Teoria dos Grafos e Computabilidade

— Connectivity —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Laboratory of Image and Multimedia Data Science – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Feb 2025

Teoria dos Grafos e Computabilidade

— Connectivity —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Laboratory of Image and Multimedia Data Science – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Feb 2025

Busca em profundidade *Caminha no grafo visitando todos os seus vértices sempre procurando o vértice mais* ***profundo****.*

Caminhamento em grafos

Busca em profundidade *Caminha no grafo visitando todos os seus vértices sempre procurando o vértice mais profundo.*

Busca em largura *Expandir o conjunto de vértices de forma uniforme em que são visitados todos os vértices de mesma distância ao início antes de visitar outros níveis.*

G = (V,E) é um grafo dirigido

Fecho transitivo de $v \in V$

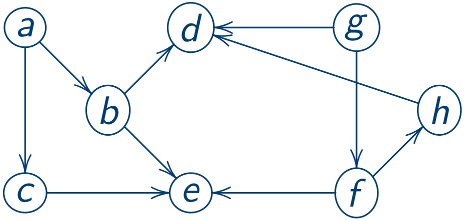
- direto: vértices **alcançáveis** de v, com caminho maior ou igual a zero

Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $v \in V$

- direto: vértices **alcançáveis** de v, com caminho maior ou igual a zero

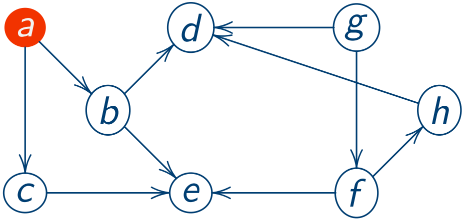


Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $v \in V$

- direto: vértices **alcançáveis** de v, com caminho maior ou igual a zero

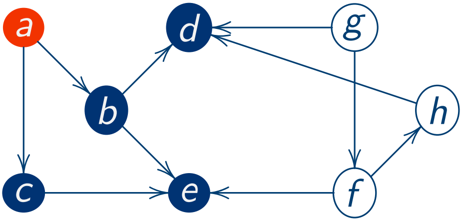


Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $v \in V$

- direto: vértices **alcançáveis** de v, com caminho maior ou igual a zero

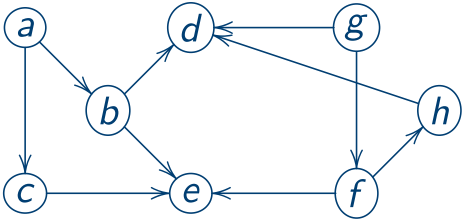


Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $v \in V$

- direto: vértices **alcançáveis** de v, com caminho maior ou igual a zero
- inverso: vértices que **alcançam** v, com caminho maior ou igual a zero

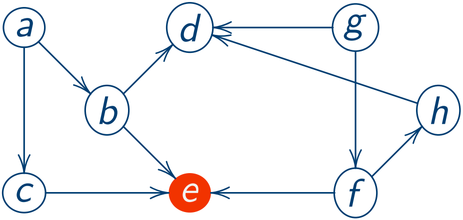


Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $v \in V$

- direto: vértices **alcançáveis** de v, com caminho maior ou igual a zero
- inverso: vértices que **alcançam** v, com caminho maior ou igual a zero

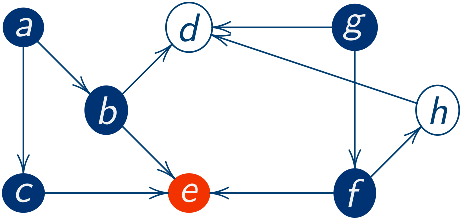


Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $v \in V$

- direto: vértices **alcançáveis** de v, com caminho maior ou igual a zero
- inverso: vértices que **alcançam** v, com caminho maior ou igual a zero



Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $X \subseteq V$

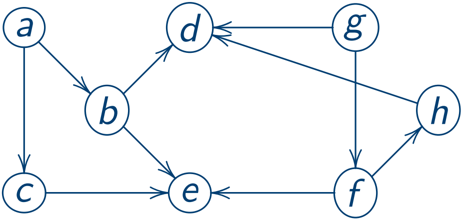
- direto: vértices **alcançáveis** de cada vértice de X, com caminho maior ou igual a zero

Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $X \subseteq V$

- direto: vértices **alcançáveis** de cada vértice de X, com caminho maior ou igual a zero

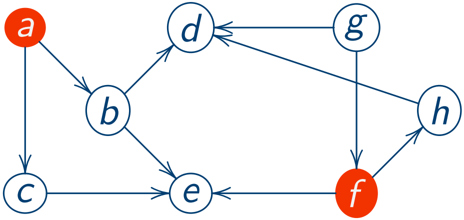


Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $X \subseteq V$

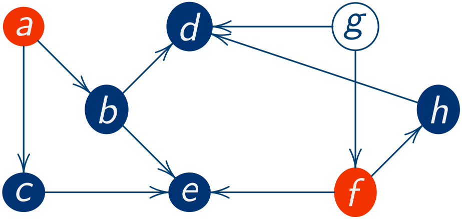
- direto: vértices **alcançáveis** de cada vértice de X, com caminho maior ou igual a zero



