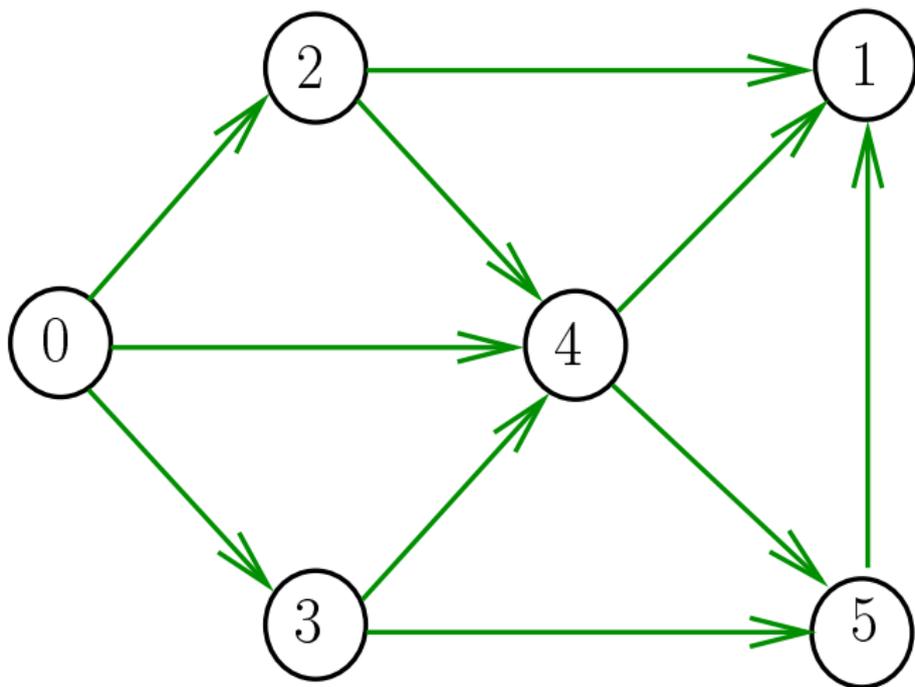


Melhores momentos

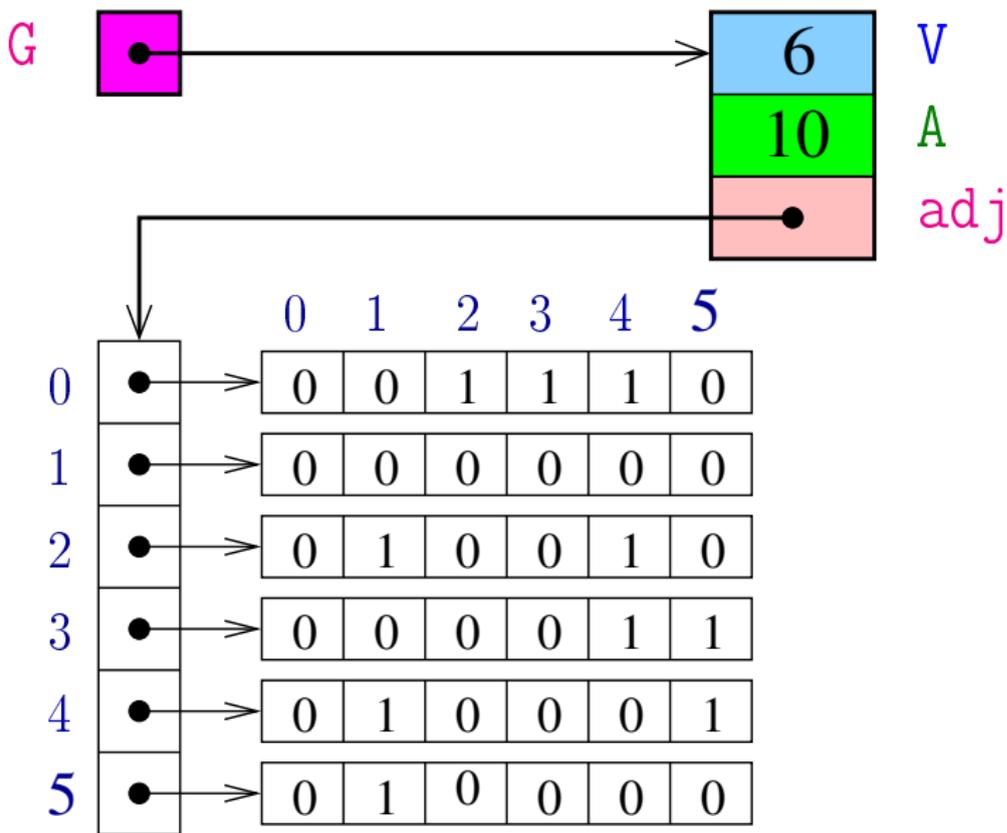
AULA 2

Digrafo

Digraph G



Estruturas de dados



Estrutura digraph

Vértices = inteiros em $0, \dots, V-1$

A estrutura **digraph** representa um digrafo

adj é um ponteiro para a matriz de adjacência

V contém o número de vértices

A contém o número de arcos do digrafo.

```
struct digraph {  
    int V;  
    int A;  
    int **adj;  
};  
typedef struct digraph *Digraph;
```

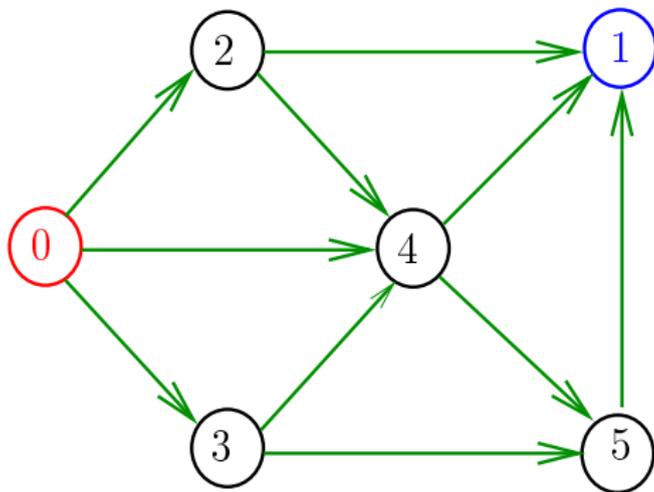
Funções básicas

```
Digraph DIGRAPHinit (int);  
void DIGRAPHinsertA (Digraph, Vertex, Vertex);  
void DIGRAPHremoveA (Digraph, Vertex, Vertex);  
void DIGRAPHshow (Digraph);
```

Procurando um caminho

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

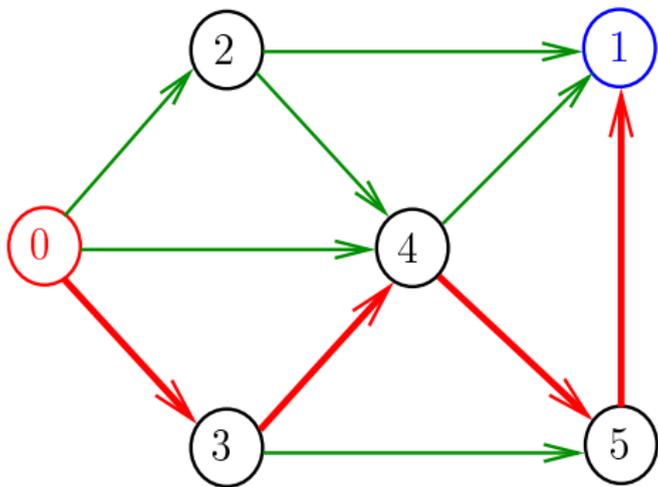
Exemplo: para $s = 0$ e $t = 1$ a resposta é SIM



