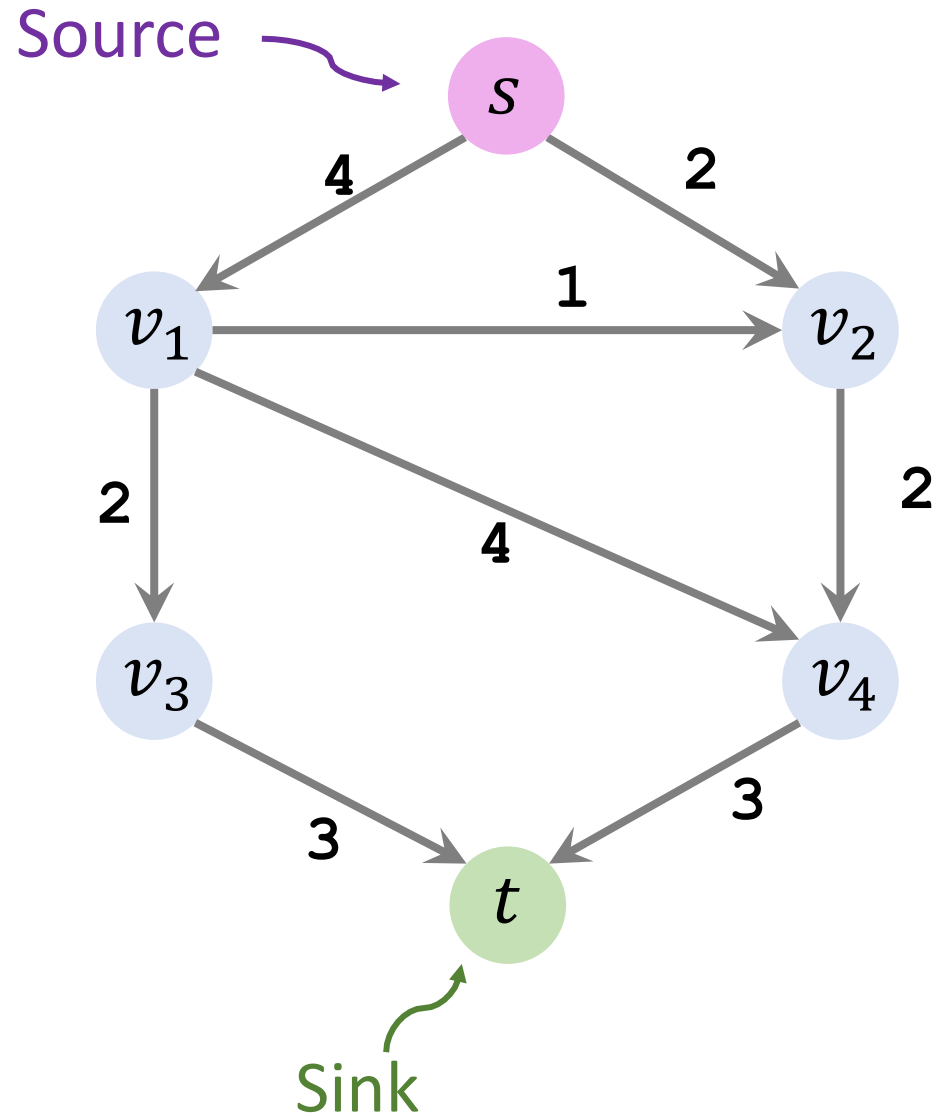


# Network Flow Problems

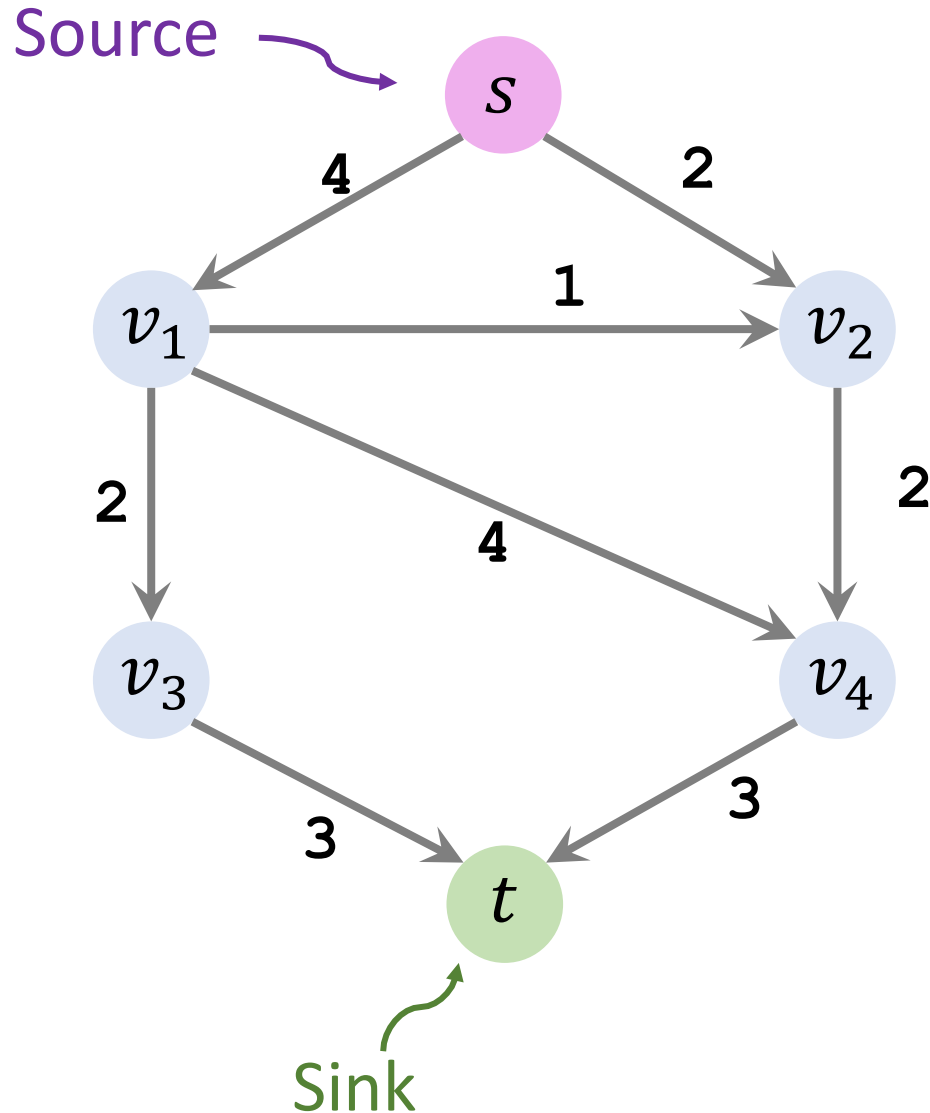
**Shusen Wang**

<http://wangshusen.github.io/>

# Maximum Flow Problem

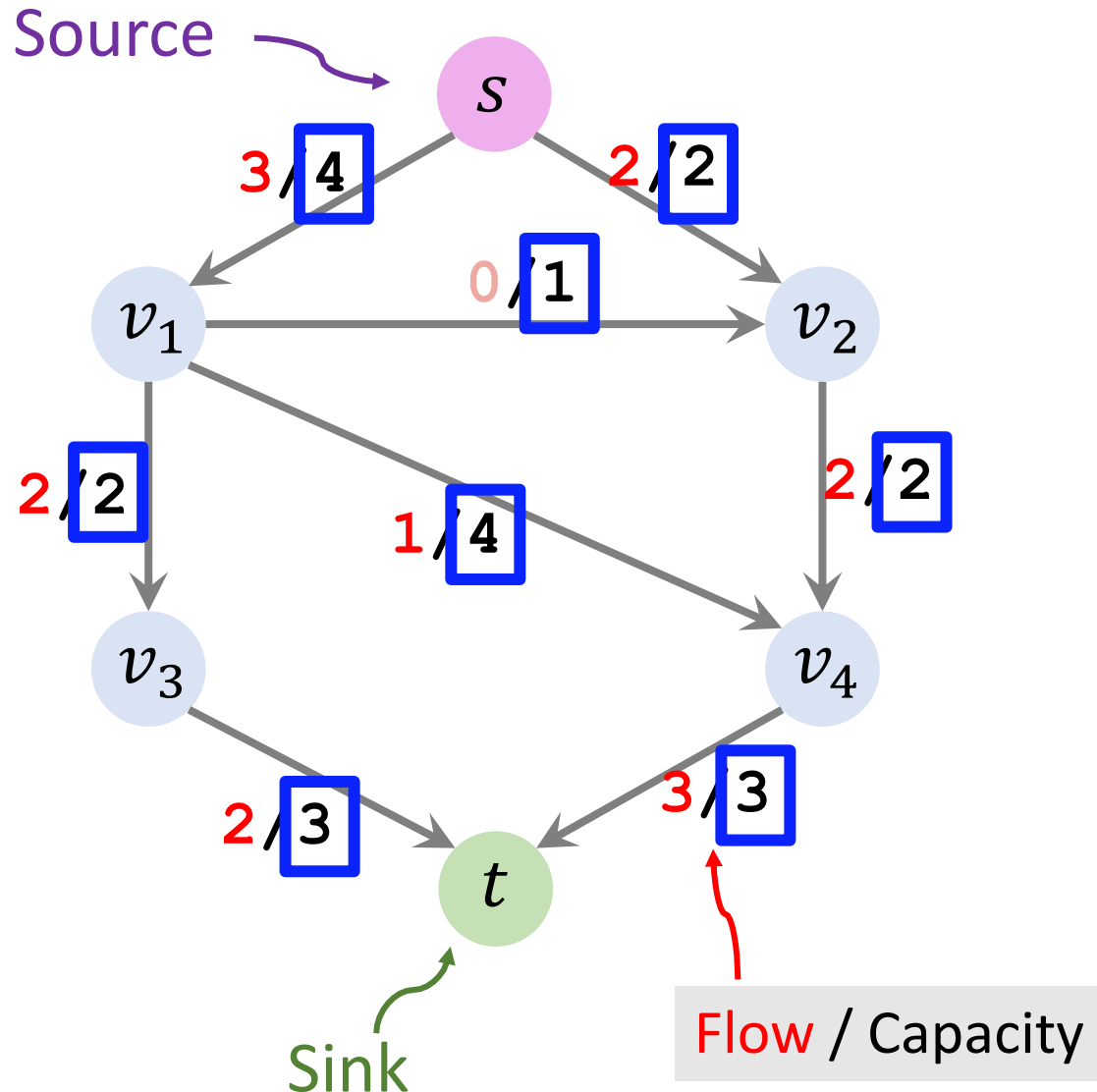


