Departamento de Sistemas e Computação – FURB
Curso de Ciência da Computação
Disciplina de Teoria dos Grafos

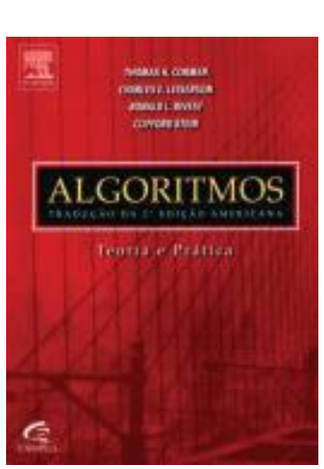
Conexidade

Prof. Aurélio Hoppe
aureliof@furb.br
http://www.inf.furb.br/~aurelio/

Grupo de Processamento de Imagens,
Análise de dados, Robótica e
Simulação computacional

Bibliografia

Márcia A. Rabuske. **Introdução à Teoria dos Grafos**. Editora da UFSC. 1992



Thomas Cormen et al. **Algoritmos**: teoria e prática. Ed. Campus. 2004.

Joan M. Aldous, Robin J. Wilson. **Graphs and Applications**: as introductory approach. Springer. 2001

Introdução

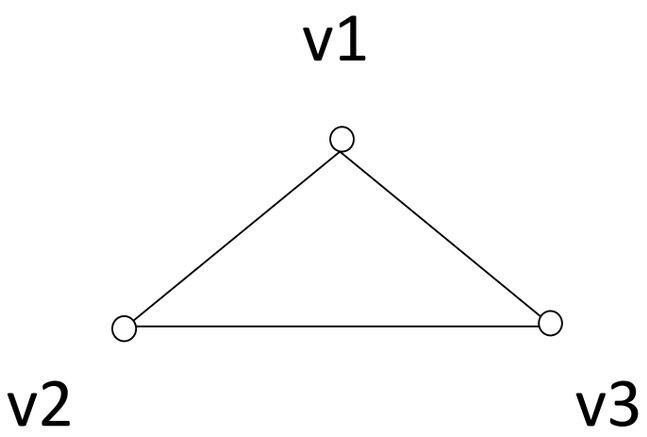
- A **conexidade** está relacionada a passagem de um vértice a outro em um grafo através de ligações existentes

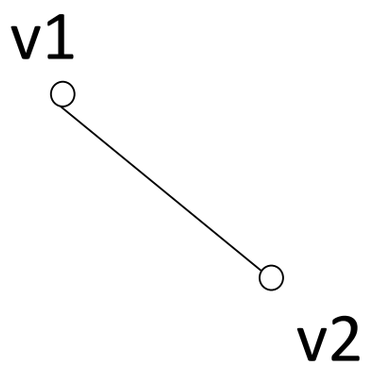
- Está passagem diz respeito a **atingibilidade**

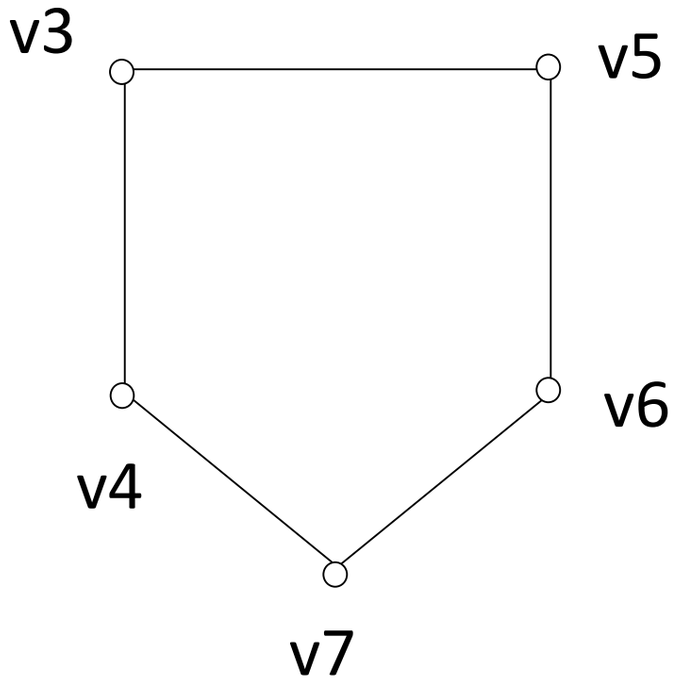
- Exemplo na prática?
  - Um vértice servidor pode enviar mensagens de dados para um determinado cliente?

Introdução

- quantos grafos há na figura abaixo?



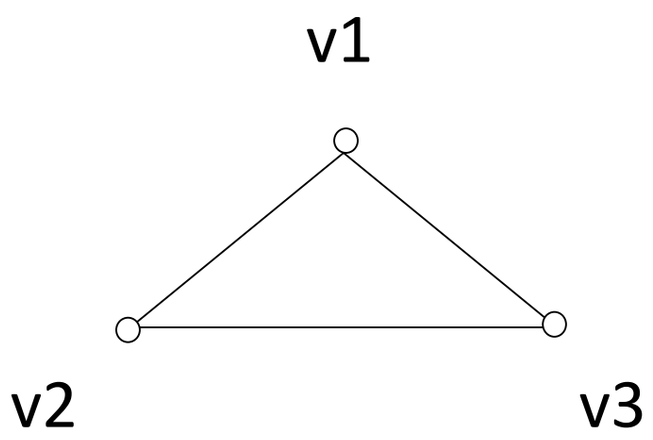




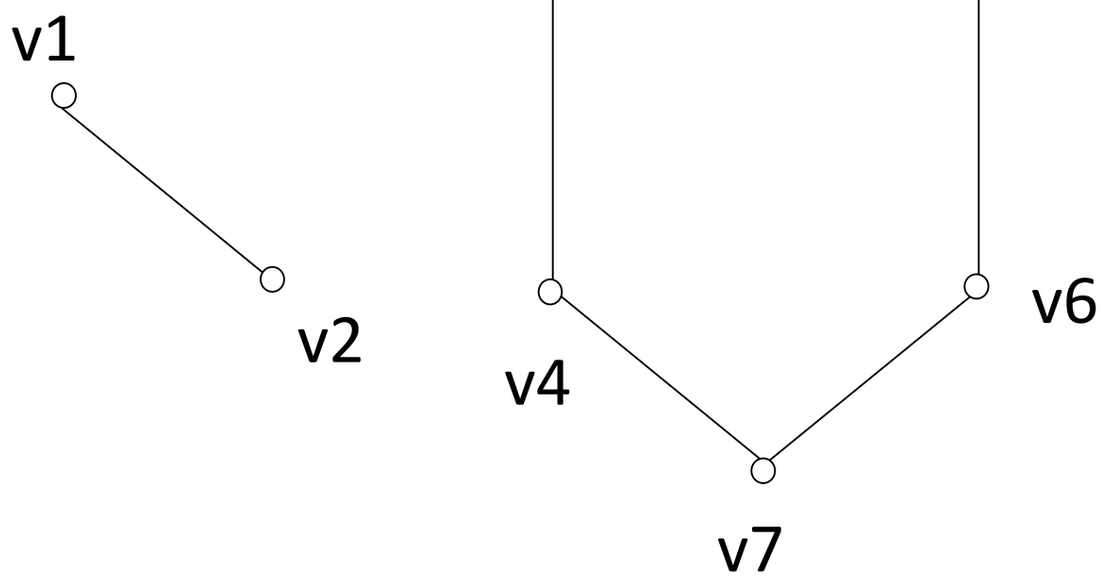
Definição 4.1 (Grafo conexo). Um grafo G é **conexo** se, para cada par de vértices u e v do grafo, existe pelo menos um caminho de u para v.

