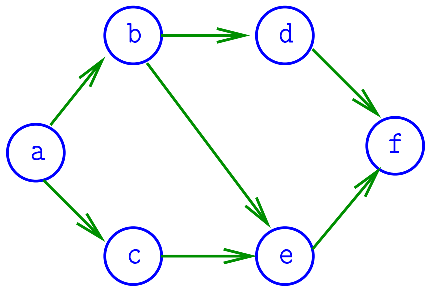
Digrafos

S 17.0, 17.1

Digrafos

Um **digrafo** (*directed graph*) consiste de um conjunto de vértices (bolas) e um conjunto de arcos (flechas)

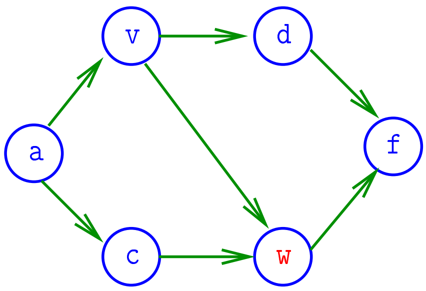
Exemplo: representação de um grafo



Arcos

Um **arco** é um par ordenado de vértices

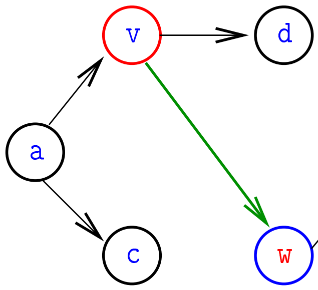
Exemplo: `v` e `w` são vértices e `v-w` é um arco



Ponta inicial e final

Para cada arco `v-w`, o vértice `v` é a **ponta inicial** e `w` é a **ponta final**

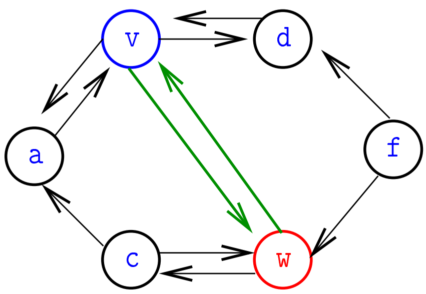
Exemplo: `v` é ponta inicial e `w` é ponta final de `v-w`



Arcos anti-paralelos

Dois arcos são **anti-paralelos** se a ponta inicial de um é ponta final do outro

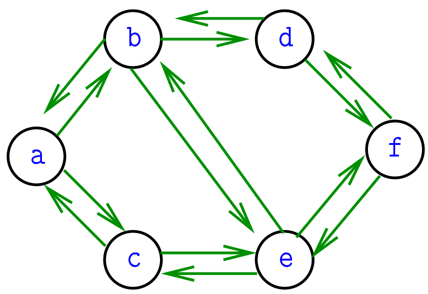
Exemplo: `v-w` e `w-v` são anti-paralelos



Digrafos simétricos

Um digrafo é **simétrico** se cada um de seus arcos é anti-paralelo a outro

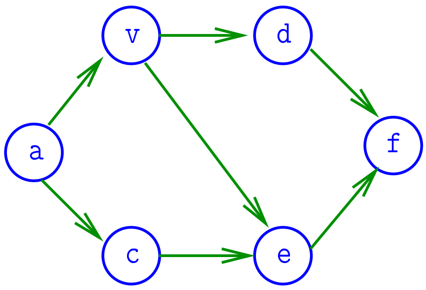
Exemplo: digrafo simétrico



Graus de entrada e saída

grau de entrada de $\mathtt{v}$= no. arcos com ponta final $\mathtt{v}$
grau de saída de $\mathtt{v}$ = no. arcos com ponta inicial $\mathtt{v}$

Exemplo: $\mathtt{v}$ tem grau de entrada 1 e de saída 2



Número de arcos

Quantos arcos, no máximo, tem um digrafo com $\mathtt{V}$ vértices?

Número de arcos

Quantos arcos, no máximo, tem um digrafo com $\mathtt{V}$ vértices?

A resposta é $\mathtt{V} \times (\mathtt{V} - 1) = \Theta(\mathtt{V}^2)$

digrafo **completo** = todo par ordenado de vértices distintos é arco

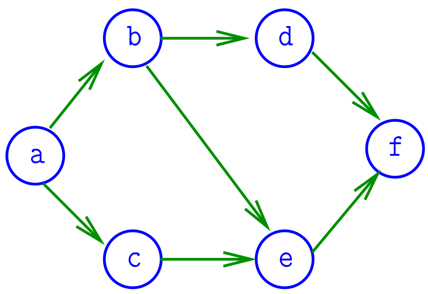
digrafo **denso** = tem "muitos" muitos arcos

digrafo **esparso** = tem "poucos" arcos

Especificação

Digrafos podem ser especificados através de sua lista de arcos

Exemplo:



d-f
b-d
a-c
b-e
e-f
a-b

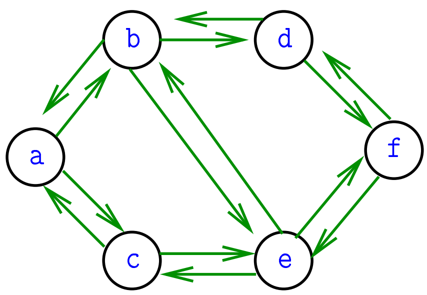
Grafos

S 17.0, 17.1

Grafos

Um **grafo** é um digrafo **simétrico**

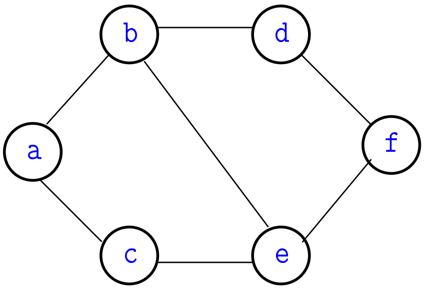
Exemplo: um grafo



Grafos

Um **grafo** é um digrafo **simétrico**

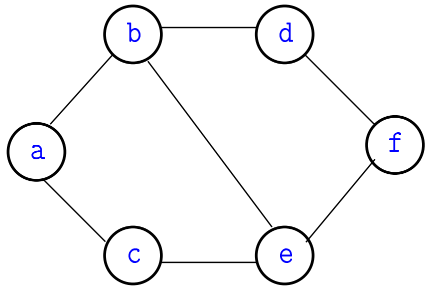
Exemplo: representação usual



Arestas

Uma **aresta** é um par de arcos anti-paralelos.

