Universidade do Estado do Amazonas
Escola Superior de Tecnologia - EST
Núcleo de Computação

Algoritmos e Estruturas de Dados II

Caminhos mínimos em grafos

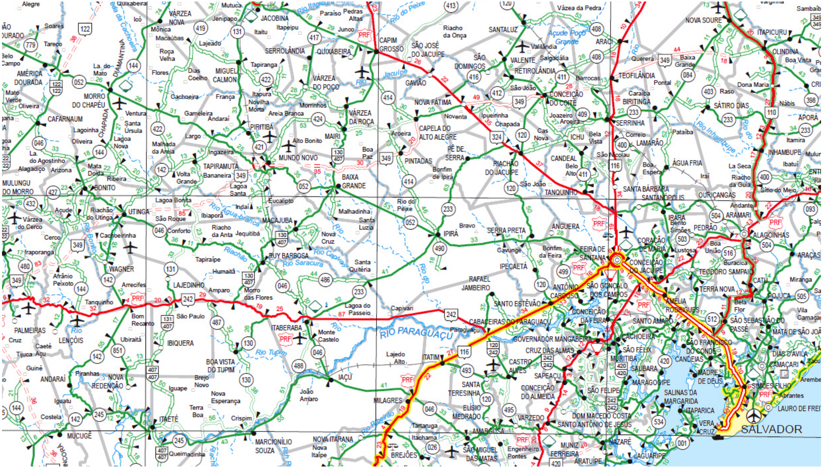
Prof. Flávio José M. Coelho

fcoelho@uea.edu.br

Objetivos

Entender o algoritmo de **Dijkstra** para caminhos mínimos em grafos.

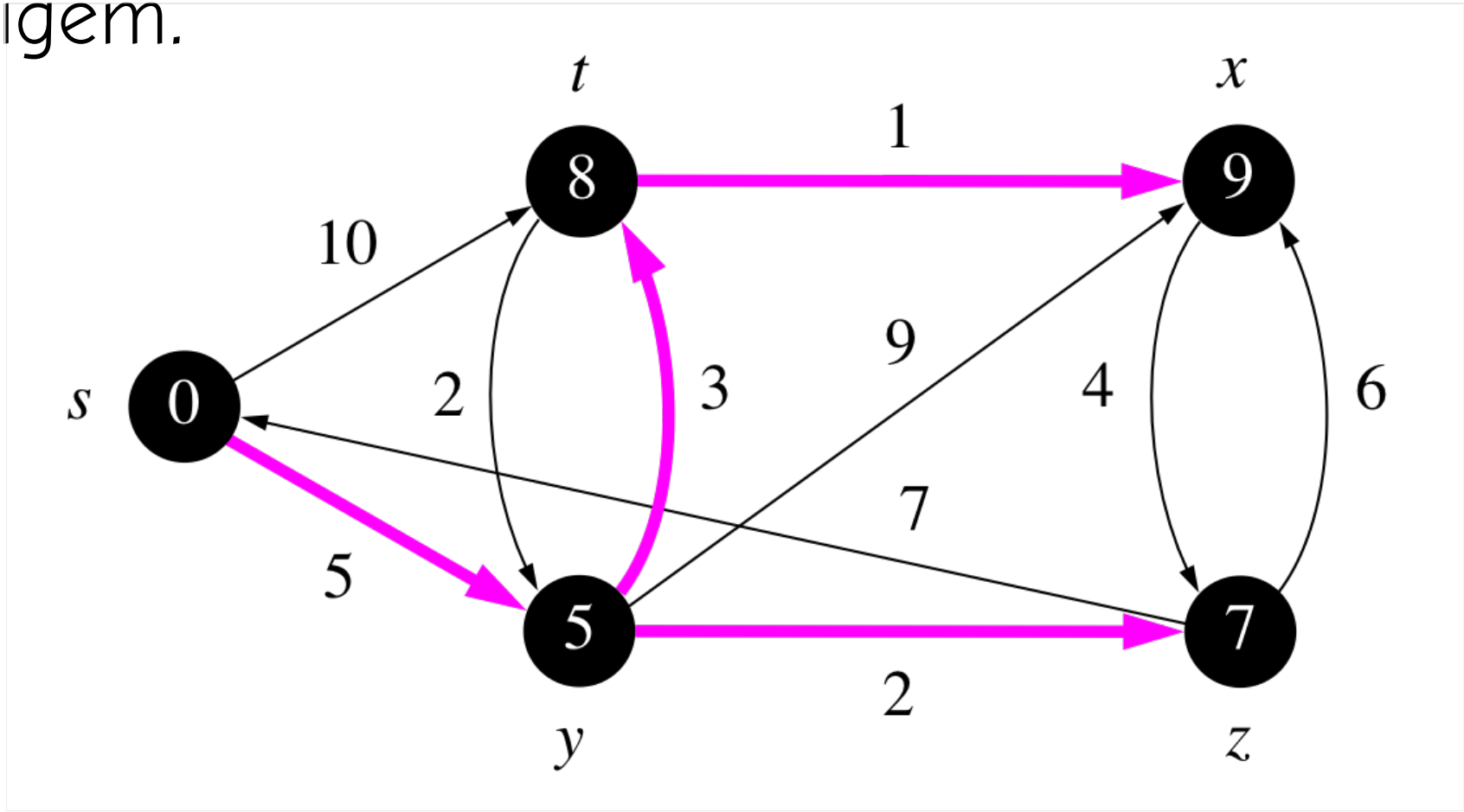
E se o grafo for ponderado?



E se o grafo for ponderado?

Cientista da Computação holandês, **Edsger W. Dijkstra** (1930-2002). Prêmio Turing (1972).

O **algoritmo de Dijkstra** descobre todos os **caminhos mínimos** em um grafo ponderado (com pesos positivos) a partir de uma origem.



Elementos do algoritmo

Elementos do algoritmo

- Algoritmo constrói um árvore geradora $A = \{(v, v.\pi) : v \in V - \{s\}\}$ do grafo $G = (V, E)$.

Elementos do algoritmo

- Algoritmo constrói um árvore geradora $A = \{(v, v.\pi) : v \in V - \{s\}\}$ do grafo $G = (V, E)$.

- $(v, v.\pi)$ aresta de A com um vértice v e seu antecessor $v.\pi$.

Elementos do algoritmo

- Algoritmo constrói um árvore geradora $A = \{(v, v.\pi) : v \in V - \{s\}\}$ do grafo $G = (V, E)$.
- $(v, v.\pi)$ aresta de A com um vértice v e seu antecessor $v.\pi$.
- s é o vértice-origem.

Elementos do algoritmo

- Uma **fila de prioridades mínimas** Q mantém os vértices ainda não processados.

Elementos do algoritmo

- Uma **fila de prioridades mínimas** Q mantém os vértices ainda não processados.
- Para cada vértice v em Q, a chave $v.d$ guarda a distância estimada (soma dos pesos) da origem s até v.

INICIALIZA(G, s)

1 **para cada** $v \in G.V$

2 $\quad v.d = \infty$

3 $\quad v.\pi = \text{NIL}$

4 $s.d = 0$

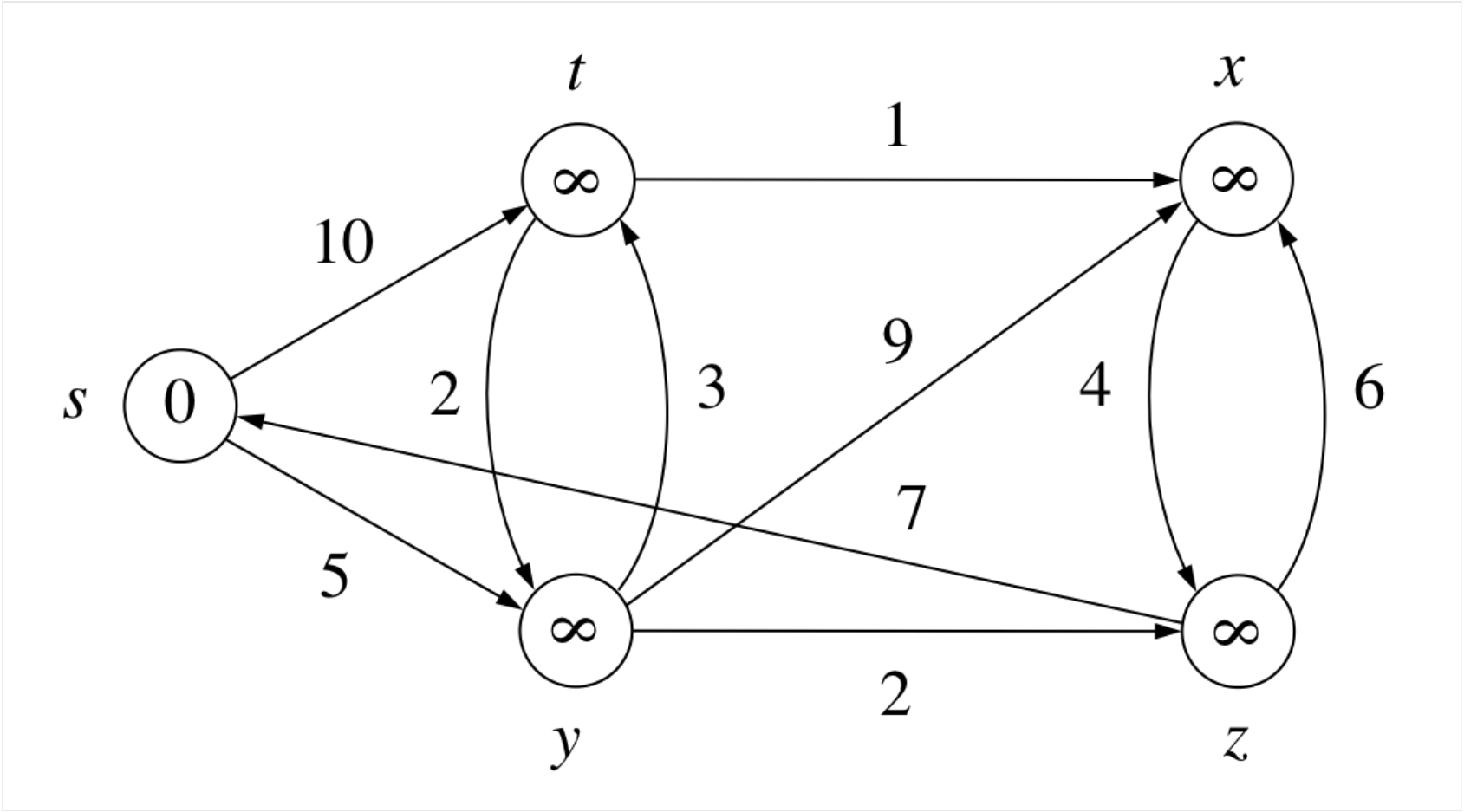
RELAXA(u, v, w)

