

TEORIA DE GRAFOS E COMPUTABILIDADE

ORDENAÇÃO TOPOLÓGICA

Prof. Alexei Machado

Ordenação topológica

2

- Dado um DAG, é possível dispor seus vértices de modo que cada vértice apareça antes de todos os seus sucessores?

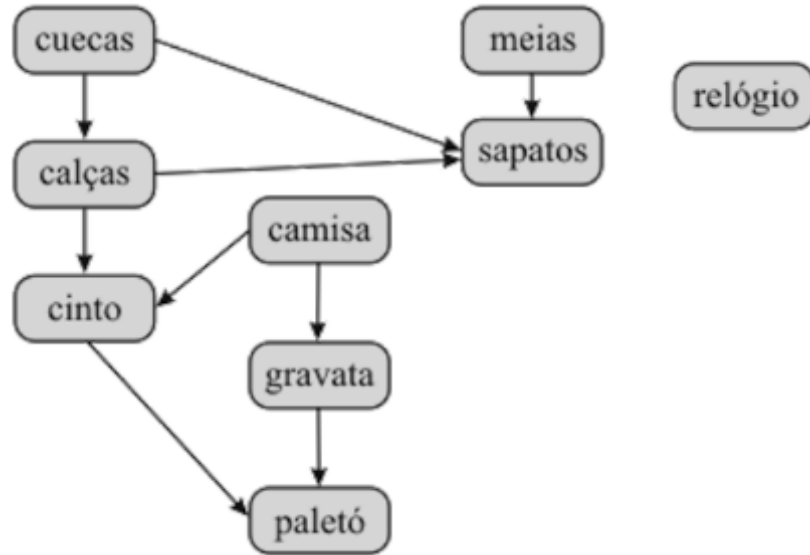
Ordenação topológica

3

- **Ordenação topológica**: ordenação linear de vértices na qual cada vértice precede o conjunto que forma seu **fecho transitivo direto**.

Ordenação topológica

4



Teorema

5

- Seja um digrafo G ,
 - Ou ele possui ciclos.
 - Ou ele apresenta uma ou mais ordenações topológicas.

Ordenação topológica - algoritmos

6

- Existem algoritmos com complexidade linear para determinar uma ordenação topológica de um DAG.
 - Algoritmo de Kahn
 - DFS modificado

Algoritmo de Kahn (1962)

7

- Baseado em duas listas:
 - ▣ S: conjunto de vértices sem arcos de entrada
 - ▣ L: lista de vértices ordenados topologicamente

Possui como princípio determinar a cada instante os vértices que não possuam arcos de entrada e inserir na solução. A cada vértice inserido na solução, todos seus arcos correspondentes são removidos do grafo. Também detecta a existência de ciclos no grafo.

- Retorna uma lista de ordenação topológica OU detecta a existência de um ciclo

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcos de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v, w existente faça

 remover o arco v, w de E

 se w não possuir mais arcos de entrada

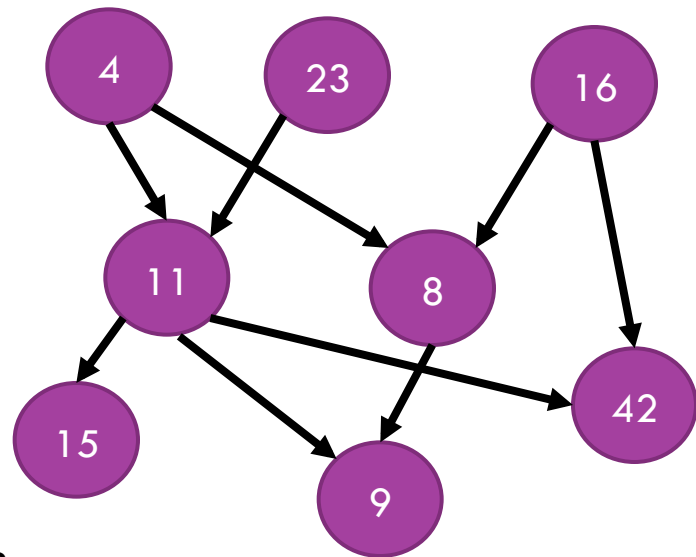
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



Algoritmo de Kahn

$L = \emptyset$;

$S =$ todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v, w existente faça

 remover o arco v, w de E

 se w não possuir mais arcs de entrada

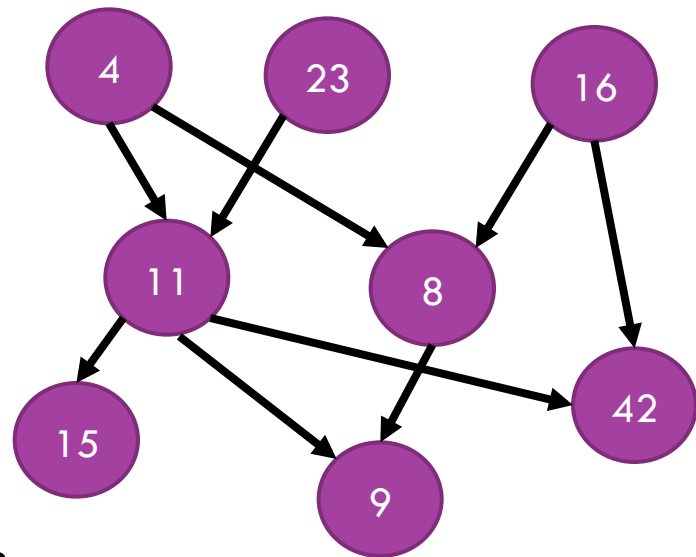
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S



L



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v, w existente faça

 remover o arco v, w de E

 se w não possuir mais arcs de entrada

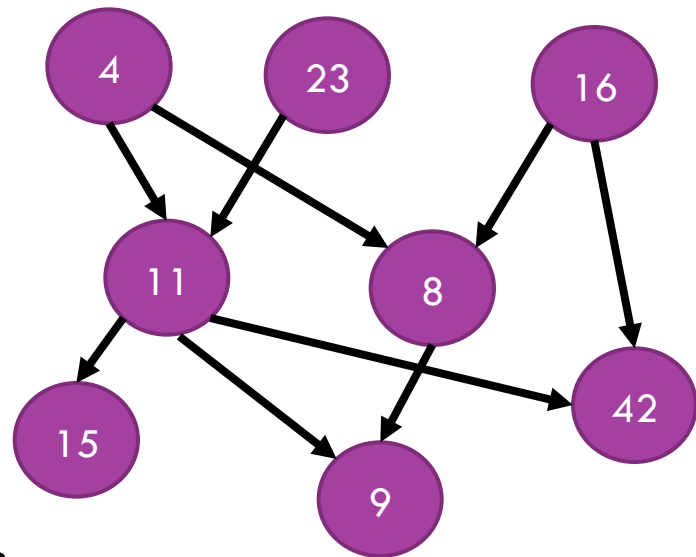
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S



L



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v, w existente faça

 remover o arco v, w de E

 se w não possuir mais arcs de entrada

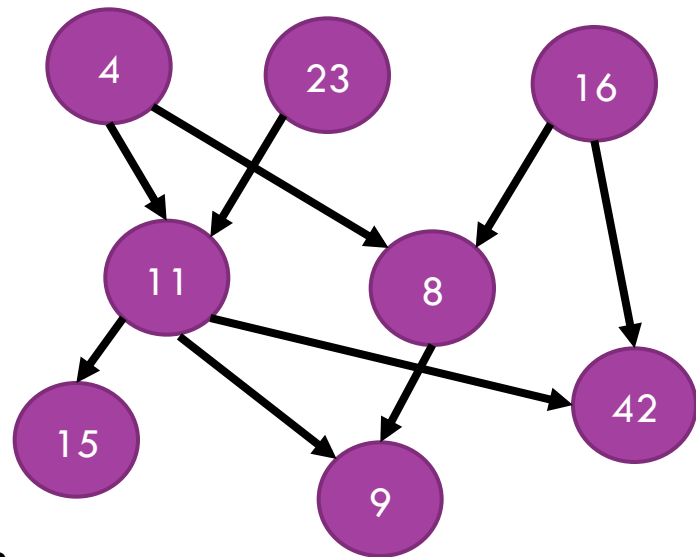
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|----|--|--|--|--|
| 23 | 16 | | | | |
|----|----|--|--|--|--|

L

| | | | | | |
|---|--|--|--|--|--|
| 4 | | | | | |
|---|--|--|--|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w **existente faça**

 remover o arco v,w de E

 se w não possuir mais arcs de entrada

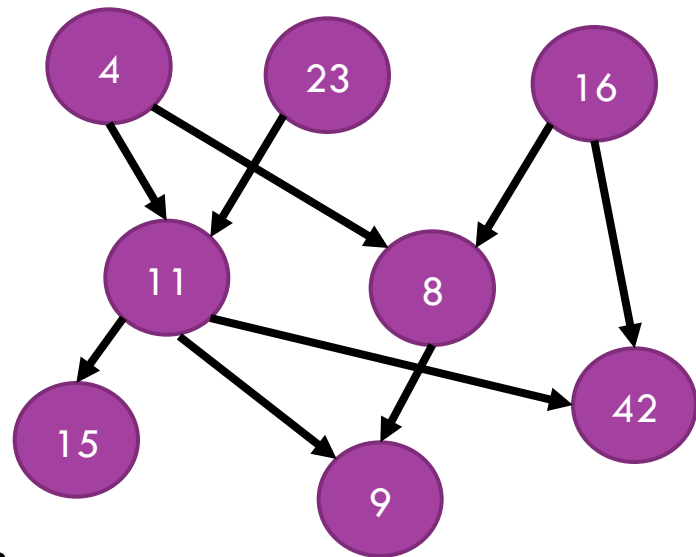
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|----|--|--|--|--|
| 23 | 16 | | | | |
|----|----|--|--|--|--|

