DISCIPLINA: ESTRUTURA DE DADOS II

2018

UEA
UNIVERSIDADE DO ESTADO DO AMAZONAS

Ordenação Topológica

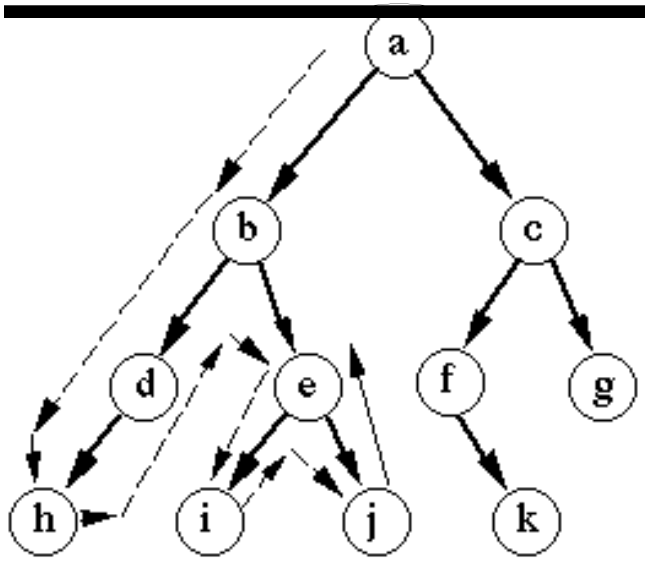
Prof. Luis Cuevas Rodríguez, PhD

EST
Escola Superior de Tecnologia da UEA

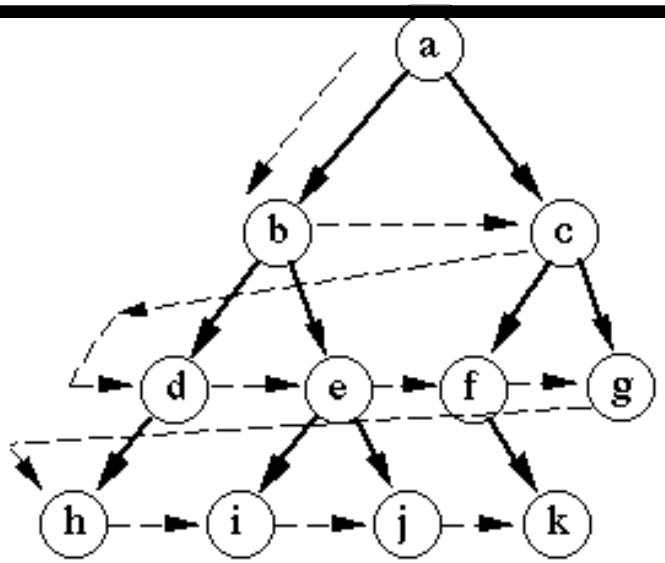
NUCOMP
NÚCLEO DE COMPUTAÇÃO

Algoritmos de caminhamento em grafos
(caminhamento, percurso, busca)

- Baseados em busca exaustiva
  - busca em profundidade (depth-first search - DFS)
  - busca em largura (breadth-first search – BFS)



Depth-first search



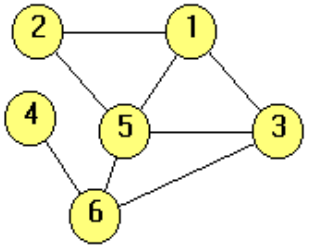
Breadth-first search

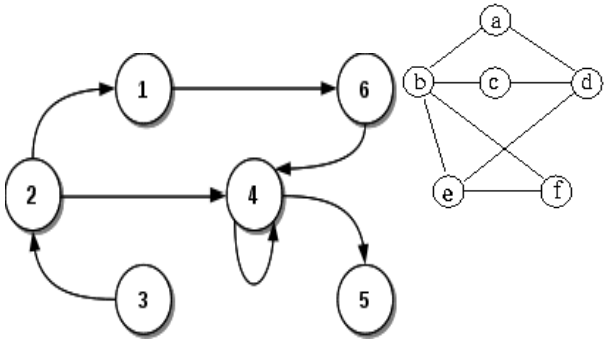
Algoritmos de caminhamento em grafos
(caminhamento, percurso, busca)

- Obter em G, dado um vértice v , o subgrafo alcançavel a partir de v
- Obter os elemento fortemente conexos em G
- testar se G é fortemente conexo
- encontrar um ciclo direcionado em G
- obter um caminho com o **menor número de arestas** entre um vértice inicial v e qualquer outro vértice alcançável no em G .

