# Melhores momentos

AULA PASSADA

# Ford e Fulkerson

Método dos caminhos de incremento

> PASSO DE INCREMENTO: encontre um caminho de incremento para o fluxo corrente. Incremente o valor do fluxo “enviando a maior quantidade possível de fluxo através do caminho”.

Note que o método não especifica como encontrar o pseudo-caminho de incremento.

# Edmonds e Karp

Método dos incrementos através de caminhos “curtos”

PASSO DE INCREMENTO: encontre na rede residual (do fluxo corrente) um caminho de

comprimento mínimo.

Incremente o valor do fluxo “enviando a maior quantidade possível de fluxo possível através desse caminho”.

# Resumo

Invariante: no início de cada iteração temos um $st$-pseudofluxo $\check{x}$ que respeita as capacidades.

| Algoritmo | número de passos de incremento | consumo de tempo |
|---|---|---|
| FORD-FULKERSON | $\mathrm{O}(nU)$ | $\mathrm{O}(nmU)$ |
| MAX-CAPACITY | $\leq 2m(1 + \lfloor \lg U \rfloor)$ <br> $\mathrm{O}(m \lg U)$ | $\mathrm{O}(n^2 m \lg U)$ |
| CAPACITY-SCALING | $\leq 2m(1 + \lfloor \lg U \rfloor)$ <br> $\mathrm{O}(m \lg U)$ | $\mathrm{O}(m^2 \lg U)$ |
| EDMONDS-KARP | $nm$ | $\mathrm{O}(nm^2)$ |

Estamos supondo que $n = \mathrm{O}(m)$ (grafo conexo)

# AULA 14

# Dinits

PF 16.1, 16.2, 16.3, 16.4

# Dinits

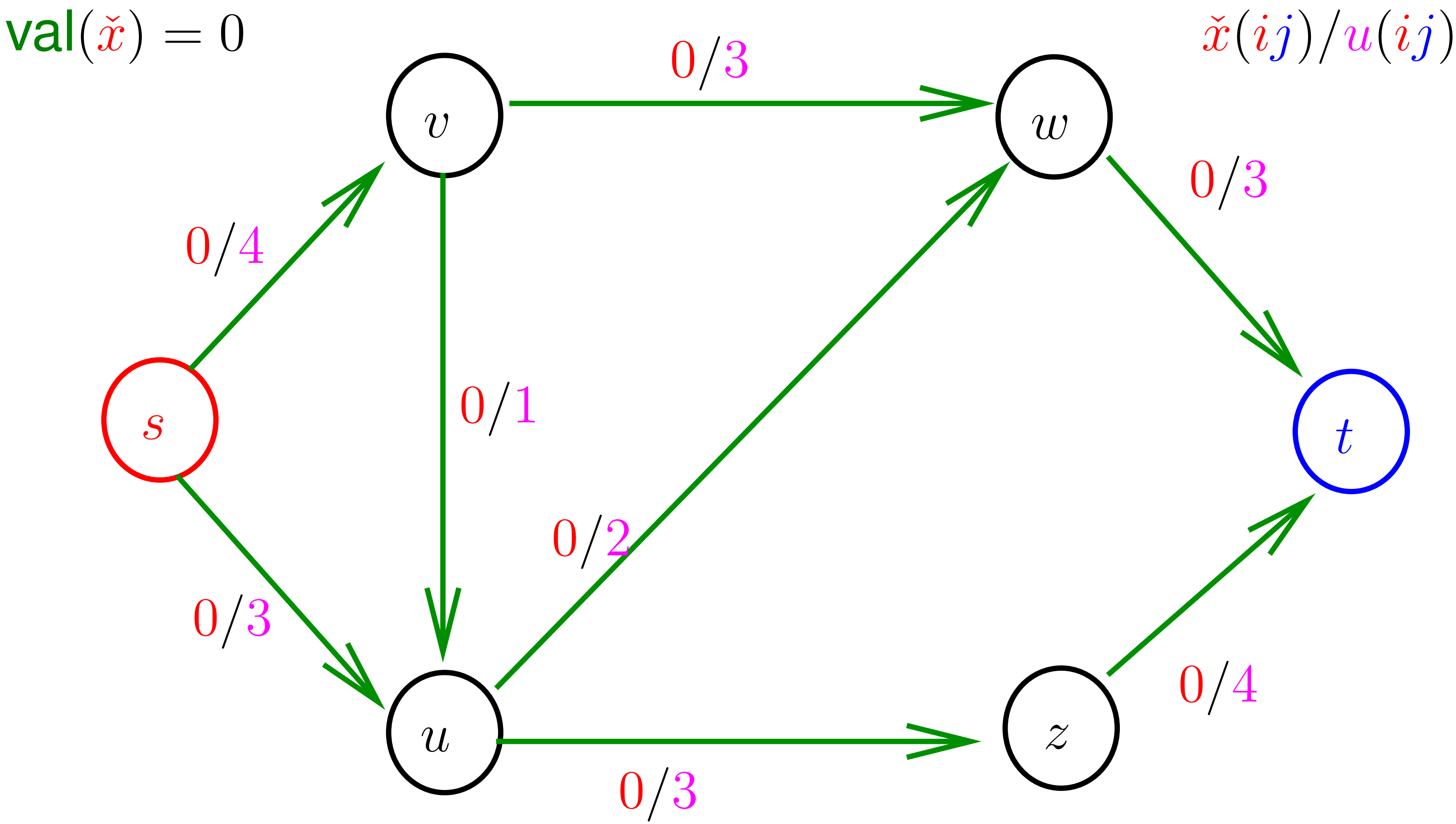
EDMONDS-KARP envia fluxos através de caminhos curtos, mas calcula um 1-potencial $(s, *)$-ótimo em cada iteração.

EDMONDS-KARP descarta os resultados da busca em largura.

Idéia: reutilizar o resultado de cada busca em largura o máximo possível.

Em cada iteração o algoritmo um “rede em camadas” ($=$ *layered network*) formadas pelos arcos que estão em algum caminho mínimo com origem em $s$ (ou com destino $t$, dependo do gosto do freguês).

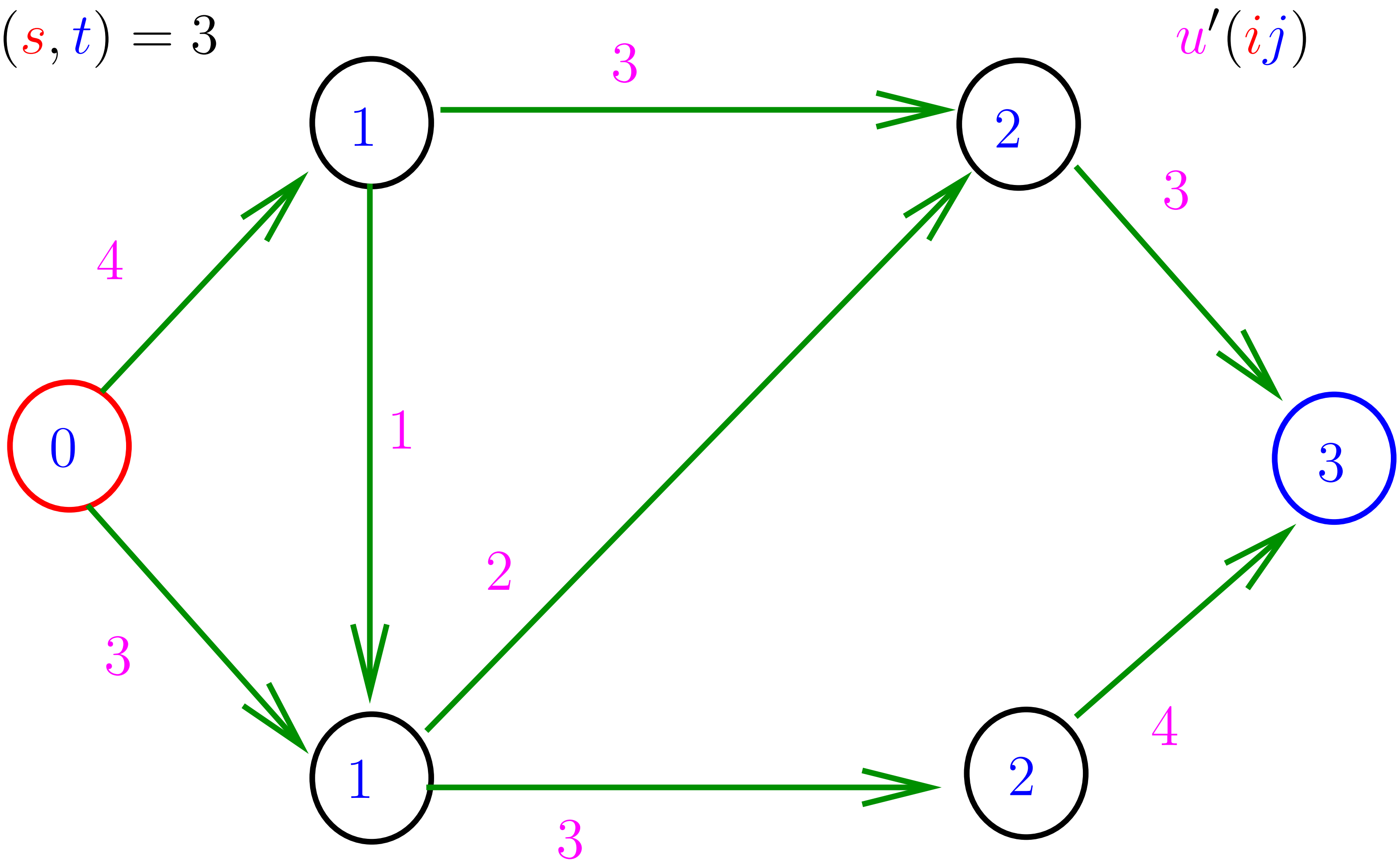
# Rede 1



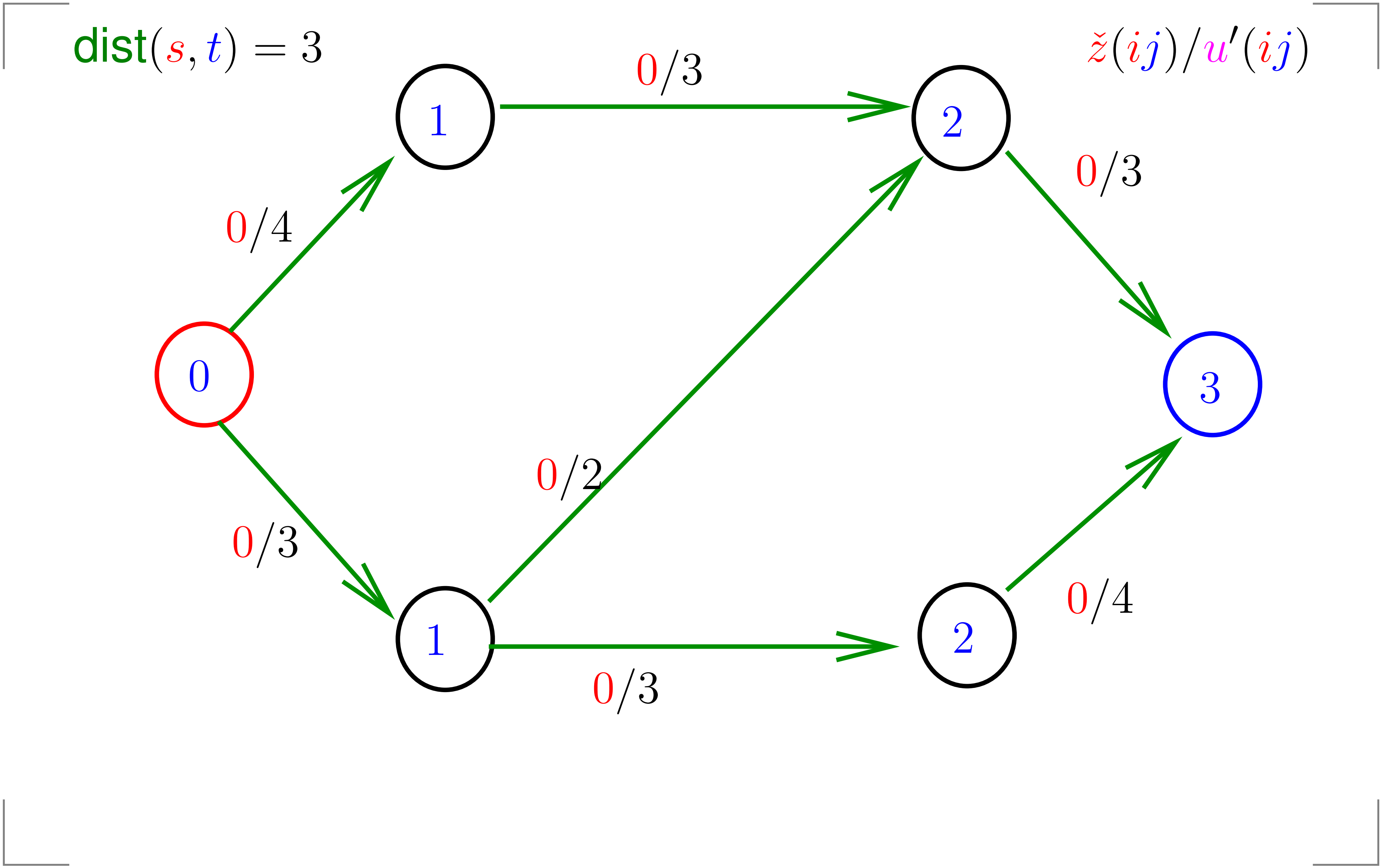

Arcos ausentes têm capacidade nula.