L

| | | | | | |
|---|--|--|--|--|--|
| 4 | | | | | |
|---|--|--|--|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w **existente faça**

remover o arco v,w **de** E

se w não possuir mais arcs de entrada

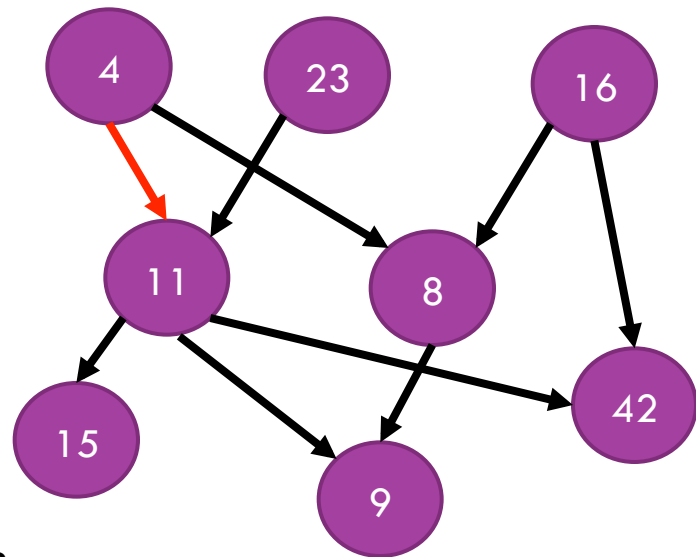
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|----|--|--|--|--|
| 23 | 16 | | | | |
|----|----|--|--|--|--|

L

| | | | | | |
|---|--|--|--|--|--|
| 4 | | | | | |
|---|--|--|--|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

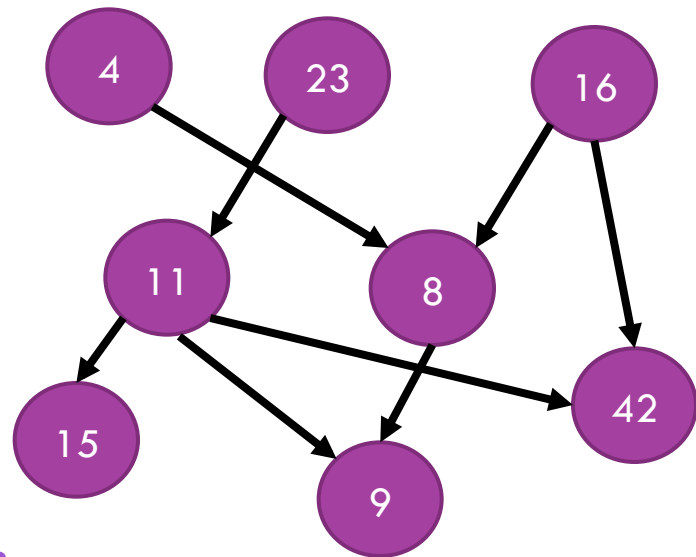
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|----|--|--|--|--|
| 23 | 16 | | | | |
|----|----|--|--|--|--|

L

| | | | | | |
|---|--|--|--|--|--|
| 4 | | | | | |
|---|--|--|--|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

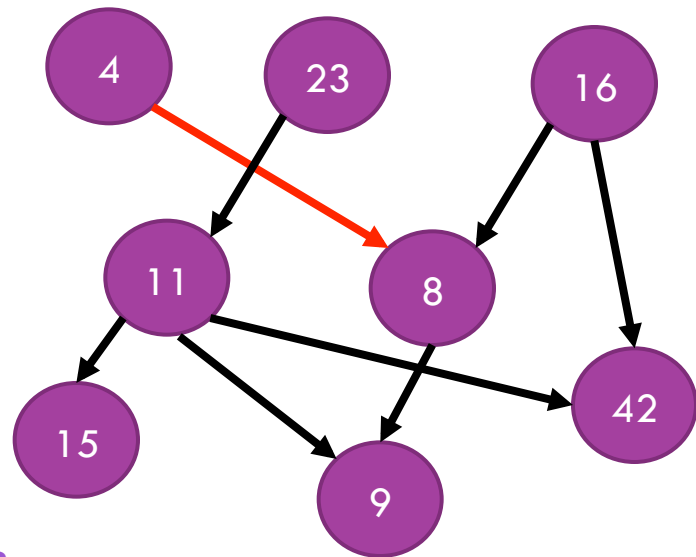
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|----|--|--|--|--|
| 23 | 16 | | | | |
|----|----|--|--|--|--|

L

| | | | | | |
|---|--|--|--|--|--|
| 4 | | | | | |
|---|--|--|--|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

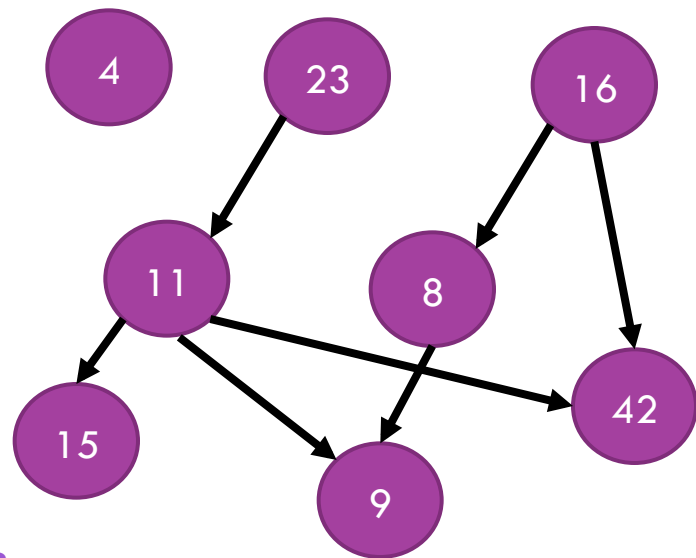
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|----|--|--|--|--|
| 23 | 16 | | | | |
|----|----|--|--|--|--|

L

| | | | | | |
|---|--|--|--|--|--|
| 4 | | | | | |
|---|--|--|--|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v, w existente faça

 remover o arco v, w de E

 se w não possuir mais arcs de entrada

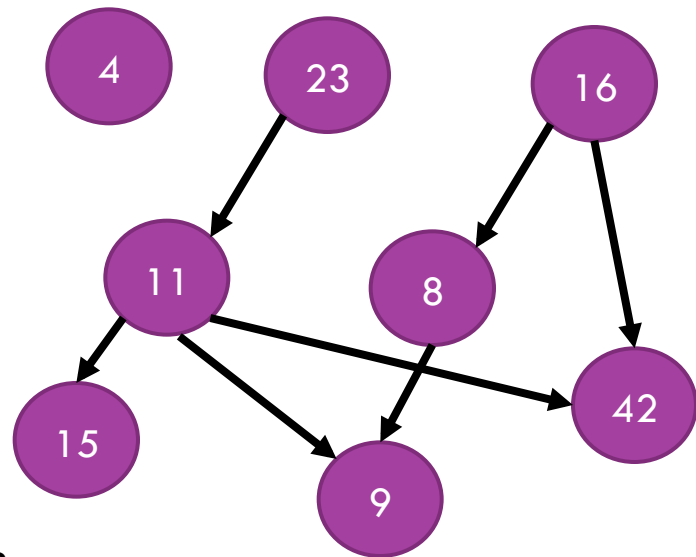
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S



L



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w **existente faça**

 remover o arco v,w de E

 se w não possuir mais arcs de entrada

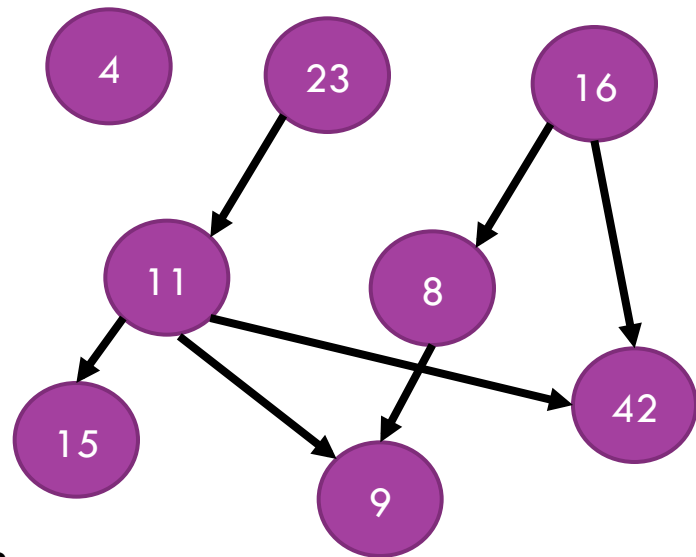
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S



L



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

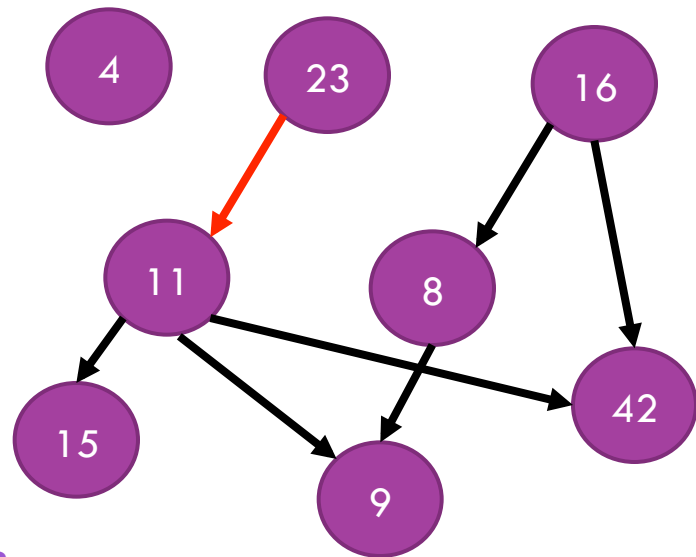
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S



L



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

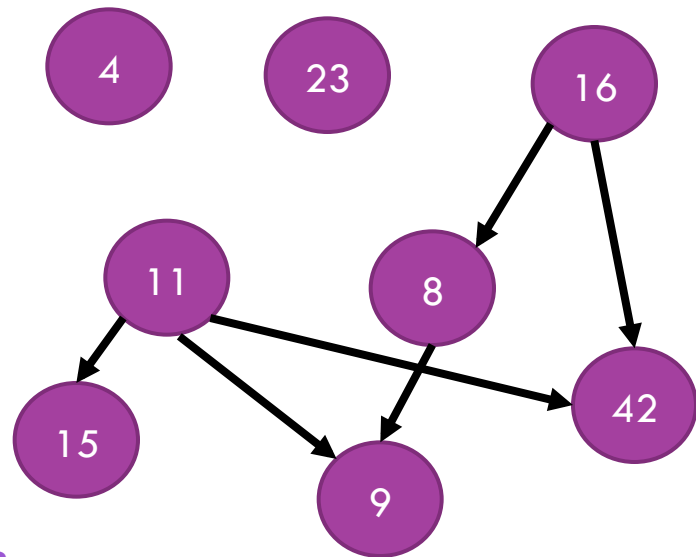
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|----|--|--|--|--|
| 16 | 11 | | | | |
|----|----|--|--|--|--|

