Maximum - Flow
Push-Relabel Algorithm

林建宇

2014/06/10

Outline

- Introduction
- Basic Operation
- The Push-Relabel algorithm

Introduction

Introduction

Algorithms for solving Maximun-Flow problem

- Ford - Fulkerson
- Edmond - Karp
- Push - Relabel

Introduction

Algorithms for solving Maximun-Flow problem

- Ford - Fulkerson O(EF)
- Edmond - Karp
- Push - Relabel

Introduction

Algorithms for solving Maximun-Flow problem

- Ford - Fulkerson O(EF)
- Edmond - Karp O(V*E^2)
- Push - Relabel

Introduction

Algorithms for solving Maximun-Flow problem

- Ford - Fulkerson O(EF)
- Edmond - Karp O(V*E^2)
- Push - Relabel O(V^3)

Introduction

Basic Ideas

Introduction

Basic Ideas

- Reservoir

Introduction

Basic Ideas

- Reservoir

Excess flow: $e(u) = \sum f(v, u) - \sum f(u, v)$ (The excess flow into u)

Introduction

Basic Ideas

- Reservoir

Excess flow: $e(u) = \sum f(v, u) - \sum f(u, v)$ (The excess flow into u)

Overflowing: $e(u) > 0$

Introduction

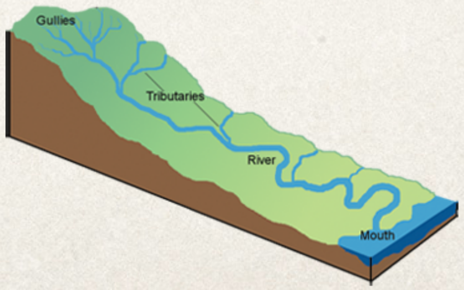
Basic Ideas

- Reservoir

  Excess flow: $e(u) = \sum f(v, u) - \sum f(u, v)$ (The excess flow into u)

  Overflowing: $e(u) > 0$

- Downhill



Introduction

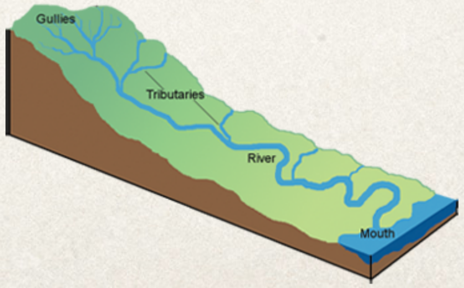
Basic Ideas

- Reservoir

Excess flow: $e(u) = \sum f(v, u) - \sum f(u, v)$ (The excess flow into u)

Overflowing: $e(u) > 0$

- Downhill



Hight

Introduction

Basic Ideas

- Admissible edge

Introduction

Basic Ideas

- Admissible edge

  if $C_f(u, v) > 0$ and $h(u) = h(v) + 1$,

  then (u, v) is an **admissible edge.**

  Otherwise, it's **inadmissible.**

The admissible network $E_{f,h}$ is a set of admissible edges.

The admissible network is **acyclic**.

Introduction

Basic Ideas

- Neighbor lists

Introduction

Basic Ideas

- Neighbor lists

The neighbor list **u.N** is a singly linked list of the neighbors of u in G.

u.N contains exactly those vertices v for which there may be a residual edge (u, v).

u.N.head points to the first vertex in u.N.

v.next-neighbor points to the vertex following v in a neighbor list.

u.current points to the vertex currently under consideration in u.N.

Basic Operation

Basic Operation

- Push
- Relabel
- Discharge
- Initialize - Preflow

Basic Operation

- Push(u, v)

1. When:
   - u is overflowing
   - $C_f(u, v) > 0$
   - $u.h = v.h + 1$

2. Action:

   Push $\Delta f(u, v) = \min(u.e, C_f(u, v))$ units of flow from u to v

Basic Operation

- Push(u, v)

3. **if** (u, v) $\in$ E

   (u, v).f += Δf(u, v)

   else

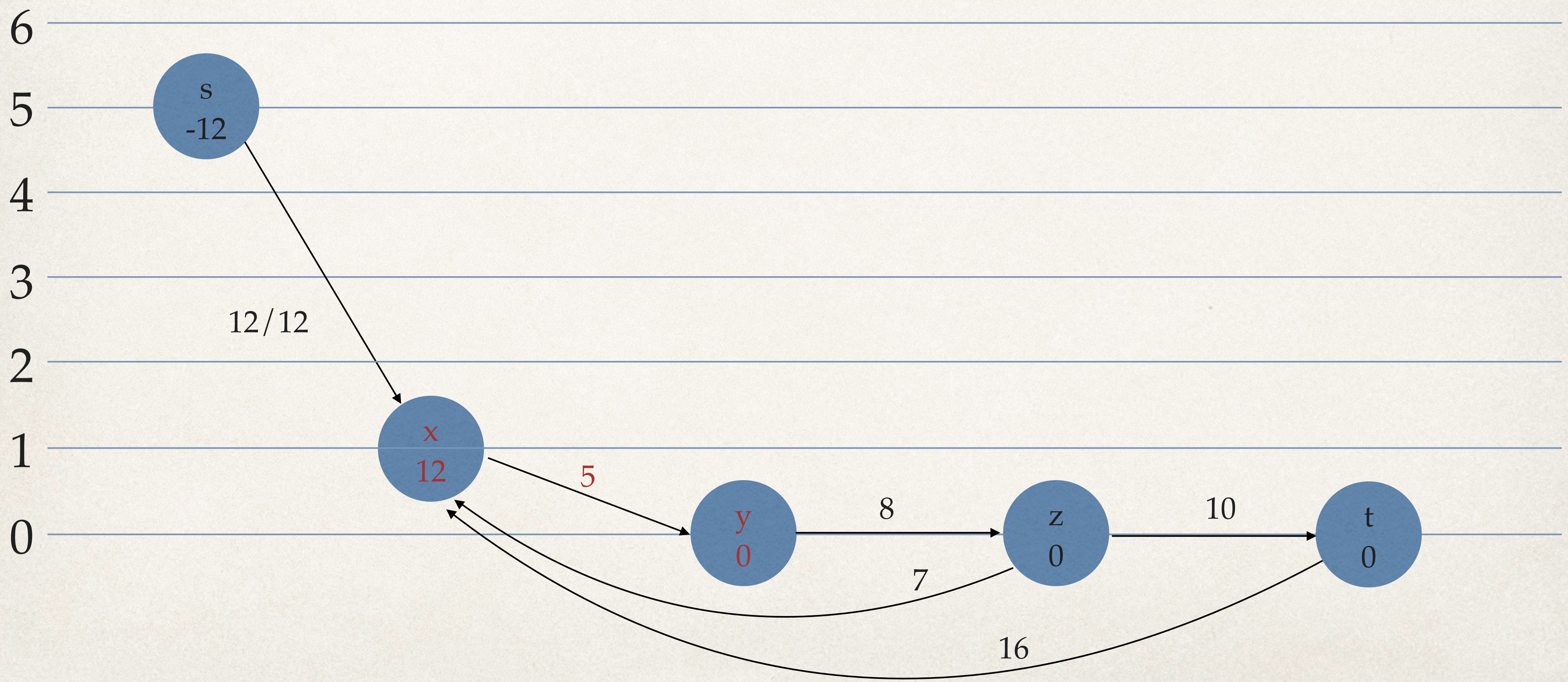
   (v, u).f -= Δf(u, v)

4. u.e -= Δf(u, v)

5. v.e += Δf(u, v)