# Maximum Flow Problem



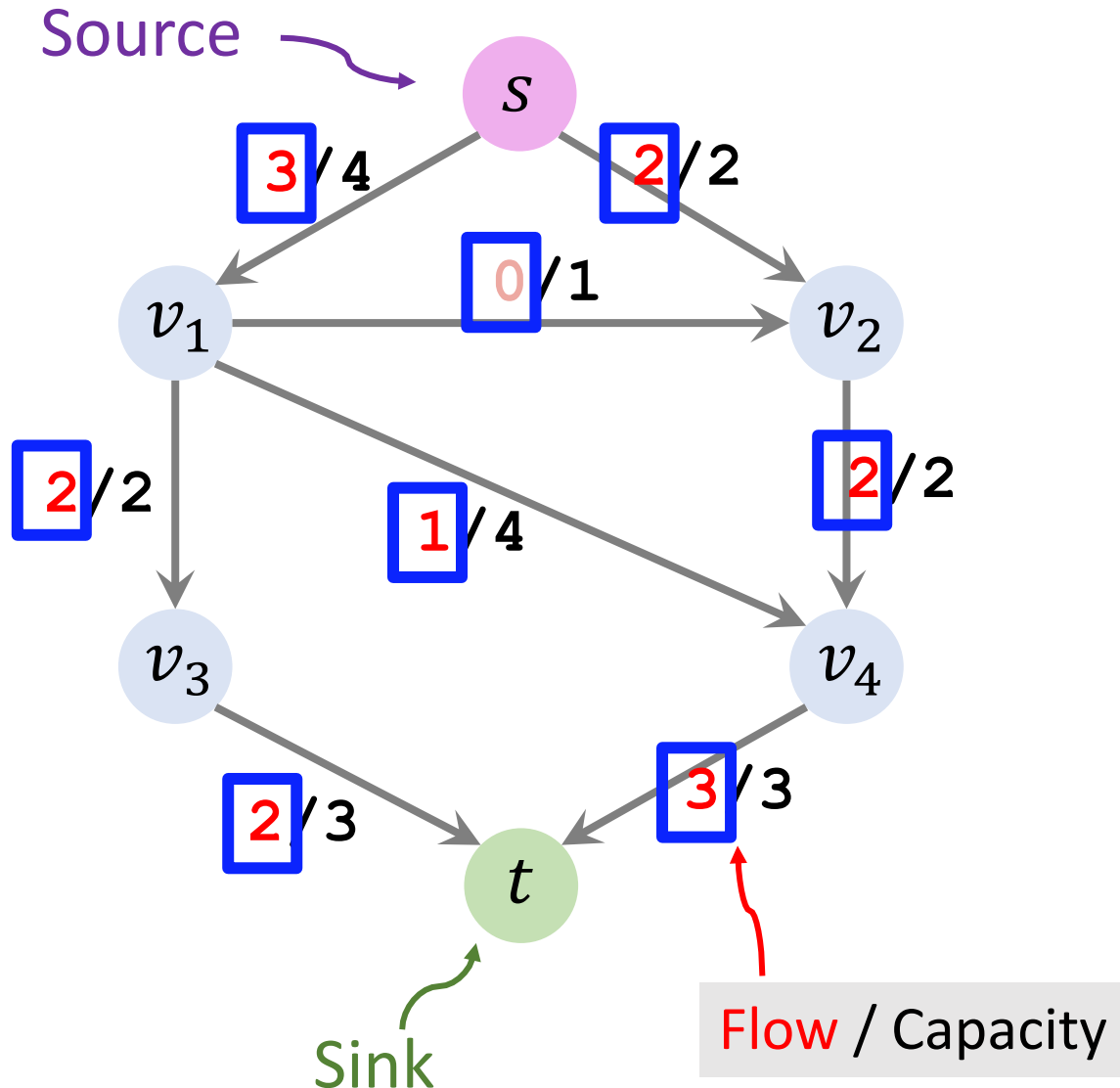
- Send water from the source  $s$  to the sink  $t$ .
- The edges are pipes which have certain capacities, e.g.,  $4 \text{ m}^3/\text{s}$ .
- How much water can flow from the source  $s$  to the sink  $t$  at most?

