TÉCNICO LISBOA

Análise e Síntese de Algoritmos

Componentes Fortemente Ligados (SCCs)

CLRS Cap. 22

Instituto Superior Técnico

2022/2023

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Caminhos mais curtos
  - Fluxos máximos
  - Árvores abrangentes
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica
  - Algoritmos greedy
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Tópicos Adicionais [CLRS, Cap.32-35]
  - Emparelhamento de Cadeias de Caracteres
  - Complexidade Computacional
  - Algoritmos de Aproximação

Resumo

Componentes Fortemente Ligados (SCCs)
Algoritmo DFS(G)+DFS(G^T)
Algoritmo Tarjan

Componentes Fortemente Ligados

Componente Fortemente Ligado
Dado um grafo dirigido $G = (V, E)$ um Componente Fortemente Ligado (ou *Strongly Connected Component* – SCC) é um conjunto máximo de vértices $U \subseteq V$, tal que para quaisquer $u, v \in U$, u é atingível a partir de v, e v é atingível a partir de u

Nota: um vértice simples pode definir um SCC

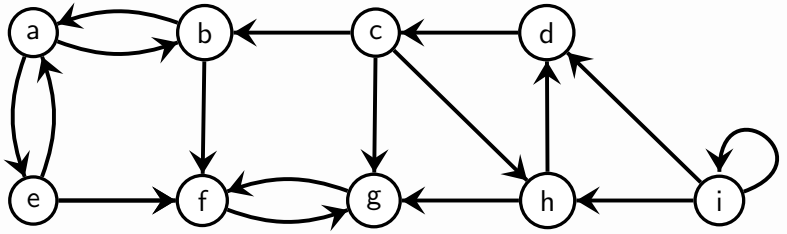
Componente Fortemente Ligado
Dado um grafo dirigido $G = (V, E)$ um Componente Fortemente Ligado (ou *Strongly Connected Component* – SCC) é um conjunto máximo de vértices $U \subseteq V$, tal que para quaisquer $u, v \in U$, u é atingível a partir de v, e v é atingível a partir de u

Nota: um vértice simples pode definir um SCC

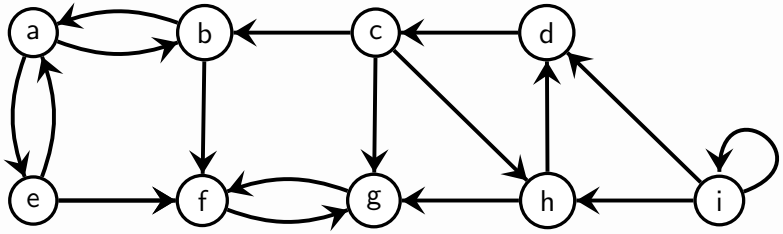
Questão

- Um DAG pode conter SCCs?

Exercício

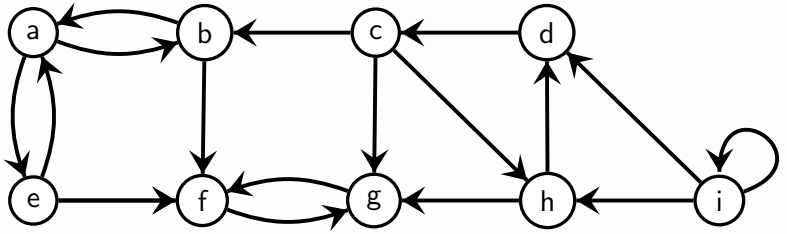


Exercício



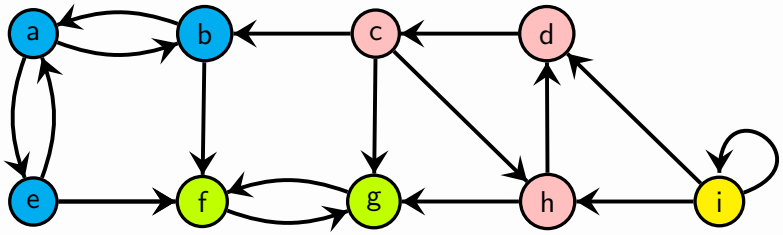
- Tem ciclos ou é um DAG?

Exercício



- Tem ciclos ou é um DAG?
- Tem quantos SCCs ?