G = (V,E) é um grafo dirigido

Fecho transitivo de $X \subseteq V$

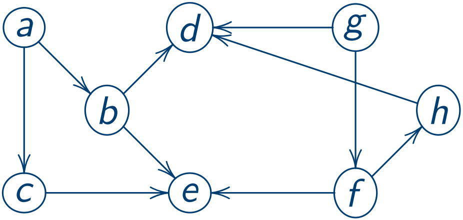
- direto: vértices **alcançáveis** de cada vértice de X, com caminho maior ou igual a zero



G = (V,E) é um grafo dirigido

Fecho transitivo de $X \subseteq V$

- direto: vértices **alcançáveis** de cada vértice de X, com caminho maior ou igual a zero
- inverso: vértices que **alcançam** algum vértice de X, com caminho maior ou igual a zero

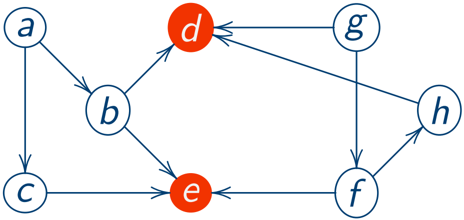


Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $X \subseteq V$

- direto: vértices **alcançáveis** de cada vértice de X, com caminho maior ou igual a zero
- inverso: vértices que **alcançam** algum vértice de X, com caminho maior ou igual a zero

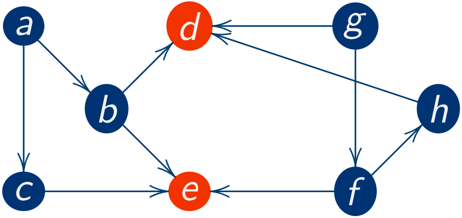


Fecho transitivo em grafos

G = (V,E) é um grafo dirigido

Fecho transitivo de $X \subseteq V$

- direto: vértices **alcançáveis** de cada vértice de X, com caminho maior ou igual a zero
- inverso: vértices que **alcançam** algum vértice de X, com caminho maior ou igual a zero

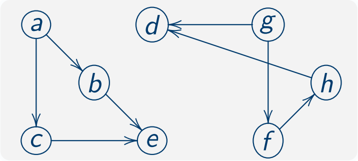


Conectividade em grafos direcionados

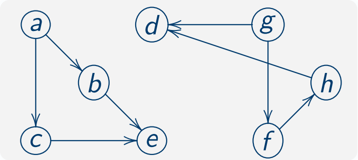
Um grafo é não-conexo ou desconexo se nem todo par de vértices é unido por uma cadeia

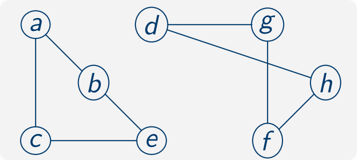
Conectividade em grafos direcionados

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

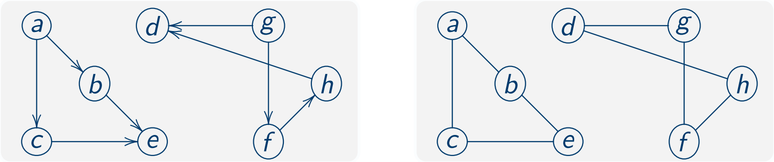


Um grafo é **não-conexo** ou desconexo se nem todo par de vértices é unido por uma cadeia



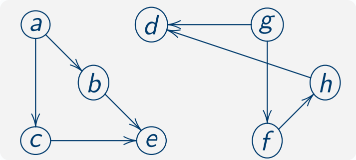


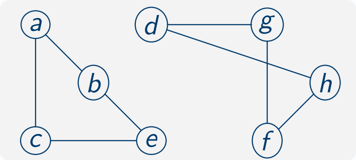
Um grafo é não-conexo ou desconexo se nem todo par de vértices é unido por uma cadeia



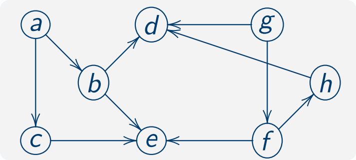
Um grafo é simplesmente conexo ou s-conexo se todo par de vértices é unido por pelo menos uma cadeia