1 **se** $v.d > u.d + w(u, v)$

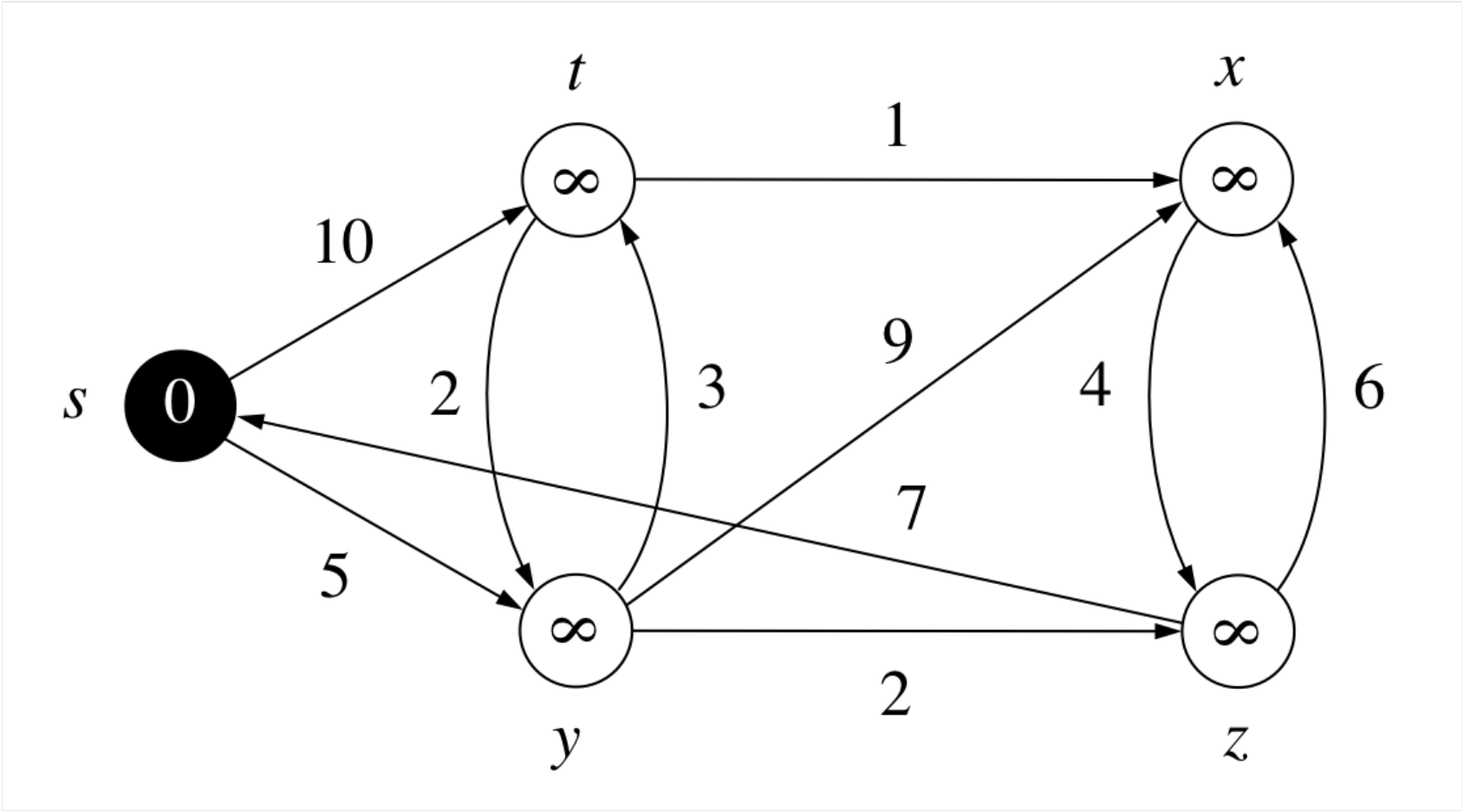
2 $\quad v.d = u.d + w(u, v)$

3 $\quad v.\pi = u$

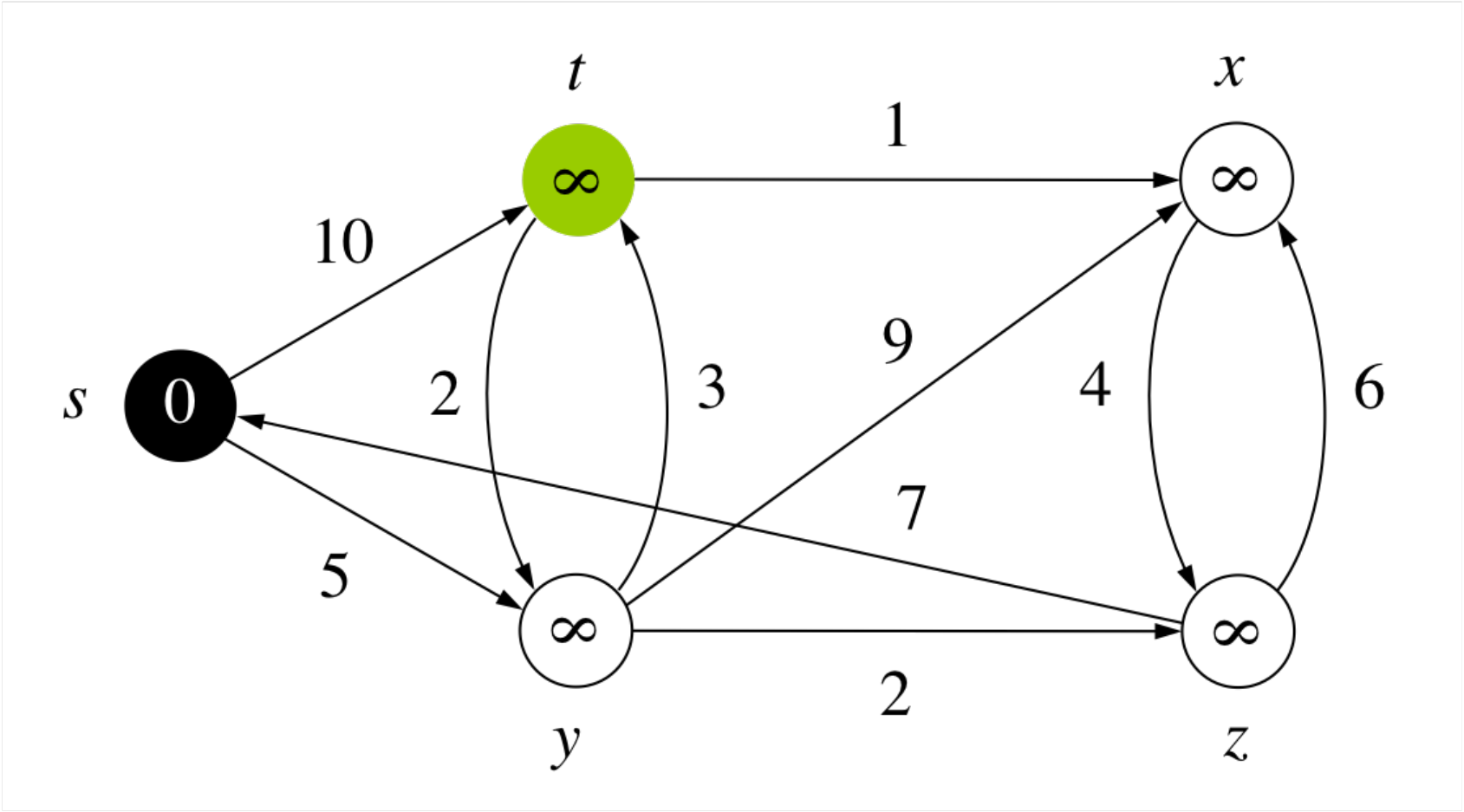
```
DIJKSTRA(G, w, s)
1   INICIALIZA(G, s)
2   S = ∅
3   Q = G.V
4   enquanto Q ≠ ∅
5       u = EXTRACT-MIN(Q)
6       S = S ∪ {u}
7       para cada v ∈ G.Adj[u]
8           RELAXA(u, v, w)
```



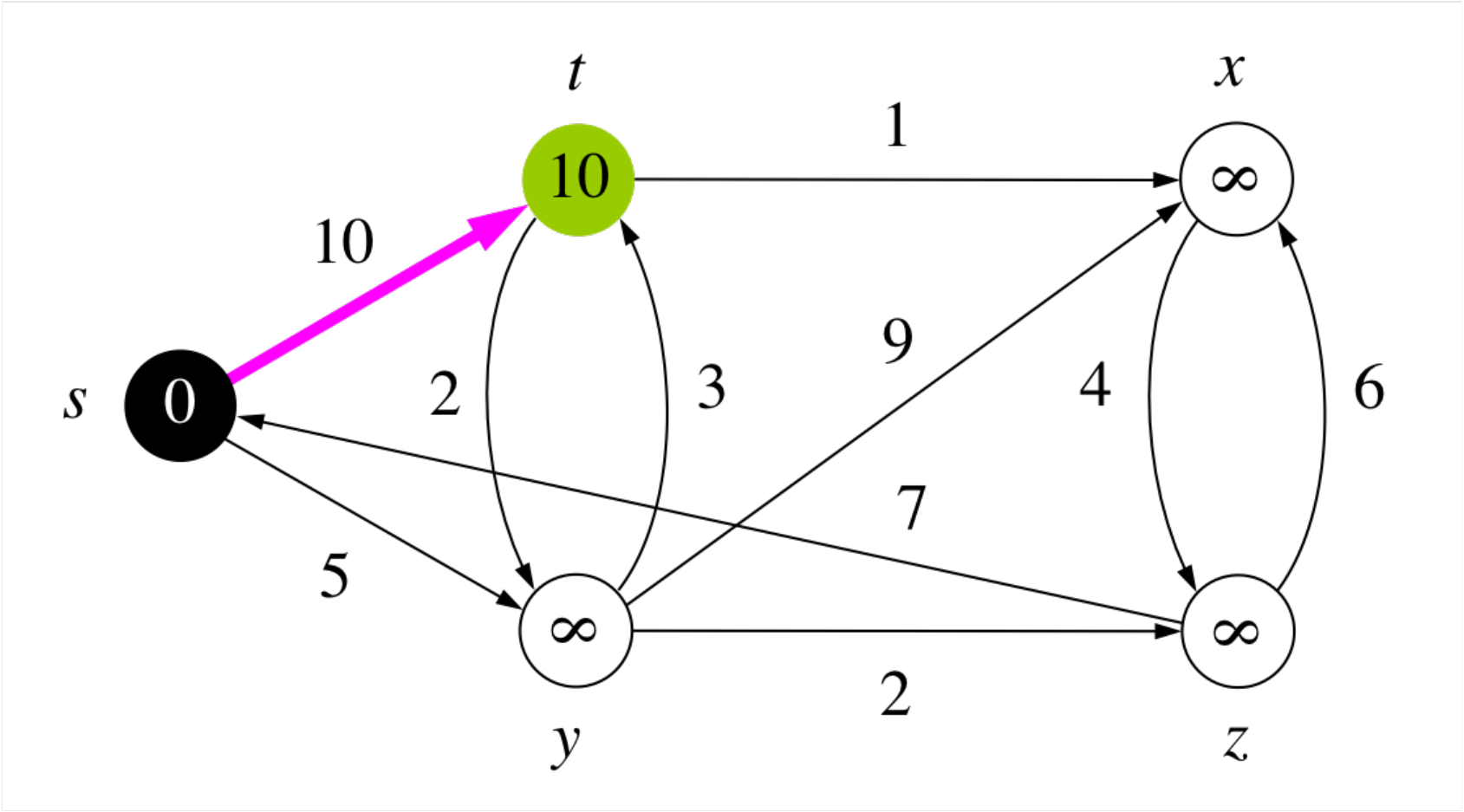
$v.d$ iniciados com ∞, exceto o vértice-origem $s.d = 0$.



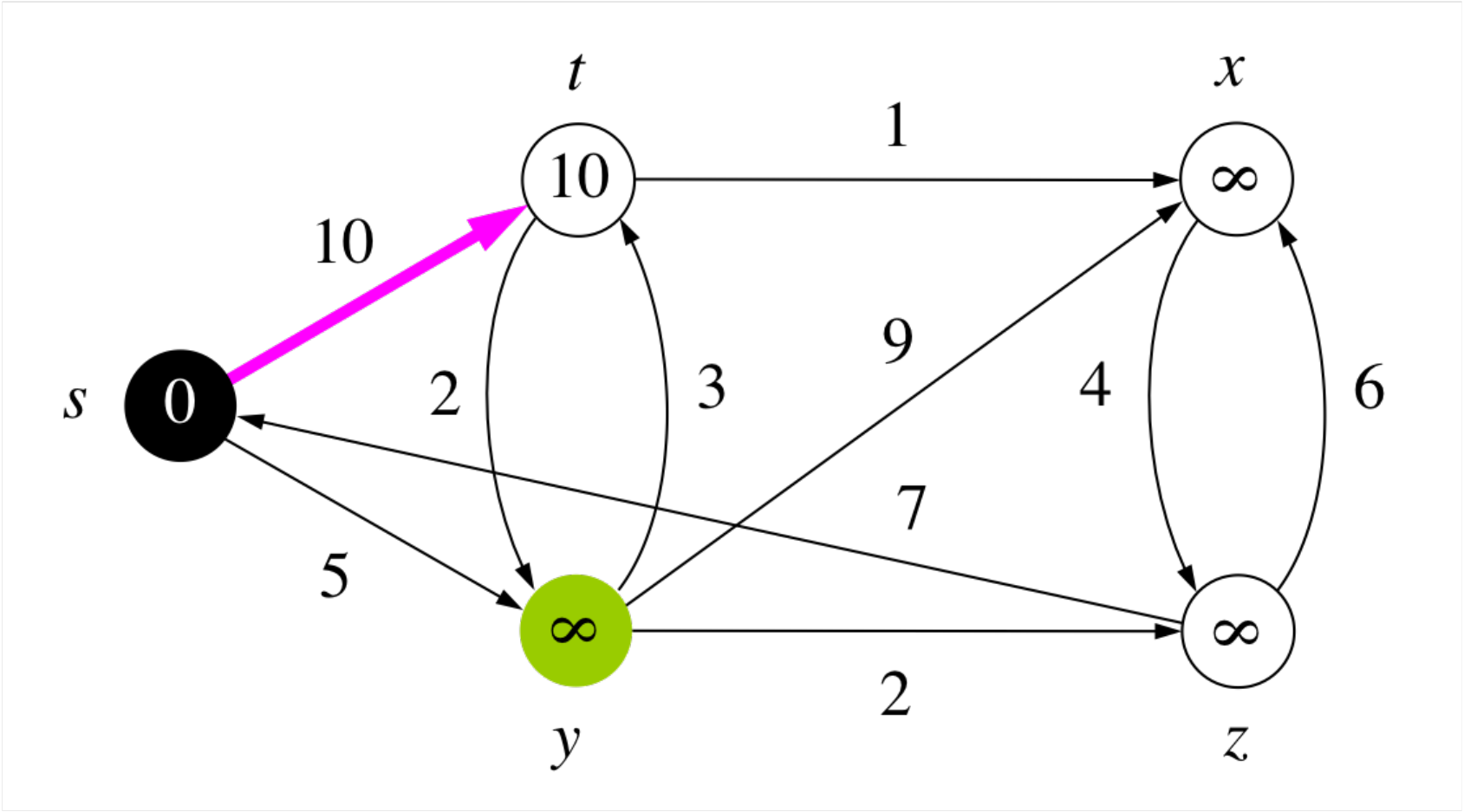
Origem s sai da fila.



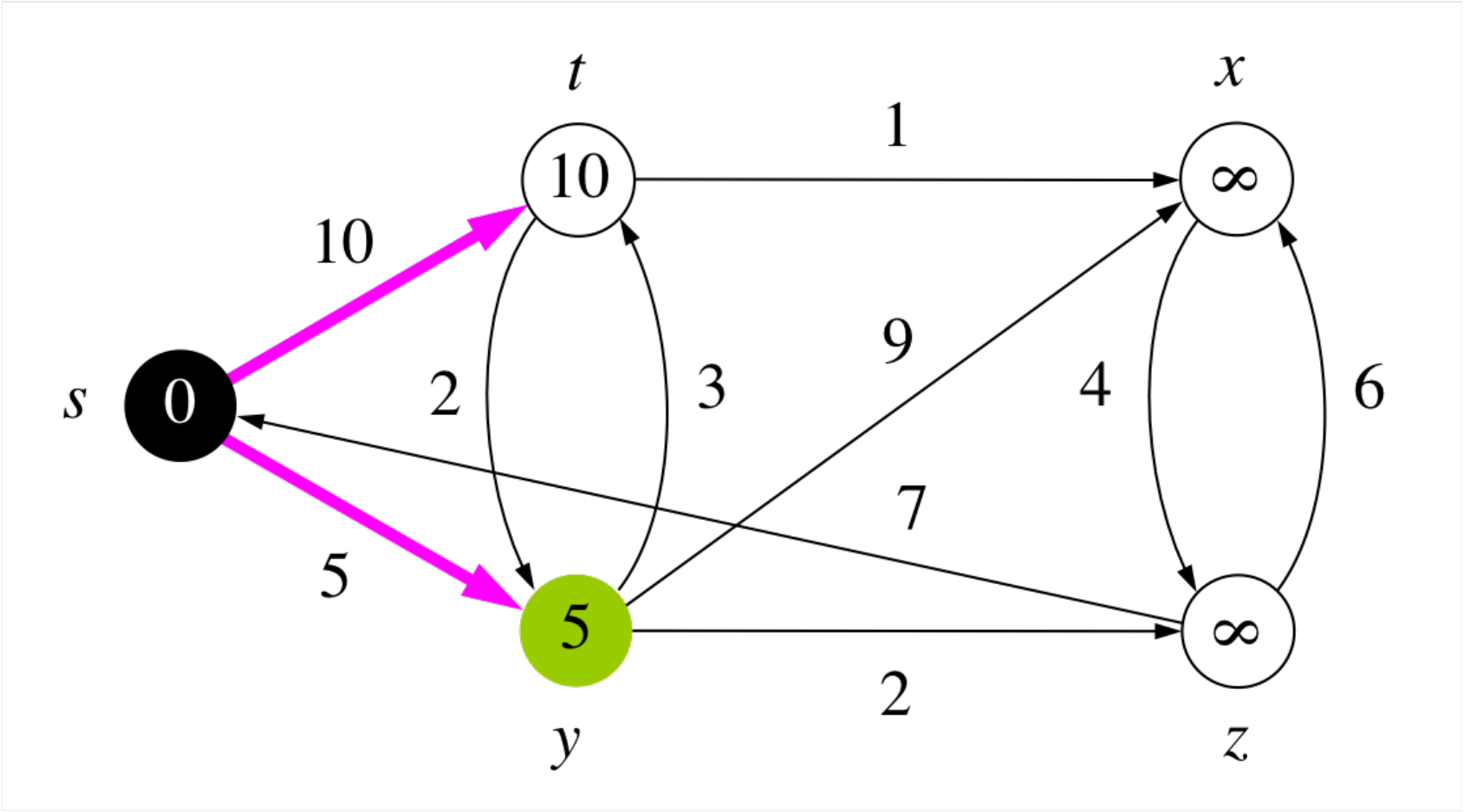
Adjacentes de s são relaxados.