L

| | | | | | |
|---|----|--|--|--|--|
| 4 | 23 | | | | |
|---|----|--|--|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v, w existente faça

 remover o arco v, w de E

 se w não possuir mais arcs de entrada

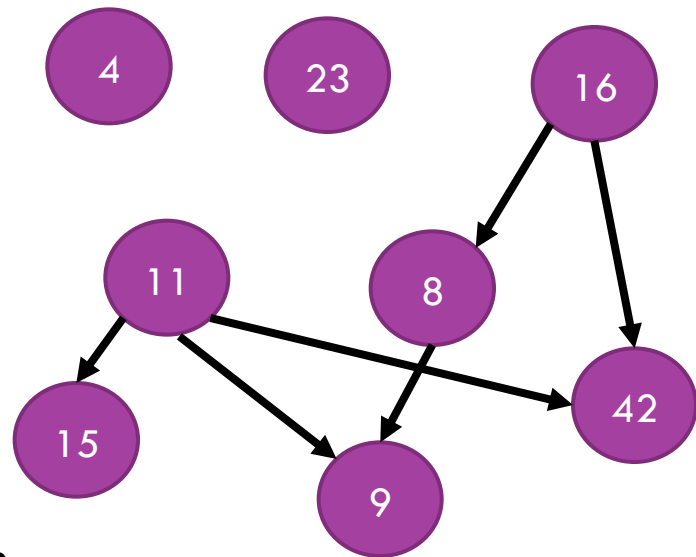
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S



L



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

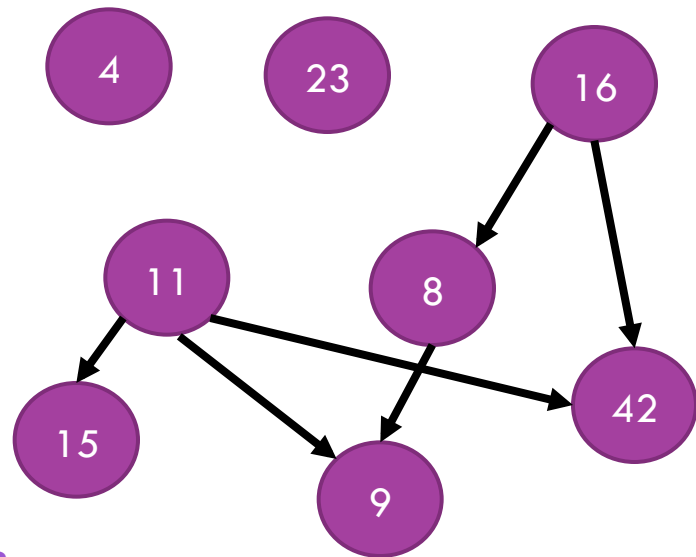
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S



L



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

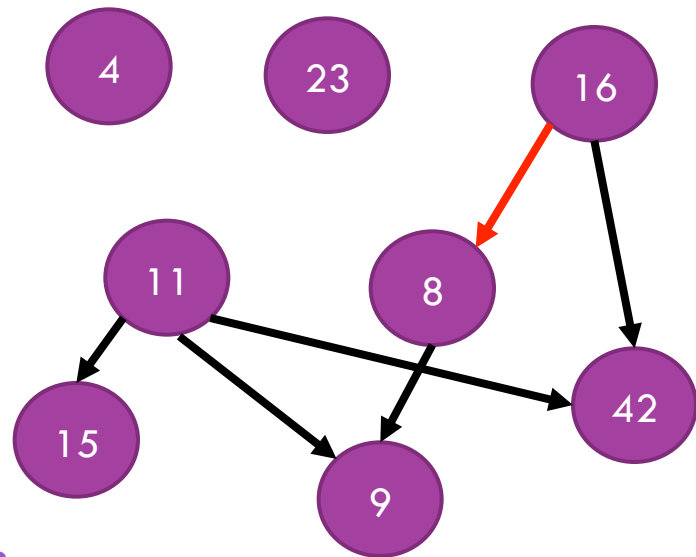
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S



L



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

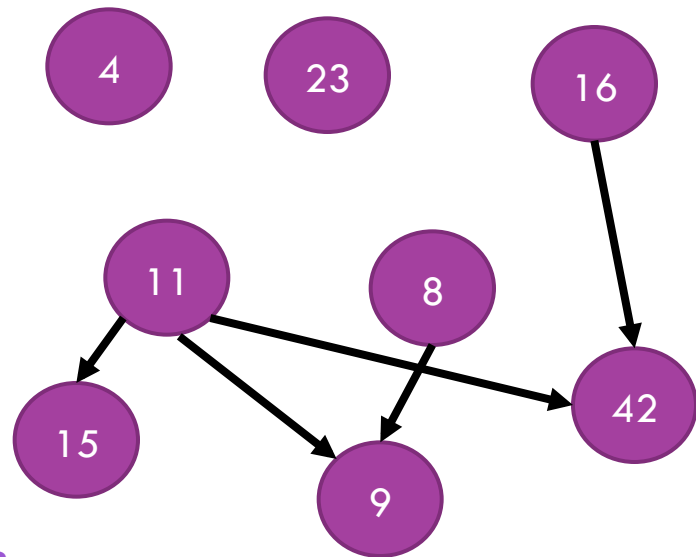
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|---|--|--|--|--|
| 11 | 8 | | | | |
|----|---|--|--|--|--|

L

| | | | | | |
|---|----|----|--|--|--|
| 4 | 23 | 16 | | | |
|---|----|----|--|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

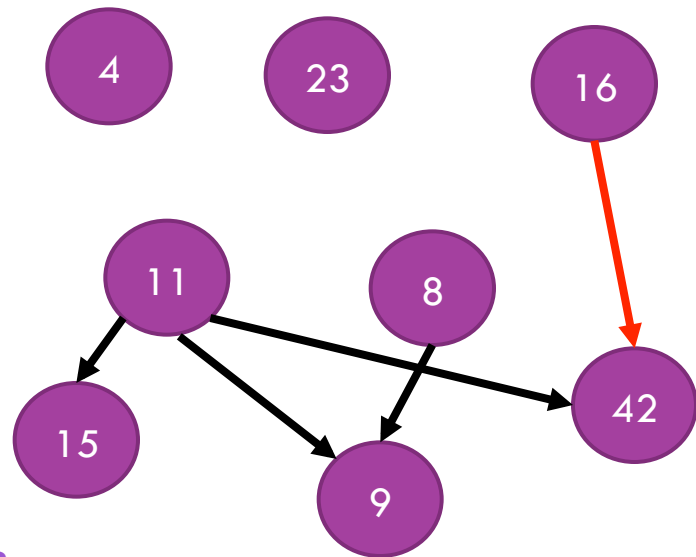
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|---|--|--|--|--|
| 11 | 8 | | | | |
|----|---|--|--|--|--|

L

| | | | | | |
|---|----|----|--|--|--|
| 4 | 23 | 16 | | | |
|---|----|----|--|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

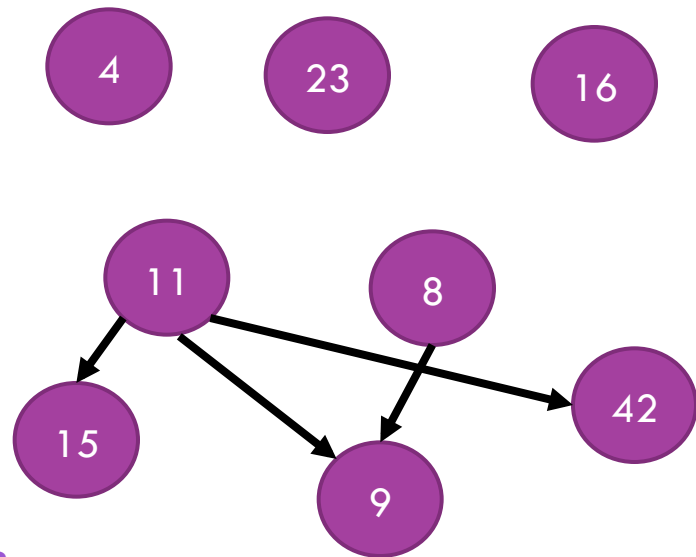
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|---|--|--|--|--|
| 11 | 8 | | | | |
|----|---|--|--|--|--|

L

| | | | | | |
|---|----|----|--|--|--|
| 4 | 23 | 16 | | | |
|---|----|----|--|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v, w existente faça

 remover o arco v, w de E

 se w não possuir mais arcs de entrada

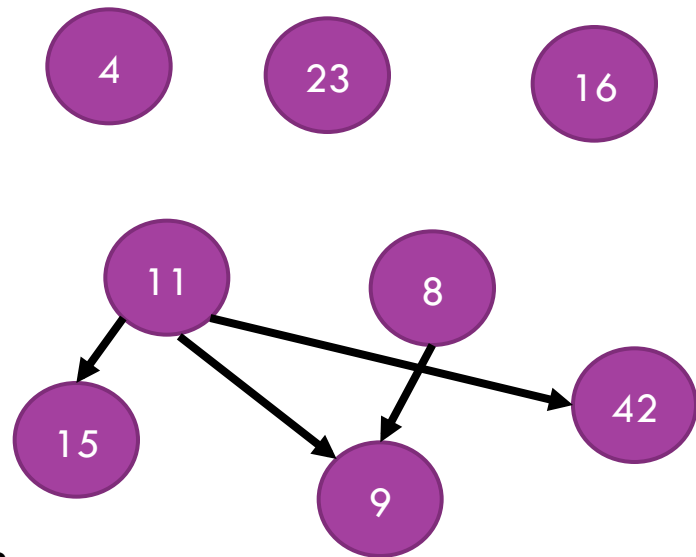
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S



L



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w **existente faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