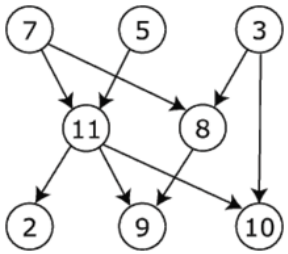
Grafos cíclicos

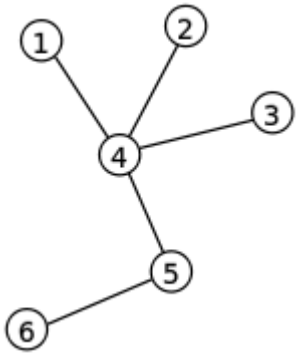
- um ciclo em um grafo é um caminho fechado sem vértices repetidos
- Grafos que tem pelo menos um ciclo ➔ **grafos cíclicos**



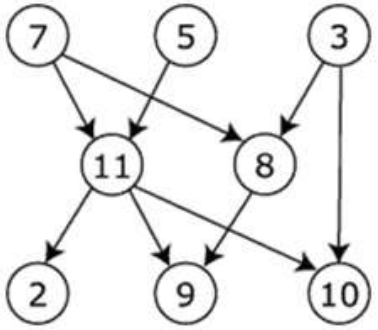


- Grafo que não tem ciclo ➔ **grafos acíclicos**





Grafos acíclicos dirigidos (DAG)



- DAG (directed acyclic graph)
- um grafo dirigido sem ciclo
- para qualquer vértice v, não há nenhuma ligação dirigida começando e acabando em v.
- Estes grafos aparecem em modelos onde não faz sentido que um vértice tenha uma ligação com si próprio.

Grafos acíclicos dirigidos (DAG)

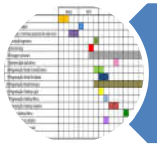
Organogramas

Arvores genealógicos

Redes de distribuição

Hierarquia de herança entre classes em orientação a objetos

Pré-requisitos entre disciplinas



Restrições de cronograma entre tarefas de um projeto

- Toda árvore direcionada é um grafo acíclico.
- Todo caminho num grafo acíclico é simples, não tem repetição de vértices.
- Como saber se um grafo é acíclico?

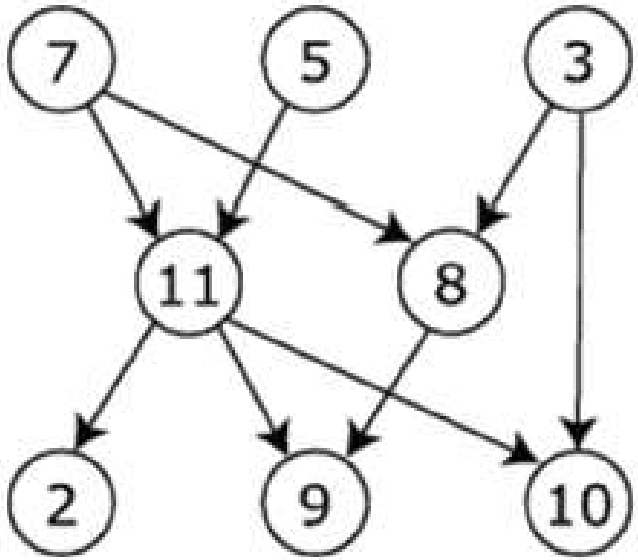
UEA
UNIVERSIDADE DO ESTADO DO AMAZONAS

Ordenação Topológica

- Ordenação linear de todos os vértices de um grafo G=(V, E), tal que
  - se G contém uma aresta **(u, v)**, então ***u*** aparece antes de ***v*** na ordenação
- Uma permutação dos vértices de um grafo (uma sequência em que cada vértice aparece uma só vez)
- O grafo tem que ser acíclico, se não impossível uma ordenação linear. Todo DAG admite ordenação topológica

