Relações binárias

Relações binárias

Uma relação R sobre um conjunto V é um subconjunto de $V \times V$.

Relações binárias

Uma relação R sobre um conjunto V é um subconjunto de $V \times V$.

Podemos enxergar R como sendo o conjunto de arcos de um digrafo com vértices V, $G(R)$.

Relações binárias

Uma relação R sobre um conjunto V é um subconjunto de $V \times V$.

Podemos enxergar R como sendo o conjunto de arcos de um digrafo com vértices V, $G(R)$.

É só outra linguagem.

Relações binárias

Uma relação R sobre um conjunto V é um subconjunto de $V \times V$.

Podemos enxergar R como sendo o conjunto de arcos de um digrafo com vértices V, $G(R)$.

É só outra linguagem.

O fecho reflexivo-transitivo de R, denotado R^* é a *menor* (no sentido de inclusão) relação contendo R, que é reflexiva e transitiva.

Relações binárias

Uma relação R sobre um conjunto V é um subconjunto de $V \times V$.

Podemos enxergar R como sendo o conjunto de arcos de um digrafo com vértices V, $G(R)$.

É só outra linguagem.

O fecho reflexivo-transitivo de R, denotado R^* é a *menor* (no sentido de inclusão) relação contendo R, que é reflexiva e transitiva.

É o mesmo que *acessibilidade* em G!

Equivalências

Equivalências

Toda relação reflexiva e transitiva embute uma relação de equivalência derivada, que consiste dos pares simétricos:

$$a \equiv b \leftrightarrow aRb \text{ e } bRa$$

Equivalências

Toda relação reflexiva e transitiva embute uma relação de equivalência derivada, que consiste dos pares simétricos:

$$a \equiv b \leftrightarrow aRb \text{ e } bRa$$

Tipicamente, essa situação envolve duas tarefas:

1. Identificar as classes de equivalência. Decidir se dois elementos são equivalentes.

Equivalências

Toda relação reflexiva e transitiva embute uma relação de equivalência derivada, que consiste dos pares simétricos:

$$a \equiv b \leftrightarrow aRb \text{ e } bRa$$

Tipicamente, essa situação envolve duas tarefas:

1. Identificar as classes de equivalência. Decidir se dois elementos são equivalentes.
2. Entender o quociente.

Equivalências

Toda relação reflexiva e transitiva embute uma relação de equivalência derivada, que consiste dos pares simétricos:

$$a \equiv b \leftrightarrow aRb \text{ e } bRa$$

Tipicamente, essa situação envolve duas tarefas:

1. Identificar as classes de equivalência. Decidir se dois elementos são equivalentes.
2. Entender o quociente.

Equivalências

Toda relação reflexiva e transitiva embute uma relação de equivalência derivada, que consiste dos pares simétricos:

$$a \equiv b \leftrightarrow aRb \text{ e } bRa$$

Tipicamente, essa situação envolve duas tarefas:

1. Identificar as classes de equivalência. Decidir se dois elementos são equivalentes.
2. Entender o quociente.
   Neste caso, R induz uma *relação de ordem*.

Em digrafos

Em digrafos

Um digrafo em que u alcança v, para todos vértices u, v, é dito fortemente conexo.

Em digrafos

Um digrafo em que u alcança v, para todos vértices u, v, é dito fortemente conexo.

Isto é, se na equivalência derivada da acessibilidade existe uma única classe.

Em digrafos

Um digrafo em que u alcança v, para todos vértices u, v, é dito fortemente conexo.

Isto é, se na equivalência derivada da acessibilidade existe uma única classe.

Em geral, existem mais classes. Os subgrafos induzidos pelas classes são as componentes fortes do digrafo.

Em digrafos

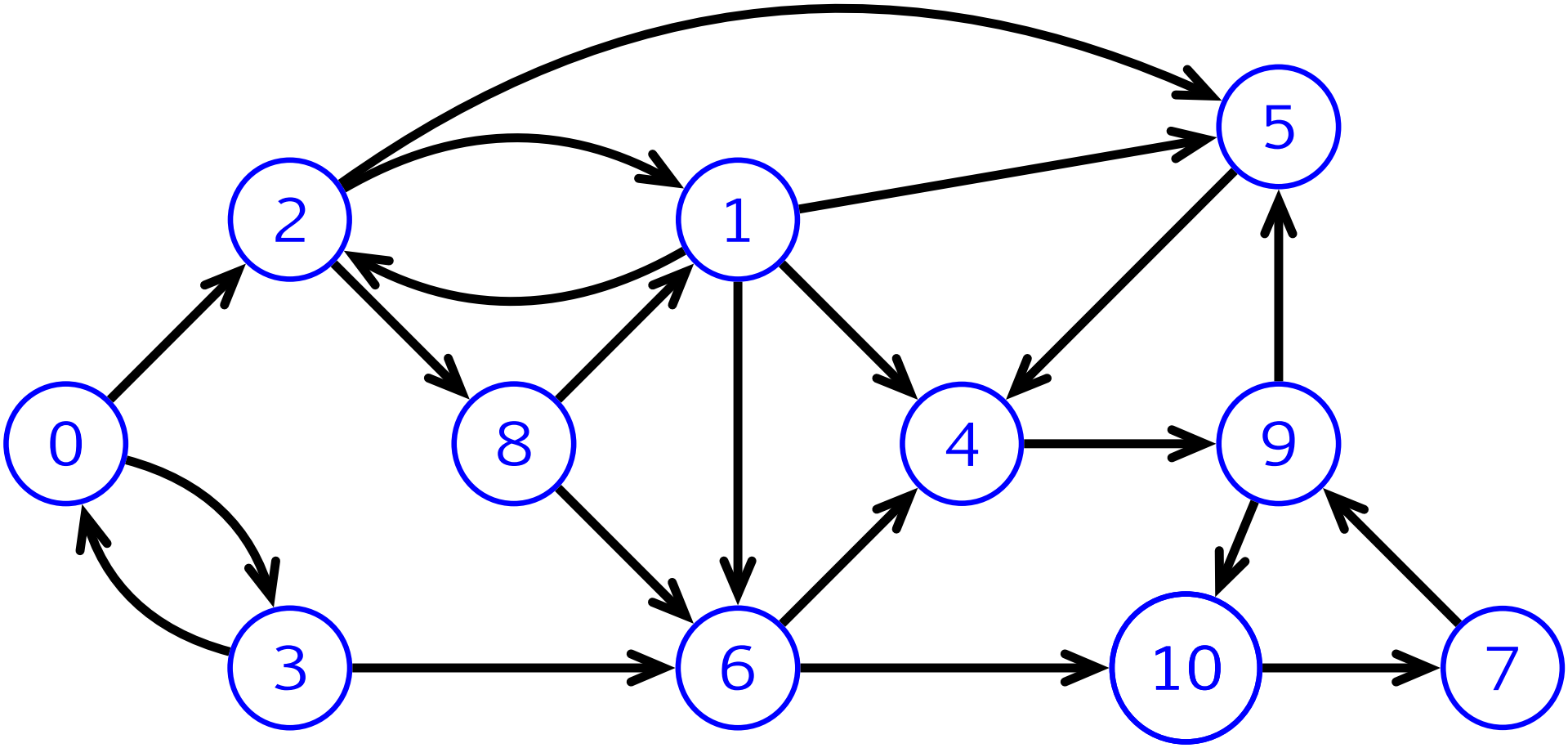
Um digrafo em que u alcança v, para todos vértices u, v, é dito fortemente conexo.