# Rede residual 1

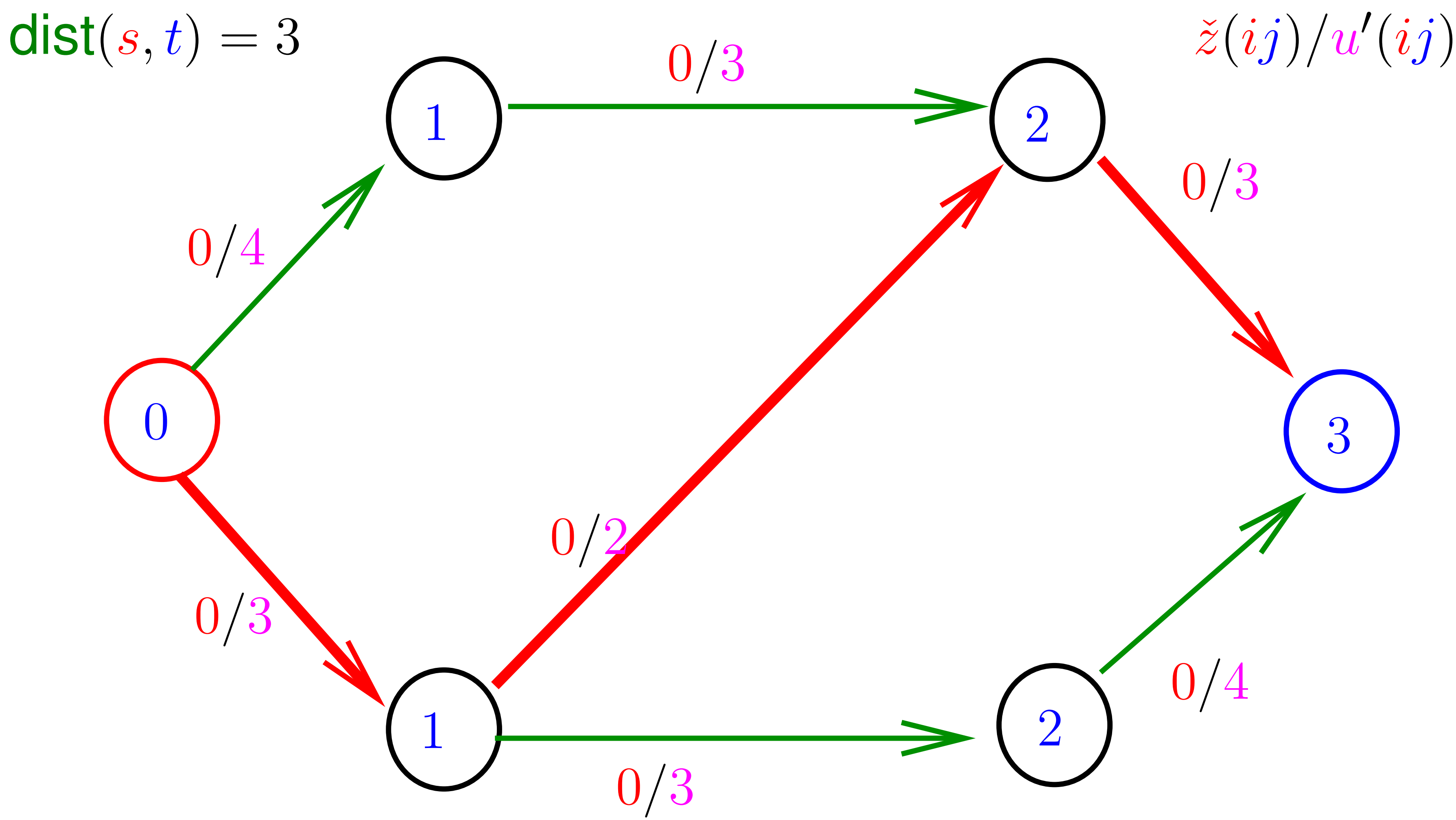
$\text{dist}(s, t) = 3$



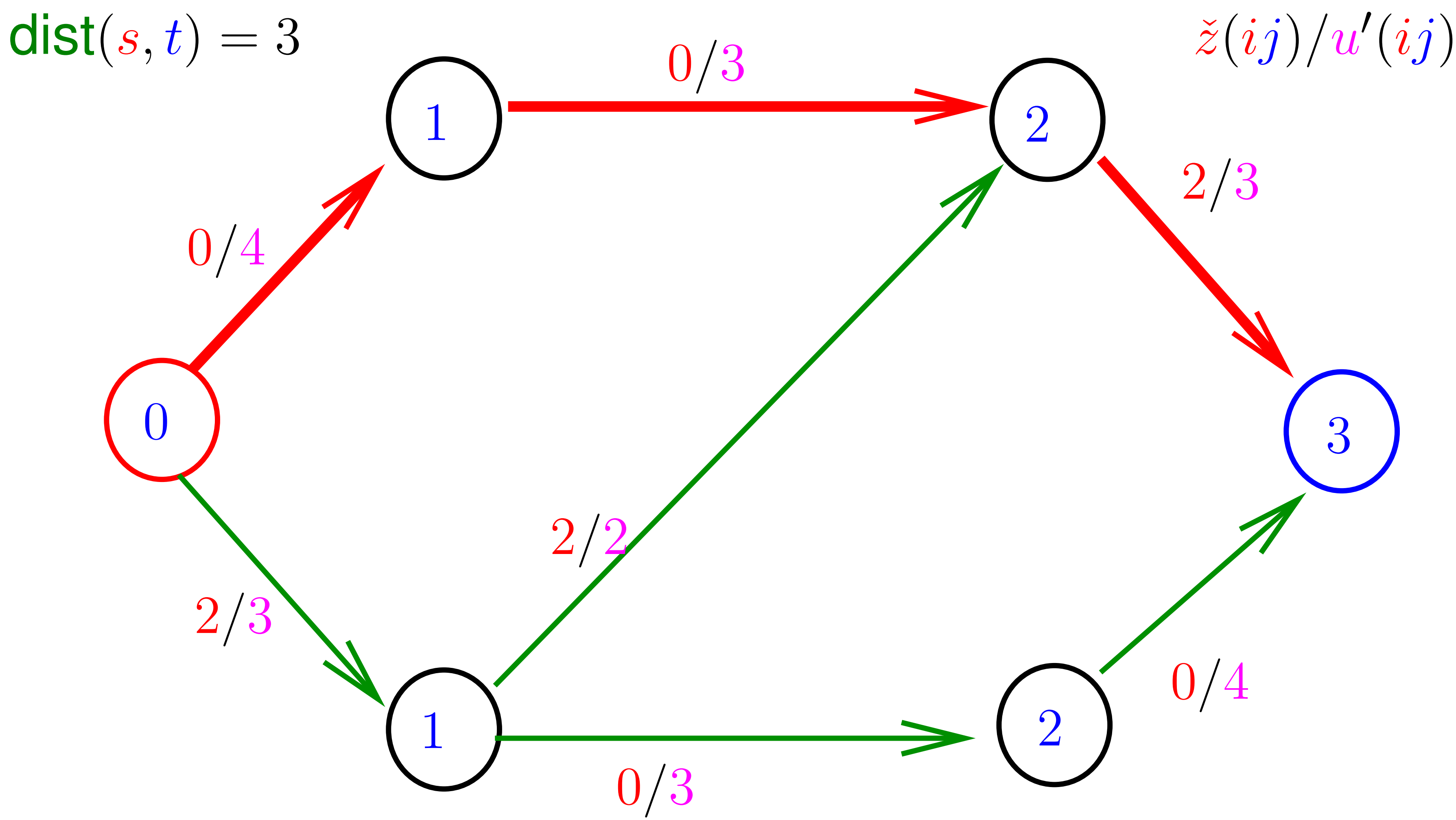
# Rede em camadas 1



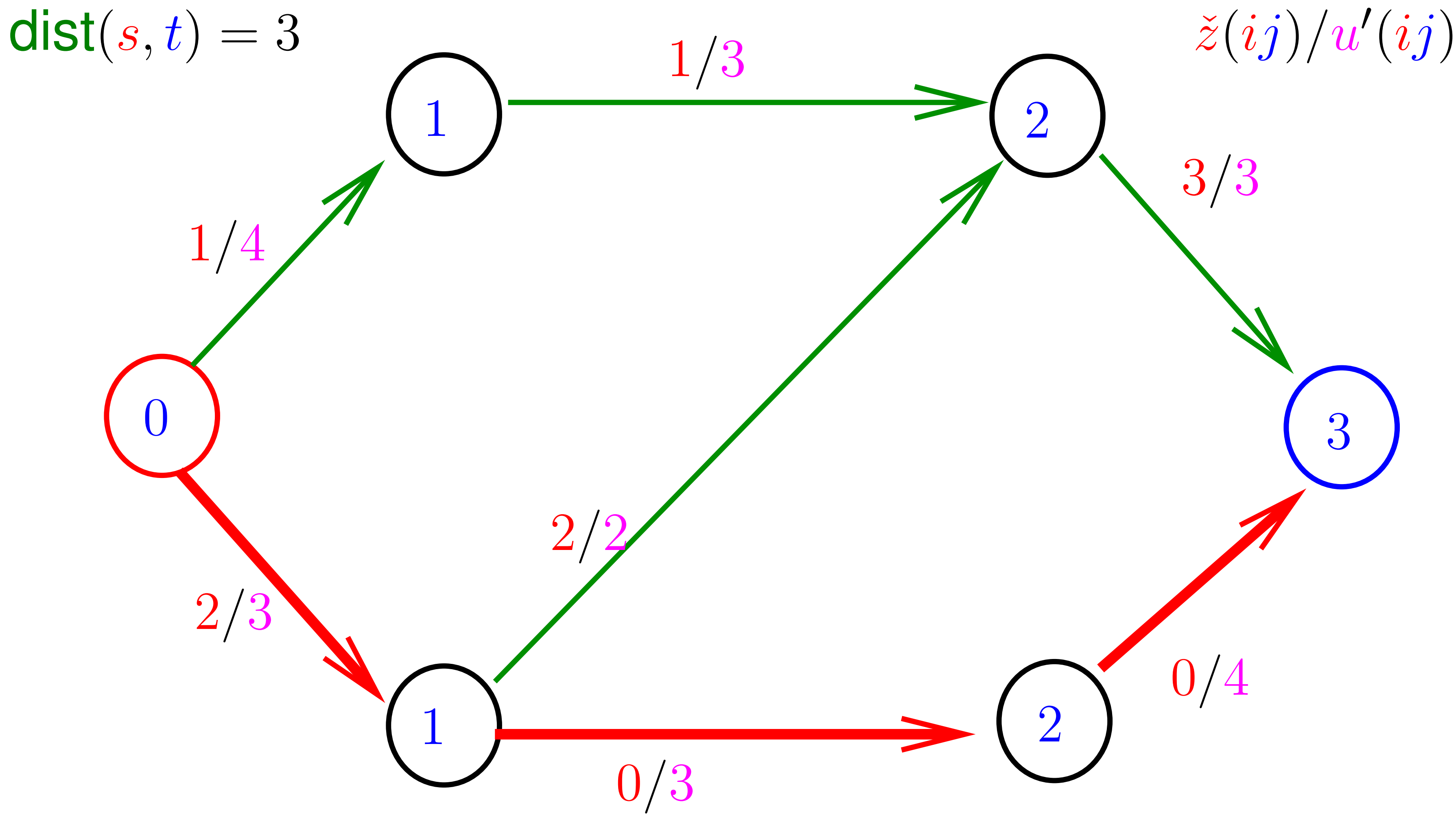
# Caminho de incremento 1



# Caminho de incremento 2



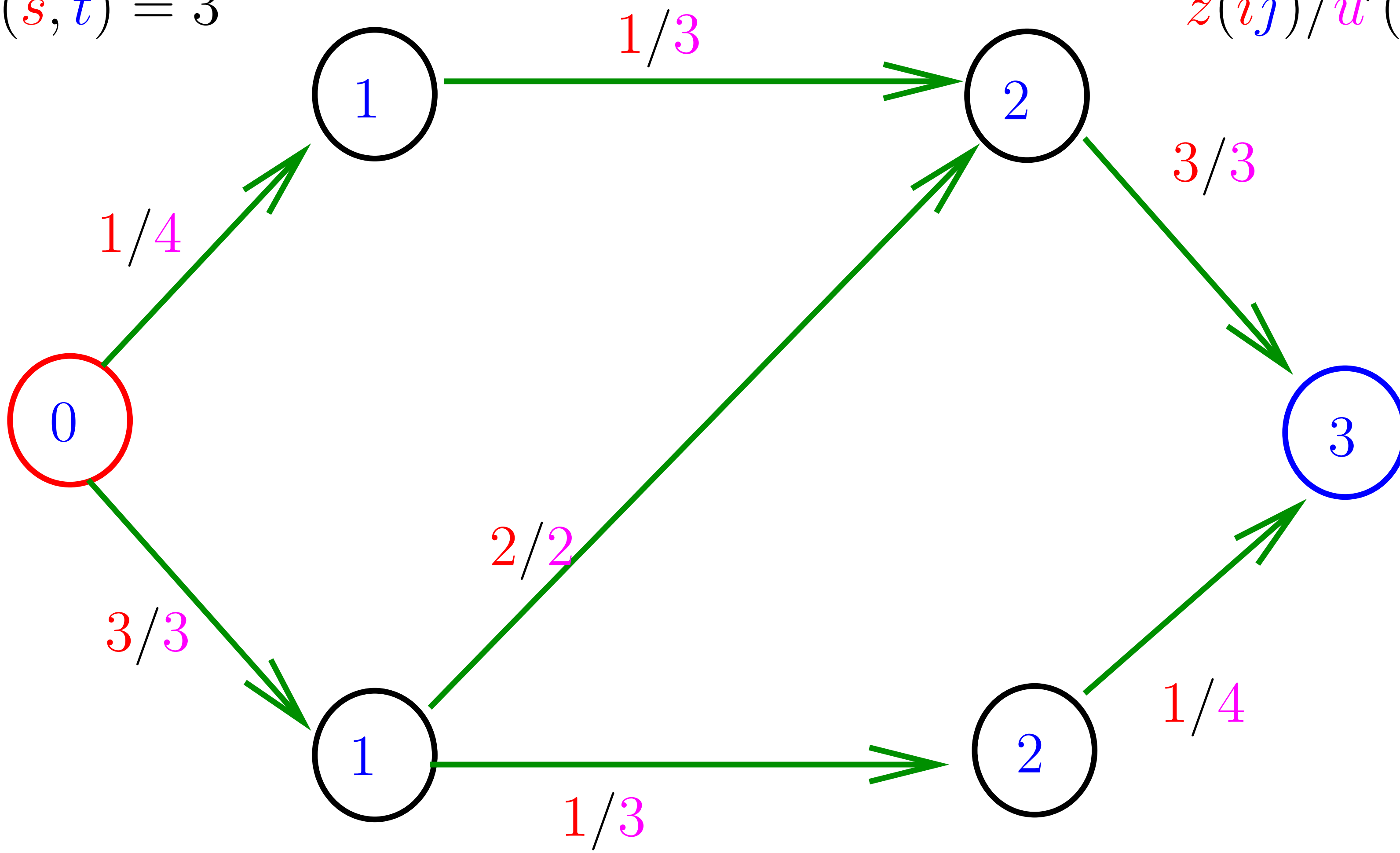