Exemplo

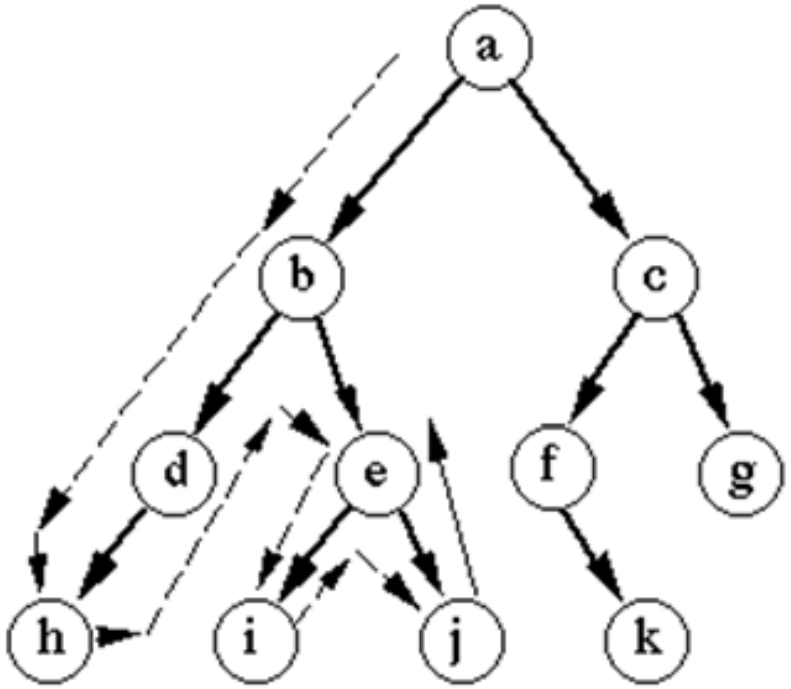
Diversas ordenações topológicas possiveis



- 7, 5, 3, 11, 8, 2, 9, 10 (visual esquerda-para-direita, de-cima-para-baixo)
- 3, 5, 7, 8, 11, 2, 9, 10 (vértice de menor número disponível primeiro)
- 3, 7, 8, 5, 11, 10, 2, 9
- 5, 7, 3, 8, 11, 10, 9, 2 (menor número de arestas primeiro)
- 7, 5, 11, 3, 10, 8, 9, 2 (vértice de maior número disponível primeiro)
- 7, 5, 11, 2, 3, 8, 9, 10

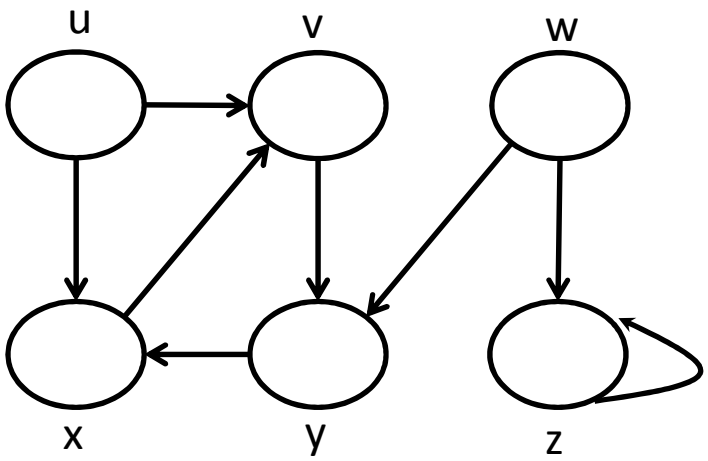
Busca em profundidade

- Busca em profundidade (Depth-first Search ou DFS): **buscar mais fundo no grafo sempre que é possível**
- Na busca, identificar cada no com dois carimbo de tempo
  - Descoberto pela primeira vez, d[u]
  - Termina de examinar sua lista de adjacência, f[u]