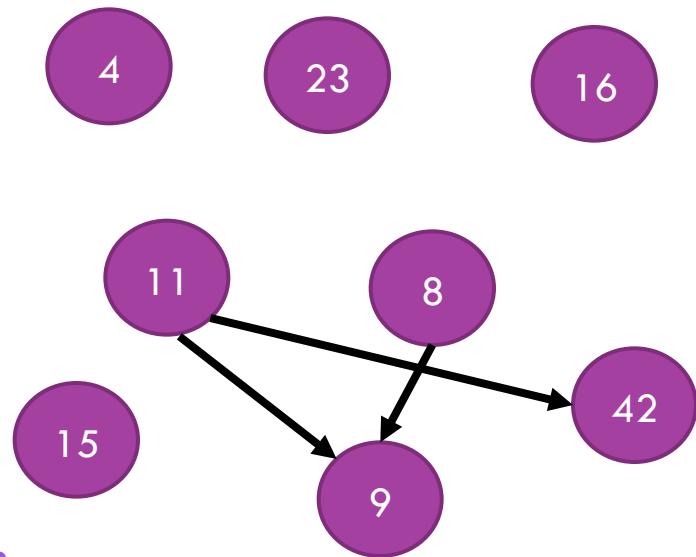
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|---|----|--|--|--|--|
| 8 | 15 | | | | |
|---|----|--|--|--|--|

L

| | | | | | |
|---|----|----|----|--|--|
| 4 | 23 | 16 | 11 | | |
|---|----|----|----|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

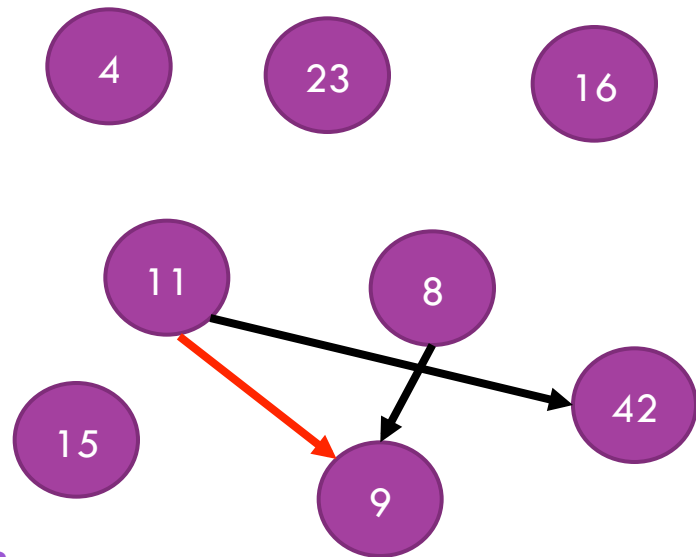
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|---|----|--|--|--|--|
| 8 | 15 | | | | |
|---|----|--|--|--|--|

L

| | | | | | |
|---|----|----|----|--|--|
| 4 | 23 | 16 | 11 | | |
|---|----|----|----|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w **existente faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

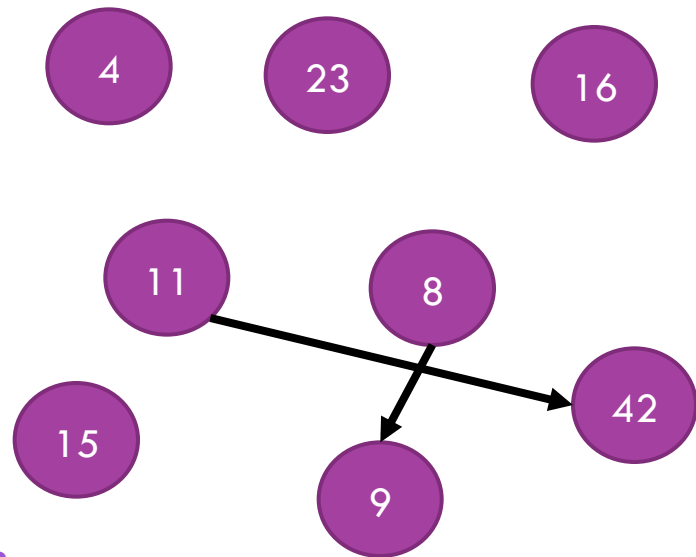
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|---|----|--|--|--|--|
| 8 | 15 | | | | |
|---|----|--|--|--|--|

L

| | | | | | |
|---|----|----|----|--|--|
| 4 | 23 | 16 | 11 | | |
|---|----|----|----|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

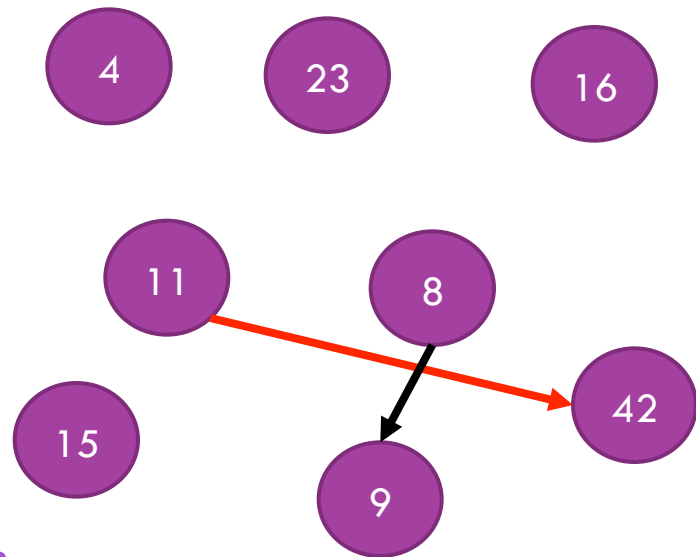
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|---|----|--|--|--|--|
| 8 | 15 | | | | |
|---|----|--|--|--|--|

L

| | | | | | |
|---|----|----|----|--|--|
| 4 | 23 | 16 | 11 | | |
|---|----|----|----|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

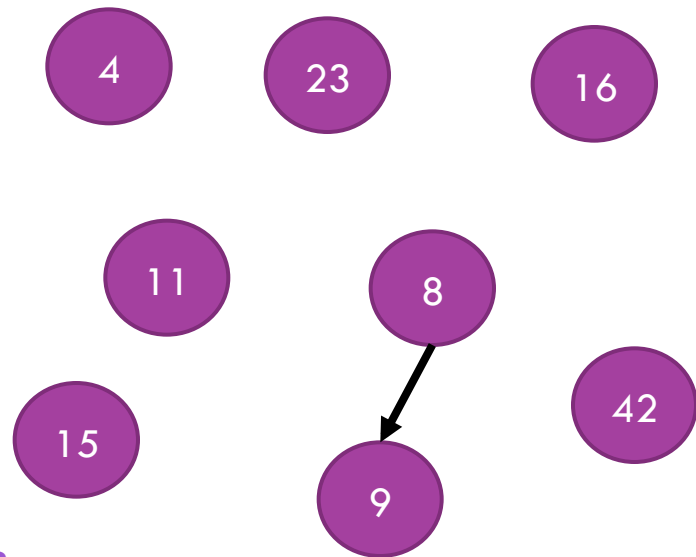
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|---|----|----|--|--|--|
| 8 | 15 | 42 | | | |
|---|----|----|--|--|--|

L

| | | | | | |
|---|----|----|----|--|--|
| 4 | 23 | 16 | 11 | | |
|---|----|----|----|--|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v, w existente faça

 remover o arco v, w de E

 se w não possuir mais arcs de entrada

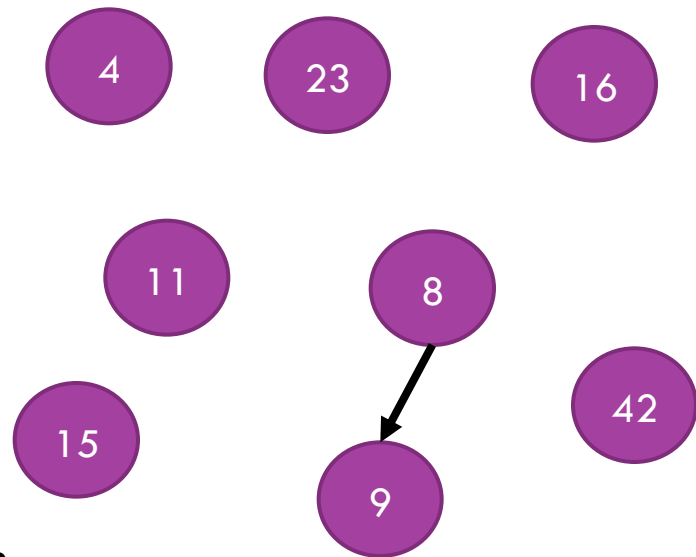
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|----|--|--|--|--|
| 15 | 42 | | | | |
|----|----|--|--|--|--|

L

| | | | | | |
|---|----|----|----|---|--|
| 4 | 23 | 16 | 11 | 8 | |
|---|----|----|----|---|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

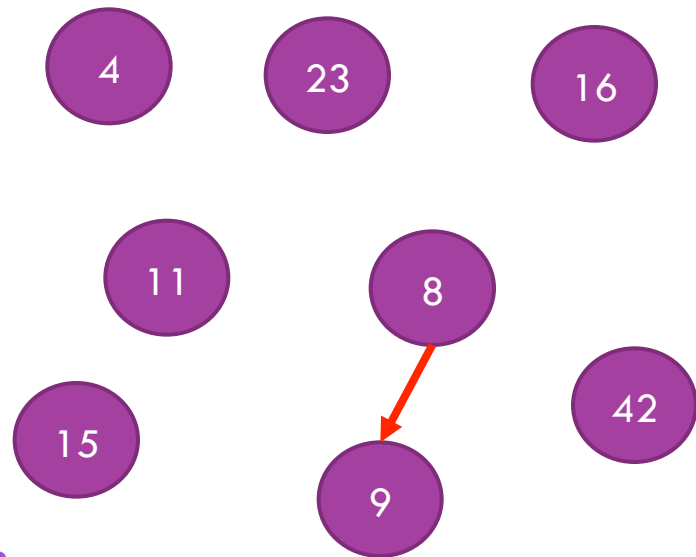
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



S

| | | | | | |
|----|----|--|--|--|--|
| 15 | 42 | | | | |
|----|----|--|--|--|--|