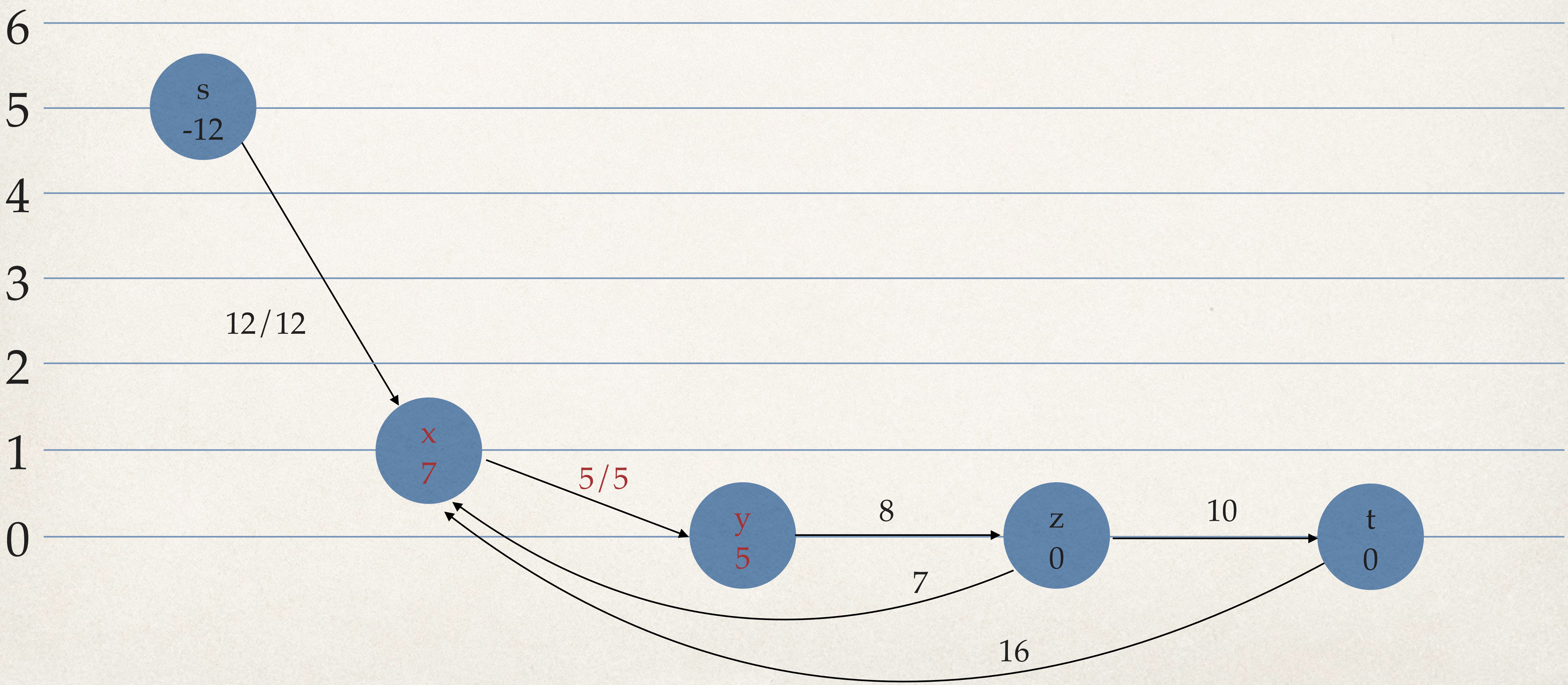
Basic Operation

- Saturating Push



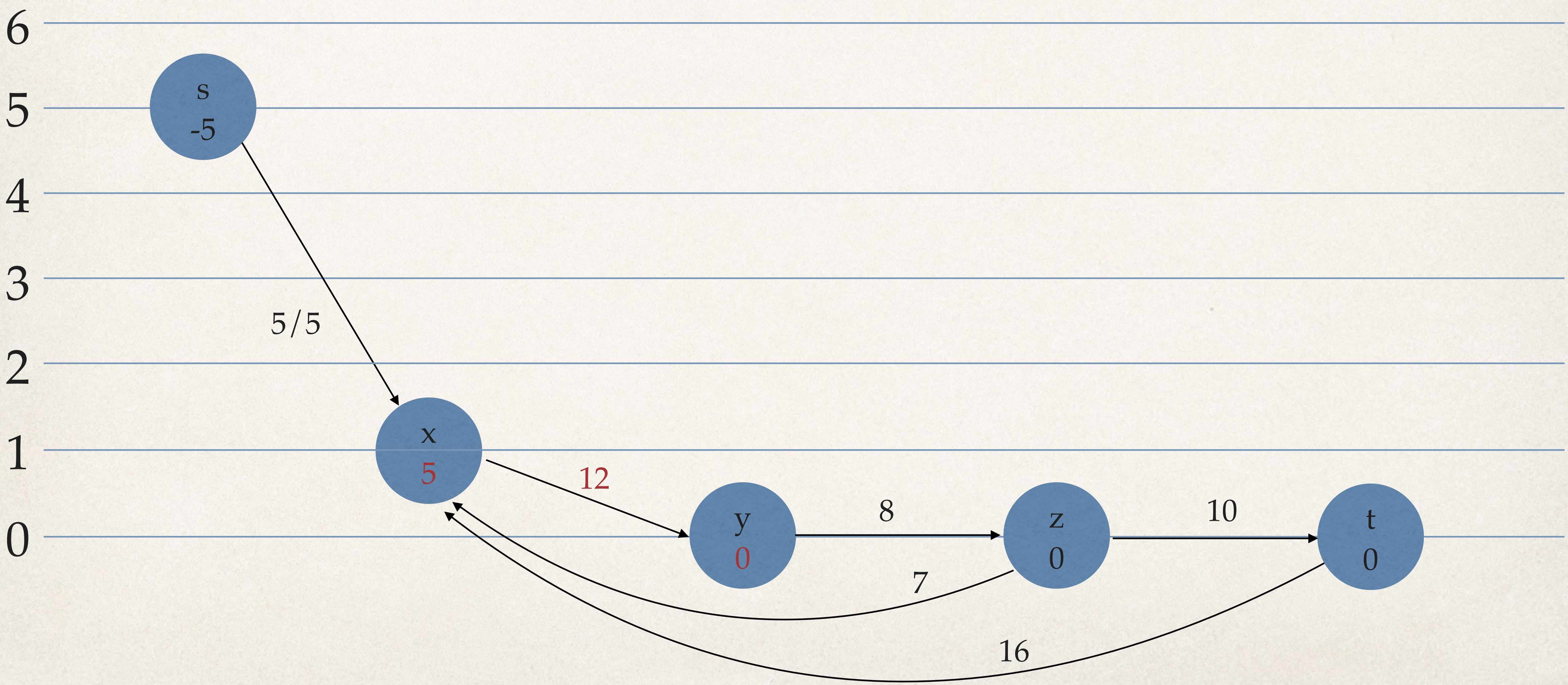
Basic Operation

- Saturating Push



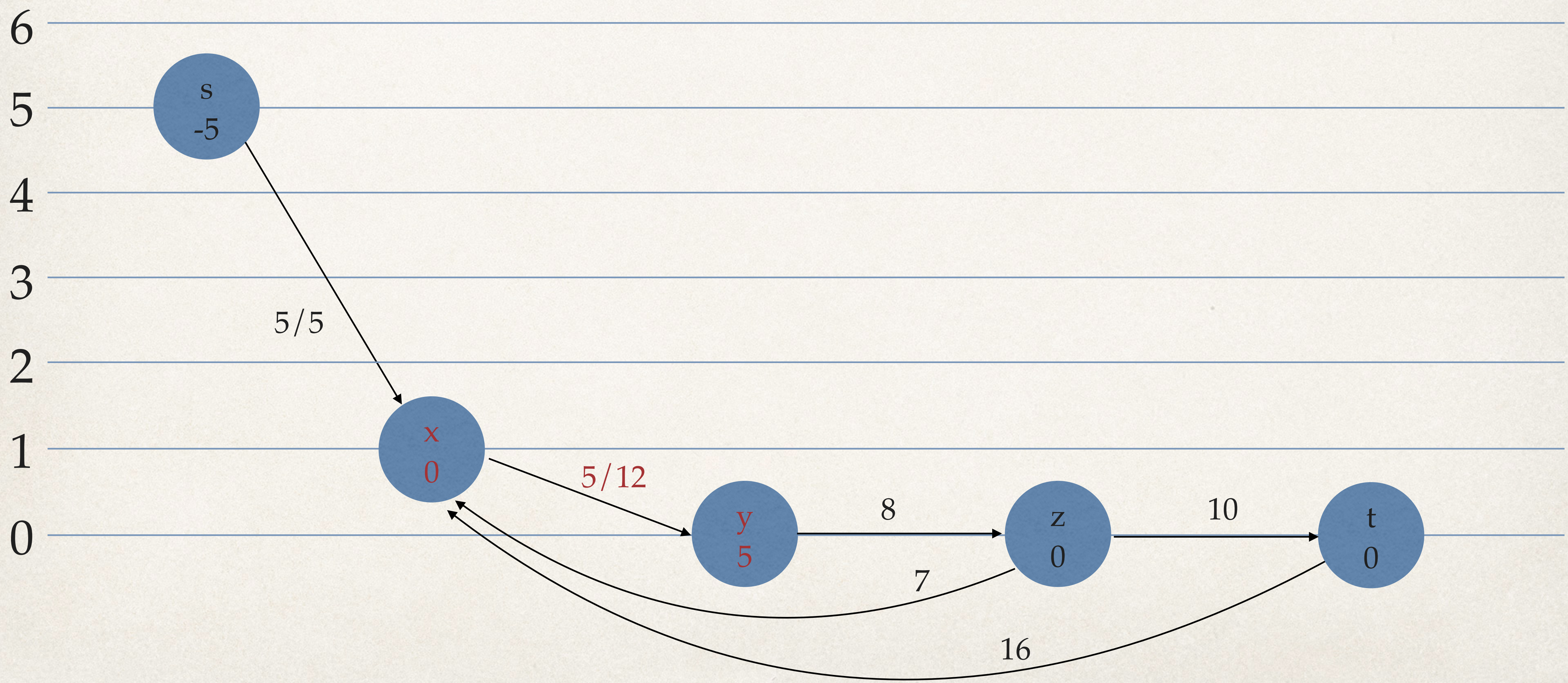
Basic Operation

- Nonsaturating Push



Basic Operation

- Nonsaturating Push



Basic Operation

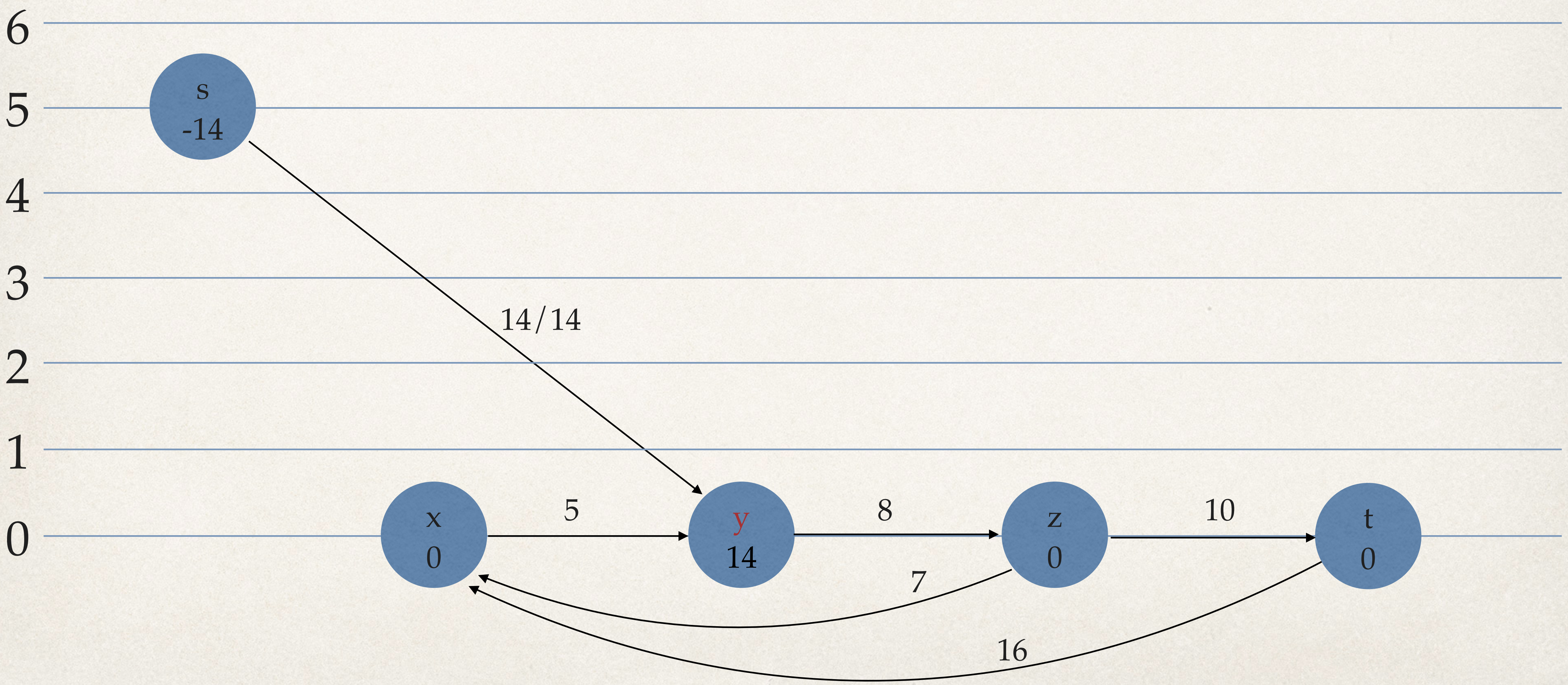
- Relabel

1. When:
   - u is overflowing
   - for all $v \in V$ such that $(u, v) \in E_f$, we have $u.h <= v.h$

2. Action:
   $u.h = 1 + \min\{v.h: (u, v) \in E_f,\}$