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

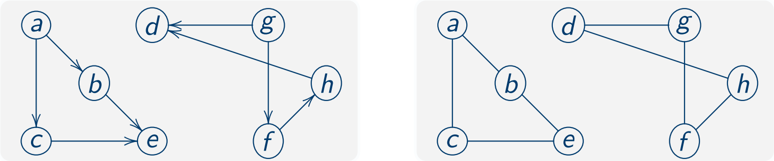




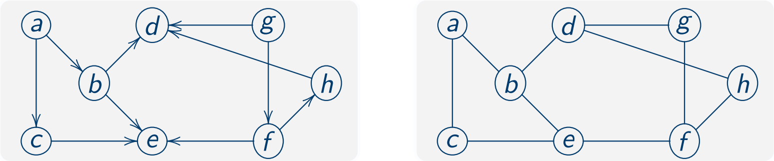
Um grafo é simplesmente conexo ou s-conexo se todo par de vértices é unido por pelo menos uma cadeia



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



Um grafo é simplesmente conexo ou s-conexo se todo par de vértices é unido por pelo menos uma cadeia

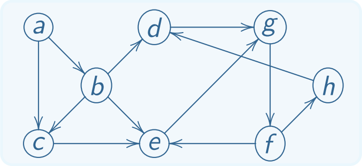


Conectividade em grafos direcionados

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

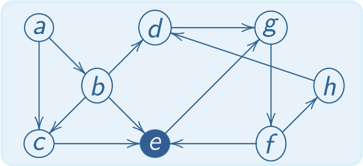
Conectividade em grafos direcionados

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



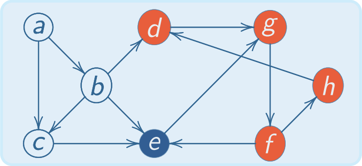
Conectividade em grafos direcionados

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



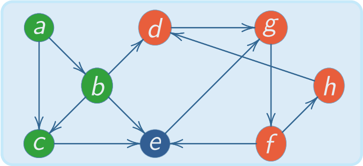
Conectividade em grafos direcionados

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



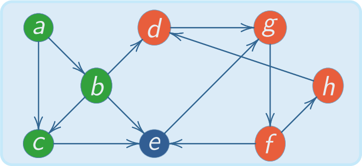
Conectividade em grafos direcionados

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 grafos direcionados

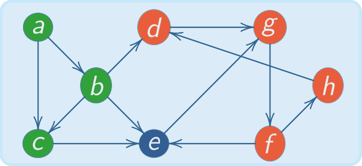
Um grafo é semi-fortemente conexo ou sf-conexo se para todo par de vértice, pelo menos um deles é alcançável a partir do outro



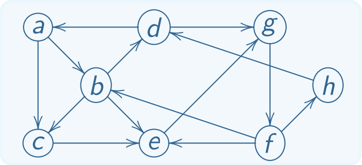
Um grafo é fortemente conexo ou f-conexo se todos os vértices são mutuamente alcançáveis

Conectividade em grafos direcionados

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

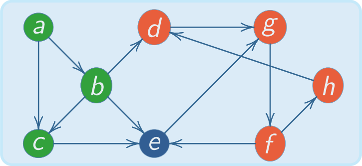


Um grafo é fortemente conexo ou f-conexo se todos os vértices são mutuamente alcançáveis

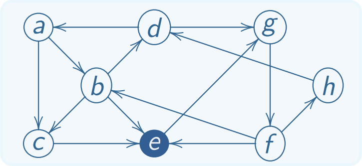


Conectividade em grafos direcionados

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

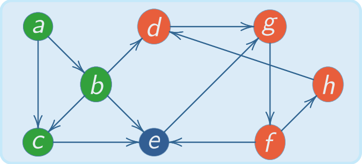


Um grafo é fortemente conexo ou f-conexo se todos os vértices são mutuamente alcançáveis

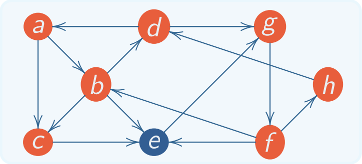


Conectividade em grafos direcionados

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



Um grafo é **fortemente conexo** ou f-conexo se todos os vértices são mutuamente alcançáveis



Conectividade

Categorias de conectividade

- C_0: Grafos desconexos não s-conexos
- C_1: Grafos s-conexos não sf-conexos
- C_2: Grafos sf-conexos não f-conexos
- C_3: Grafos f-conexos

Propriedade

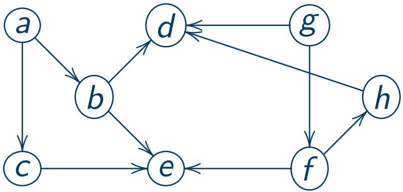
- Se $G \in C_0$ então $C(G) \in C_3$
- Se $G \in C_1$ então $C(G) \notin C_0$
- Se $G \in C_2$ então $C(G) \notin C_0$
- Se $G \in C_3$ então $C(G) \in Ci$ (i = 0,1,2,3)

Atingibilidade

Seja $G = (V,E)$ um grafo dirigido.