Definição 4.2 (Componente conexa). Uma **componente conexa** de um grafo G é um sub-grafo conexo de G

Conexo



Não-Conexo



Definições

- Um grafo não dirigido é “conexo” se para cada par de vértices u e v ∈ V, existe caminho entre u e v
- caso contrário, o grafo é “não-conexo”
- Cada subgrafo conexo de um grafo não conexo é chamado de “**componente conexa**” do grafo

Conexidade

Algoritmos para avaliação de conexidade em grafos não dirigidos:

- busca em largura
- busca em profundidade
- algoritmo de Goodman (Rabuske, 1992)
- estruturas de conjuntos (Cormen, 2004)

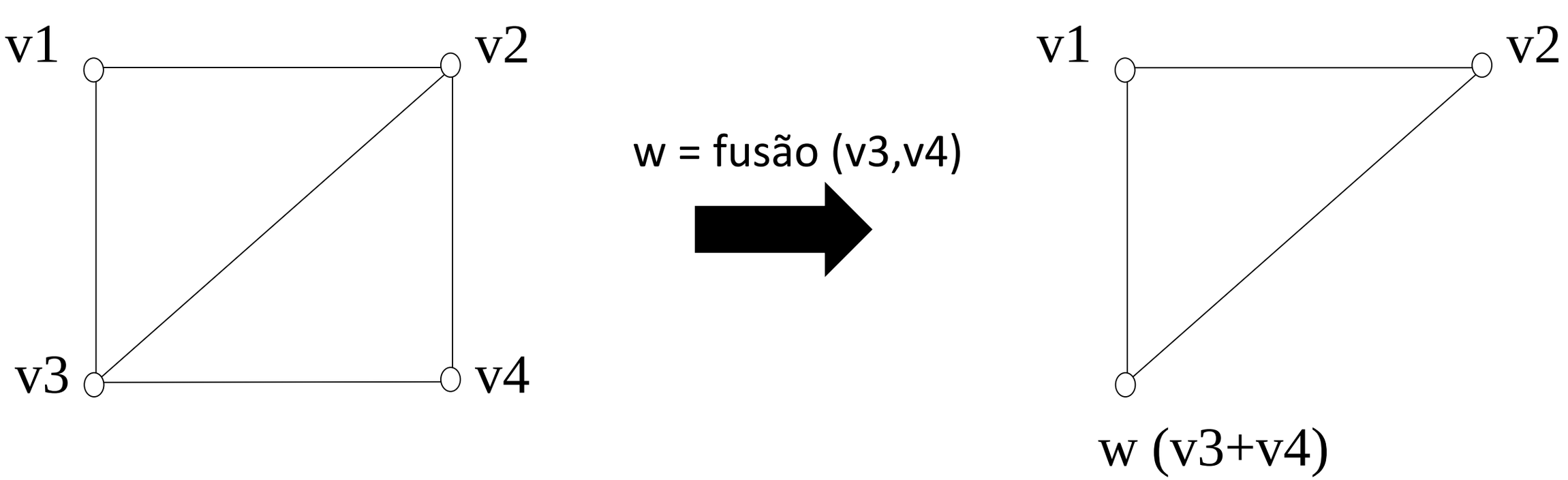
Algoritmo de Goodman

Ideia Geral

- o algoritmo realiza uma redução sequencial do grafo, por meio de fusão de vértices, até que cada componente conexa seja reduzida a um **único vértice**
- na fusão de dois vértices adjacentes u e v, a aresta (u,v) é eliminada, é criado um novo vértice w, adjacente a todos os vértices adjacentes a u e v antes da fusão. Os vértices u e v são removidos

Algoritmo de Goodman

- Fusão de vértices:

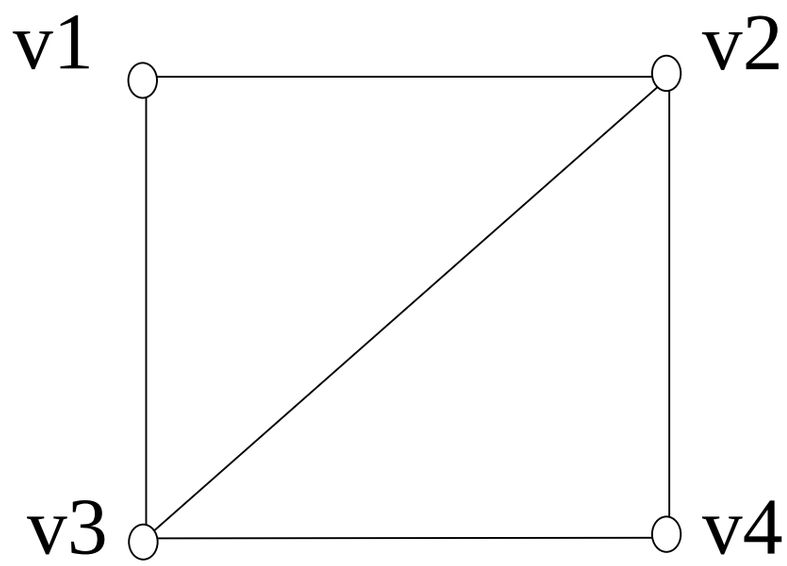


Algoritmo de Goodman

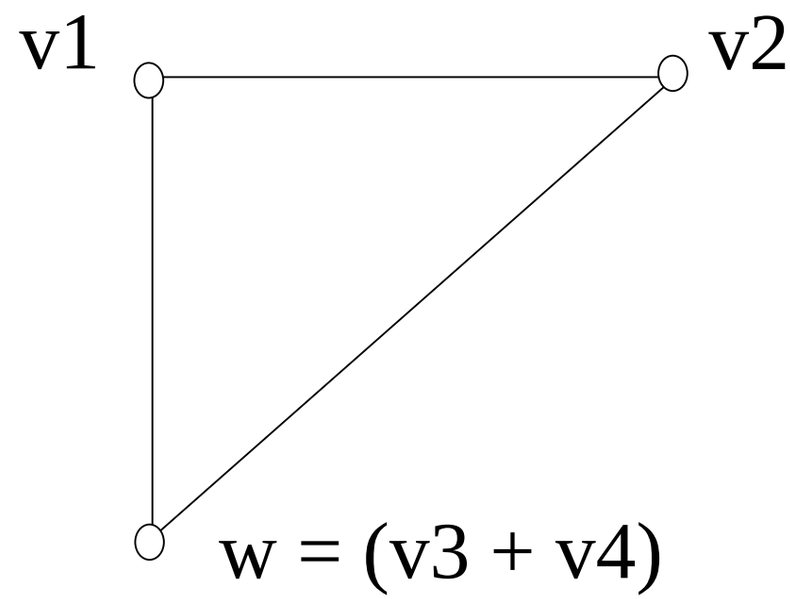
```
[Inicialização]
01. H = G;
02. C = 0;

[Gere a próxima componente conexa]
03. Enquanto ( H ≠ Ø )
04.     Selecione um vértice v pertencente a H
05.     Enquanto ( v for adjacente a algum vértice u ∈ H)
05.         w = grafo resultante da fusão de u com v
06.     Remova v, isto é, faça H = H - w
07.     c = c + 1

[Teste de conexidade]
08. Se ( c > 1 ) G é não conexo
09. Senão G é Conexo
```