   (Increase the hight of u)

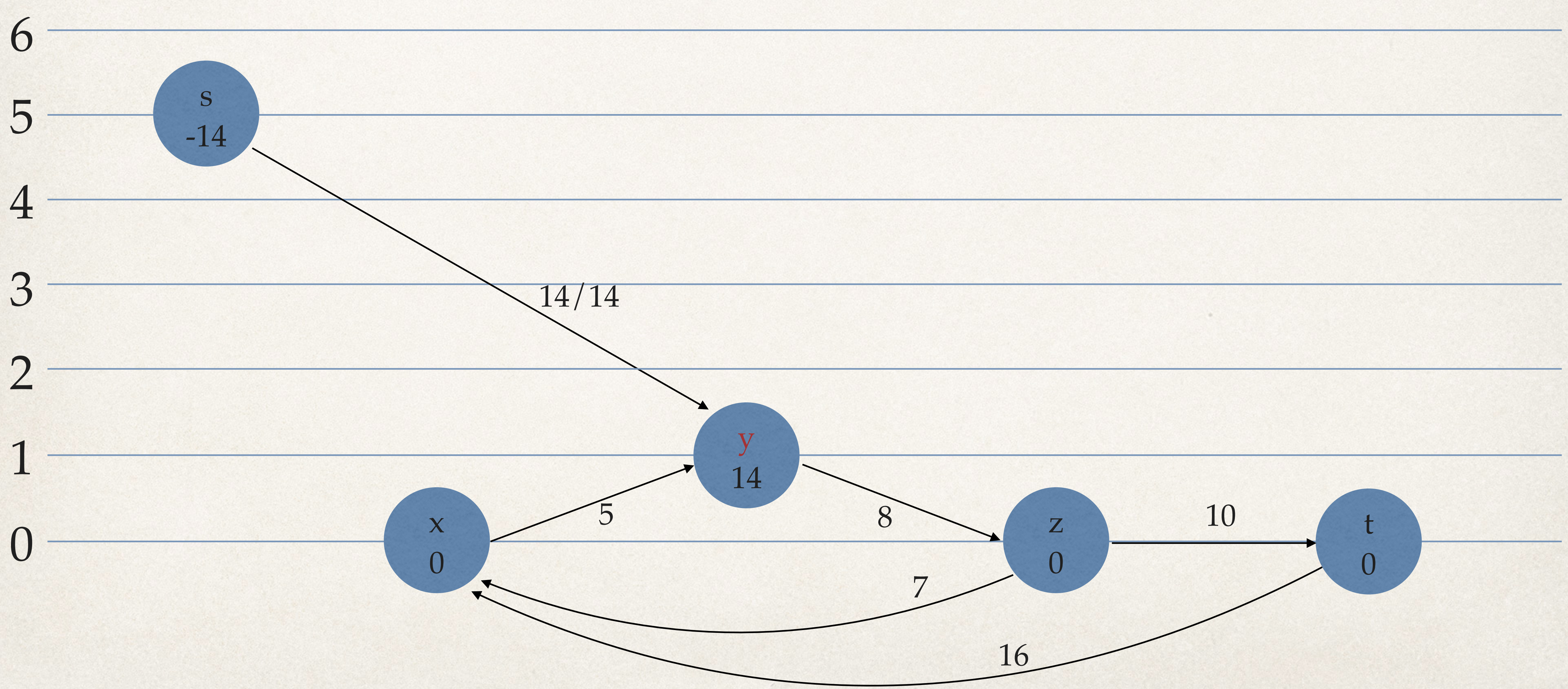
Basic Operation

- Relabel



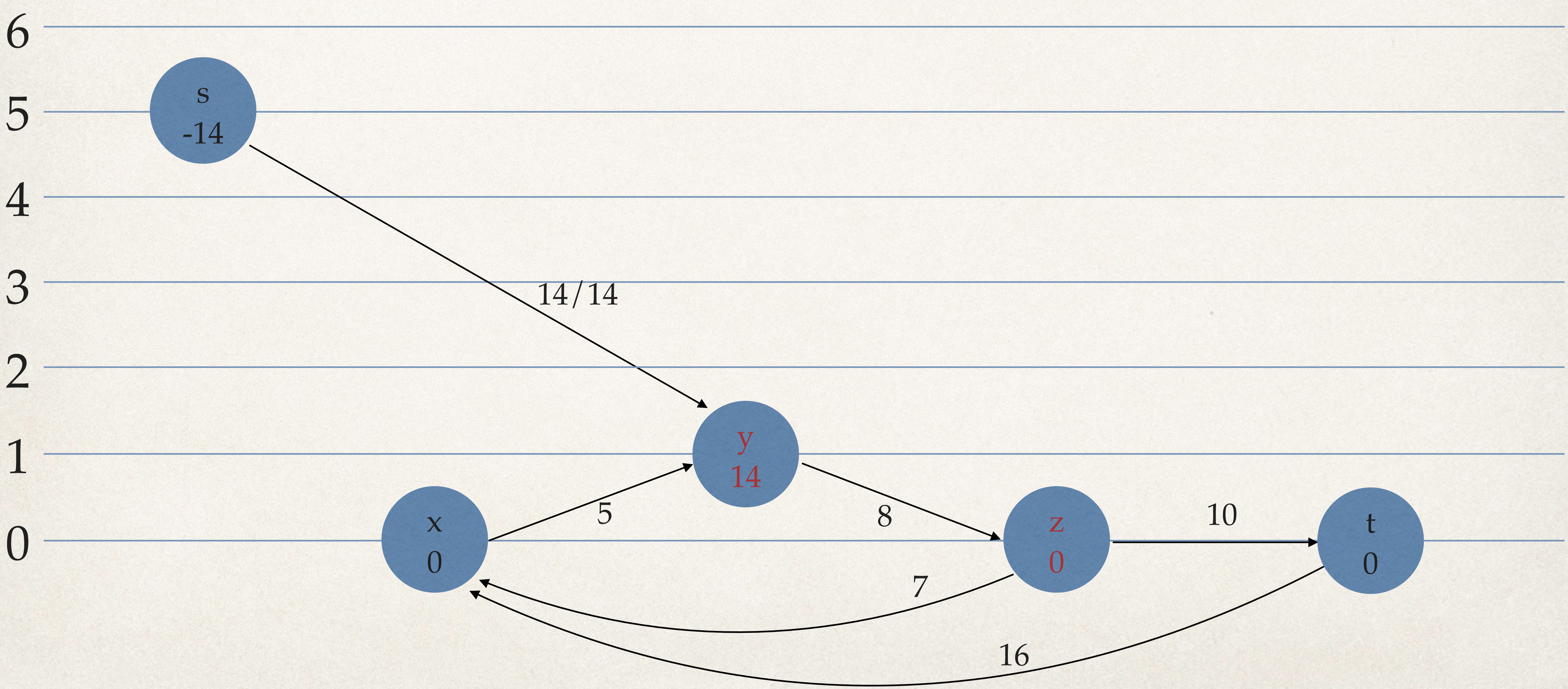
Basic Operation

- Relabel



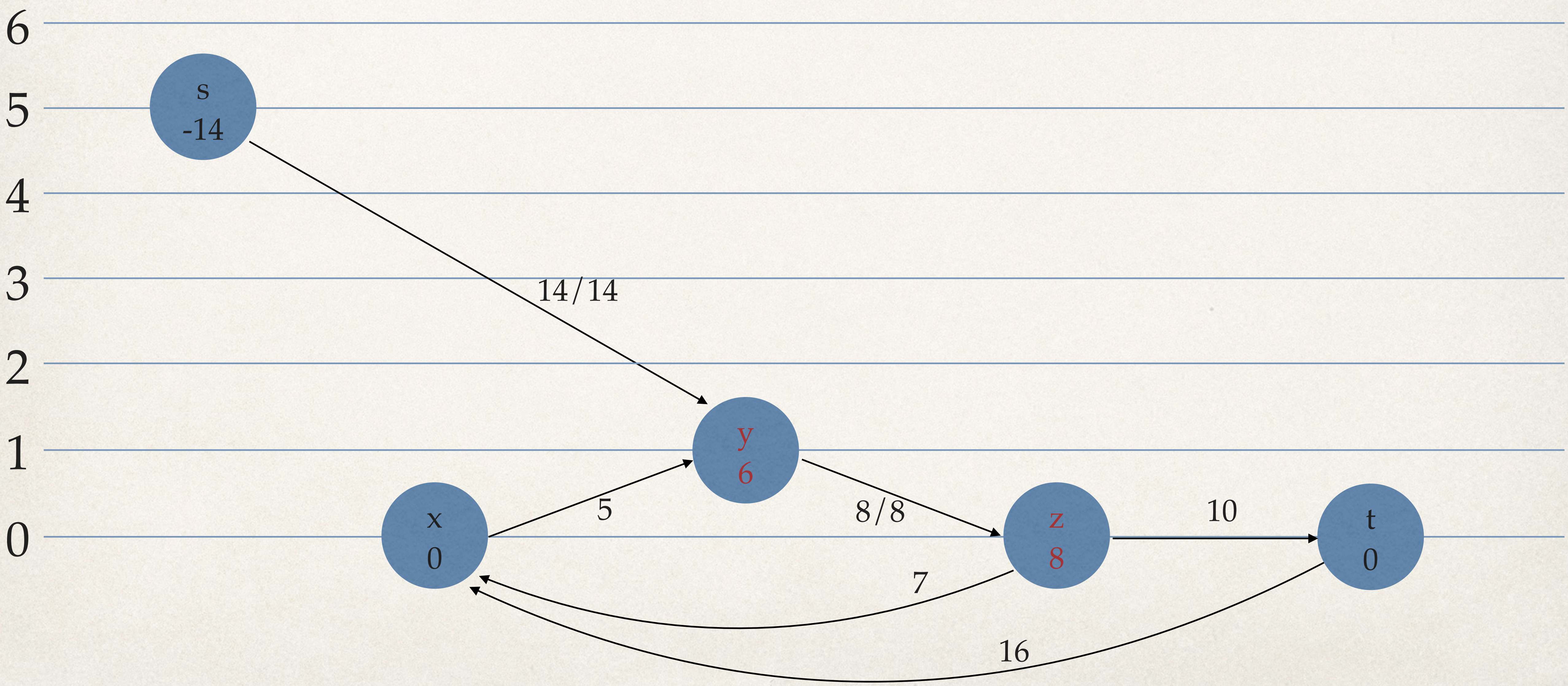
Basic Operation

- Relabel → Push



Basic Operation

- Relabel → Push



Basic Operation

- Discharge

1. When:
   - u is overflowing

2. Action:

   Push all excess flow of a vertex u to through admissible edges to neighboring vertices.

   Relabel u as necessary to cause edges leaving u to become admissible.

