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	$\leq 2m(1+\lfloor \lg U \rfloor)$ $\mathrm{O}(m \lg U)$	$\mathrm{O}(n^2 m \lg U)$
CAPACITY-SCALING	$\leq 2m(1+\lfloor \lg U \rfloor)$ $\mathrm{O}(m \lg U)$	$\mathrm{O}(m^2 \lg U)$
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.

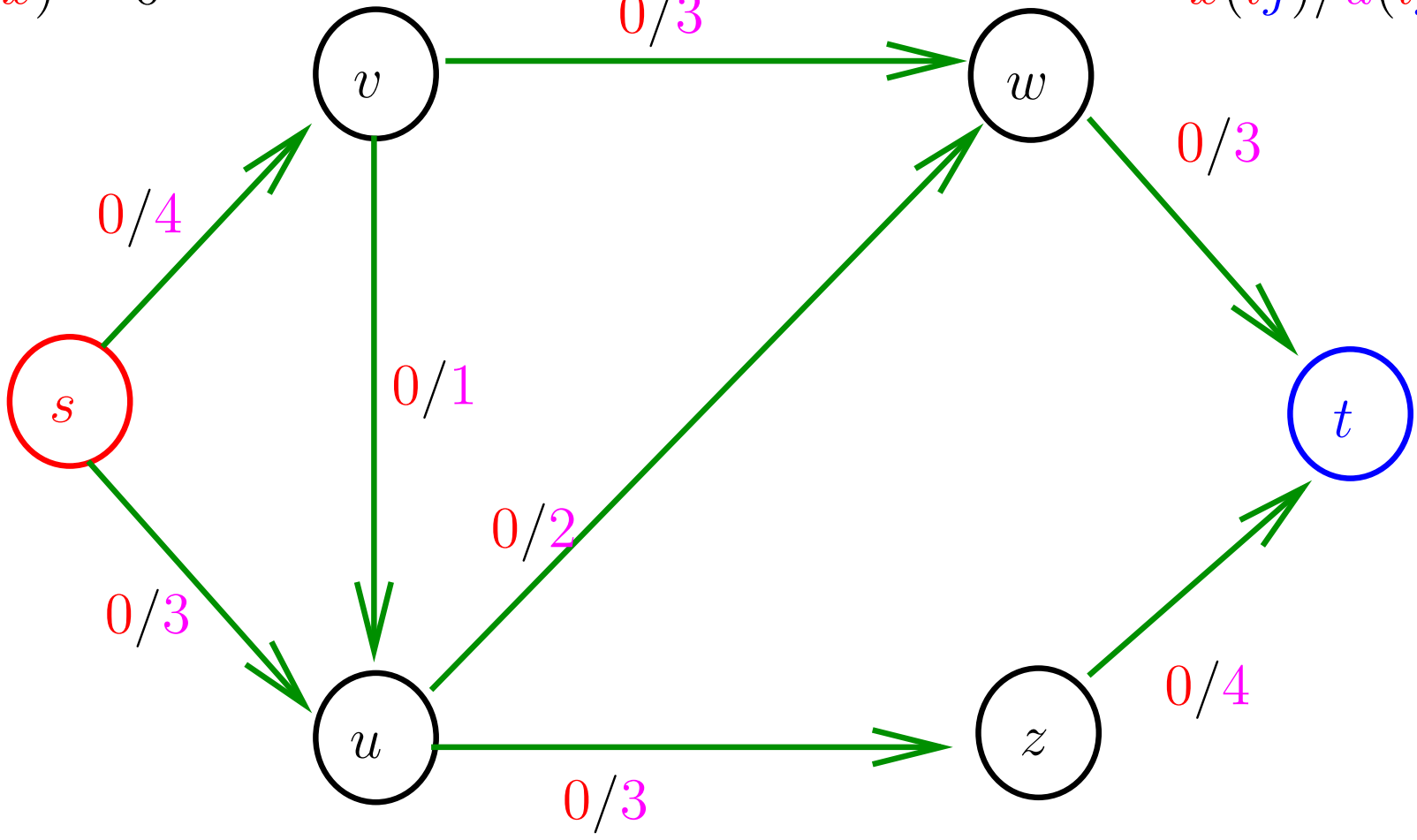
Idéia: reutilizar o resultado de cada busca em largura o máximo possível.

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

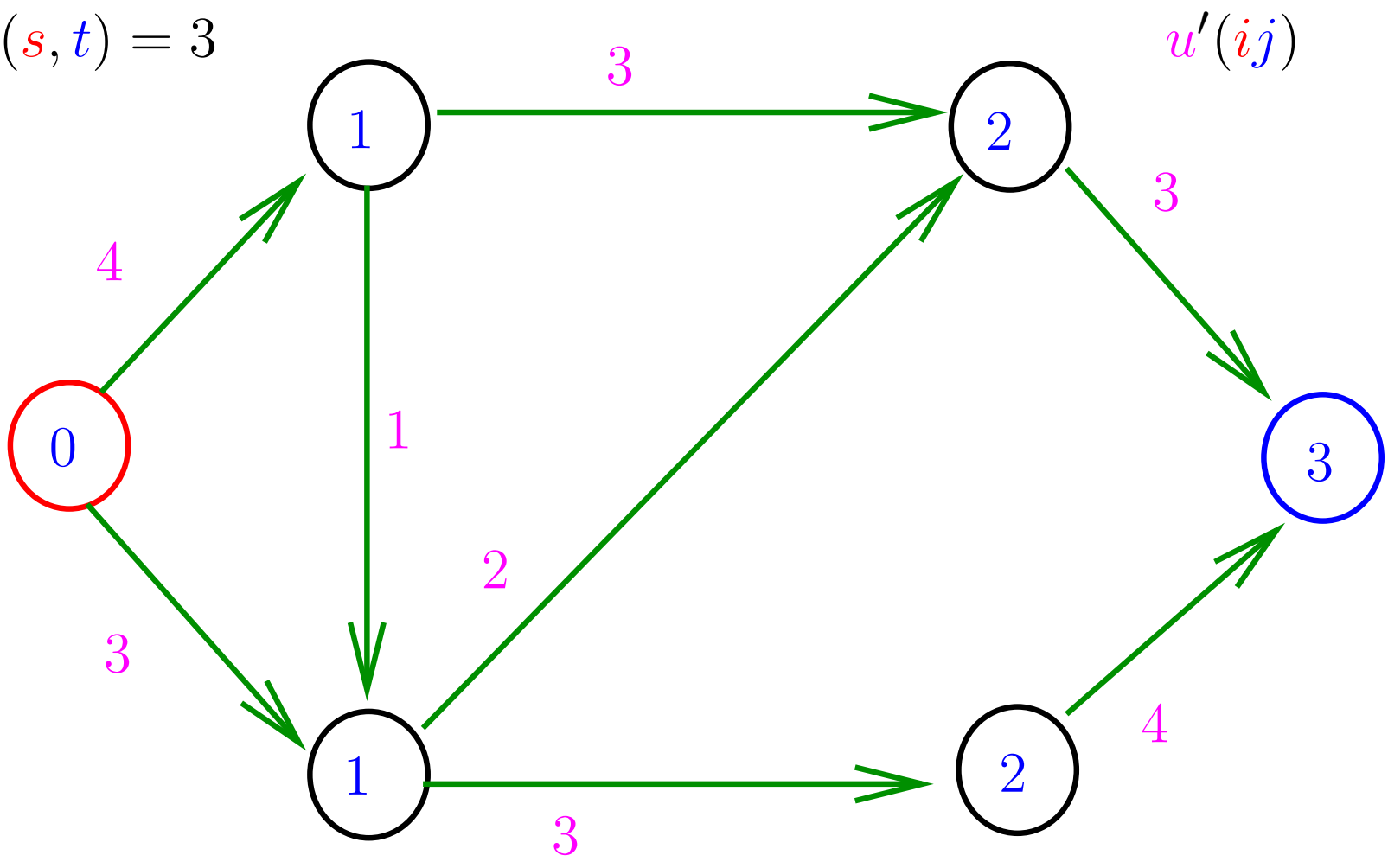
$\mathsf{val}(\check{x}) = 0$

$\check{x}(ij)/u(ij)$

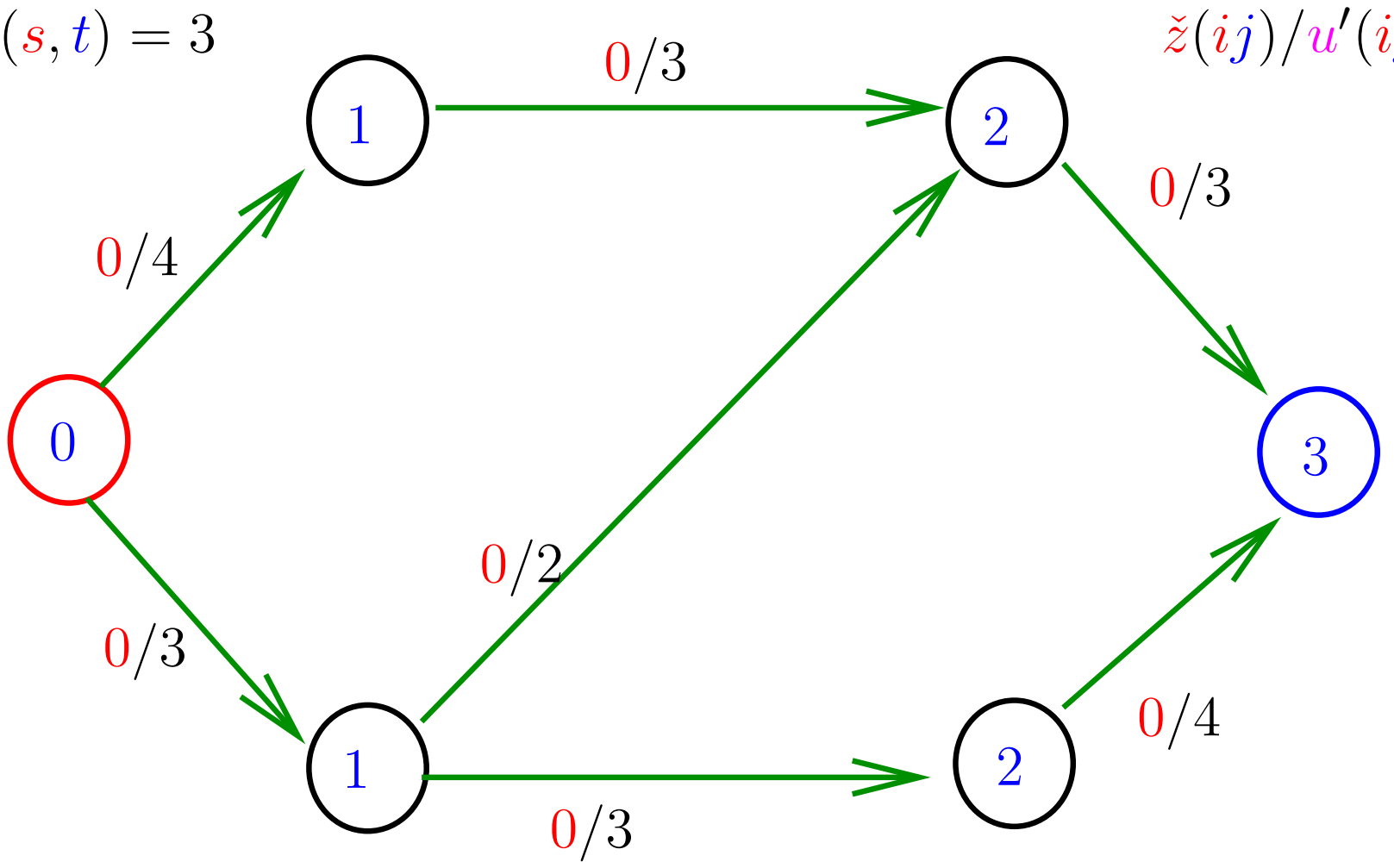


Arcos ausentes têm capacidade nula.