# Maximum Flow Problem



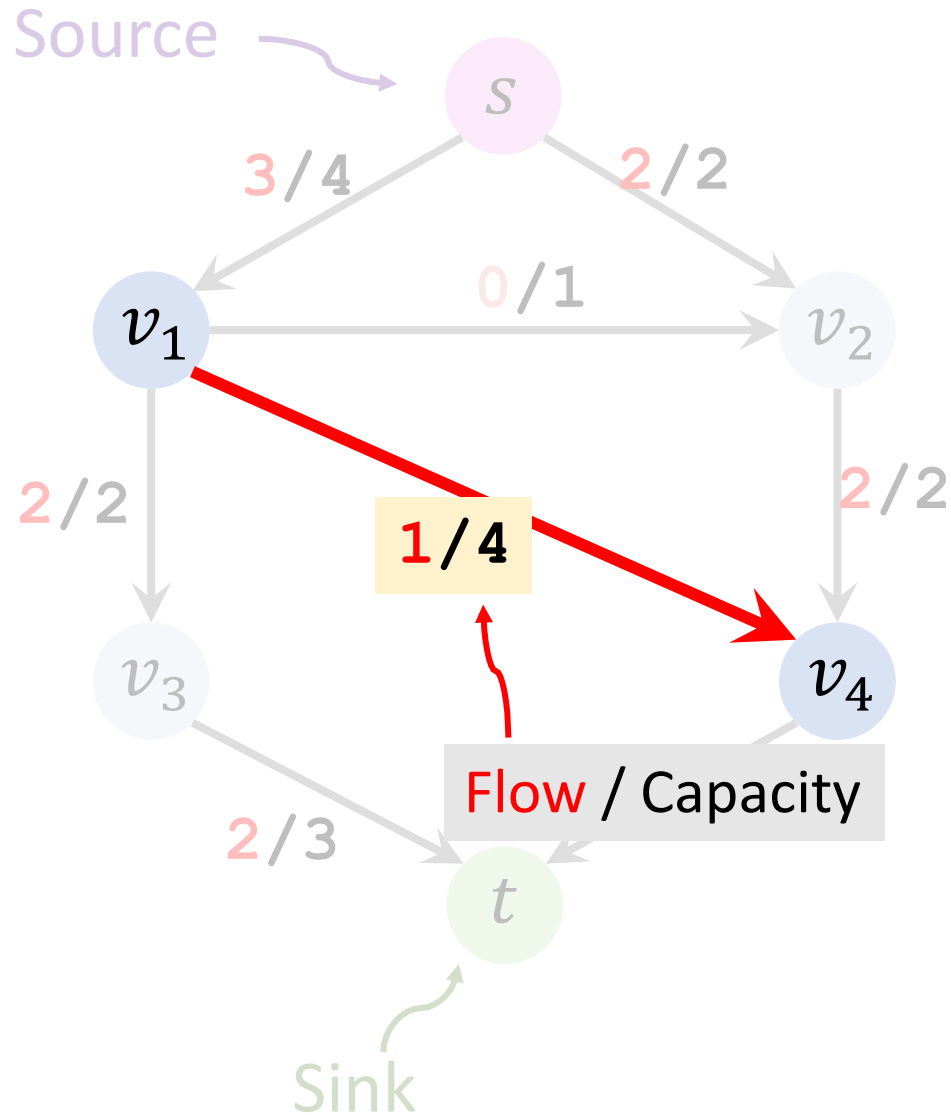
- Send water from the source  $s$  to the sink  $t$ .
- The edges are pipes which have certain capacities, e.g.,  $4 \text{ m}^3/\text{s}$ .
- How much water can flow from source  $s$  to the sink  $t$  at most?

# Maximum Flow Problem



- Send water from the source  $s$  to the sink  $t$ .
- The edges are pipes which have certain capacities, e.g.,  $4 \text{ m}^3/\text{s}$ .
- How much water can flow from source  $s$  to the sink  $t$  at most?
- Flow = 5.