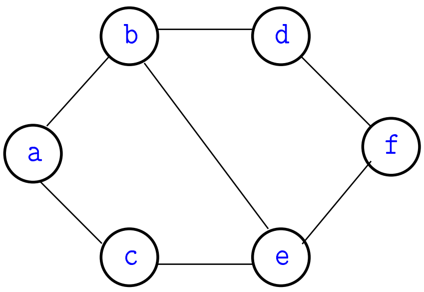
Exemplo: `b-a` e `a-b` são a **mesma** aresta



Especificação

Grafos podem ser especificados através de sua lista de arestas

Exemplo:



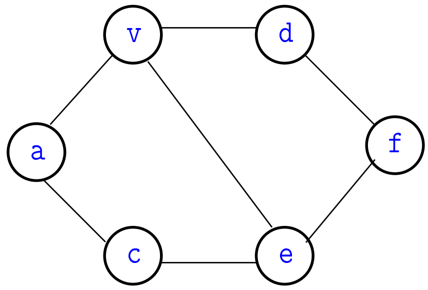

```
f-d
b-d
c-a
e-b
e-f
a-b
```

Graus de vértices

Em um grafo

grau de `v` = número de arestas com ponta em `v`

Exemplo: `v` tem grau 3



Número de arestas

Quantas arestas, no máximo, tem um grafo com V vértices?

Número de arestas

Quantas arestas, no máximo, tem um grafo com V vértices?

A resposta é $V \times (V-1)/2 = \Theta(V^2)$

grafo **completo** = todo par **não**-ordenado de vértices distintos é aresta

Matrizes de adjacência

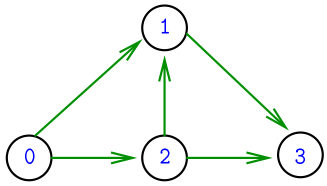
S 17.3

Matriz de adjacência de digrafos

Matriz de adjacência de um digrafo tem linhas e colunas indexadas por vértices:

`adj[v][w]` = 1 se `v-w` é um arco
`adj[v][w]` = 0 em caso contrário

Exemplo:



	0	1	2	3
0	0	1	1	0
1	0	0	0	1
2	0	1	0	1
3	0	0	0	0

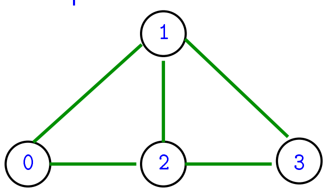
Consumo de espaço: $\Theta(V^2)$ fácil de implementar

Matriz de adjacência de grafos

Matriz de adjacência de um grafo tem linhas e colunas indexadas por vértices:

`adj[v][w]` = 1 se `v-w` é um aresta
`adj[v][w]` = 0 em caso contrário

Exemplo:

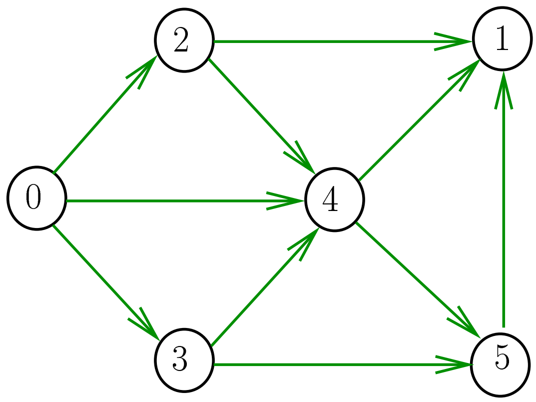


	0	1	2	3
0	0	1	1	0
1	1	0	1	1
2	1	1	0	1
3	0	1	1	0

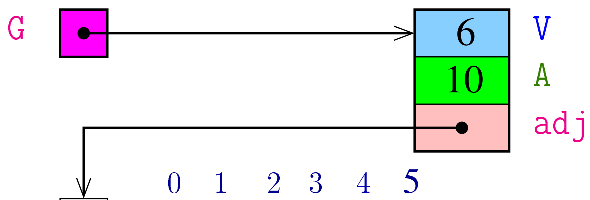
Consumo de espaço: $\Theta(V^2)$ fácil de implementar

Digrafo

`Digraph G`



Estruturas de dados



	0	1	2	3	4	5
0	0	0	1	1	1	0
1	0	0	0	0	0	0
2	0	1	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	1
5	0	1	0	0	0	0

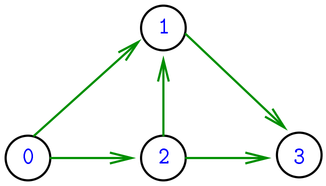
Vetor de listas de adjacência

S 17.4

Vetor de listas de adjacência de digrafos

Na representação de um digrafo através de **listas de adjacência** tem-se, para cada vértice `v`, uma lista dos vértices que são vizinhos `v`.

Exemplo:



```
0:  1,  2
1:   3
2:  1,  3
3:
```

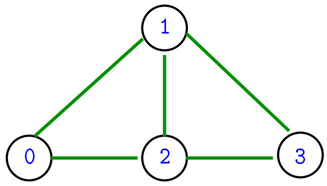
Consumo de espaço: $\Theta(\mathtt{V}+\mathtt{A})$ (linear)

Manipulação eficiente

Vetor de lista de adjacência de grafos

Na representação de um grafo através de **listas de adjacência** tem-se, para cada vértice `v`, uma lista dos vértices que são pontas de arestas incidentes a `v`

Exemplo:



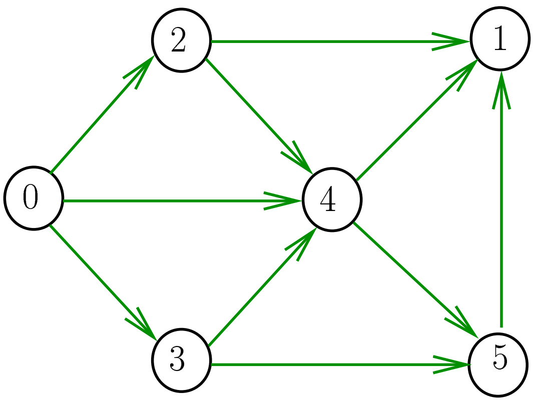
```
0:  1,   2
1:  3,  0,  2
2:  1,  3,  0
3:  1,   2
```

Consumo de espaço: $\Theta(\mathtt{V}+\mathtt{A})$ (linear)