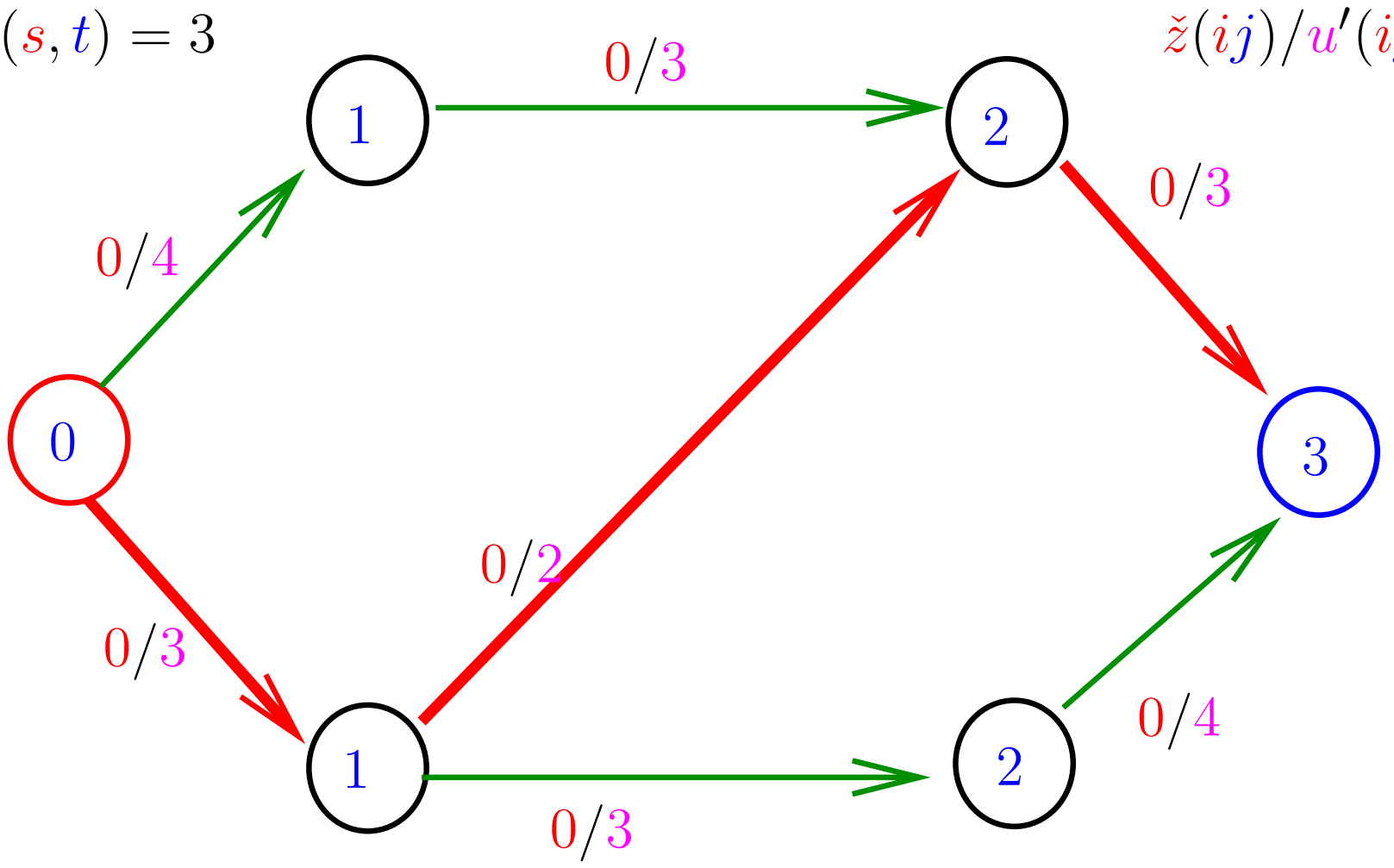
Rede residual 1



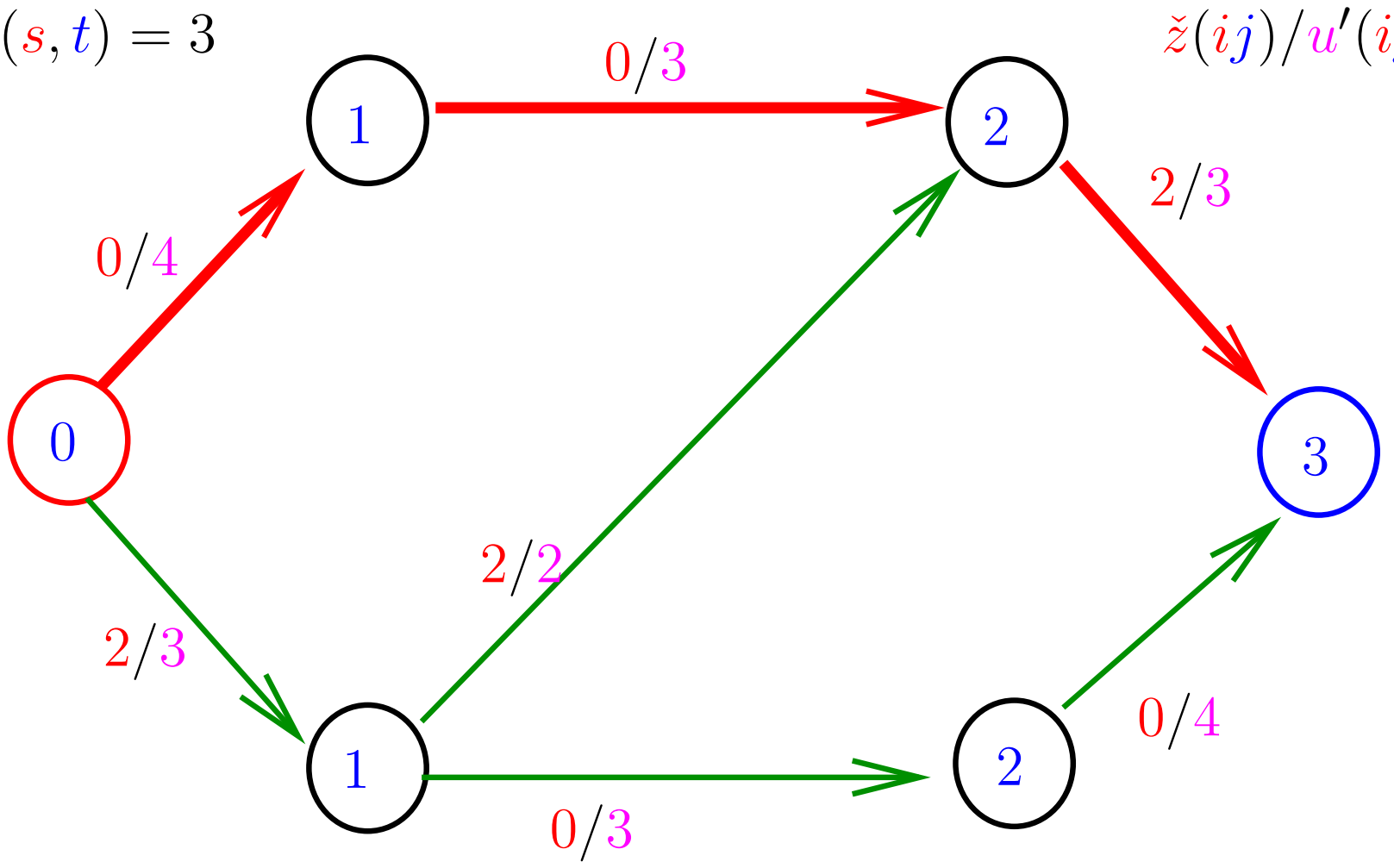
Rede em camadas 1



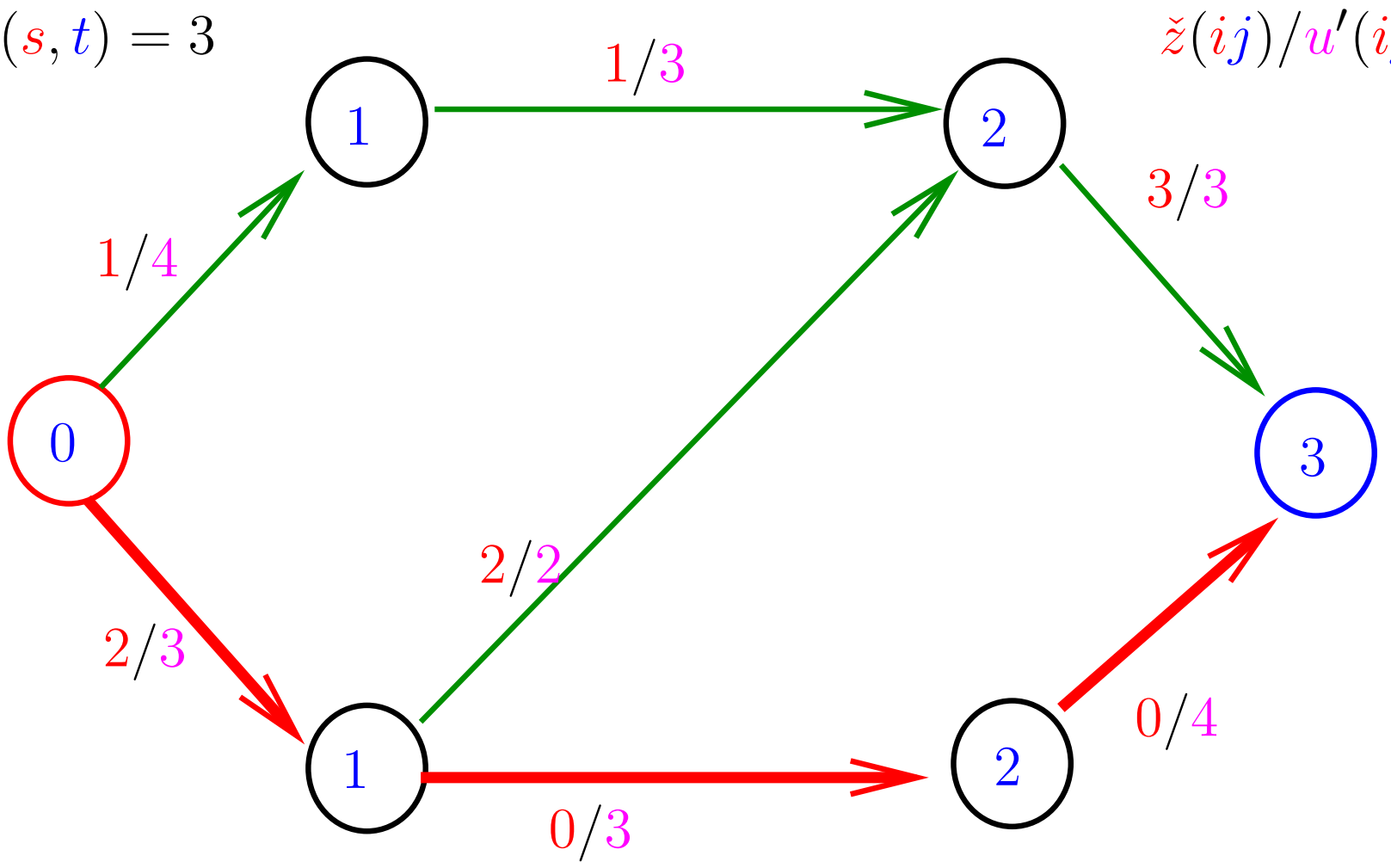
Caminho de incremento 1



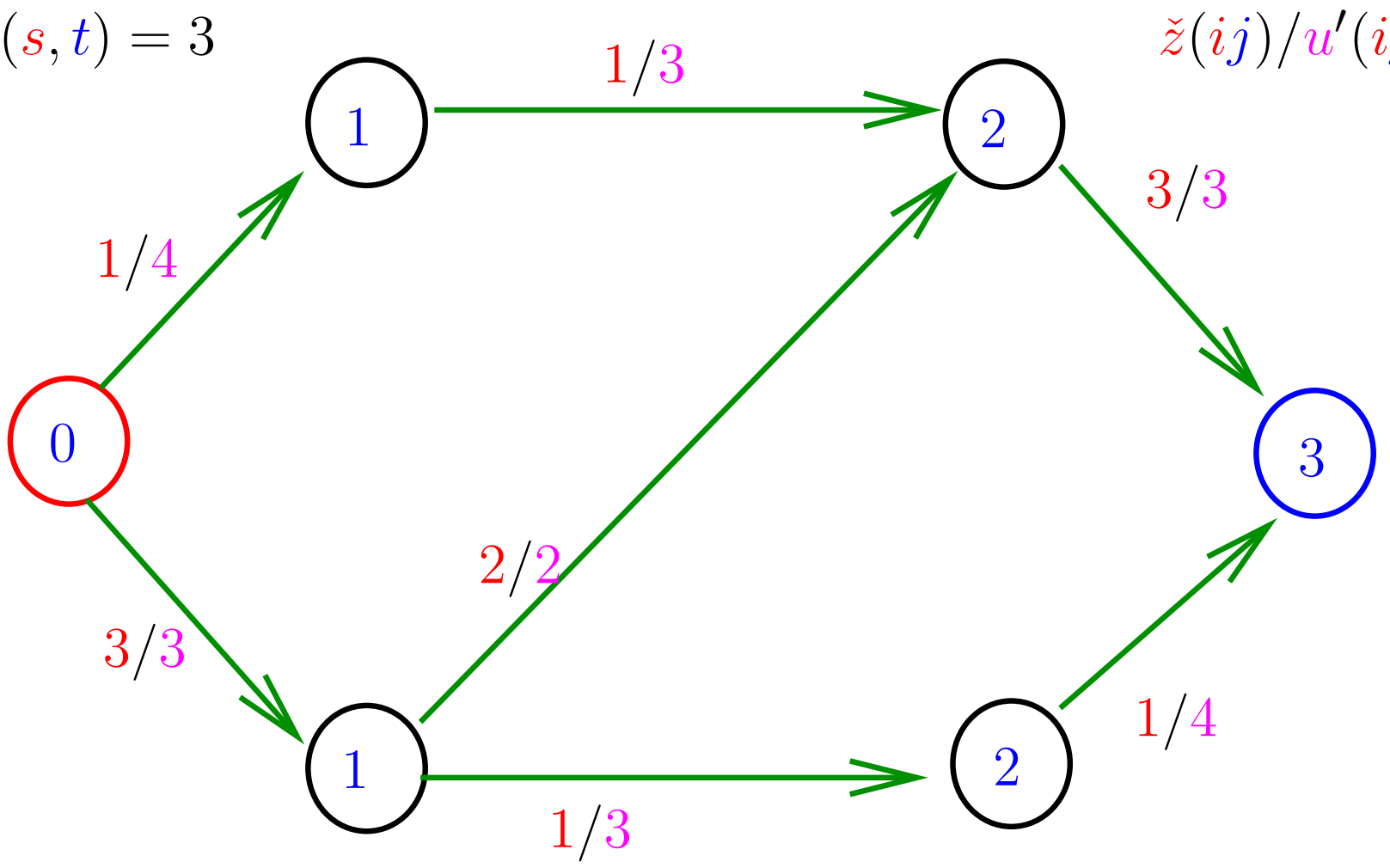
Caminho de incremento 2