Exercício



- Tem ciclos ou é um DAG?
- Tem quantos SCCs ? 4

Soluções algorítmicas

- Algoritmo baseado na DFS(G)+DFS(G^T) (Kosaraju-Sharir)
- Algoritmo de Tarjan

Grafo Transposto

Dado um grafo dirigido $G = (V, E)$, o grafo transposto de G é definido da seguinte forma:

$$G^T = (V, E^T) \text{ tal que : } E^T = \{(u, v) : (v, u) \in E\}$$

Grafo Transposto

Dado um grafo dirigido $G = (V, E)$, o grafo transposto de G é definido da seguinte forma:

$$G^T = (V, E^T) \text{ tal que : } E^T = \{(u, v) : (v, u) \in E\}$$

Complexidade

Componentes Fortemente Ligados

Grafo Transposto

Dado um grafo dirigido $G = (V, E)$, o grafo transposto de G é definido da seguinte forma:

$$G^T = (V, E^T) \text{ tal que : } E^T = \{(u, v) : (v, u) \in E\}$$

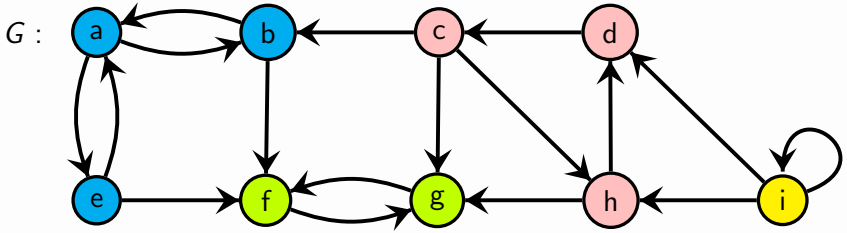
Complexidade

- $O(V + E)$

Algoritmo DFS(G) + DFS(G^T)

Observação

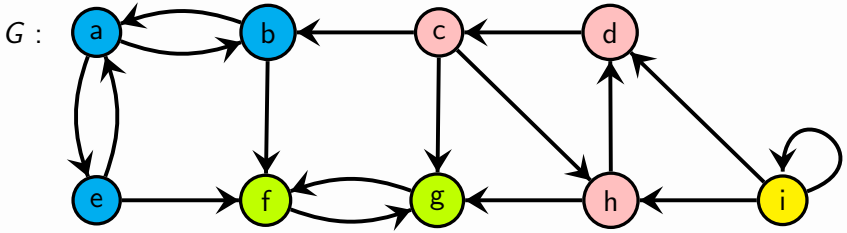
G :



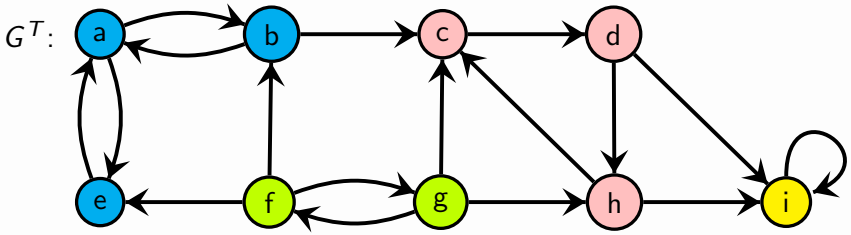
Algoritmo DFS(G) + DFS(G^T)

Observação

G :



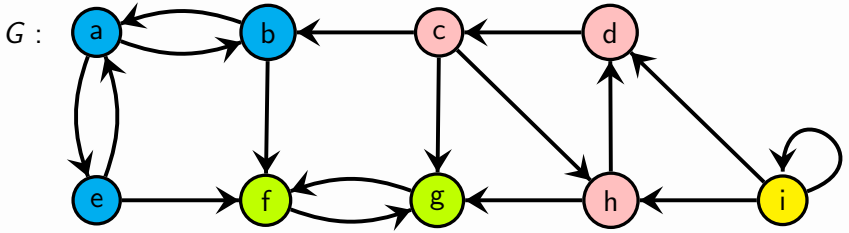
G^T:



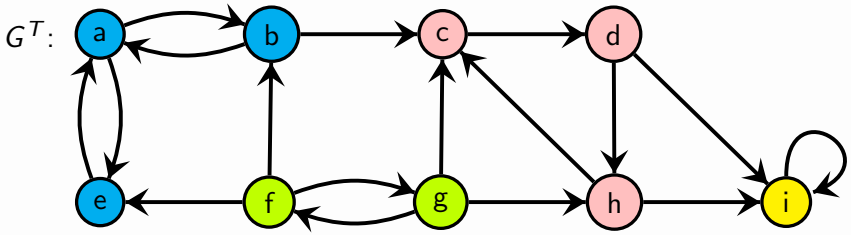
Algoritmo DFS(G) + DFS(G^T)

Observação

G :



G^T:



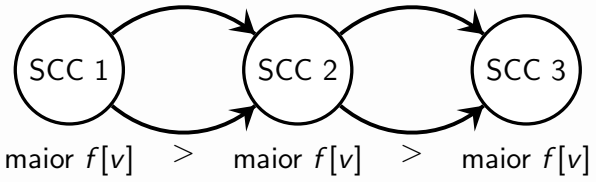
- Os SCCs de G^T são os mesmos de G

Algoritmo DFS(G)+DFS(G^T)

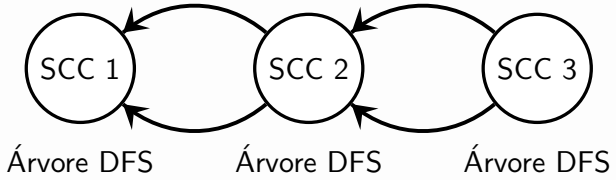
Relembrar: Num DAG, se existe caminho de u para v, então $f[u] > f[v]$!