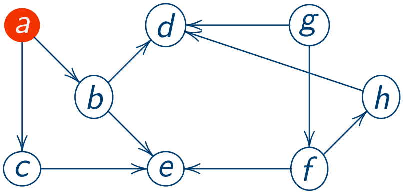
Seja G = (V,E) um grafo dirigido. Um subconjunto $B \subseteq V$ é uma base de G se

- não há caminho entre vértices de B
- todo vértice não pertencente a B pode ser atingido por algum vértice de B



Seja G = (V,E) um grafo dirigido. Um subconjunto $B \subseteq V$ é uma base de G se

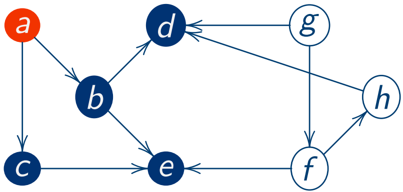
- não há caminho entre vértices de B
- todo vértice não pertencente a B pode ser atingido por algum vértice de B



Atingibilidade

Seja G = (V,E) um grafo dirigido. Um subconjunto $B \subseteq V$ é uma base de G se

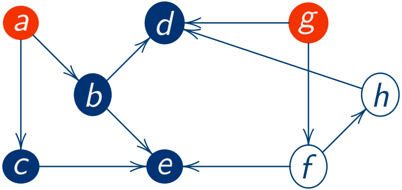
- não há caminho entre vértices de B
- todo vértice não pertencente a B pode ser atingido por algum vértice de B



Atingibilidade

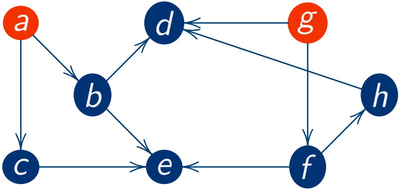
Seja G = (V,E) um grafo dirigido. Um subconjunto $B \subseteq V$ é uma base de G se

- não há caminho entre vértices de B
- todo vértice não pertencente a B pode ser atingido por algum vértice de B



Seja G = (V,E) um grafo dirigido. Um subconjunto $B \subseteq V$ é uma base de G se

- não há caminho entre vértices de B
- todo vértice não pertencente a B pode ser atingido por algum vértice de B

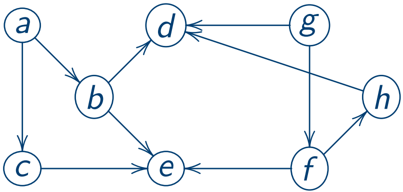


Seja $G = (V,E)$ um grafo dirigido.

Atingibilidade

Seja G = (V,E) um grafo dirigido. Um subconjunto $A \subseteq V$ é uma anti-base de G se

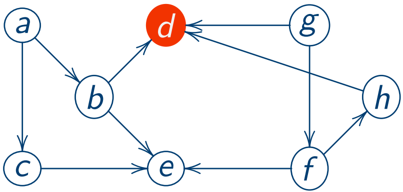
- não há caminho entre vértices de A
- todo vértice não pertencente a A pode atingir A por um caminho.



Atingibilidade

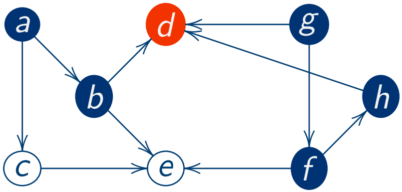
Seja G = (V,E) um grafo dirigido. Um subconjunto $A \subseteq V$ é uma anti-base de G se

- não há caminho entre vértices de A
- todo vértice não pertencente a A pode atingir A por um caminho.



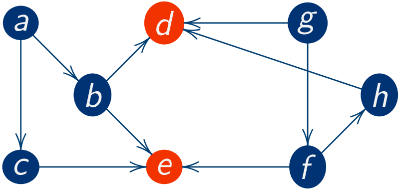
Seja G = (V,E) um grafo dirigido. Um subconjunto $A \subseteq V$ é uma anti-base de G se

- não há caminho entre vértices de A
- todo vértice não pertencente a A pode atingir A por um caminho.



Seja G = (V,E) um grafo dirigido. Um subconjunto $A \subseteq V$ é uma anti-base de G se

- não há caminho entre vértices de A
- todo vértice não pertencente a A pode atingir A por um caminho.



Atingibilidade

Sejam B uma base de G e A uma antibase de G

Raiz *Se B for um conjunto unitário, então dizemos que B é a RAIZ de G*

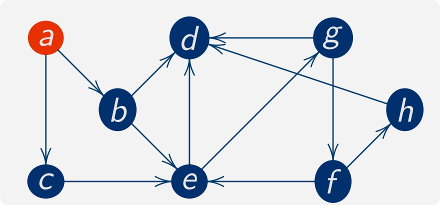
Anti-raiz *Se A for um conjunto unitário, então dizemos que A é a ANTIRAIZ de G*