Caminho de incremento 3



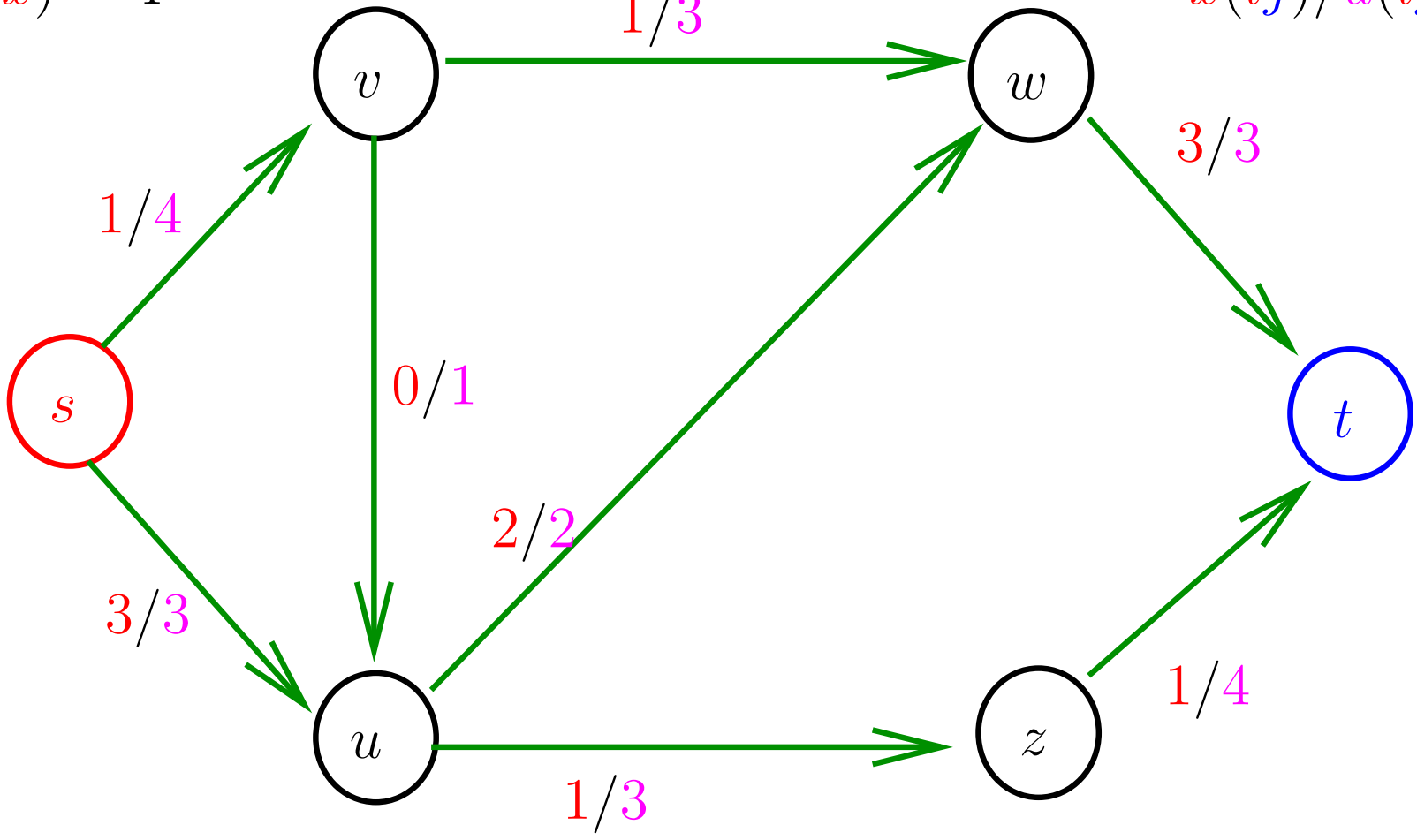
Fluxo bloqueador 1



Rede 2

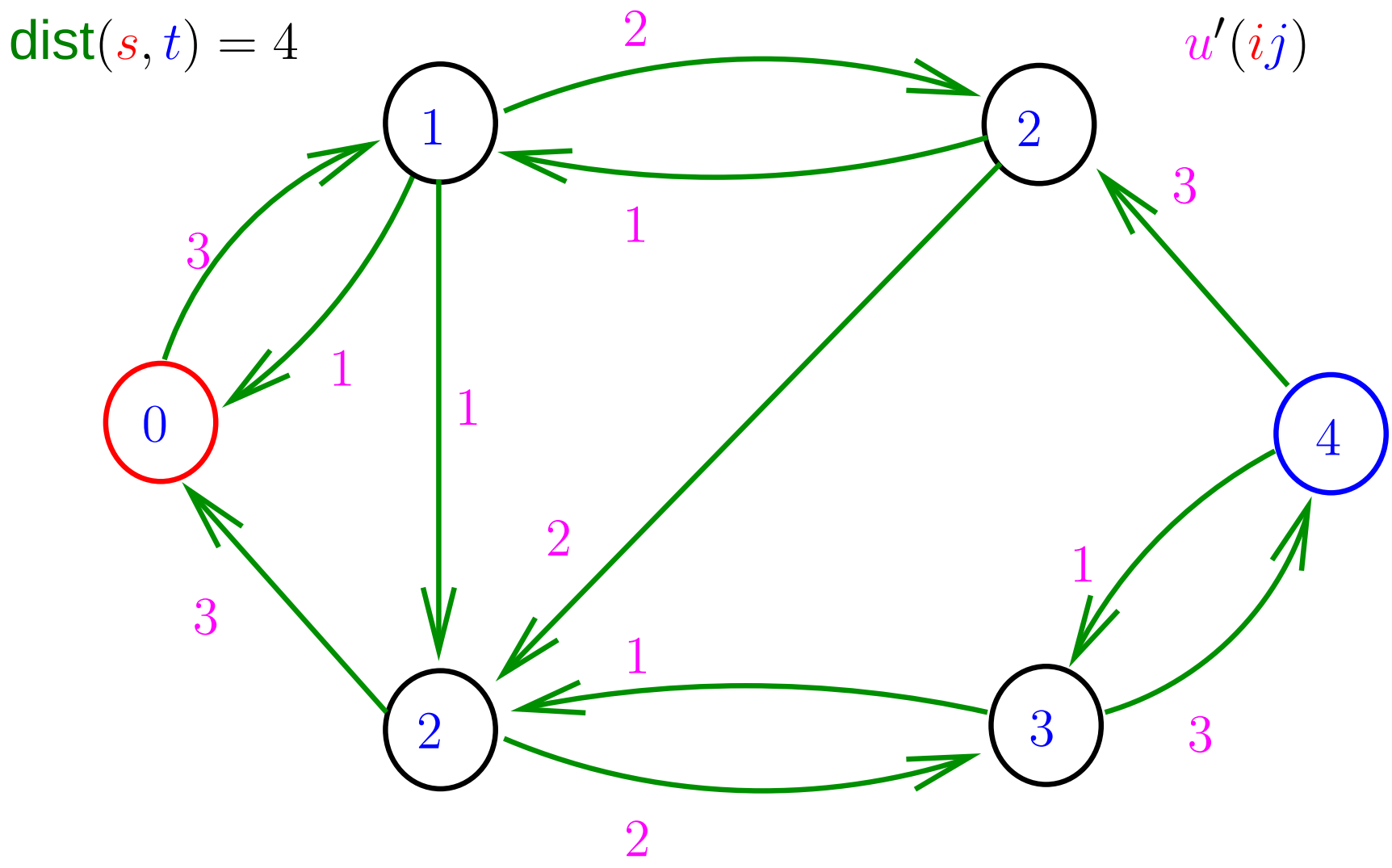
$\mathsf{val}(\check{x}) = 4$

$\check{x}(ij)/u(ij)$

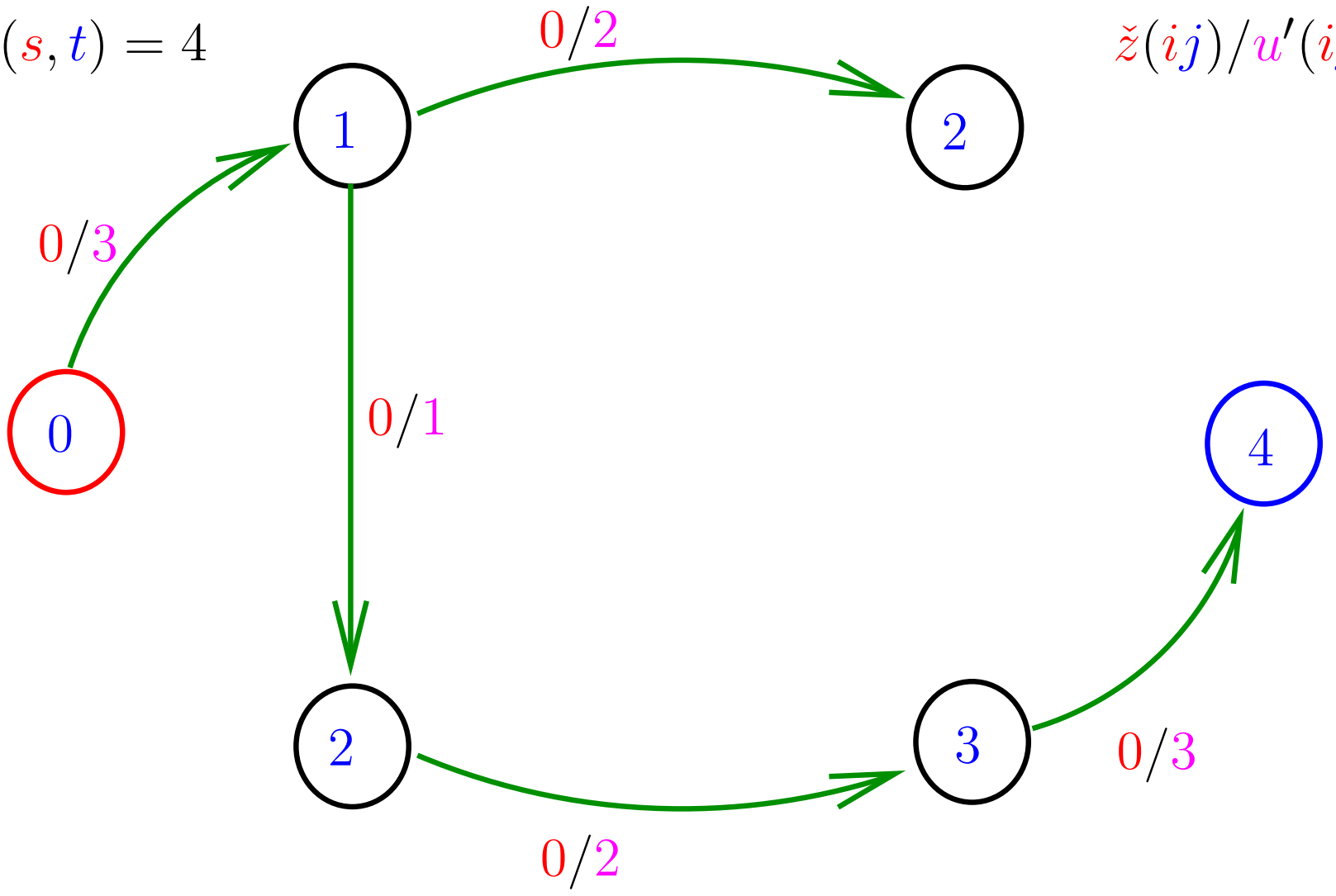


Arcos ausentes têm capacidade nula.