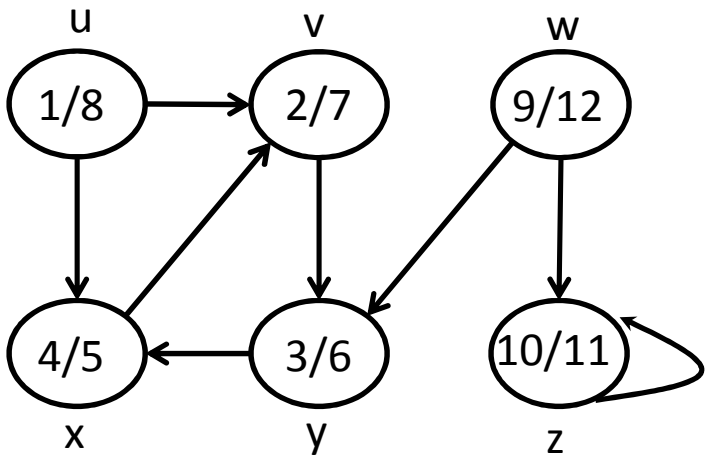


Depth-first search

Busca em profundidade – Carimbo de Tempo

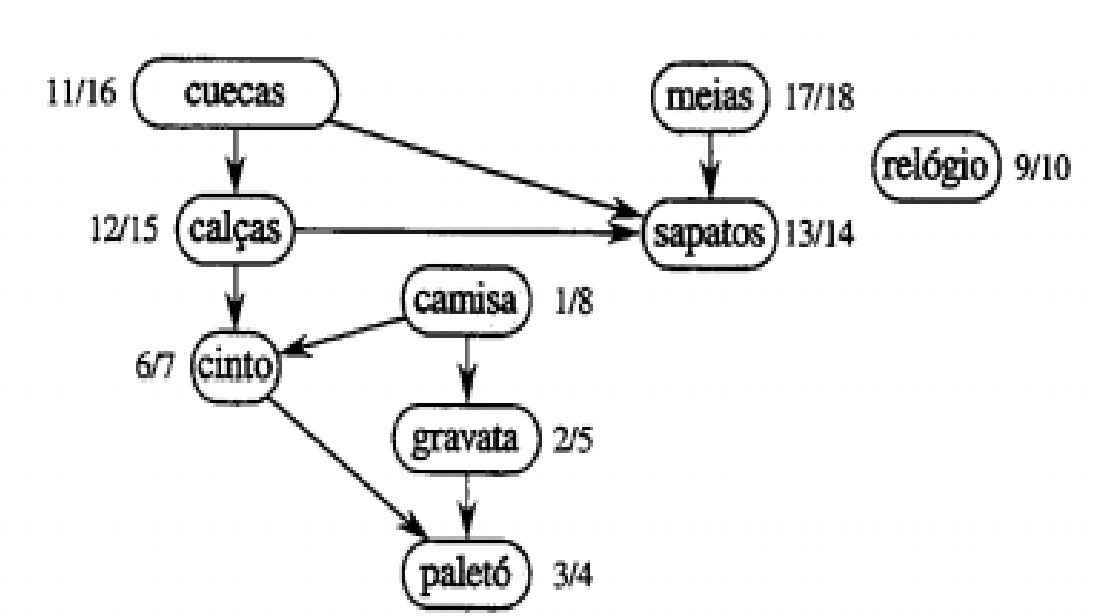


	u	v	w	x	y	z
d	1	2	9	4	3	10
f	8	7	12	5	6	11

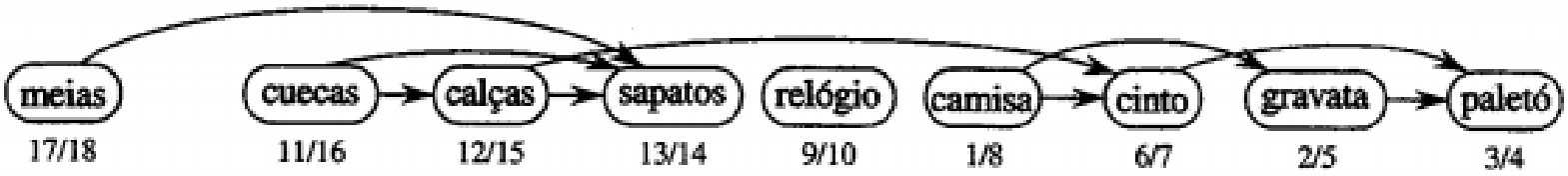


UEA
UNIVERSIDADE DO ESTADO DO AMAZONAS

Ordenação Topológica



1. Calcular os tempos para cada vértice.
2. A medida que vai terminando cada vértice inserir ele à frente de uma lista.
3. Retornar a lista