Intuição

Em G:



Em G^T:



Algoritmo DFS(G)+DFS(G^T)

SCC-DFS(G)

1. Executar DFS(G) para cálculo de $f[v]$, para cada $v \in G.V$
2. Representar G^T
3. Executar DFS(G^T), considerar vértices por ordem decrescente de $f[v]$

return floresta DF

- Cada árvore da floresta DF corresponde a um SCC

Algoritmo DFS(G)+DFS(G^T)

SCC-DFS(G)

1. Executar DFS(G) para cálculo de $f[v]$, para cada $v \in G.V$
2. Representar G^T
3. Executar DFS(G^T), considerar vértices por ordem decrescente de $f[v]$

return floresta DF

- Cada árvore da floresta DF corresponde a um SCC

Complexidade

- $\Theta(V + E)$

Algoritmo DFS(G)+DFS(G^T)

Propriedades

- Um grafo de componentes $G^{SCC} = (V^{SCC}, E^{SCC})$ é um DAG

Algoritmo DFS(G)+DFS(G^T)

Propriedades

- Um grafo de componentes $G^{SCC} = (V^{SCC}, E^{SCC})$ é um DAG
- Se C e C' forem SCCs distintos do grafo G, e existir um caminho de $u \in C$ para $v \in C'$:
  - Não pode haver um caminho de v para u
  - $max_{x \in C}\{f[x]\} > max_{y \in C'}\{f[y]\}$
- Segunda DFS visita G^{SCC} seguindo uma ordem topológica

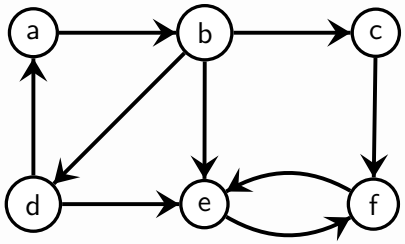
Algoritmo Tarjan

Intuição

- Baseado no algoritmo DFS
- Raíz de um SCC: primeiro vértice do SCC a ser descoberto
- Utilização de arcos para trás e de cruzamento na mesma árvore DF para identificação de ciclos
- $d[v]$: Número de vértices visitados quando v é descoberto
- $low[v]$: O menor valor de $d[]$ atingível por um arco para trás ou de cruzamento na sub-árvore de v
- Se $d[v] = low[v]$, então v é raíz de um SCC

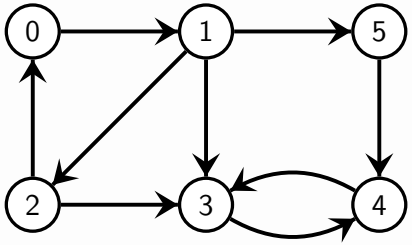
Algoritmo Tarjan

Exemplo valor $low[v]$

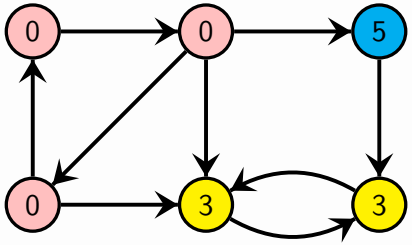


Algoritmo Tarjan

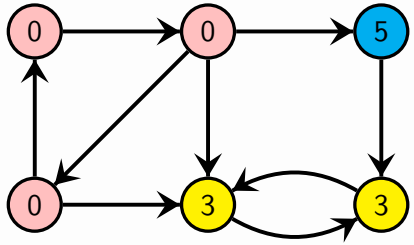
Exemplo valor $low[v]$



Exemplo valor $low[v]$



Exemplo valor $low[v]$



Invariante de Pilha: conjunto de vértices dos quais é permitido actualizar o valor $low[v]$

SCC-Tarjan(G)

```
visited ← 0
L ← 0
for u ∈ G.V do
    d[u] ← ∞
end for
for u ∈ G.V do
    if d[u] == ∞ then
        Tarjan-Visit(G, u)
    end if
end for
```

Tarjan-Visit(G, u)