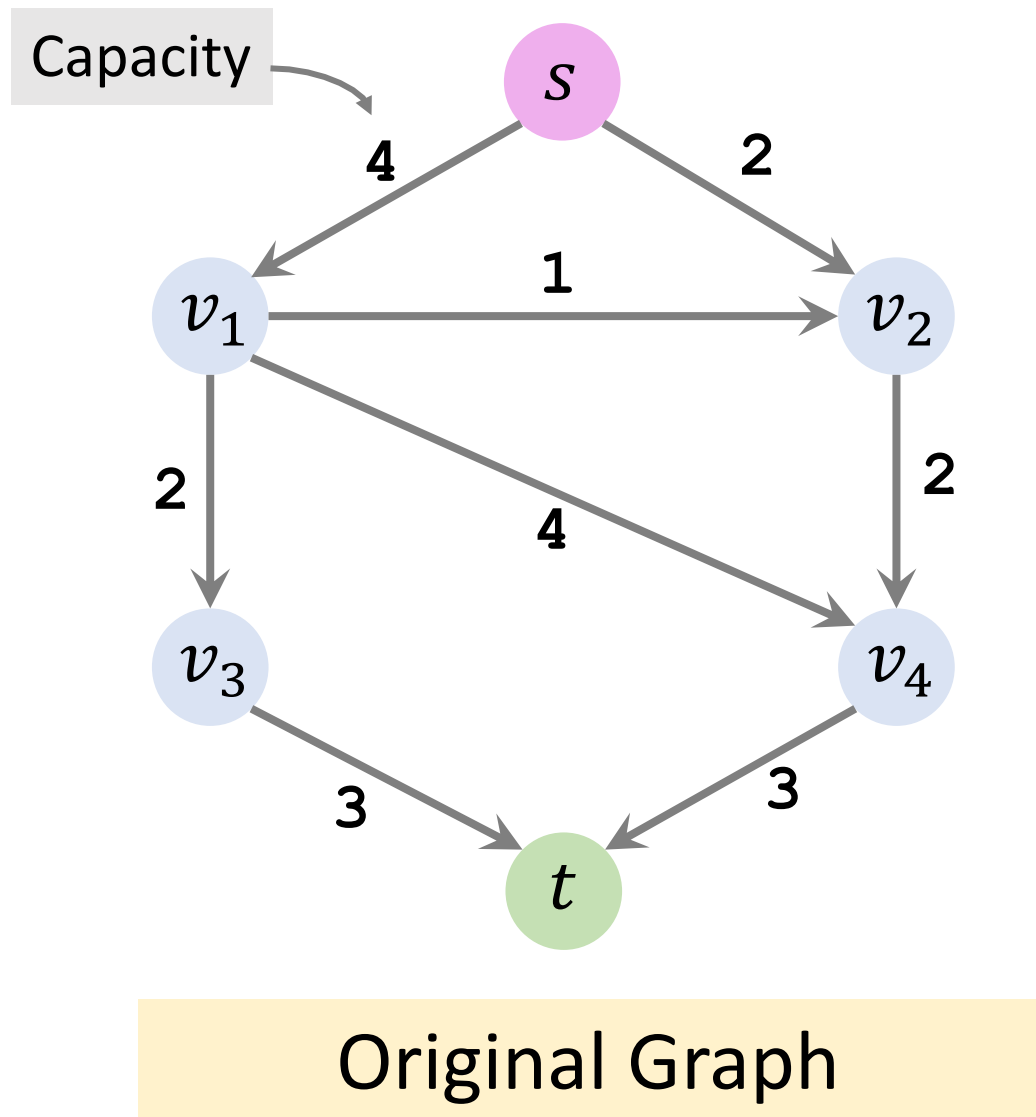
# Maximum Flow Problem



- Capacity of the red pipe is  $4 \text{ m}^3/\text{s}$ .
- A **flow** of  $1 \text{ m}^3/\text{s}$  goes through the red pipe.
- It has a **residual** of  $3 \text{ m}^3/\text{s}$ .

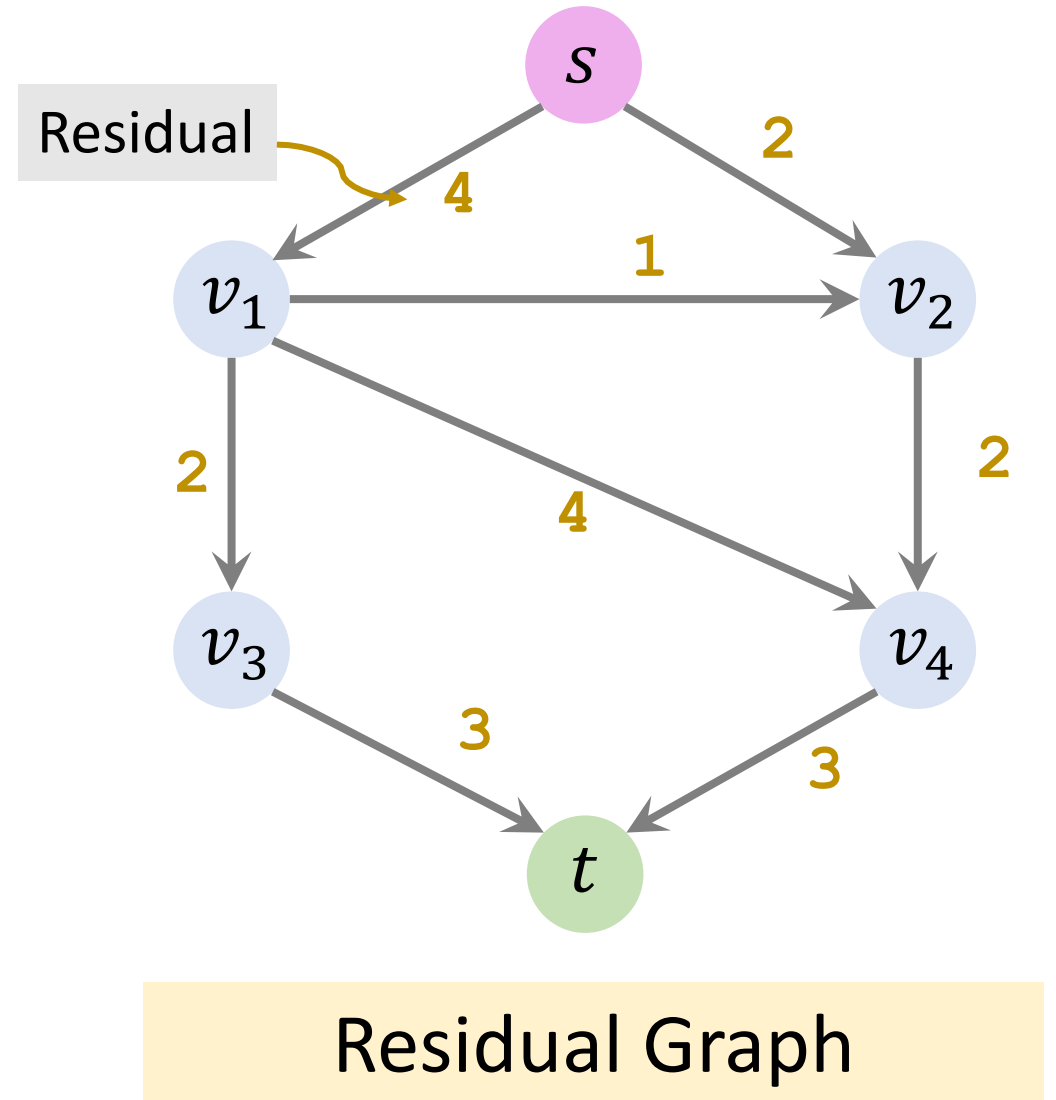
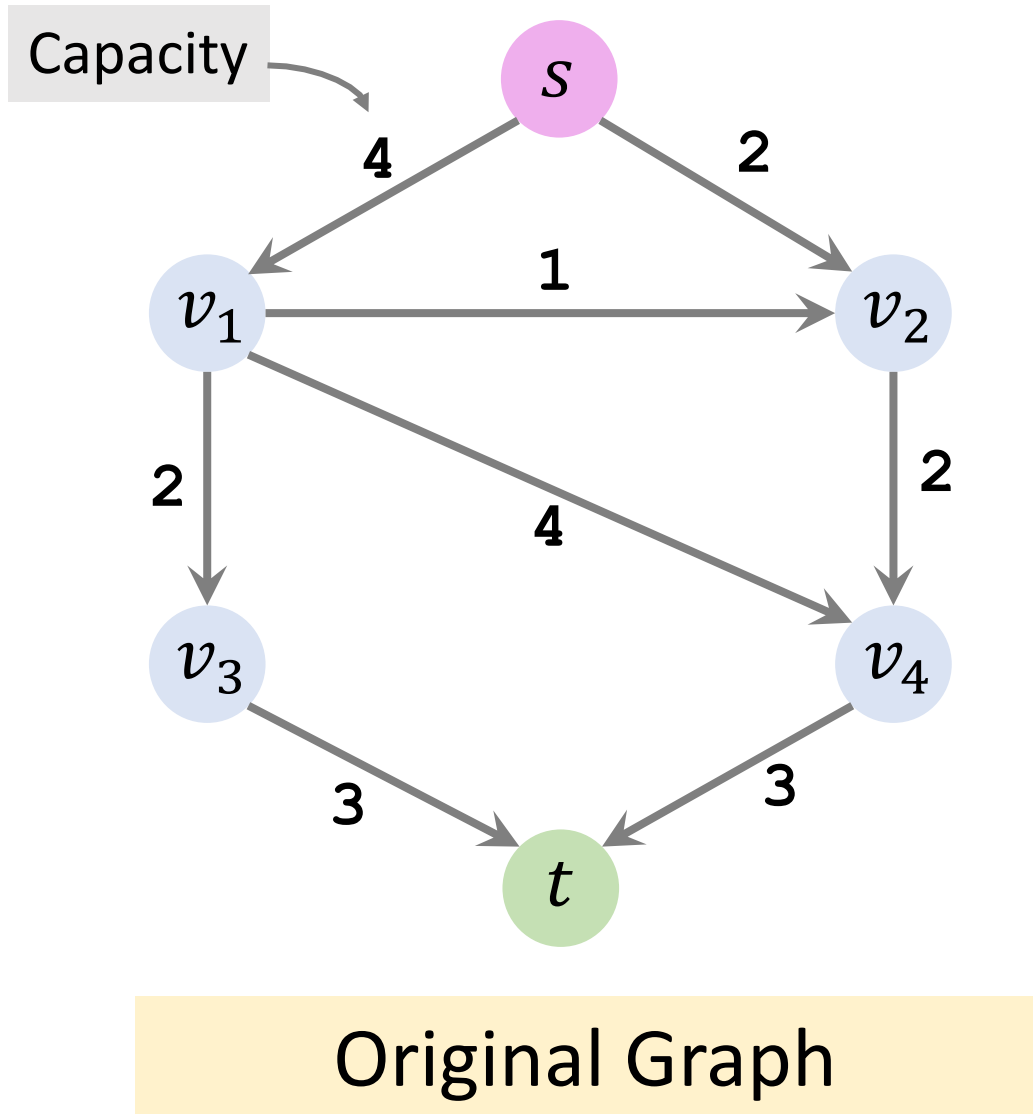
# **Naïve Algorithm**