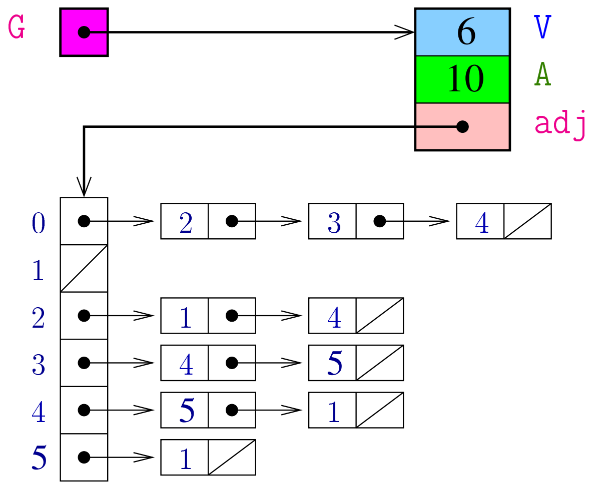
Manipulação eficiente

Digrafo

`Digraph G`



Estruturas de dados



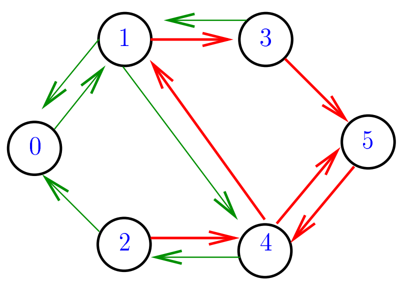
Caminhos em digrafos

S 17.1

Caminhos

Um **caminho** num digrafo é qualquer seqüência da forma $\mathtt{v}_0$–$\mathtt{v}_1$–$\mathtt{v}_2$–…–$\mathtt{v}_{k-1}$–$\mathtt{v}_p$, onde $\mathtt{v}_{k-1}$-$\mathtt{v}_k$ é um arco para $k = 1, \ldots, p$.

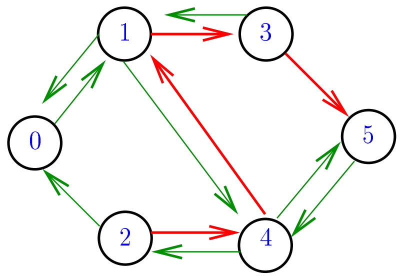
Exemplo: 2-4-1-3-5-4-5 é um caminho com **origem** 2 é **término** 5



Caminhos simples

Um caminho é **simples** se não tem vértices repetidos

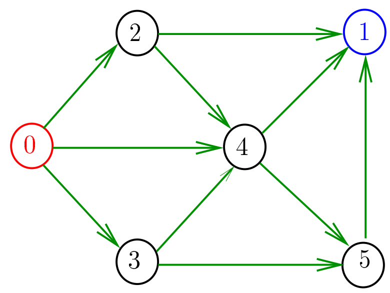
Exemplo: 2-4-1-3-5 é um caminho simples de 2 a 5



Procurando um caminho

Problema: dados um digrafo `G` e dois vértices `s` e `t` decidir se existe um caminho de `s` a `t`

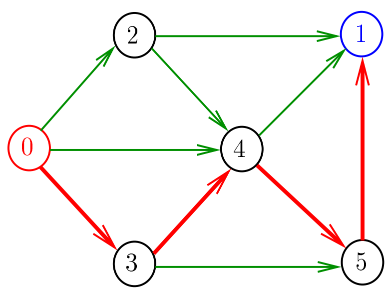
Exemplo: para `s` = 0 e `t` = 1 a resposta é SIM



Procurando um caminho

Problema: dados um digrafo `G` e dois vértices `s` e `t` decidir se existe um caminho de `s` a `t`

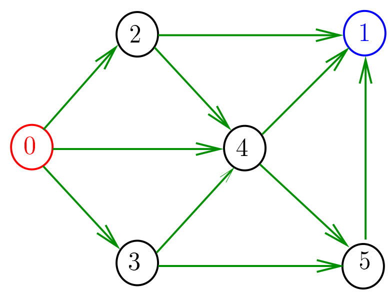
Exemplo: para `s` = 0 e `t` = 1 a resposta é SIM



Procurando um caminho

Problema: dados um digrafo `G` e dois vértices `s` e `t` decidir se existe um caminho de `s` a `t`

Exemplo: para `s` = 5 e `t` = 4 a resposta é NÃO

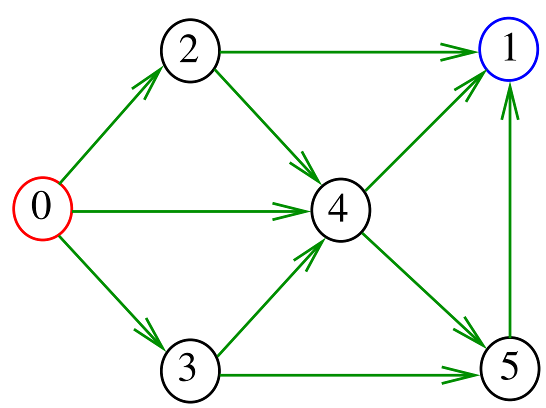


DFSpaths

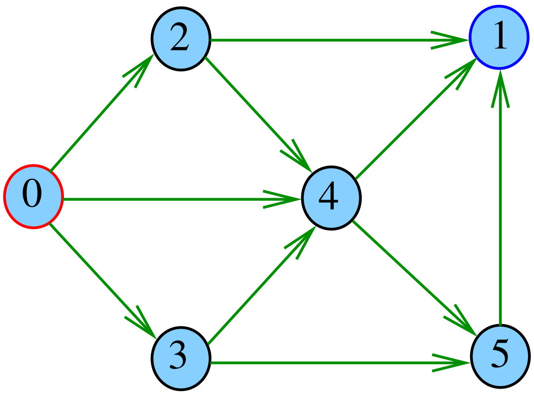
Recebe um digrafo `G` e vértices `s` e `t` e devolve **1** se existe um caminho de `s` a `t` ou devolve **0** em caso contrário
Supõe que o digrafo tem no máximo `maxV` vértices.

public class DFSpaths
public boolean hasPathTo(**int** `v`)
public Iterable<Integer> pathTo(**int** `v`)

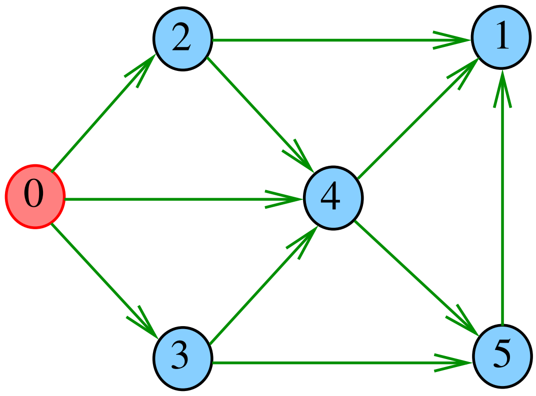
DFSpaths(`G`,0,1)