Isto é, se na equivalência derivada da acessibilidade existe uma única classe.

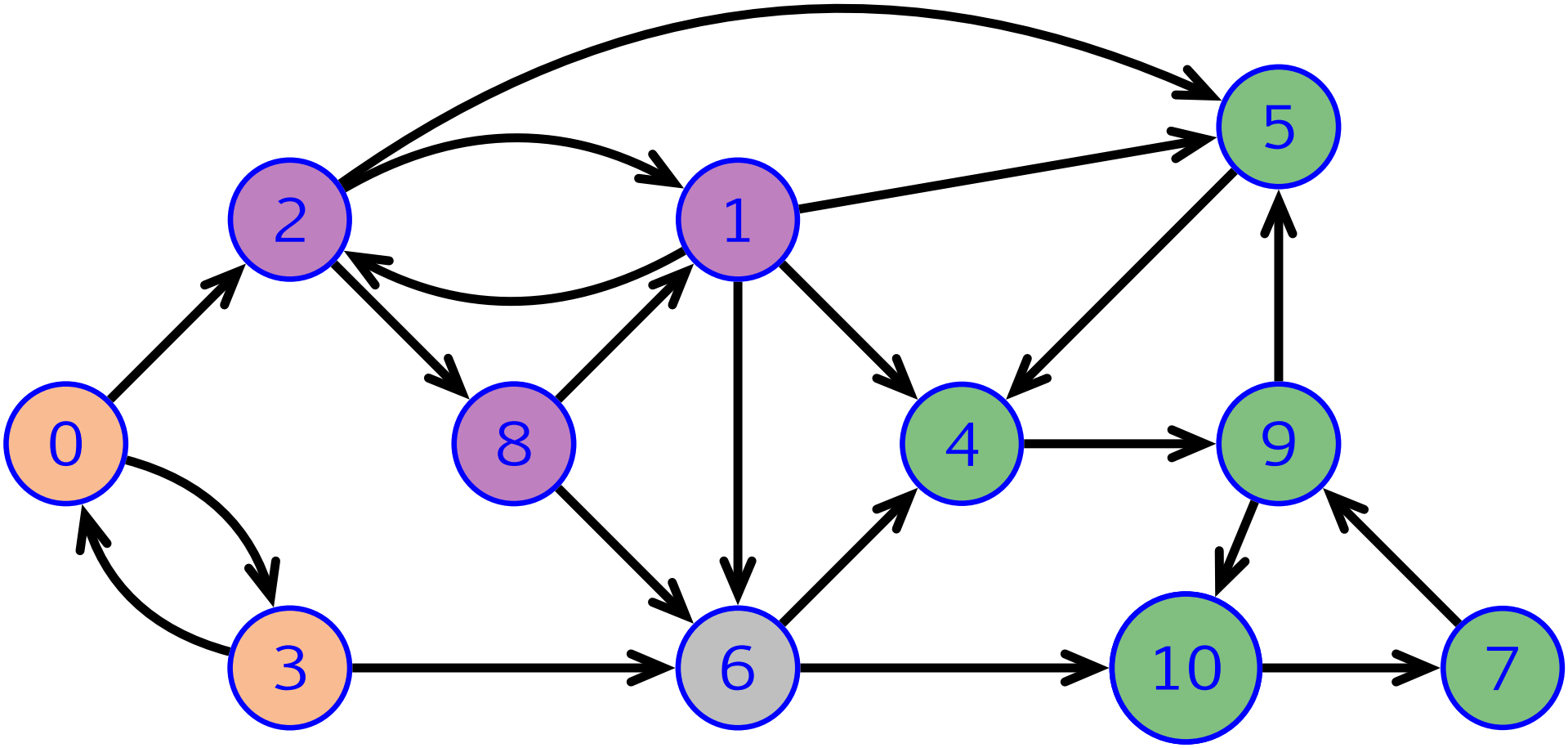
Em geral, existem mais classes. Os subgrafos induzidos pelas classes são as componentes fortes do digrafo.