Atingibilidade

Sejam B uma base de G e A uma antibase de G

Raiz *Se B for um conjunto unitário, então dizemos que B é a RAIZ de G*

Anti-raiz *Se A for um conjunto unitário, então dizemos que A é a ANTIRAIZ de G*

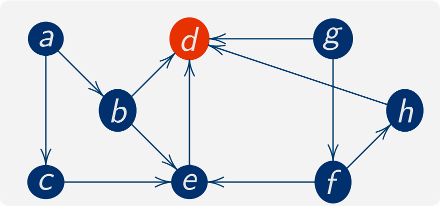


Atingibilidade

Sejam B uma base de G e A uma antibase de G

Raiz *Se B for um conjunto unitário, então dizemos que B é a RAIZ de G*

Anti-raiz *Se A for um conjunto unitário, então dizemos que A é a ANTIRAIZ de G*

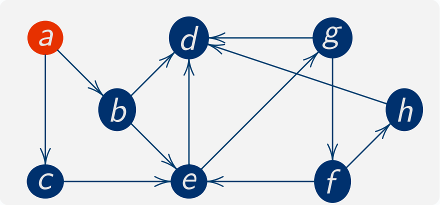


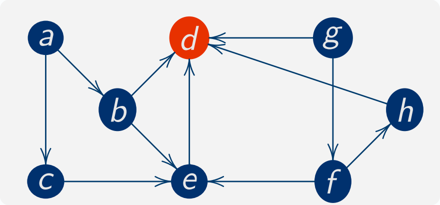
Atingibilidade

Sejam B uma base de G e A uma antibase de G

Raiz *Se B for um conjunto unitário, então dizemos que B é a RAIZ de G*

Anti-raiz *Se A for um conjunto unitário, então dizemos que A é a ANTIRAIZ de G*





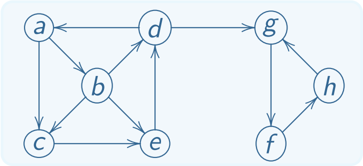
Teoria dos Grafos e Computabilidade

— Strongly connected componentes —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Laboratory of Image and Multimedia Data Science – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

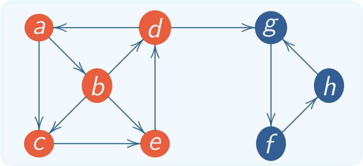
- Seja G = (V,E) um grafo dirigido



Componentes F-Conexas

- Seja G = (V,E) um grafo dirigido
- Considere a seguinte partição em V, em que cada S_i é f-conexo:

$$S = \{S_i | S_i \subseteq V, S_i \cap S_j = \emptyset, i, j = 1, ..., n\}$$

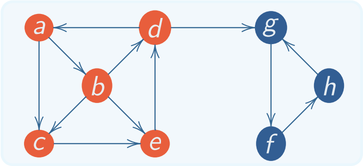


Componentes F-Conexas

- Seja G = (V,E) um grafo dirigido
- Considere a seguinte partição em V, em que cada S_i é f-conexo:

$$S = \{S_i | S_i \subseteq V, S_i \cap S_j = \emptyset, i, j = 1, ..., n\}$$

- Os elementos $S_i \in S$ são chamados de componentes f-conexas de G

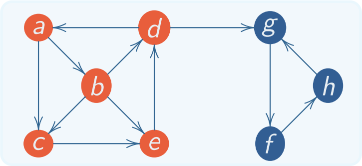


Componentes F-Conexas

- Seja G = (V,E) um grafo dirigido
- Considere a seguinte partição em V, em que cada S_i é f-conexo:

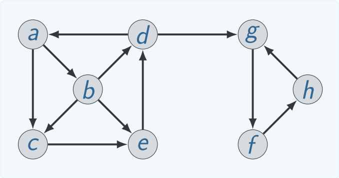
$$S = \{S_i | S_i \subseteq V, S_i \cap S_j = \emptyset, i, j = 1, ..., n\}$$

- Os elementos $S_i \in S$ são chamados de componentes f-conexas de G
- Se G for f-conexo, então S = V



Strongly connected components

- Let G = (V,E) be a directed graph

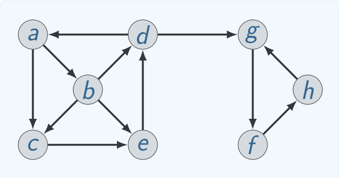


Strongly connected components

- Let G = (V,E) be a directed graph
- Let S be a partition of V in which

$$S = \{S_i | S_i \subseteq V, S_i \cap S_j = \emptyset, i, j = 1, ..., n\}$$

$$\bigcup_{i=1,...,n} S_i = V$$



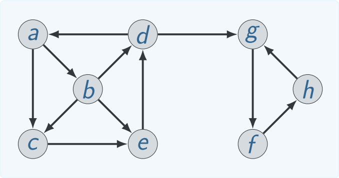
Strongly connected components

- Let G = (V,E) be a directed graph
- Let S be a partition of V in which

$$S = \{S_i | S_i \subseteq V, S_i \cap S_j = \emptyset, i, j = 1, ..., n\}$$

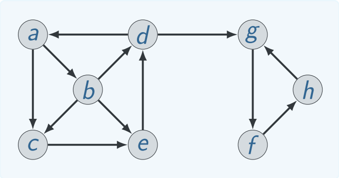
$$\bigcup_{i=1,...,n} S_i = V$$

- The induced subgraph of G by $S_i \in S$ is so-called strongly connected component if the subgraph is strongly connected.