# Caminho de incremento 3



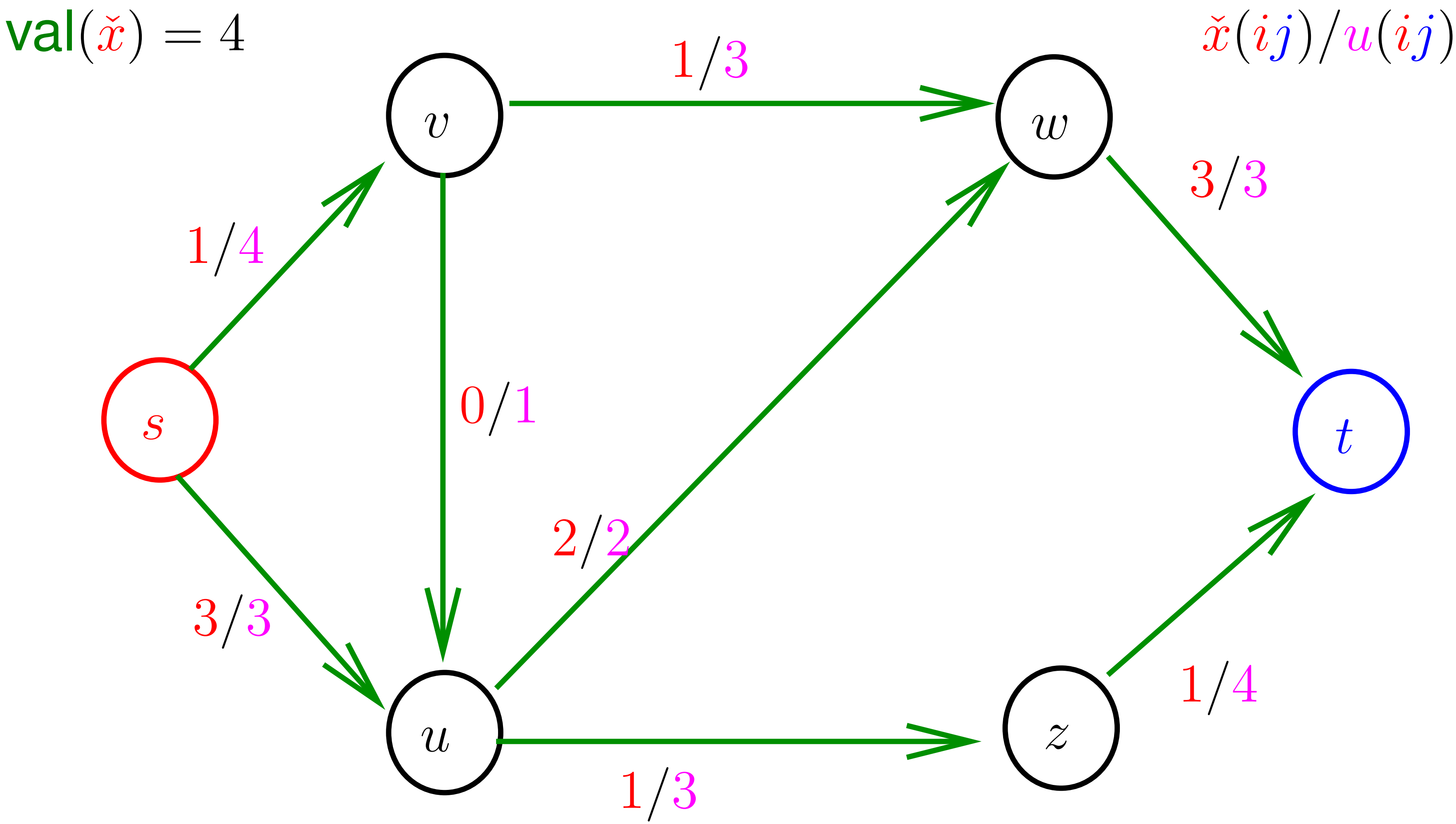
# Fluxo bloqueador 1

$\text{dist}(s,t) = 3$

$\check{z}(ij)/u'(ij)$

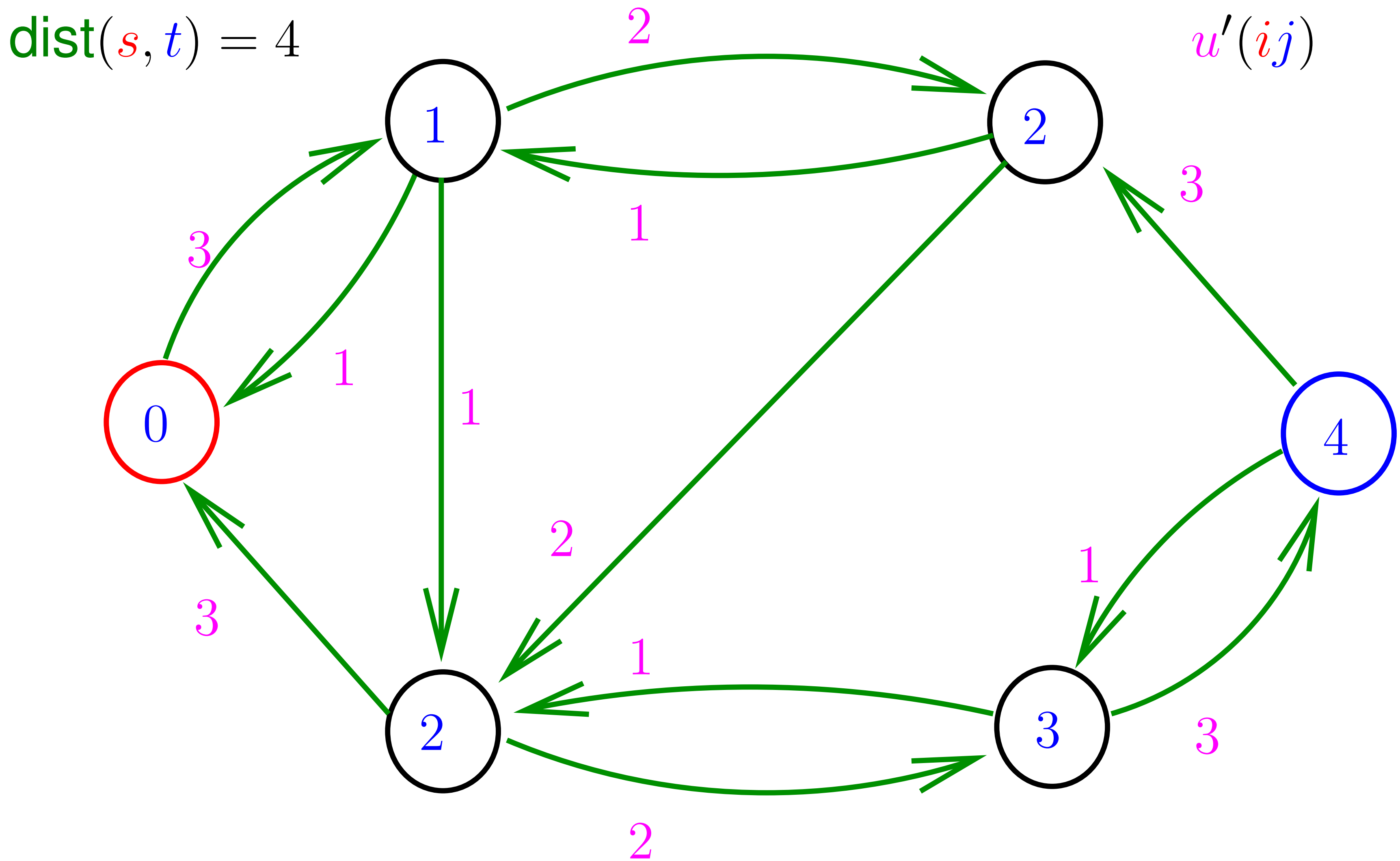


# Rede 2

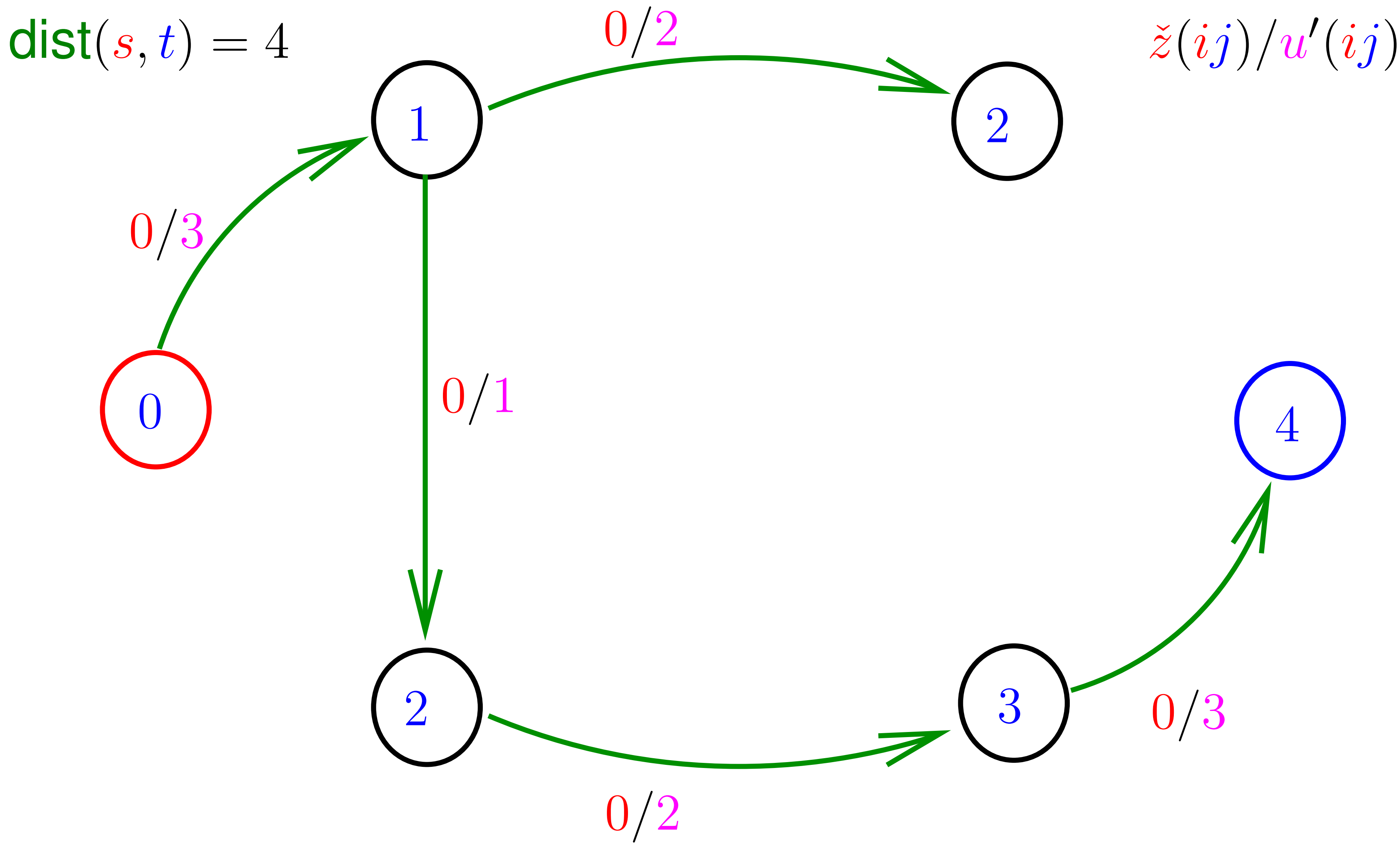



Arcos ausentes têm capacidade nula.