Algoritmo de Goodman

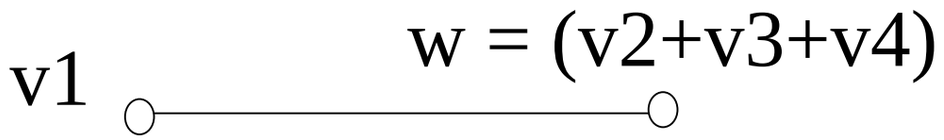
Estágio 1



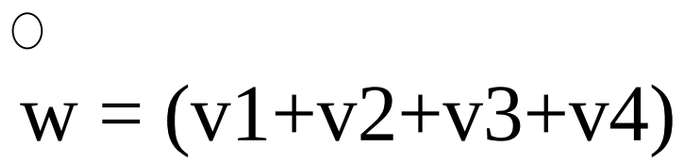
Estágio 2



Estágio 3



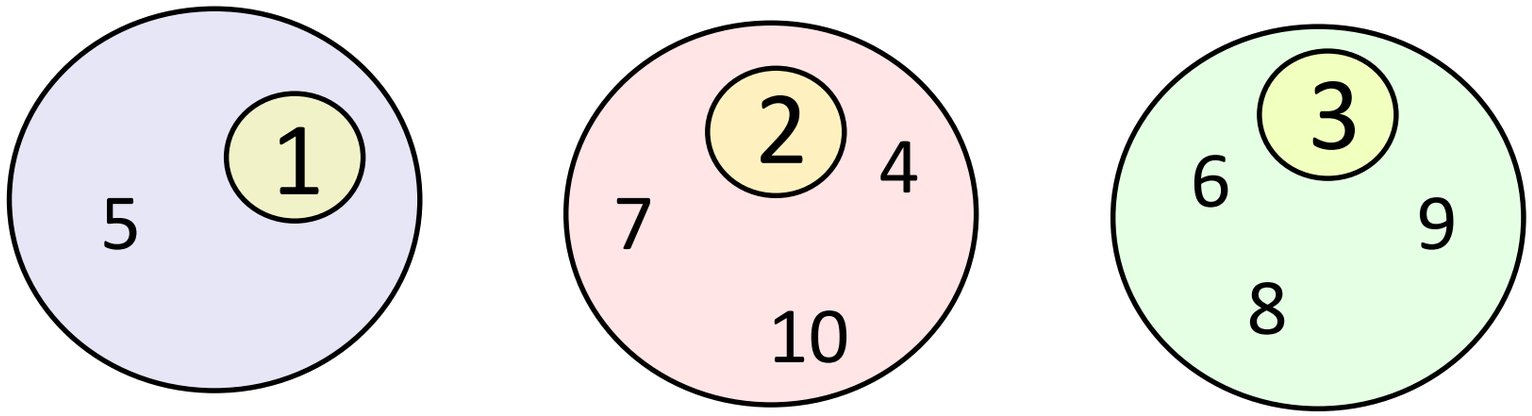
Estágio 4



Estrutura de conjuntos disjuntos

Algumas aplicações envolvem o agrupamento de n elementos em uma coleção de conjuntos disjuntos, ou seja, um particionamento dos elementos em conjuntos

- Cada conjunto C_k é identificado por um representante,que é um membro do conjunto



C	1	2	3	4	5	6	7	8	9	10
	1	2	3	2	1	3	2	3	3	2

Estrutura de conjuntos disjuntos

Operações:

`Make-Set(x):`	cria um novo conjunto cujo único elemento é apontado por x. x não pode pertencer a outro conjunto da coleção
`Union(x, y):`	executa a união dos conjuntos que contêm x e y, digamos Sx e Sy ,em um conjunto único. – $Sx \cap Sy = \emptyset$ – O representante de $S = Sx \cup Sy$ é um elemento de S
`Find-Set(x):`	retorna um ponteiro para o representante (único) do conjunto que contém x

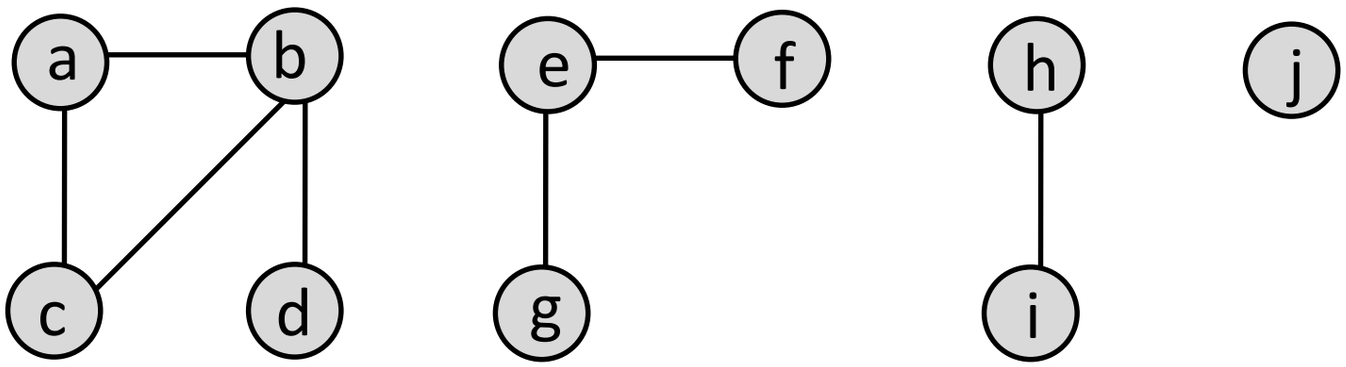
Estrutura de conjuntos disjuntos

Algoritmo:

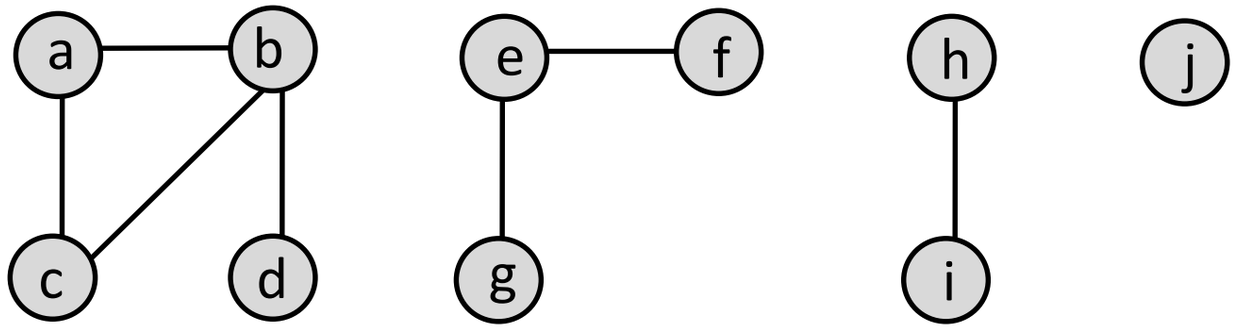
```
CONNECTED-COMPONENTS(G)
01. for each vertex v ∈ V[G]
02.     do MAKE-SET(v)
03. for each edge(u,v) ∈ E[G]
04.     do if FIND-SET(u) ≠ FIND-SET(v)
05.         then UNION(u,v)

SAME-COMPONENT(u,v)
01. if FIND-SET(u) = FIND-SET(V)
02.     then return TRUE
03.     else return FALSE
```