Ordenação Topológica

TOPOLOGICAL-SORT(G)
1 chamar DFS(G) para calcular o tempo de término $f[v]$ para cada vértice v
2 à medida que cada vértice é terminado, inserir o vértice à frente de uma lista ligada
3 **return** a lista ligada de vértices

- Complexidade da busca em profundidade DFS(G) é O(V+E)
- O(1) para inserir vértice na lista
- O(1) retorna a lista

Complexidade **O(V+E)**

COMPONENTES FORTEMENTE CONECTADOS

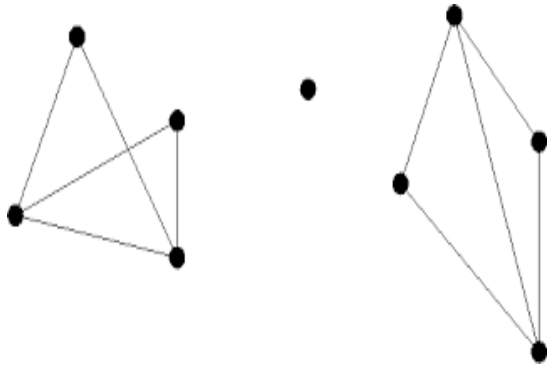
Prof. Luis Cuevas Rodríguez, PhD

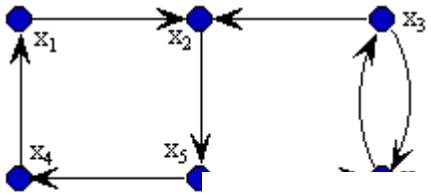
UEA
UNIVERSIDADE
DO ESTADO DO
AMAZONAS

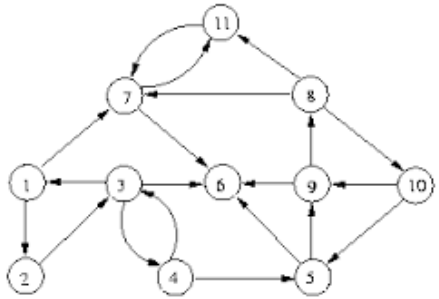
Grafo conexo

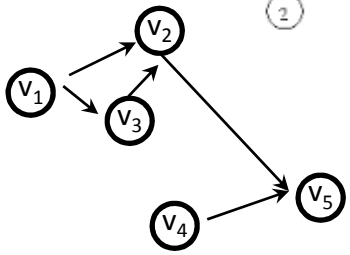
Grafo **conexo**

- ***Não dirigido***: se cada par de vértices nele estiver conectado por um caminho. Não esta quebrado
- ***Dirigido***
  - **Fortemente conexo**: existe um caminho entre qualquer par de vértices nas duas direções.
  - **Conexo**: qualquer par de vértice existe um caminho em uma das direções.
  - **Fracamente conexo**: não tem caminho entre dois vértices, mas se suas arestas são substituídas por não direcionadas produz um grafo conexo.