Procurando um caminho

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

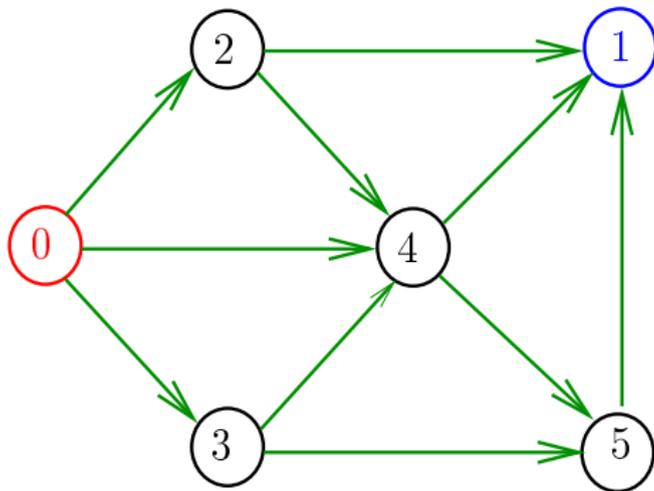
Exemplo: para $s = 0$ e $t = 1$ a resposta é **SIM**



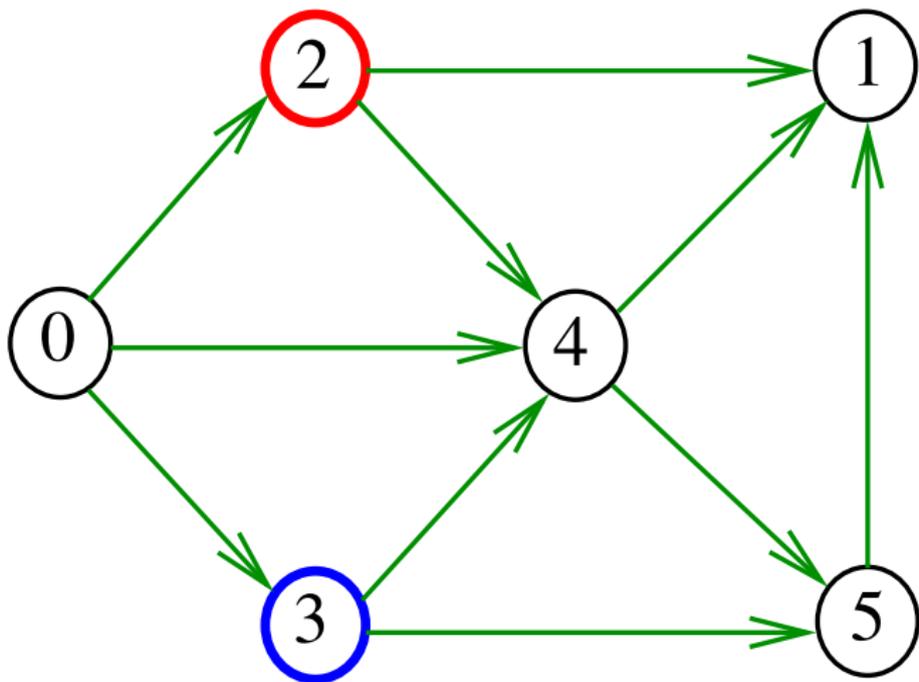
Procurando um caminho

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

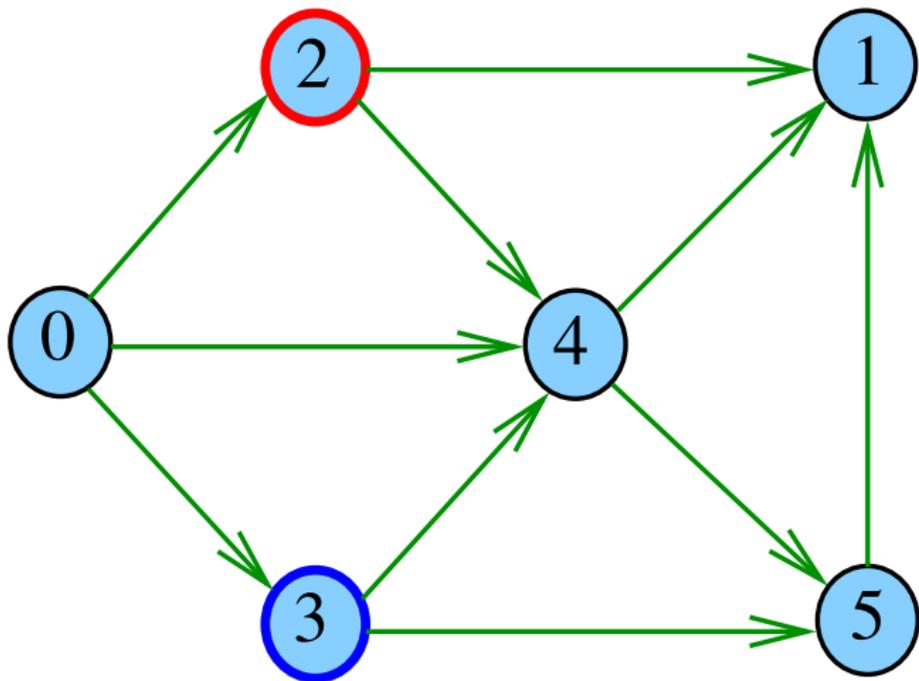
Exemplo: para $s = 5$ e $t = 4$ a resposta é **NÃO**