Estrutura de conjuntos disjuntos

```
CONNECTED-COMPONENTS(G)
01. for each vertex v ∈ V[G]
02.     do MAKE-SET(v)
03. for each edge(u,v) ∈ E[G]
04.     do if FIND-SET(u) ≠ FIND-SET(v)
05.         then UNION(u,v)

SAME-COMPONENT(u,v)
01. if FIND-SET(u) = FIND-SET(V)
02.     then return TRUE
03.     else return FALSE
```



Estrutura de conjuntos disjuntos



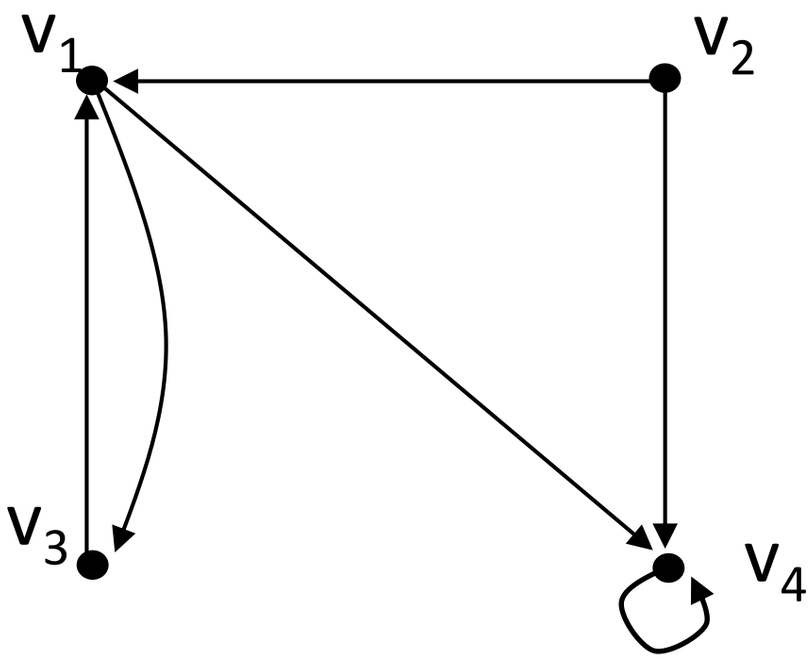

```
CONNECTED-COMPONENTS(G)
01. for each vertex v ∈ V[G]
02.     do MAKE-SET(v)
03. for each edge(u,v) ∈ E[G]
04.     do if FIND-SET(u) ≠ FIND-SET(v)
05.         then UNION(u,v)

SAME-COMPONENT(u,v)
01. if FIND-SET(u) = FIND-SET(V)
02.     then return TRUE
03.     else return FALSE
```

Edge processed	**Collection of disjoint sets**									
Initial sets	{a}	{b}	{c}	{d}	{e}	{f}	{g}	{h}	{i}	{j}
(b,d)	{a}	{b,d}	{c}		{e}	{f}	{g}	{h}	{i}	{j}
(e,g)	{a}	{b,d}	{c}		{e,g}	{f}		{h}	{i}	{j}
(a,c)	{a,c}	{b,d}			{e,g}	{f}		{h}	{i}	{j}
(h,i)	{a,c}	{b,d}			{e,g}	{f}		{h,i}		{j}
(a,b)	{a,b,c,d}				{e,g}	{f}		{h,i}		{j}
(e,f)	{a,b,c,d}				{e,f,g}			{h,i}		{j}
(b,c)	{a,b,c,d}				{e,f,g}			{h,i}		{j}

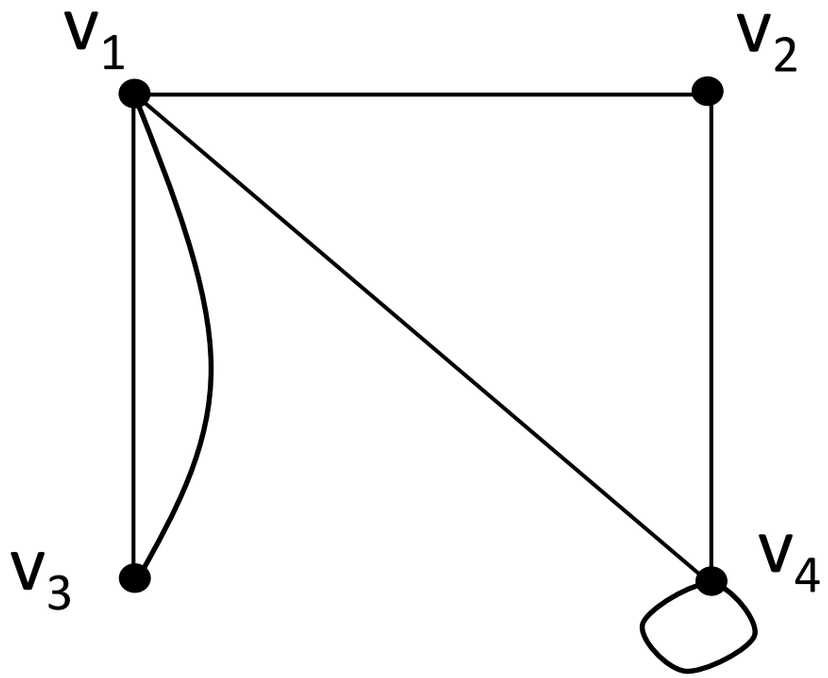
Digrafos - Conexidade

Definição 4.3 (Grafo subjacente). Dado um digrafo D, seu grafo subjacente é um grafo não dirigido obtido pela substituição de todas as arestas dirigidas de D por arestas não dirigidas

Definição 4.4 (Digrafo conexo). Um digrafo D é conexo se seu grafo subjacente for conexo.