Elas são os subgrafos fortemente conexos maximais.

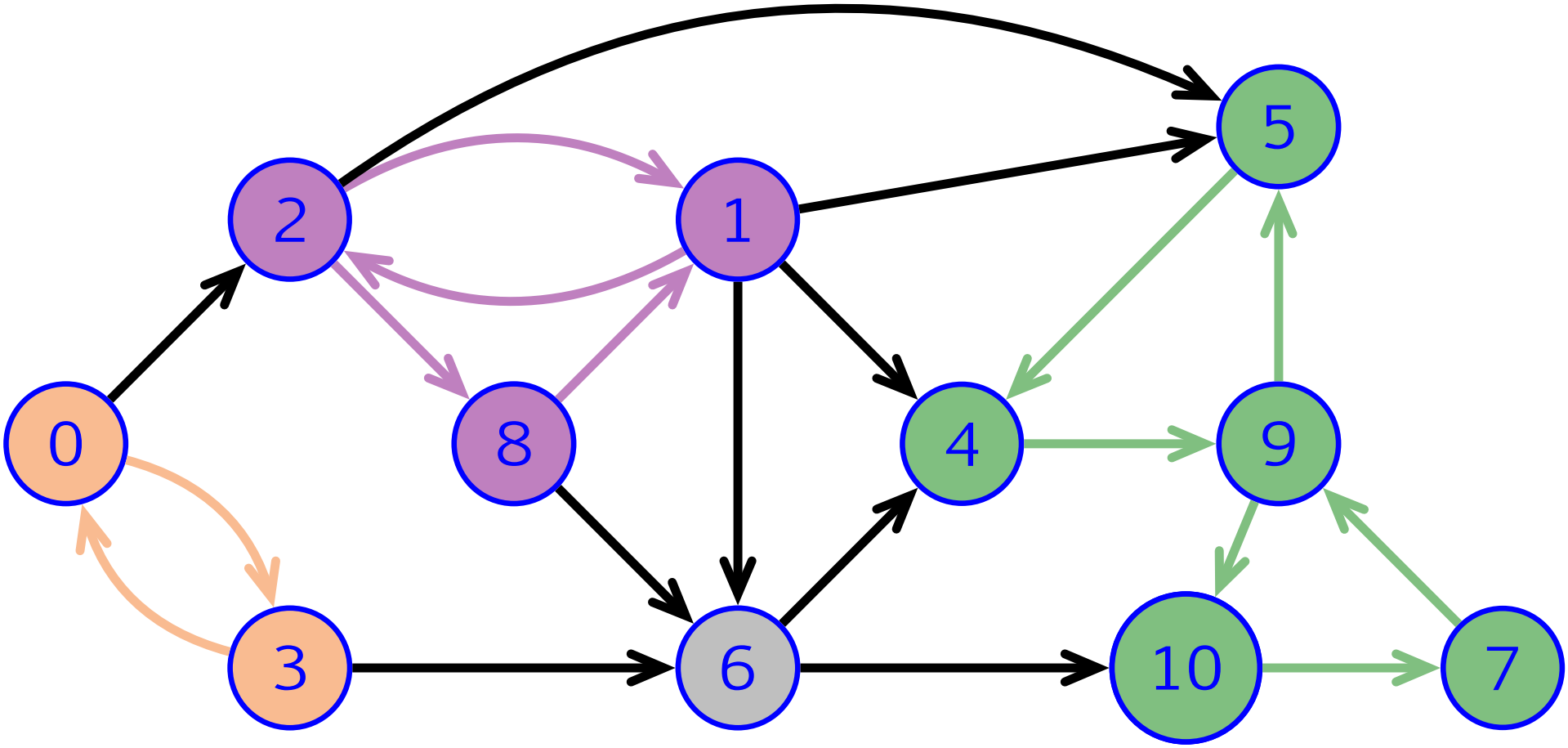
Componentes fortes



Componentes fortes



Componentes fortes



O quociente

O quociente

Sempre que uma relação de equivalência está presente junto com outra estrutura, é natural ver como essa estrutura se reflete no quociente.

O quociente

Sempre que uma relação de equivalência está presente junto com outra estrutura, é natural ver como essa estrutura se reflete no quociente.

Isso é muito comum em Álgebra: inteiros módulo n, espaços vetoriais quocientes, e uma imensa lista de outros exemplos.

O quociente

Sempre que uma relação de equivalência está presente junto com outra estrutura, é natural ver como essa estrutura se reflete no quociente.

Isso é muito comum em Álgebra: inteiros módulo n, espaços vetoriais quocientes, e uma imensa lista de outros exemplos.

No caso de digrafos, é natural estudar o digrafo quociente pela equivalência de acessibilidade.

O quociente

Sempre que uma relação de equivalência está presente junto com outra estrutura, é natural ver como essa estrutura se reflete no quociente.

Isso é muito comum em Álgebra: inteiros módulo n, espaços vetoriais quocientes, e uma imensa lista de outros exemplos.