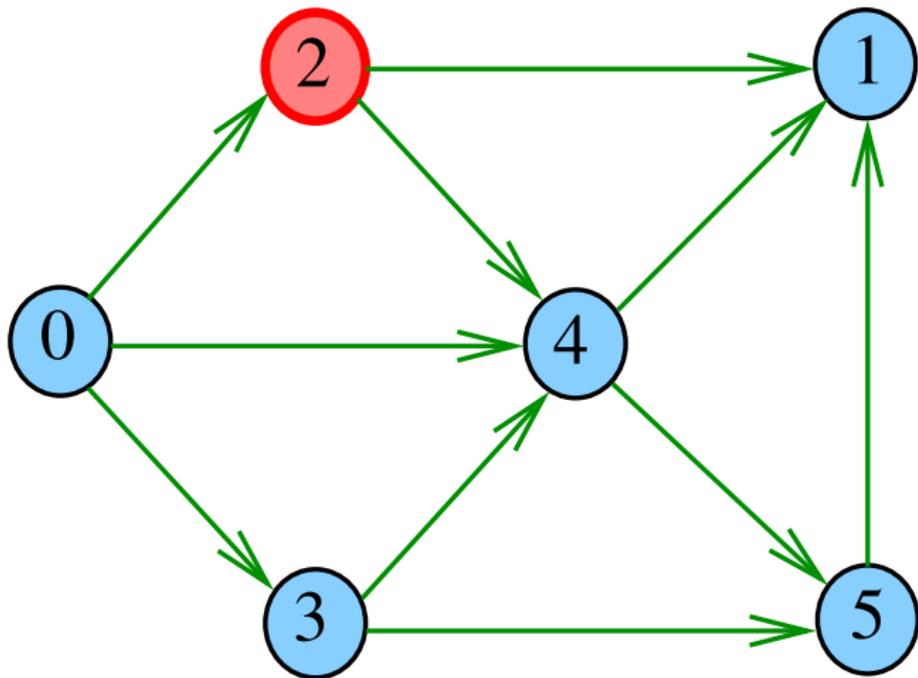
DIGRAPH $\text{path}(G, 2, 3)$



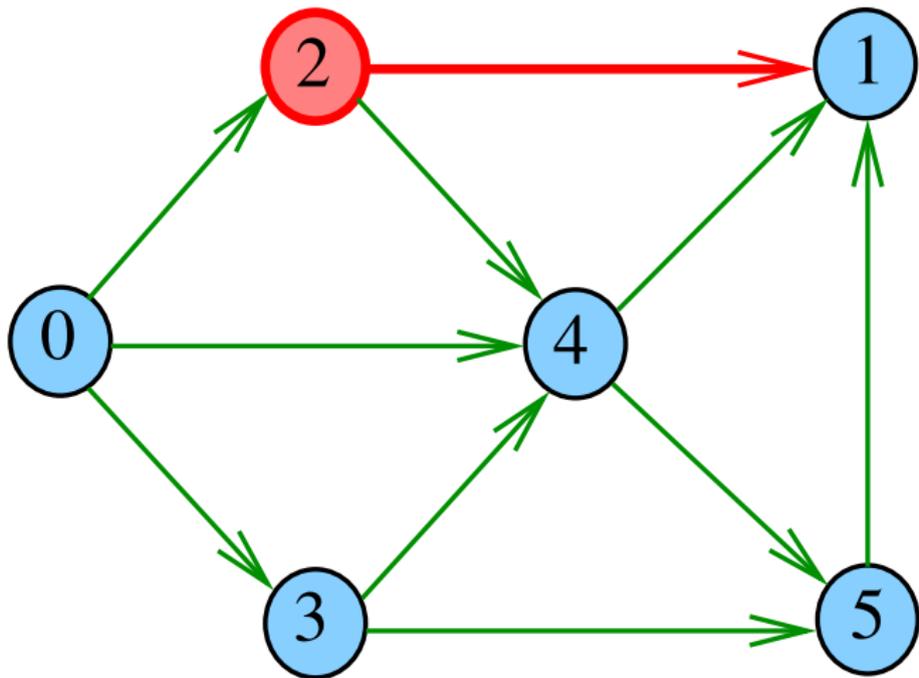
DIGRAPH $\text{path}(G, 2, 3)$



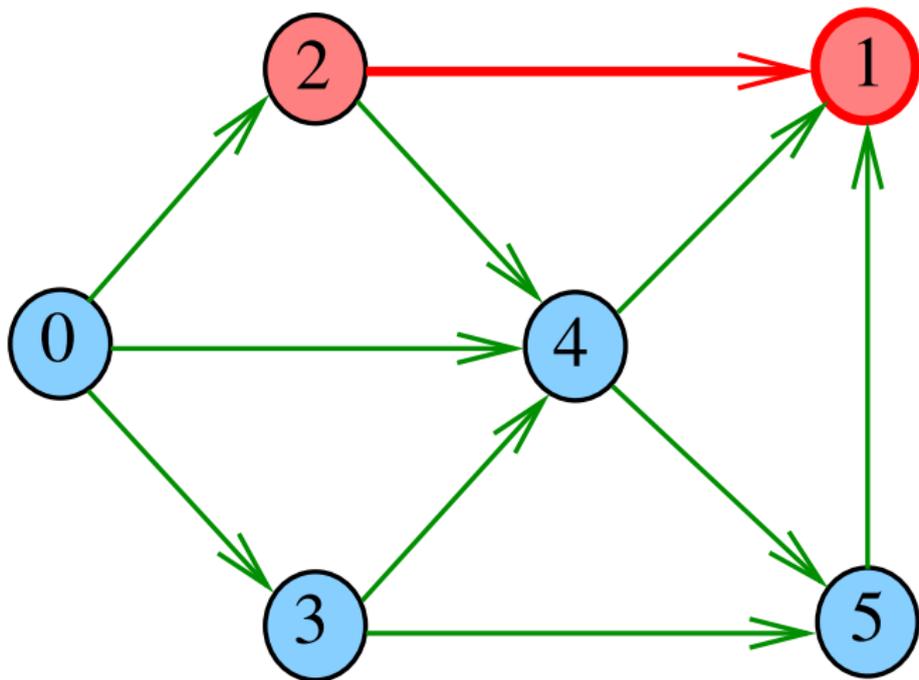
pathR($G, 2$)



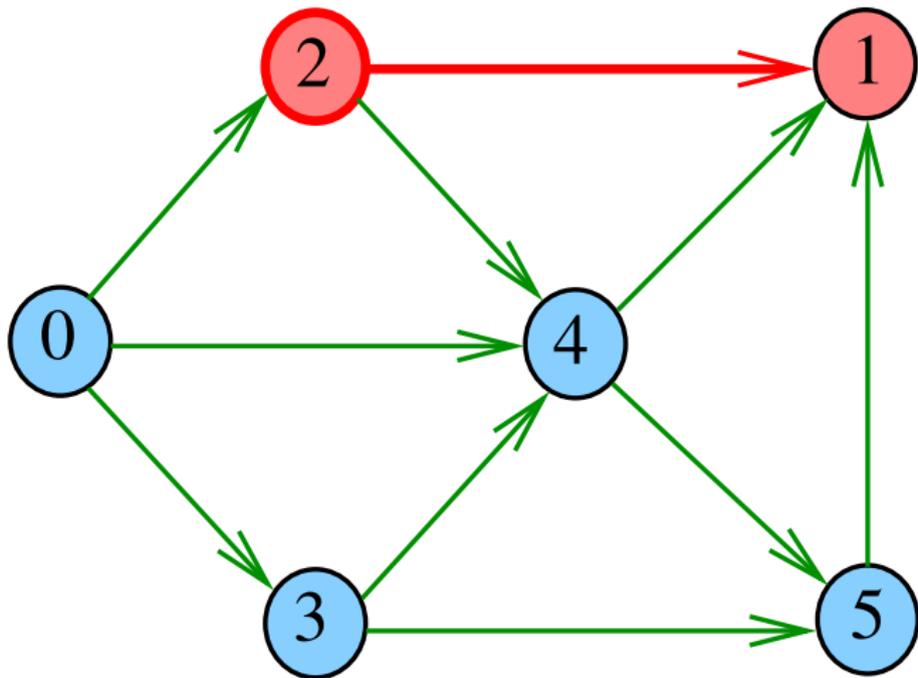
pathR(G,2)



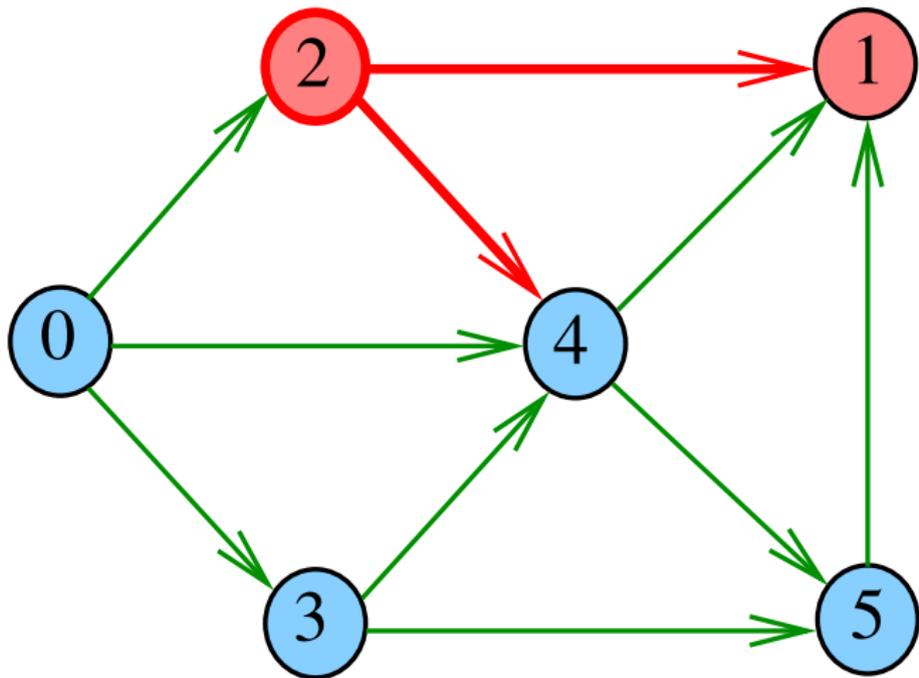
pathR(G,1)