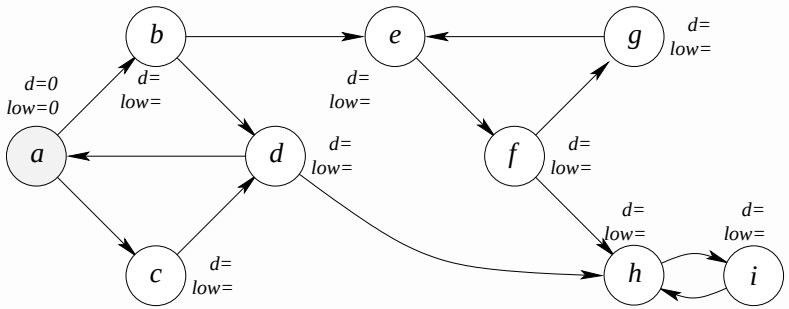
```
d[u] ← low[u] ← visited
visited ← visited + 1
Push(L, u)
for v ∈ G.Adj[u] do
    if d[v] == ∞ or v ∈ L then
        if d[v] == ∞ then
            Tarjan-Visit(G, v)
        end if
        low[u] ← min(low[u], low[v])
    end if
end for
if d[u] == low[u] then
    repeat
        v ← Pop(L)
    until u == v
end if
```


Complexidade

Tempo de execução: $\Theta(V + E)$

- Inicialização: $\Theta(V)$
- Chamadas a Tarjan-Visit: $O(V)$
- Lista de adjacência de cada vértice analisada apenas 1 vez: $\Theta(E)$

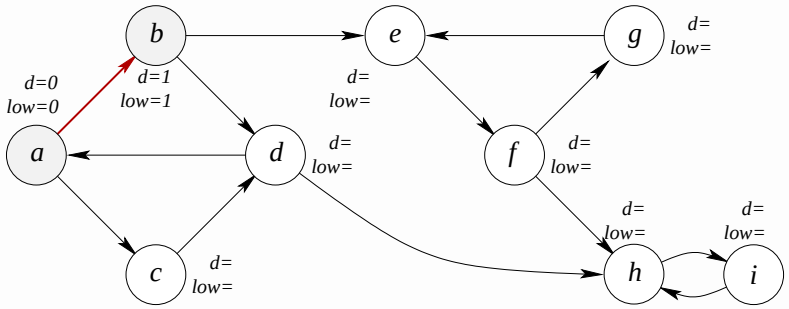
Algoritmo Tarjan



L: a

SCCs:

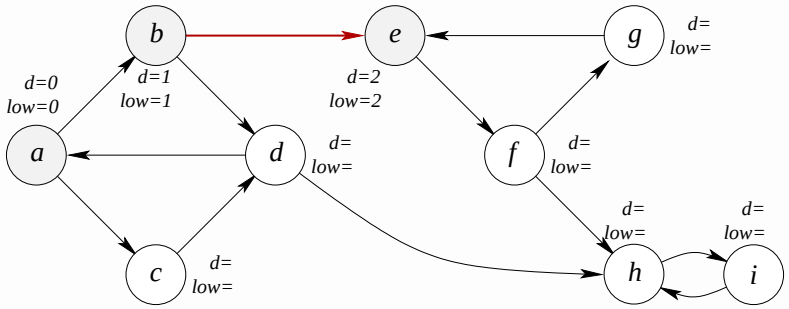
Algoritmo Tarjan



L: a, b

SCCs:

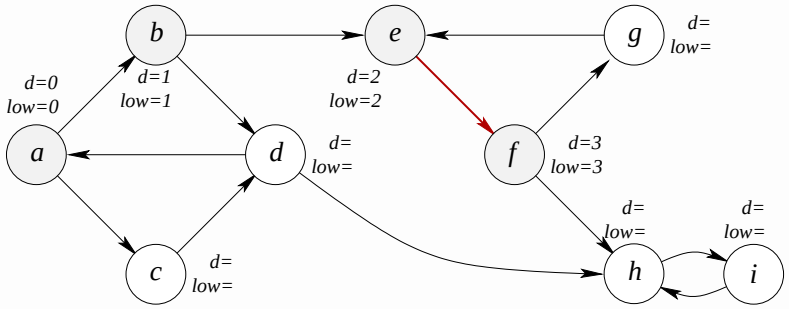
Algoritmo Tarjan



L: a, b, e

SCCs:

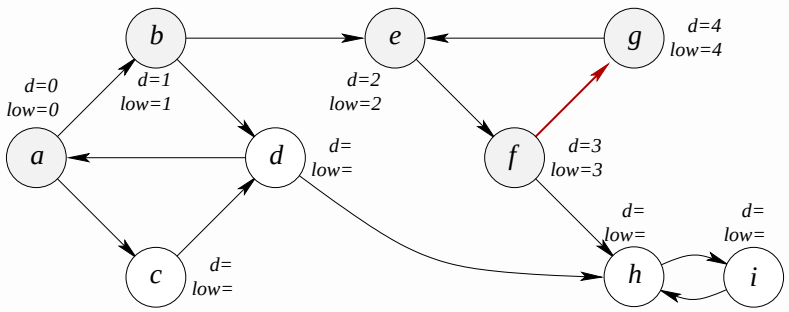
Algoritmo Tarjan



L: a, b, e, f

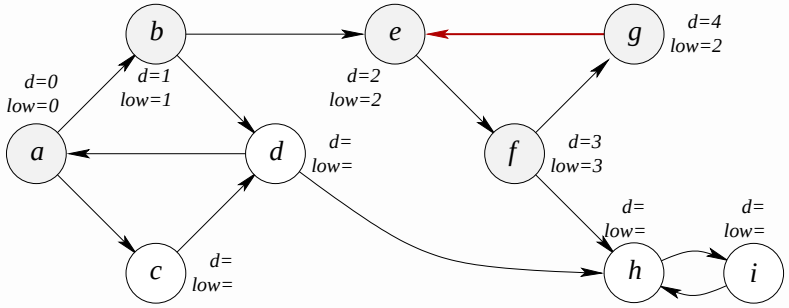
SCCs:

Algoritmo Tarjan



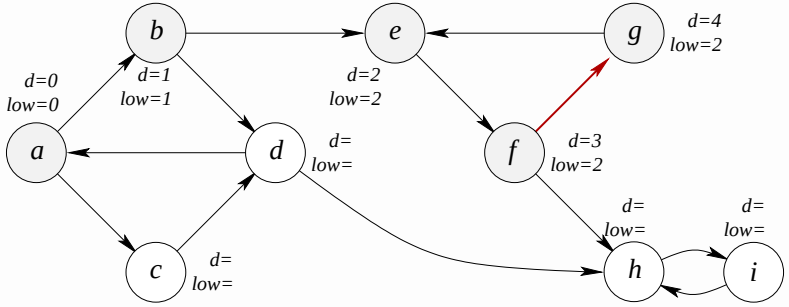
L: a, b, e, f, g *SCCs:*

Algoritmo Tarjan



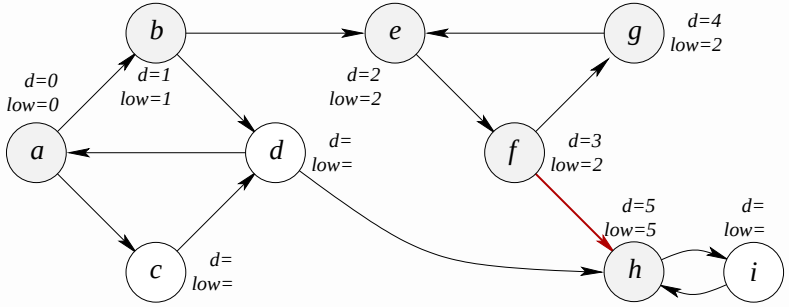
L: a, b, e, f, g *SCCs:*

Algoritmo Tarjan



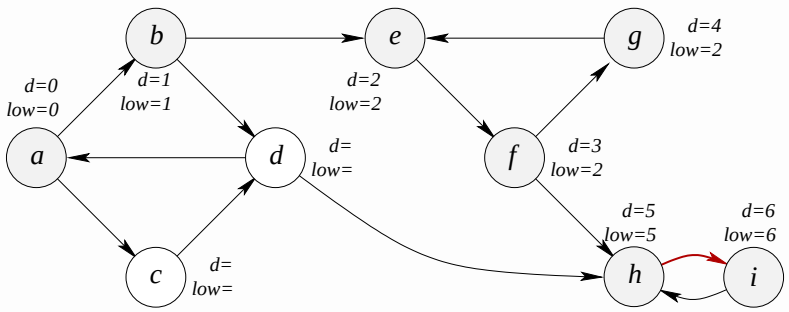
L: a, b, e, f, g *SCCs:*

Algoritmo Tarjan



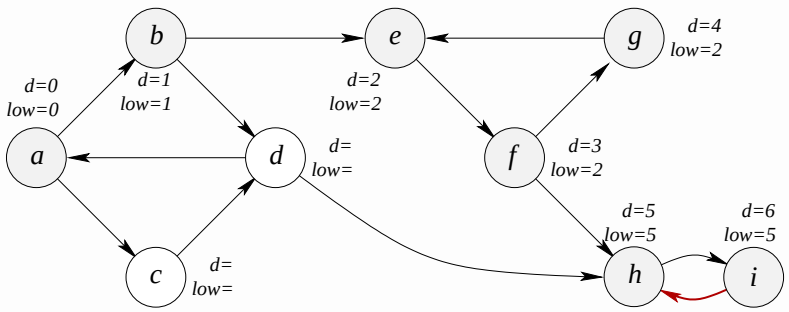
L: a, b, e, f, g, h *SCCs:*

Algoritmo Tarjan



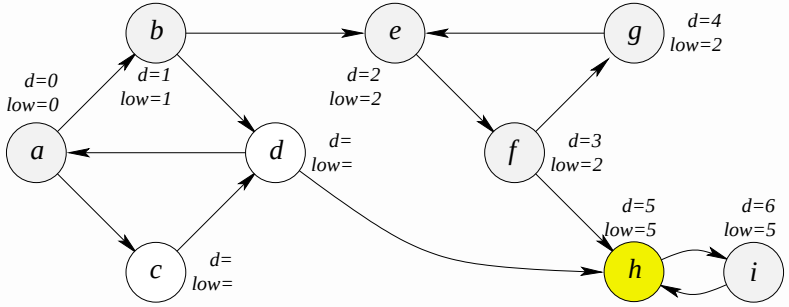
L: a, b, e, f, g, h, i *SCCs:*

Algoritmo Tarjan



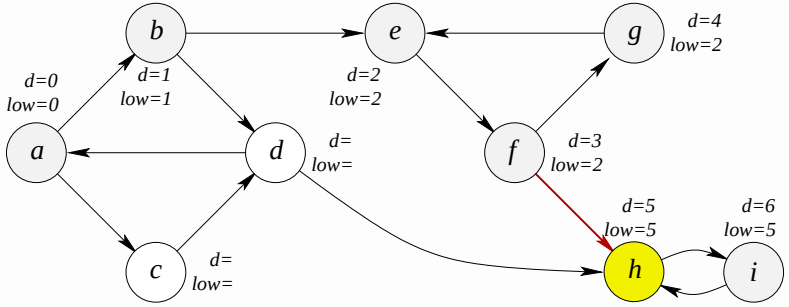
L: a, b, e, f, g, h, i *SCCs:*

Algoritmo Tarjan



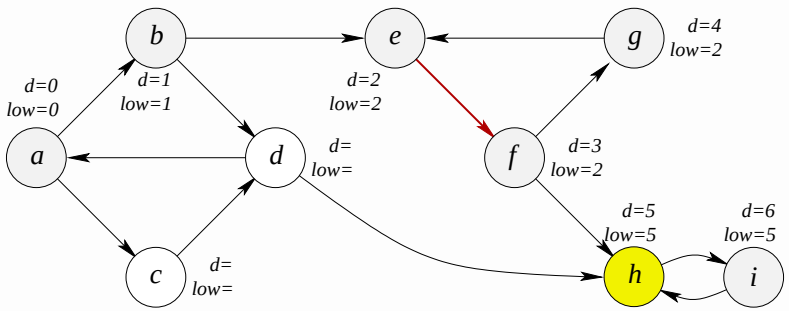
L: a, b, e, f, g *SCCs: {h, i}*

Algoritmo Tarjan



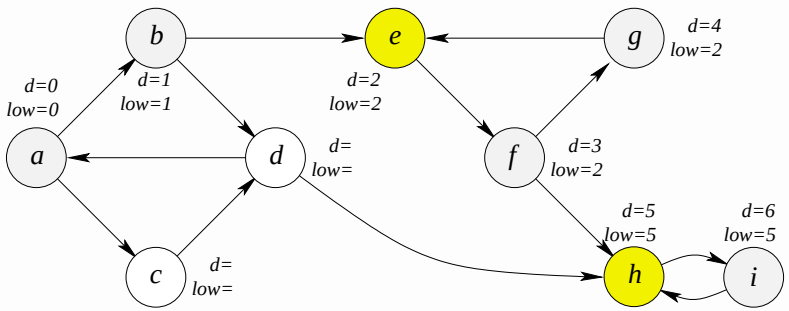