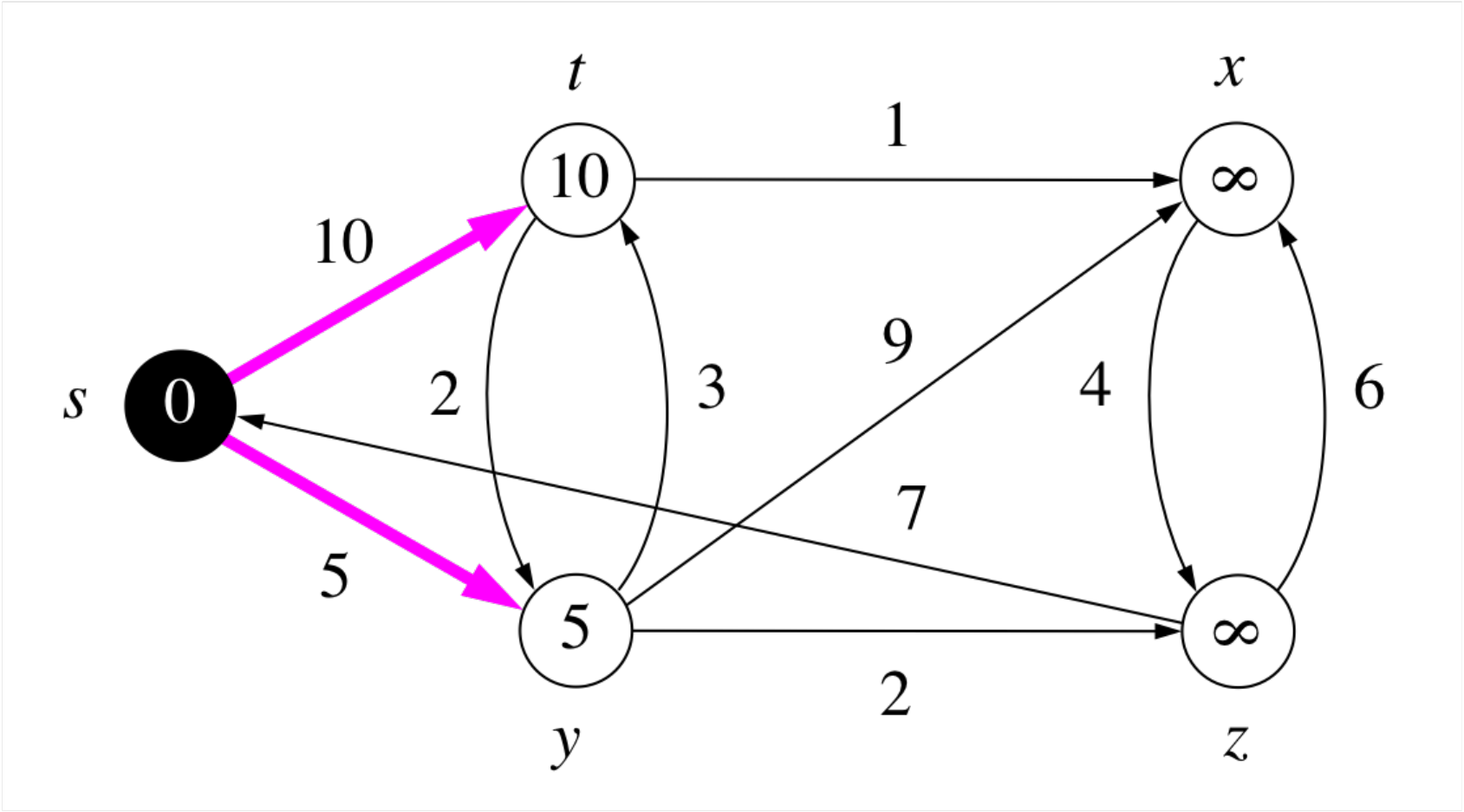
Adjacentes de s são relaxados.



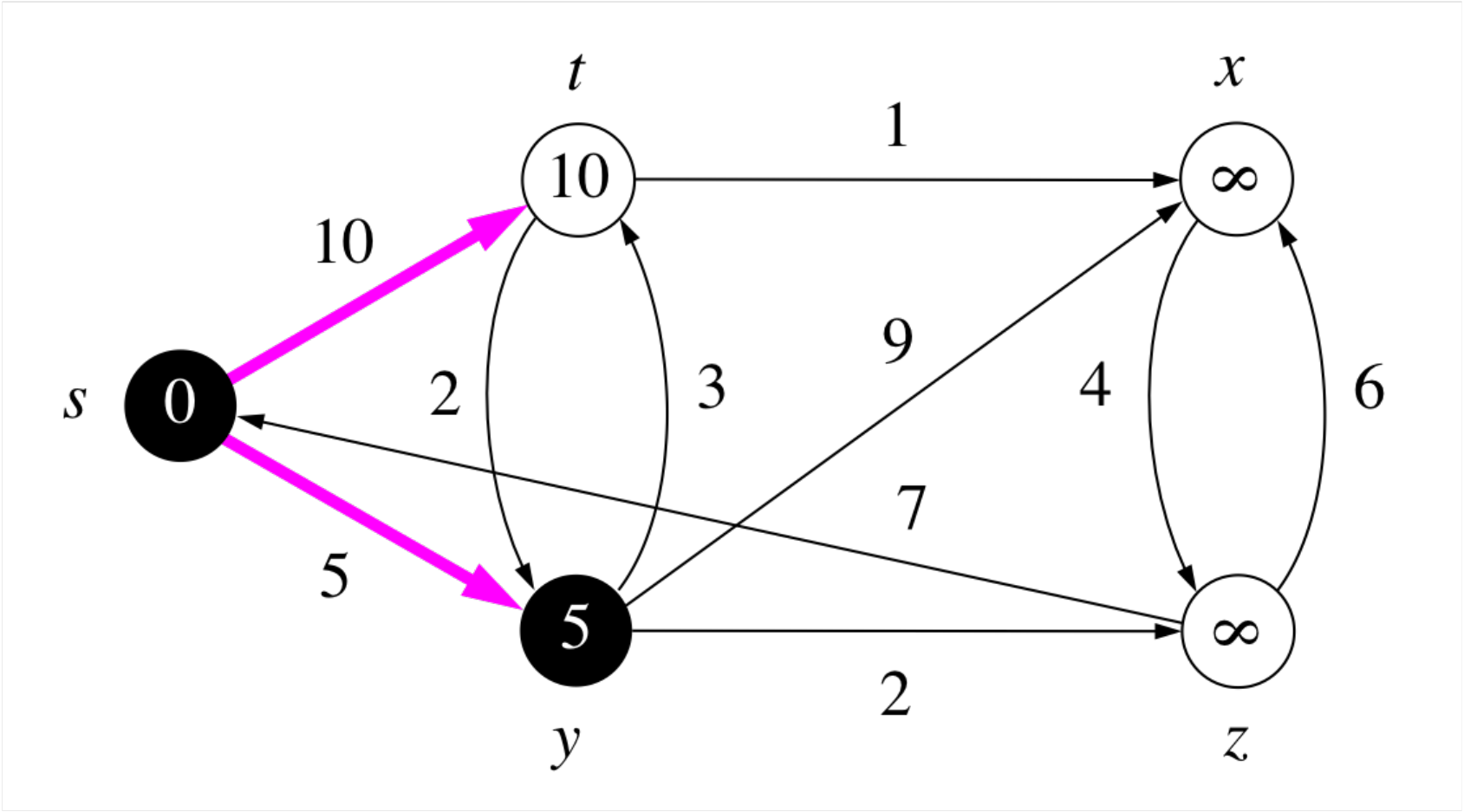
Adjacentes de s são relaxados.



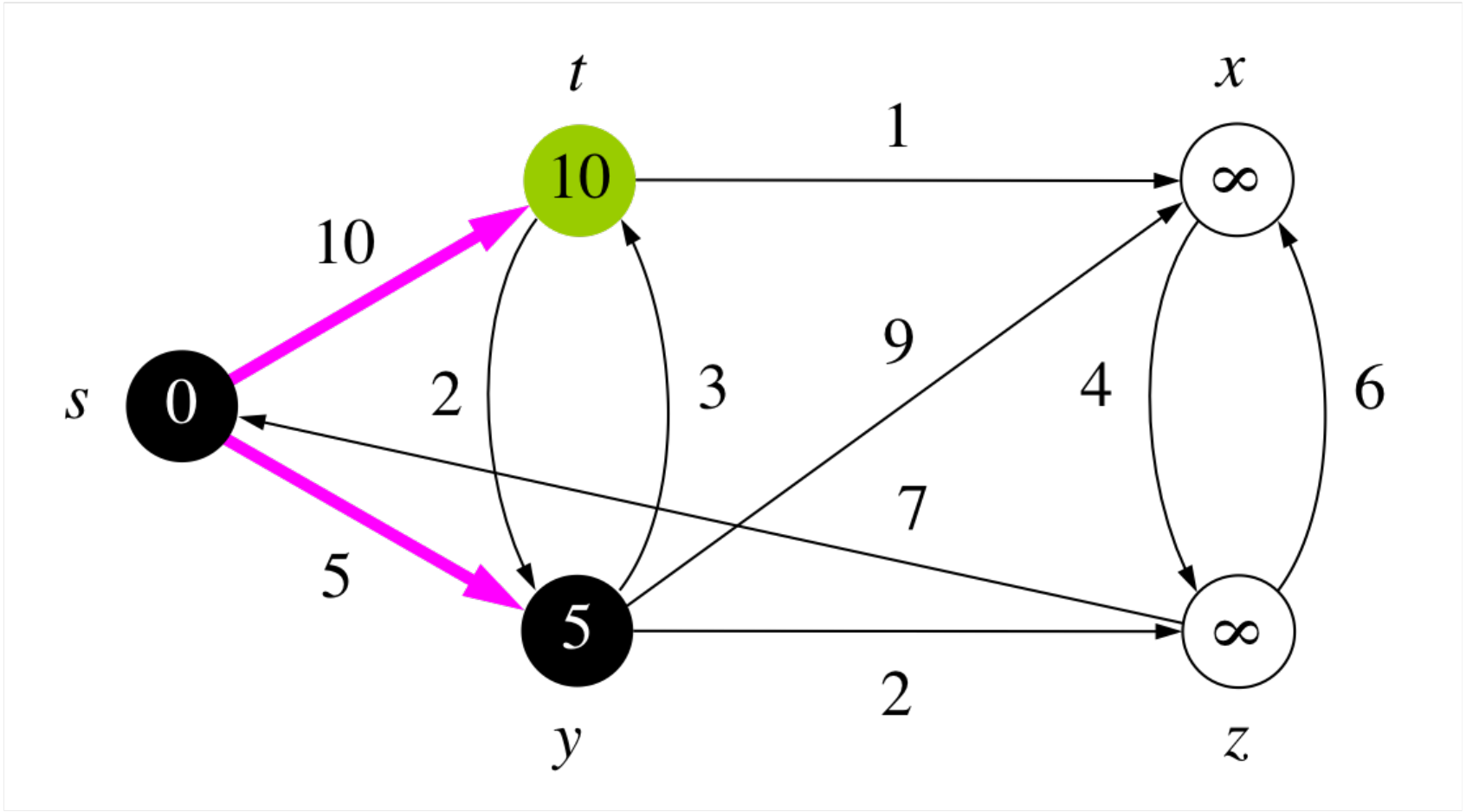
Adjacentes de s são relaxados.



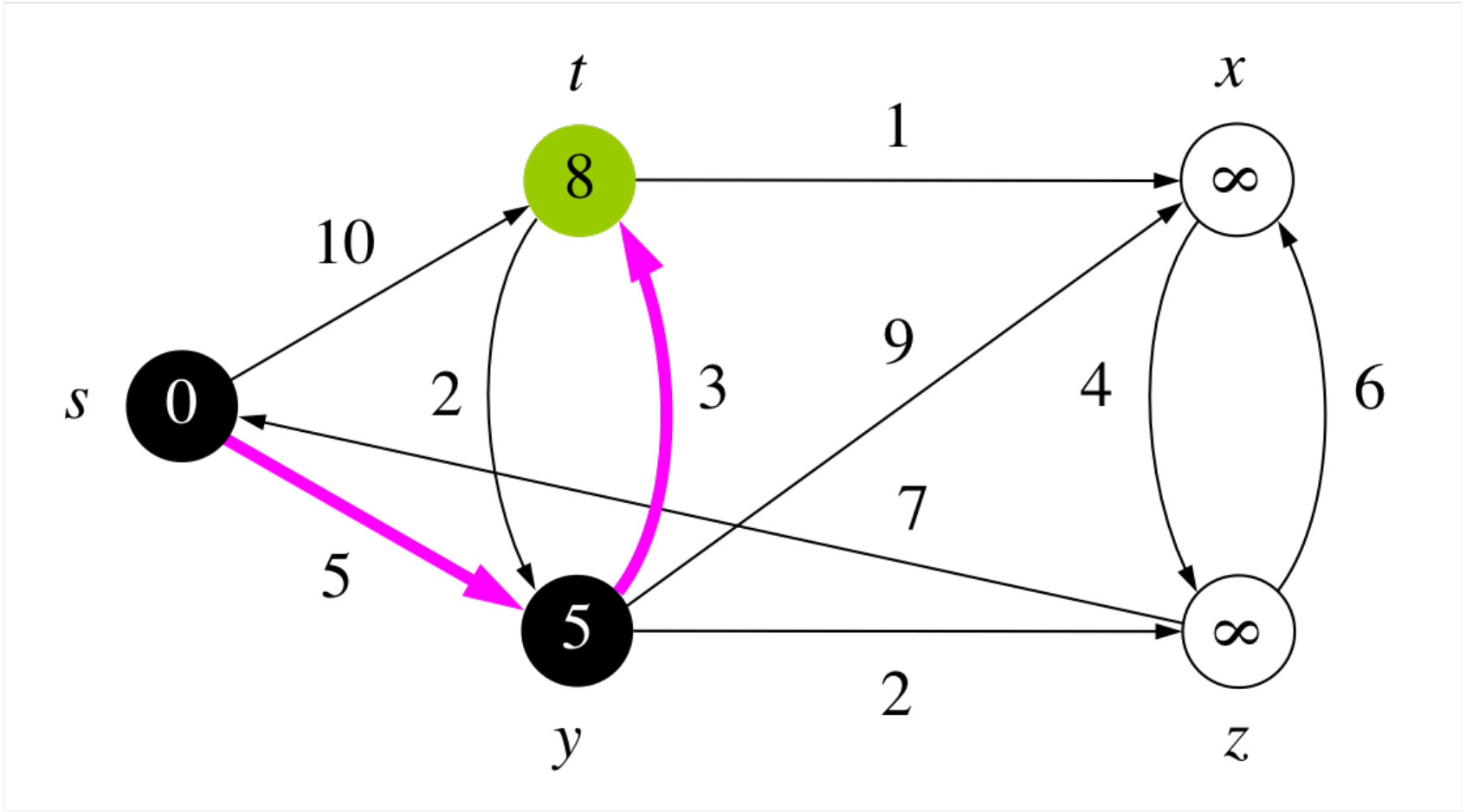
Adjacentes de s são relaxados.