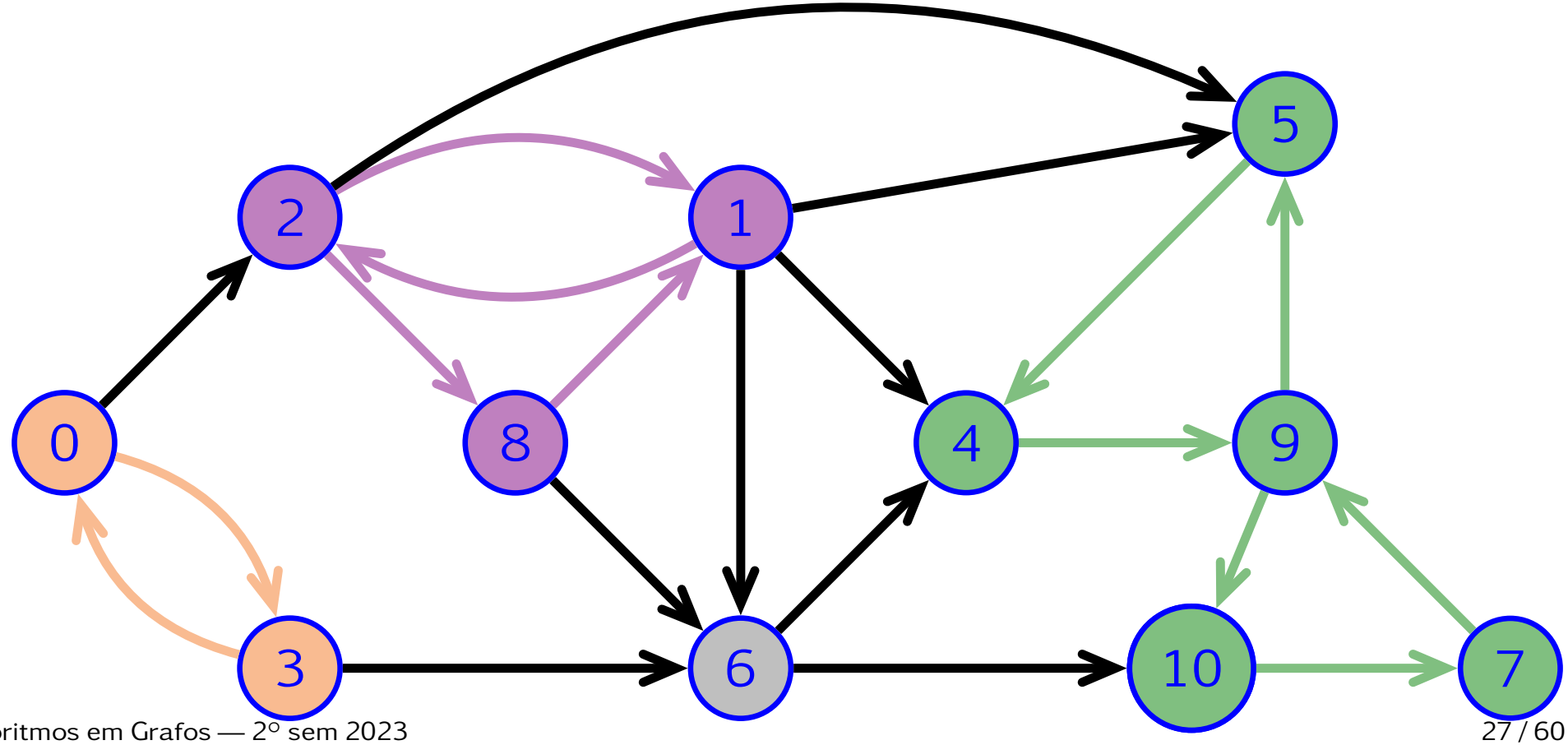
No caso de digrafos, é natural estudar o digrafo quociente pela equivalência de acessibilidade.

Ele é chamado de condensação (ou grafo das componentes) do grafo original.