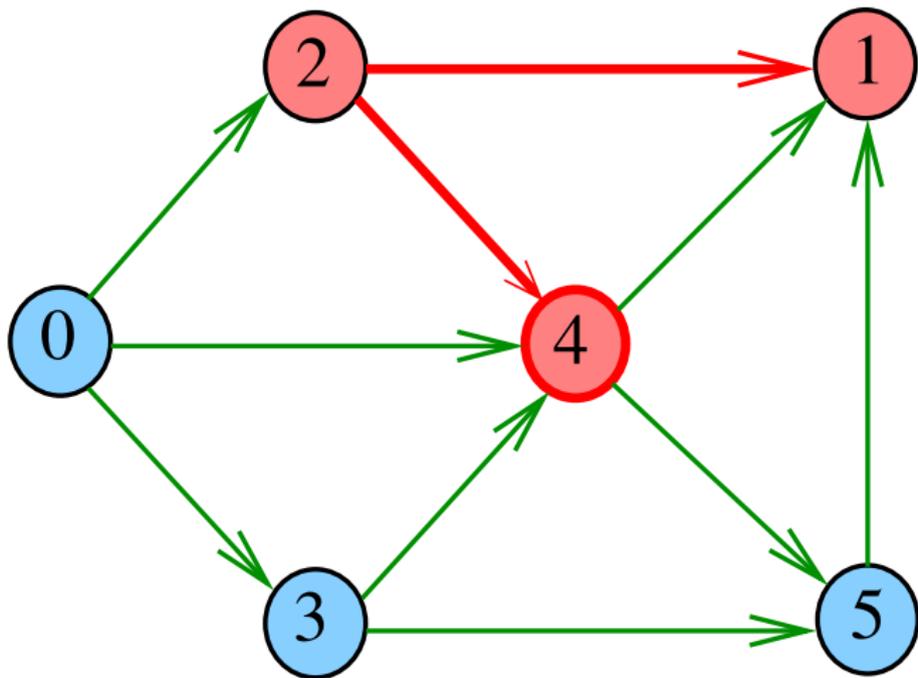
pathR(G,2)



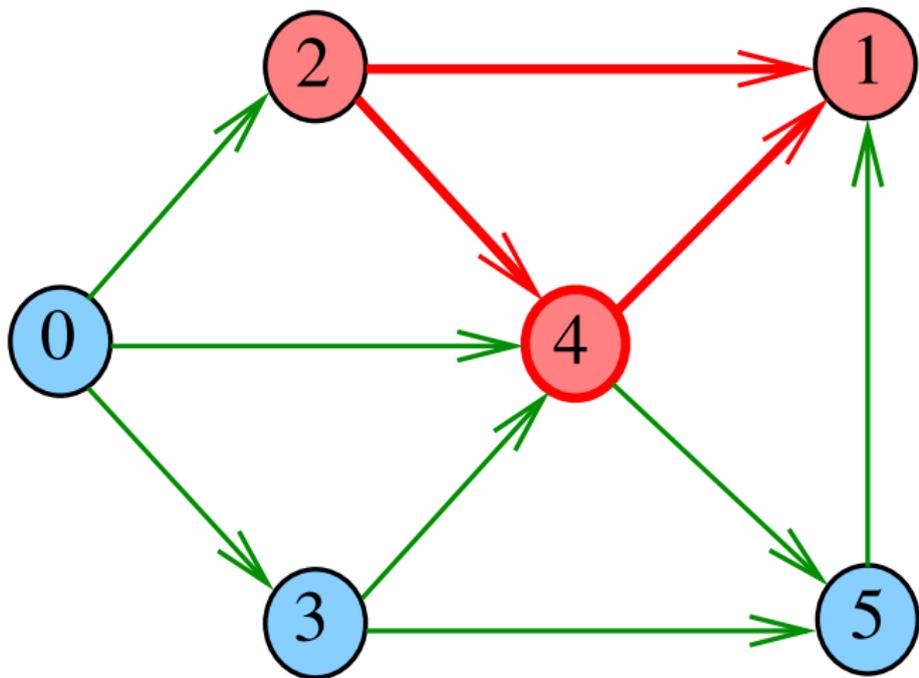
pathR(G,2)



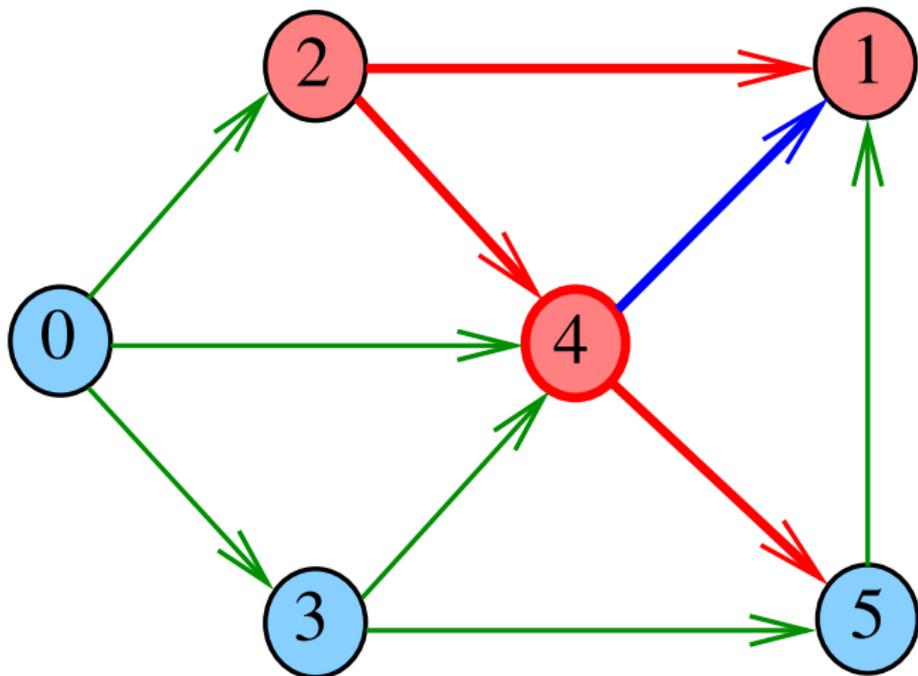
pathR(G,4)



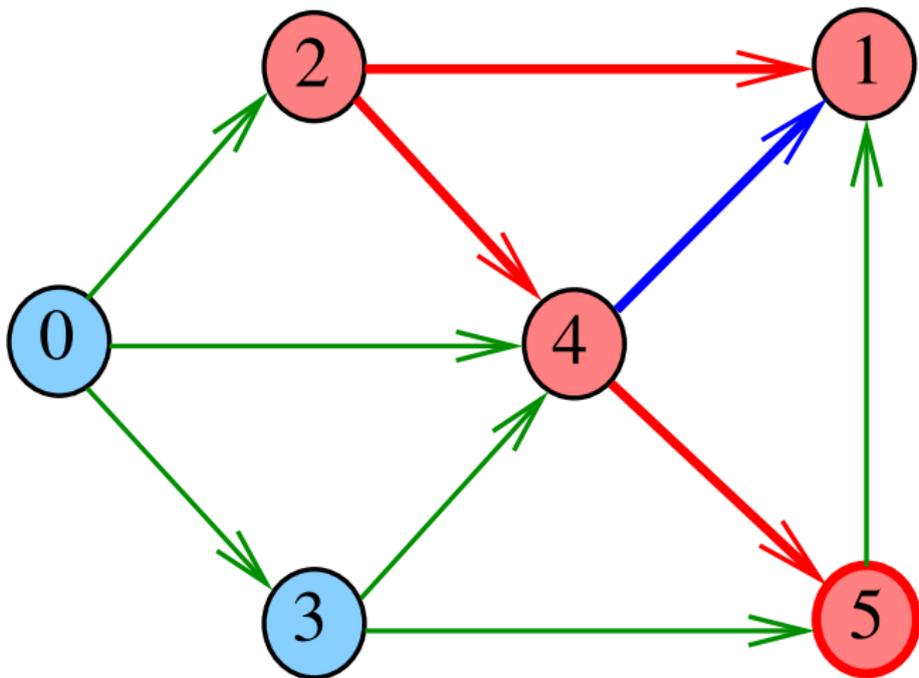
pathR(G,4)