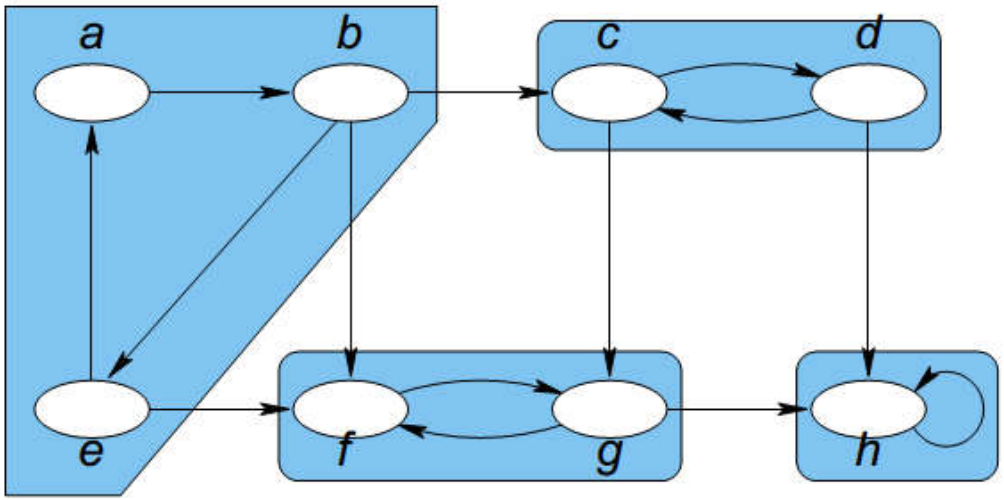




UEA
UNIVERSIDADE DO ESTADO DO AMAZONAS

Componentes fortemente conectados

- Dos vértices de G, ***u*** e ***v***, estão fortemente conexos sem eles são acessíveis um a partir do outro.



Componentes fortemente conectados

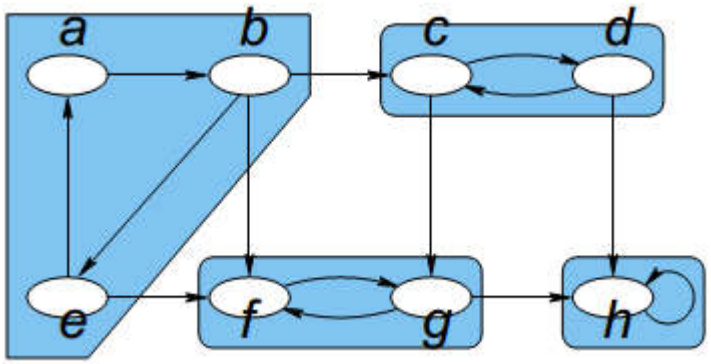
Componentes fortemente conectados

- Verificar se um grafo é fortemente conexo.
  - verificar, para cada par ***u, v*** de vértices, se existe um caminho de ***u*** a ***v*** e um caminho de ***v*** a ***u***
  - Algoritmo complexo $O(V^3+E)$

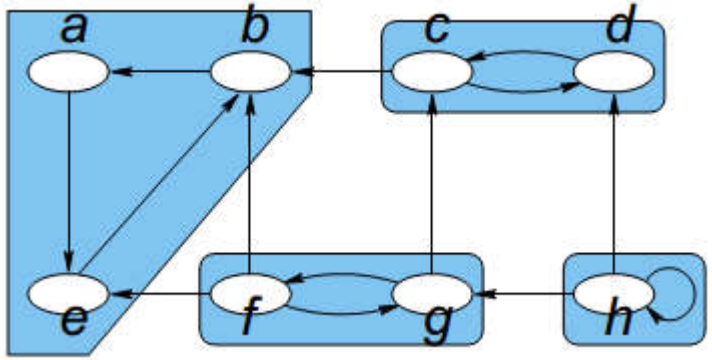
- Decomposição de um grafo dirigido em seus componentes fortemente conexas (Em um tempo razoável)

Componentes fortemente conectados

- Transposta de G é $G^T=(V,E^T)$. E^T são as arestas de G mas com sentido invertido.
- G e G^T tem os mesmos elementos fortemente conectados.



G



G^T

Componentes fortemente conectados

- Algoritmo baseado em:
  - Busca em profundidade.
  - Carimbo de tempo para cada vértice.
  - Trasposta do grafo G que esta no analises.
- O algoritmo calcula os componentes fortemente conectados de G usando duas Buscas em profundidade (DFS) uma sobre G e outra sobre G^T

Componentes fortemente conectados

STRONGLY-CONNECTED-COMPONENTS(G)