A condensação

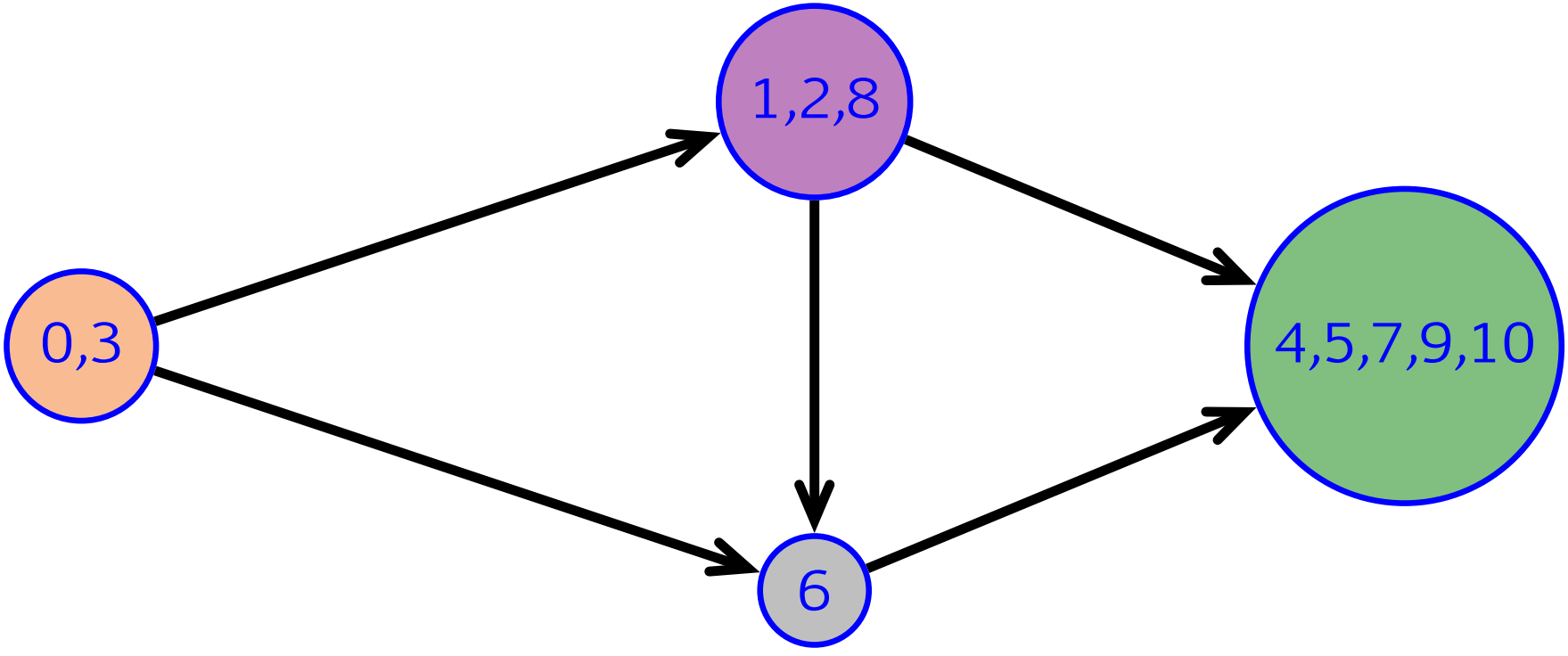
A condensação

A condensação de um digrafo G é o grafo que tem um vértice
para cada componente forte de G, e
um arco de u a v se existe um arco de um vértice de u a um de v.



A condensação

A condensação de um digrafo G é o grafo que tem um vértice
para cada componente forte de G, e
um arco de u a v se existe um arco de um vértice de u a um de v.



Teorema

A condensação de um digrafo é um digrafo acíclico.

Teorema

A condensação de um digrafo é um digrafo acíclico.

Para provar isso, usamos o

Lema

Sejam A e B classes de acessibilidade de G, e $a \in A$, $b \in B$. Se na condensação de G existe arco $A \to B$, então existe em G caminho de a a b.

Teorema

A condensação de um digrafo é um digrafo acíclico.

Para provar isso, usamos o

Lema

Sejam A e B classes de acessibilidade de G, e $a \in A$, $b \in B$. Se na condensação de G existe arco $A \to B$, então existe em G caminho de a a b.

Prova: Existem vértices $u \in A$, $v \in B$ tal que a aresta $u \to v$ está em G. Como a alcança u e v alcança b, basta concatenar: $a \rightsquigarrow u \to v \rightsquigarrow b$.

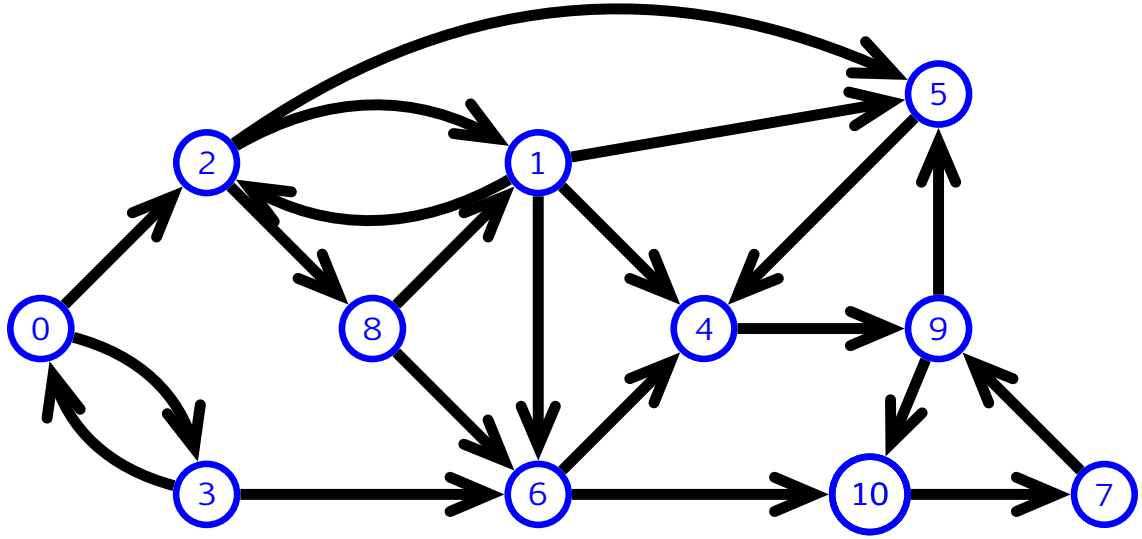
Achar componentes fortes

Dado G, devolve uma lista de listas com as classes de G

Achar componentes fortes

Dado G, devolve uma lista de listas com as classes de G

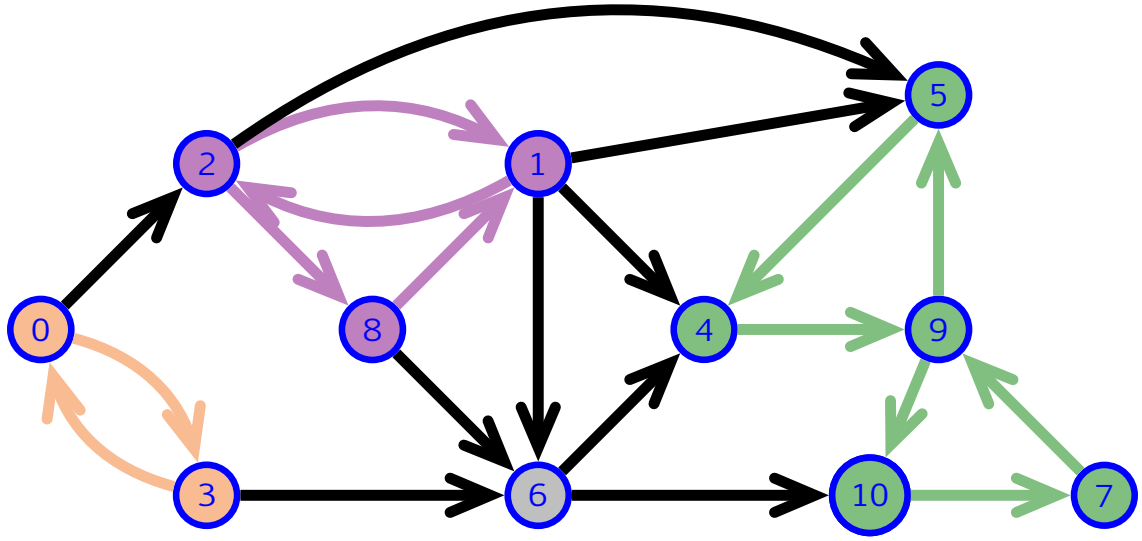
Dado



Achar componentes fortes

Dado G, devolve uma lista de listas com as classes de G

Dado



Devolve [[0,3], [1,2,8], [6], [4,5,7,9,10]].