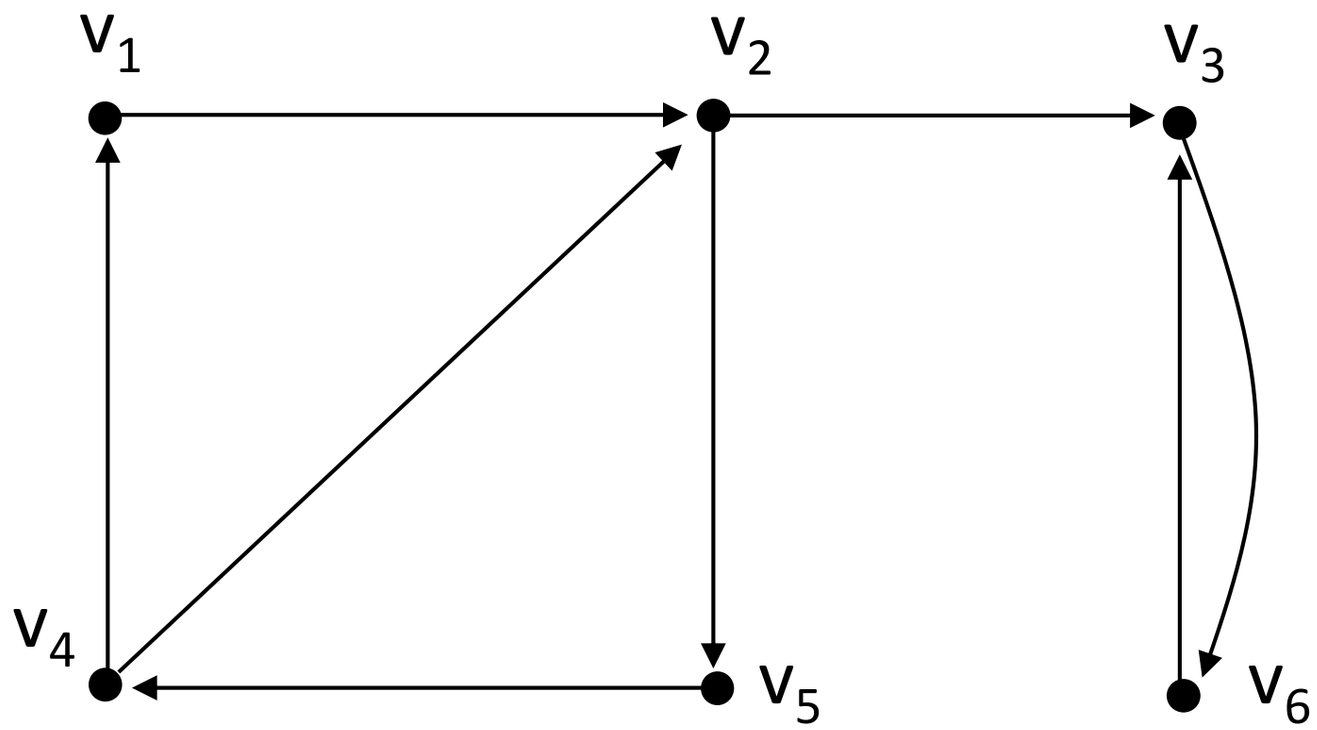
Digrafo D



Grafo subjacente D

Componentes fortemente conexas

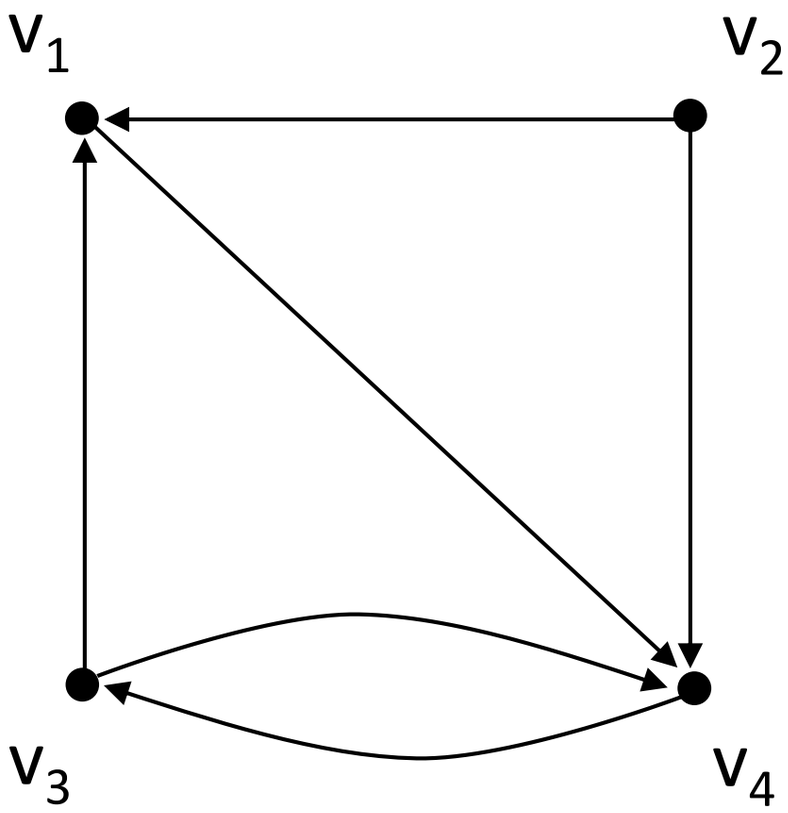
Definição 4.5 (Componente fortemente conexa). Uma componente fortemente conexa de um digrafo G=(V,E) é um conjunto maximal de vértices $C \subseteq V$ tal que para cada par de vértices u e v em C, existe caminho do vértice u para o vértice v e vice-versa (de v para u).



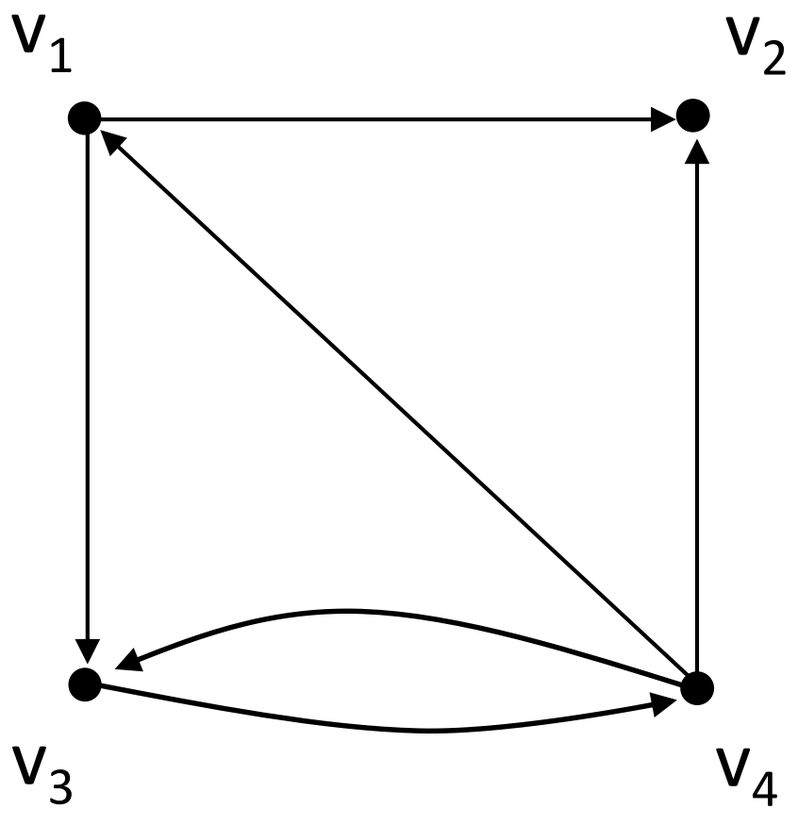
$R \cap Q$ | $R(v_i)$ – conjunto de vértices que podem ser atingidos
$Q(v_i)$ – vértices tomados como início de um caminho para atingir v_i

Componentes fortemente conexas

Definição 4.6 (Grafo transposto). Dado um digrafo G=(V,E), seu grafo transposto é definido por $G^T = (V,E^T)$, onde $E^T = \{(u,v) : (v,u) \in E\}$



Grafo G



Grafo G^T

Componentes fortemente conexas

Passo 1. Faça a pesquisa em profundidade em G e calcule o tempo de finalização em cada vértice u;