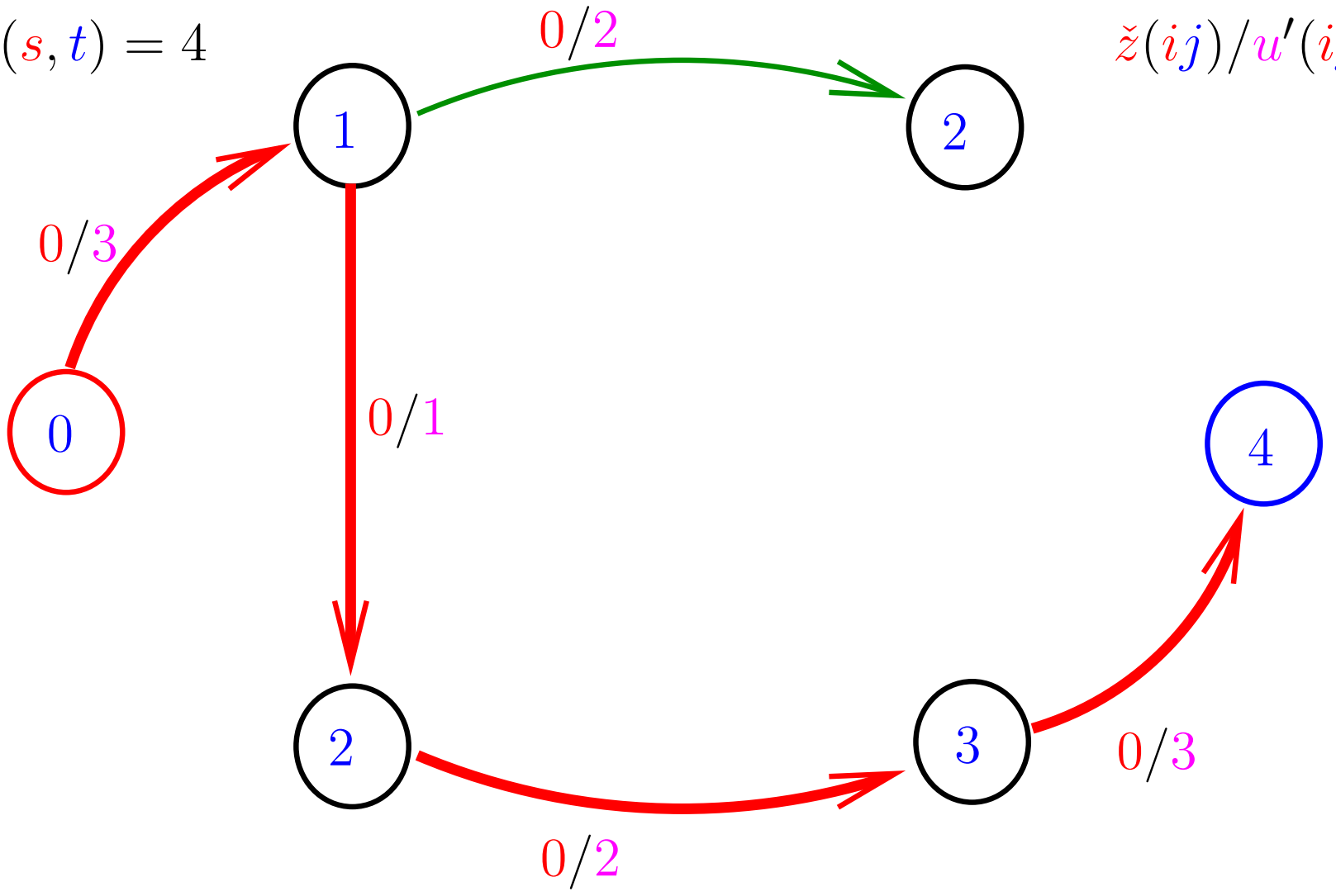
Rede residual 2



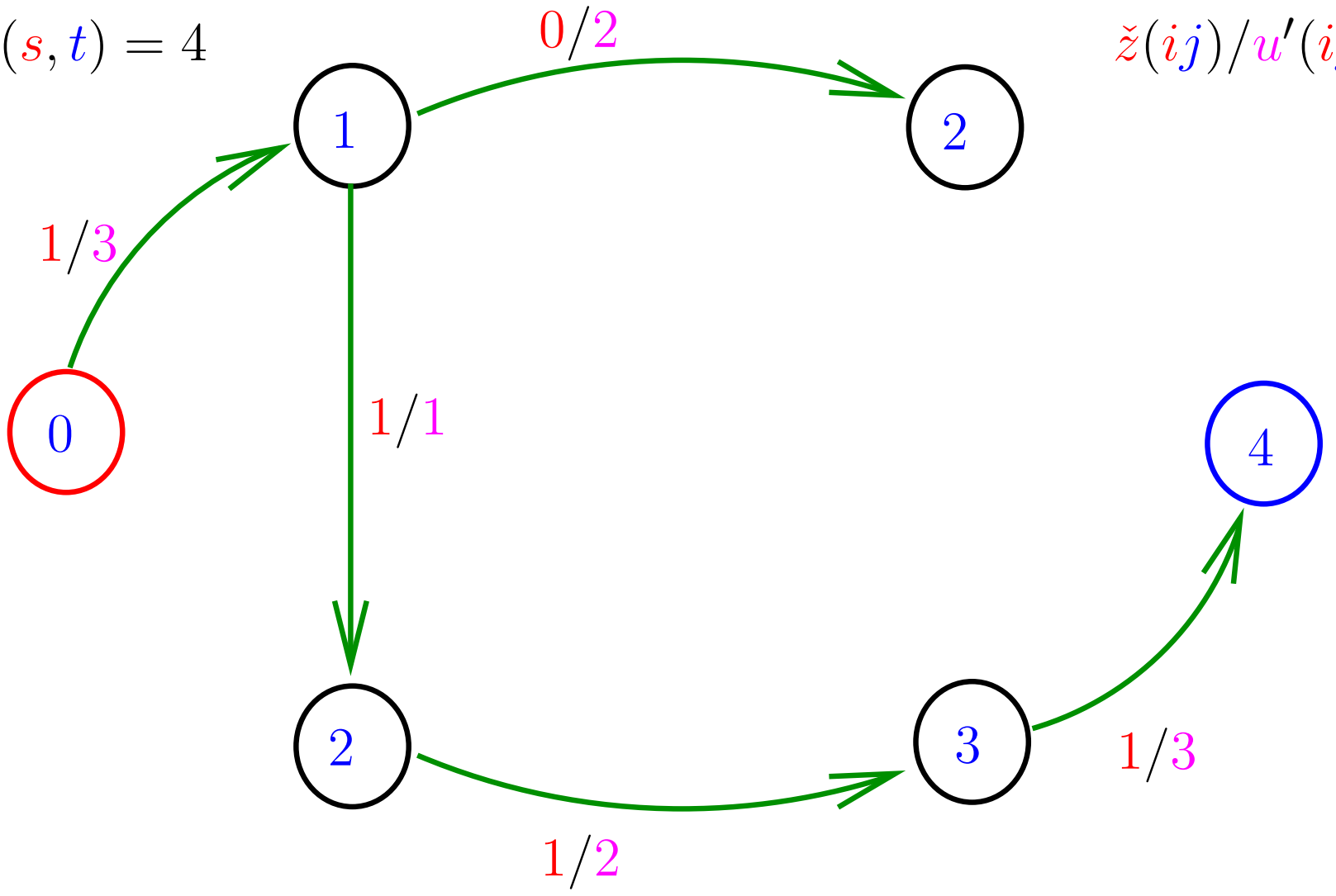
Rede em camadas 2



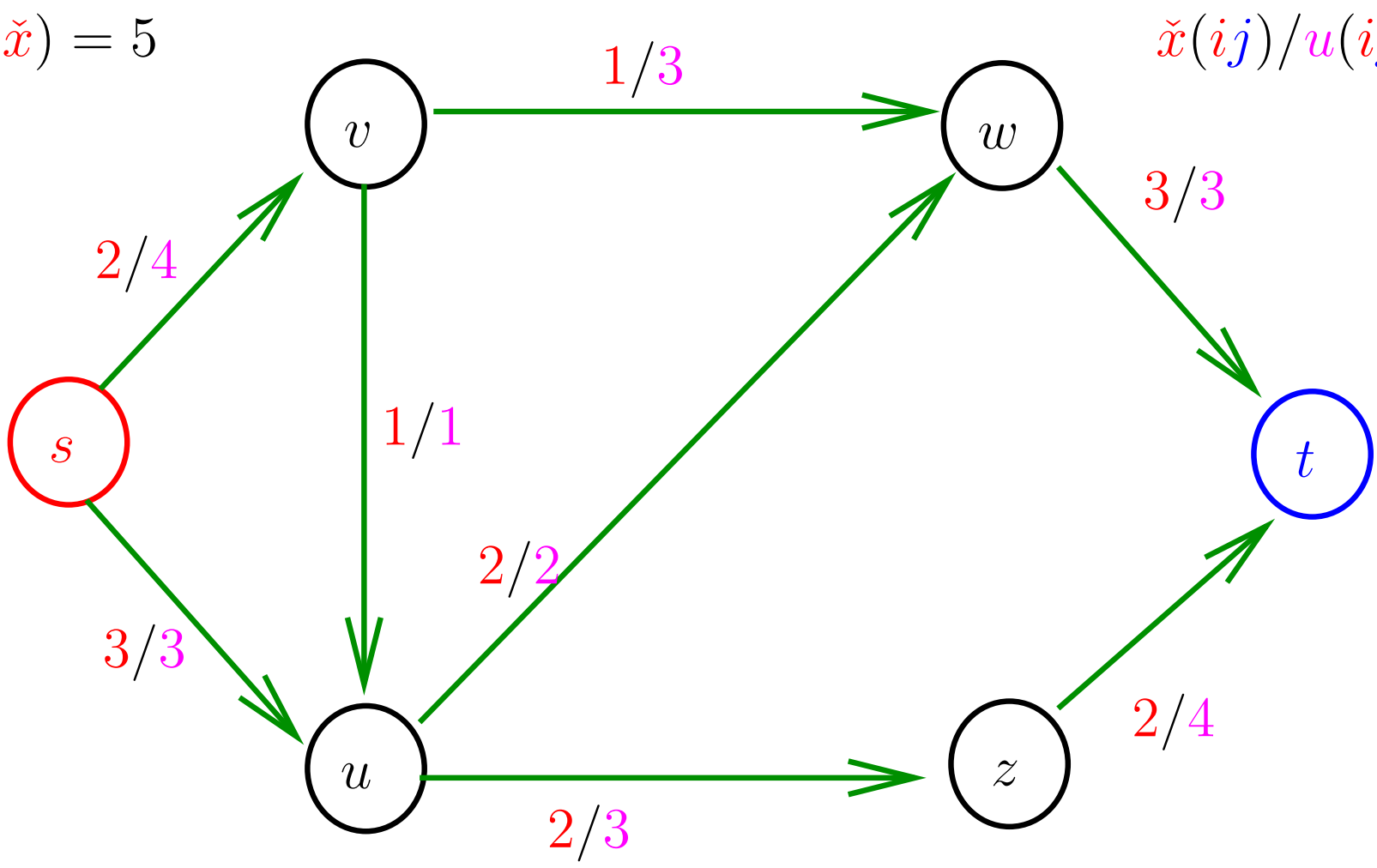