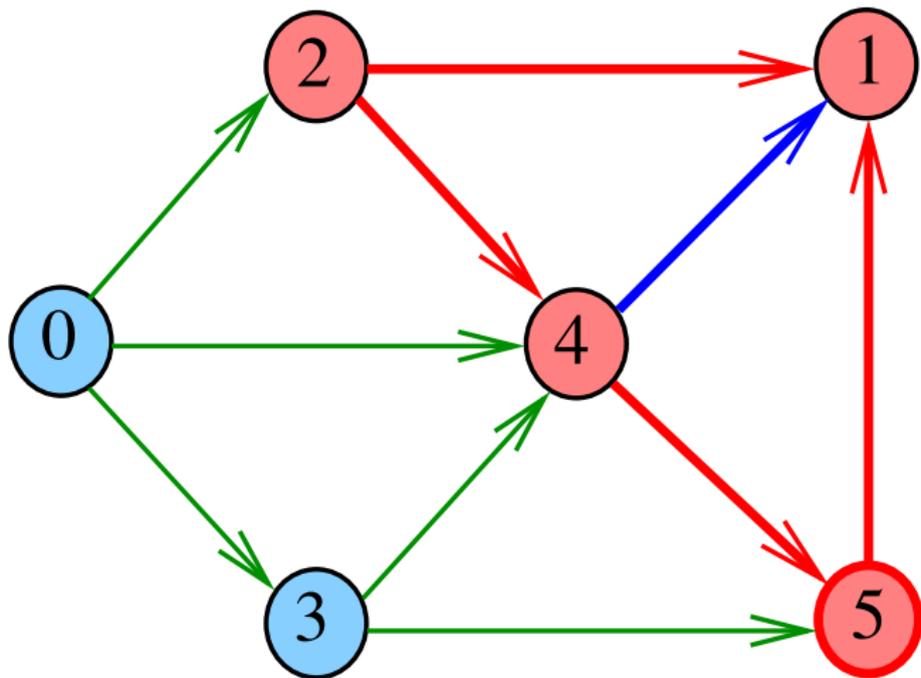
pathR(G,4)



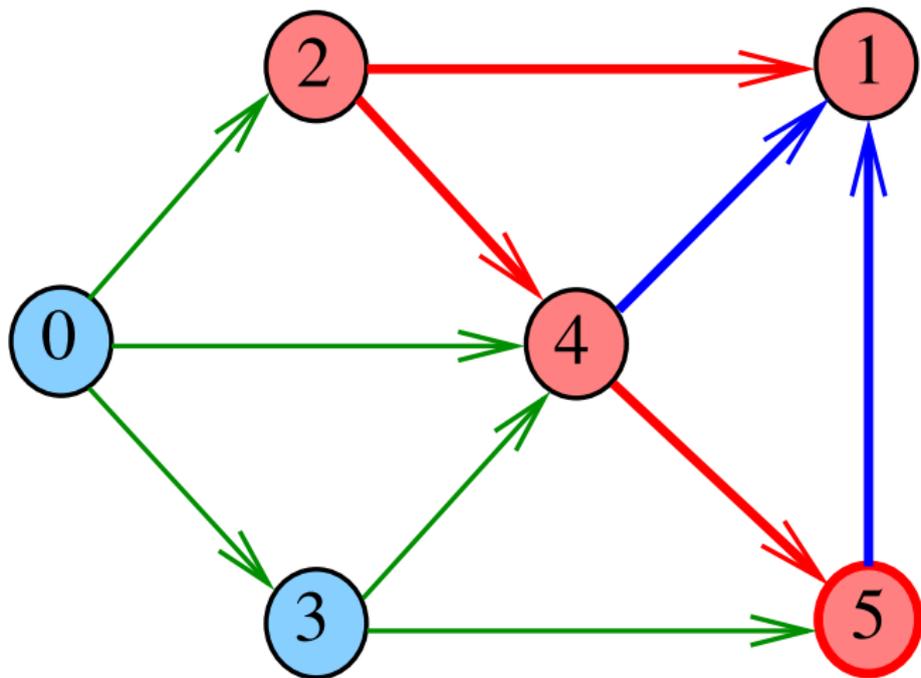
pathR(G,5)



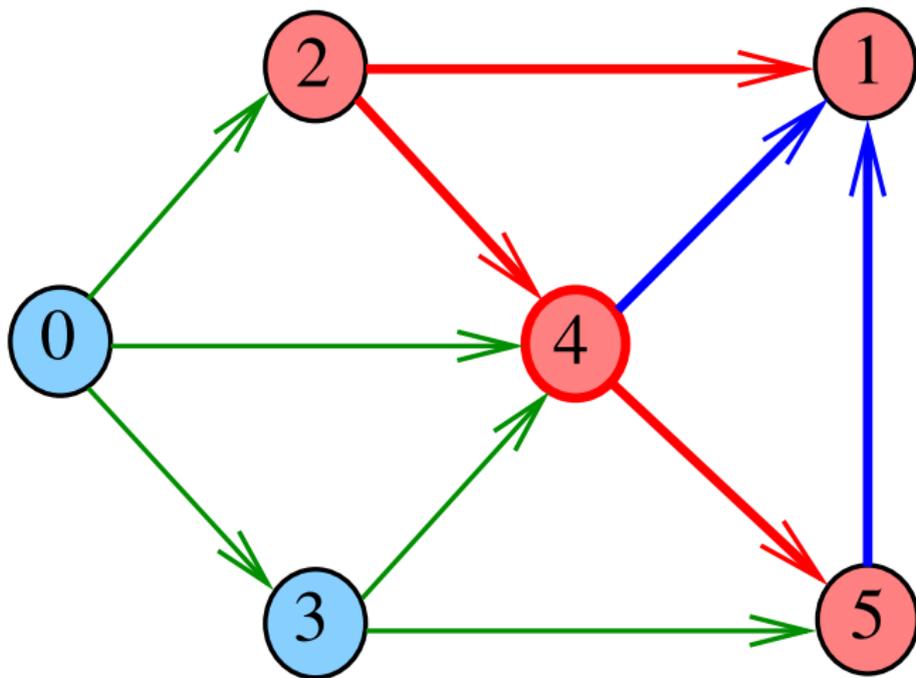
pathR(G,5)