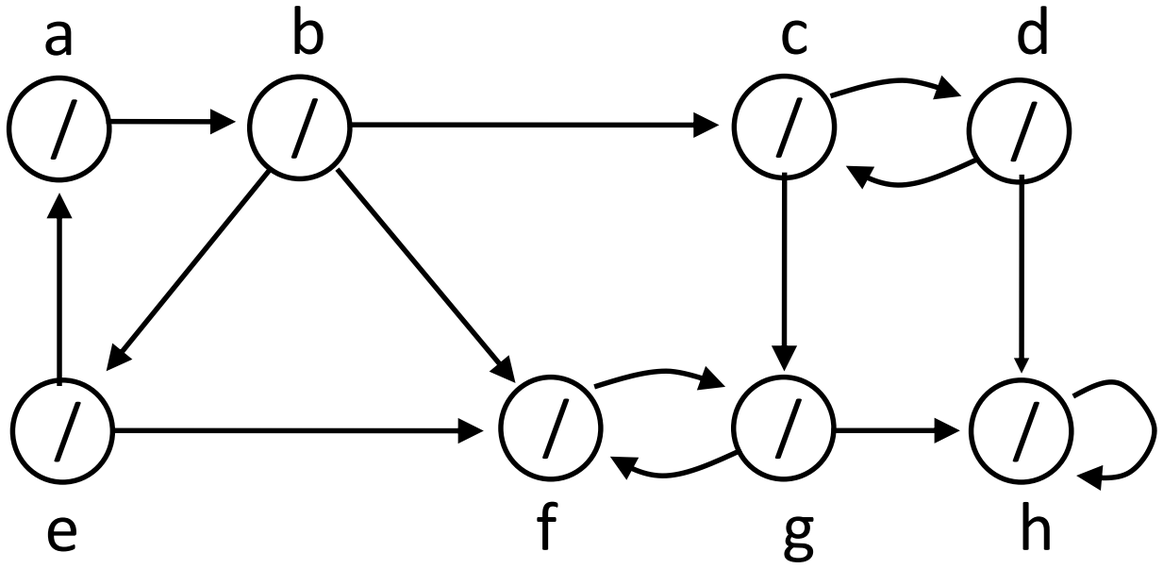
Passo 2. Gere o grafo transposto GT (grafo dual) do grafo G.

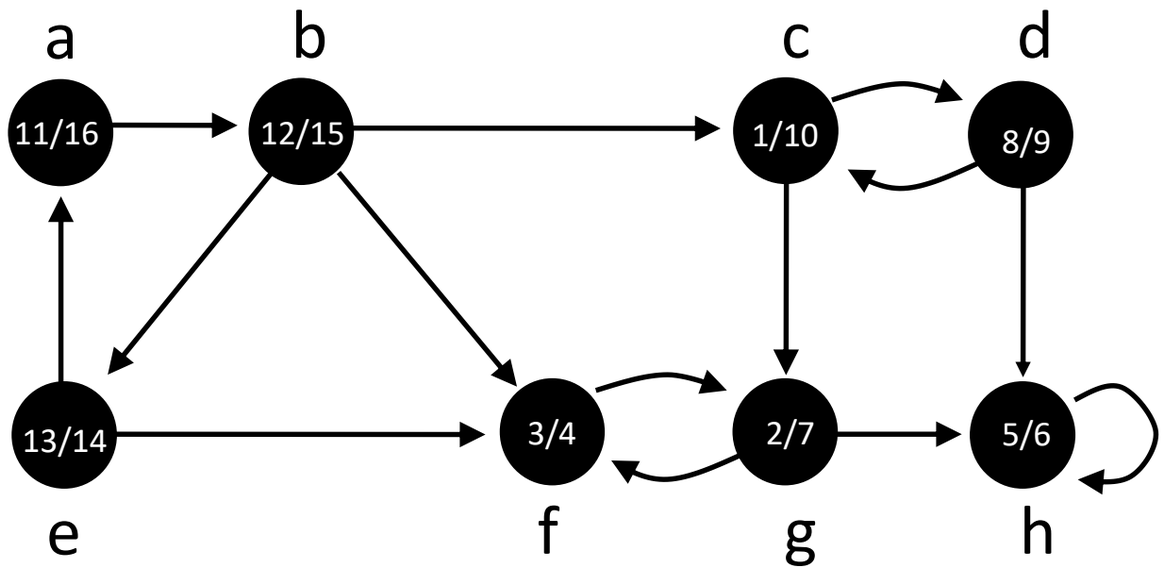
Passo 3. Faça a pesquisa em profundidade em GT, mas considerando os vértices acessíveis na ordem decrescente ao seu tempo de finalização encontrado no passo 1.

Cada floresta encontrada no passo 3, corresponde a um componente fortemente conexo de G

Componentes fortemente conexas

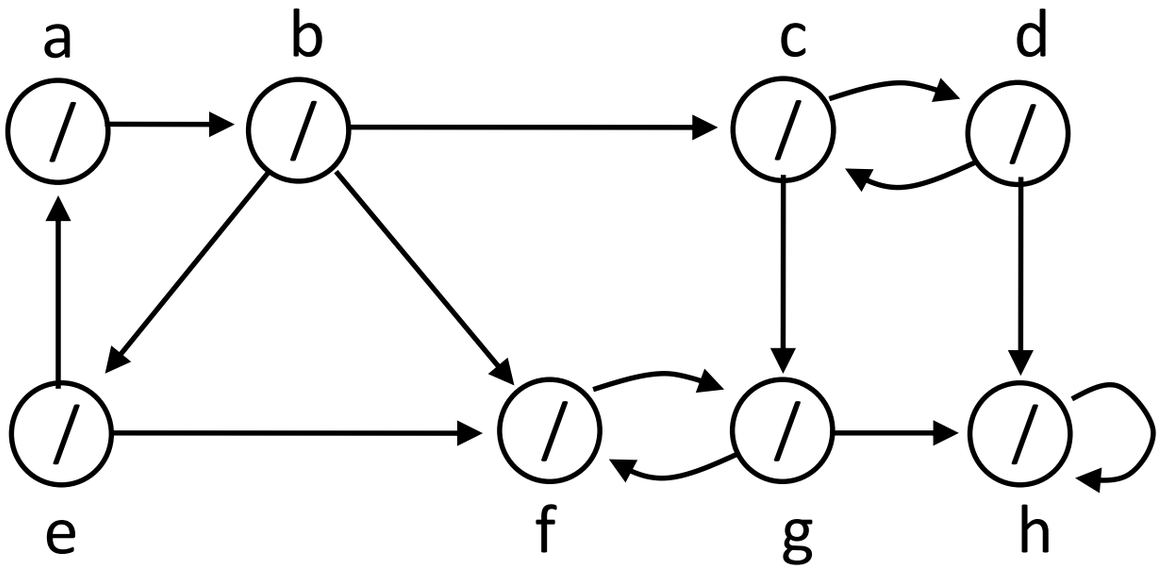
Passo 1. Faça a pesquisa em profundidade em G e calcule o tempo de finalização em cada vértice u