# Initialization

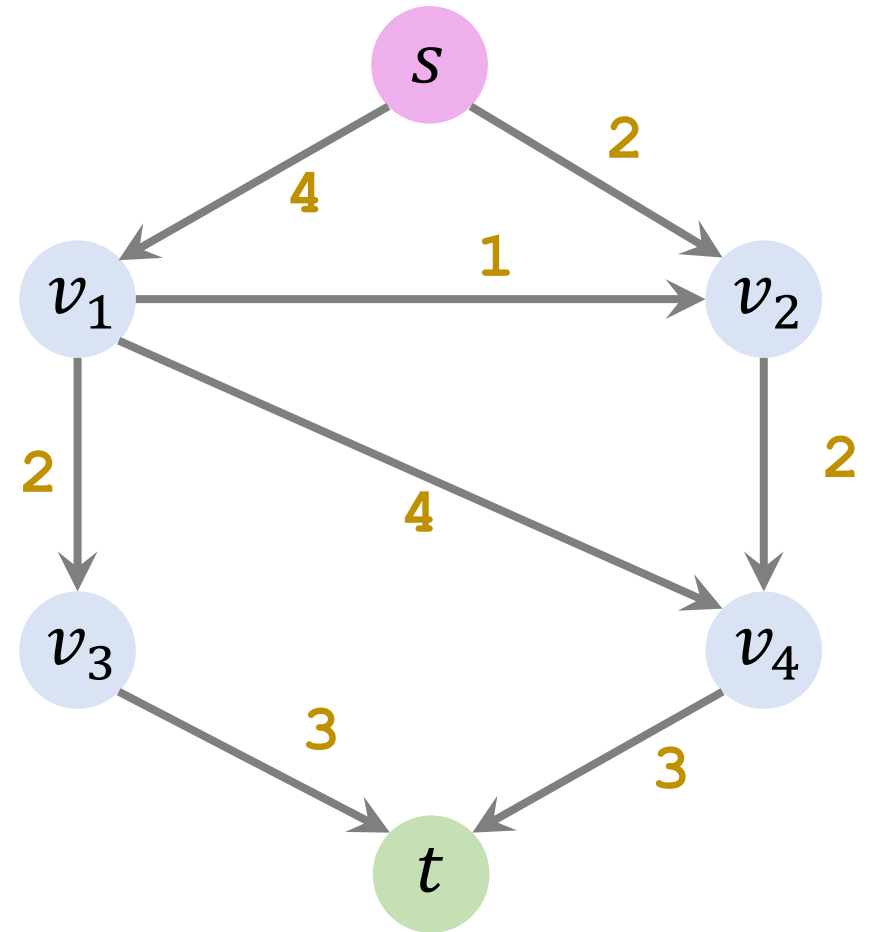
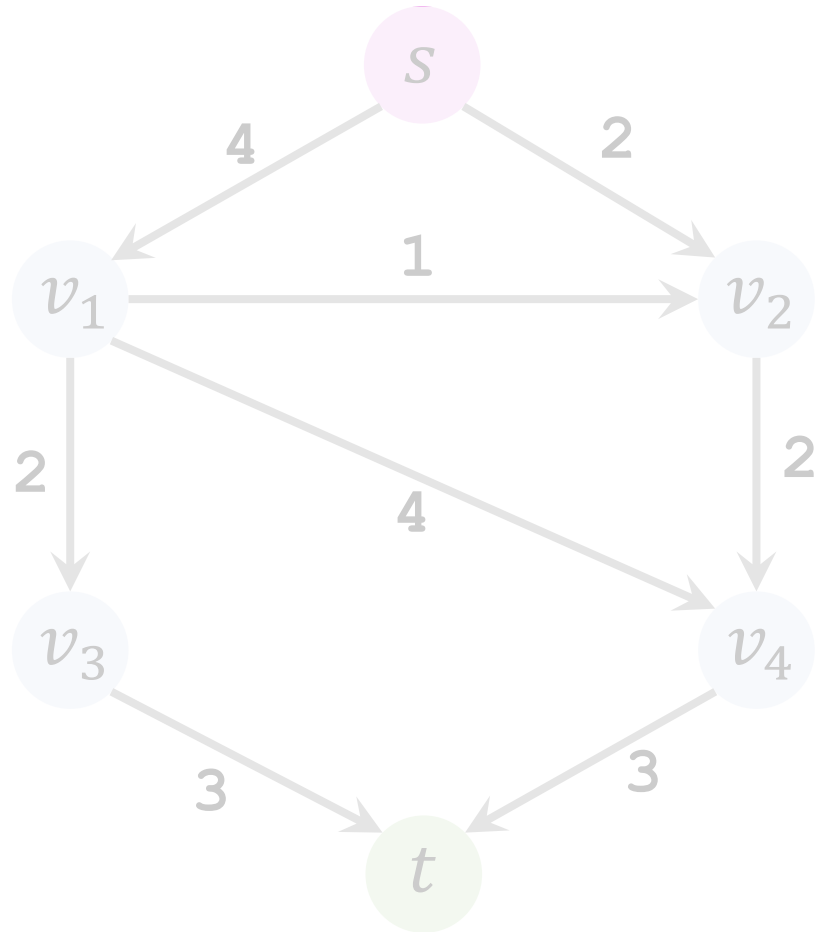