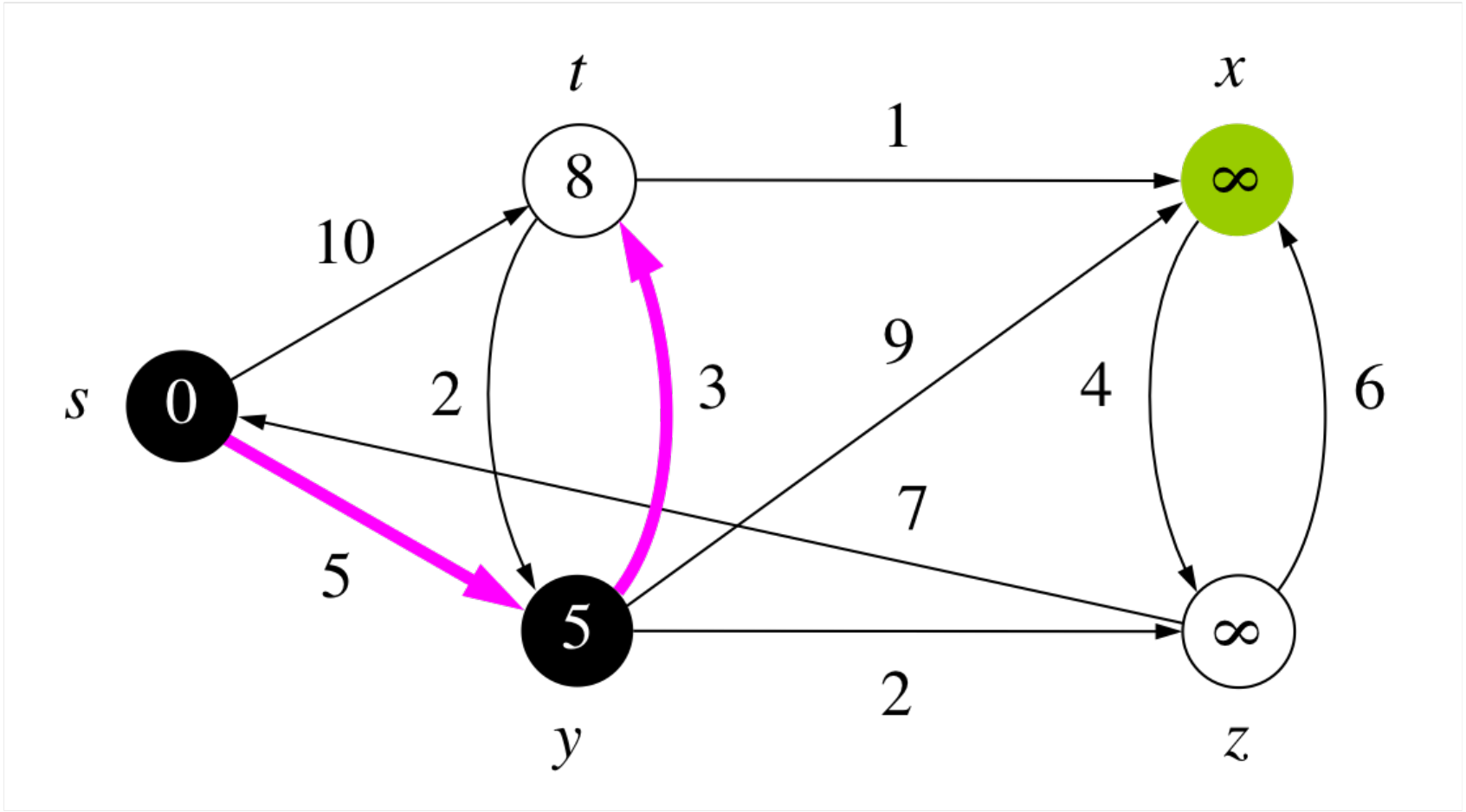
Vértice y sai da fila.



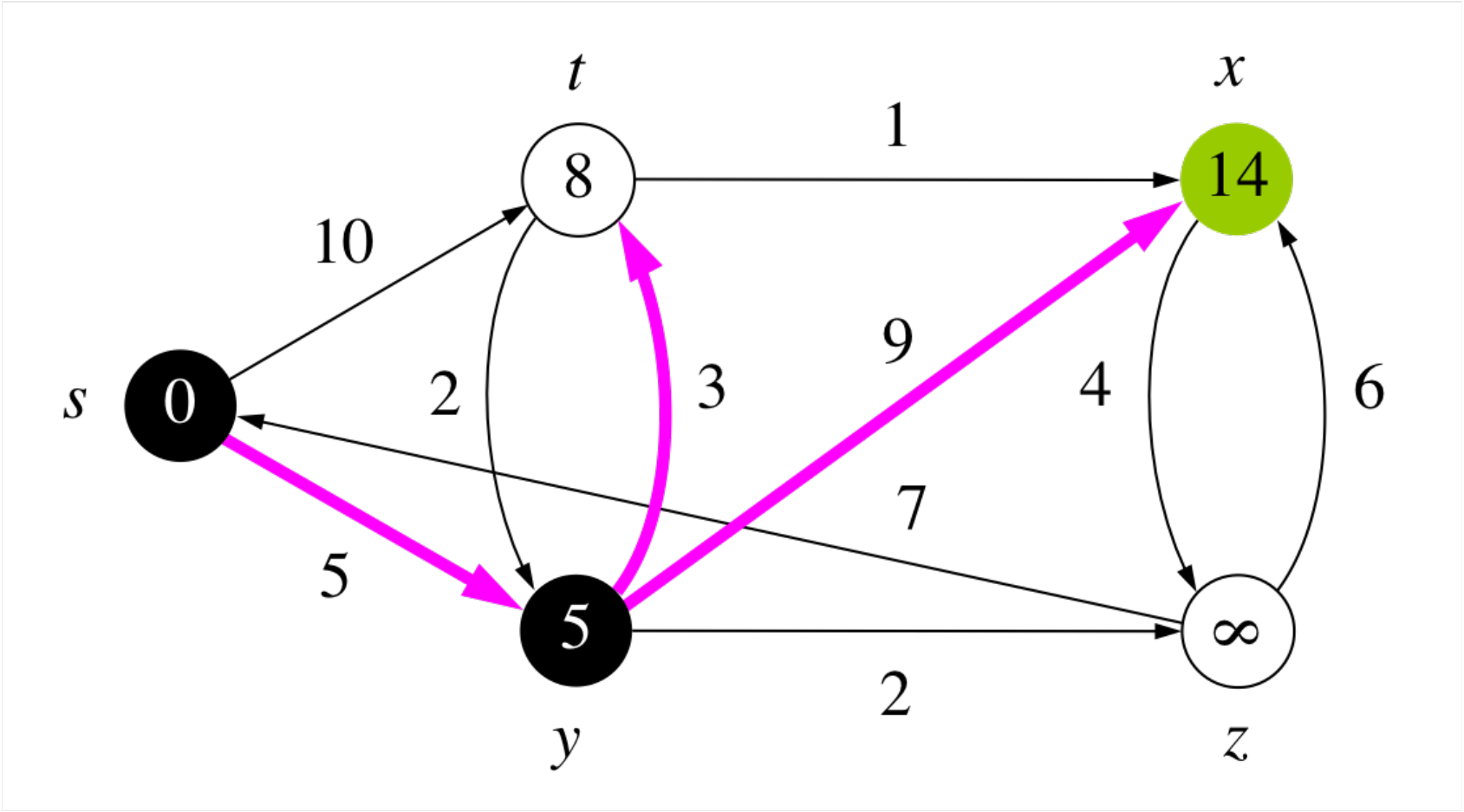
Adjacentes de y são relaxados.



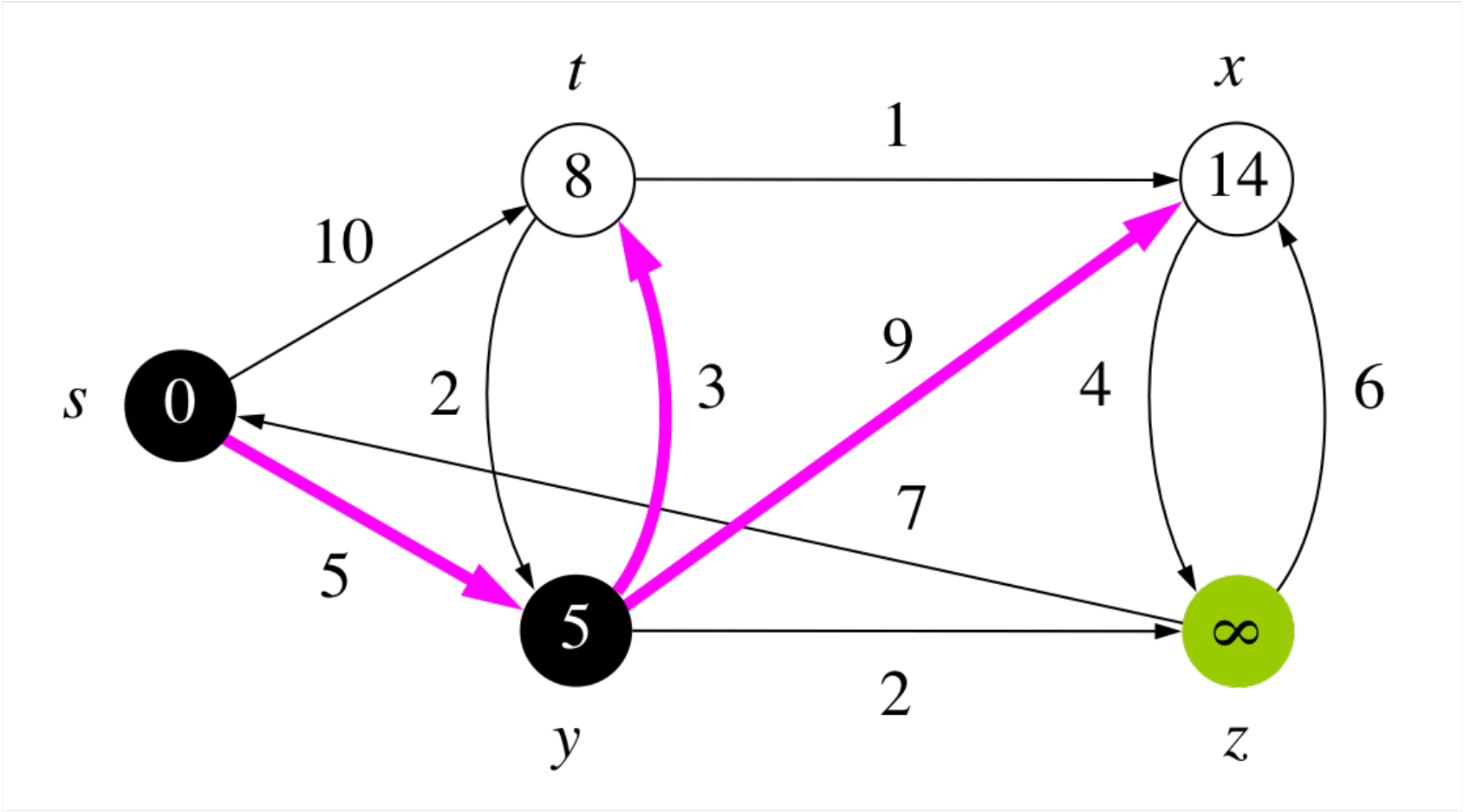
Adjacentes de y são relaxados.



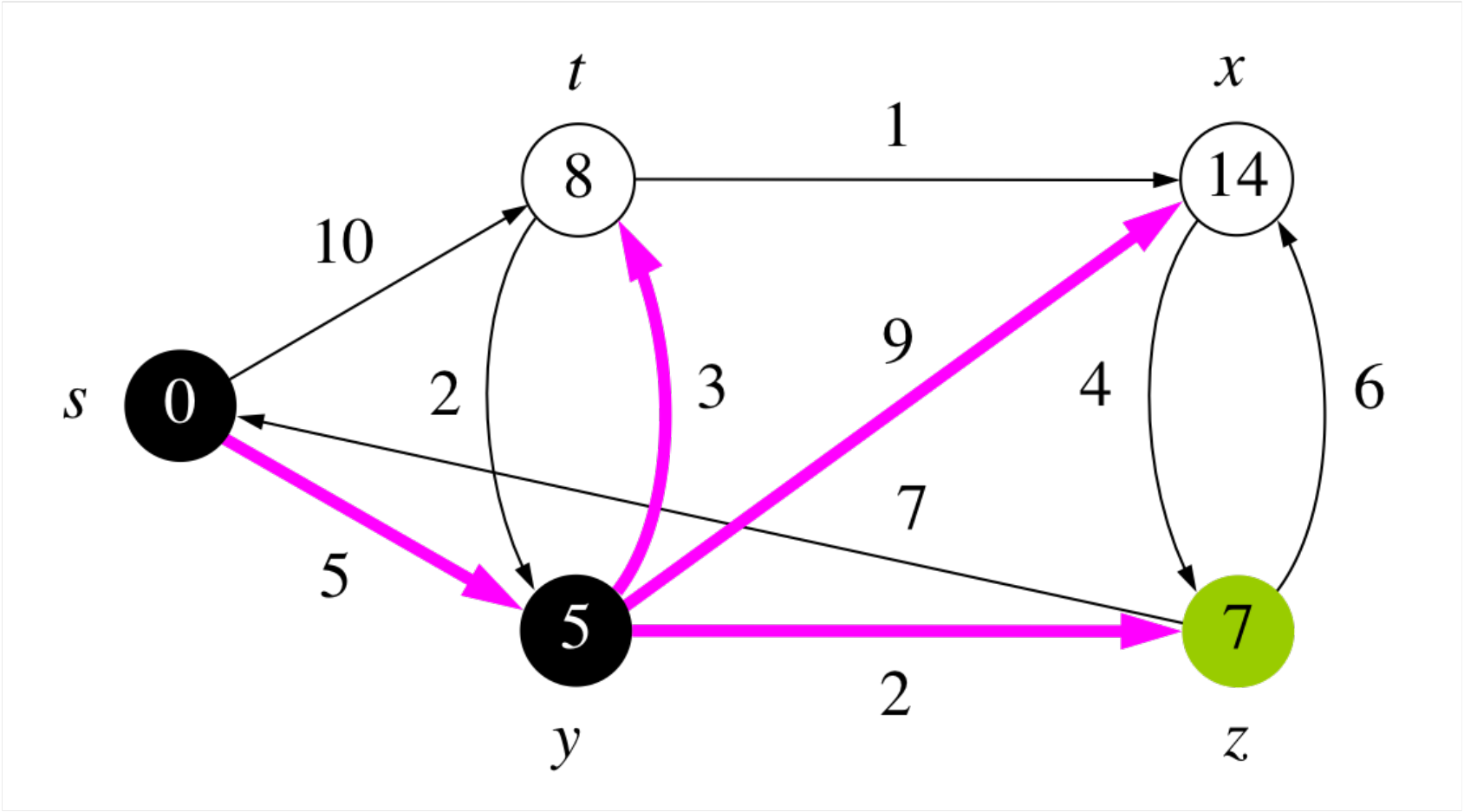
Adjacentes de y são relaxados.