Caminho de incremento 4



Fluxo bloqueador 2

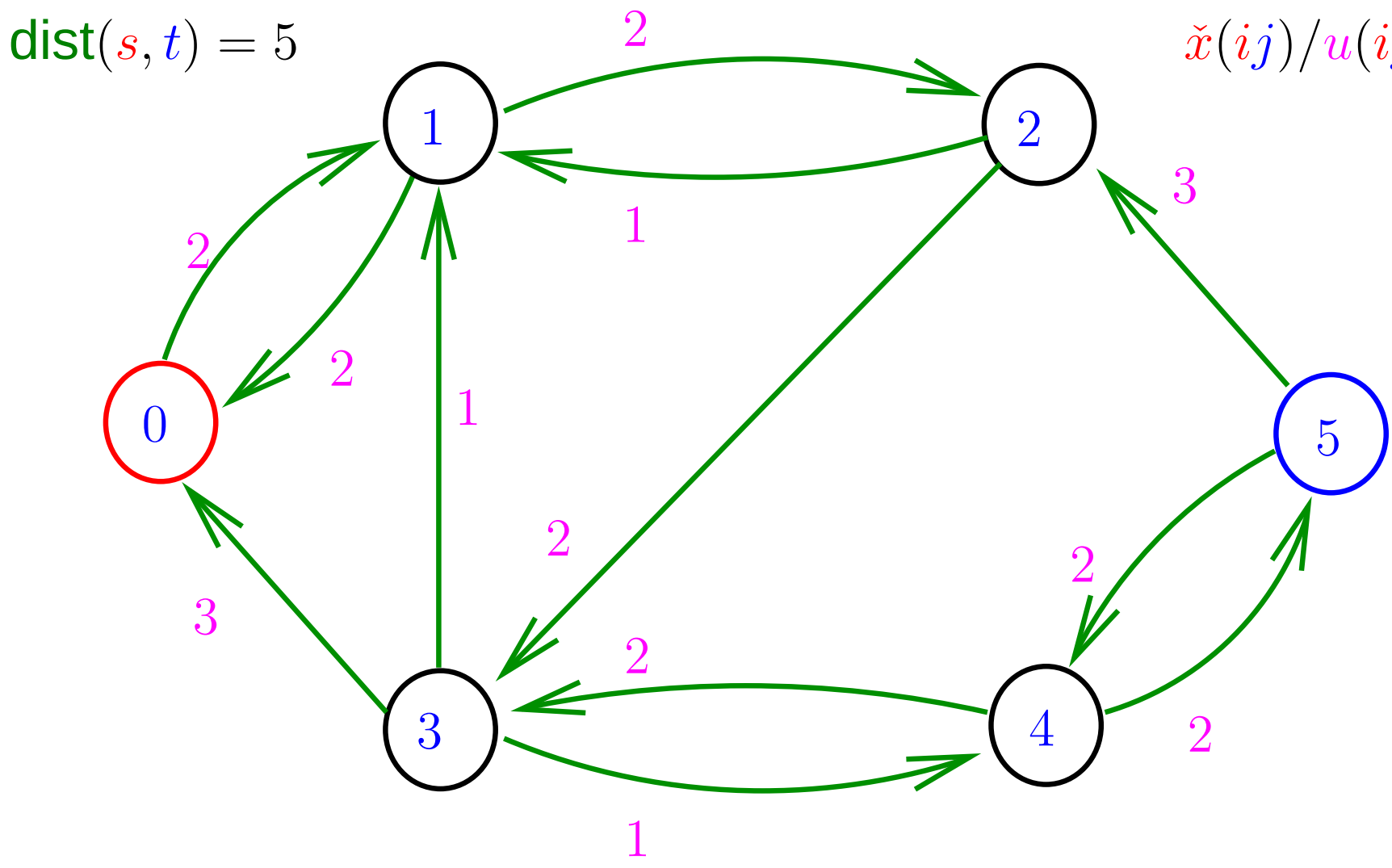


Rede 3

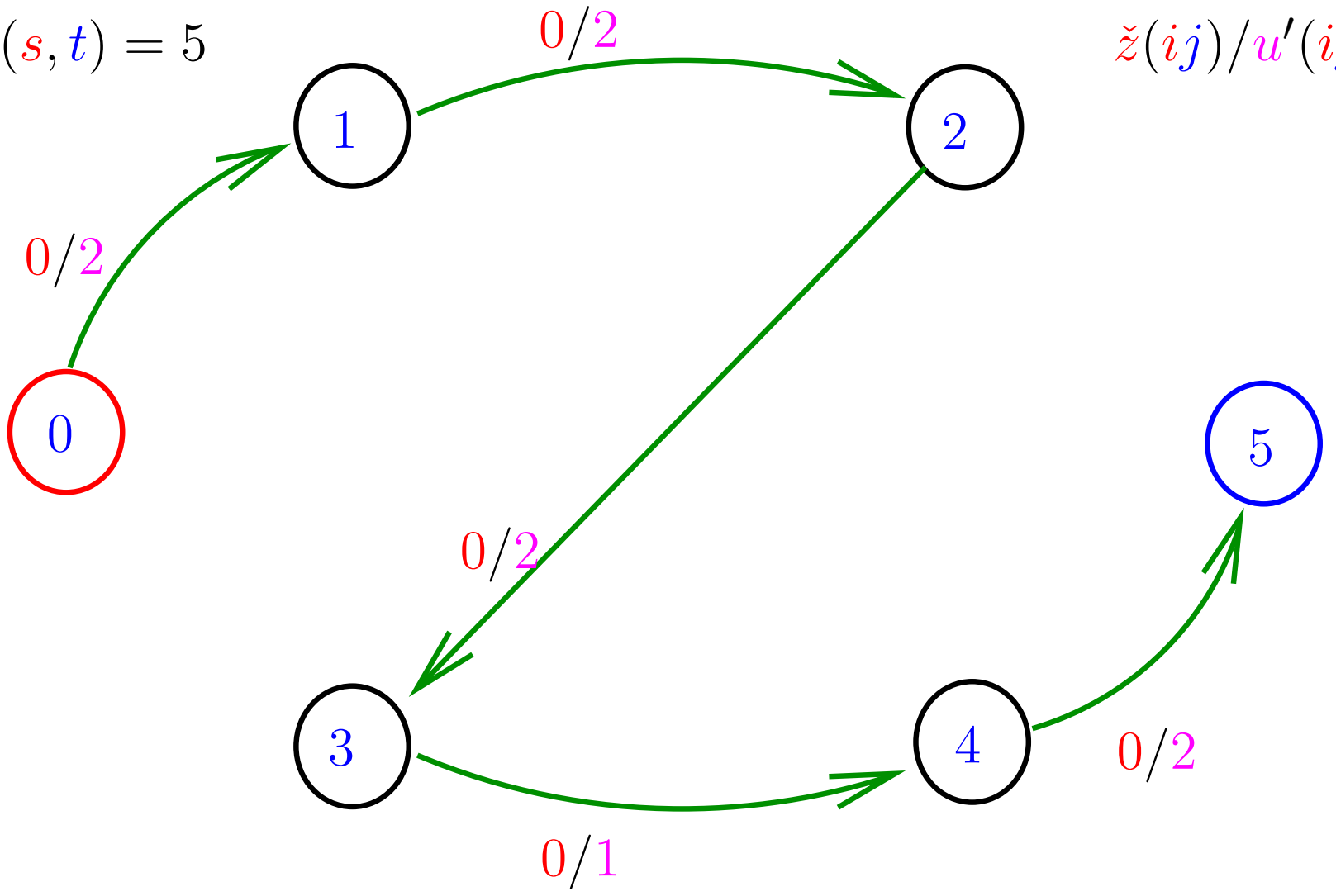


Arcos ausentes têm capacidade nula.