L

| | | | | | |
|---|----|----|----|---|--|
| 4 | 23 | 16 | 11 | 8 | |
|---|----|----|----|---|--|

Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ **faça**

 remover um vértice v de S

 inserir o vértice v em L

para cada arco v,w existente **faça**

 remover o arco v,w de E

se w não possuir mais arcs de entrada

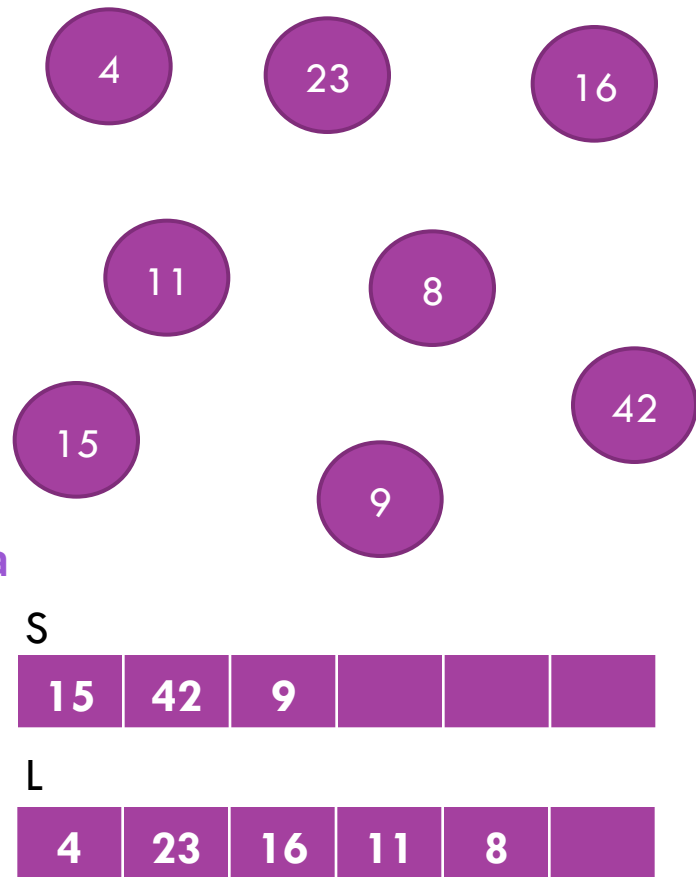
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v,w existente faça

 remover o arco v,w de E

 se w não possuir mais arcs de entrada

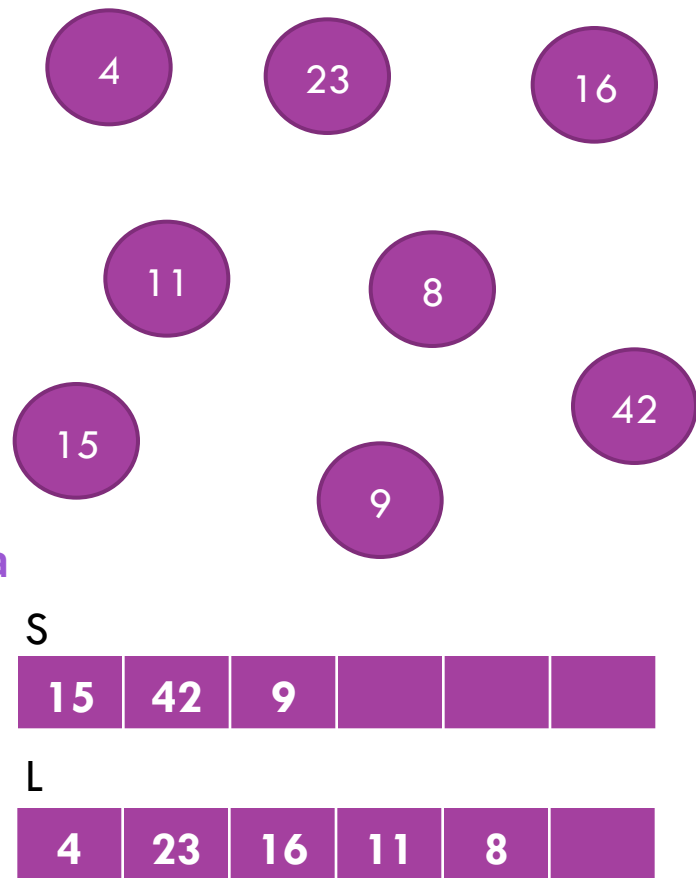
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v,w existente faça

 remover o arco v,w de E

 se w não possuir mais arcs de entrada

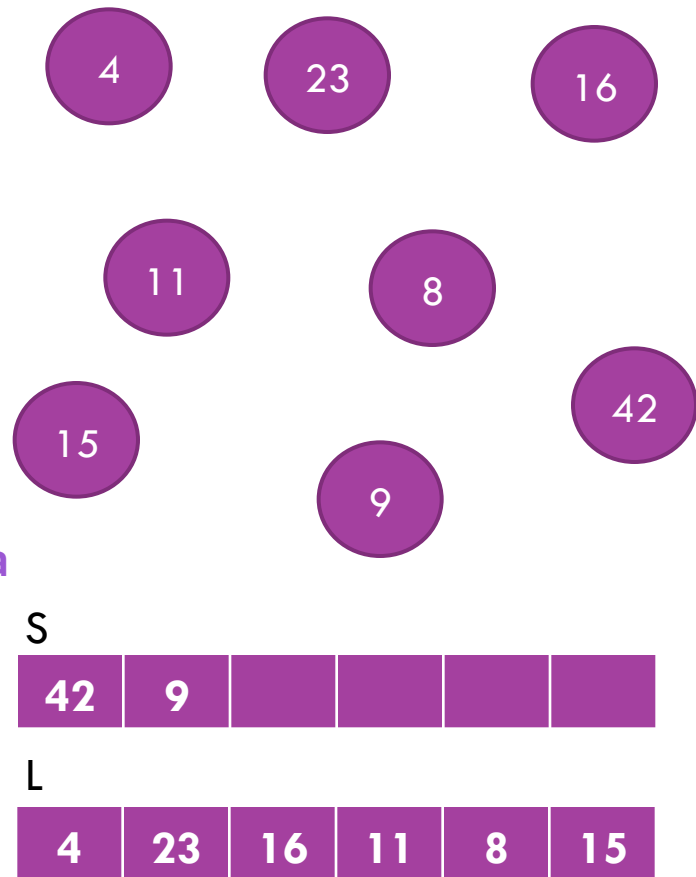
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcos de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v,w existente faça

 remover o arco v,w de E

 se w não possuir mais arcos de entrada

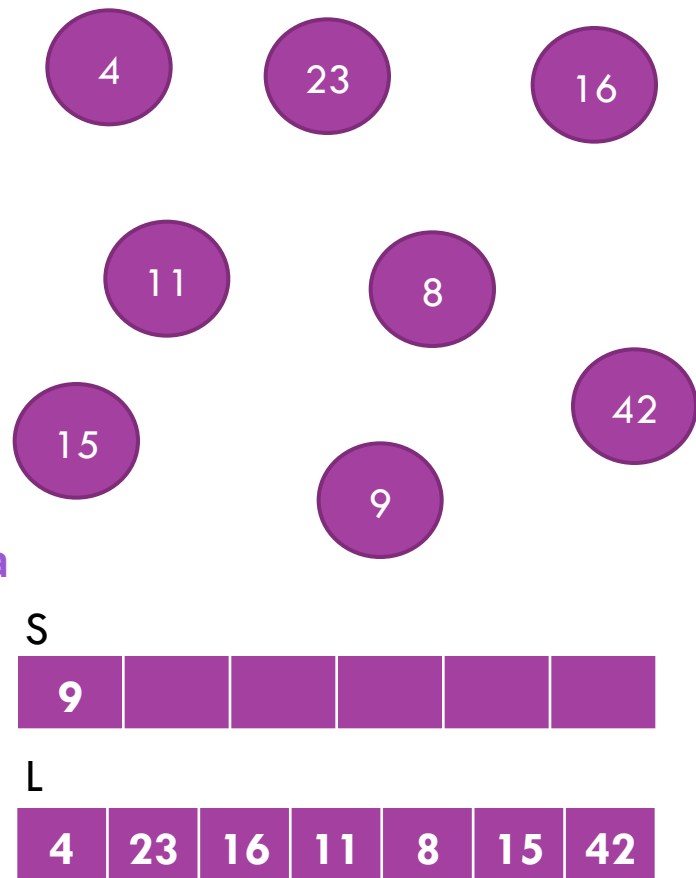
 inserir w em S

 Fim para

Fim enquanto

Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



Algoritmo de Kahn

$L = \emptyset$;

S = todos os vértices sem arcs de entrada;