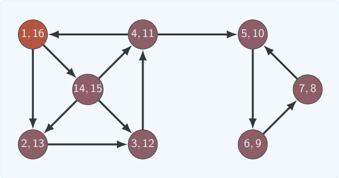
Strongly connected components

Let G = (V,E) be a directed graph



Let G = (V,E) be a directed graph

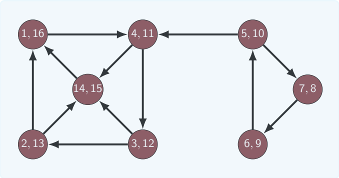
- Compute a depth-first search from a selected vertex indicating the discovery and last times



Strongly connected components

Let G = (V,E) be a directed graph

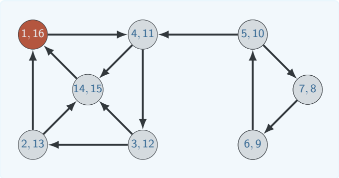
- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^{-} = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.



Strongly connected components

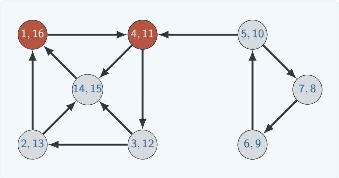
Let G = (V,E) be a directed graph

- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^{-} = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.
- Compute a depth-first search in the inversal order of last time



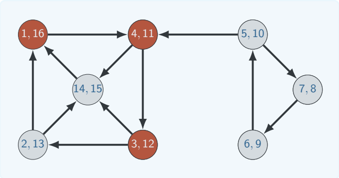
Let G = (V,E) be a directed graph

- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^{-} = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.
- Compute a depth-first search in the inversal order of last time



Let G = (V,E) be a directed graph

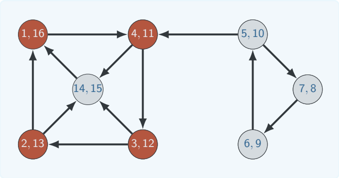
- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^- = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.
- Compute a depth-first search in the inversal order of last time



Strongly connected components

Let G = (V,E) be a directed graph

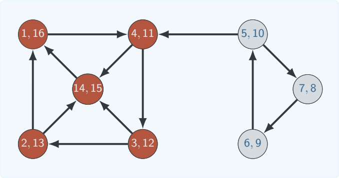
- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^- = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.
- Compute a depth-first search in the inversal order of last time



Strongly connected components

Let G = (V,E) be a directed graph

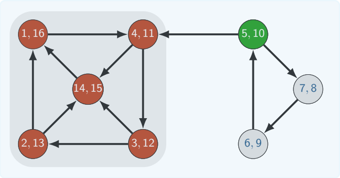
- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^- = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.
- Compute a depth-first search in the inversal order of last time



Strongly connected components

Let G = (V,E) be a directed graph

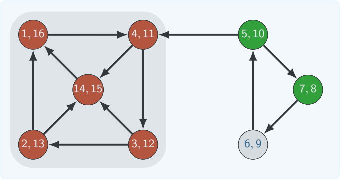
- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^{-} = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.
- Compute a depth-first search in the inversal order of last time



Strongly connected components

Let G = (V,E) be a directed graph

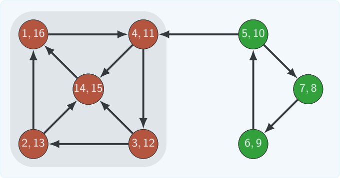
- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^{-} = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.
- Compute a depth-first search in the inversal order of last time



Strongly connected components

Let G = (V,E) be a directed graph

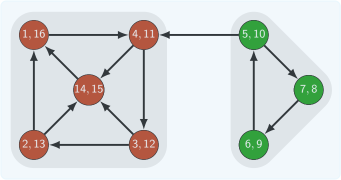
- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^{-} = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.
- Compute a depth-first search in the inversal order of last time



Strongly connected components

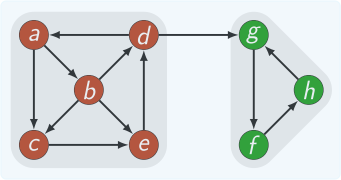
Let G = (V,E) be a directed graph

- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^- = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.
- Compute a depth-first search in the inversal order of last time



Let G = (V,E) be a directed graph

- Compute a depth-first search from a selected vertex indicating the discovery and last times
- Compute a symmetric graph $G^{-} = (V, E')$ in which there is an edge $e' \in E'$ if for each $e = (u, v) \in E$ then $e' = (v, u)$.
- Compute a depth-first search in the inversal order of last time



Teoria dos Grafos e Computabilidade

— Graph cut —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Laboratory of Image and Multimedia Data Science – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

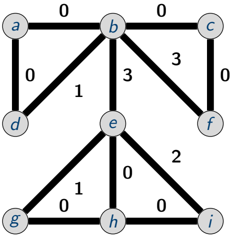
Feb 2025

Graph Cuts

- A **cut** (or cut-set) in a graph $G = (V, E)$ is a set of edges whose removal **disconnects** the graph (into two or more connected components).