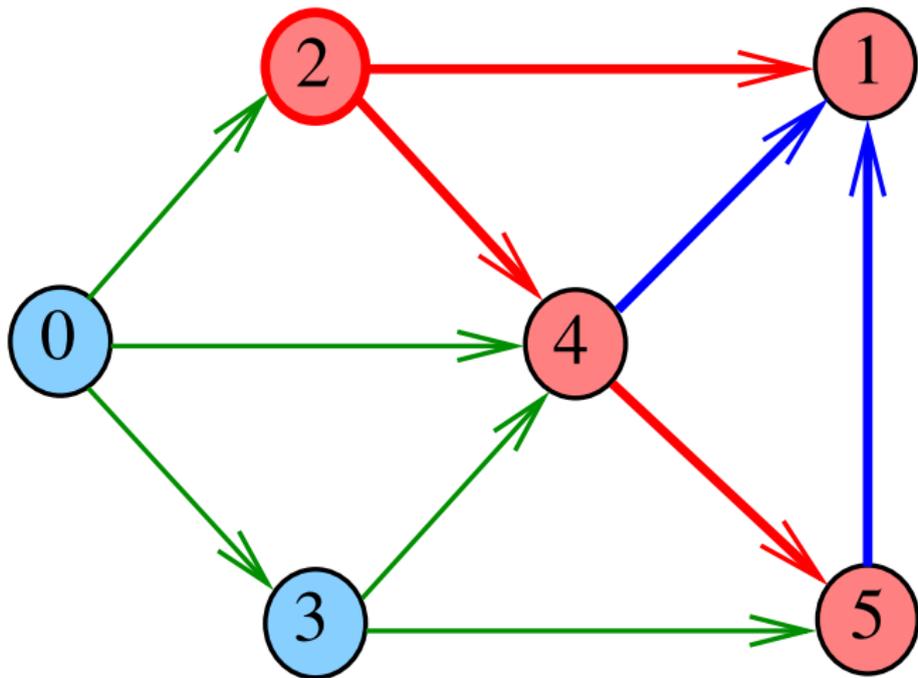
pathR(G,5)



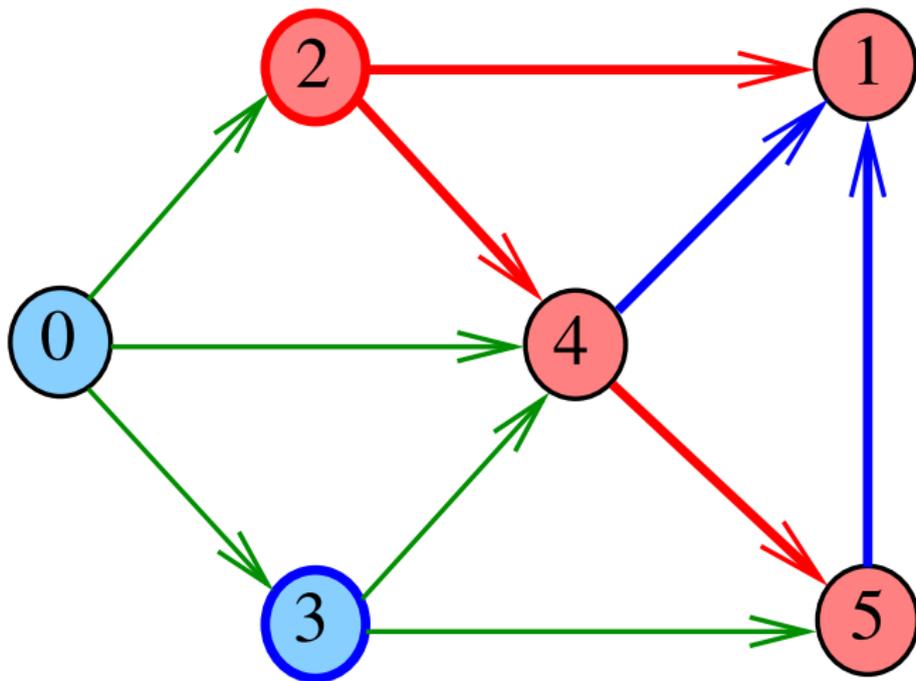
pathR(G,4)



pathR(G,2)



DIGRAPH $\text{path}(G, 2, 3)$



DIGRAPHpath

```
static int lbl[maxV];  
int DIGRAPHpath (Digraph G, Vertex s, Vertex t)  
{  
    Vertex v;  
1   for (v = 0; v < G->V; v++)  
2       lbl[v] = -1;  
3   pathR(G, s);  
4   if (lbl[t] == -1) return 0;  
5   else return 1;  
}
```

pathR

Visita todos os vértices que podem ser atingidos a partir de v

```
void pathR (Digraph G, Vertex v)
{
    Vertex w;
1   lbl[v] = 0;
2   for (w = 0; w < G->V; w++)
3       if (G->adj[v][w] == 1)
4           if (lbl[w] == -1)
5               pathR(G, w);
}
```

Consumo de tempo

O consumo de tempo da função `PathR` para matriz de adjacência é $O(V^2)$.

O consumo de tempo da função `DIGRAPHpath` para matriz de adjacência é $O(V^2)$.

AULA 3

Caminhos em digrafos (continuação)

S 17.1

DIGRAPHpath

Esta versão pára assim que encontra t

```
static int lbl[maxV];
```

```
int DIGRAPHpath (Digraph G, Vertex s, Vertex t)
```

DIGRAPHpath

Esta versão pára assim que encontra t

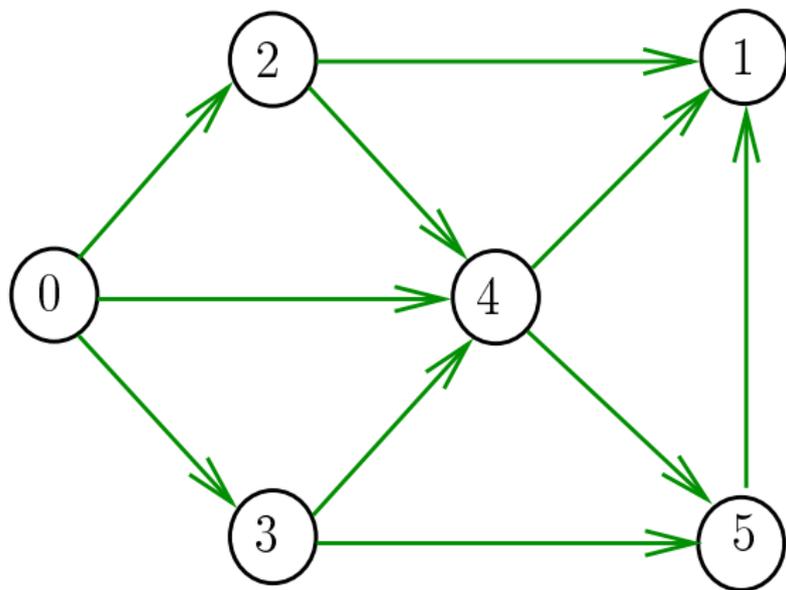
```
static int lbl[maxV];  
int DIGRAPHpath (Digraph G, Vertex s, Vertex t)  
{  
    Vertex v;  
1   for (v = 0; v < G->V; v++)  
2       lbl[v] = -1;  
3   return pathR(G, s, t);  
}
```

pathR

Pára assim que encontra t

```
int pathR (Digraph G, Vertex v, Vertex t) {  
    Vertex w;  
0   if (v == t) return 1;  
1   lbl[v] = 0;  
2   for (w = 0; w < G->V; w++)  
3       if (G->adj[v][w] == 1 && lbl[w] == -1)  
4           if (pathR(G, w) == 1)  
5               return 1;  
6   return 0;  
}
```

DIGRAPHpath(G,0,1)



0-2 pathR(G,2)

2-1 pathR(G,1)

2-4 pathR(G,4)

4-1

4-5 pathR(G,5)

