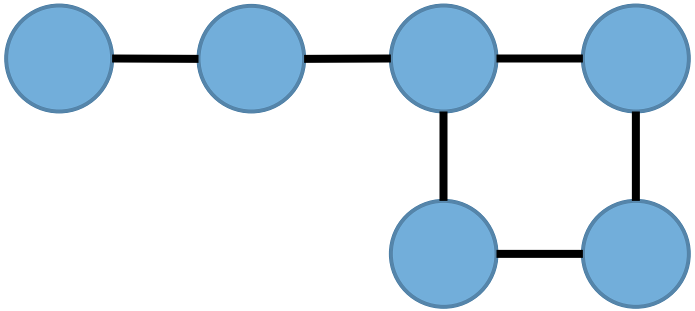
- O *rank* **r** ou posto de um grafo G com **n** vértices e **c** componentes conexos é dado por:
  - r = n – c



n = 7
c = 2
r = 7 – 2 = 5

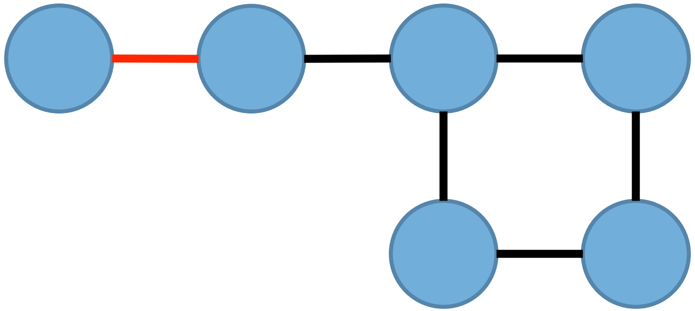
Cut-edge

- Um *cut*-*edge* ou uma ponte é uma aresta cuja remoção desconecta o grafo



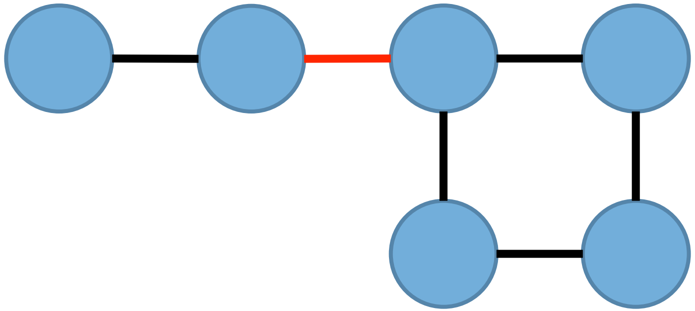
Cut-edge

- Um *cut-edge* ou uma ponte é uma aresta cuja remoção desconecta o grafo



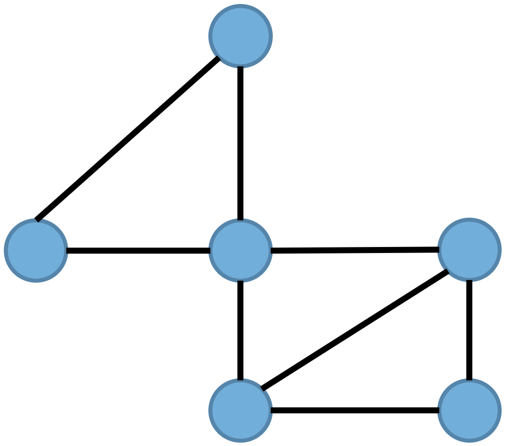
Cut-edge

- Um *cut*-*edge* ou uma ponte é uma aresta cuja remoção desconecta o grafo



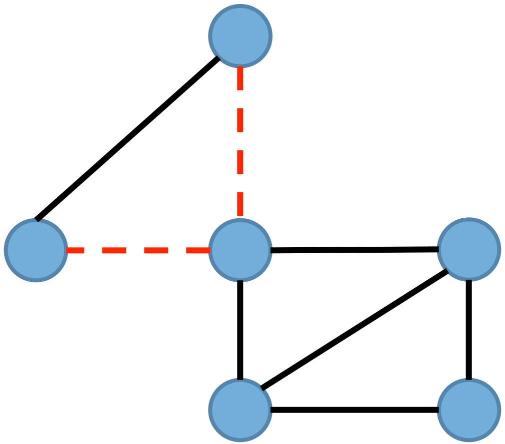
Cut-set