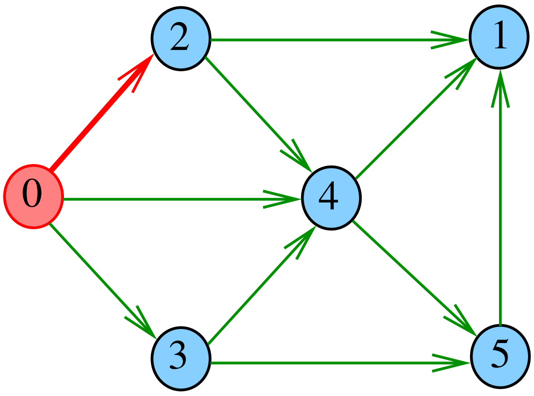
DFSpaths(G,0,1)



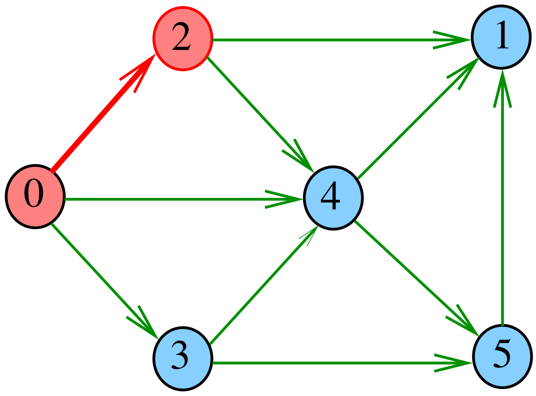
dfs(G,0)



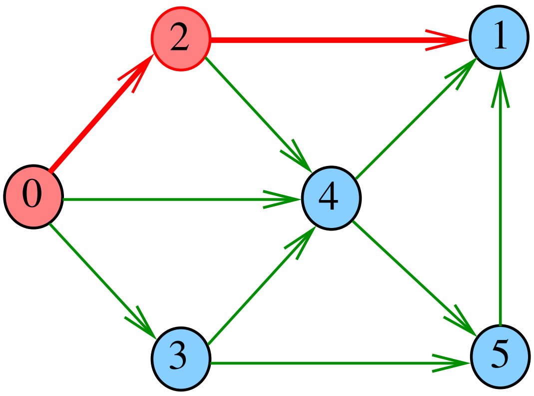
dfs(G,0)



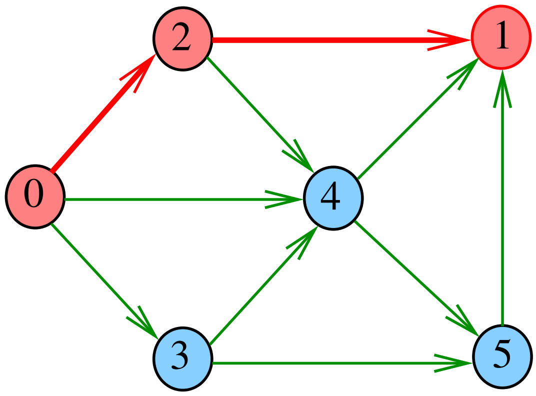
dfs(G,2)



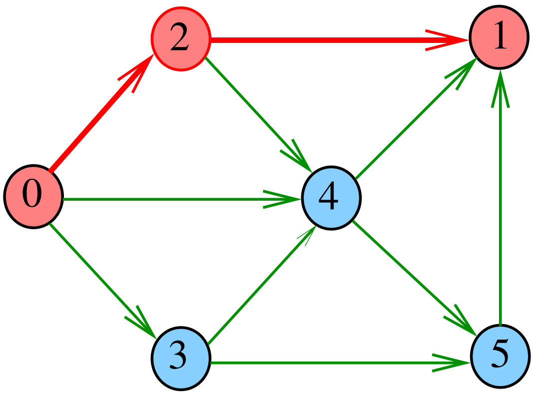
dfs(G,2)



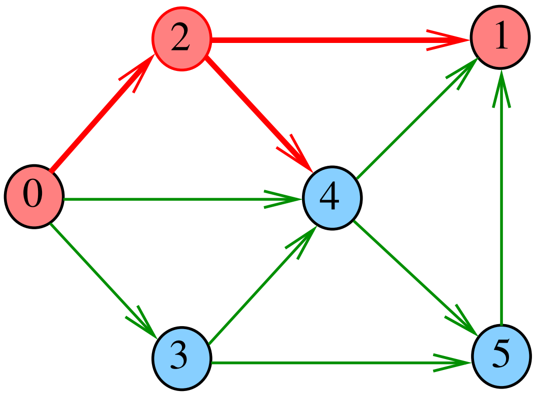
dfs(G,1)



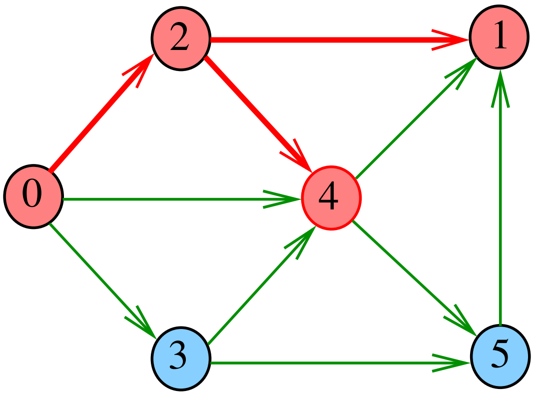
dfs(G,2)



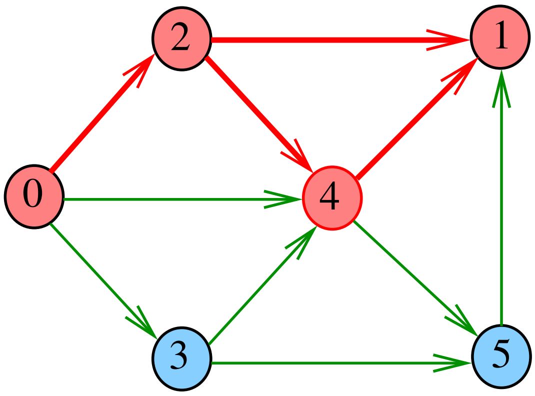
dfs(G,2)



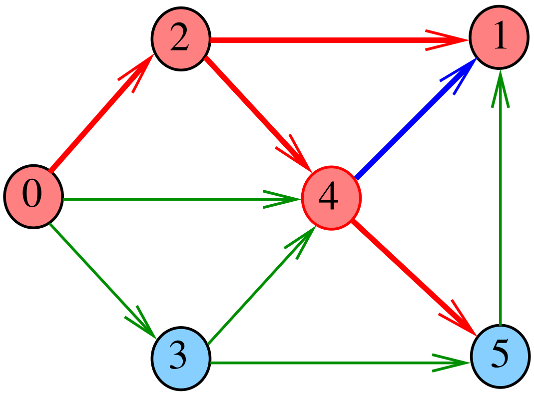
dfs(G,4)