5-1

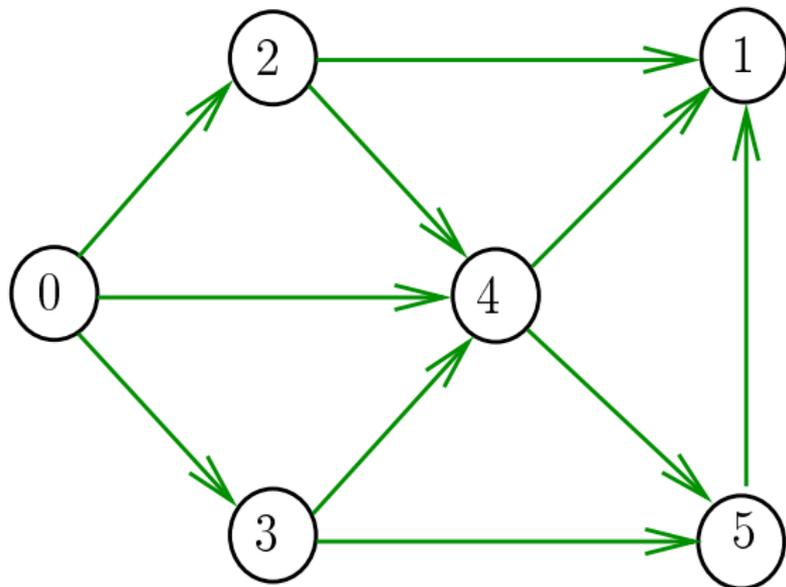
0-3 pathR(G,3)

3-4

0-4

existe caminho

DIGRAPHpath(G,2,3)



2-1 pathR(G,1)

2-4 pathR(G,4)

4-1

4-5 pathR(G,5)

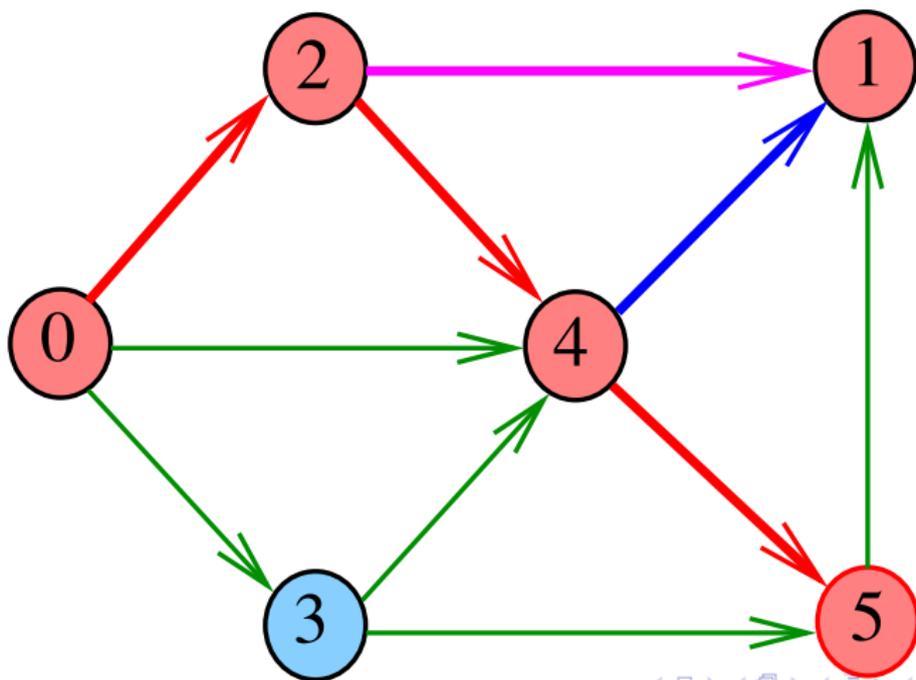
5-1

nao existe caminho

DIGRAPHpath (versão iterativa)

DIGRAPHpath (versão iterativa)

Relação invariante chave: no início de cada iteração
caminho[0] - caminho[1] - ... - caminho[k-1]
é um caminho de **s** a **v**.



DIGRAPHpath (versão iterativa)

```
static int lbl[maxV];  
int DIGRAPHpath (Digraph G, Vertex s, Vertex t)  
{  
    Vertex v, w;  
    Vertex caminho[maxV];  
    int k;  
1   for (v = 0; v < G->V; v++)  
2       lbl[v] = -1;  
3   lbl[s] = 0;  
4   caminho[0] = s;  
5   k = 1;    v = s;    w = 0;
```

DIGRAPHpath (versão iterativa)

```
6  while (k != 1 || w != G->V)
7      if (w == G->V) { /* volta */
8          w = v+1;    k--;
9          v = caminho[k-1];
10     } else if (G->adj[v][w] == 1
11                && lbl[w] == -1) {
12         /* avança */
13         lbl[w] = 0;   caminho[k++] = w;
14         v = w;      w = 0;
15     } else w = w+1; /* tenta próximo */

16  if (lbl[t] == -1) return 0;
17  return 1;
```

Certificados

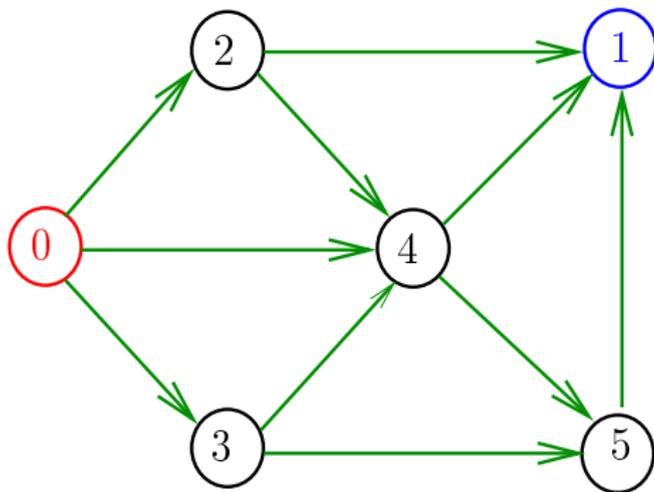
Cortes e arborescências

S páginas 84,91,92, 373

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



Certificados

Como é possível 'verificar' a resposta?

Como é possível 'verificar' que **existe** caminho?

Como é possível 'verificar' que **não existe** caminho?

Certificados

Como é possível 'verificar' a resposta?

Como é possível 'verificar' que **existe** caminho?

Como é possível 'verificar' que **não existe** caminho?

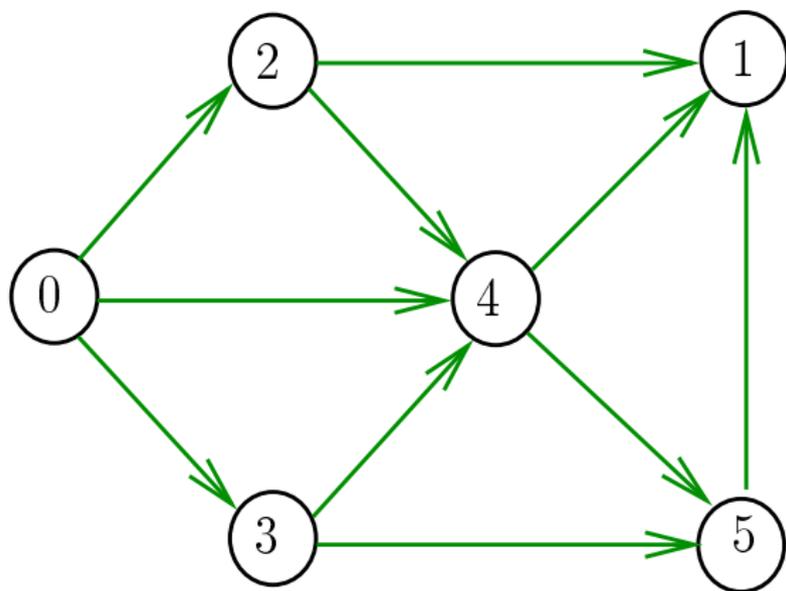
Veremos questões deste tipo freqüentemente

Elas terão um papel **suuupeer** importante no final de **MAC0338 Análise de Algoritmos**

Elas estão relacionadas com o **Teorema da Dualidade** visto em **MAC0315 Programação Linear**

Certificado de inexistência

Como é possível demonstrar que o problema não tem solução?



pathR(G, 2)

2-1 pathR(G, 1)

2-4 pathR(G, 4)