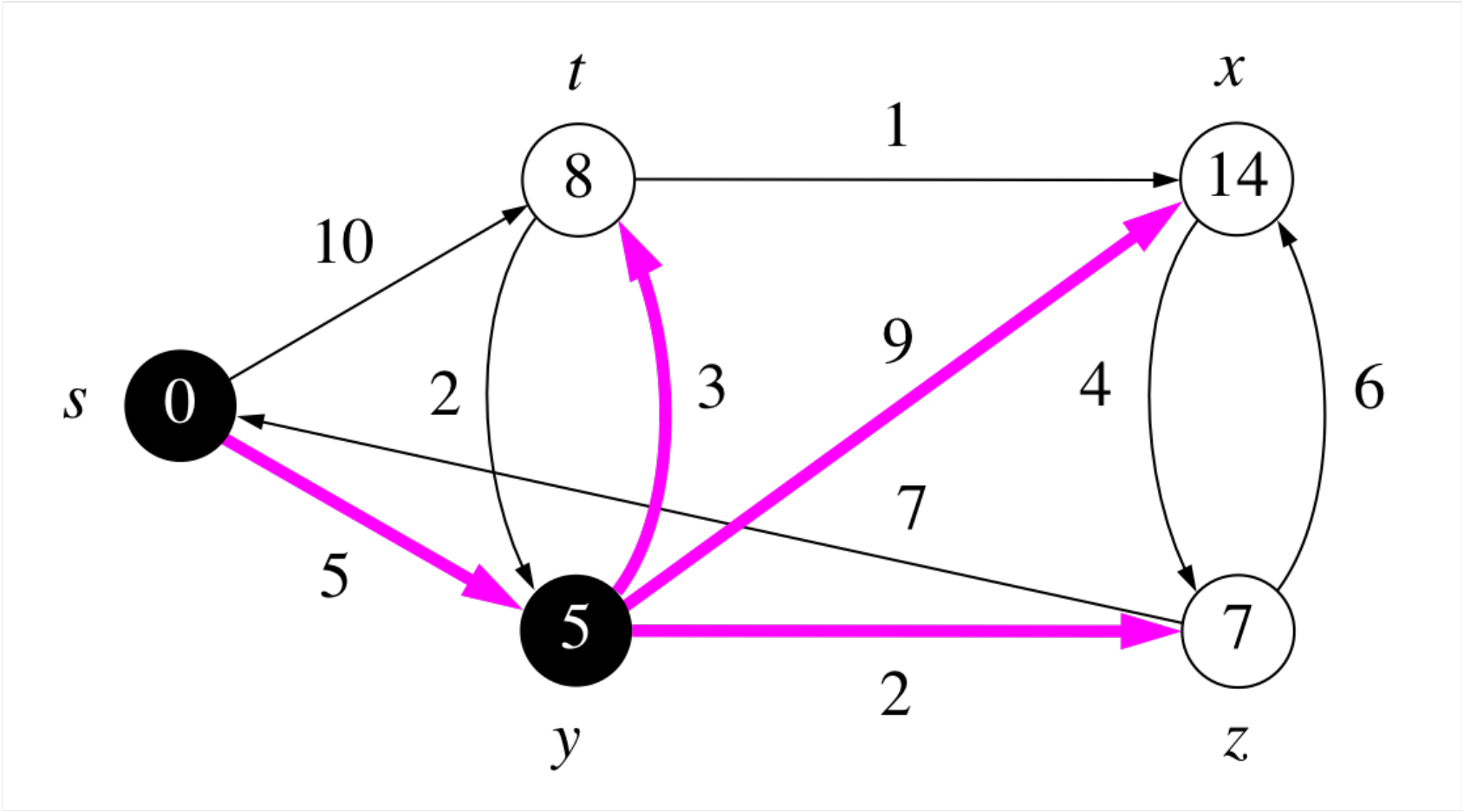
Adjacentes de y são relaxados.



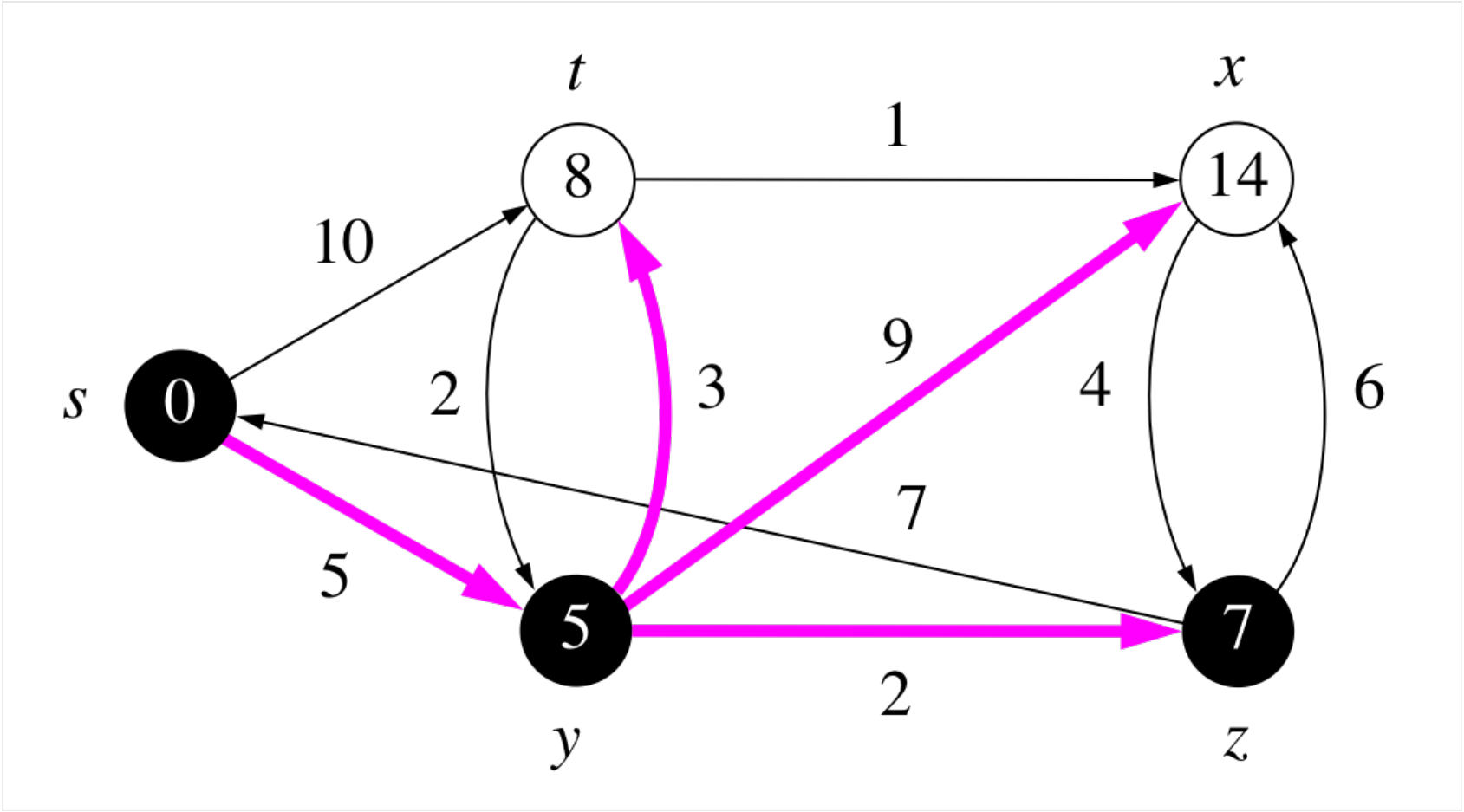
Adjacentes de y são relaxados.



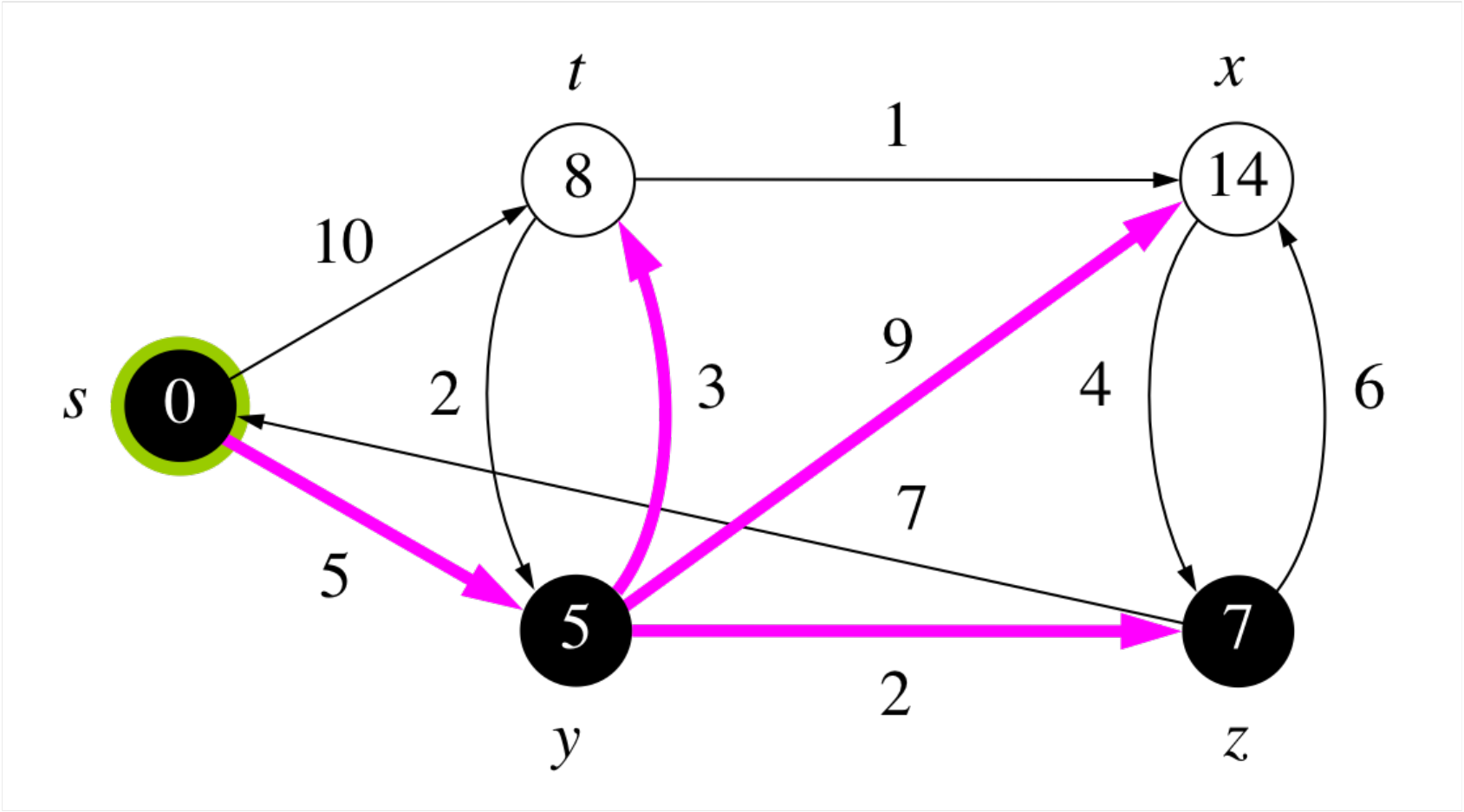
Adjacentes de y são relaxados.



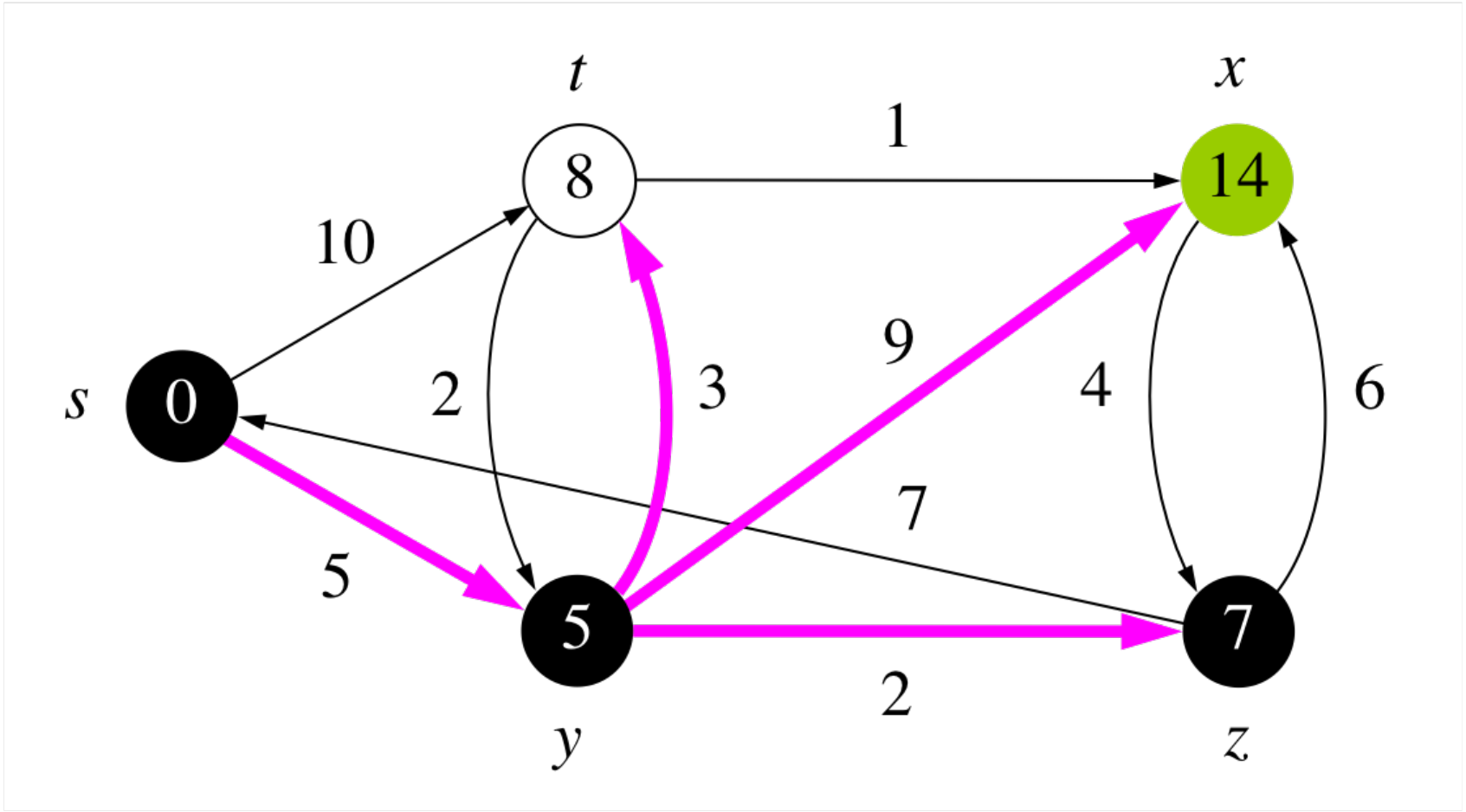
Adjacentes de y são relaxados.