L: a, b, e, f, g *SCCs: {h, i}*

Algoritmo Tarjan



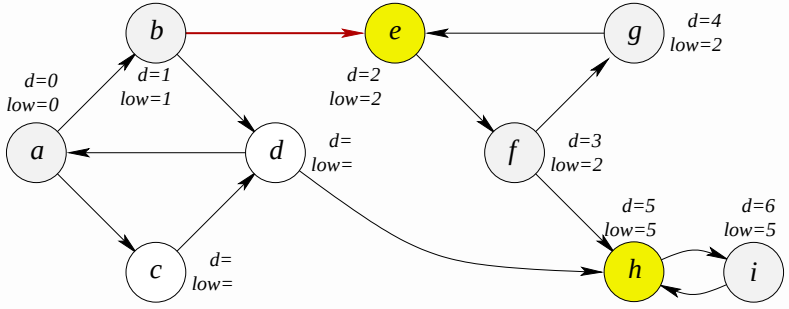
L: a, b, e, f, g *SCCs: {h, i}*

Algoritmo Tarjan



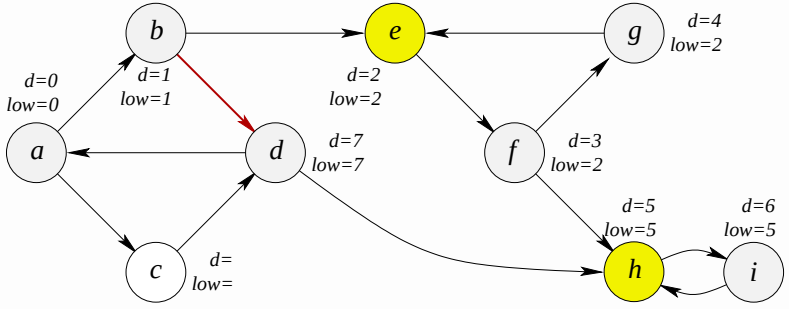
L: a, b *SCCs: {h, i} {e, f, g}*

Algoritmo Tarjan



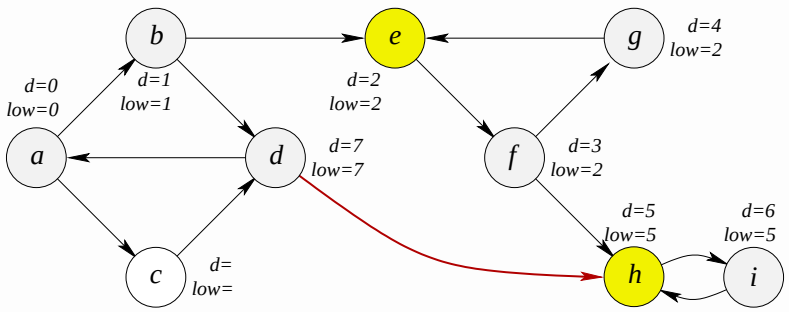
L: a, b *SCCs: {h, i} {e, f, g}*

Algoritmo Tarjan



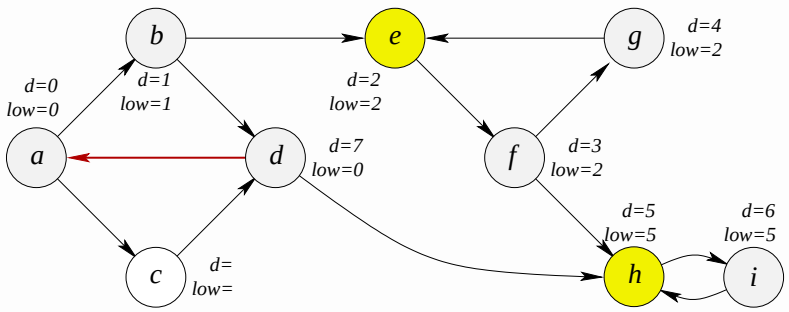
L: a, b, d *SCCs: {h, i} {e, f, g}*

Algoritmo Tarjan



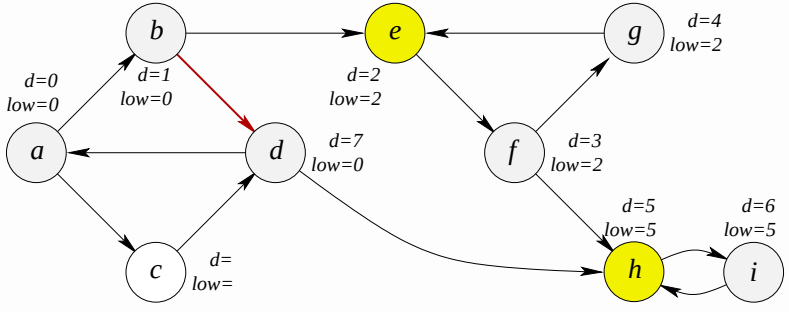
L: a, b, d *SCCs: {h, i} {e, f, g}*

Algoritmo Tarjan



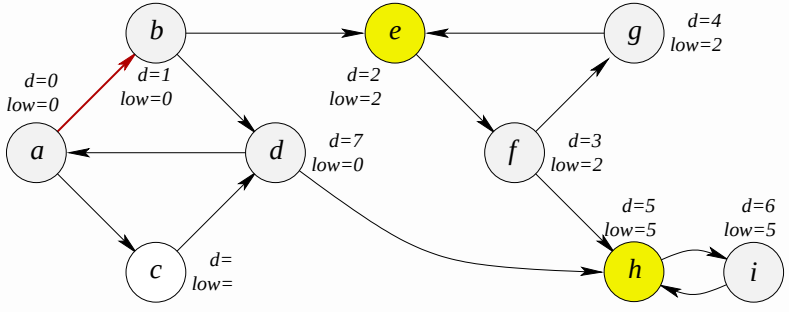
L: a, b, d *SCCs: {h, i} {e, f, g}*

Algoritmo Tarjan



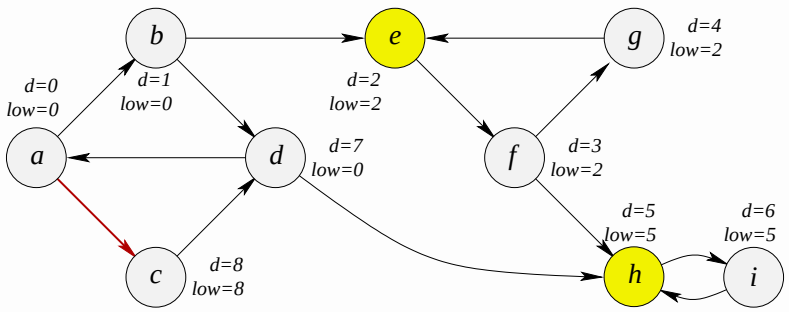
L: a, b, d *SCCs: {h, i} {e, f, g}*