- Conjunto de arestas de um grafo conexo G cuja remoção desconecta G



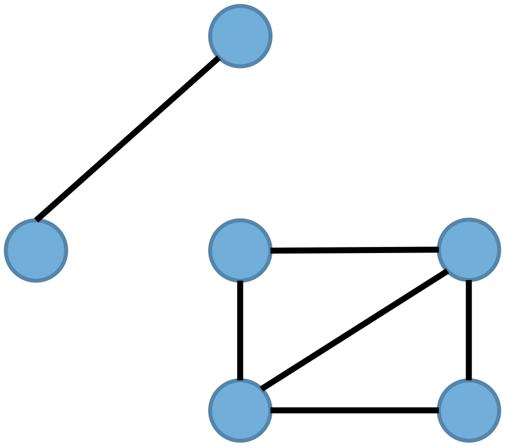
Cut-set

- Conjunto de arestas de um grafo conexo G cuja remoção desconecta G



Cut-set

- Conjunto de arestas de um grafo conexo G cuja remoção desconecta G

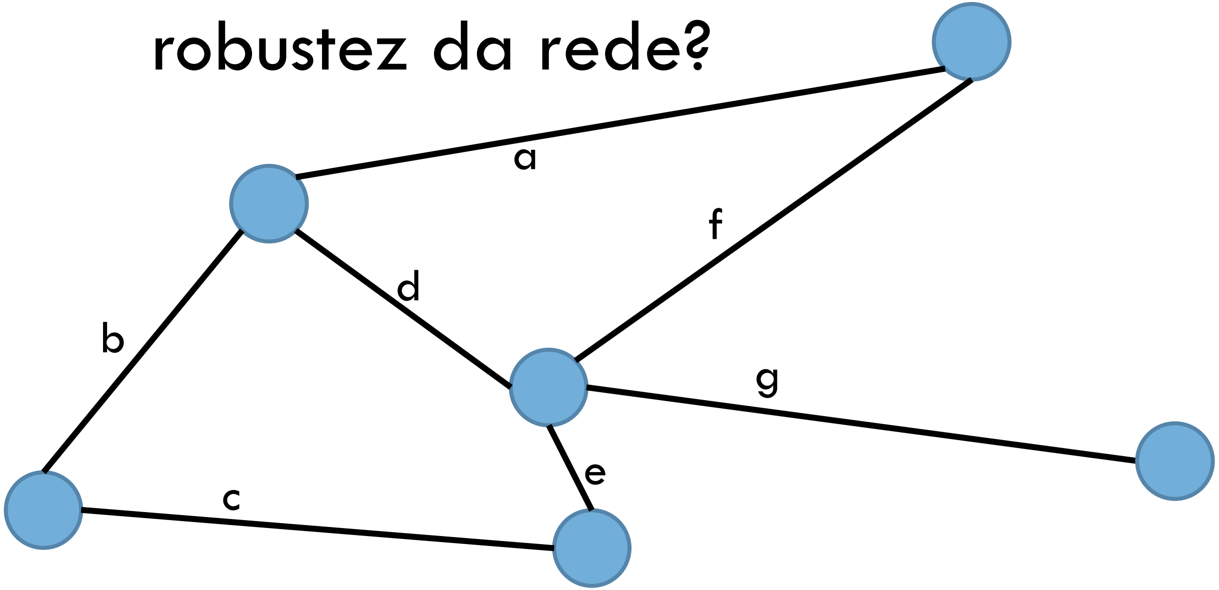


Cut-set

- Um *cut-set* particiona o grafo em dois subgrafos disjuntos
- Um *cut-set* pode ser definido como o conjunto de arestas em um grafo conexo cuja remoção reduz o *rank* do grafo em 1 unidade.

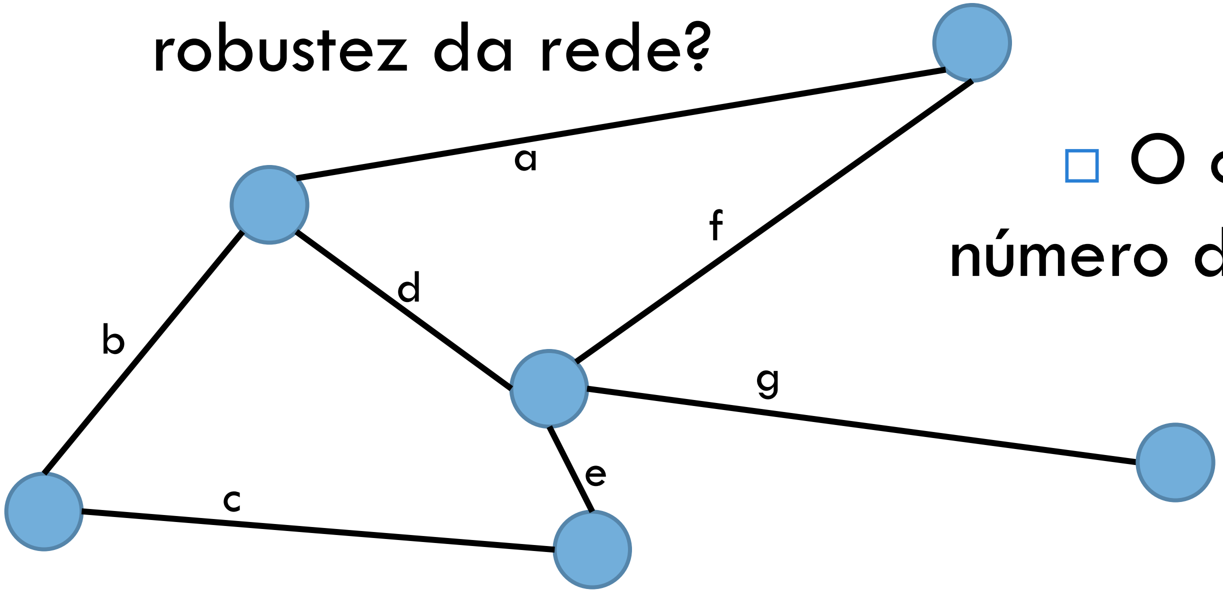
Cut-set: aplicação

- Dada uma rede de comunicação, como medir a robustez da rede?