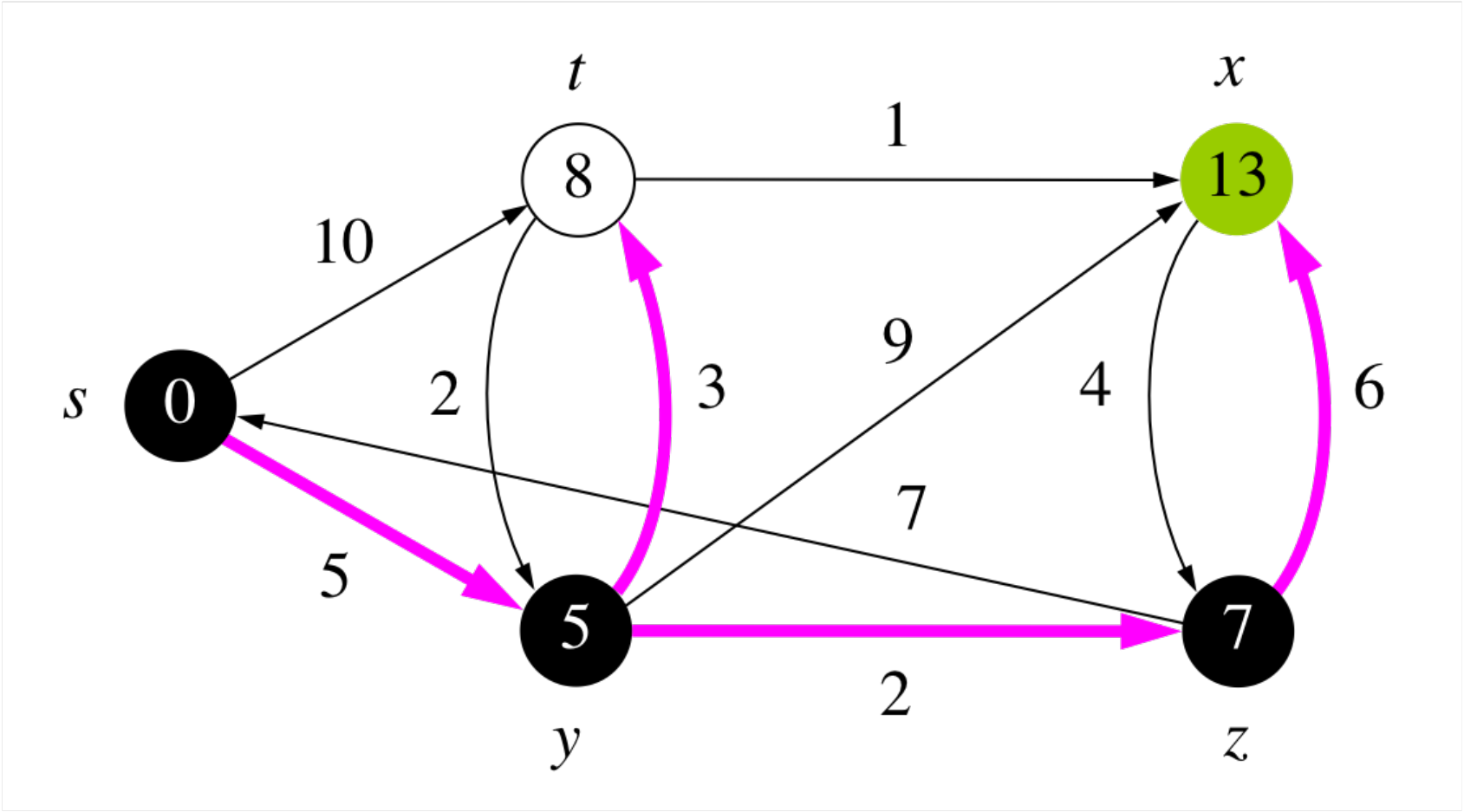
Vértice z sai da fila.



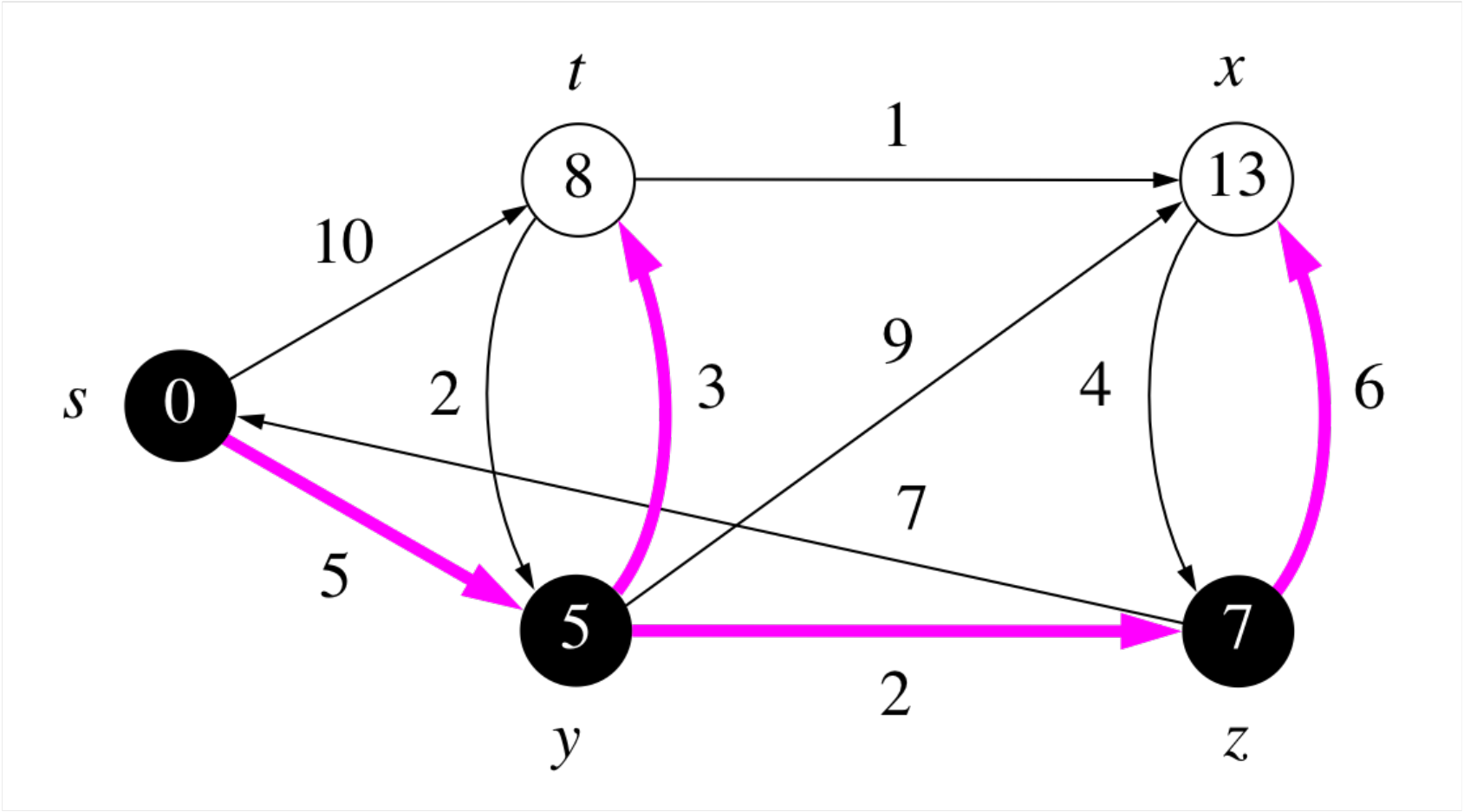
Adjacentes de z são relaxados.



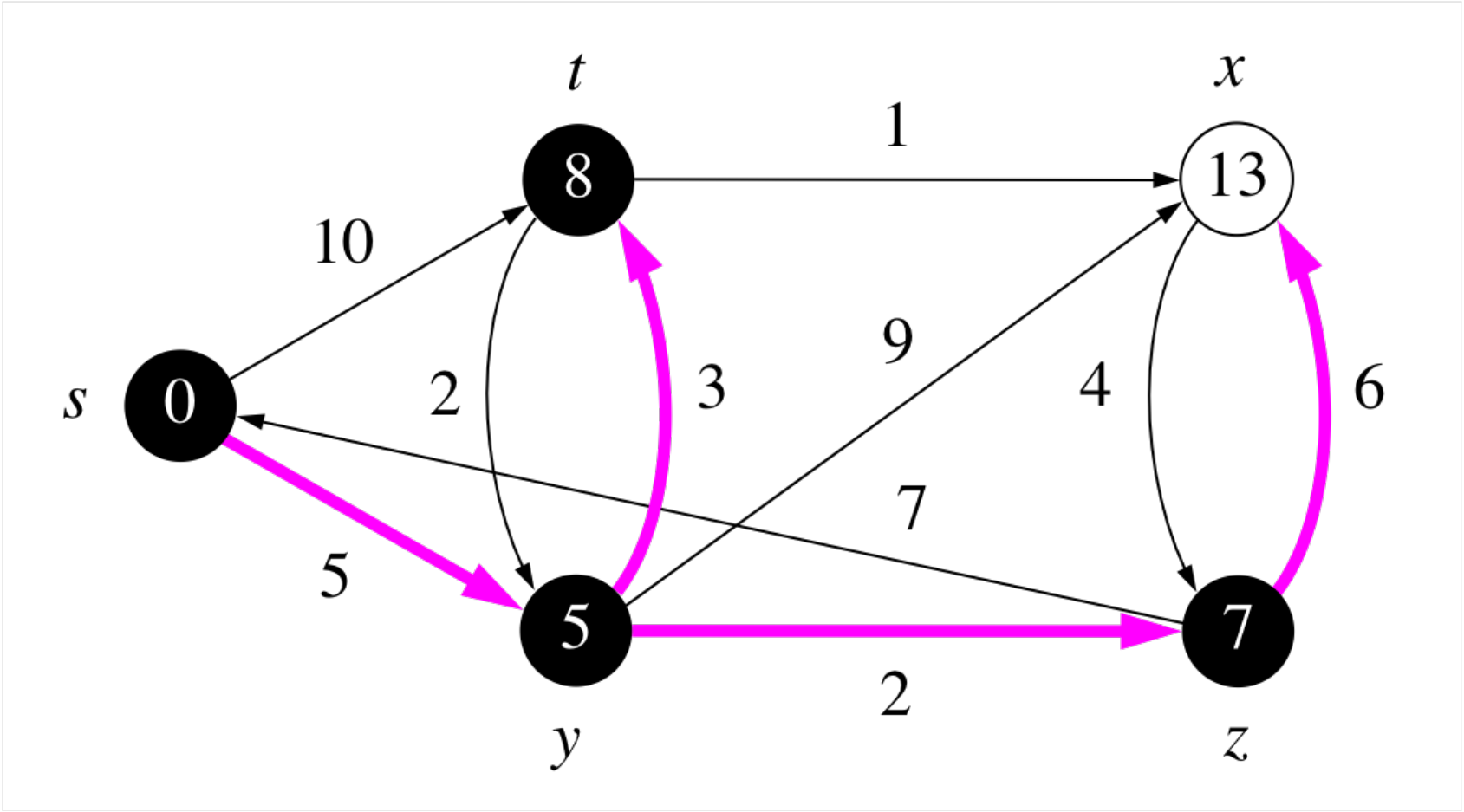
Adjacentes de z são relaxados.