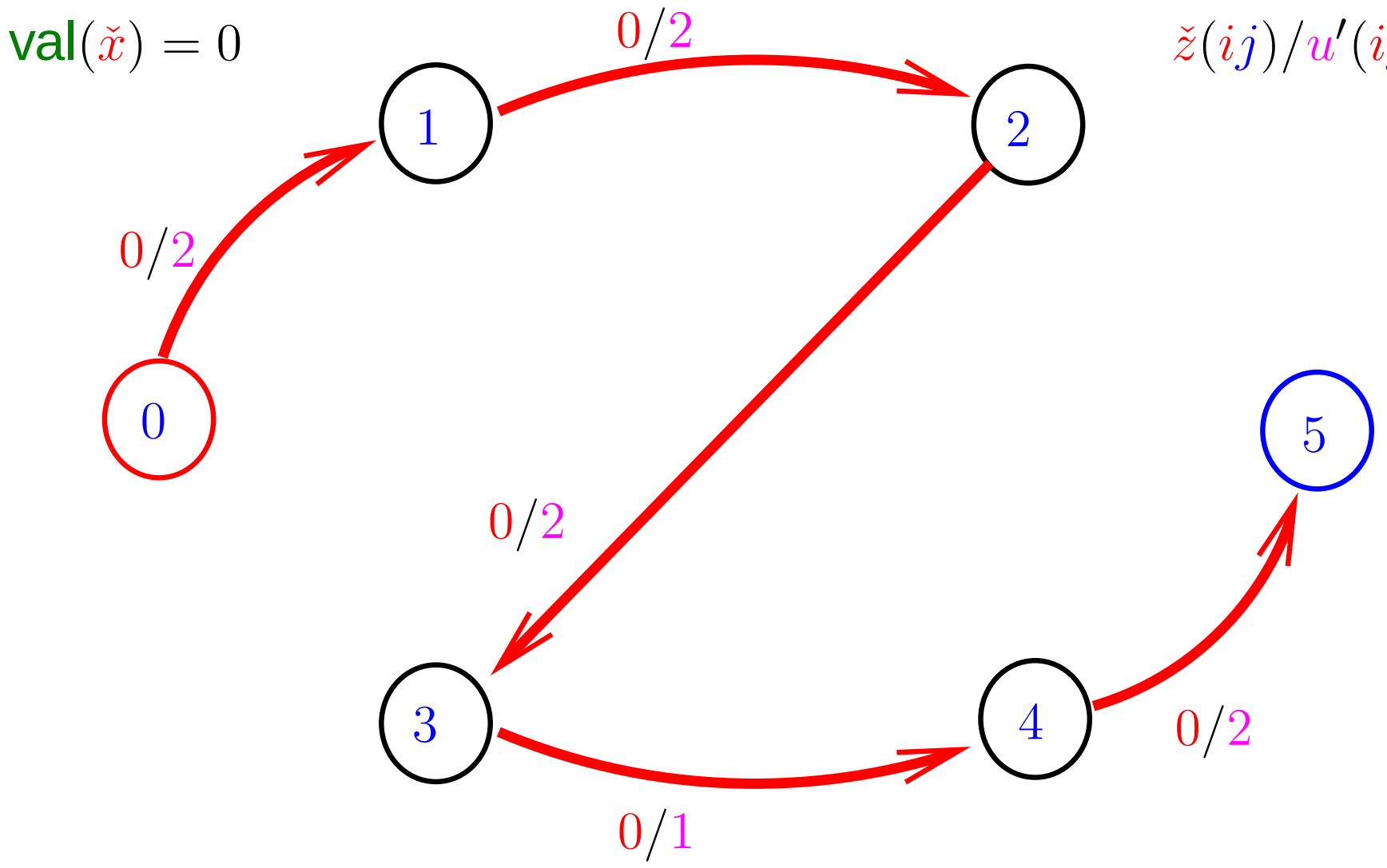
Rede residual 3



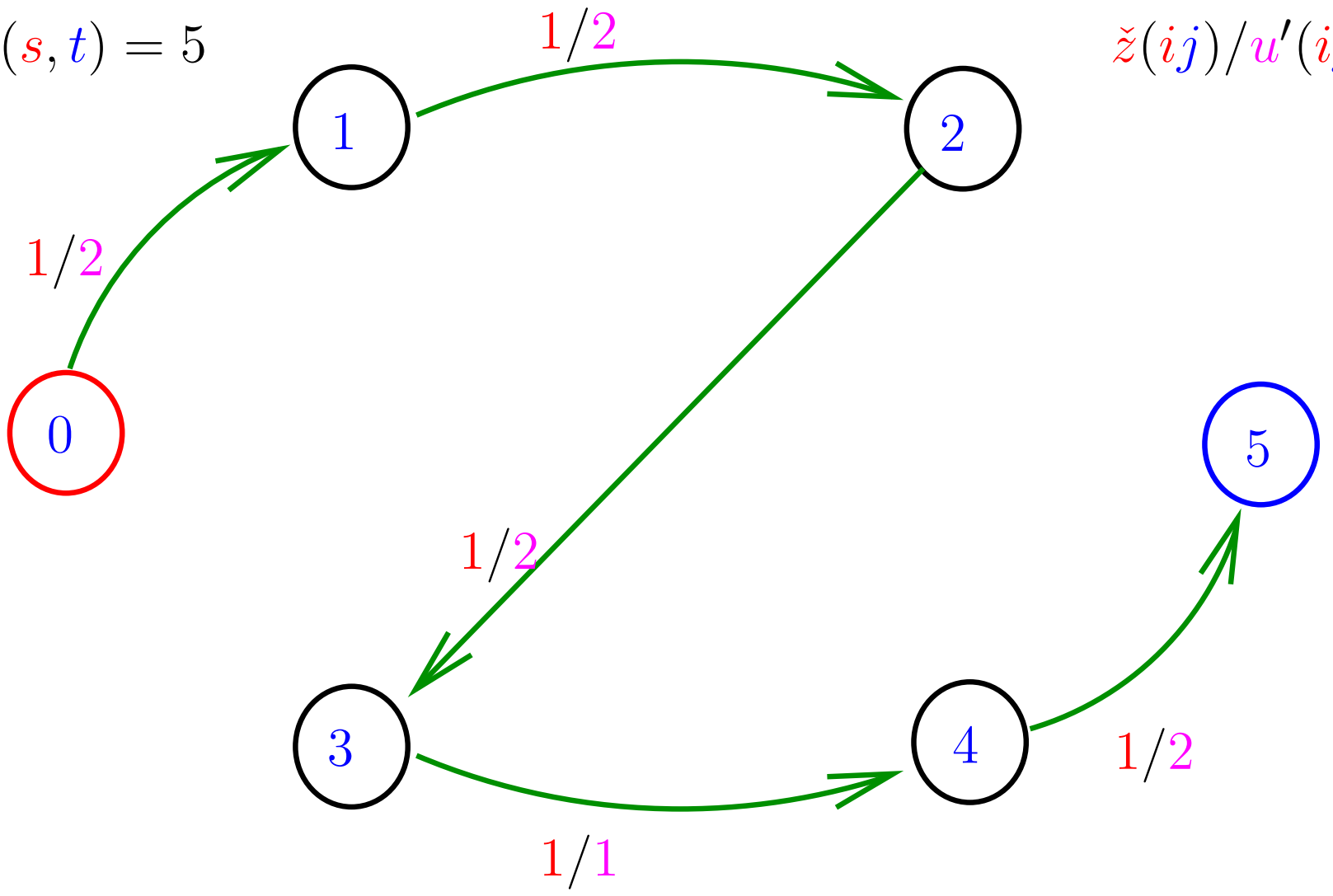
Rede em camadas 3



Caminho de incremento 5



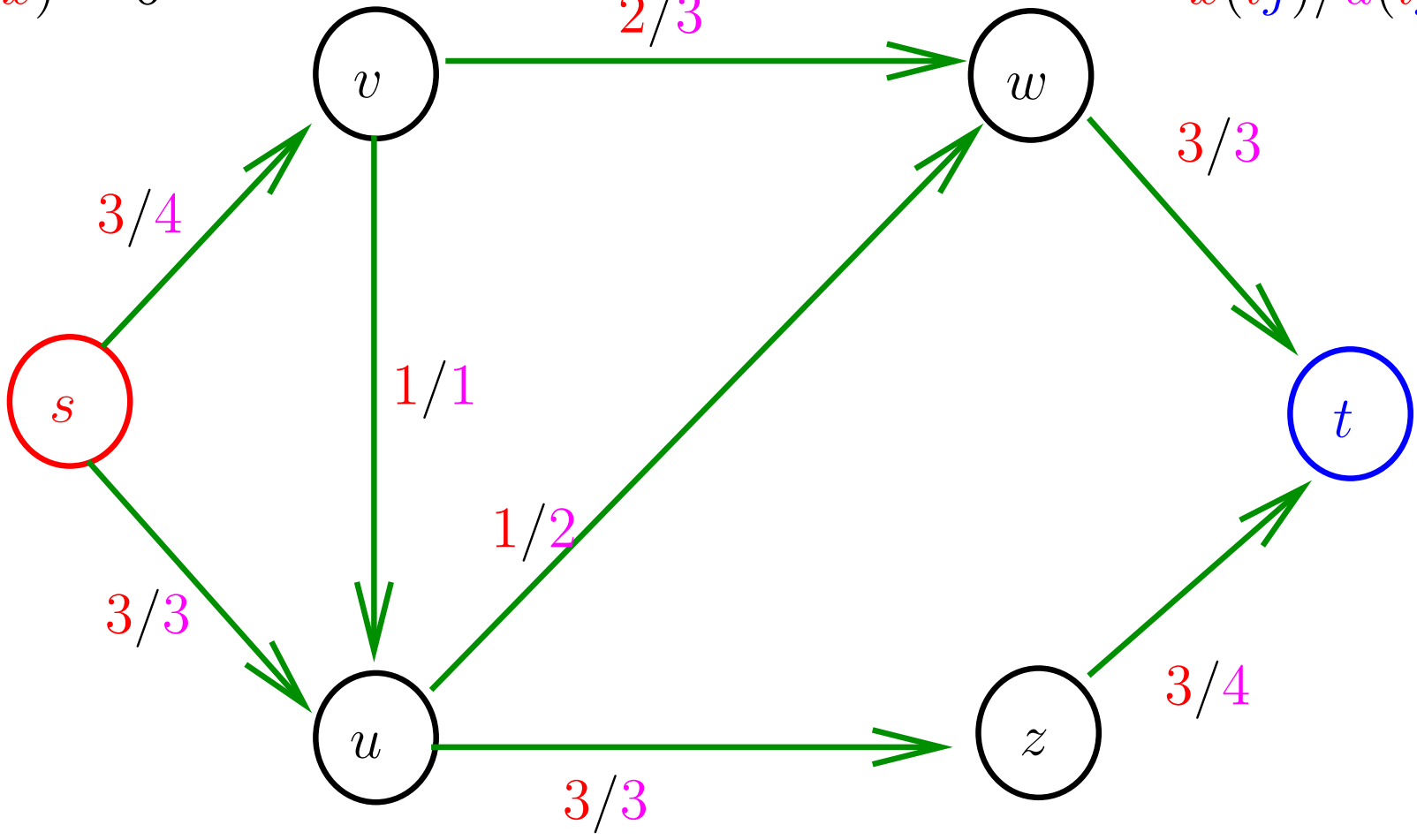