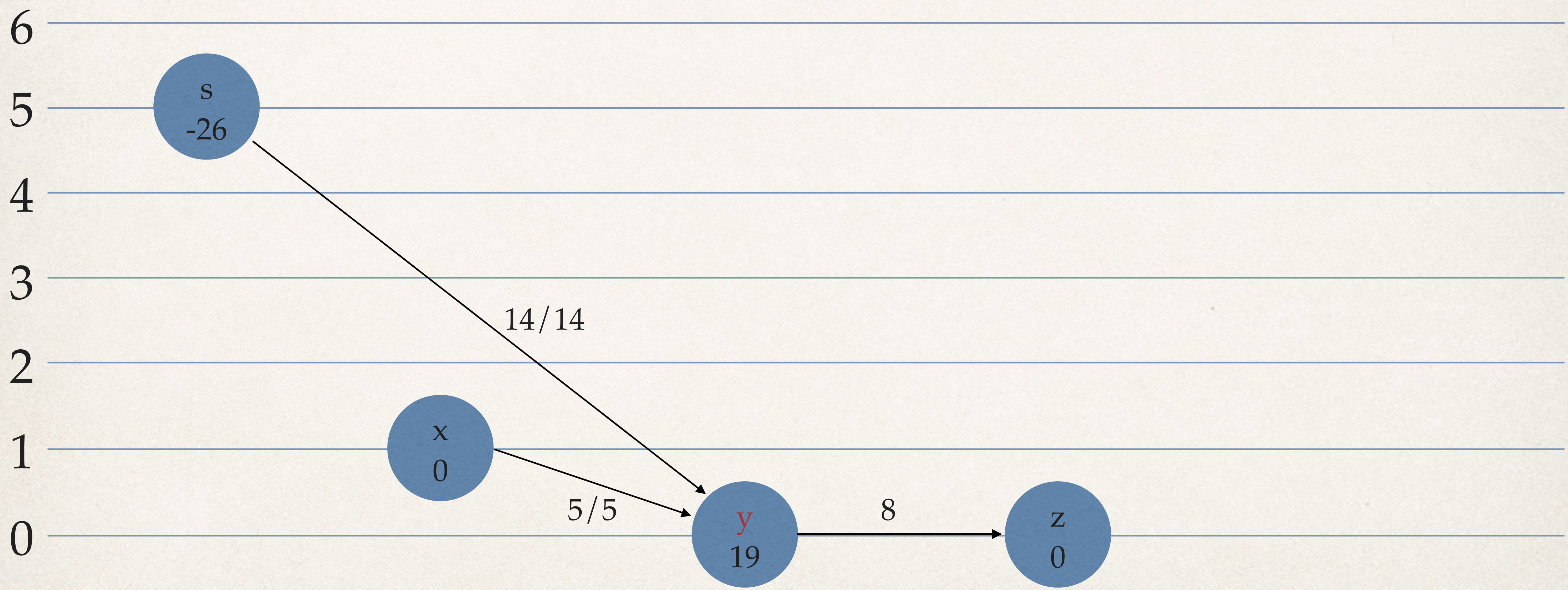
Basic Operation

- Discharge (u)

```
While u.e > 0
    v = u.current
    if v == NIL
        Relabel (u)
        u.current = u.N.head
    else if Cf(u, v) > 0 and u.h == v.h + 1
        Push(u, v)
    else u.current = v.next-neighbor
```

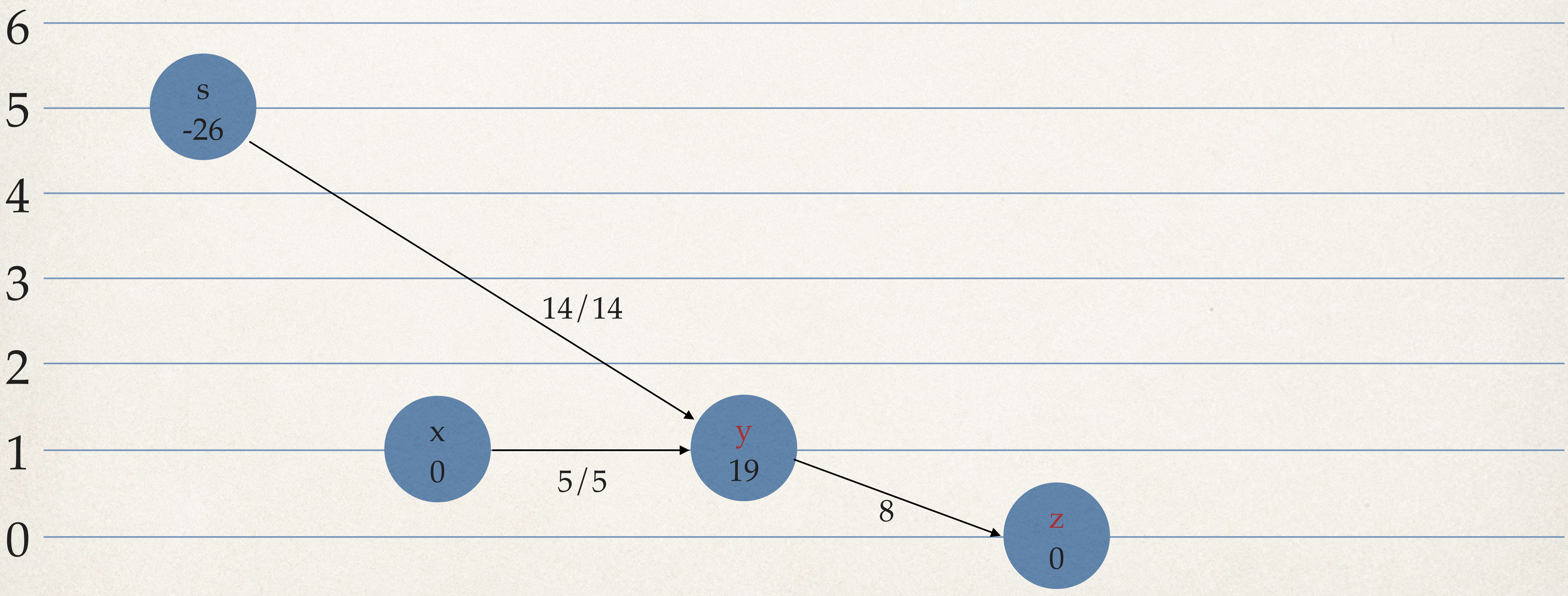
Basic Operation

- Discharge (y)



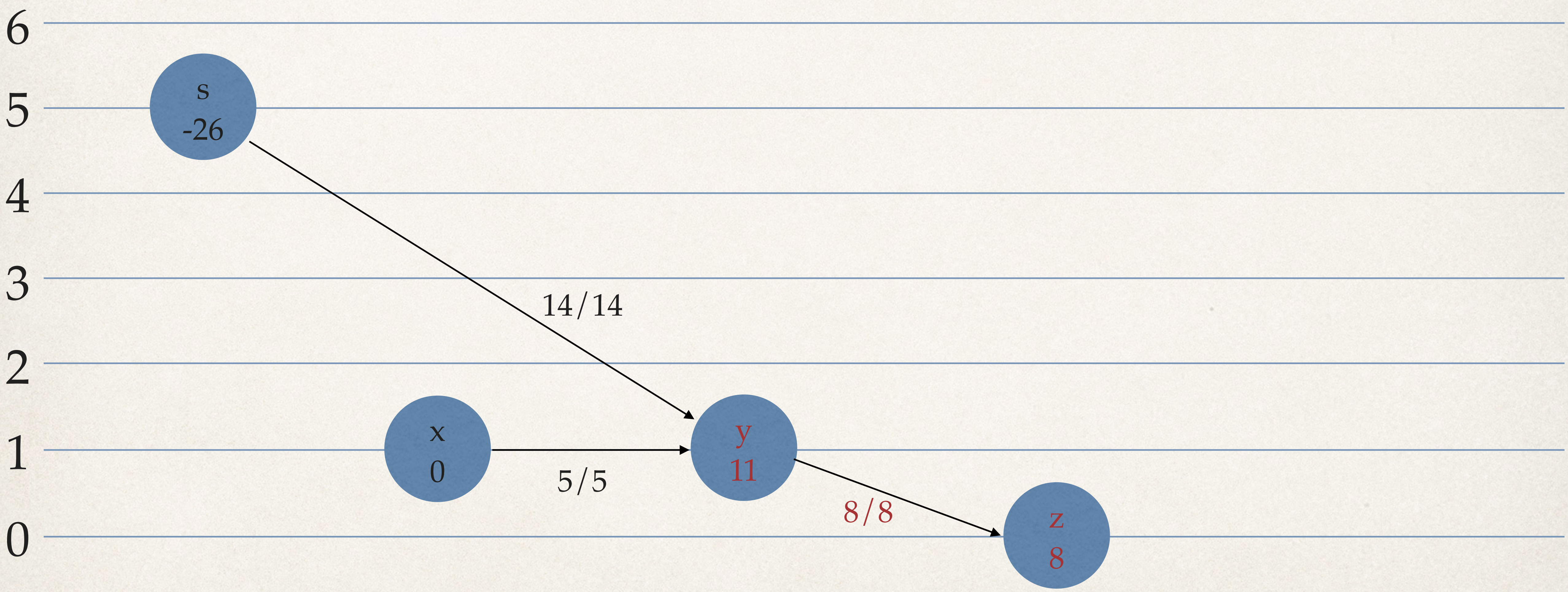
Basic Operation

- Discharge (y)



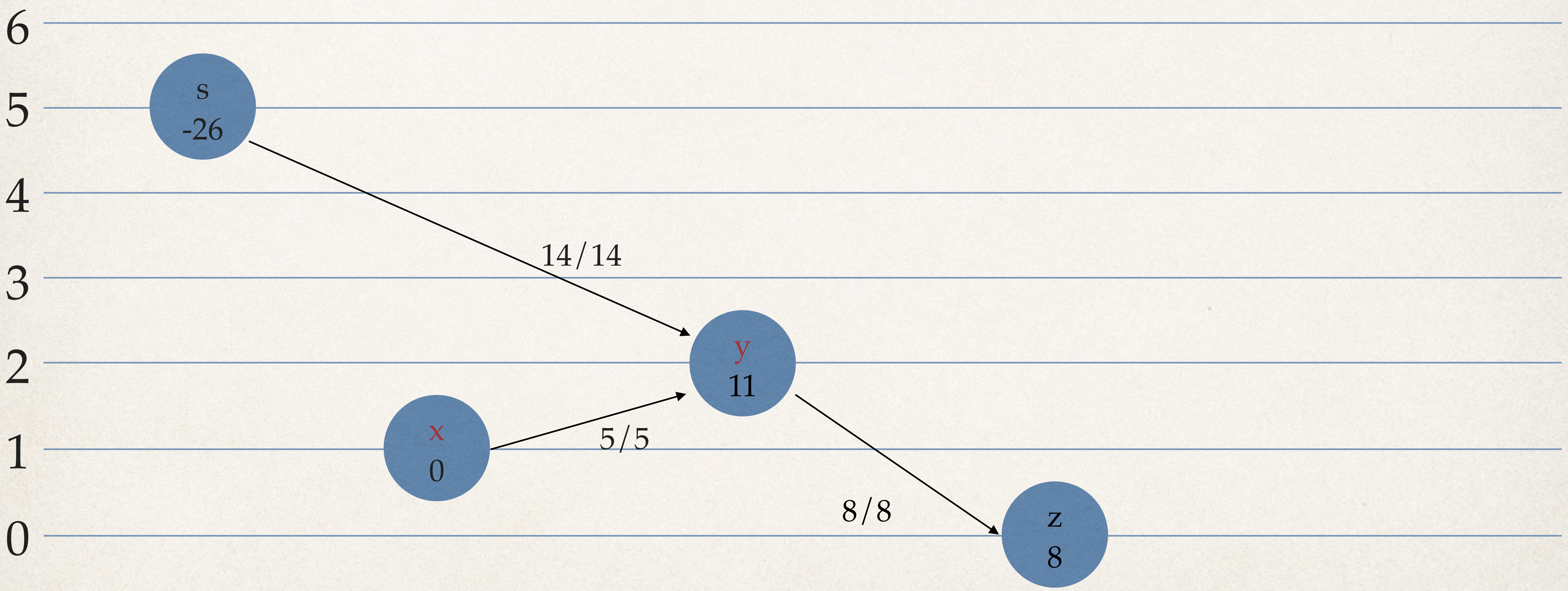
Basic Operation

- Discharge (y)