Enquanto $S \neq \emptyset$ faça

 remover um vértice v de S

 inserir o vértice v em L

 para cada arco v,w existente faça

 remover o arco v,w de E

 se w não possuir mais arcs de entrada

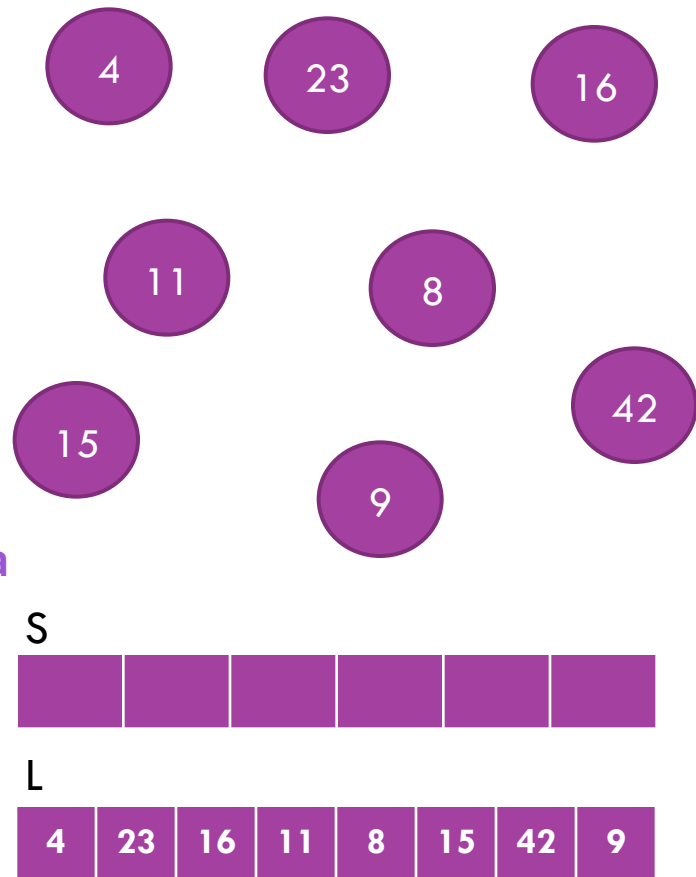
 inserir w em S

 Fim para

Fim enquanto

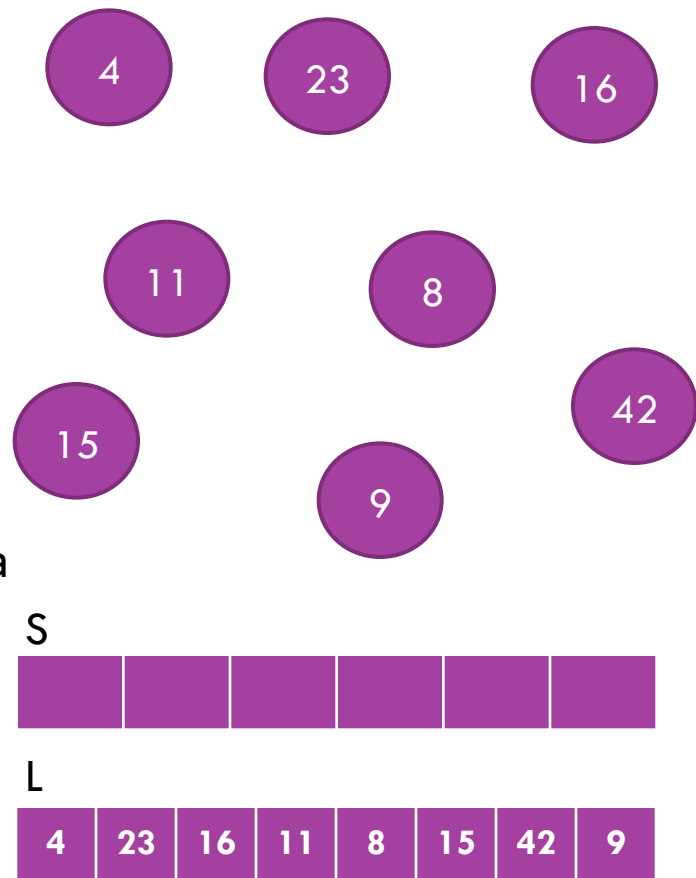
Se $E = \emptyset$ retorna L //lista ordenada

Senão o grafo possui pelo menos um ciclo



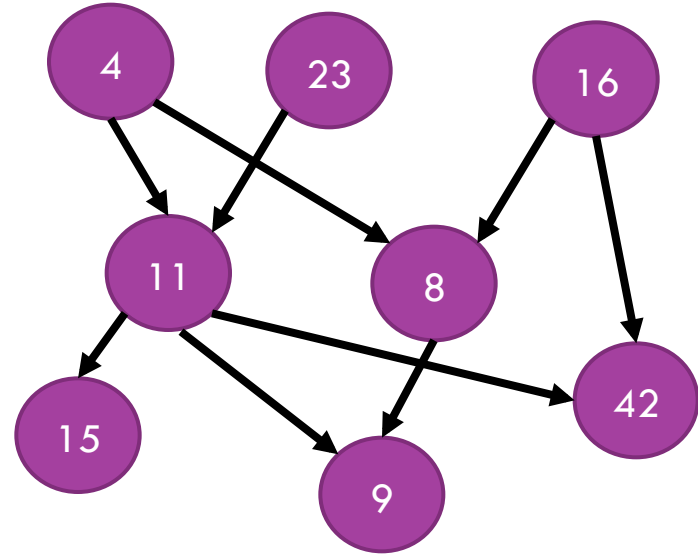
Algoritmo de Kahn

```
L = ∅;  
S = todos os vértices sem arcs de entrada;  
Enquanto S ≠ ∅ faça  
    remover um vértice v de S  
    inserir o vértice v em L  
    para cada arco v,w existente faça  
        remover o arco v,w de E  
        se w não possuir mais arcs de entrada  
            inserir w em S  
Fim para  
Fim enquanto  
Se E = ∅ retorna L //lista ordenada  
Senão o grafo possui pelo menos um ciclo
```



Algoritmo de Kahn

L



DFS e ordenação

42

- Podemos utilizar uma versão levemente alterada da busca em profundidade para realizar a ordenação topológica
- Basta, ao finalizar um vértice preto, inseri-lo no início de uma lista L

Algoritmo DFS – principal (visita)

```
timestamp = timestamp + 1;  
u.descoberta = timestamp;  
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        v.pai = u;  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
timestamp = timestamp+1;  
u.término = timestamp;
```

Algoritmo DFS – principal (visita)

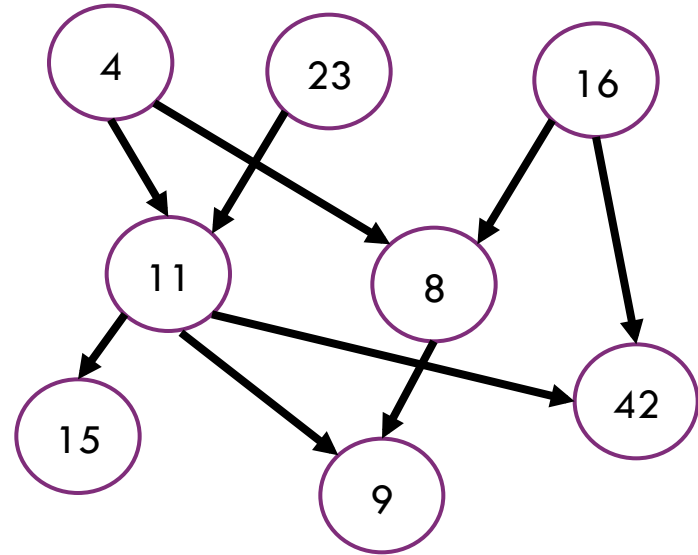
```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;
```

Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u)
```

Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



Algoritmo DFS – principal (visita)

```
u.cor = cinza;
```

```
Para cada vértice v vizinho de u faça
```

```
    se v.cor == branco
```

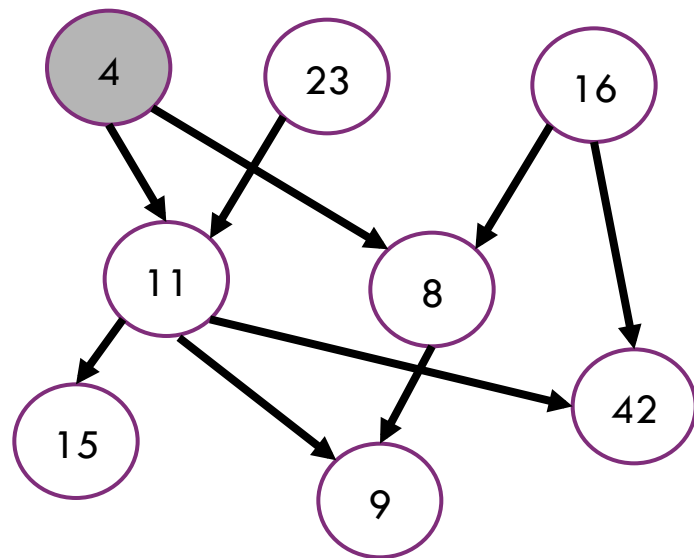
```
        Visitar(v);
```

```
    Fim se
```

```
Fim Para
```

```
u.cor = preto;
```

```
L.inserirInicio(u);
```



L



Algoritmo DFS – principal (visita)

```
u.cor = cinza;
```

```
Para cada vértice v vizinho de u faça
```

```
    se v.cor == branco
```

```
        Visitar(v);
```

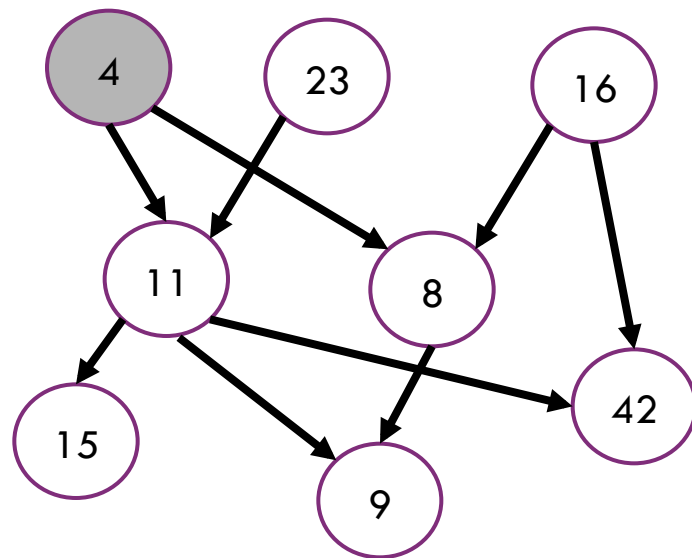
```
    Fim se
```

```
Fim Para
```

```
u.cor = preto;
```

```
L.inserirInicio(u);
```

L



Algoritmo DFS – principal (visita)

```
u.cor = cinza;
```

```
Para cada vértice v vizinho de u faça
```

```
    se v.cor == branco
```

```
        Visitar(v);
```

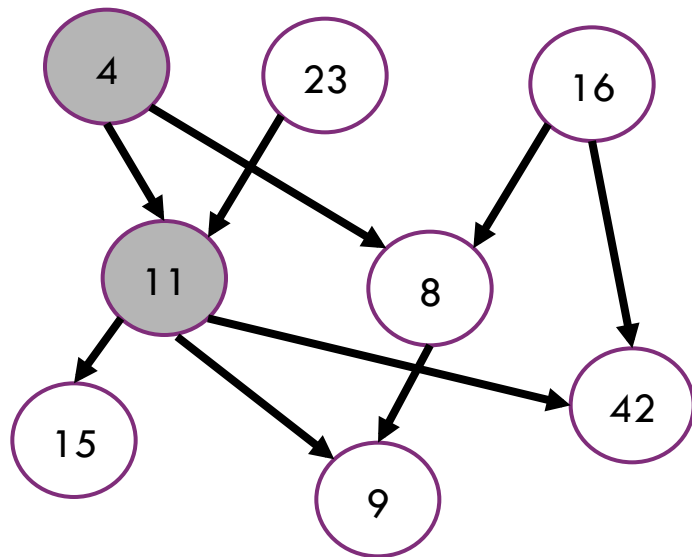
```
    Fim se
```