Cut-set: aplicação

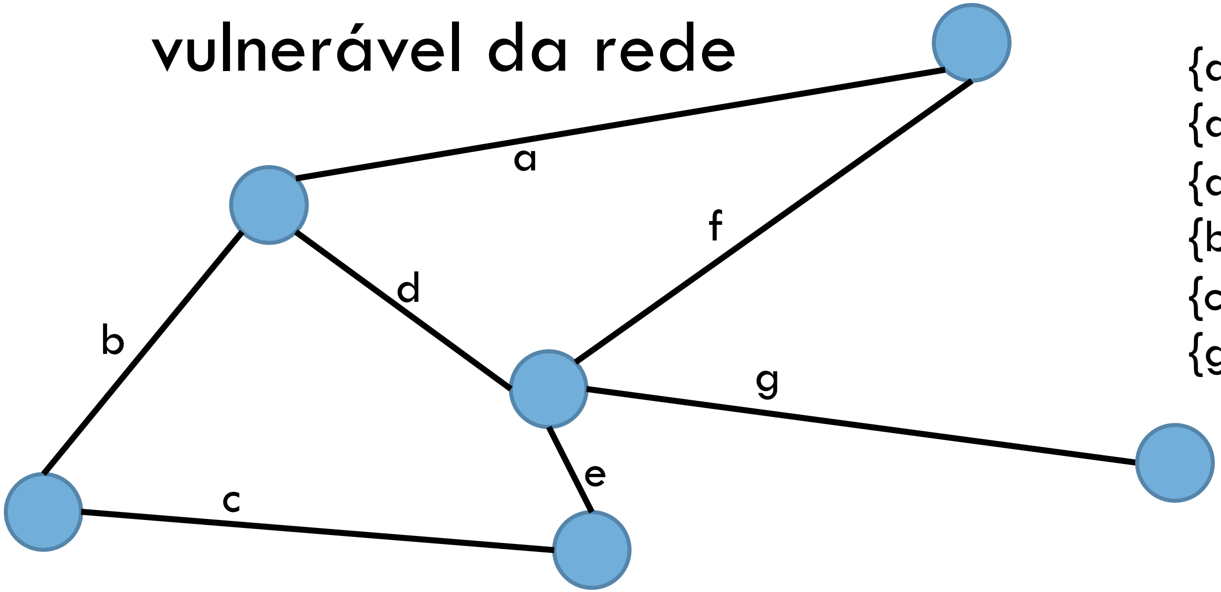
- Dada uma rede de comunicação, como medir a robustez da rede?



- O cut-set com o menor número de arestas é o mais vulnerável da rede

Cut-set: aplicação

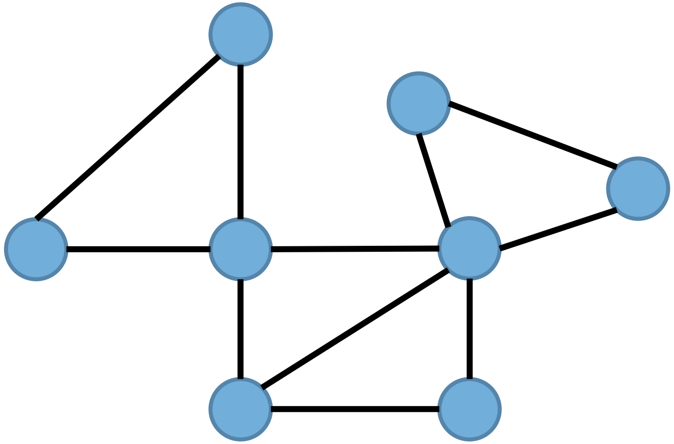
- O cut-set com o menor número de arestas é o mais vulnerável da rede



{a,b,d}
{a,c,d}
{a,f}
{b,c}
{c,e}
{g}

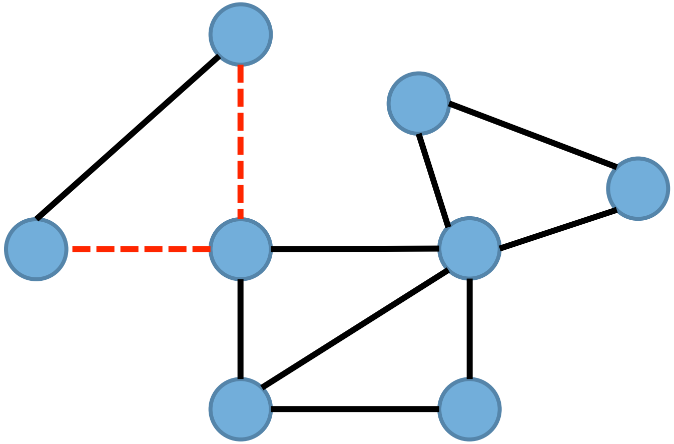
Conectividade e separabilidade

- Conectividade de aresta $\lambda(G)$:
  - menor número de arestas do grafo cuja remoção o desconecta. É o número de arestas do menor *cut-set*



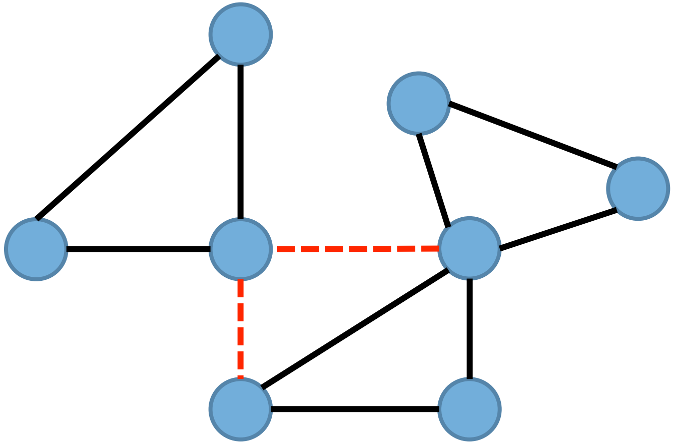
Conectividade e separabilidade

- Conectividade de aresta $\lambda(G)$:
  - menor número de arestas do grafo cuja remoção o desconecta. É o número de arestas do menor *cut-set*