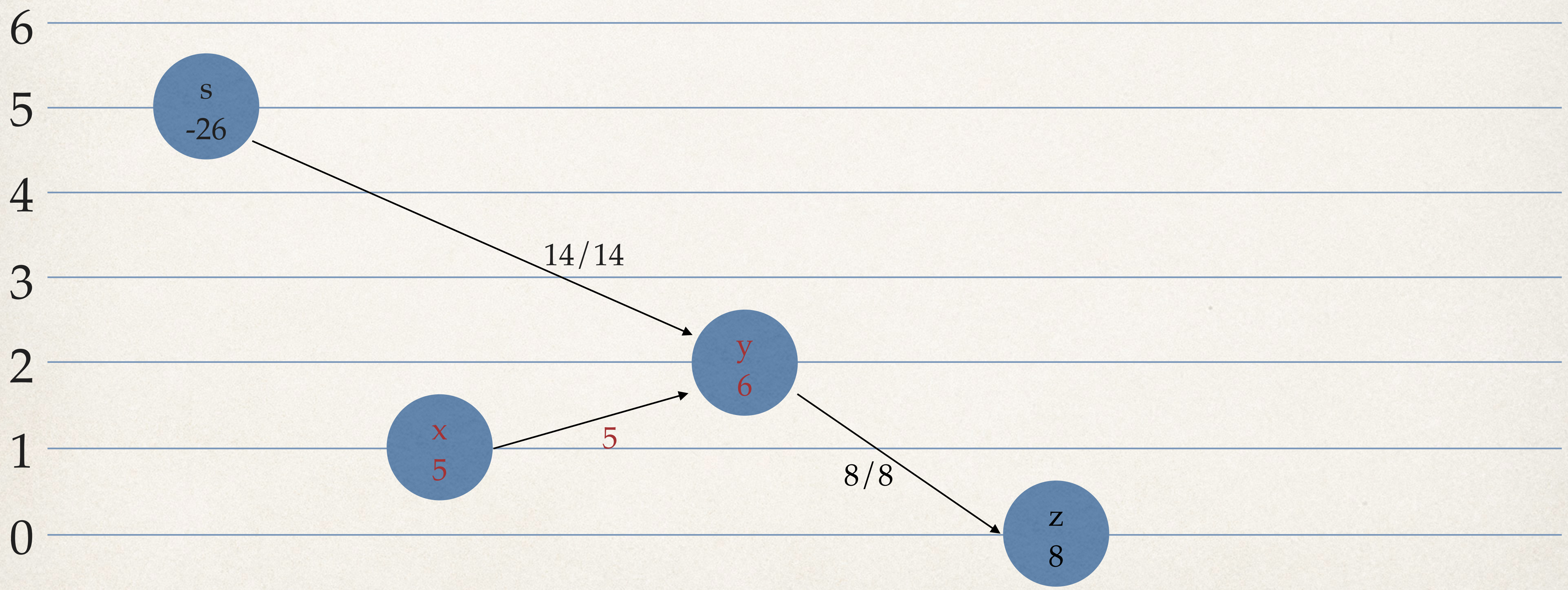
Basic Operation

- Discharge (y)



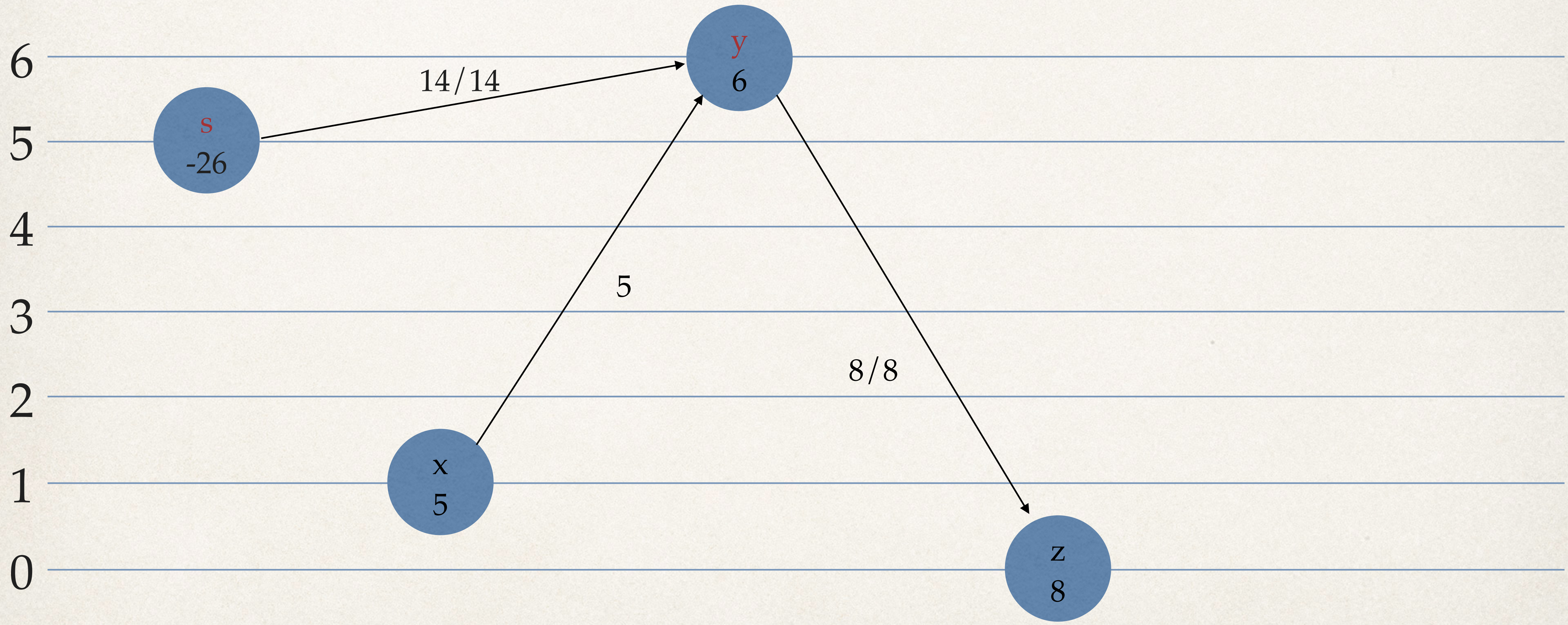
Basic Operation

- Discharge (y)



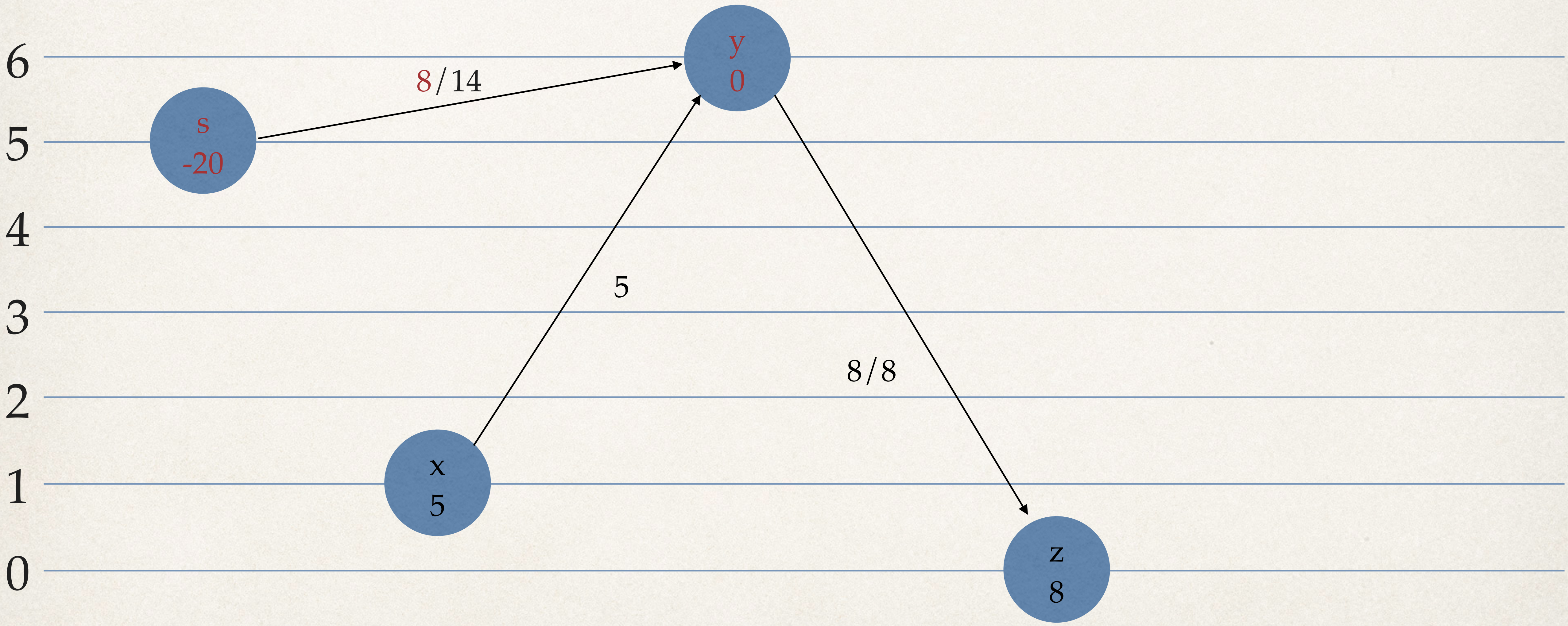
Basic Operation

- Discharge (y)



Basic Operation

- Discharge (y)