Algoritmo Tarjan



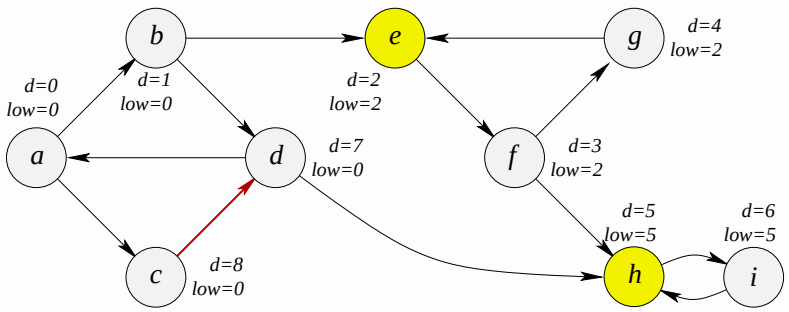
L: a, b, d *SCCs: {h, i} {e, f, g}*

Algoritmo Tarjan



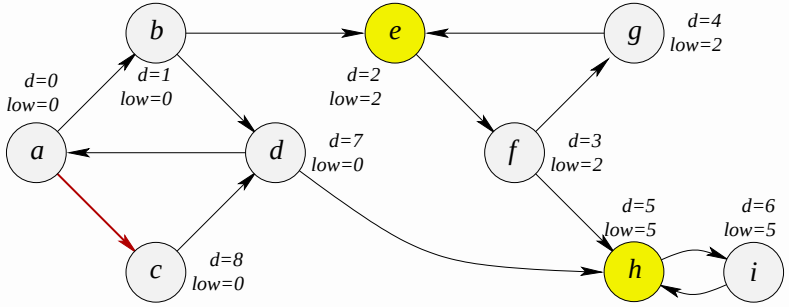
L: a, b, d, c *SCCs: {h, i} {e, f, g}*

Algoritmo Tarjan



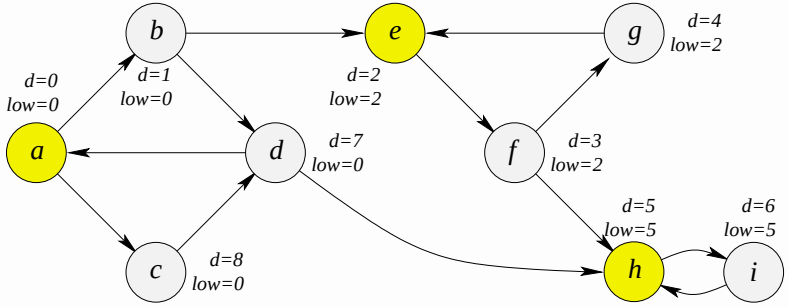
L: a, b, d, c *SCCs: {h, i} {e, f, g}*

Algoritmo Tarjan



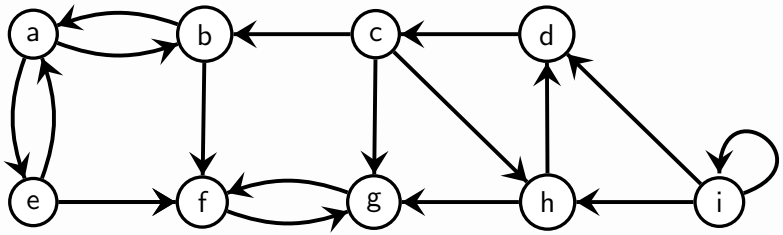
L: a, b, d, c *SCCs: {h, i} {e, f, g}*

Algoritmo Tarjan



L: *SCCs: {h, i} {e, f, g} {a, b, c, d}*

Exercício (fazer em casa)



Resultado secundário

O Algoritmo de Tarjan adicionalmente indica uma ordem topológica entre os SCCs descobertos

- Por ordem crescente do valor *low* das raízes dos SCCs
- Por ordem inversa da apresentação dos SCCs