# Initialization

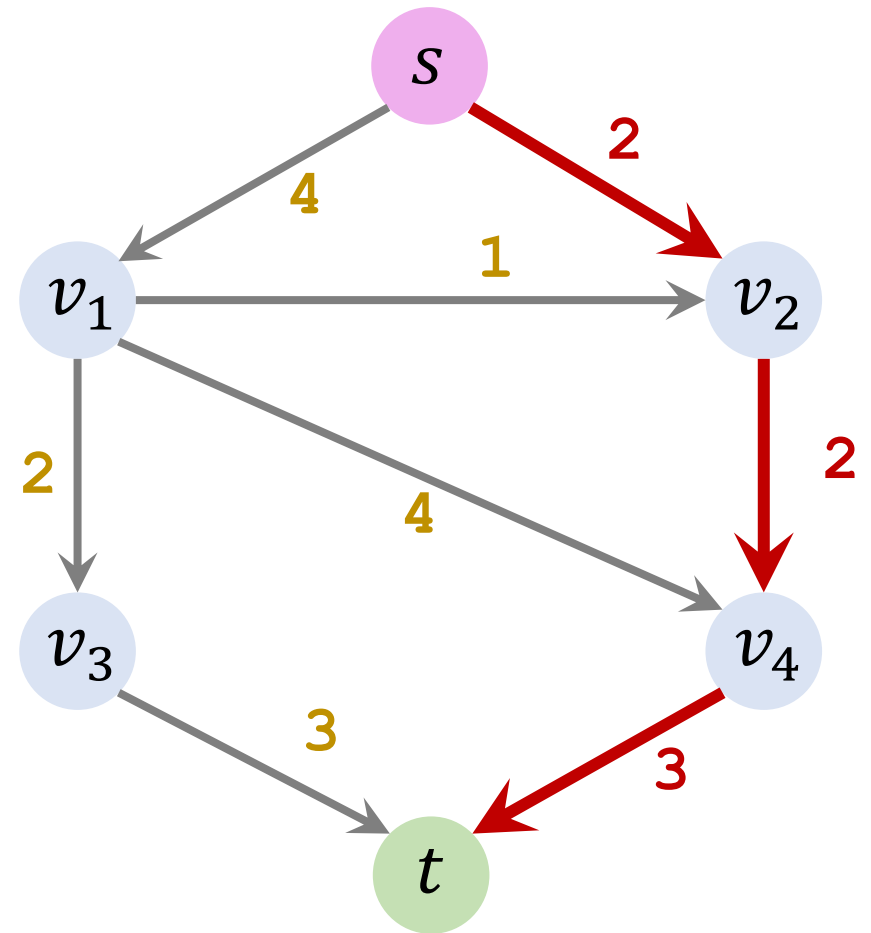
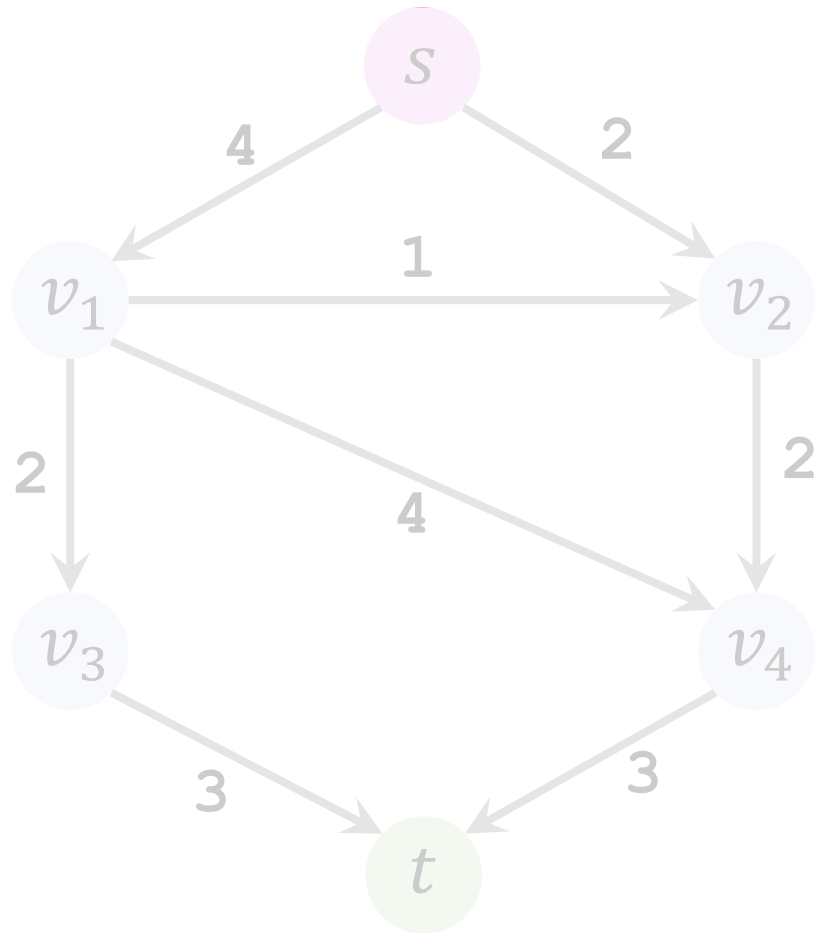