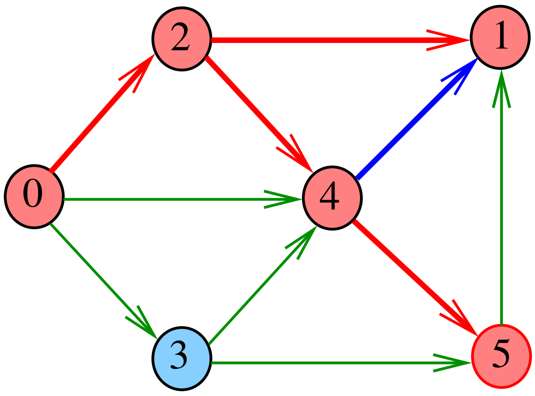
dfs(G,4)



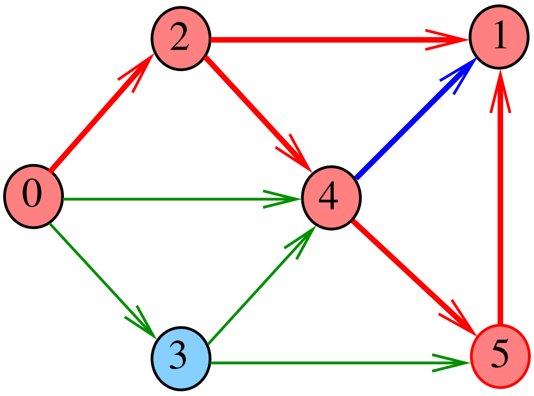
dfs(G,4)



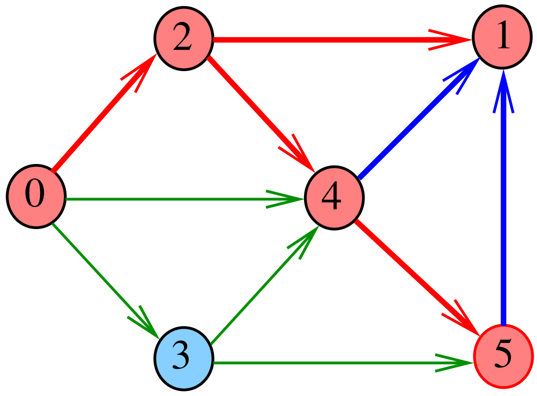
dfs(G,5)



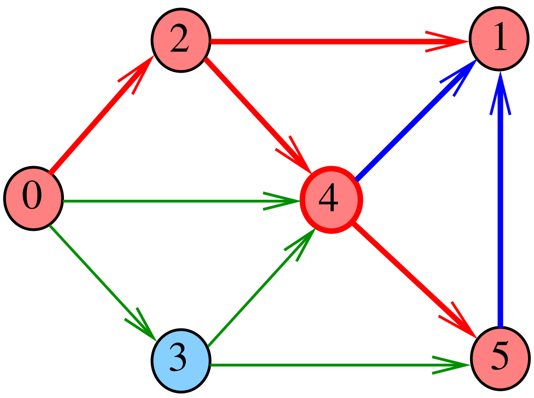
dfs(G,5)



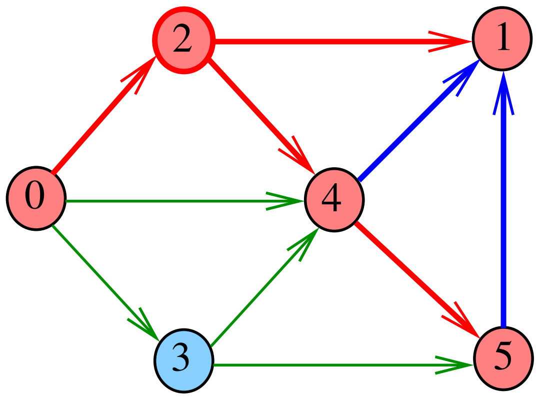
dfs(G,5)



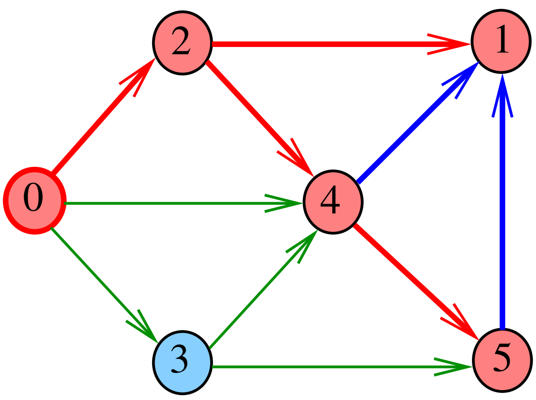
dfs(G,4)



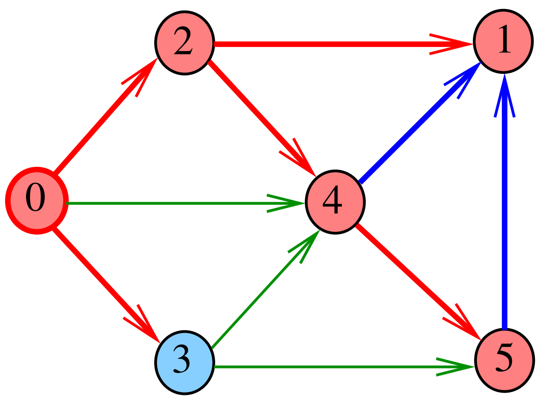
dfs(G,2)



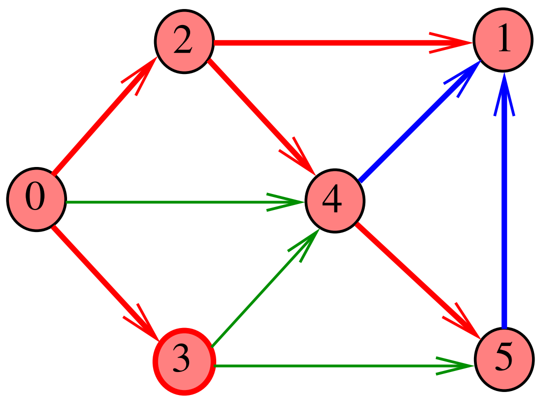
dfs(G,0)