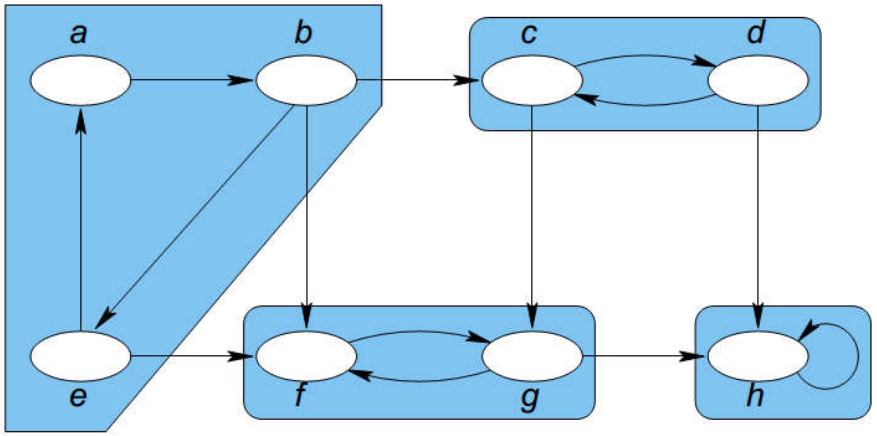
1 chamar DFS(G) para calcular o tempo de término $f[u]$ para cada vértice u

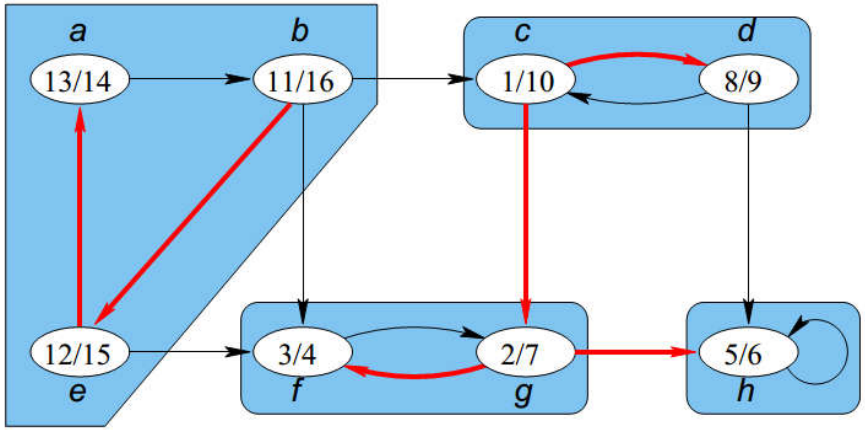
2 calcular G^T

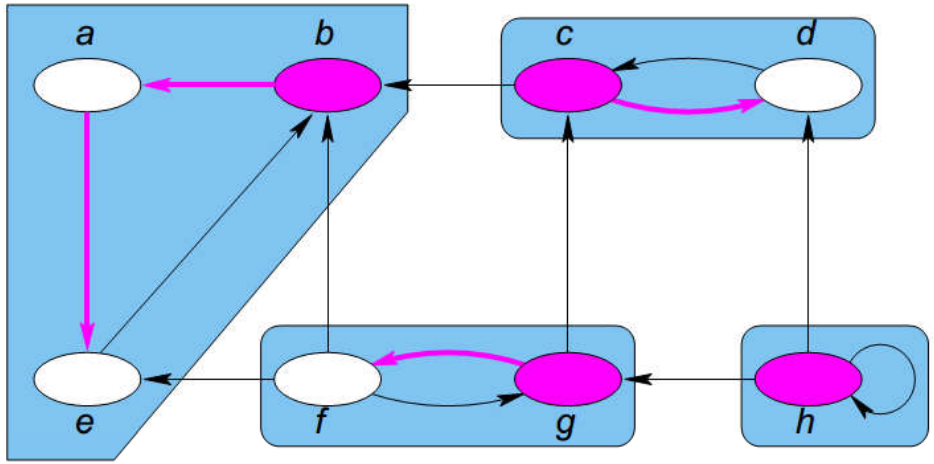
3 chamar DFS(G^T) mas, no loop principal de DFS, considerar os vértices em ordem decrescente de $f[u]$ (calculada na linha 1)

4 dar saída aos vértices de cada árvore na floresta primeiro na profundidade formada na linha 3 como um componente fortemente conectado separado