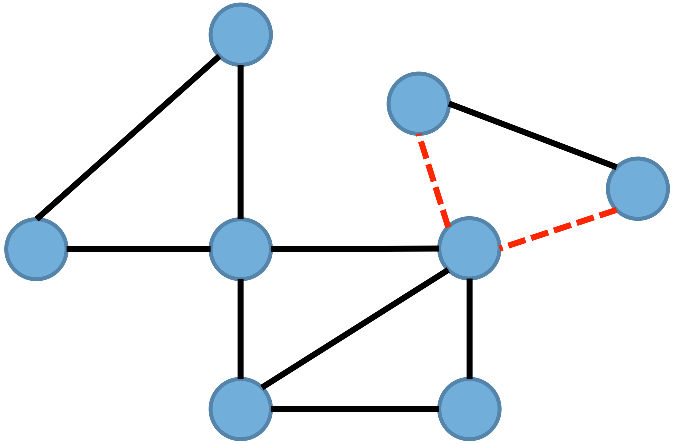
Conectividade e separabilidade

- Conectividade de aresta $\lambda(G)$:
  - menor número de arestas do grafo cuja remoção o desconecta. É o número de arestas do menor *cut-set*



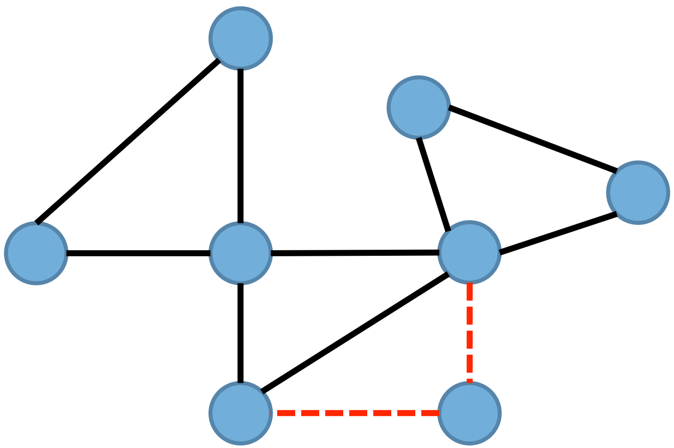
Conectividade e separabilidade

- Conectividade de aresta $\lambda(G)$:
  - menor número de arestas do grafo cuja remoção o desconecta. É o número de arestas do menor *cut-set*



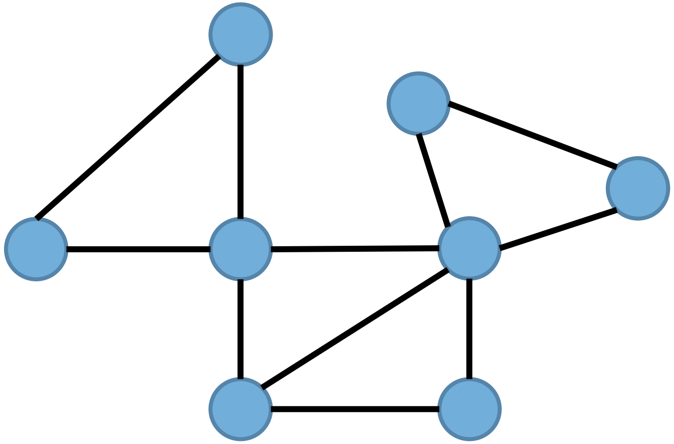
Conectividade e separabilidade

- Conectividade de aresta $\lambda(G)$:
  - menor número de arestas do grafo cuja remoção o desconecta. É o número de arestas do menor *cut-set*



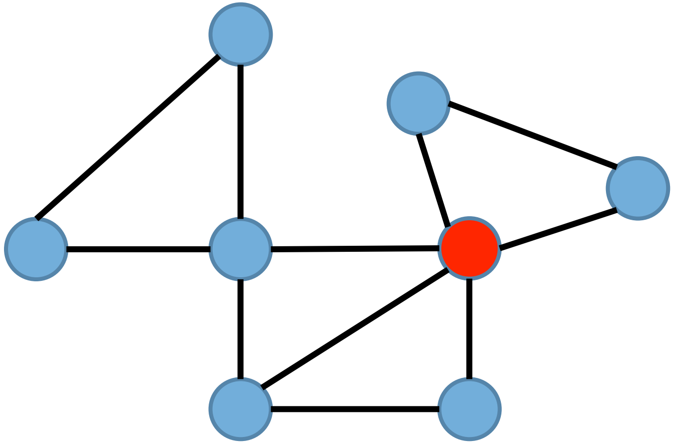
Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta



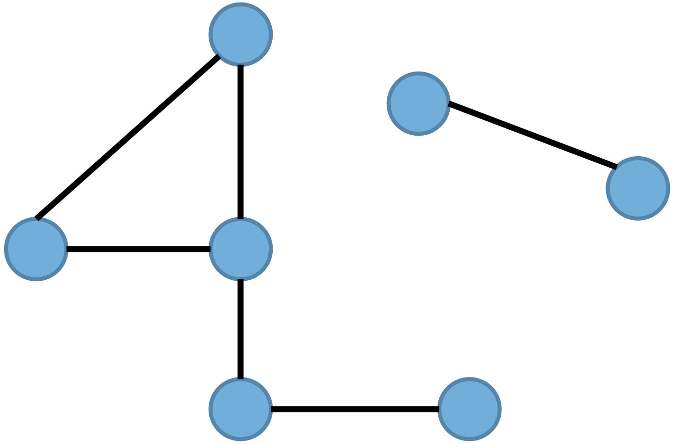
Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta



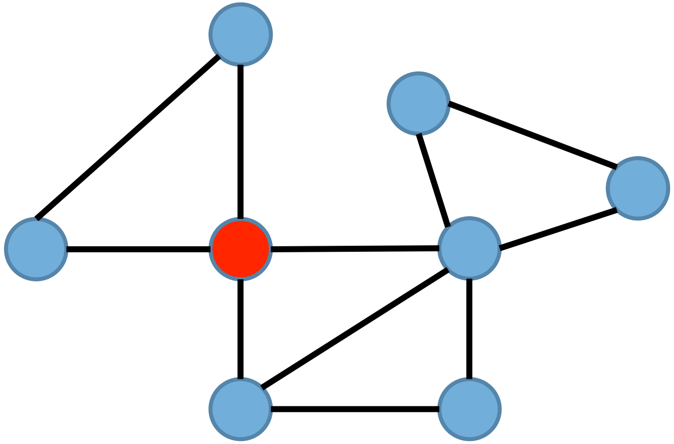
Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta



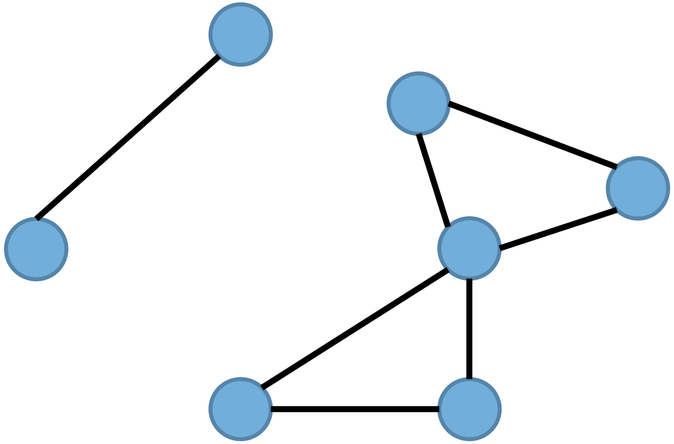
Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta



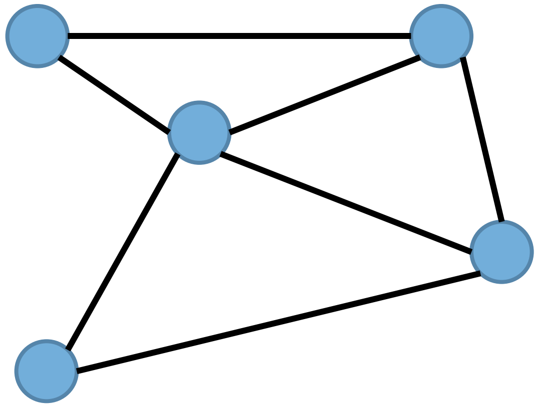
Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta



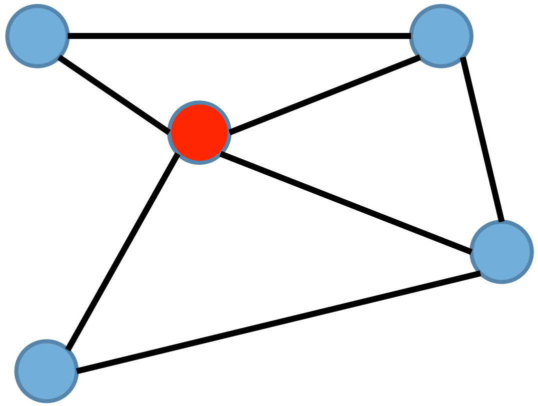
Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta



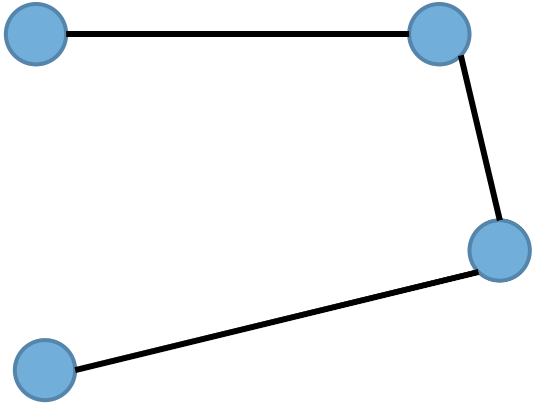
Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta



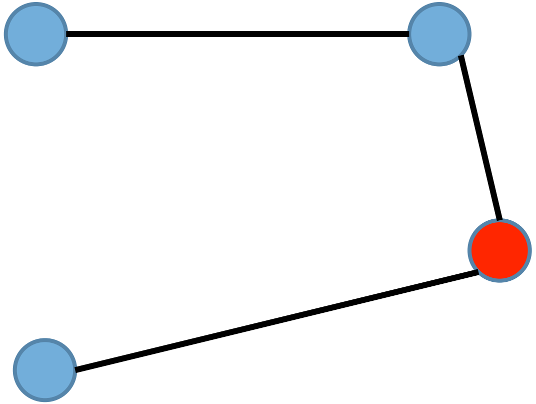
Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta



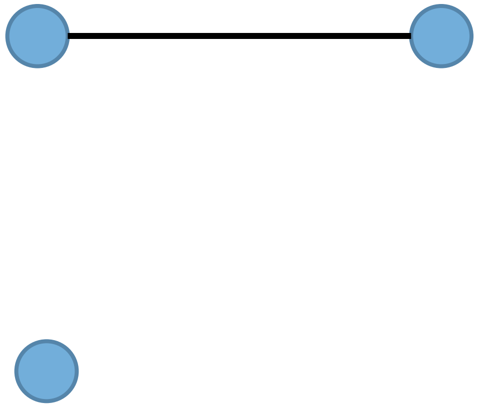
Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta



Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta



Conectividade e separabilidade

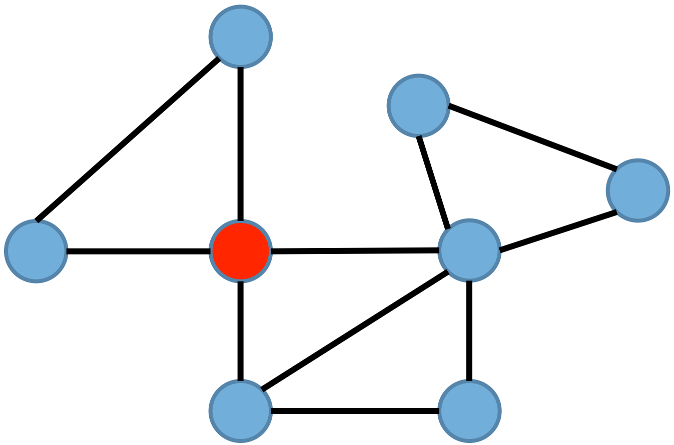
- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta
- Grafo K-conexo: grafo de conectividade de vértice igual a K