# Iteration 1: Find an augmenting path



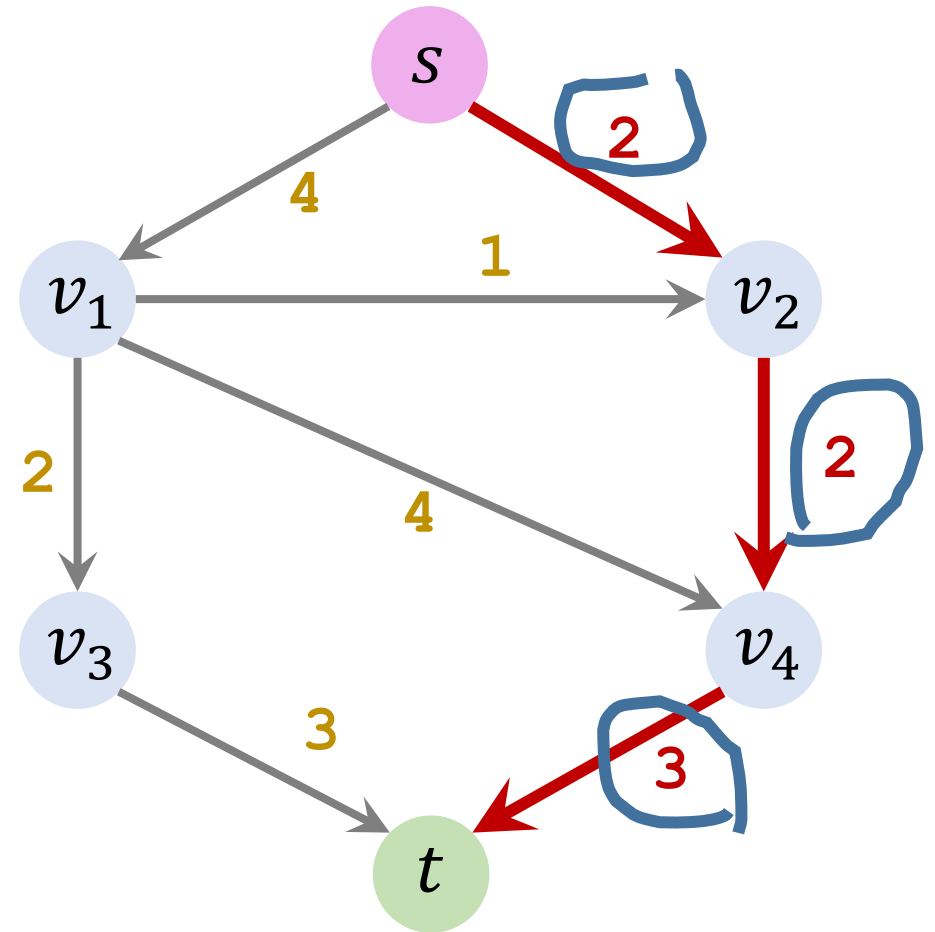
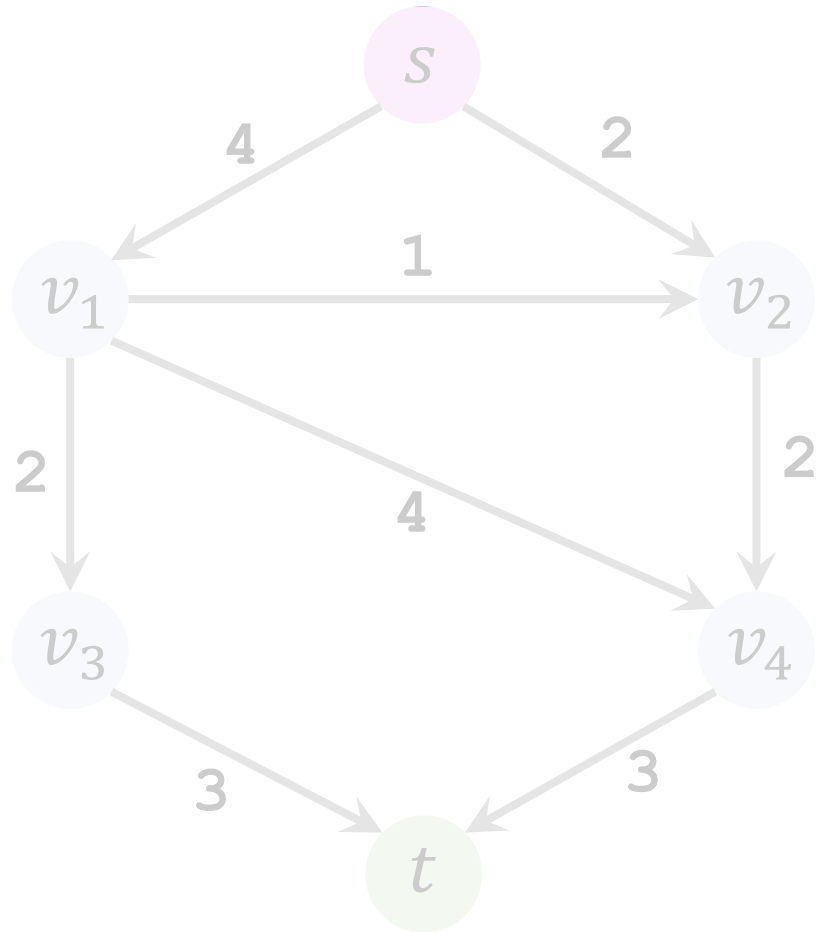
**Augmenting path:** a path from  $s$  to  $t$  that does not contain cycles.