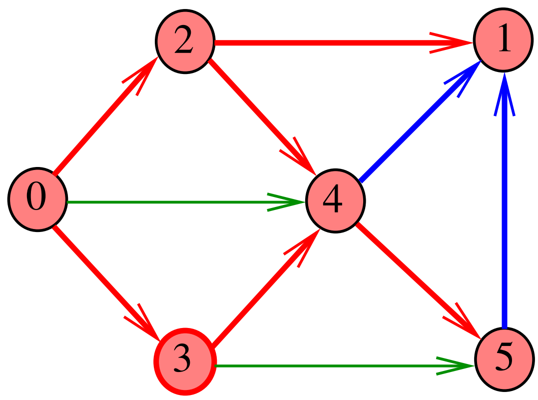
dfs(G,0)



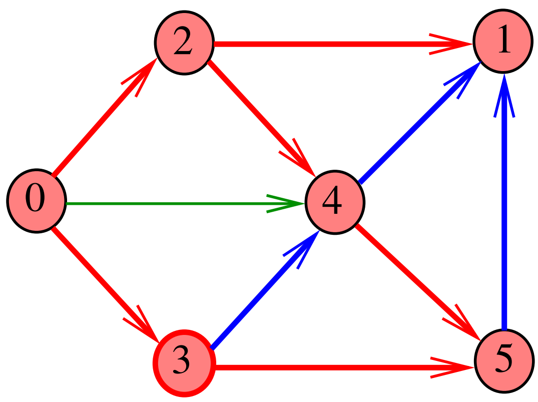
dfs(G,3)



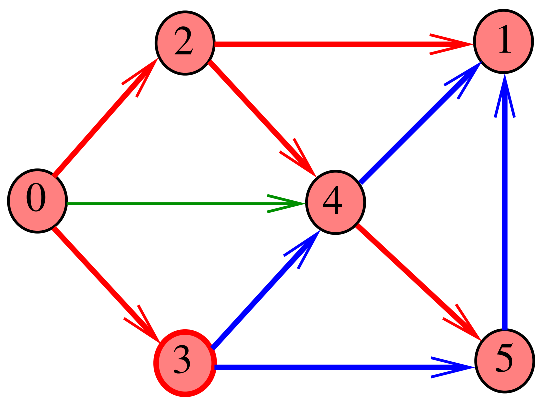
dfs(G,3)



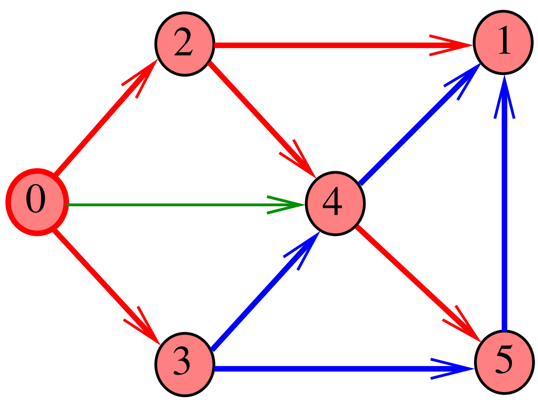
dfs(G,3)



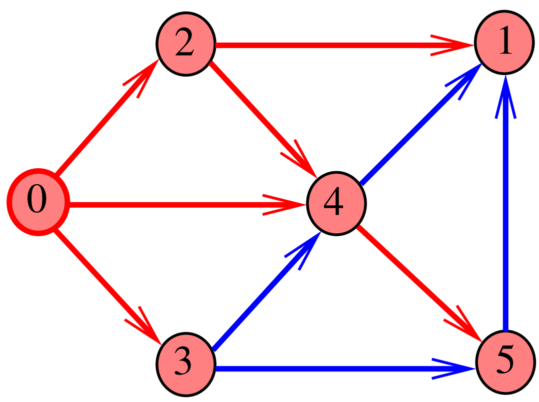
dfs(G,3)



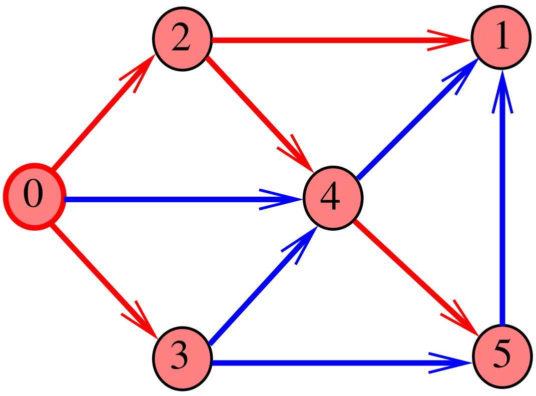
dfs(G,0)



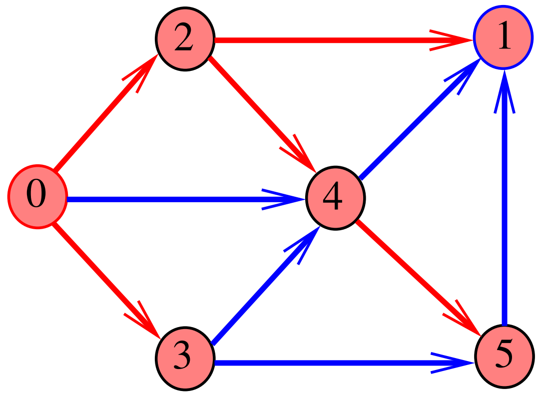
dfs(G,0)



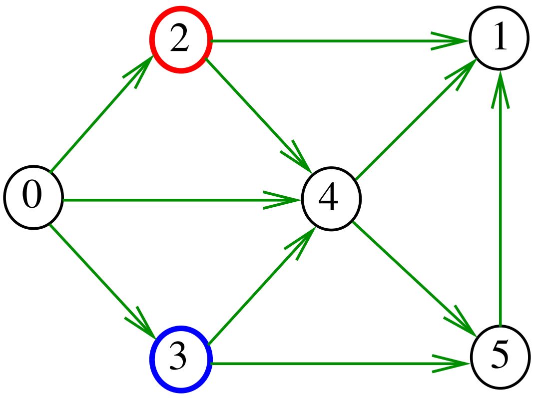
dfs(G,0)



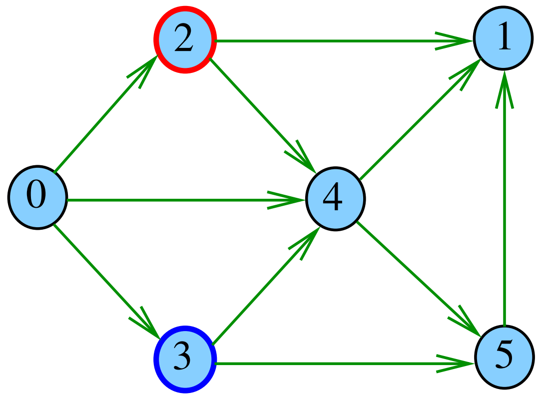
DFSpaths(G,0,1)