Conectividade e separabilidade

- Conectividade de vértice K(G):
  - menor número de vértices do grafo cuja remoção (em conjunto com suas arestas adjacentes) o desconecta
- Grafo K-conexo: grafo de conectividade de vértice igual a K.
- Grafo separável: grafo com conectividade de vértice igual a 1

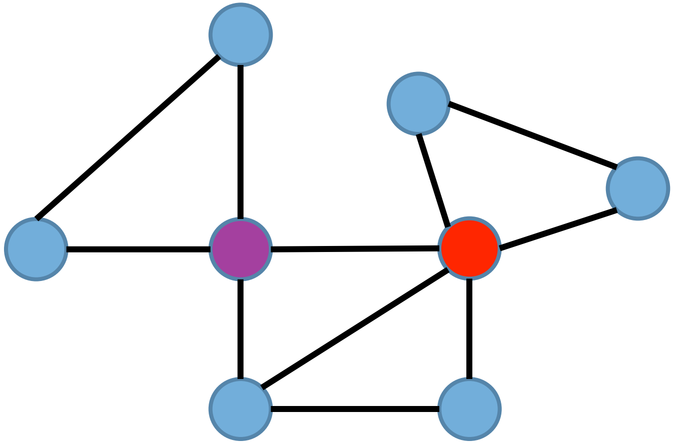
Cut-vértice

- Vértice que desconecta um grafo separável (também chamado *cut vertex ou ponto de articulação*)



Cut-vértice

- Vértice que desconecta um grafo separável (também chamado *cut vertex ou ponto de articulação*)



Teoremas

- Teorema 1:
  - A conectividade de aresta de um grafo G, $\lambda(G)$, não pode exceder o grau do vértice de menor grau
  - **$\lambda(G) \leq$ grau do vértice de menor grau**
- Se um grafo apresenta, pelo menos, um vértice de grau 1
  - então a conectividade de aresta desse grafo é composta por apenas uma aresta

Teoremas

- Teorema 2:
  - A conectividade de vértice de um grafo G, K(G), não pode exceder a conectividade de aresta de G
  - Para todo grafo conexo G tem-se:
    - **K(G) ≤ λ(G) ≤ grau do vértice de menor grau**

Conectividade

- Seja $\delta(G)$ o menor grau de vértice em G

Para todo grafo conexo G, tem-se:

$$K(G) \leq \lambda(G) \leq \delta(G)$$

Teoremas

- TEOREMA 3: A máxima conectividade de vértice de um grafo G com n vértices e e arestas ($e \geq n-1$) é $\frac{2e}{n}$

Para todo grafo conexo G, tem-se:

$$K(G) \leq \lambda(G) \leq \frac{2e}{n}$$

Aplicação - exemplo

- Oito computadores serão conectados por linhas remotas privadas. Existem 16 linhas disponíveis. Como organizar a rede de computadores de maneira que ela fique o mais invulnerável (robusta) possível a falhas nas máquinas individuais ou nas linhas de comunicação?

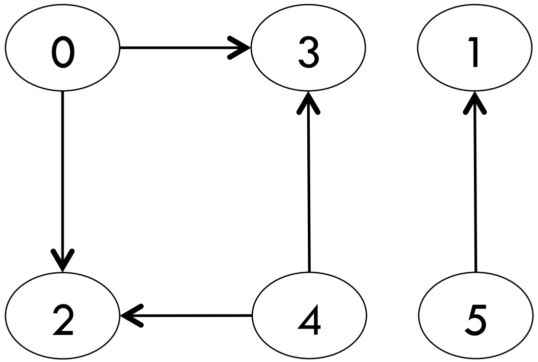
Aplicação - exemplo

- $$\frac{2e}{n} = \frac{2*16}{8} = 4$$
- Grafo com conectividade de aresta 4 e 16 arestas:
  - $K_{4,4}$
- Solução: dois conjuntos de 4 computadores cada; os computadores de um conjunto ligados a todos os computadores do outro conjuno

Conectividade em digrafos

- Desconexo
  - Um grafo é não-conexo ou desconexo se nem todo par de vértices é unido por uma cadeia

Desconexo

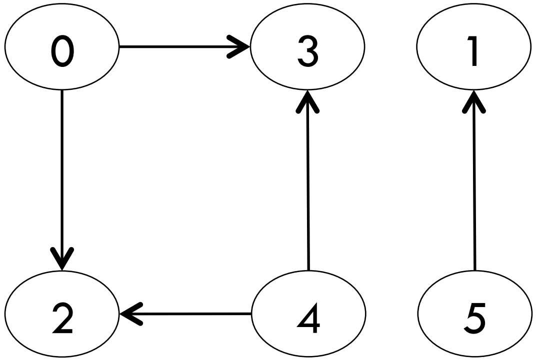


Conectividade em digrafos

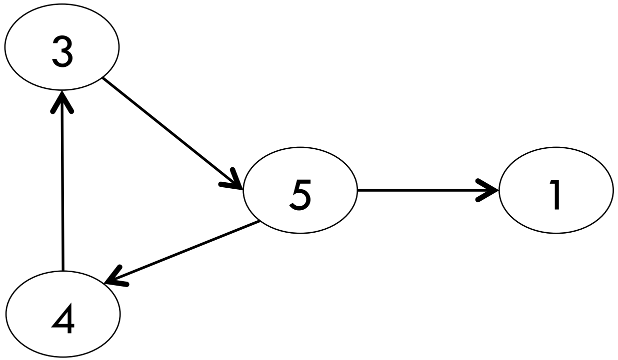
- Desconexo
  - Um grafo é não-conexo ou desconexo se nem todo par de vértices é unido por uma cadeia
- S-conexo
  - Um grafo é simplesmente conexo ou s-conexo se todo par de vértices é unido por pelo menos uma cadeia