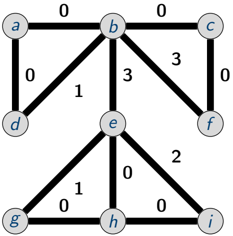
Graph Cuts

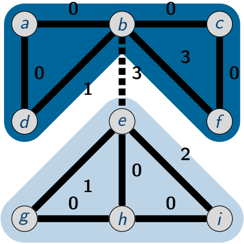
- A cut (or cut-set) in a graph $G = (V, E)$ is a set of edges whose removal **disconnects** the graph (into two or more connected components).



Graph Cuts

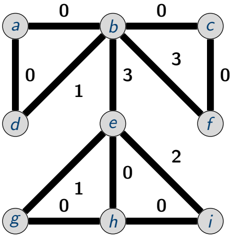
- A cut (or cut-set) in a graph $G = (V, E)$ is a set of edges whose removal **disconnects** the graph (into two or more connected components).

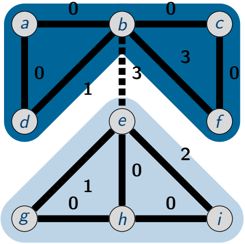




Graph Cuts

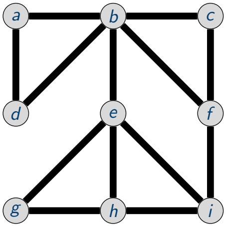
- A **cut** (or cut-set) in a graph $G = (V, E)$ is a set of edges whose removal **disconnects** the graph (into two or more connected components).
- Every set $S \subset V$ (S cannot be empty or the entire set V) has a corresponding cut: cut(S) is the set of edges (v, w) such that $v \in S$ and $w \in V - S$.





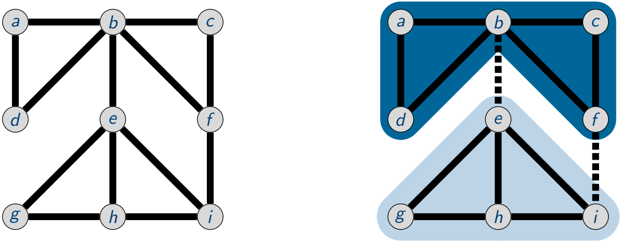
Graph Cuts

- A cut (or cut-set) in a graph $G = (V, E)$ is a set of edges whose removal **disconnects** the graph (into two or more connected components).
- Every set $S \subset V$ (S cannot be empty or the entire set V) has a corresponding cut: cut(S) is the set of edges (v, w) such that $v \in S$ and $w \in V - S$.



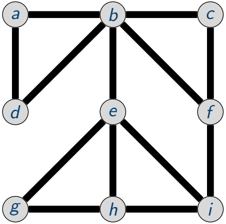
Graph Cuts

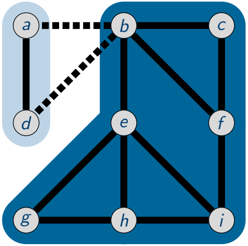
- A cut (or cut-set) in a graph $G = (V, E)$ is a set of edges whose removal **disconnects** the graph (into two or more connected components).
- Every set $S \subset V$ (S cannot be empty or the entire set V) has a corresponding cut: $\text{cut}(S)$ is the set of edges (v, w) such that $v \in S$ and $w \in V - S$.



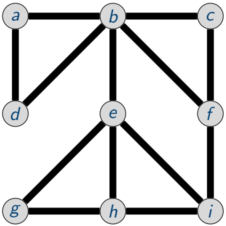
Graph Cuts

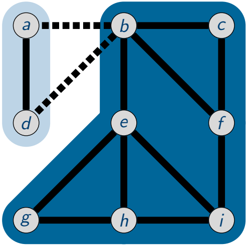
- A cut (or cut-set) in a graph $G = (V, E)$ is a set of edges whose removal **disconnects** the graph (into two or more connected components).
- Every set $S \subset V$ (S cannot be empty or the entire set V) has a corresponding cut: $\text{cut}(S)$ is the set of edges (v, w) such that $v \in S$ and $w \in V - S$.