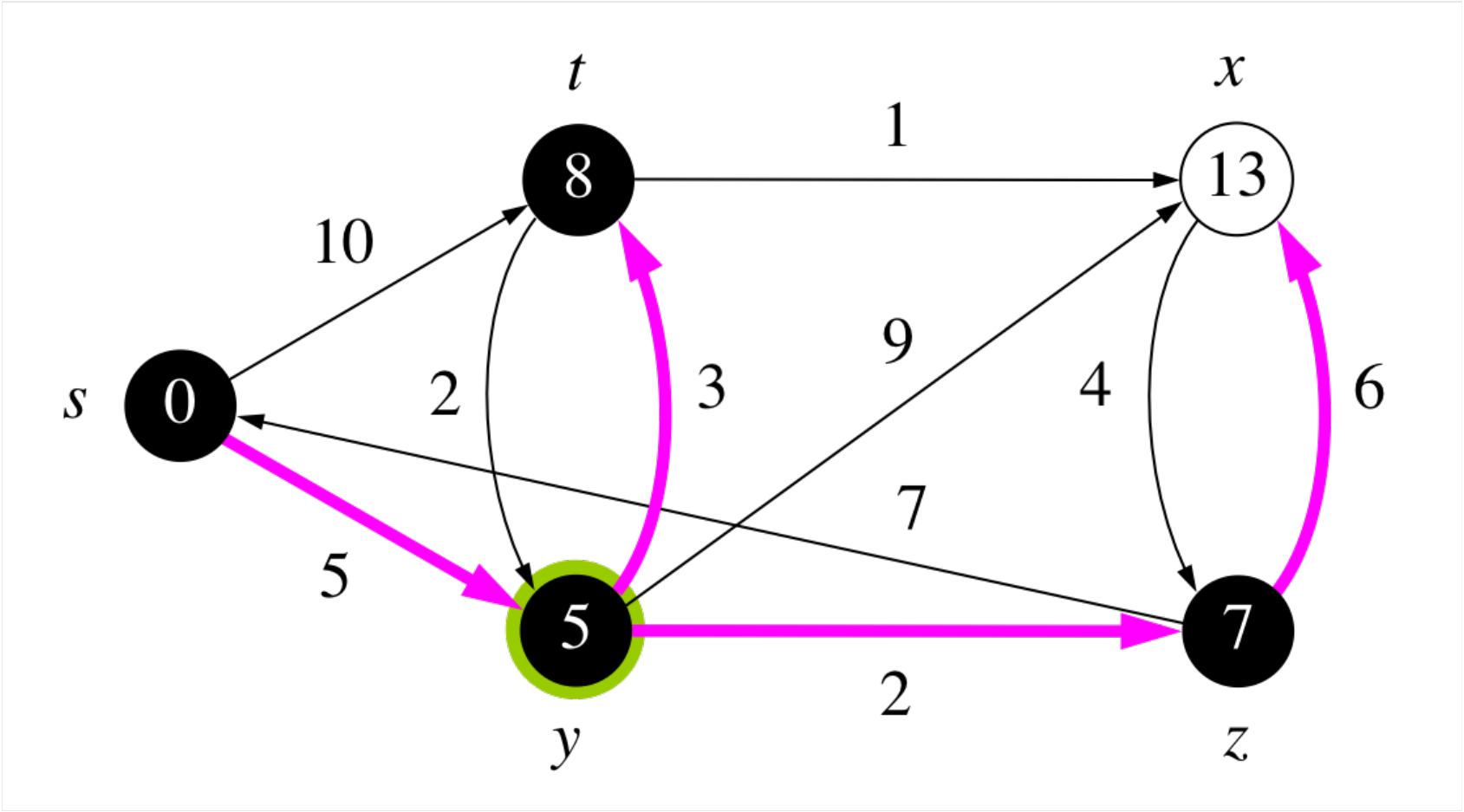
Adjacentes de z são relaxados.



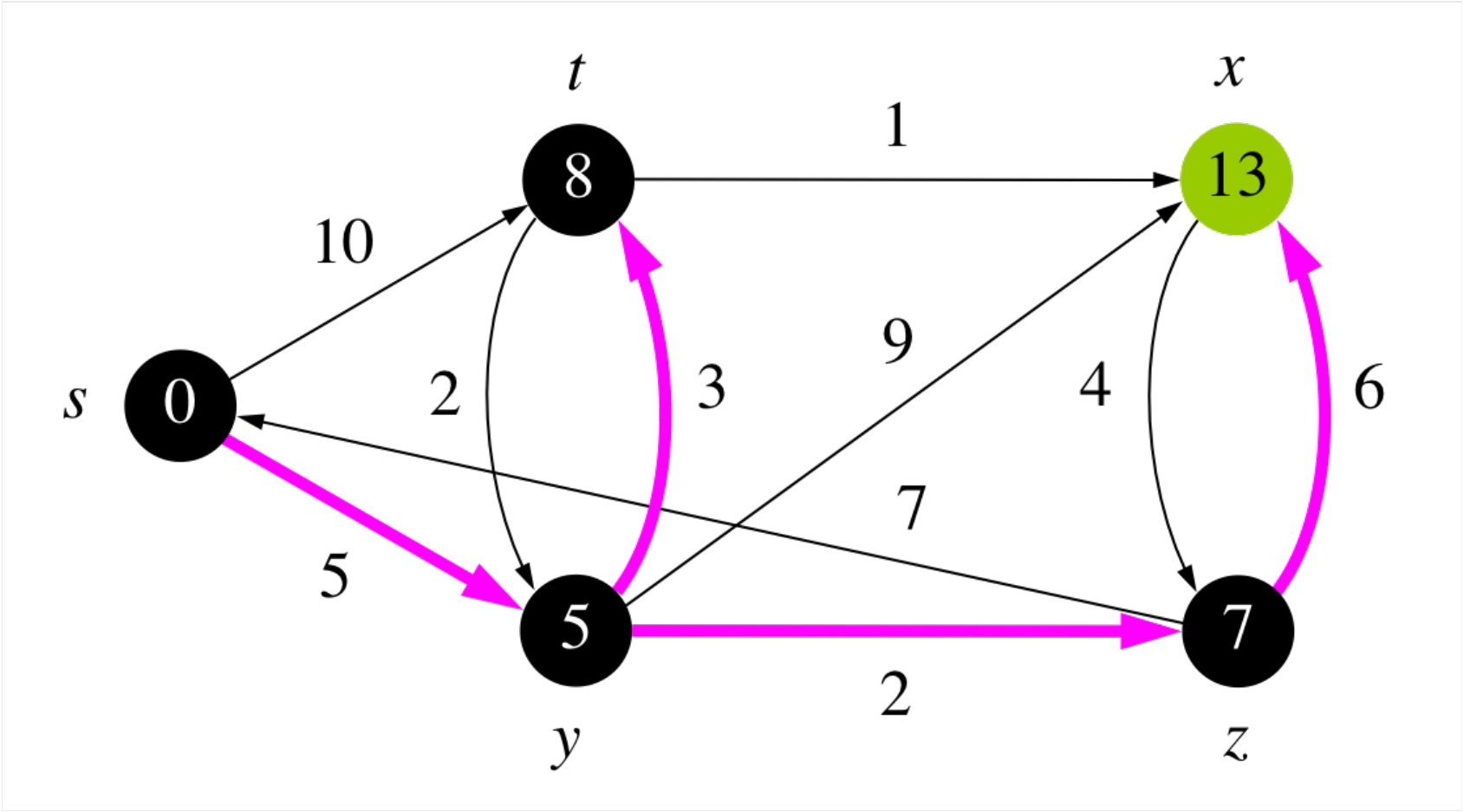
Adjacentes de z são relaxados.



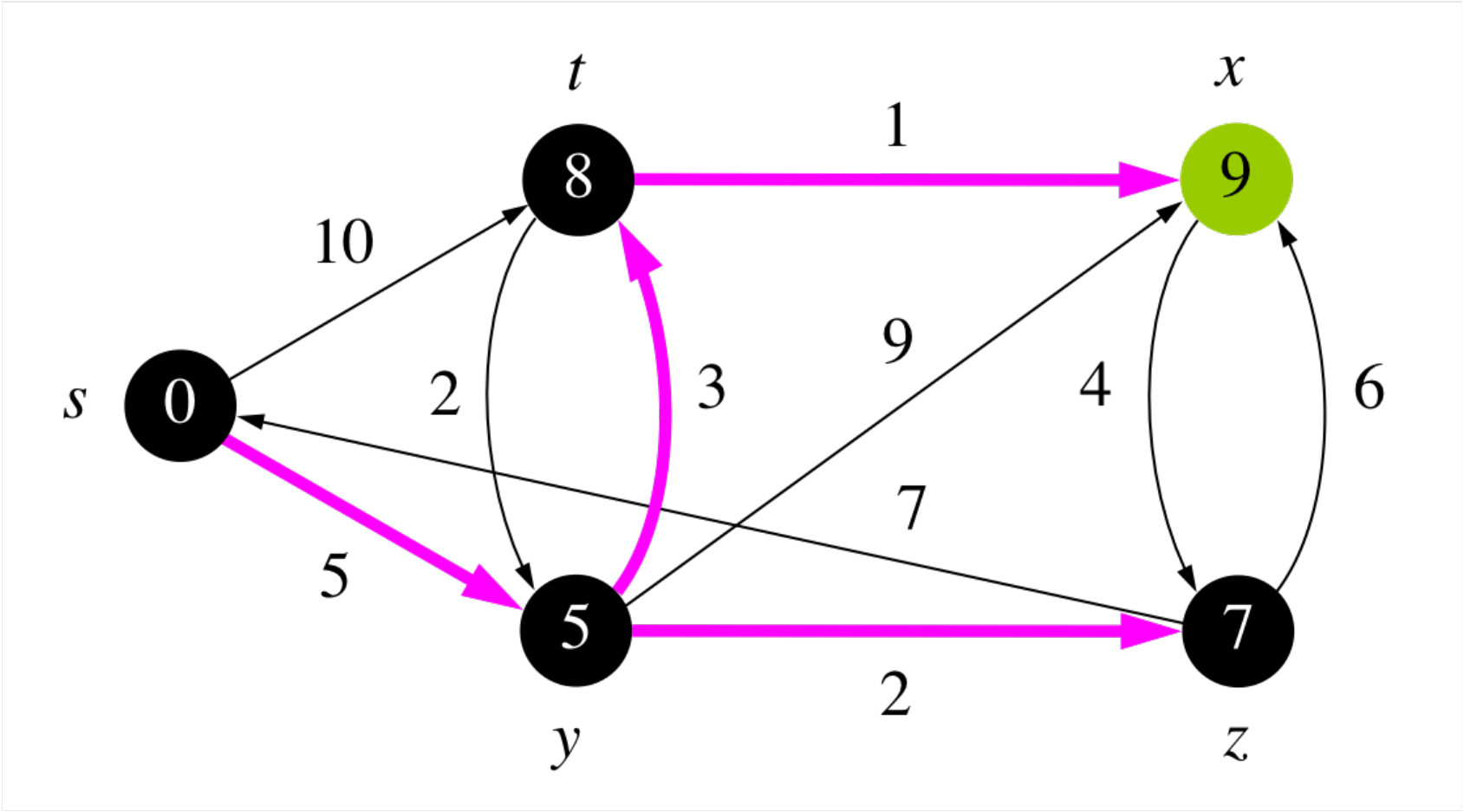
Vértice t sai da fila.



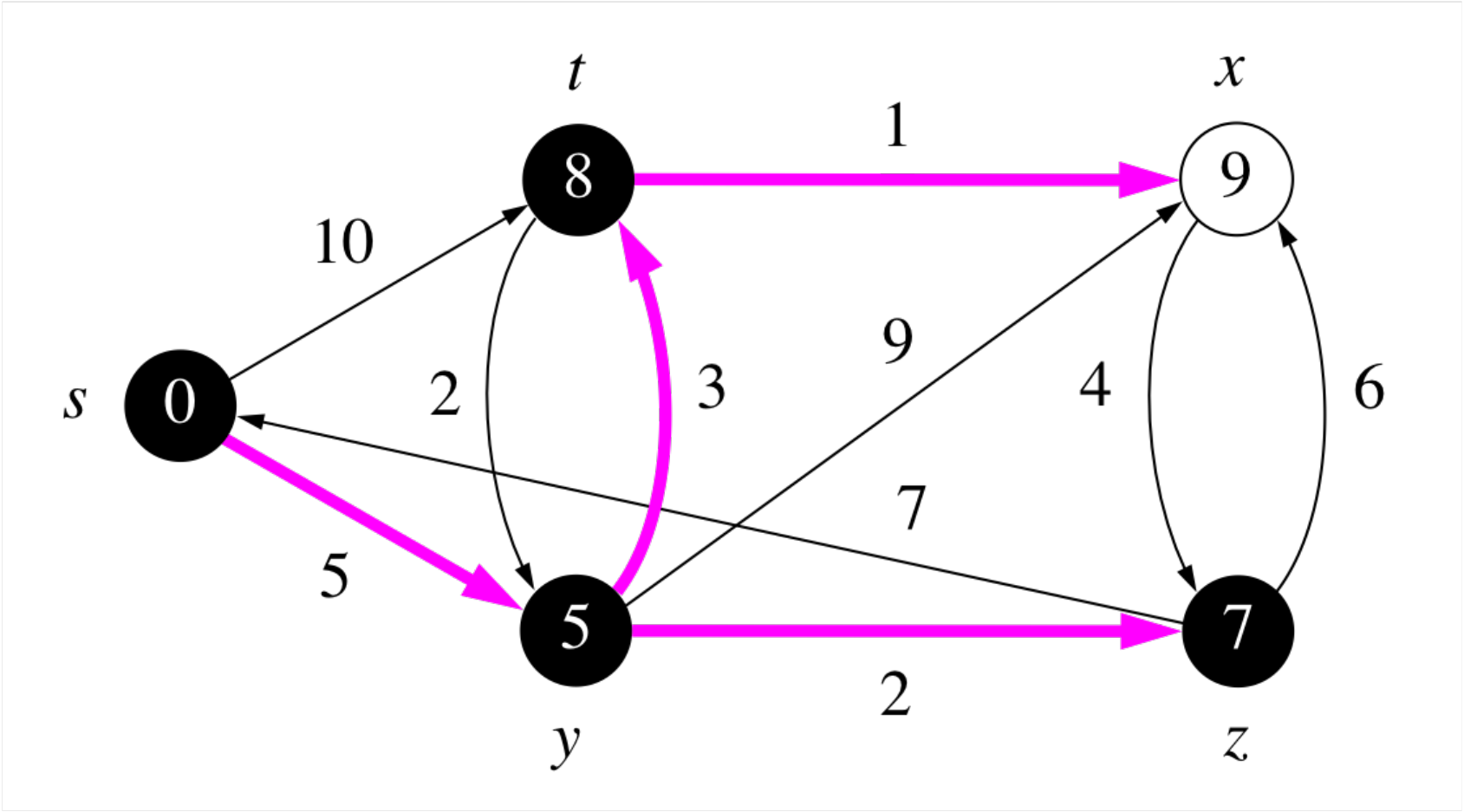
Adjacentes de t são relaxados.