Conectividade em digrafos

Desconexo



S-conexo

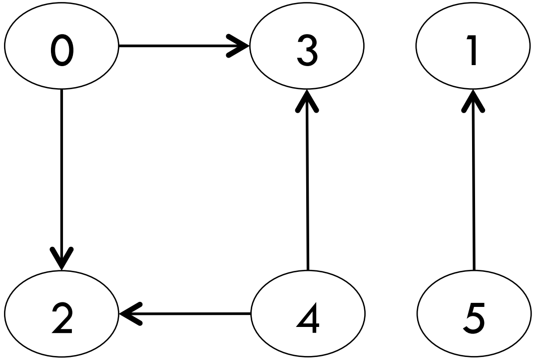


Conectividade em digrafos

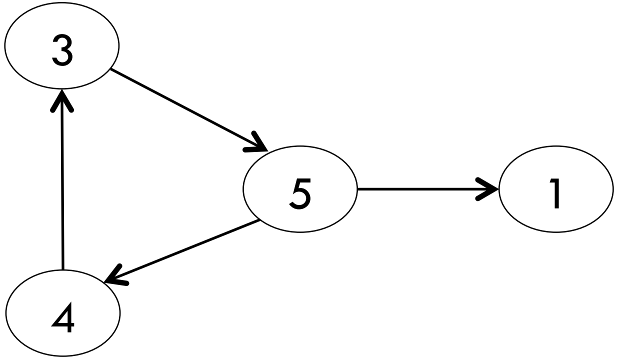
- Desconexo
  - Um grafo é não-conexo ou desconexo se nem todo par de vértices é unido por uma cadeia
- S-conexo
  - Um grafo é simplesmente conexo ou s-conexo se todo par de vértices é unido por pelo menos uma cadeia
- SF-conexo
  - Um grafo é semi-fortemente conexo ou sf-conexo se para todo par de vértice, pelo menos um deles é alcançável a partir do outro

Conectividade em digrafos

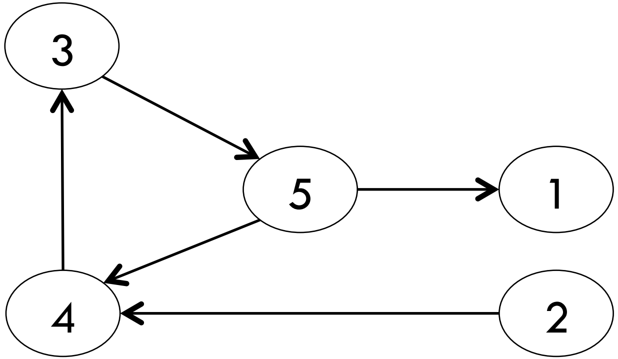
Desconexo



S-conexo



SF-conexo

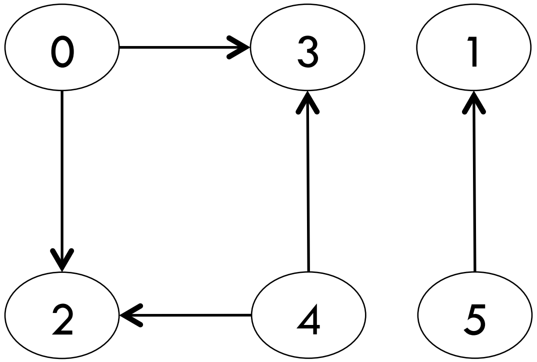


Conectividade em digrafos

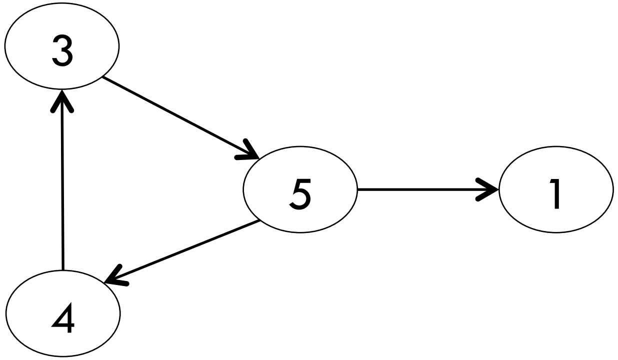
- Desconexo
  - Um grafo é não-conexo ou desconexo se nem todo par de vértices é unido por uma cadeia
- S-conexo
  - Um grafo é simplesmente conexo ou s-conexo se todo par de vértices é unido por pelo menos uma cadeia
- SF-conexo
  - Um grafo é semi-fortemente conexo ou sf-conexo se para todo par de vértice, pelo menos um deles é alcançável a partir do outro
- F-conexo
  - Um grafo é fortemente conexo ou f-conexo se todos os vértices são mutuamente alcançáveis