Fluxo bloqueador 3



Rede 4

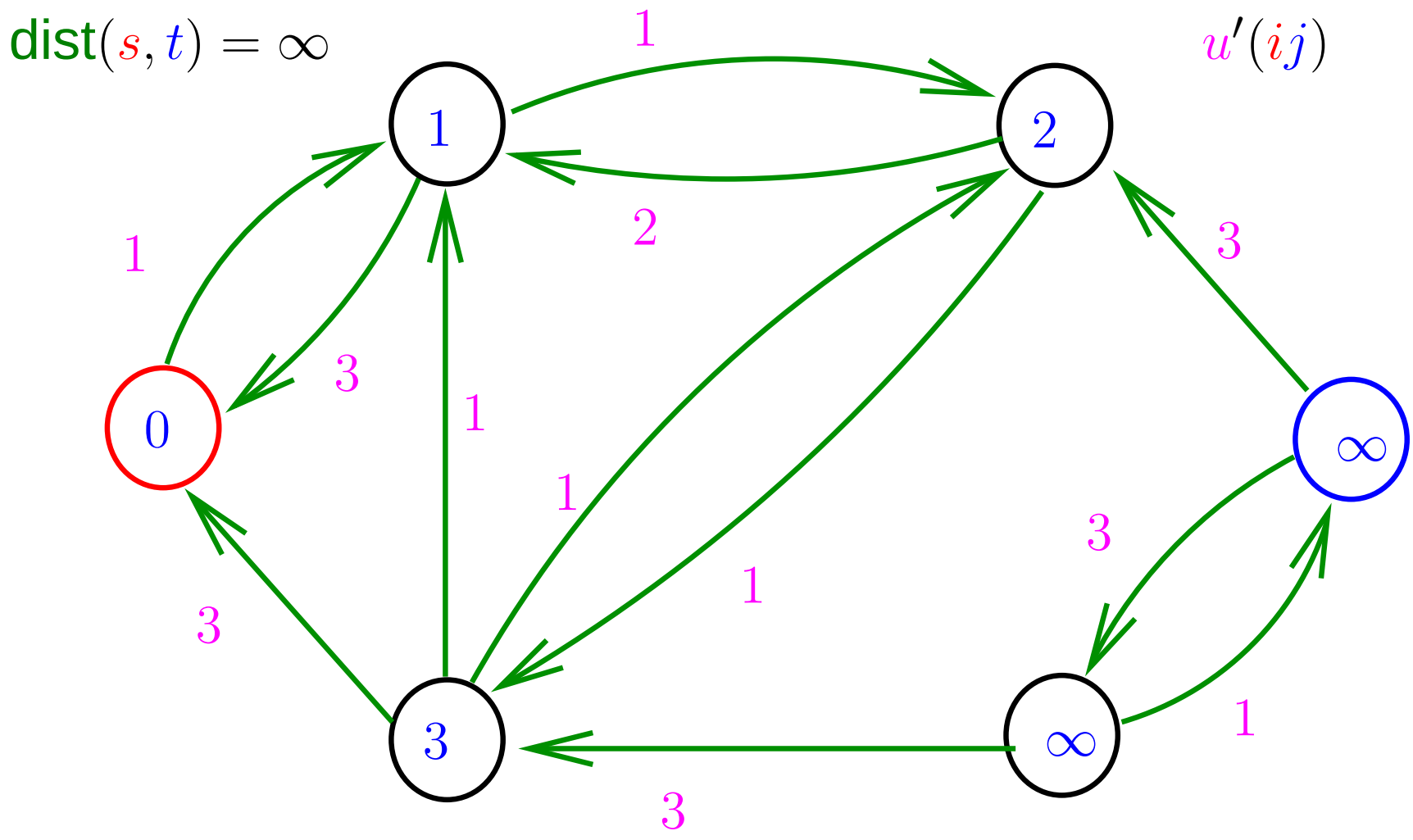
$\mathsf{val}(\check{x}) = 6$

$\check{x}(ij)/u(ij)$

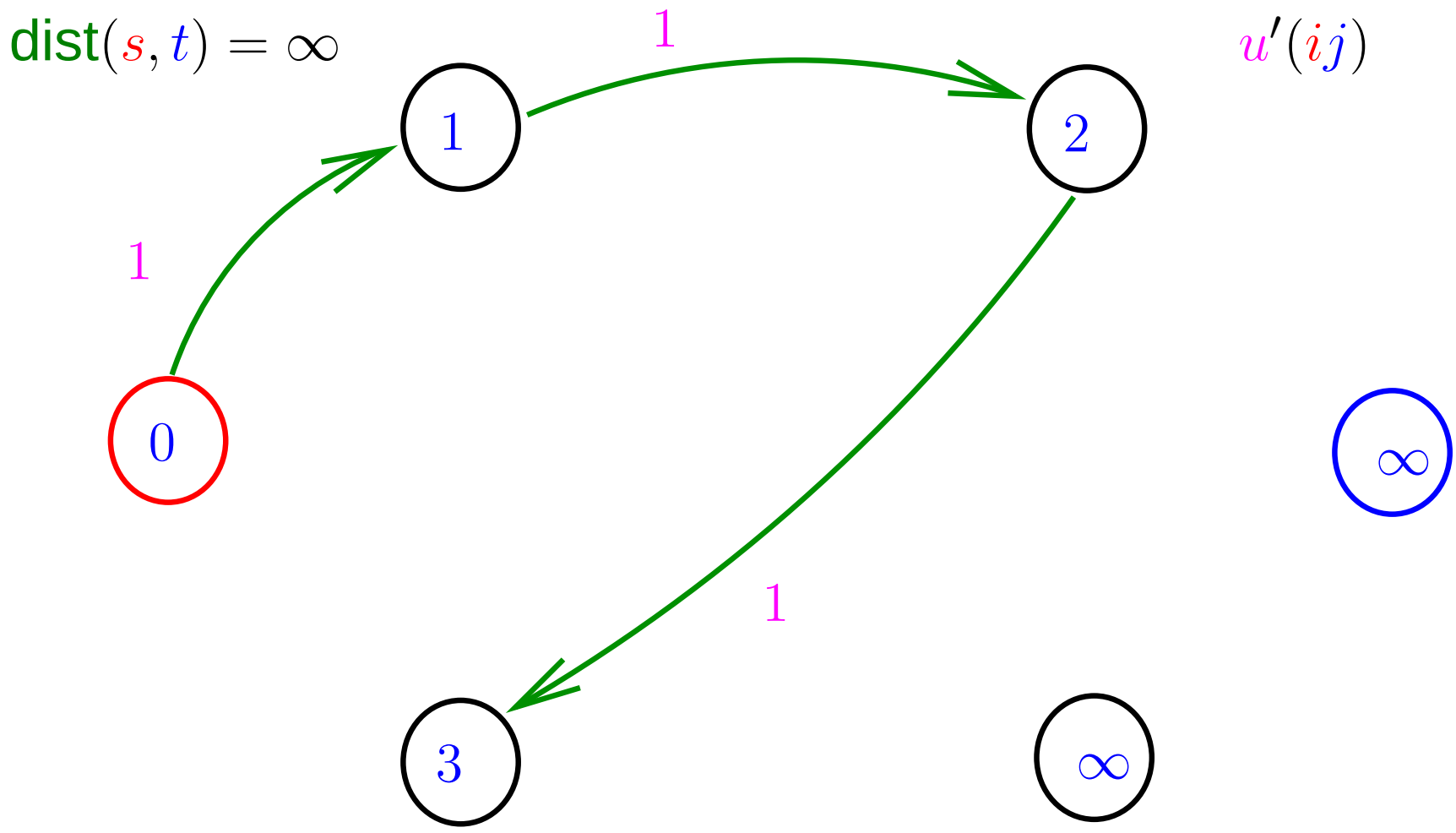


Arcos ausentes têm capacidade nula.