Basic Operation

- Initialize - Preflow

for each vertex $v \in G.V$

    $v.h = 0$

    $v.e = 0$

for each edge $(u, v) \in G.E$

    $(u, v).f = 0$

$s.h = |G.V|$

for each vertex $v \in s.Adj$

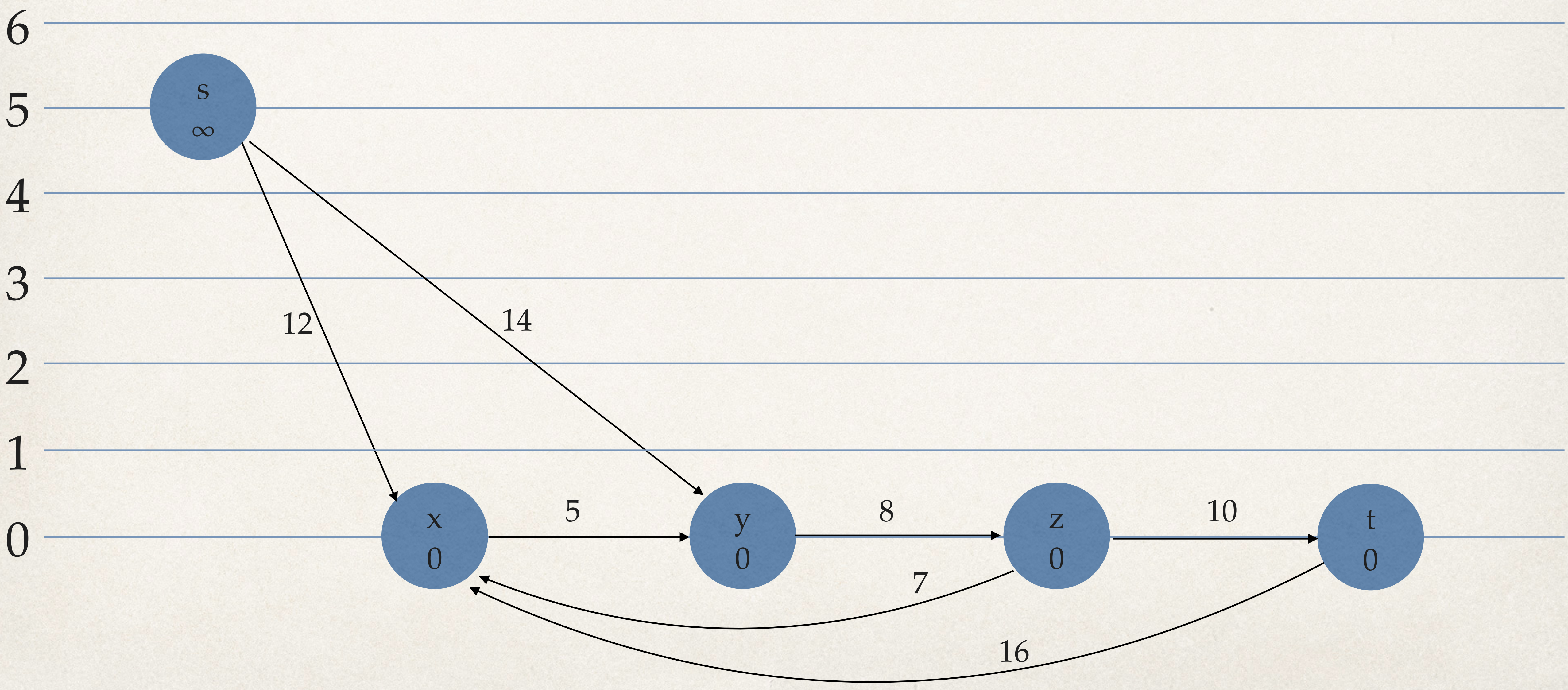
    $(s, v).f = c(s, v)$

    $v.e = c(s, v)$

    $s.e = s.e - c(s, v)$

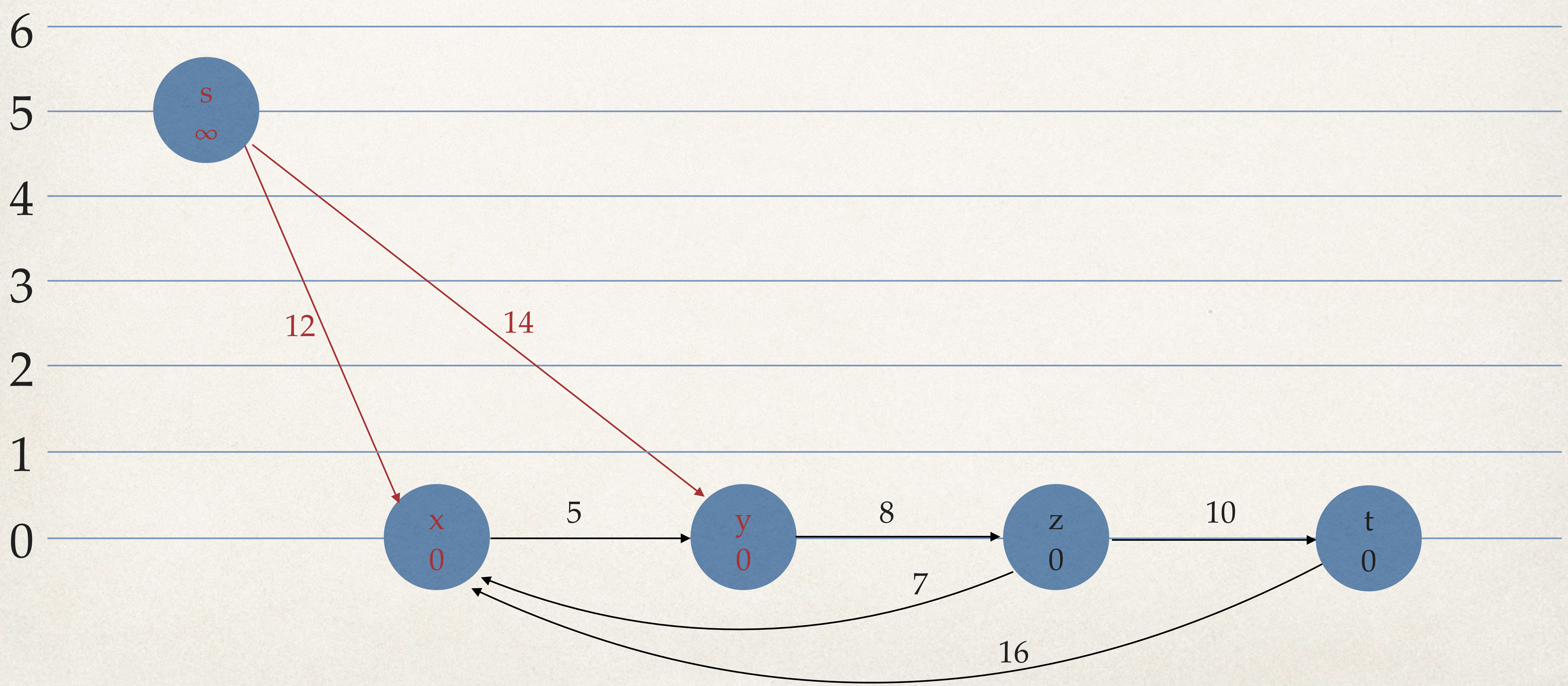
Basic Operation

- Preflow



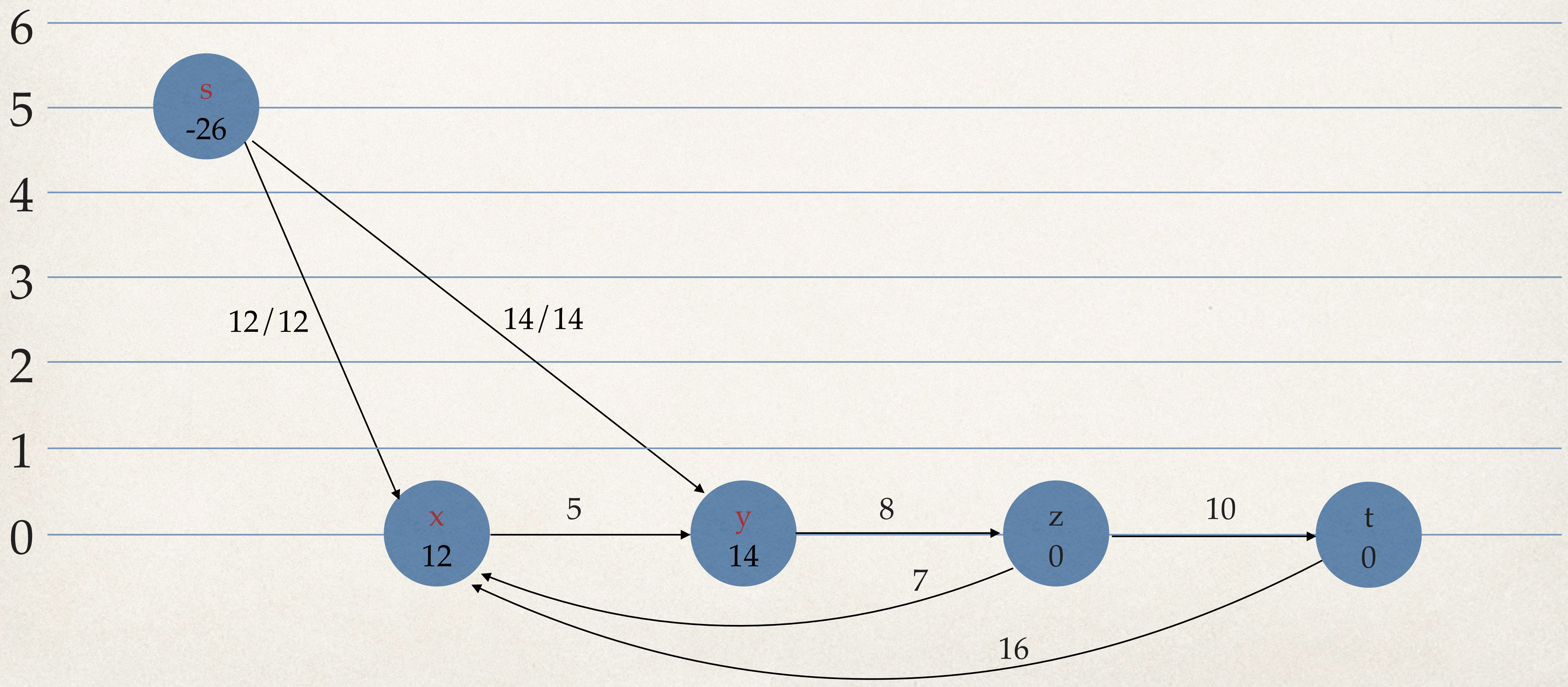
Basic Operation

- Preflow



Basic Operation

- Preflow



The Push-Relabel Algorithm

The Push-Relabel algorithm

Initialize - Preflow (G, s)