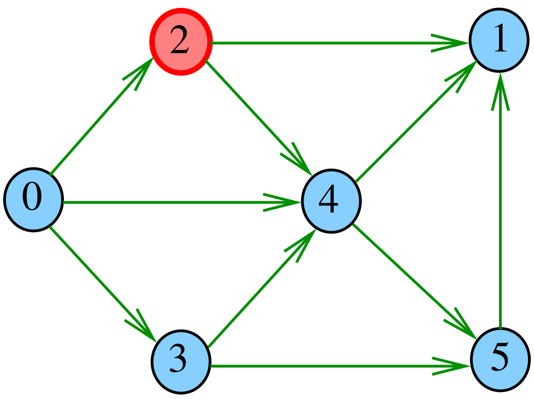
DFSpaths(G,2,3)



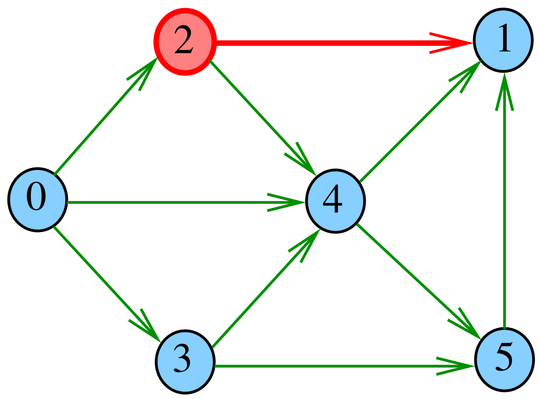
DFSpaths(G,2,3)



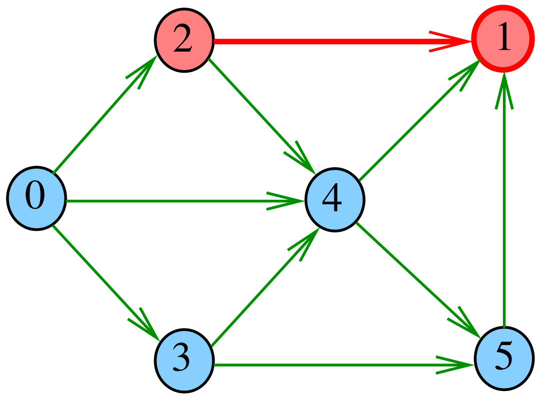
dfs(G,2)



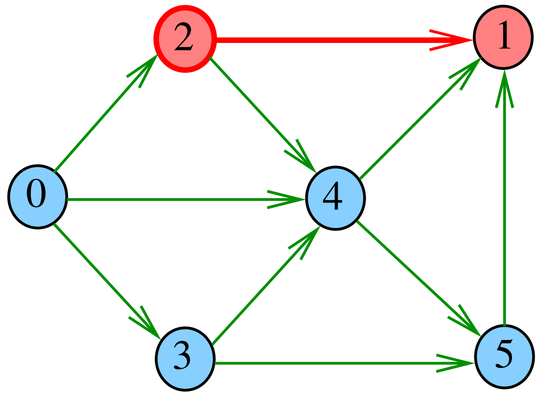
dfs(G,2)



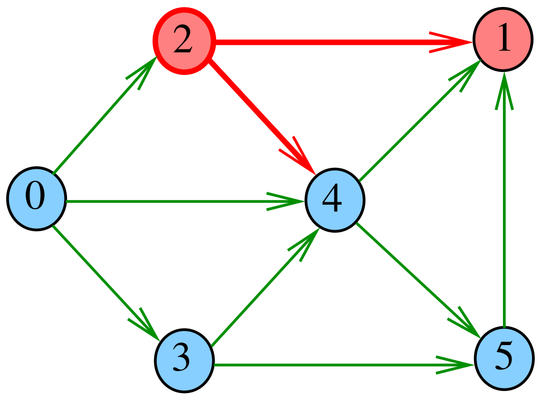
dfs(G,1)



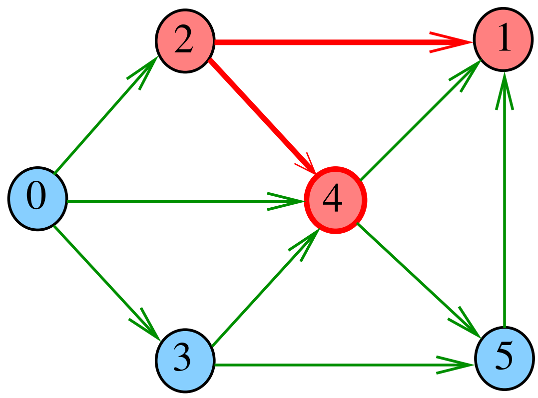
dfs(G,2)



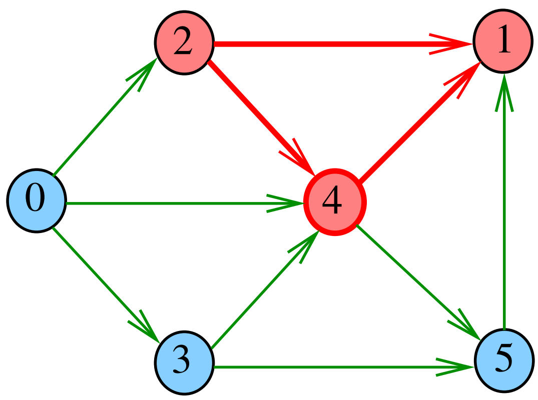
dfs(G,2)



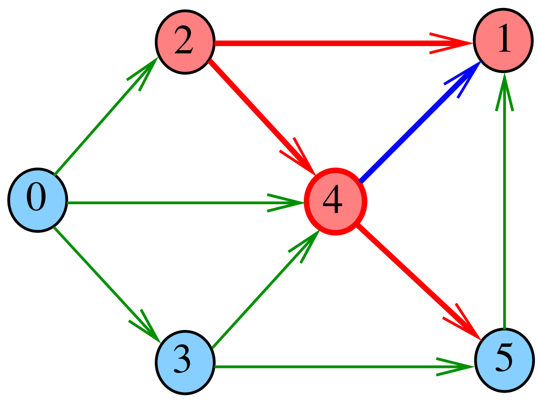
dfs(G,4)



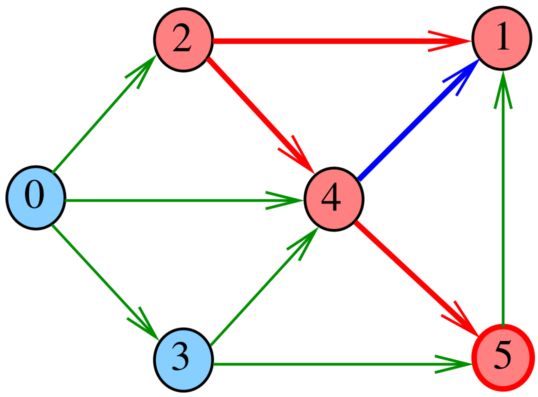
dfs(G,4)