Algoritmo de Tarjan/Gabow

Algoritmo de Tarjan/Gabow

Calcula recursivamente $v.low = u$, onde $u.d$ é mínimo tal que ou $u = v$ ou existe arco $w \longrightarrow u$ para algum descendente de v (v inclusive).

Algoritmo de Tarjan/Gabow

Calcula recursivamente $v.low = u$, onde $u.d$ é mínimo tal que ou $u = v$ ou existe arco $w \to u$ para algum descendente de v (v inclusive).

No driver: inicializar $v.low = v$.

Algoritmo de Tarjan/Gabow

Calcula recursivamente $v.low = u$, onde $u.d$ é mínimo tal que ou $u = v$ ou existe arco $w \longrightarrow u$ para algum descendente de v (v inclusive).

No driver: inicializar $v.low = v$.

Usa uma pilha auxiliar P de vértices. Um vértice é empilhado quando descoberto.

Algoritmo de Tarjan/Gabow

Calcula recursivamente $v.low = u$, onde $u.d$ é mínimo tal que ou $u = v$ ou existe arco $w \longrightarrow u$ para algum descendente de v (v inclusive).

No driver: inicializar $v.low = v$.

Usa uma pilha auxiliar P de vértices. Um vértice é empilhado quando descoberto.

Desempilhamento a cada vez que uma componente forte é descoberta.

Algoritmo de Tarjan/Gabow

Calcula recursivamente $v.low = u$, onde $u.d$ é mínimo tal que ou $u = v$ ou existe arco $w \longrightarrow u$ para algum descendente de v (v inclusive).

No driver: inicializar $v.low = v$.

Usa uma pilha auxiliar P de vértices. Um vértice é empilhado quando descoberto.

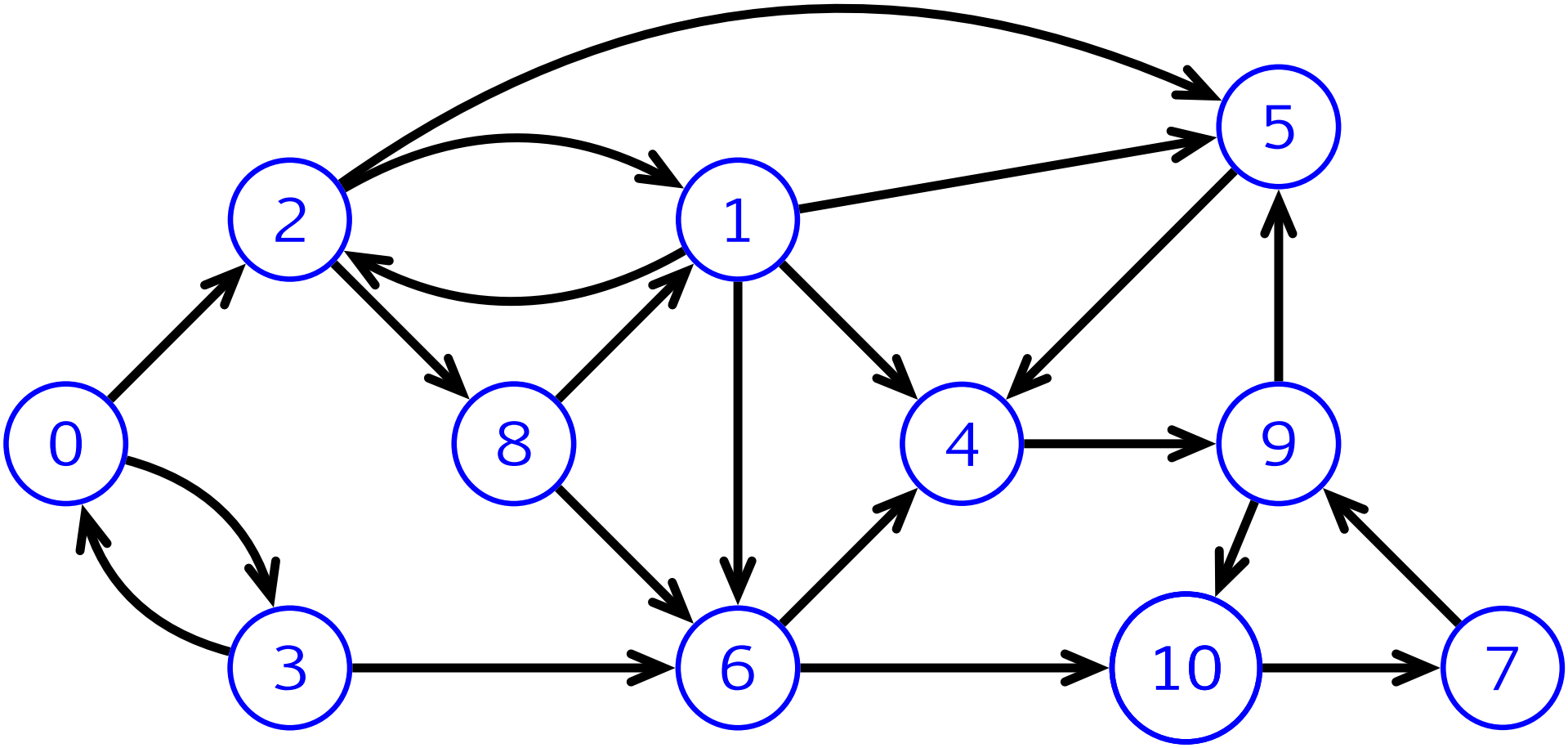
Desempilhamento a cada vez que uma componente forte é descoberta.