L = G.V - {s, t} (A linked list in any order)

for each vertex u ∊ G.V - {s, t}

    u.current = u.N.head

While u != NIL

    old-height = u.h

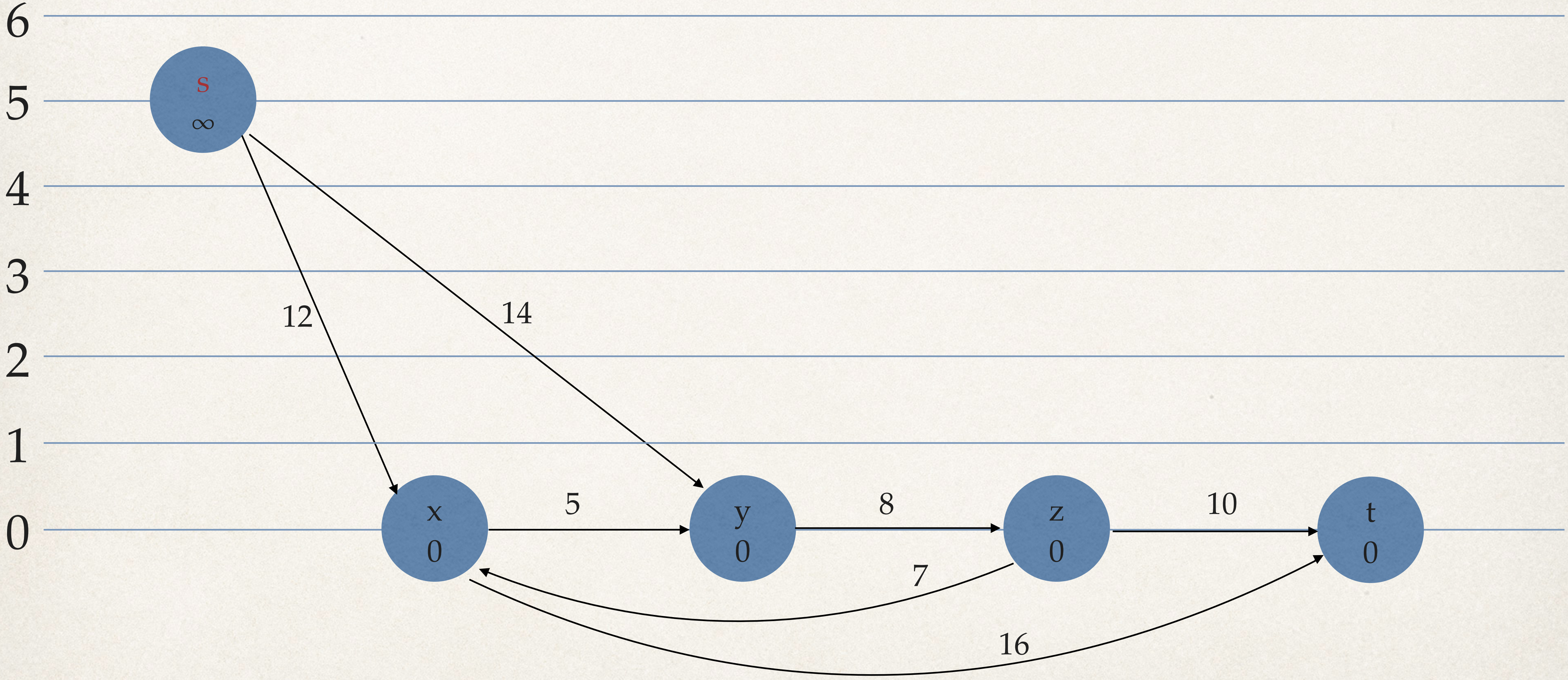
    Discharge(u)

    if u.h > old-height

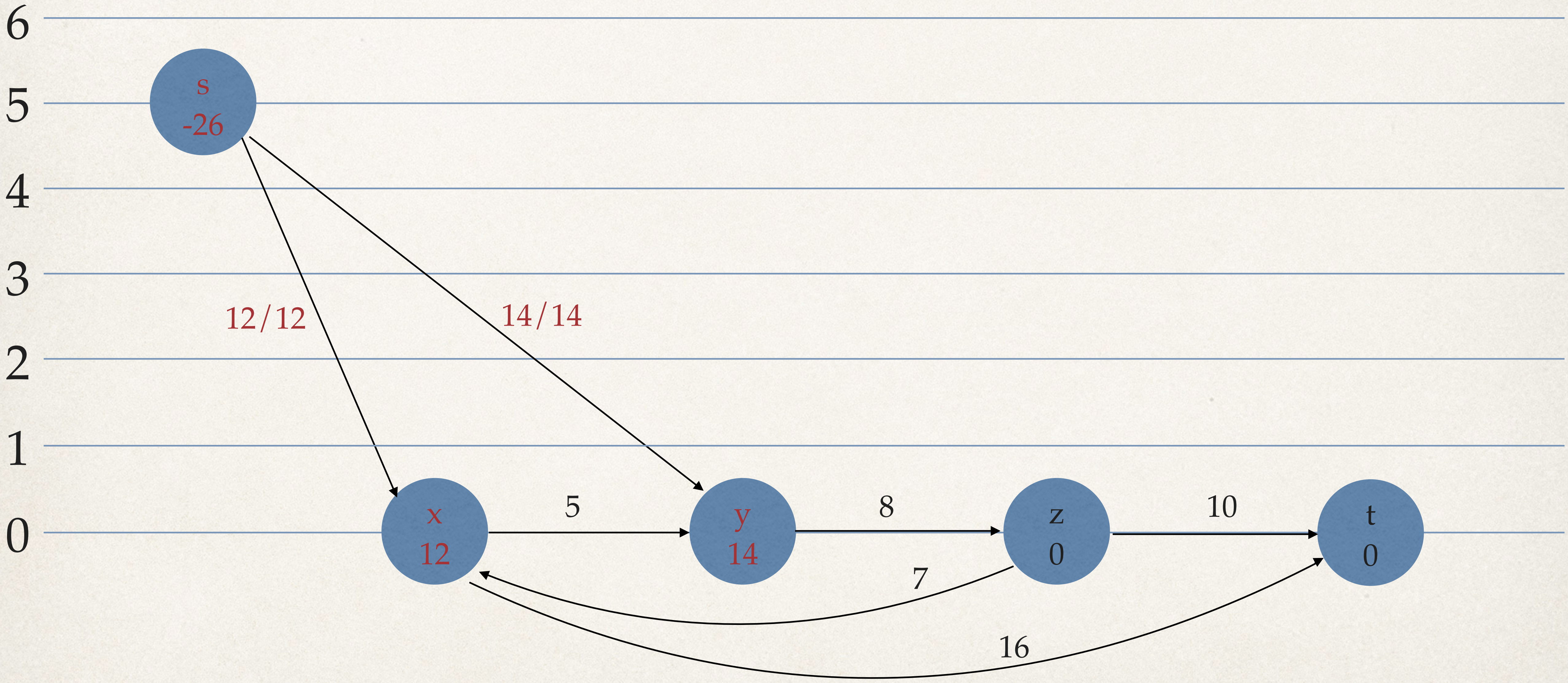
        move u to the front of the list L