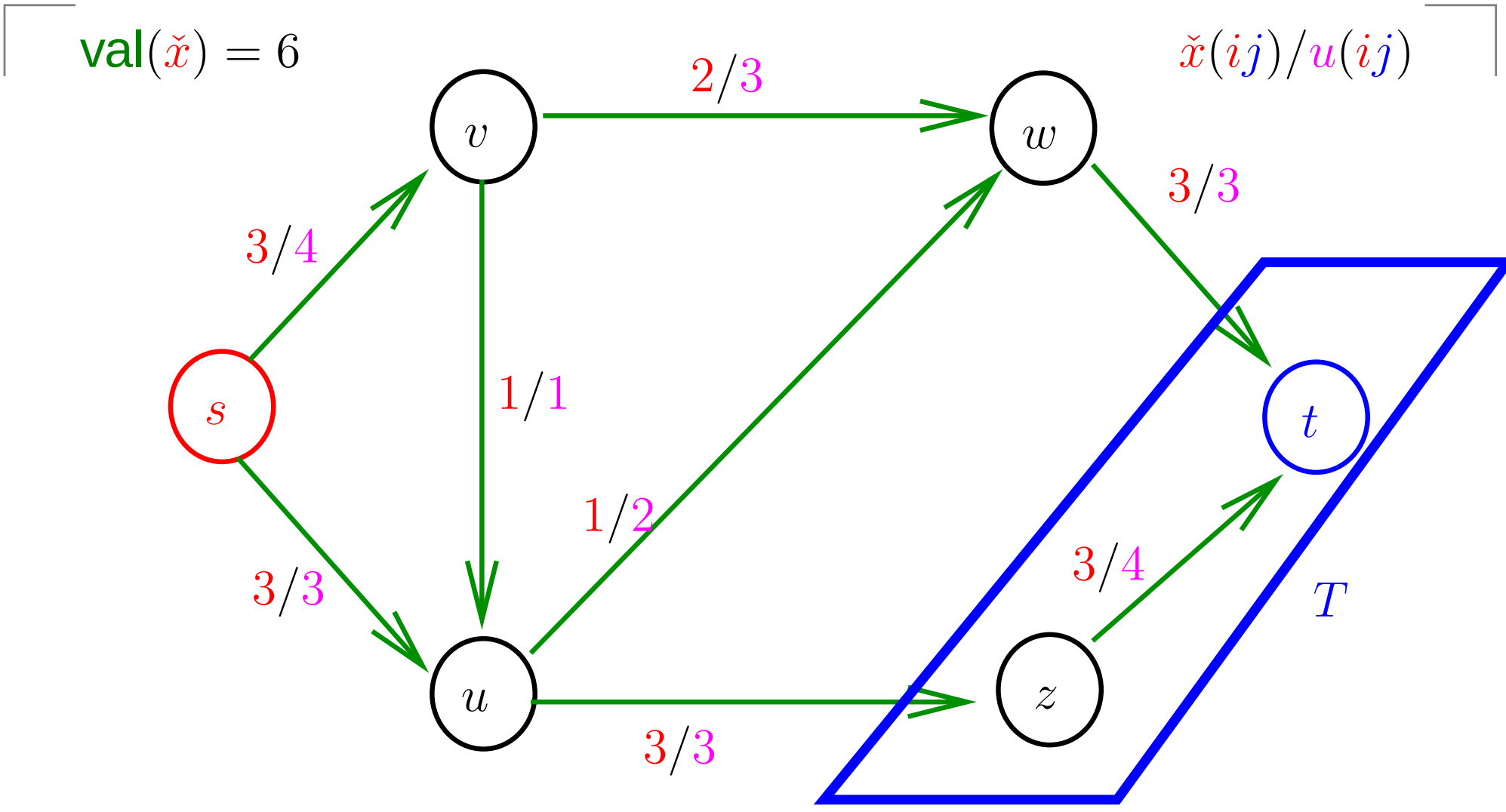
Rede residual 4



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 $(N, A_{\check{x}})$.

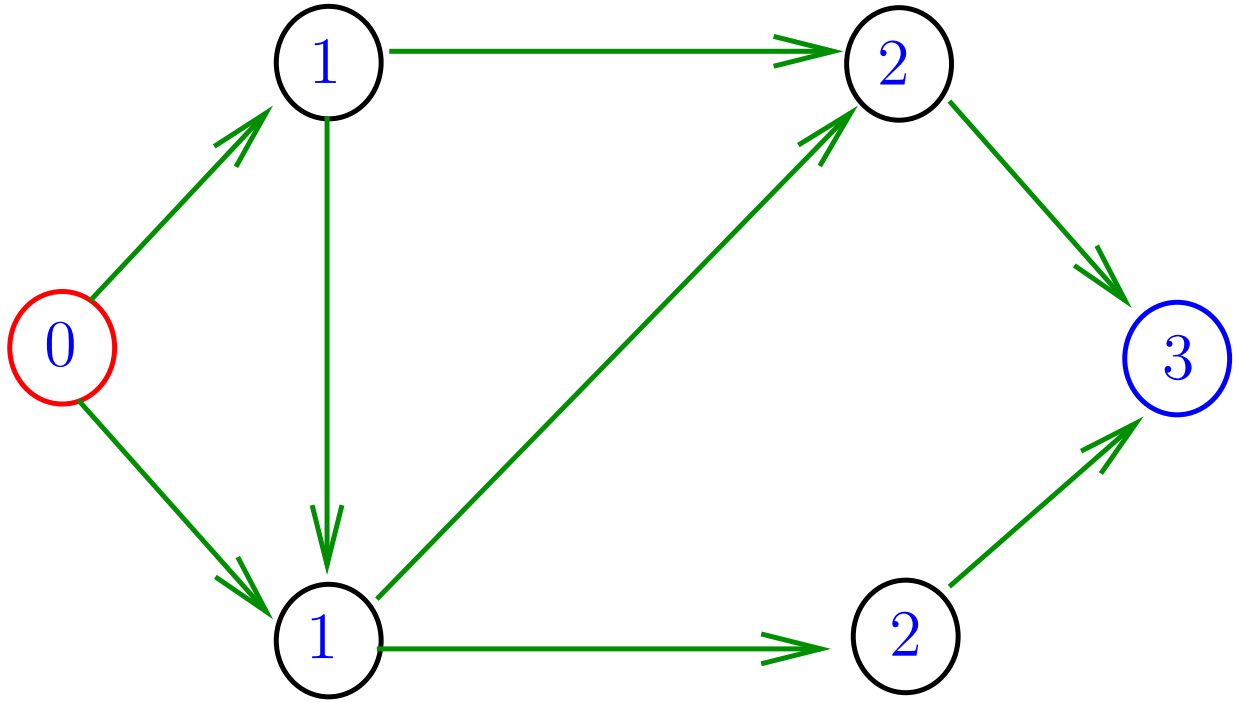
Observação: pode-se usar um 1-potencial $(*, t)$-ótimo ao invés de $(s, *)$-ótimo.

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)

```
 1  para cada i em N faça
 2        y(i) ← n
 3  y(s) ← 0
 4  L ← ⟨s⟩      ▷ L funciona como uma fila
 5  enquanto L ≠ ⟨⟩ faça
 6        retire o primeiro elemento, digamos i, de L
 7        para cada arco ij em A(i) faça      ▷ ij entra em j
 8              se y(j) > y(i) + 1      ▷ y(j) = n
 9                    então y(j) ← y(i) + 1
10                          acrescente j ao final de L
11  devolva y
```

Consumo de tempo: $O(n + m)$

Algoritmo de Dinits

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

1 $\check{x} \leftarrow 0$

2 **repita**

3 $\quad A_{\check{x}} \leftarrow \{ij : \check{x}(ij) < u(ij)\}$

4 $\quad y \leftarrow$ POTENCIAL-ÓTIMO$(N, A_{\check{x}}, t)$

5 $\quad$ **se** $y(t) - y(s) < n$ **então**

6 $\quad\quad \check{x} \leftarrow$ PASSO-BLOQUEADOR$(N, A_{\check{x}}, s, t, \check{x}, y)$

7 **até que** $y(t) - y(s) \geq n$

8 $x \leftarrow$ FLUXO$(\check{x})$

9 $T \leftarrow \{j \in N : y(j) - y(s) \geq n\}$

10 **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 núnemo 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) $y_k(j) - y_k(i) \leq 1$ 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_x̌, s, t, x̌, y)
 1   B ← ∅      ▷ armazena os nós bloqueados
 2   P ← ⟨s⟩
 3   i ← s
 4   enquanto s ∉ B faça
 5         se i = t então
 6               ⟨x̌, A_x̌⟩ ← INCREMENTE-FLUXO (x̌, P, A_x̌, u)
 7               P ← ⟨s⟩
 8               i ← s
 9         se algum ij em A_x̌ é justo e j ∉ B
10               então AVANCE(ij)
11               senão RETROCEDA(i)
12   devolva x̌
```

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 $B \leftarrow B \cup \{i\}$ ▷ 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 "δ unidades de fluxo através de P" e os arcos que ainda não estão saturados.

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

1 $\quad \delta \leftarrow \min\{u(ij) - \check{x}(ij) : ij$ é arco de $P\}$
2 $\quad$ **para cada** arco ij em P **faça**
3 $\qquad \check{x}(ij) \leftarrow \check{x}(ij) + \delta$
4 $\qquad \check{x}(ji) \leftarrow \check{x}(ji) - \delta$
5 $\qquad$ **se** $\check{x}(ij) = u(ij)$
6 $\qquad\quad$ **então** $A_{\check{x}} \leftarrow A_{\check{x}} - \{ij\}$
5 $\quad$ **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 $\mathrm{O}(n^2 m)$.

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 $\mathrm{O}(n+m)$?