A DFS não precisa de $u.f$ nem de $u.sob$

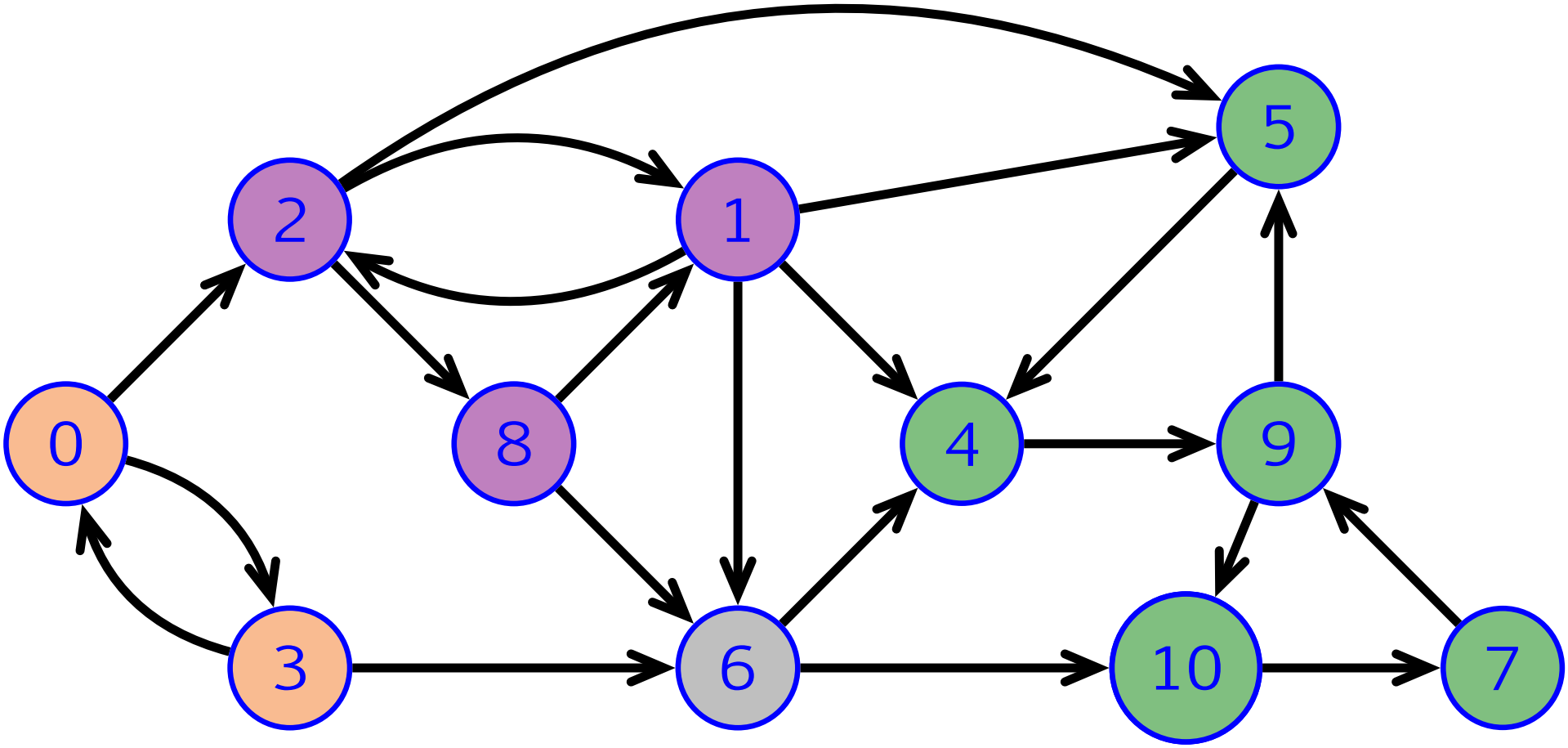
TARJAN-STRONG-COMPONENTS-VISIT(G, u)

```
 1  P.empilha(u)
 2          ...
 3          DFS-VISIT(G, v)
 4          if v.low.d < u.low.d
 5                  u.low = v.low
 6  ...
 7  u.state = finalizado
 8  if u.low == u
 9          Desempilha P até tirar u;
10          o que foi desempilhado é uma componente.
```