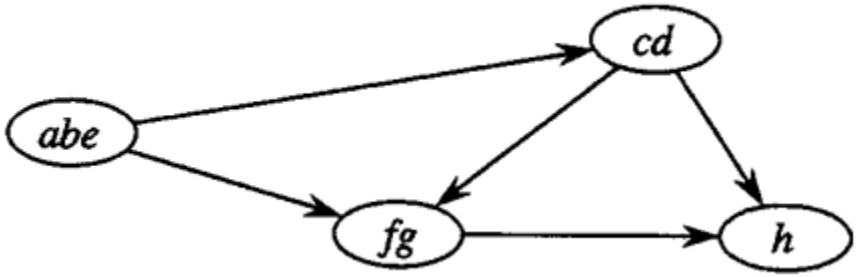
Exemplo







- Os Componentes fortemente conectados são cada arvore da floresta



- Grafo de componentes acíclicos
- Um vértice para cada componente.

UEA
UNIVERSIDADE DO ESTADO DO AMAZONAS

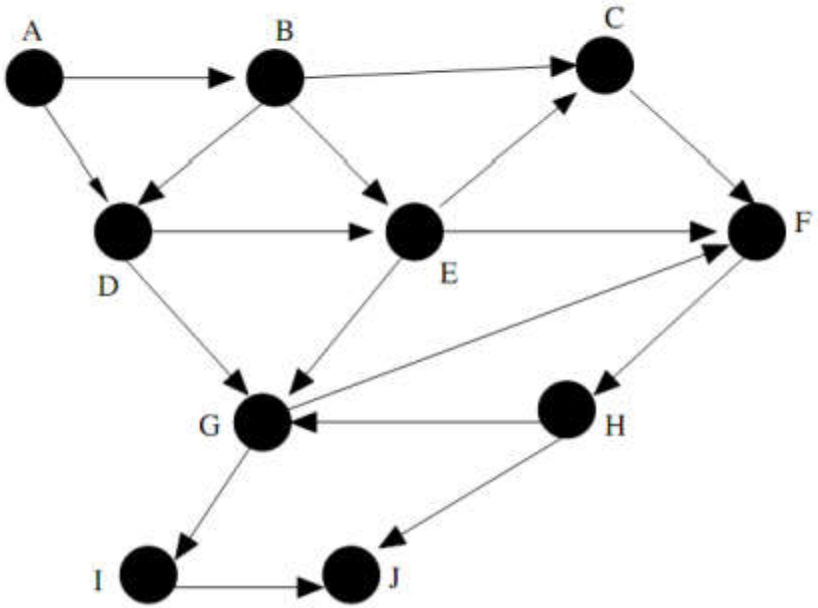
Complexidade

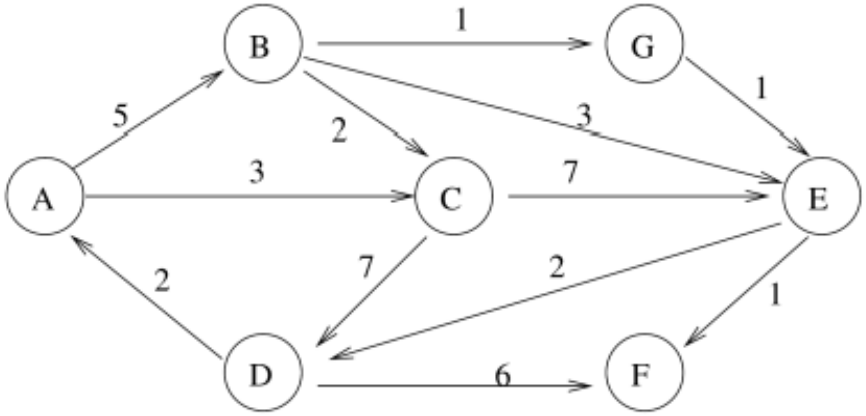
- Complexidade da busca em profundidade DFS(G) para calcular f[u] é **O(V+E)**
- Tempo de criar G^T é **O(V+E)**
- Complexidade da busca em profundidade DFS(G^T) para calcular f[u] é **O(V+E)**
  - Determinar ordem decrescente
- Dar saída aos vértices **O(1)**

O(V+E)

Exercícios

- Forneça a ordem topológica dos grafos





Exercícios

- Forneça os Componentes fortemente conectados dos grafos