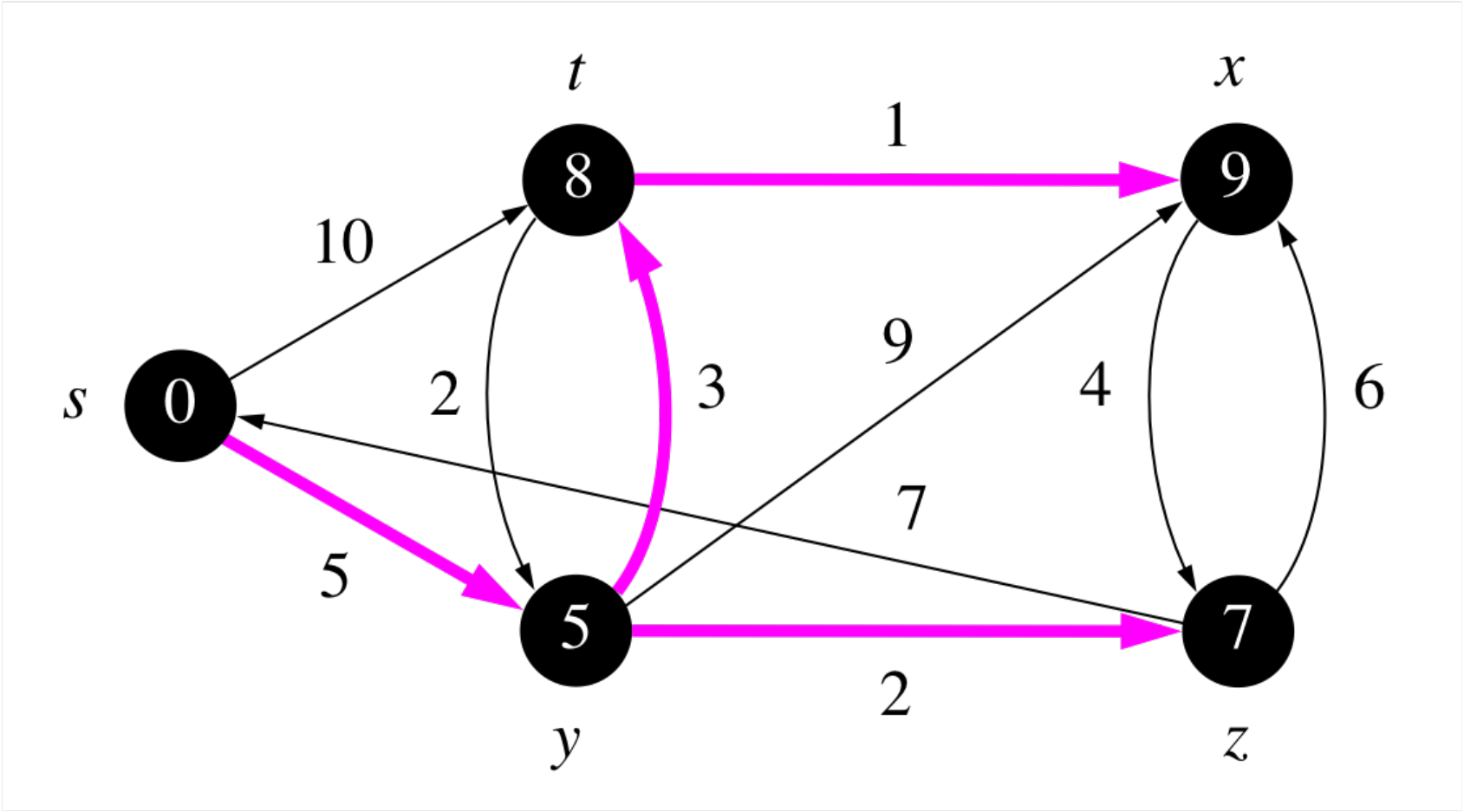
Adjacentes de t são relaxados.



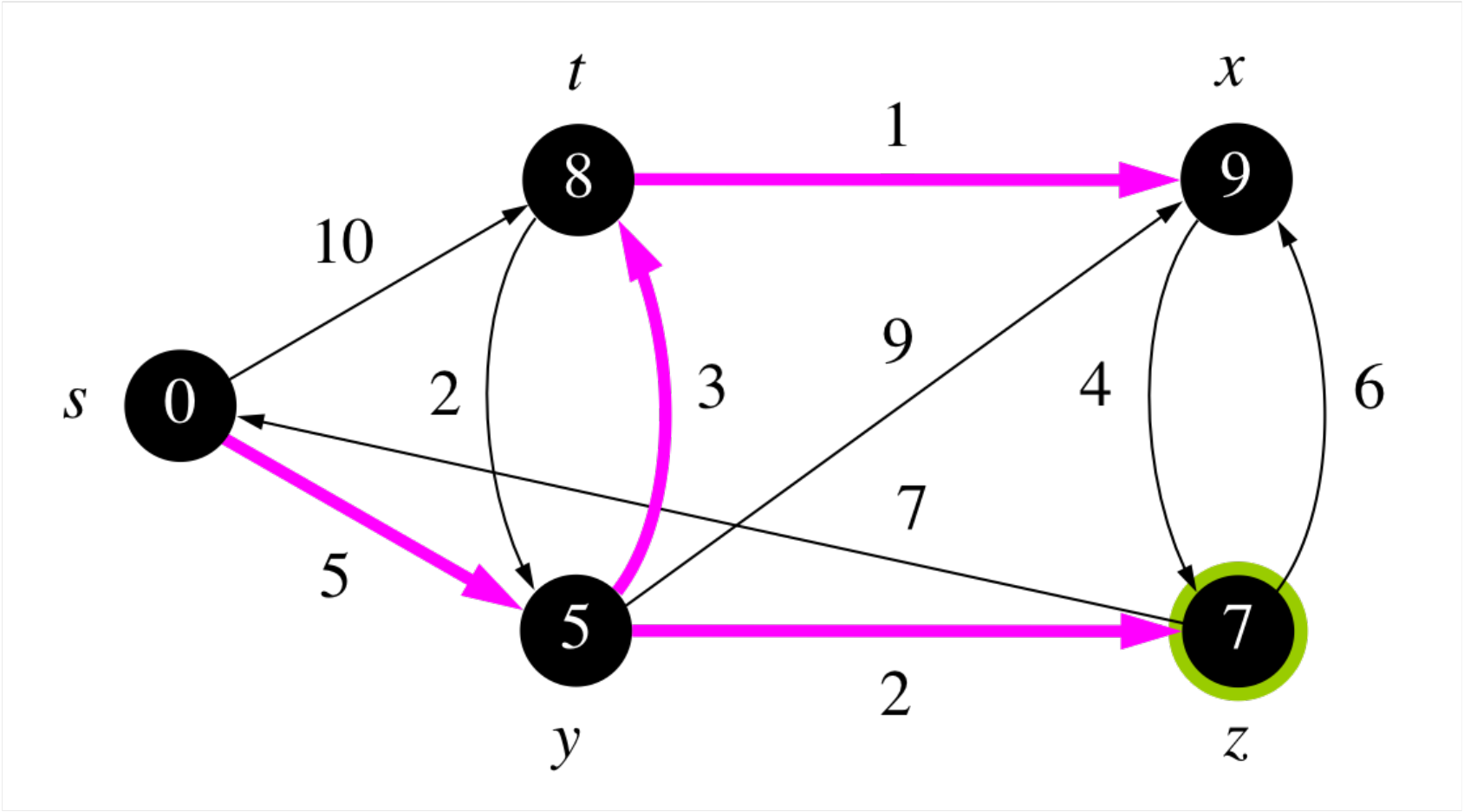
Adjacentes de t são relaxados.



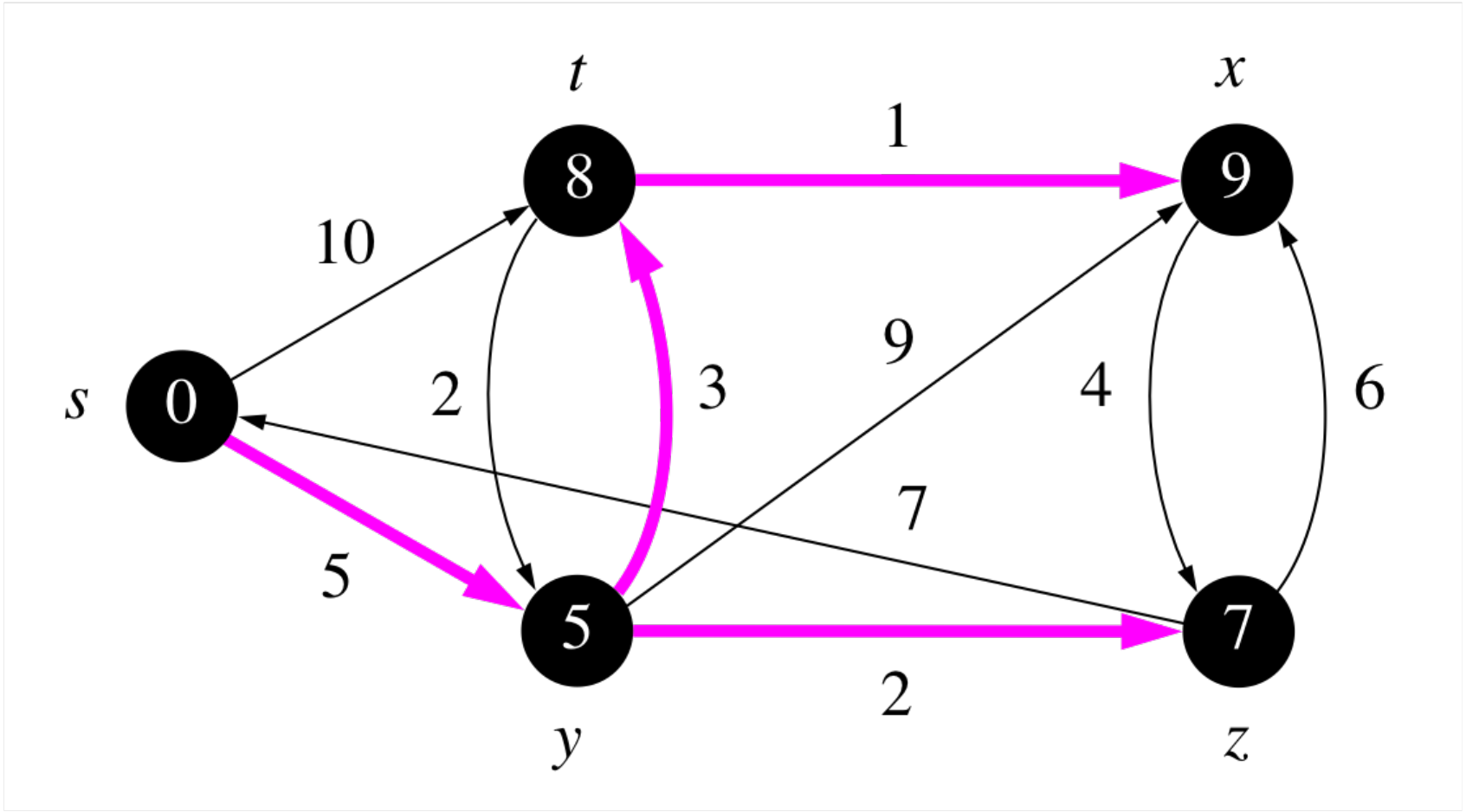
Adjacentes de t são relaxados.



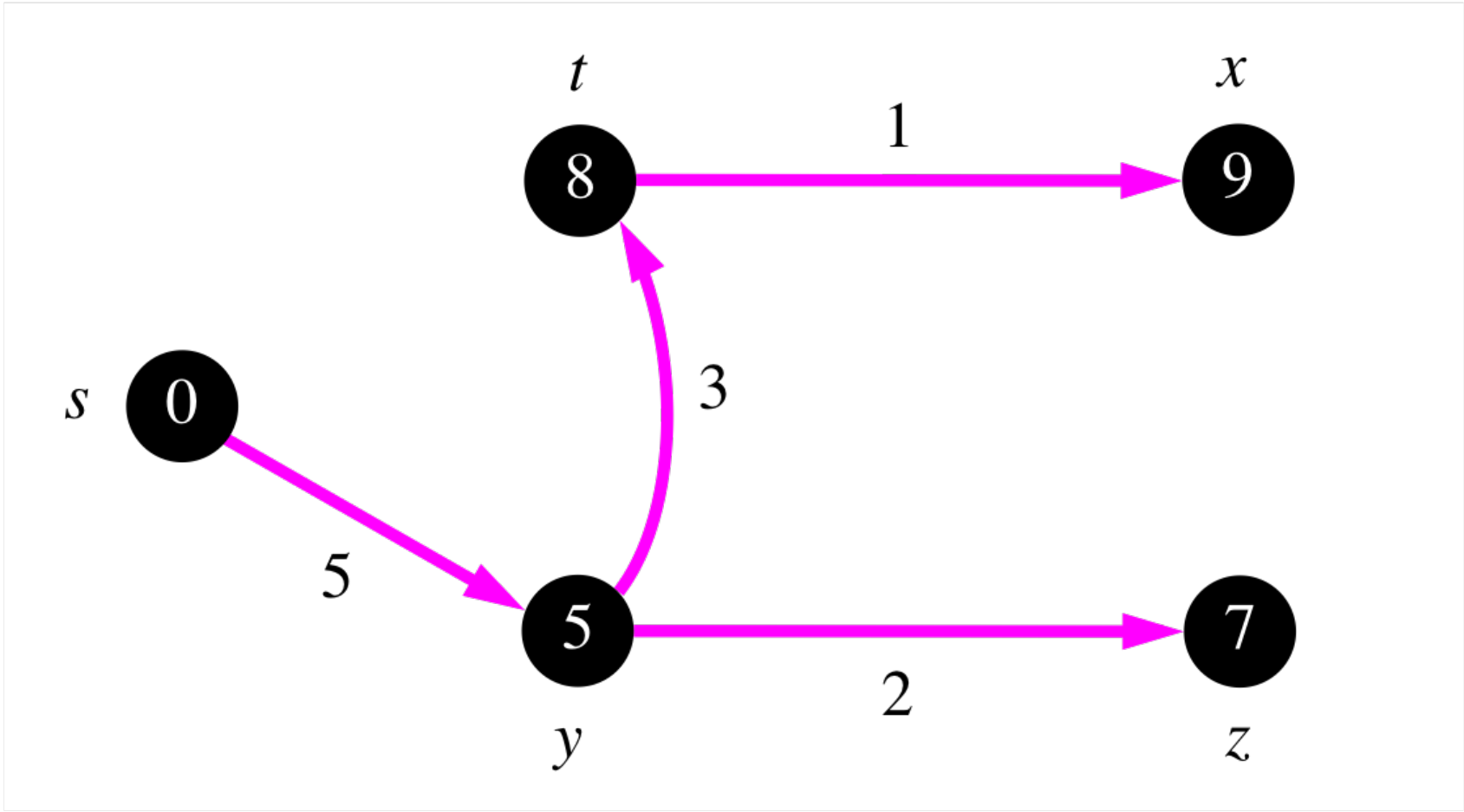
Vértice x sai da fila (Q vazia).



Adjacentes de x são relaxados.



Adjacentes de x são relaxados.



Árvore geradora com **caminhos de custo mínimo** a partir da origem.

Complexidade de tempo de Dijkstra

- Q usando vetor simples: $O(V^2)$.
- Q usando heap binário: $O(E \lg V)$.
- Q usando heap de Fibonacci: $O(V \lg V + E)$.

Referências

- T. H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd edition, MIT Press, 2010
- A. Levitin. Introduction to the Design and Analysis of Algorithms. 3rd edition. Addison-Wesley,2007
- R. Sedgewick, K. Wayne. Algorithms. 4th edition, Addison-Wesley Professional, 2011
- N. Ziviani. Projeto de Algoritmos com Implementação em Pascal C. Cengage Learning, 2012

Onde obter este material:

est.uea.edu.br/fcoelho