```
Fim Para
```

```
u.cor = preto;
```

```
L.inserirInicio(u);
```

L



Algoritmo DFS – principal (visita)

```
u.cor = cinza;
```

```
Para cada vértice v vizinho de u faça
```

```
    se v.cor == branco
```

```
        Visitar(v);
```

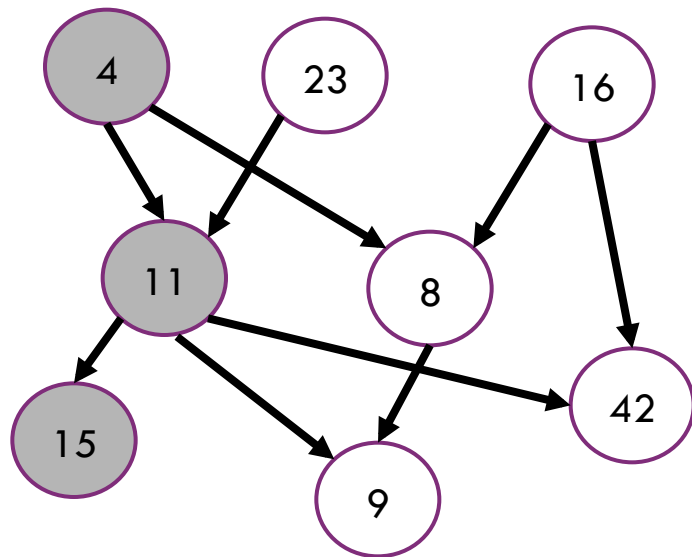
```
    Fim se
```

```
Fim Para
```

```
u.cor = preto;
```

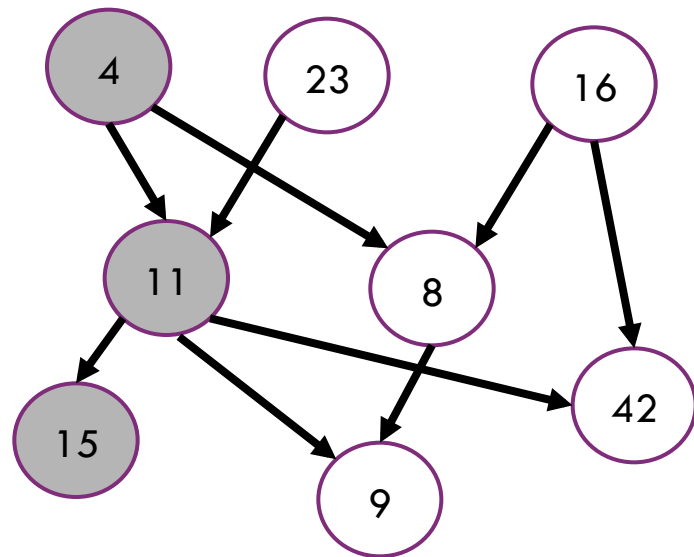
```
L.inserirInicio(u);
```

L



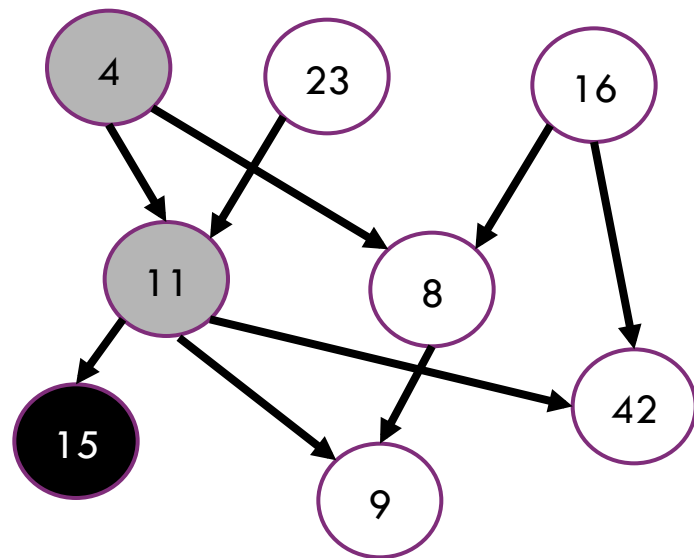
Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



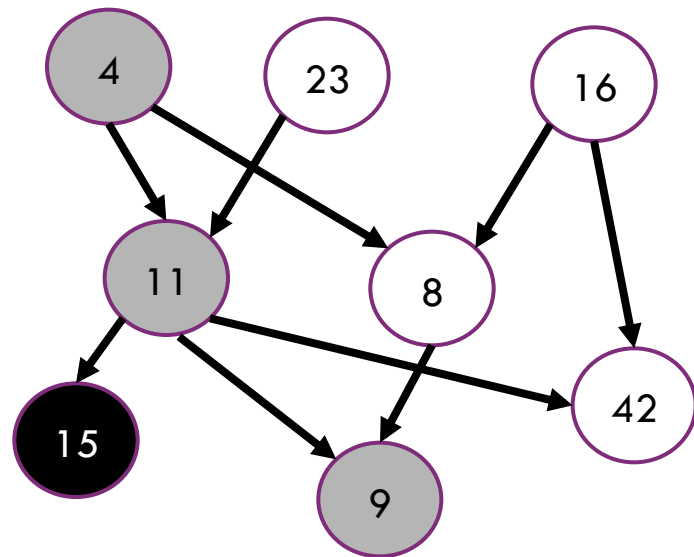
Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



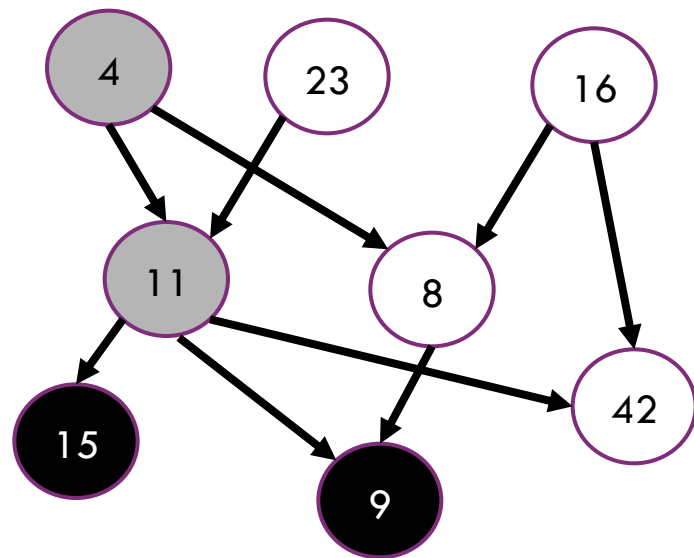
Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



Algoritmo DFS – principal (visita)

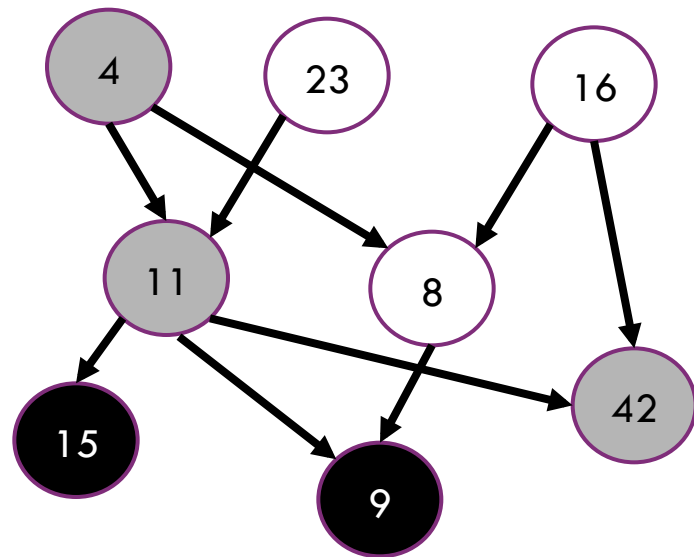
```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



Algoritmo DFS – principal (visita)