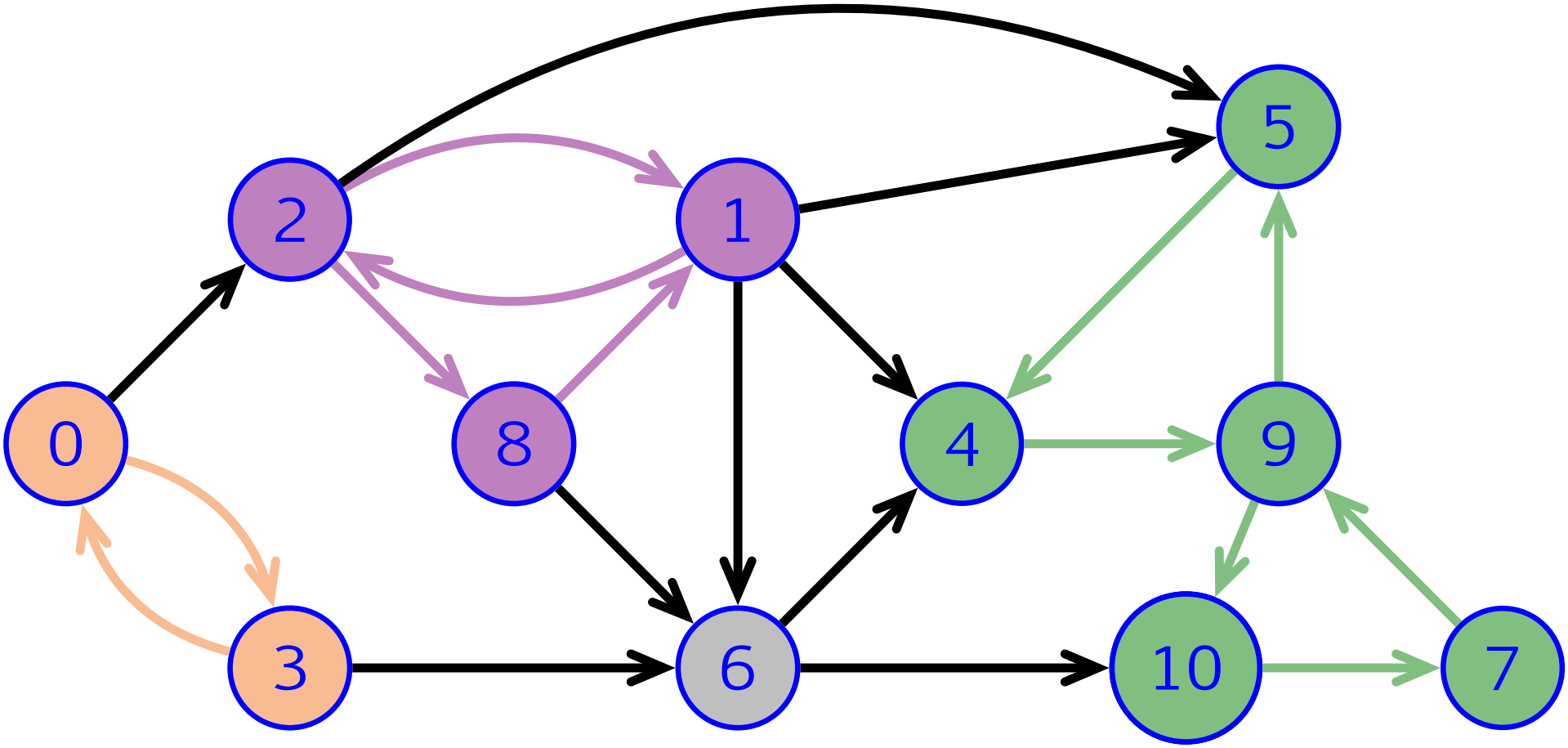
Componentes fortes



Componentes fortes



Componentes fortes



Algoritmo de Kosaraju

1. Efetue uma DFS em G^R, devolvendo uma lista (vetor) F com vértices em ordem de finalização. G^R é G como todas arestas revertidas.

Algoritmo de Kosaraju

1. Efetue uma DFS em G^R, devolvendo uma lista (vetor) F com vértices em ordem de finalização. G^R é G como todas arestas revertidas.
2. Efetue uma DFS em G com driver modificado:

Algoritmo de Kosaraju

1. Efetue uma DFS em G^R, devolvendo uma lista (vetor) F com vértices em ordem de finalização. G^R é G como todas arestas revertidas.
2. Efetue uma DFS em G com driver modificado:
   1. Os vértices seguem F em ordem reversa.

Algoritmo de Kosaraju

1. Efetue uma DFS em G^R, devolvendo uma lista (vetor) F com vértices em ordem de finalização. G^R é G como todas arestas revertidas.
2. Efetue uma DFS em G com driver modificado:
   1. Os vértices seguem F em ordem reversa.
   2. Após cada chamada de DFS-VISIT, os vértices visitados formam uma componente forte.

Algoritmo de Kosaraju

1. Efetue uma DFS em G^R, devolvendo uma lista (vetor) F com vértices em ordem de finalização. G^R é G como todas arestas revertidas.
2. Efetue uma DFS em G com driver modificado:
   1. Os vértices seguem F em ordem reversa.
   2. Após cada chamada de DFS-VISIT, os vértices visitados formam uma componente forte.

Algoritmo de Kosaraju

1. Efetue uma DFS em G^R, devolvendo uma lista (vetor) F com vértices em ordem de finalização. G^R é G como todas arestas revertidas.
2. Efetue uma DFS em G com driver modificado:
   1. Os vértices seguem F em ordem reversa.
   2. Após cada chamada de DFS-VISIT, os vértices visitados formam uma componente forte.

No SAGE, não precisa construir G^R. Basta alterar a chamada a `G.neighbors_out(v)` para `G.neighbors_in(v)`.

Por que? Por que?

Por que? Por que?

1. As classes de acessibilidade em G^R são as mesmas de G.

Por que? Por que?

1. As classes de acessibilidade em G^R são as mesmas de G.
2. A condensação de G^R é o reverso da de G.

Por que? Por que?

1. As classes de acessibilidade em G^R são as mesmas de G.
2. A condensação de G^R é o reverso da de G.
3. A ordenação em F corresponde a uma ordenação topológica da condensação de G. Só que existe mistura das classes.

Por que? Por que?

1. As classes de acessibilidade em G^R são as mesmas de G.
2. A condensação de G^R é o reverso da de G.
3. A ordenação em F corresponde a uma ordenação topológica da condensação de G.
   Só que existe mistura das classes.
4. Percorrendo F ao contrário, as classes vêm em ordenação topológica revertida.
   Quando começamos uma classe, as classes que vêm depois já foram processadas.