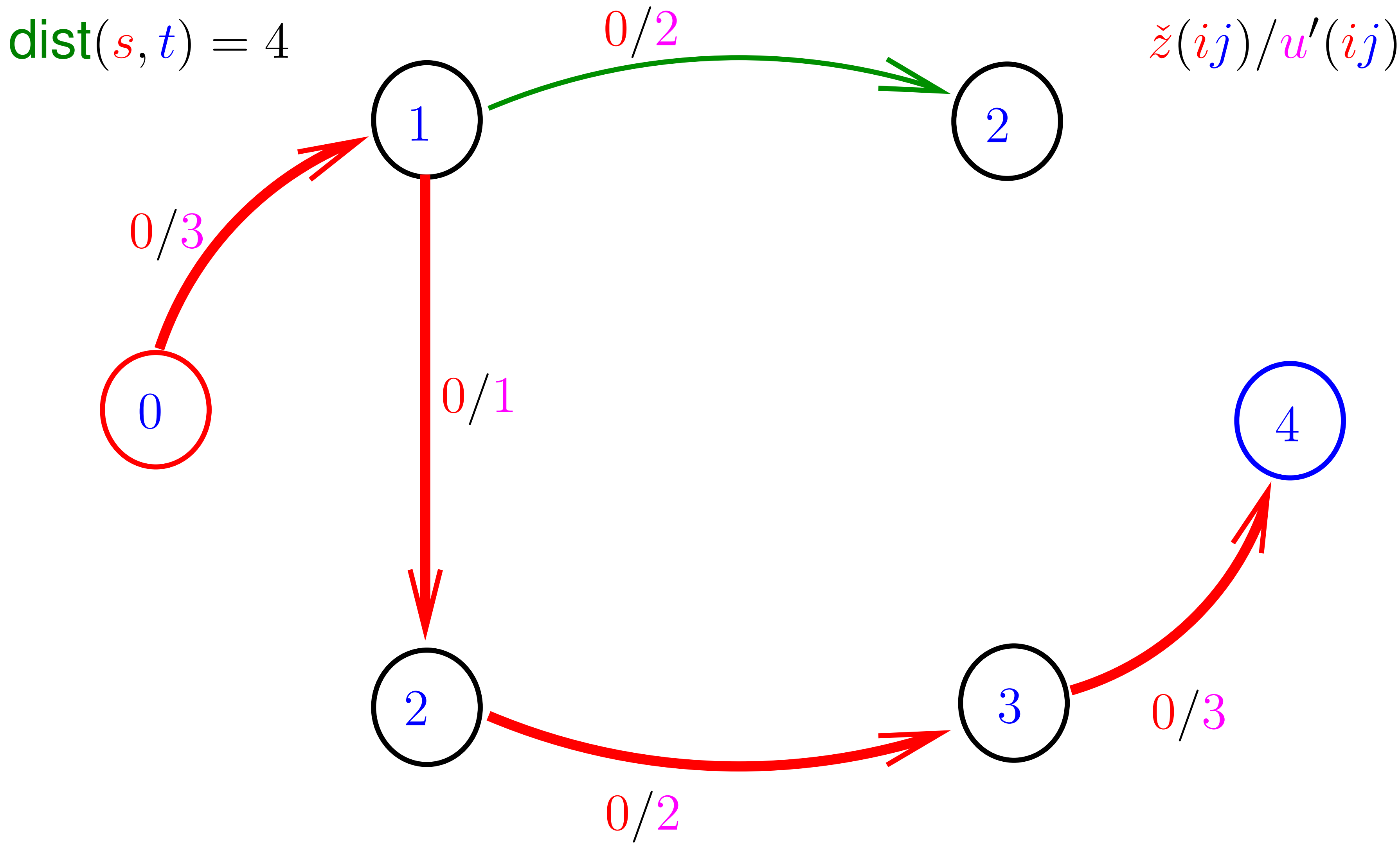
# Rede residual 2



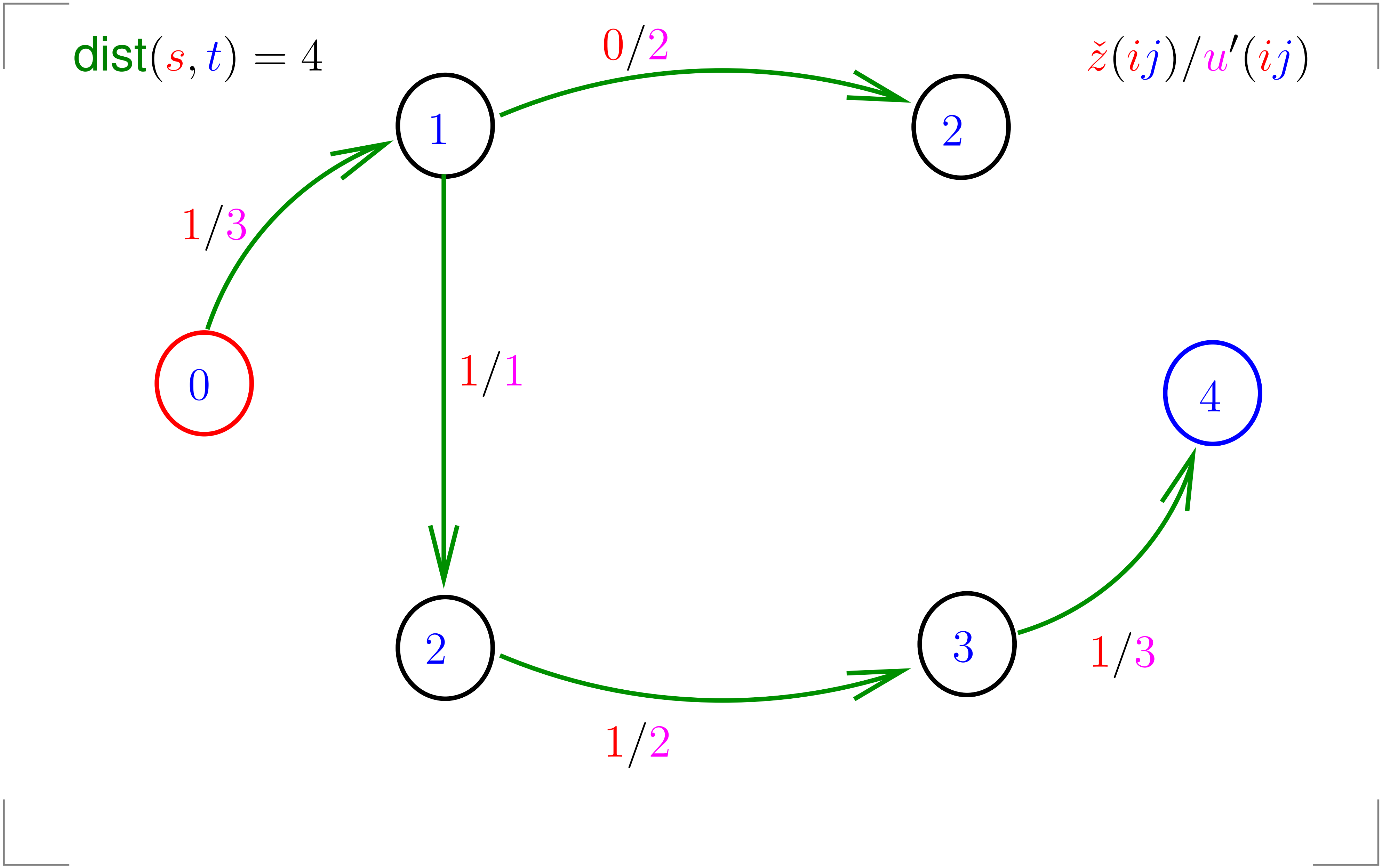
# Rede em camadas 2



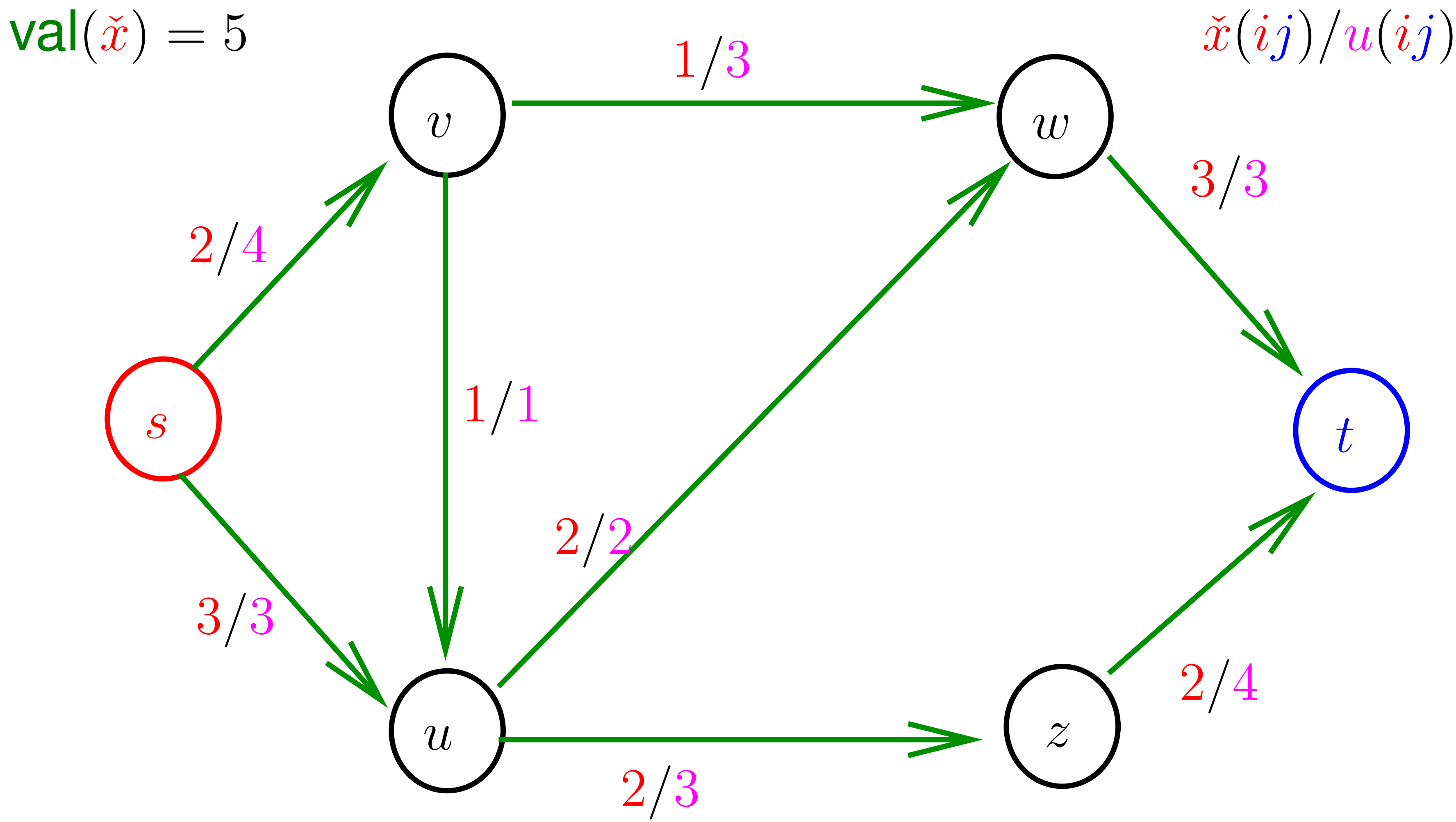