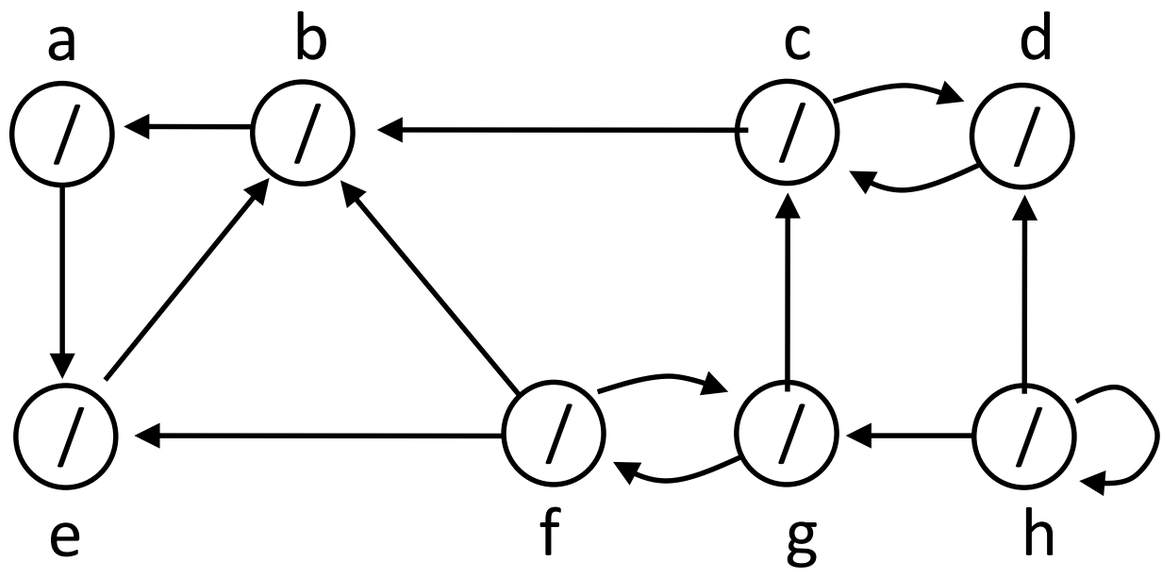


Componentes fortemente conexas

Passo 2. Gere o grafo transposto G^T (grafo dual) do grafo G

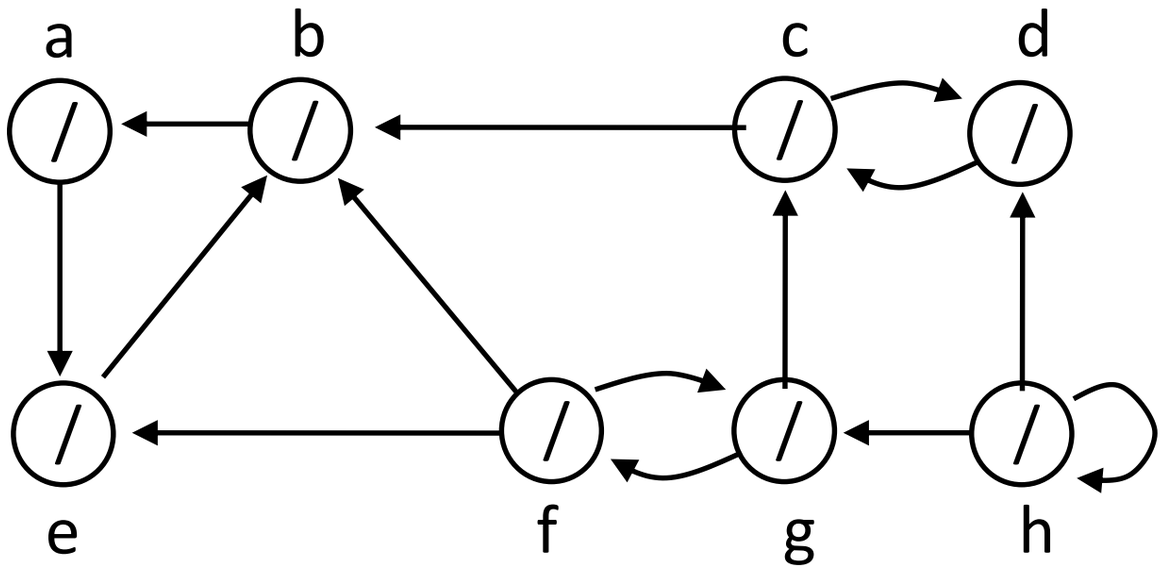


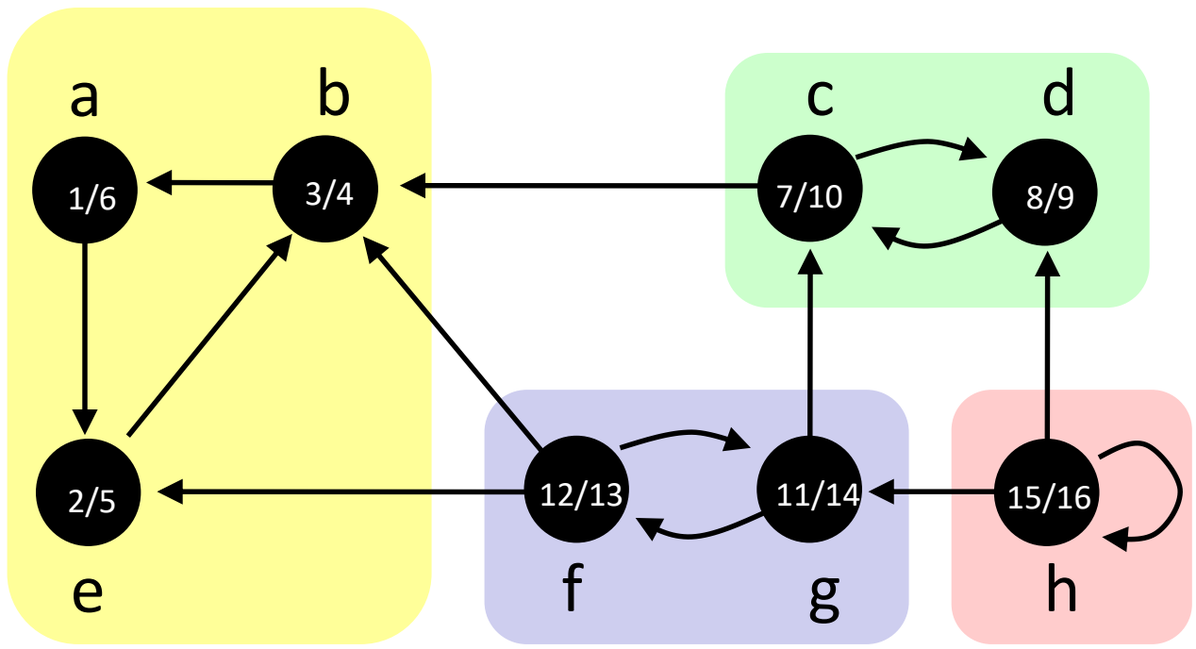
G^T



Componentes fortemente conexas

Passo 3. Faça a pesquisa em profundidade em G^T, mas considerando os vértices acessíveis na ordem decrescente ao seu tempo de finalização encontrado no passo 1