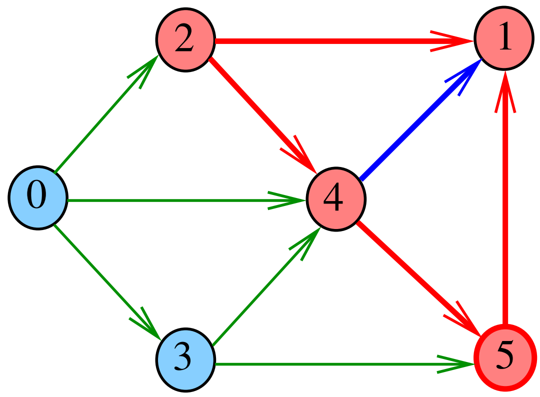
dfs(G,4)



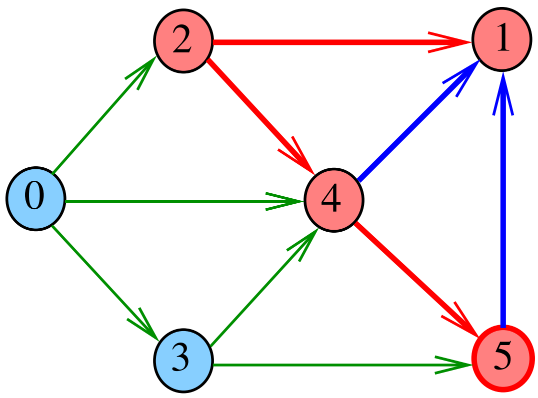
dfs(G,5)



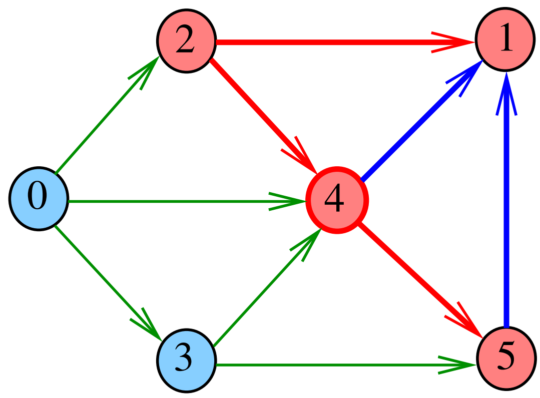
dfs(G,5)



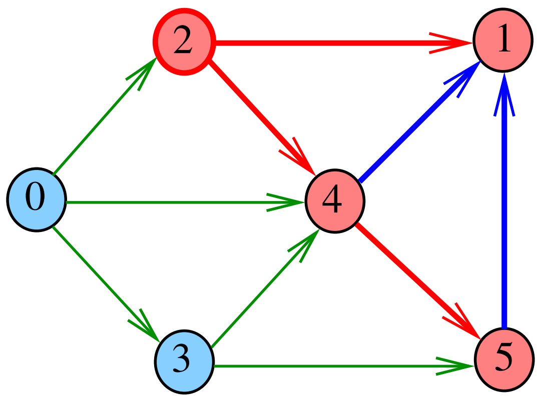
dfs(G,5)



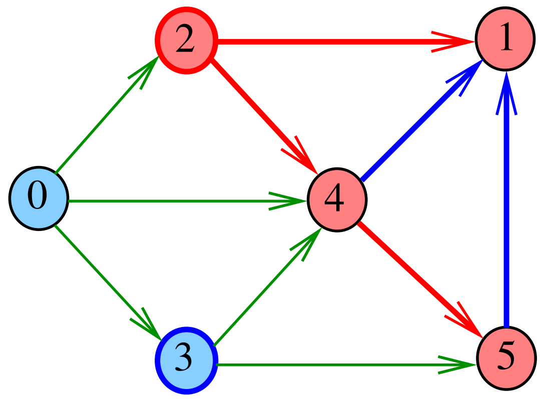
dfs(G,4)



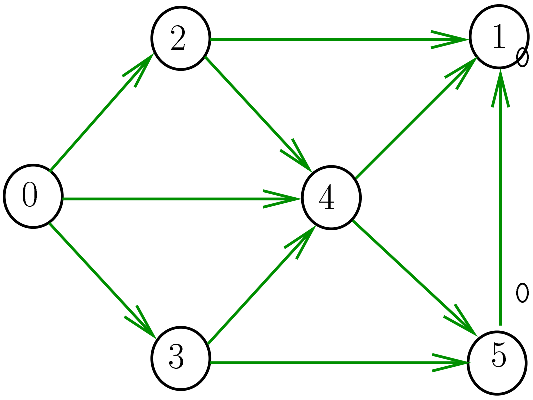
dfs(G,2)



DFSpaths(G,2,3)

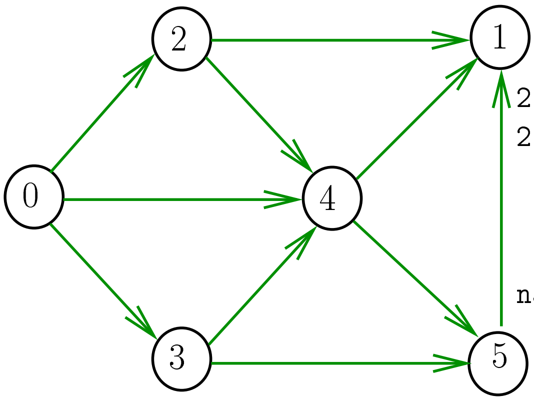


DFSpath(G,0,1)




```
0-2 dfs(G,2)
  2-1 dfs(G,1)
  2-4 dfs(G,4)
    4-1
    4-5 dfs(G,5)
      5-1
0-3 dfs(G,3)
  3-4
  3-5
0-4
existe caminho
```

DFSpath(G,2,3)




```
2-1 dfs(G,1)
2-4 dfs(G,4)
  4-1
  4-5 dfs(G,5)
    5-1
nao existe caminh
```

Consumo de tempo

Qual é o consumo de tempo da função DFSpaths?

Consumo de tempo

Qual é o consumo de tempo da função DFSpaths?

Qual é o consumo de tempo da função dfs()?

Conclusão

O consumo de tempo da função DIGRAPHpath é $\Theta(V)$ mais o consumo de tempo da função dfsR.

Conclusão

O consumo de tempo da função dfs() para vetor de listas de adjacência é $\sim V + E$.

O consumo de tempo de DFSPaths para vetor de listas de adjacência é $\sim V + E$.