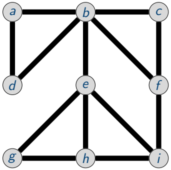
- A cut (or cut-set) in a graph $G = (V, E)$ is a set of edges whose removal **disconnects** the graph (into two or more connected components).
- Every set $S \subset V$ (S cannot be empty or the entire set V) has a corresponding cut: cut(S) is the set of edges (v, w) such that $v \in S$ and $w \in V - S$.
- cut(S) is a cut because deleting the edges in cut(S) disconnects S from $V - S$.



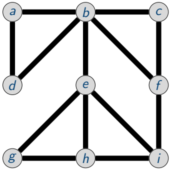


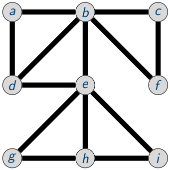
Connectivity and separability

- Edge-connectivity $\lambda(G)$ corresponds to the smallest number of edges of the graph in which their removal will disconnect the graph;

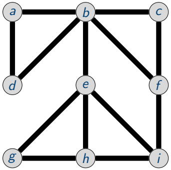


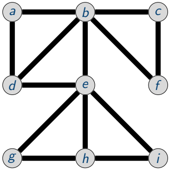
- Edge-connectivity $\lambda(G)$ corresponds to the smallest number of edges of the graph in which their removal will disconnect the graph;





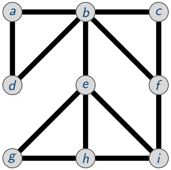
- Edge-connectivity $\lambda(G)$ corresponds to the smallest number of edges of the graph in which their removal will disconnect the graph;

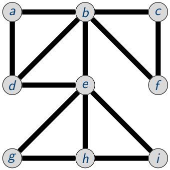




Connectivity and separability

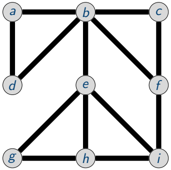
- Edge-connectivity $\lambda(G)$ corresponds to the smallest number of edges of the graph in which their removal will disconnect the graph;
- Vertex-connectivity $K(G)$ corresponds to the smallest number of vertices in which their removal will disconnect the graph;

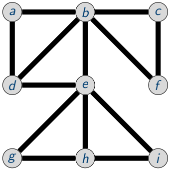




Connectivity and separability

- Edge-connectivity $\lambda(G)$ corresponds to the smallest number of edges of the graph in which their removal will disconnect the graph;
- Vertex-connectivity $K(G)$ corresponds to the smallest number of vertices in which their removal will disconnect the graph;

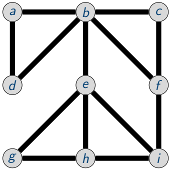


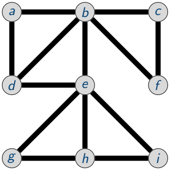


Connectivity and separability

- Edge-connectivity $\lambda(G)$ corresponds to the smallest number of edges of the graph in which their removal will disconnect the graph;
- Vertex-connectivity $K(G)$ corresponds to the smallest number of vertices in which their removal will disconnect the graph;

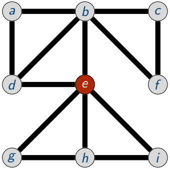
- K-connected graph is a graph with vertex-connectivity equal to K;
- Separable-graph is a graph with vertex-connectivity equal to 1.





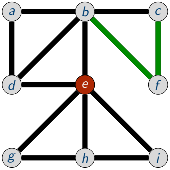
Connectivity and separability

- The vertex that disconnects the separable-graph is called articulation point (or cut-vertex)



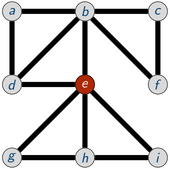
Connectivity and separability

- The vertex that disconnects the separable-graph is called articulation point (or cut-vertex)
- The edge-connectivity of a graph is smaller than or equal to the smallest degree between all vertex degree of the graph;



Connectivity and separability

- The vertex that disconnects the separable-graph is called articulation point (or cut-vertex)
- The edge-connectivity of a graph is smaller than or equal to the smallest degree between all vertex degree of the graph;
- The vertex-connecticity is smaller than or equal to the edge-connectivity;
- Let $\delta(G)$ be the smallest vertex degree of G. So, $K(G) \leq \lambda(G) \leq \delta(G)$.



Exemplo 1

Consider a network (transportation, computers, and so on) represented by a graph. How to measure the robustness of this network?

Robustness of a network – an example

Exemplo 1

Consider a network (transportation, computers, and so on) represented by a graph. How to measure the robustness of this network?

The robustness of a graph is based on the cut-sets.

Vulnerability of a network – an example

Exemplo 2

Let a private network containing eight computers. Consider that you have 16 lines to connect these computers. How to organize this network in order to decrease the vulnerability as low as possible?

Vulnerability of a network – an example

Exemplo 2

Let a private network containing eight computers. Consider that you have 16 lines to connect these computers. How to organize this network in order to decrease the vulnerability as low as possible?

How to model this problem as a graph problem?