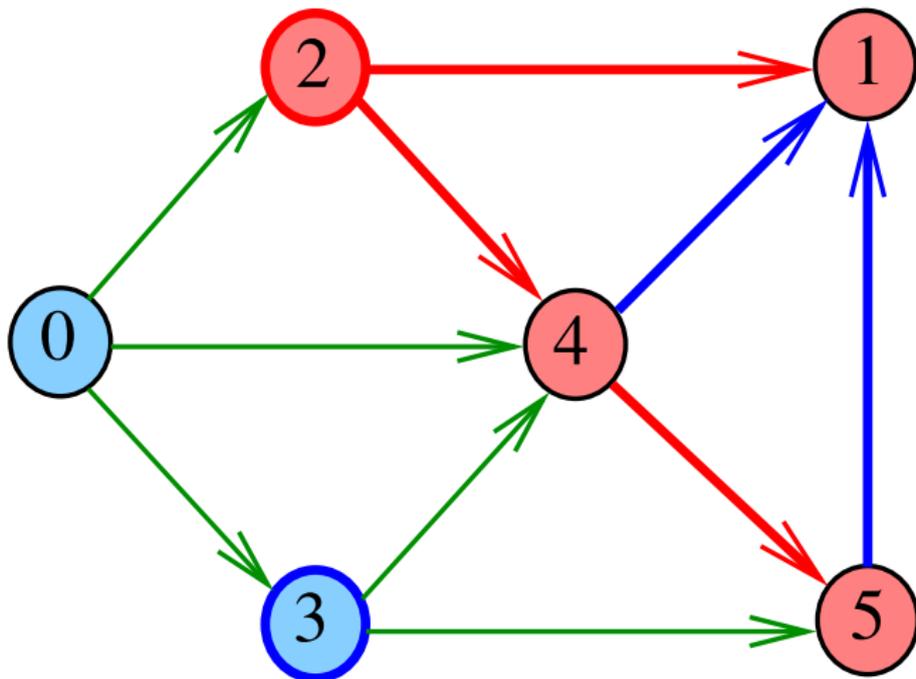
4-1

4-5 pathR(G, 5)

5-1

nao existe caminho

DIGRAPHpath($G, 2, 3$)

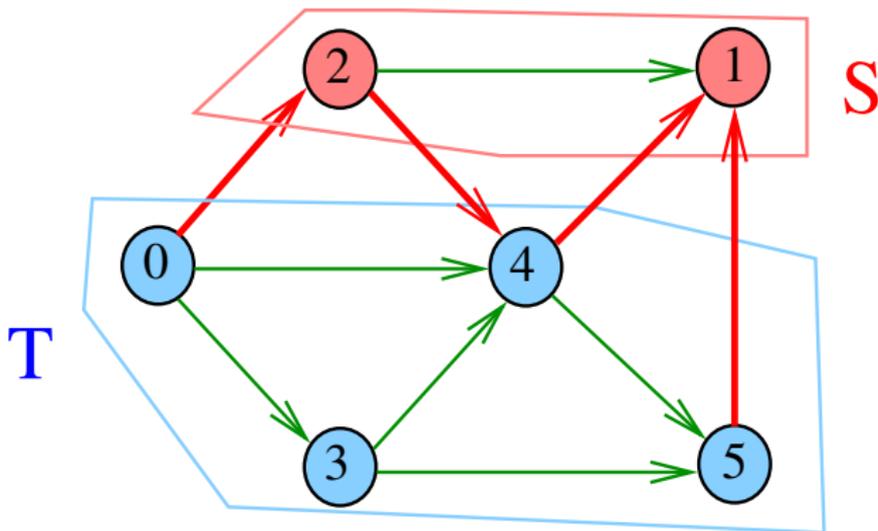


Cortes (= cuts)

Um **corte** é uma bipartição do conjunto de vértices

Um arco **pertence** ou **atravessa** um corte (S, T) se tiver uma ponta em S e outra em T

Exemplo 1: arcos em **vermelho** estão no corte (S, T)

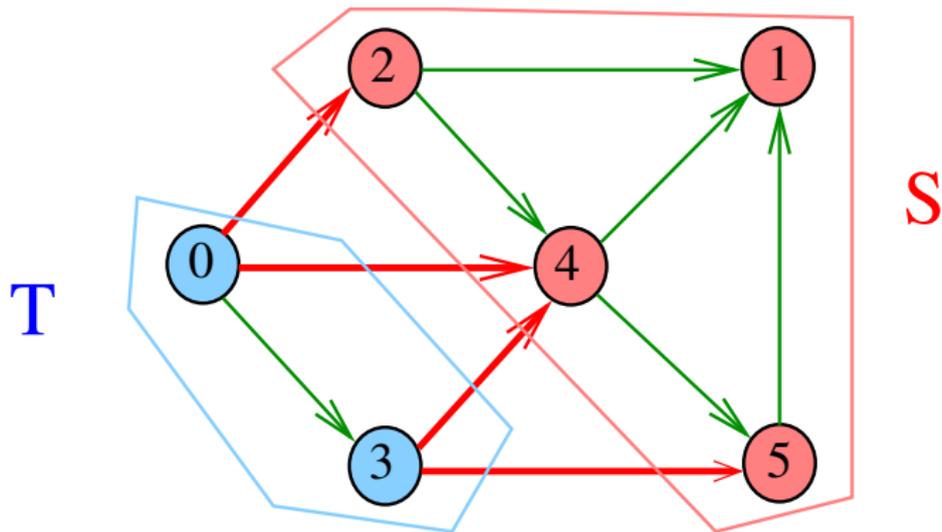


Cortes (= cuts)

Um **corte** é uma bipartição do conjunto de vértices

Um arco **pertence** ou **atravessa** um corte (S, T) se tiver uma ponta em S e outra em T

Exemplo 2: arcos em **vermelho** estão no corte (S, T)

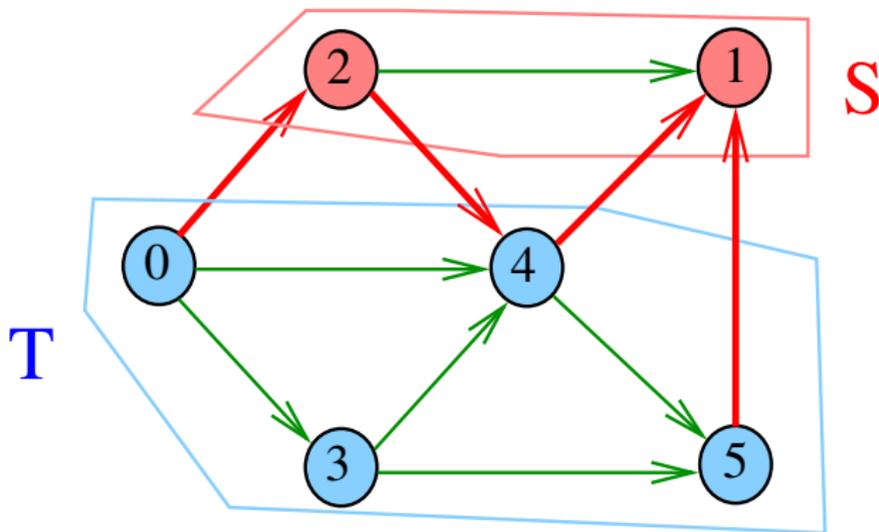


st-Cortes (= st-cuts)

Um corte (S, T) é um **st-corte** se

s está em S e t está em T

Exemplo: (S, T) é um 1-3-corte um 2-5-corte ...



Certificado de inexistência

Para demonstrarmos que **não existe** um caminho de **s** a **t** basta exibirmos um **st**-corte (S, T) em que ***todo arco** no corte tem ponta inicial em T e ponta final em S*

Certificado de inexistência

Exemplo: certificado de que não há caminho de 2 a 3