    u = u.next

The Push-Relabel algorithm

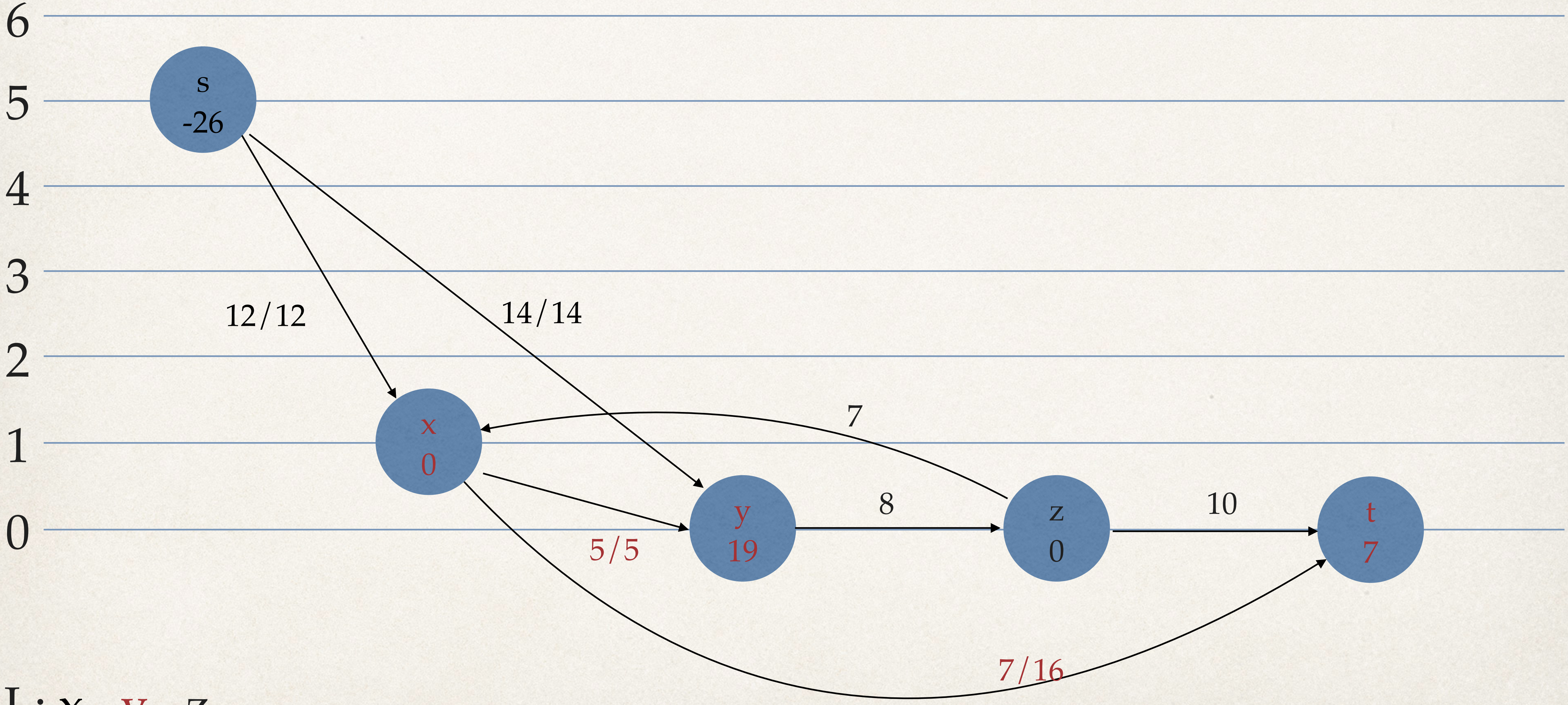


The Push-Relabel algorithm



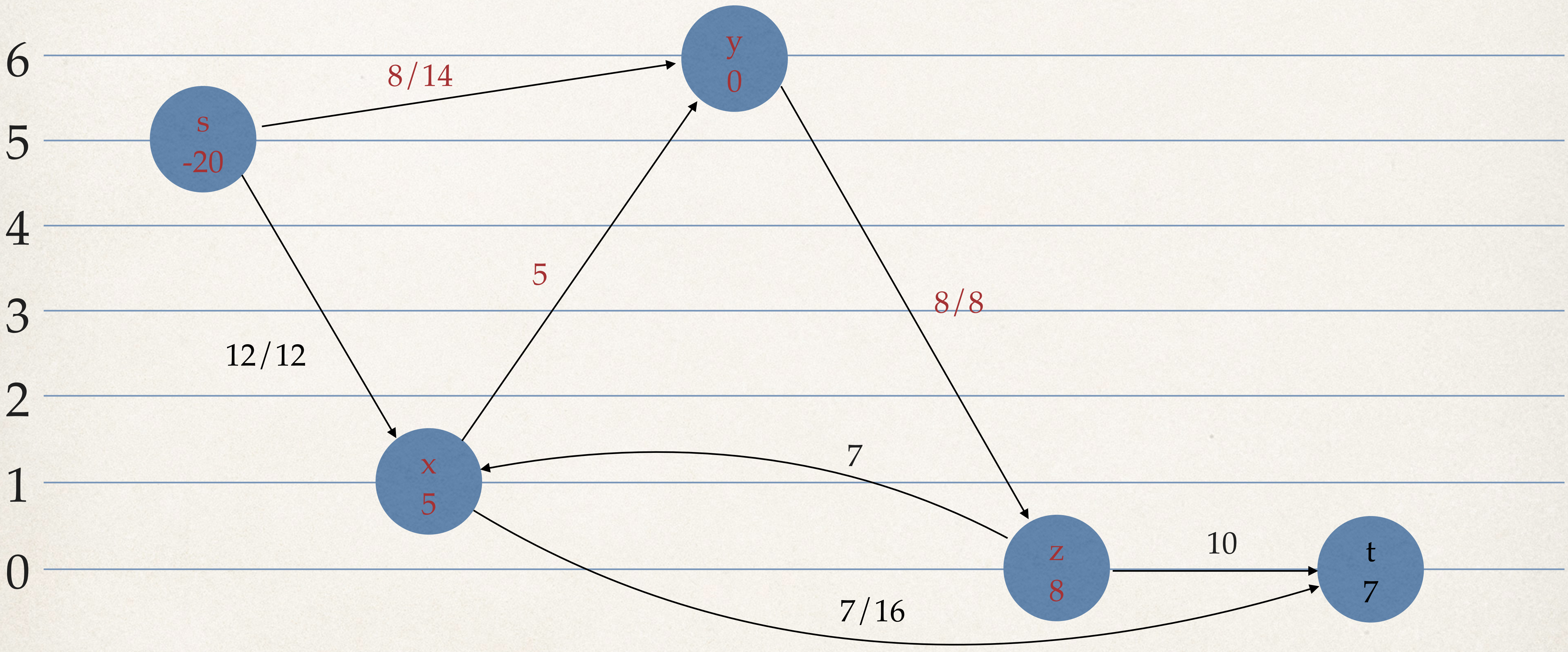
L: x y x

The Push-Relabel algorithm



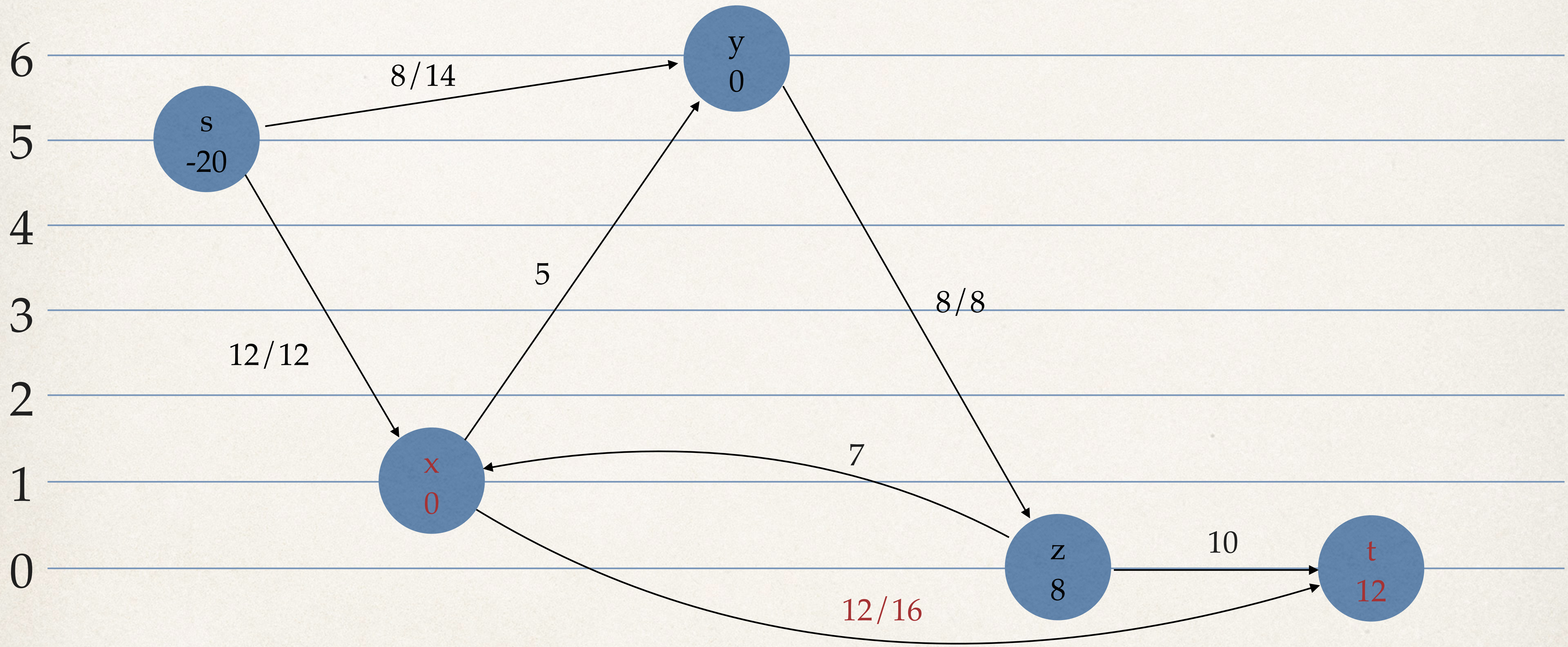
L: x y z

The Push-Relabel algorithm



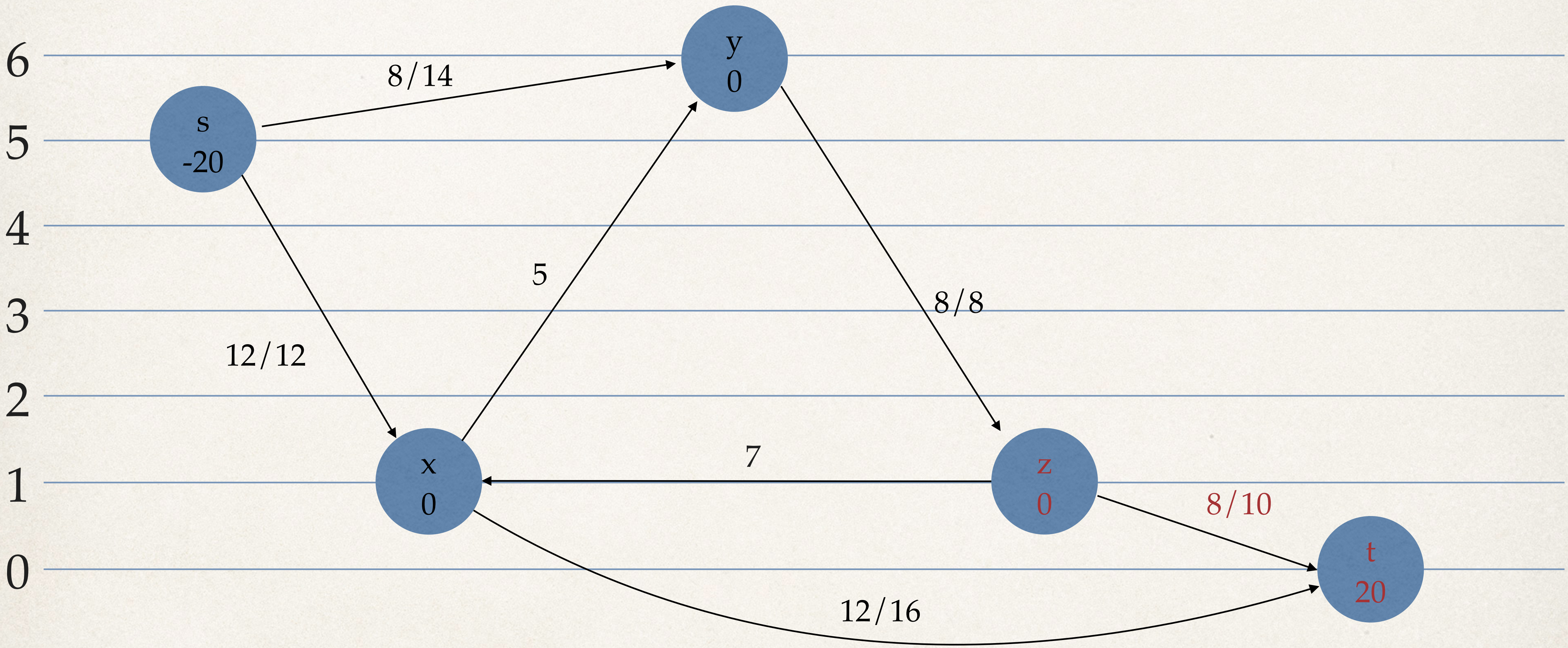
L: y x z

The Push-Relabel algorithm



L: y x z

The Push-Relabel algorithm



L: z y x

The Push-Relabel algorithm

- Analysis

Lemma 26.20

Let G = (V, E) be a flow network with source s and sink t. At any time during the execution of Push-Relabel on G, we have $u.h <= 2|V|-1$ for all vertices $u \in V$

The Push-Relabel algorithm

- Analysis

Proof

When a vertex u is relabelled, it is overflowing and there will be a simple path p from u to s in G_f.

$p = \{v_0, v_1, \ldots v_k\}$, where $v_0 = u$, $v_k = s$, and $k <= |V|-1$

We have $v_i.h <= v_{i+1}.h + 1$.

Expanding it to the path p gets

u.h $= v_0.h <= v_k.h + k <= s.h + (|V|-1) =$ **$2|V|-1$**

The Push-Relabel algorithm

- Analysis

Corollary 26.21 (Bound on relabel operations)
Let $G = (V, E)$ be a flow network with source s and sink t. During the execution of Push-Relabel on G, the number of relabel operation is at most $2|V|-1$ per vertex and at most $(2|V|-1)(|V|-2) < 2|V|^2$ overall.

The Push-Relabel algorithm

- Analysis

Proof

Only the $|V|$-2 vertices in V-{s,t} may be relabelled. Let u $\in$ V-{s,t}.

The height of u is at most $2|V|-1$ by Lemma 26.20.

So each vertex is relabelled at most $2|V|-1$ times.

The total number of relabel operations performed is at most

$(2|V|-1)(|V|-2) < 2|V|^2$

The Push-Relabel algorithm

- Analysis

Theorem 26.30
The running time of Push-Relabel on any flow network $G = (V, E)$ is $O(V^3)$

The Push-Relabel algorithm

- Analysis

Proof
Let us consider a "phase" is the time between two consecutive relabel operations.
There are $O(V^2)$ phases, since there are $O(V^2)$ relabel operations.(By corollary 26.21)
Each phase consists of at most $|V|$ calls to Discharge.
The number of times **Discharge** is called is $O(V^3)$.

The Push-Relabel algorithm

- Analysis

Proof
Total number of saturating push operations is O(VE).
And there is only one non saturating push per call to Discharge.
As observed, total time spent on non saturating pushes is $O(V^3)$.
The running time of Push-Relabel is therefore $O(V^3 + VE)$,
which is $O(V^3)$

Thanks!