# Caminho de incremento 4



# Fluxo bloqueador 2

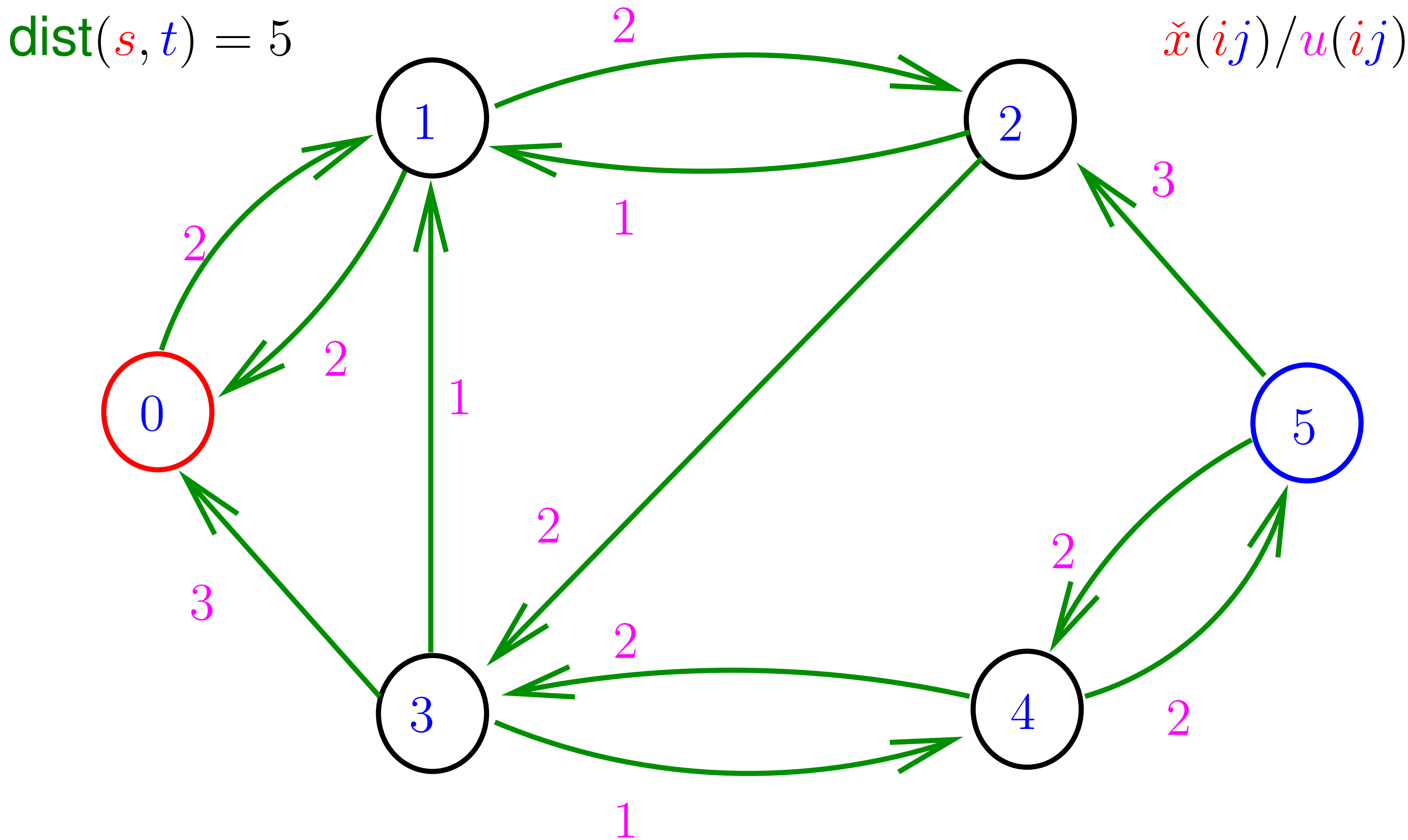


# Rede 3

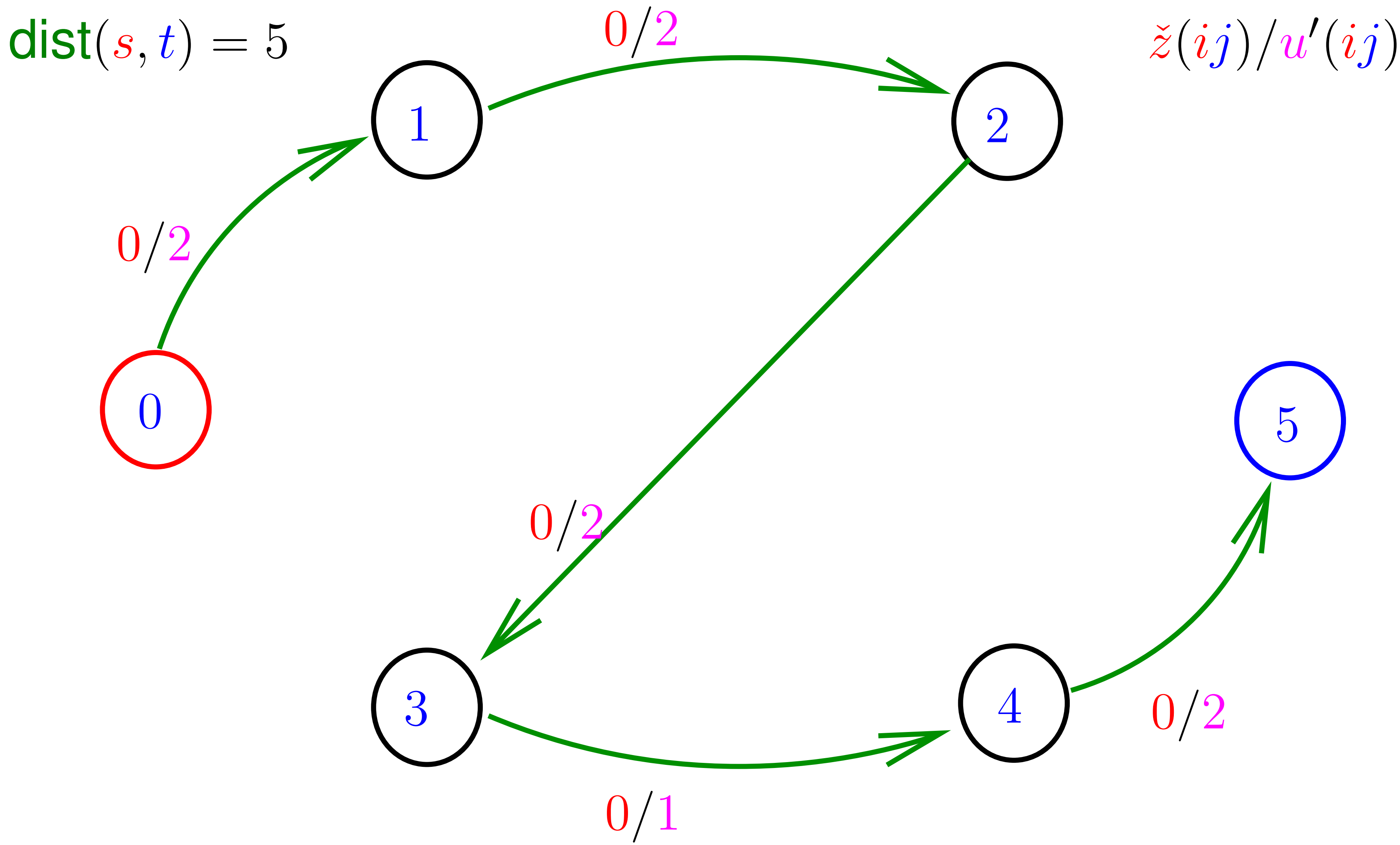



Arcos ausentes têm capacidade nula.