Conectividade em digrafos

Desconexo

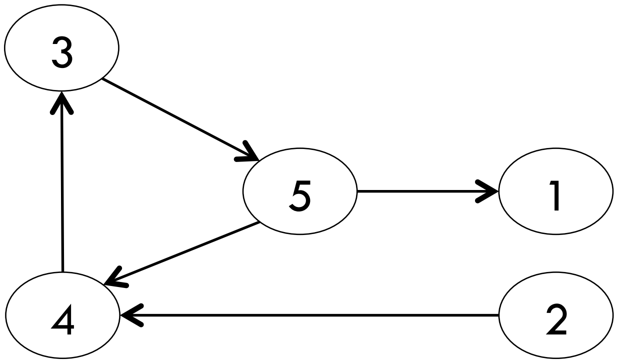


S-conexo

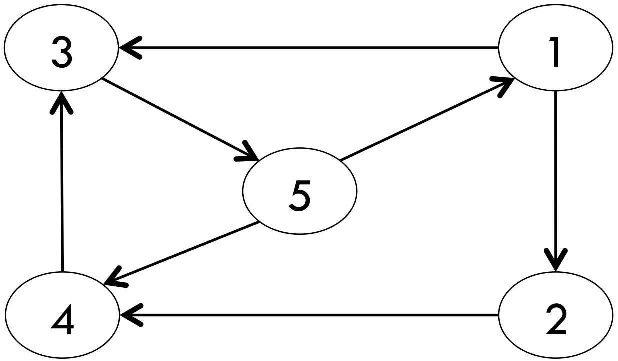


Conectividade em digrafos

SF-conexo

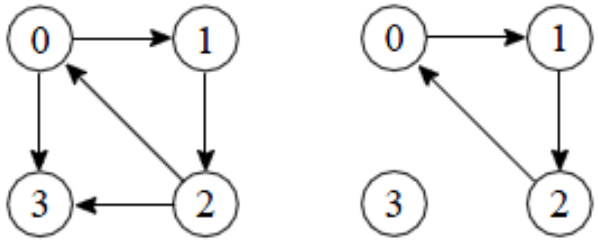


F-conexo



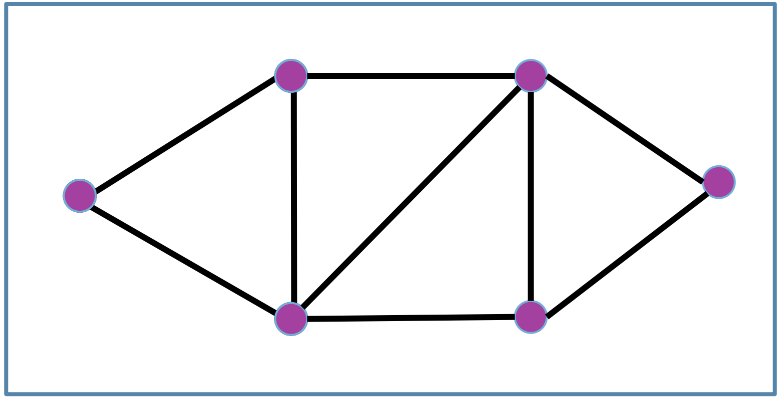
Componente fortemente conexo

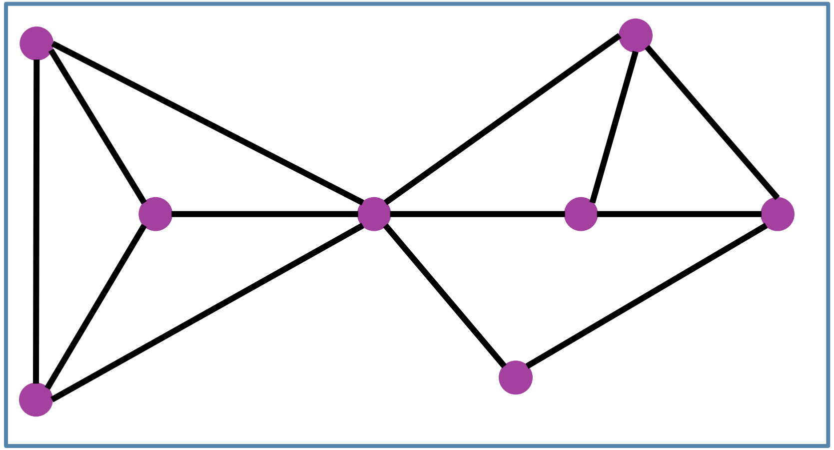
- Componente fortemente conexo de um digrafo G = (V, E):
  - **conjunto máximo de vértices $C_i \subseteq$ V** tal que;
    - **para todo par de vértices u** e **v** $\in$ **C_i**, temos que os vértices **u** e **v são acessíveis um a partir do outro**



Exercícios

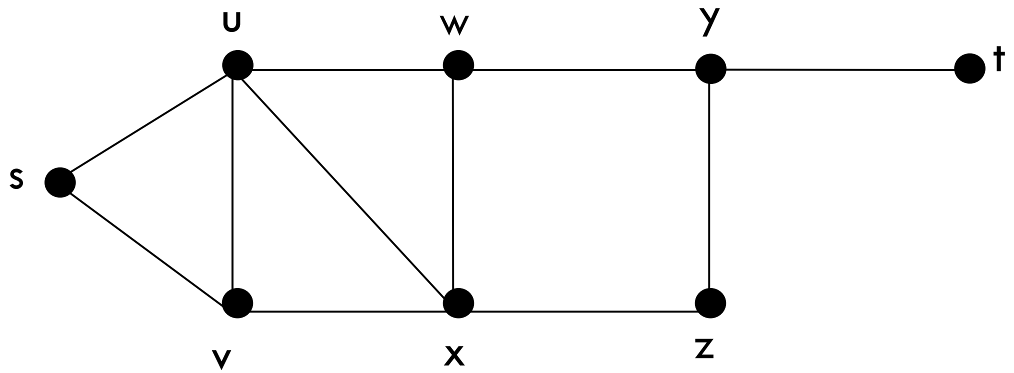
- Quais os valores de $\lambda(G)$ e $K(G)$ para os grafos abaixo? Eles são separáveis?





Exercícios

- Quais dos seguintes conjuntos de arestas são *cut-sets* do grafo G a seguir?



Exercícios

- Quais os valores de conectividade para
  - Grafos completos?