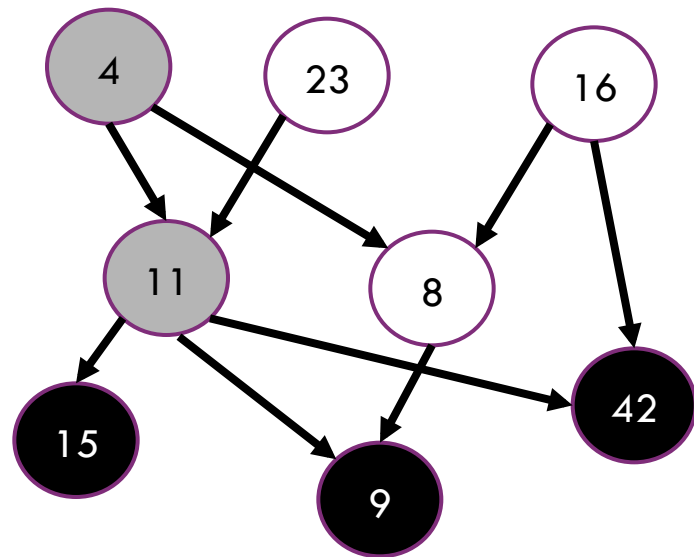
```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```

L



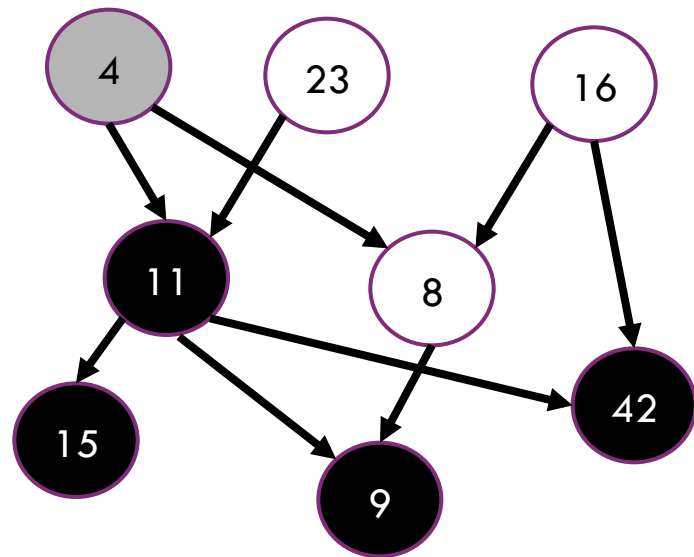
Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



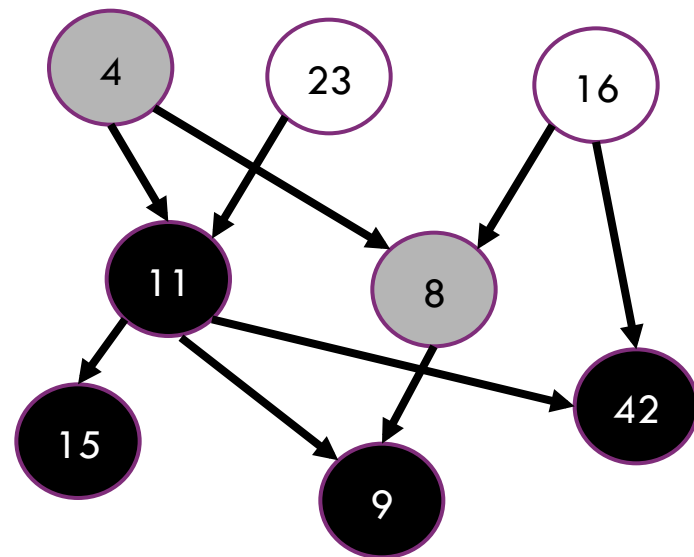
Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



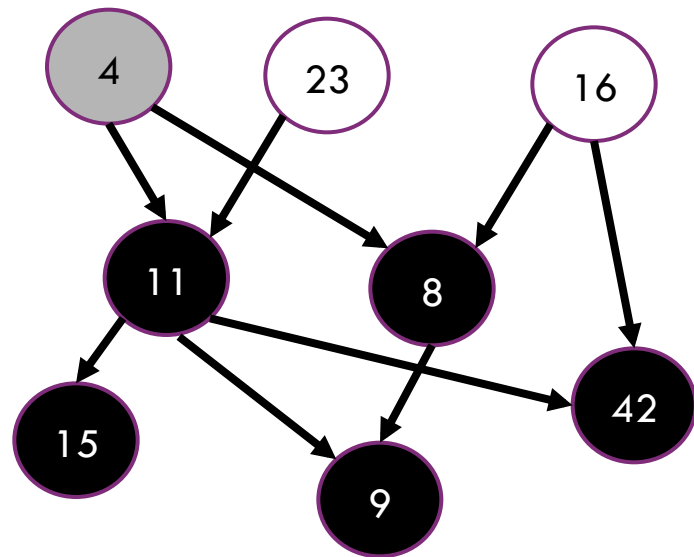
Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



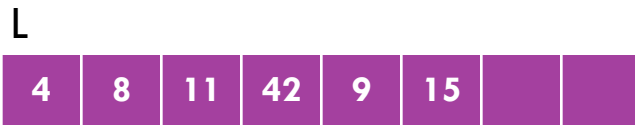
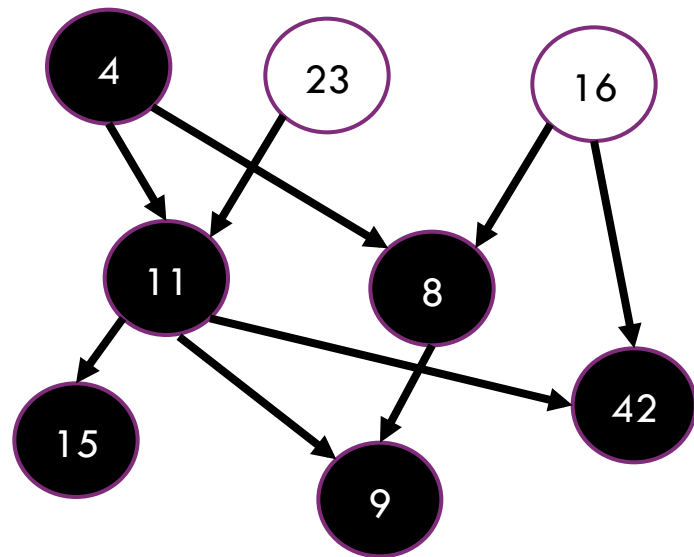
Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



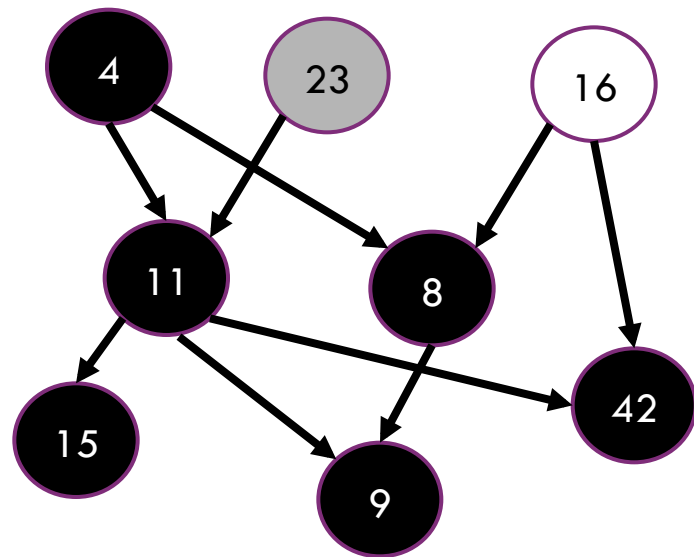
Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```

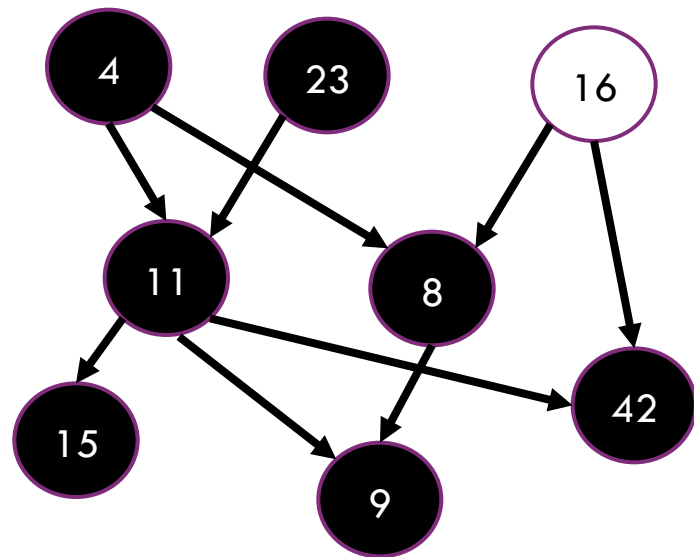


Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```

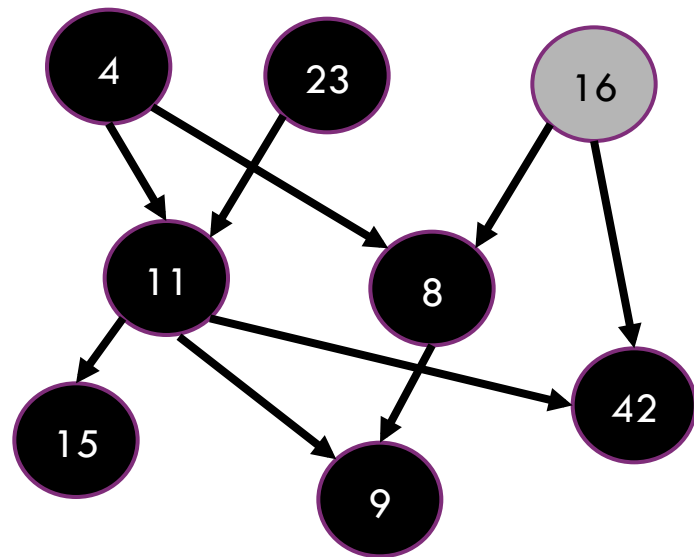
L

| | | | | | | | |
|----|---|---|----|----|---|----|--|
| 23 | 4 | 8 | 11 | 42 | 9 | 15 | |
|----|---|---|----|----|---|----|--|



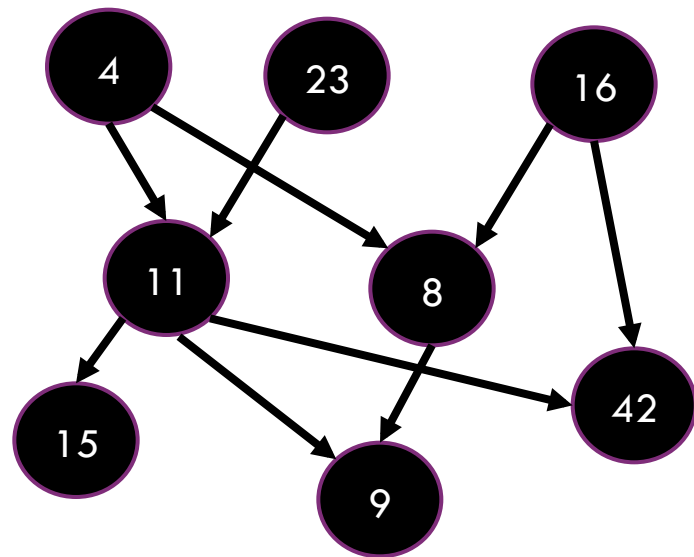
Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



Algoritmo DFS – principal (visita)

```
u.cor = cinza;  
Para cada vértice v vizinho de u faça  
    se v.cor == branco  
        Visitar(v);  
    Fim se  
Fim Para  
u.cor = preto;  
L.inserirInicio(u);
```



DFS e ordenação

65

- Como detectar ciclos?
- Relembrando: arestas de retorno
 - ▣ Ao alcançarmos um vértice cinza

Ordenação topológica

66

- Aplicações
 - ▣ Planejamento e sequenciamento de tarefas
 - ▣ Compilação de módulos
 - ▣ Dicionários
 - ▣ Pré-requisitos
 - ▣ Verificação de dependências (bibliotecas, etc)