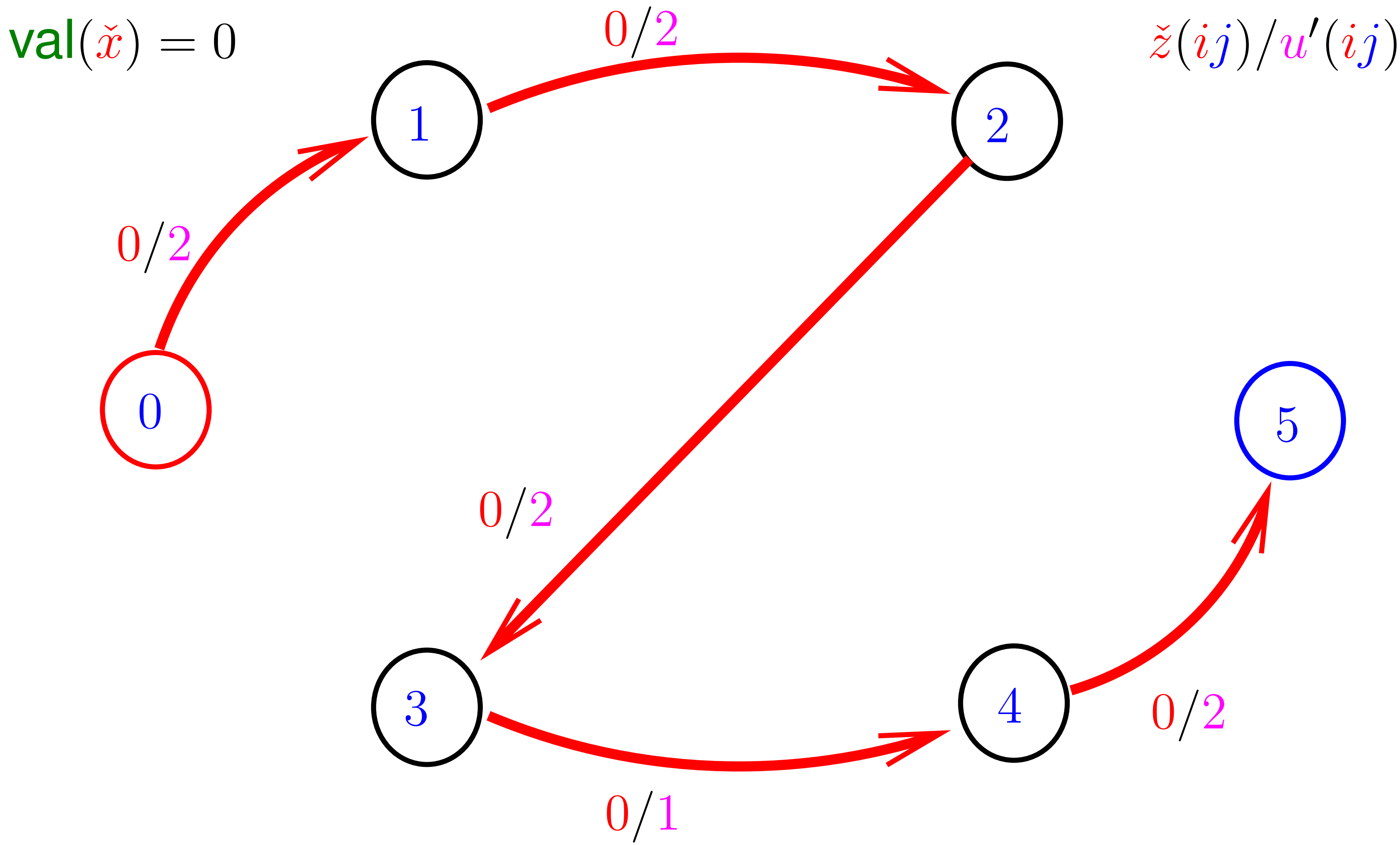
# Rede residual 3



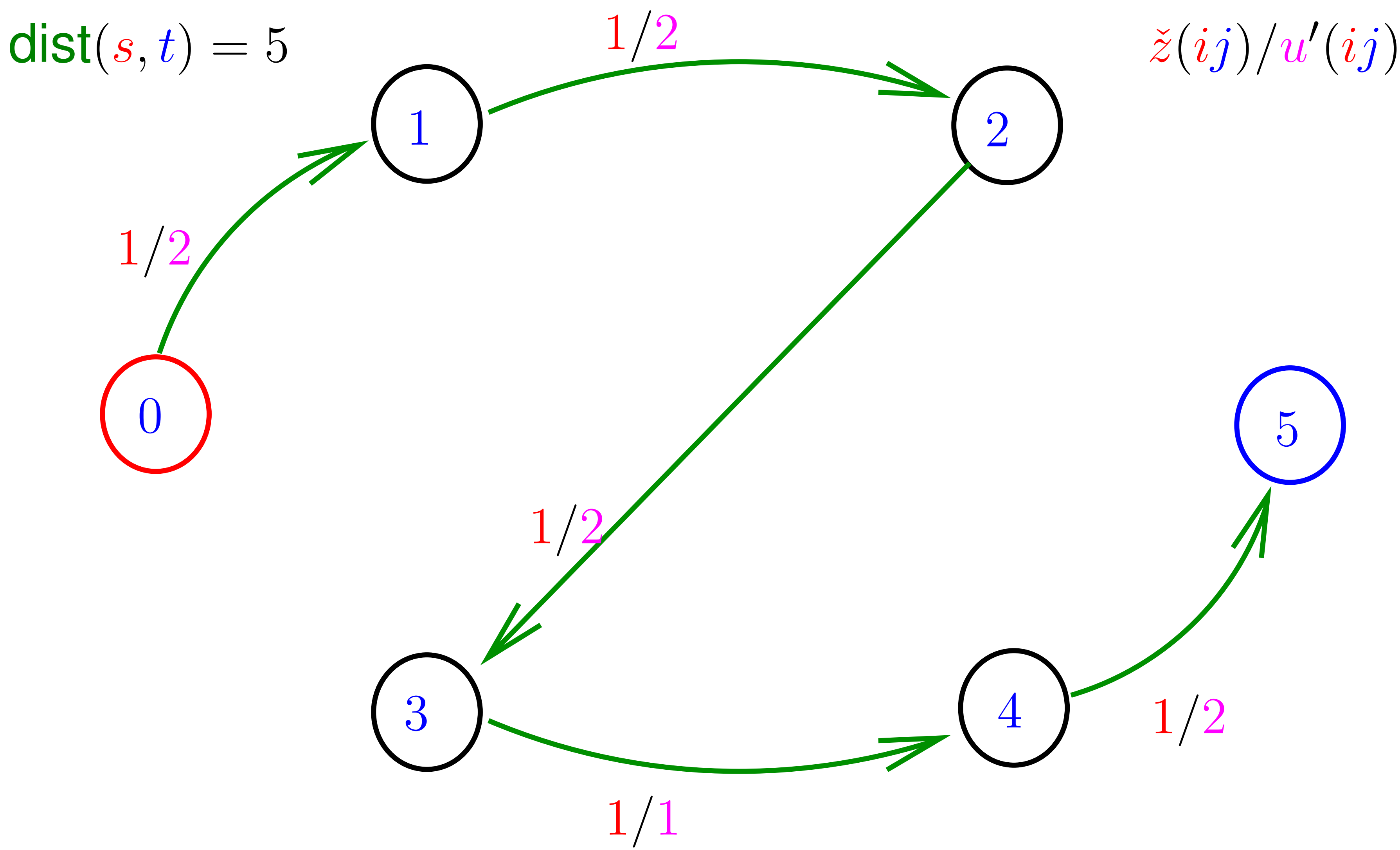
# Rede em camadas 3



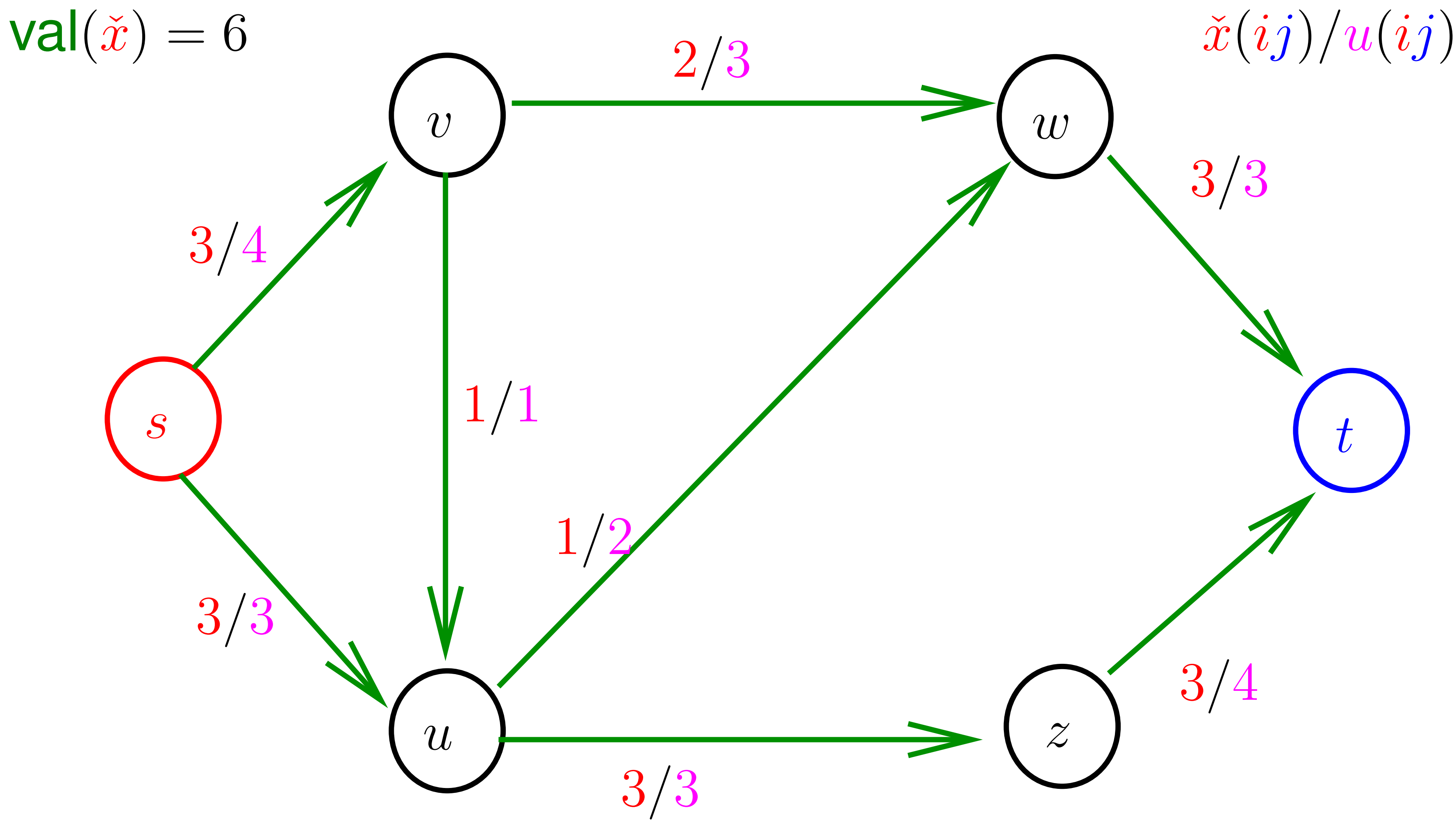
# Caminho de incremento 5



# Fluxo bloqueador 3

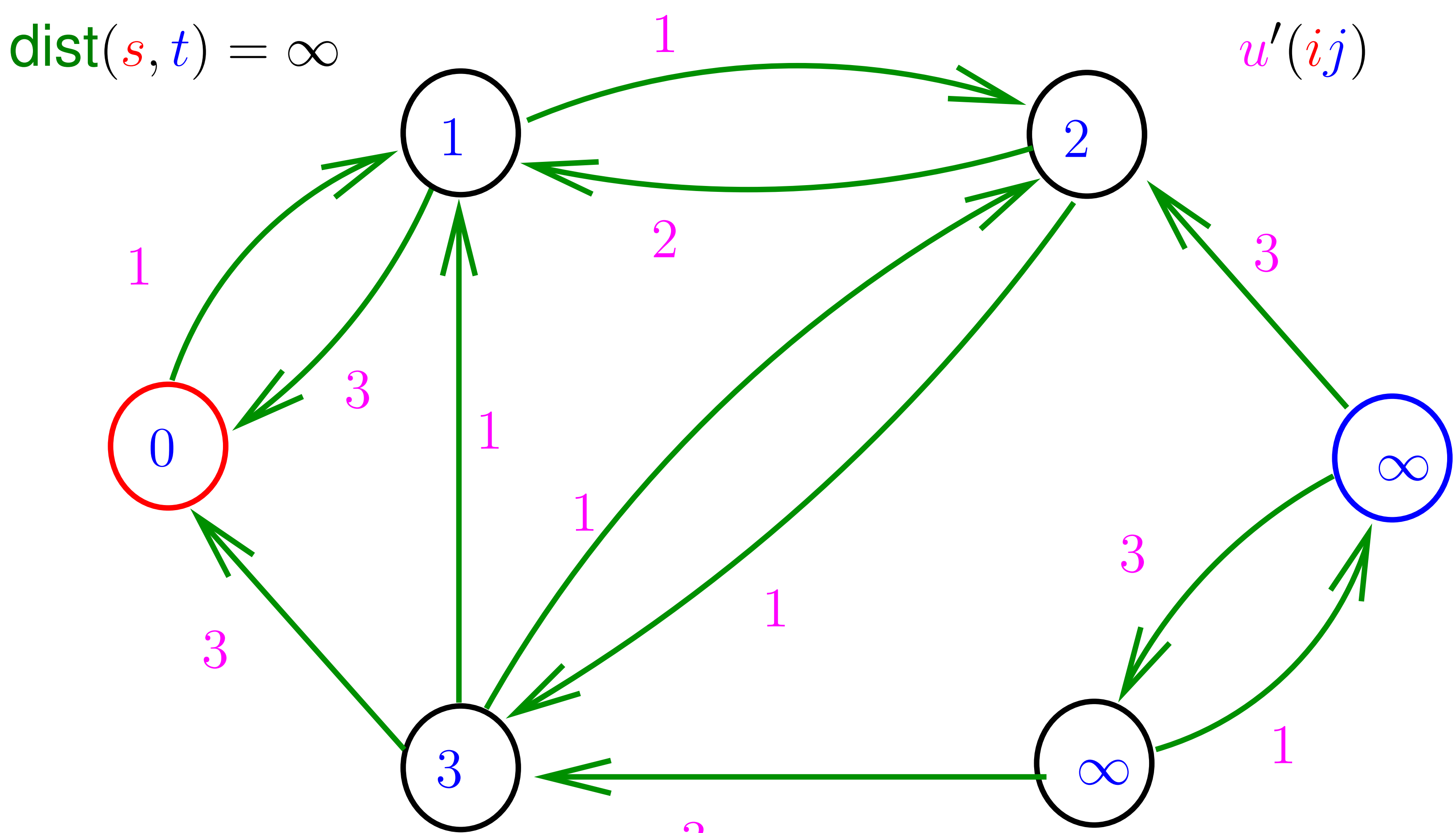


# Rede 4

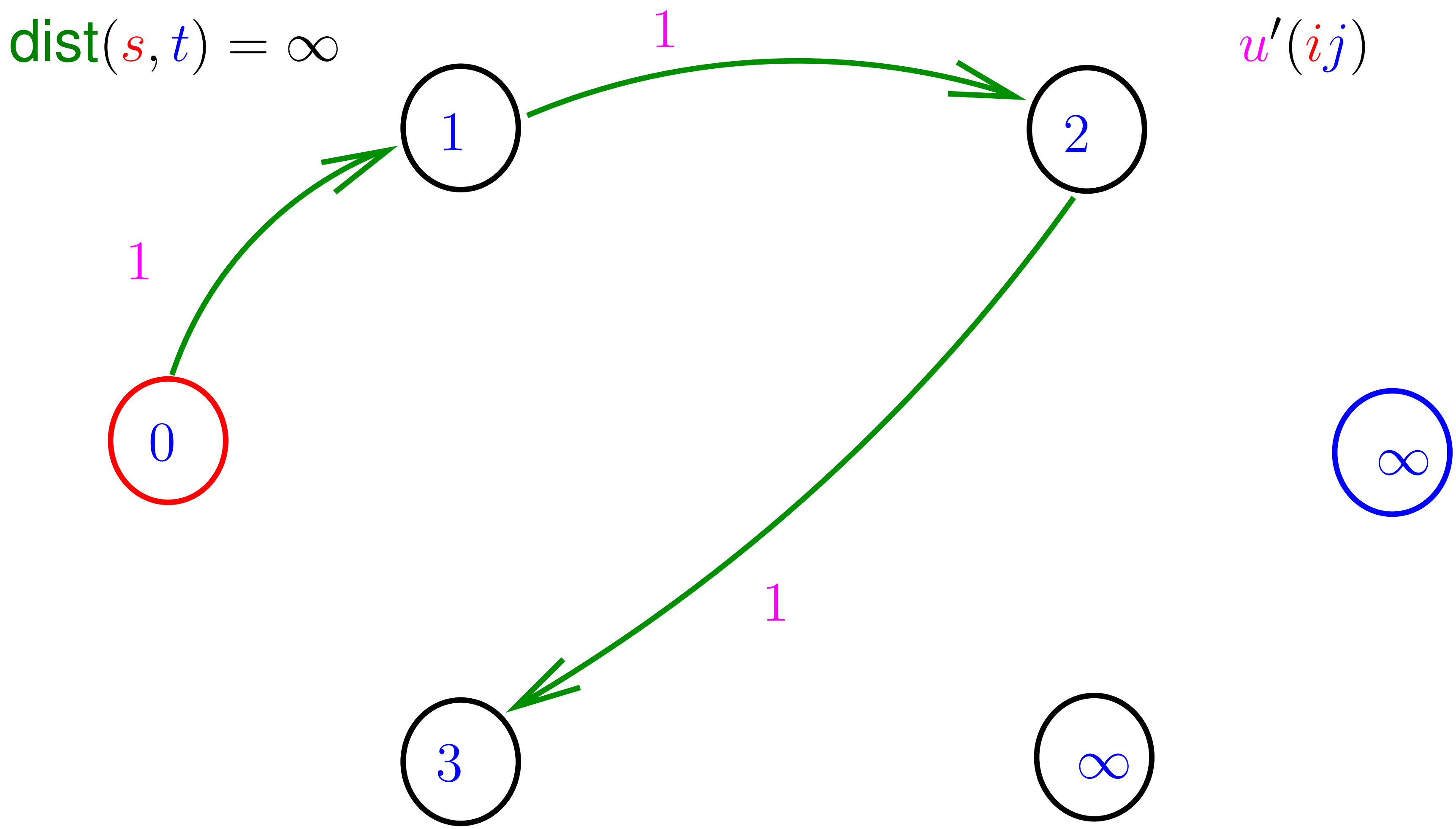



Arcos ausentes têm capacidade nula.

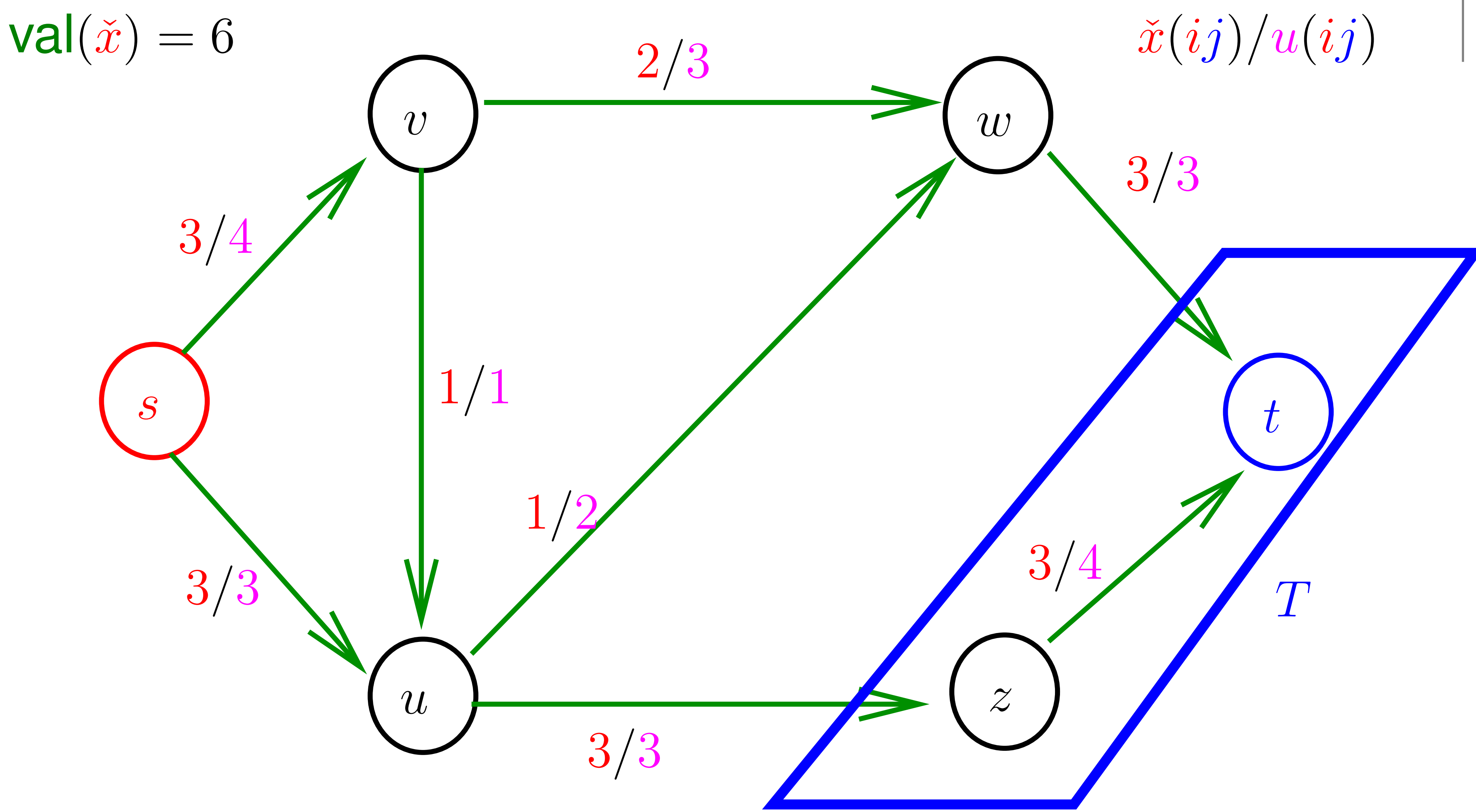
# Rede residual 4



# Rede em camadas 4



# Fluxo máximo e corte mínimo



# Algoritmo de Dinits

Método dos “fluxos boqueadores”

PASSO BLOQUEADOR:

encontre um fluxo bloqueador $\check{z}$ na “rede em camadas” do fluxo corrente $\check{x}$.

Intere novamente com $\check{x} + \check{z}$ no papel de $\check{x}$.

Na descrição do algoritmo de Dinits que veremos a rede em camadas será representada implicitamente por um 1-potencial $(s, *)$-ótimo no grafo residual $(N, A_{\check{x}})$.