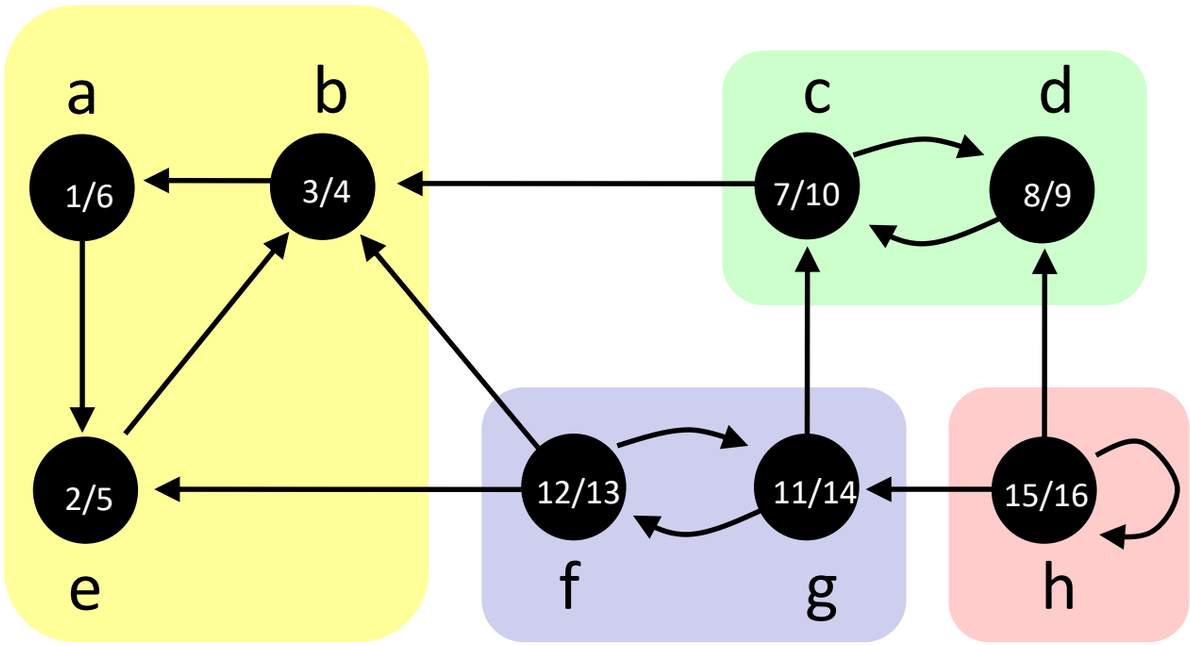


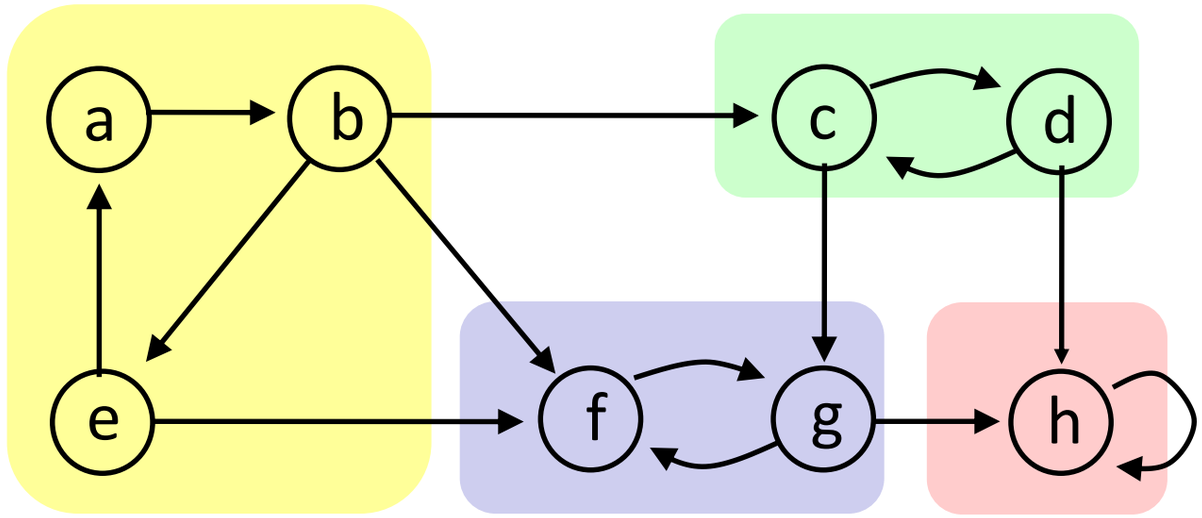


Lista em ordem decrescente ao tempo de finalização:
[a,b,e,c,d,g,h,f]

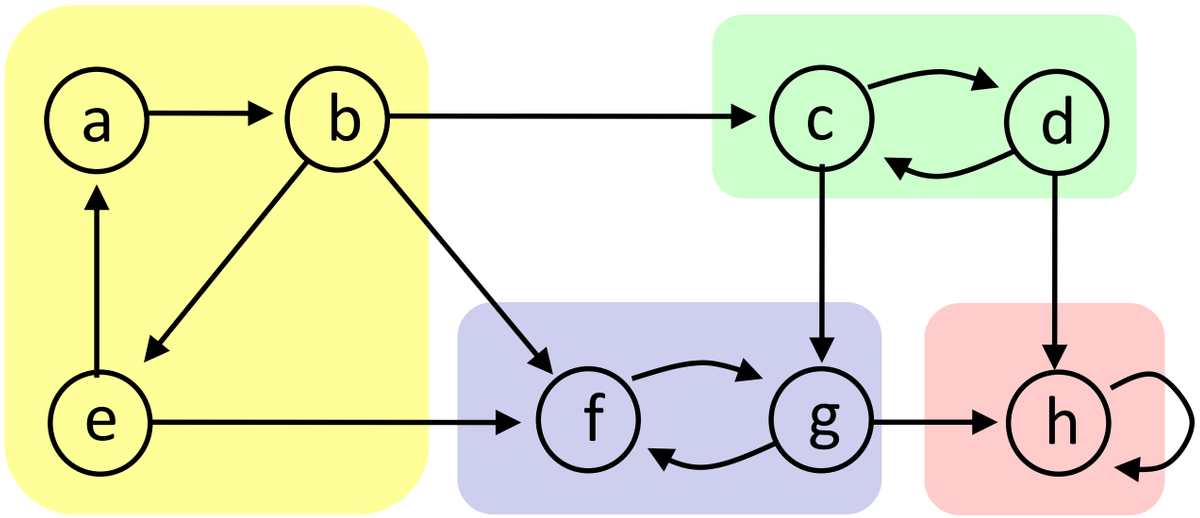
Componentes fortemente conexas

Cada floresta encontrada no passo 3, corresponde a um componente fortemente conexo de G

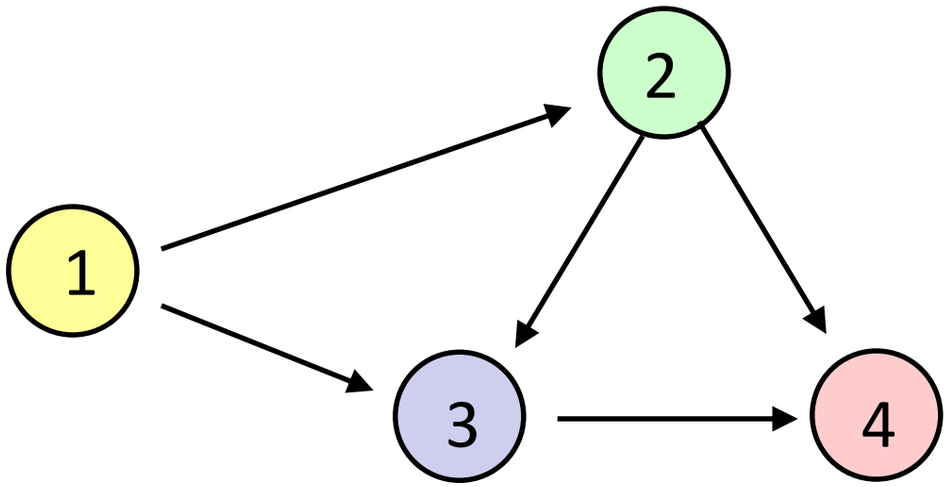




Redução por componentes fortemente conexos



Grafo Reduzido



Próxima aula...

CAMINHO MÍNIMO