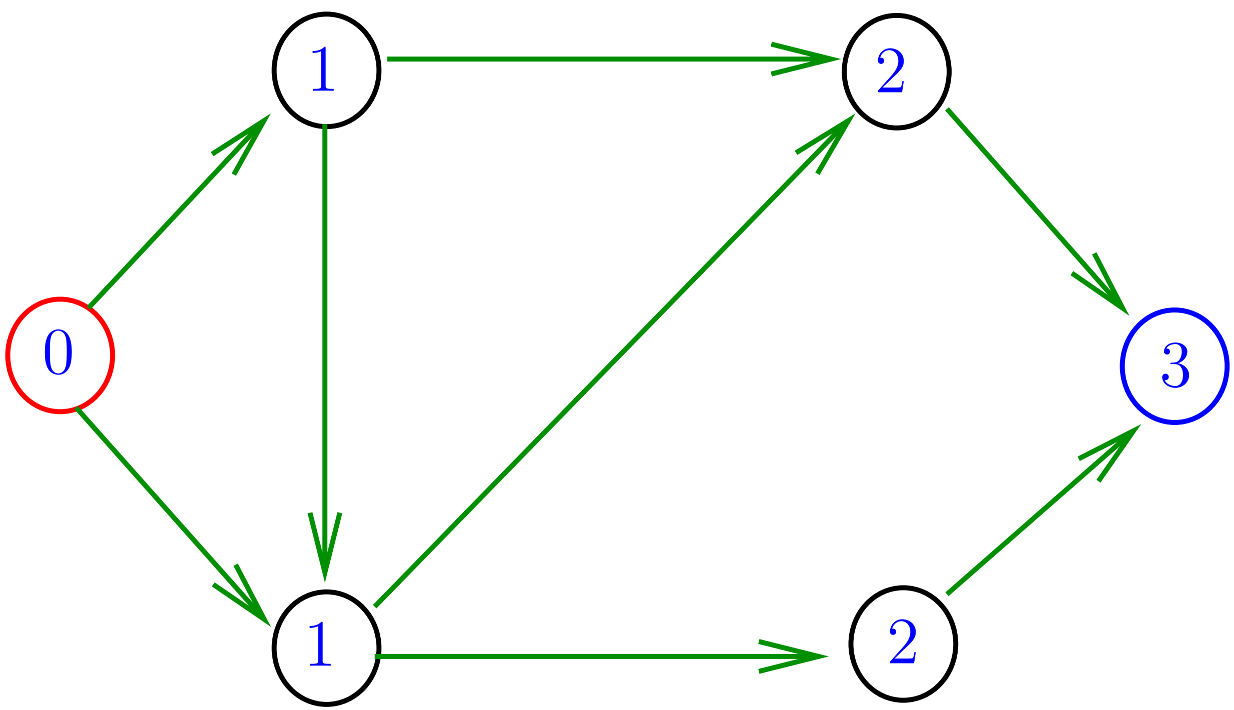
Observação: pode-se usar um 1-potencial $(*, t)$-ótimo ao invés de $(s, *)$-ótimo.

# Potencial ótimo

Dado um nó $s$, um 1-potencial $y$ é $(s,*)$-ótimo se:

- existe um caminho $P$ de $s$ a $i$ tal que $|P| = y(i) - y(s)$ ou
- não existe um caminho de $s$ a $i$ e $y(i) - y(s) \geq n$.

Um arco $ij$ é justo se $y(j) - y(i) = 1$.



# Potencial ótimo

POTENCIAL-ÓTIMO $(N, A, s)$

**para cada** $i$ em $N$ **faça**<br>
    $y(i) \leftarrow n$<br>
$y(s) \leftarrow 0$<br>
$L \leftarrow \langle s \rangle$     $\triangleright$ $L$ funciona como uma fila<br>
**enquanto** $L \neq \langle\rangle$ **faça**<br>
    retire o primeiro elemento, digamos $i$, de $L$<br>
    **para cada** arco $ij$ em $A(i)$ **faça**     $\triangleright$ $ij$ entra em $j$<br>
        **se** $y(j) > y(i) + 1$     $\triangleright$ $y(j) = n$<br>
            **então** $y(j) \leftarrow y(i) + 1$<br>
                acrescente $j$ ao final de $L$<br>
**devolva** $y$

Consumo de tempo: $\mathrm{O}(n + m)$

# Algoritmo de Dinits

DINITS $(N, A, u, s, t)$

$\check{x} \leftarrow 0$

**repita**

  $A_{\check{x}} \leftarrow \{ij : \check{x}(ij) < u(ij)\}$

  $y \leftarrow$ POTENCIAL-ÓTIMO$(N, A_{\check{x}}, t)$

  **se** $y(t) - y(s) < n$ **então**

    $\check{x} \leftarrow$ PASSO-BLOQUEADOR$(N, A_{\check{x}}, s, t, \check{x}, y)$

**até que** $y(t) - y(s) \geq n$

$x \leftarrow$ FLUXO$(\check{x})$

$T \leftarrow \{j \in N : y(j) - y(s) \geq n\}$

**devolva** $x$ e $T$

# Invariantes

No início de cada iteração das linhas 3–6 vale que:

(i0) $\check{x}$ é um $st$-pseudofluxo que respeita $u$

(i1) $y$ é um 1-potencial $(s, *)$-ótimo no de $(N, A_{\check{x}})$

# Número de iterações

Digamos que uma fase do algoritmo é uma execução das linhas 3–6.

Fato. O númemo de fases é $\leq n - 1$.

Demonstração (rascunho):

- $(N, A_k)$ := rede residual construída da fase $k$ (linha 3);
- $y_k$ := 1-potencial $(s, *)$-ótimo obtido na fase $k$ (linha 4). Podemos supor que $y_k(s) = 0$.

Para $k = 1, \ldots, r - 1$ ($r$ = número de fases) vale que:

$(\star)$ $y_{k+1}(t) > y_k(t)$.

Isto implica que o número de fases é $\leq n - 1$.

# Lembra da observação que ajudou?

Para cada arco $ij$ em $A_{k+1}$ vale que

(2A) $y_k(j) - y_k(i) \leq \quad 1$ se $ij \in A_k$;
(2B) $y_k(j) - y_k(i) = -1$ se $ij \notin A_k$;

Demonstração de (2A) (rascunho).

Segue do fato de $y_k$ ser um 1-potencial.

Demonstração de (2B) (rascunho).

Se $ij \notin A_k$, então $ji \in A_k$ e $ji$ pertence a um caminho de incremento $P$ utilizado na linha 6 pelo algoritmo PASSO-BLOQUEADOR durante a fase $k$.

Logo, $y_k(i) - y_k(j) = 1 \Rightarrow y_k(j) - y_k(i) = -1$.

# Rascunho da demonstração de $(\star)$

Seja $P = \langle s = i_0, i_1, i_2, \ldots, i_{q-1}, i_q = t\rangle$ um $st$-caminho mínimo em $(N, A_{k+1})$.

Temos que

$$\begin{aligned} y_{k+1}(t) &= |P| \\ &\geq (y_k(t) - y_k(i_{q-1})) + \cdots + (y_k(i_1) - y_k(s)) \\ &= y_k(t) - y_k(s) \\ &= y_k(t), \end{aligned}$$

onde a desigualdade é devida a (2A) e (2B).

Como pelo menos um arco em cada $st$-caminho mínimo em $(N, A_k)$ foi saturado no passo bloqueador e não está em $(N, A_{k+1})$, então $y_{k+1}(t) > y_k(t)$ (por (2B)).

# Conclusão

O número de fases do algoritmo DINITS é $\leq n - 1$.

# Passo bloqueador

PASSO-BLOQUEADOR $(N, A_{\check{x}}, s, t, \check{x}, y)$

$B \leftarrow \emptyset$ $\quad \triangleright$ armazena os nós bloqueados

$P \leftarrow \langle s \rangle$

$i \leftarrow s$

**enquanto** $s \notin B$ **faça**

  **se** $i = t$ **então**

    $\langle \check{x}, A_{\check{x}} \rangle \leftarrow$ INCREMENTE-FLUXO $(\check{x}, P, A_{\check{x}}, u)$

    $P \leftarrow \langle s \rangle$

    $i \leftarrow s$

  **se** algum $ij$ em $A_{\check{x}}$ é justo e $j \notin B$

    **então** AVANCE$(ij)$

    **senão** RETROCEDA$(i)$

**devolva** $\check{x}$

# Invariantes

No início de cada iteração das linhas 4–11 vale que:

(i0) $\check{x}$ é um $st$-pseudofluxo que respeita $u$;

(i2) para cada $i$ em $B$ não existe caminho de $i$ a $t$ em $(N, A_{\check{x}})$ contendo apenas arcos justos (caminho de $i$ a $t$ no "grafo em camadas");

(i3) $P$ é um caminho de $s$ a $i$ em $(N, A_{\check{x}})$;

(i4) todos os arcos de $P$ são justos;

(i5) todos os nós de $P$ estão em $N - B$, exceto talvez se $P = \langle s \rangle$;

# Avance e retroceda

AVANCE $(ij)$
1 acrescente $ij$ ao final de $P$
2 $i \leftarrow j$

RETROCEDA $(i)$
1 $B \leftarrow B \cup \{i\}$ $\triangleright$ nó $i$ está "bloqueado"
2 **se** $i \neq s$
3 **então** remova o último nó de $P$
4 $i \leftarrow$ último nó de $P$

Consumo de tempo: $\mathrm{O}(1)$.

# Algoritmo de incremento

Recebe um $st$-pseudofluxo $\check{x}$ e um caminho de incremento $P$ e devolve o pseudofluxo $\check{x}$ após enviar "$\delta$ unidades de fluxo através de $P$" e os arcos que ainda não estão saturados.

INCREMENTE-FLUXO $(\check{x}, P, A_{\check{x}}, u)$

$\delta \leftarrow \min\{u(ij) - \check{x}(ij) : ij$ é arco de $P\}$
**para cada** arco $ij$ em $P$ **faça**
  $\check{x}(ij) \leftarrow \check{x}(ij) + \delta$
  $\check{x}(ji) \leftarrow \check{x}(ji) - \delta$
  **se** $\check{x}(ij) = u(ij)$
    **então** $A_{\check{x}} \leftarrow A_{\check{x}} - \{ij\}$
**devolva** $\check{x}$ e $A_{\check{x}}$

Consumo de tempo: $\mathrm{O}(|P|) = \mathrm{O}(n)$

# Consumo de tempo do passo bloqueador

Considere uma execução de PASSO-BLOQUEADOR.

Fato 1. O número de execuções de INCREMENTE-FLUXO é $\leq m$.

Demonstração (rascunho): Em cada execução pelo menos um arco é saturado e removido de $A_x$.

Fato 2. O número de execuções de RETROCEDA é $\leq n$.

Demonstração (rascunho): Em cada execução um nó é acrescentado a $B$ e jamais é removido.

Fato 3. O número de execuções de AVANCE é $\leq nm$.

Demonstração (rascunho): Hmmm, aqui precisamos pensar... *current-arc data structure*.

# Consumo de tempo do passo bloqueador

| Algoritmo | número máximo de execuções | consumo total de tempo |
| --- | --- | --- |
| AVANCE | $\leq nm$ | $\mathrm{O}(nm)$ |
| RETROCEDA | $\leq n$ | $\mathrm{O}(n)$ |
| INCREMENTE-FLUXO | $\leq m$ | $\mathrm{O}(nm)$ |

O consumo de tempo do algoritmo
PASSO-BLOQUEADOR é $\mathrm{O}(nm)$ .

# Consumo de tempo do Dinits

| Algoritmo | número máximo de execuções | consumo total de tempo |
|---|---|---|
| POTENCIAL-ÓTIMO | $\leq n-1$ | $\mathrm{O}(n(n+m))$ |
| PASSO-BLOQUEADOR | $\leq n$ | $\mathrm{O}(n^2 m)$ |

O consumo de tempo do algoritmo DINITS é $\mathrm{O}(n^2 m)$.

# Resumo

O número de fases do algoritmo DINITS é $\leq n - 1$.

O consumo de tempo do algoritmo PASSO-BLOQUEADOR é $\mathrm{O}(nm)$ .

O consumo de tempo do algoritmo DINITS é $\mathrm{O}(n^2m)$.

# Observações

| | consumo de tempo passo bloqueador | consumo de tempo |
|---|---|---|
| DINITS | $\mathrm{O}(nm)$ | $\mathrm{O}(n^2m)$ |
| Karzanov | $\mathrm{O}(n^2)$ | $\mathrm{O}(n^3)$ |
| Sleator-Tarjan | $m \log n$ | $\mathrm{O}(nm \log n)$ |

Karzanov utilizou "pré-fluxos".

Sleator-Tarjan utilizaram "*dynamic-tree data structure*".

Problema: É possível encontrar um fluxo bloqueador em tempo $\mathrm{O}(n+m)$?