# Iteration 1: Find an augmenting path



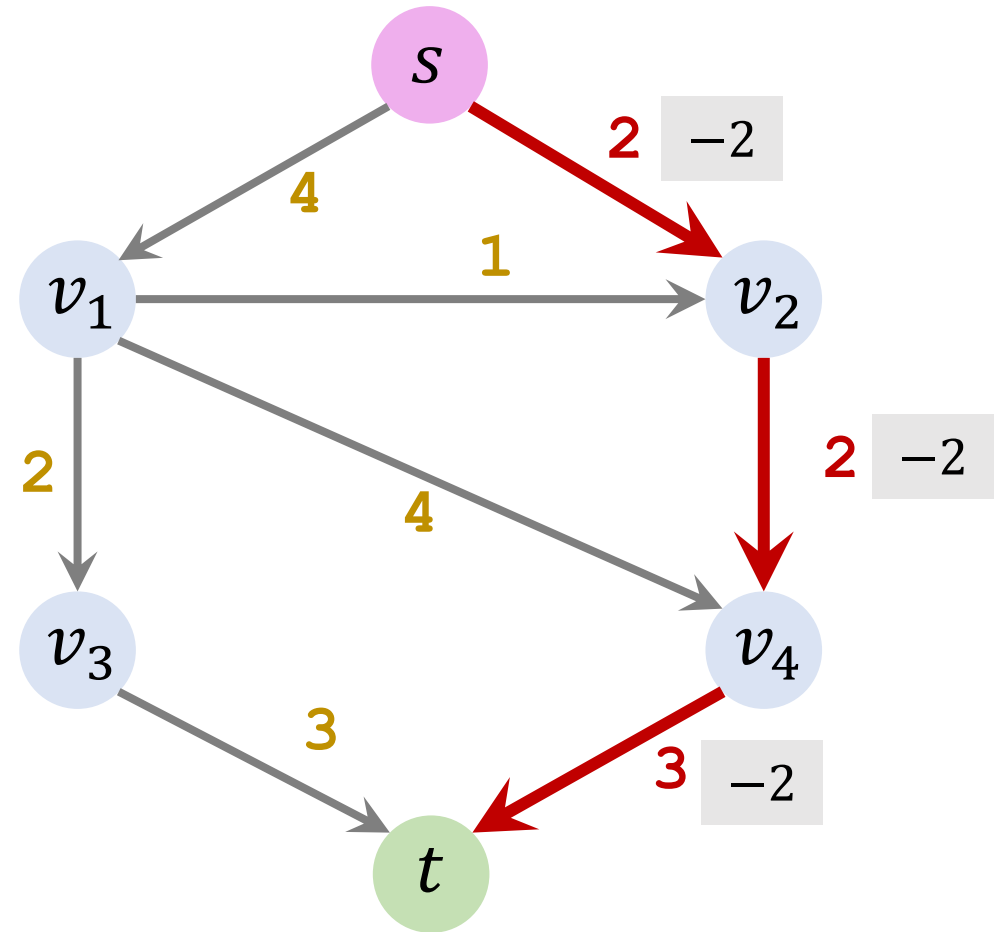
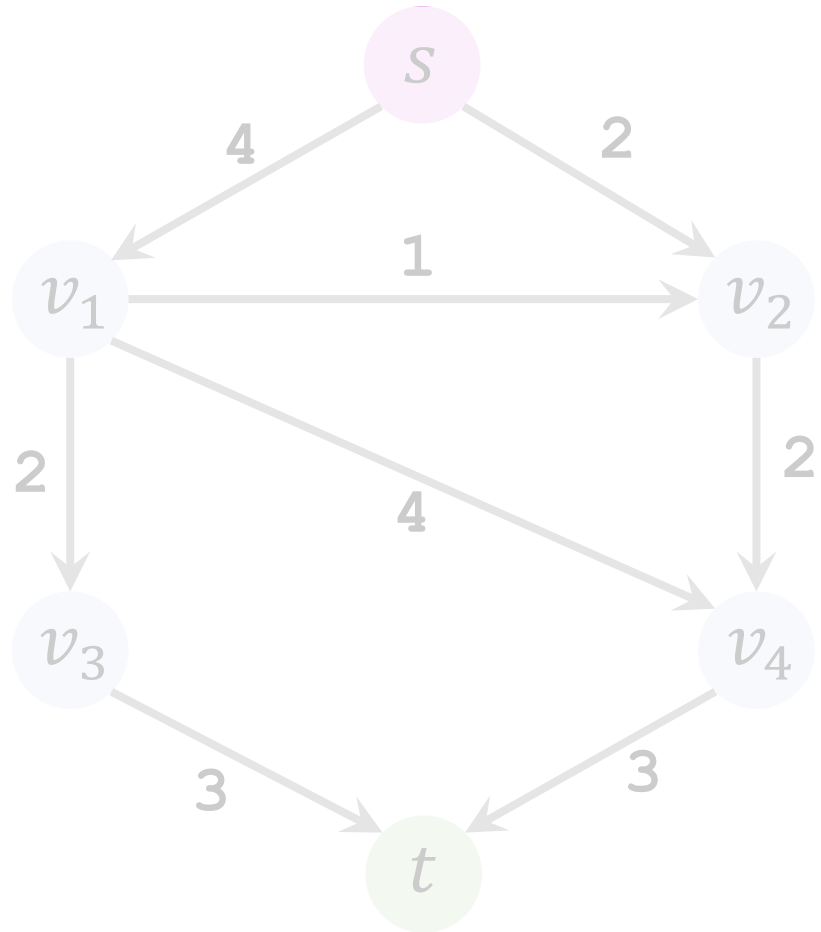
Found path  $s \rightarrow v_2 \rightarrow v_4 \rightarrow t$ .

# Iteration 1: Find an augmenting path



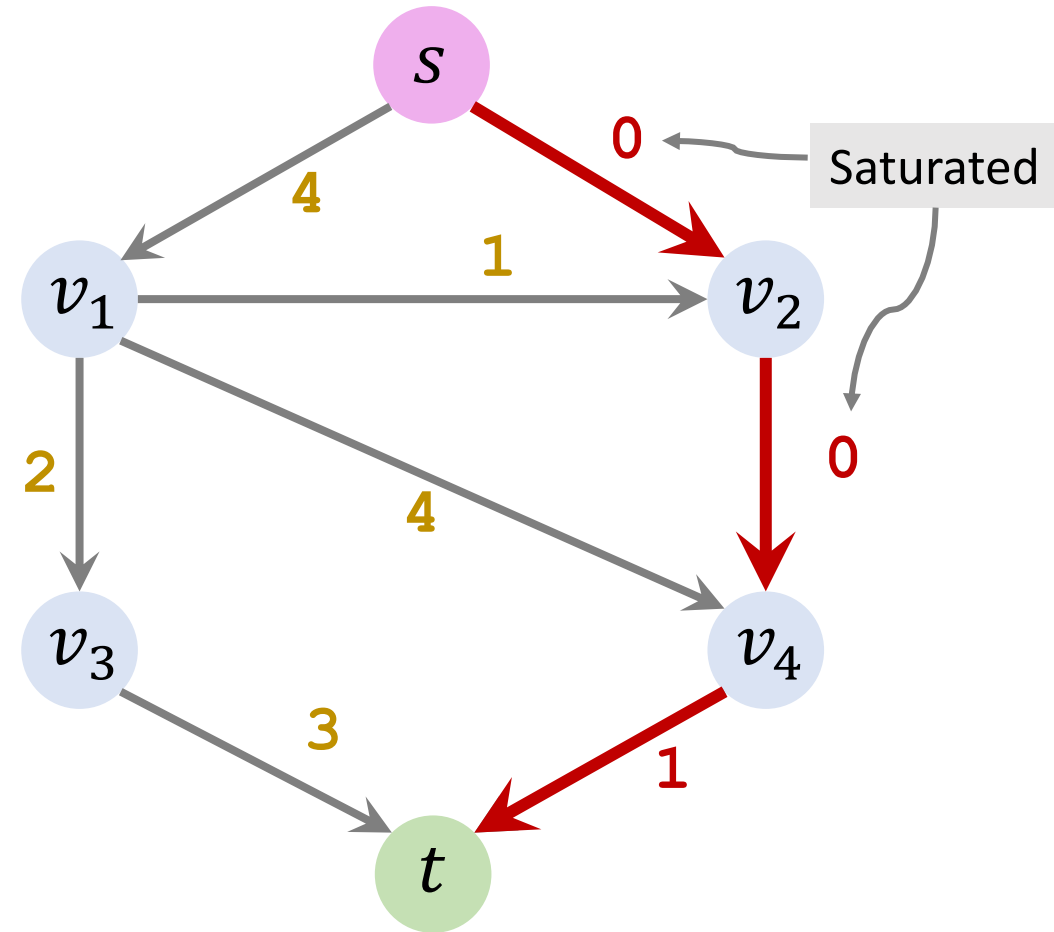
Found path  $s \rightarrow v_2 \rightarrow v_4 \rightarrow t$ . (Bottleneck capacity = 2.)

# Iteration 1: Update residuals

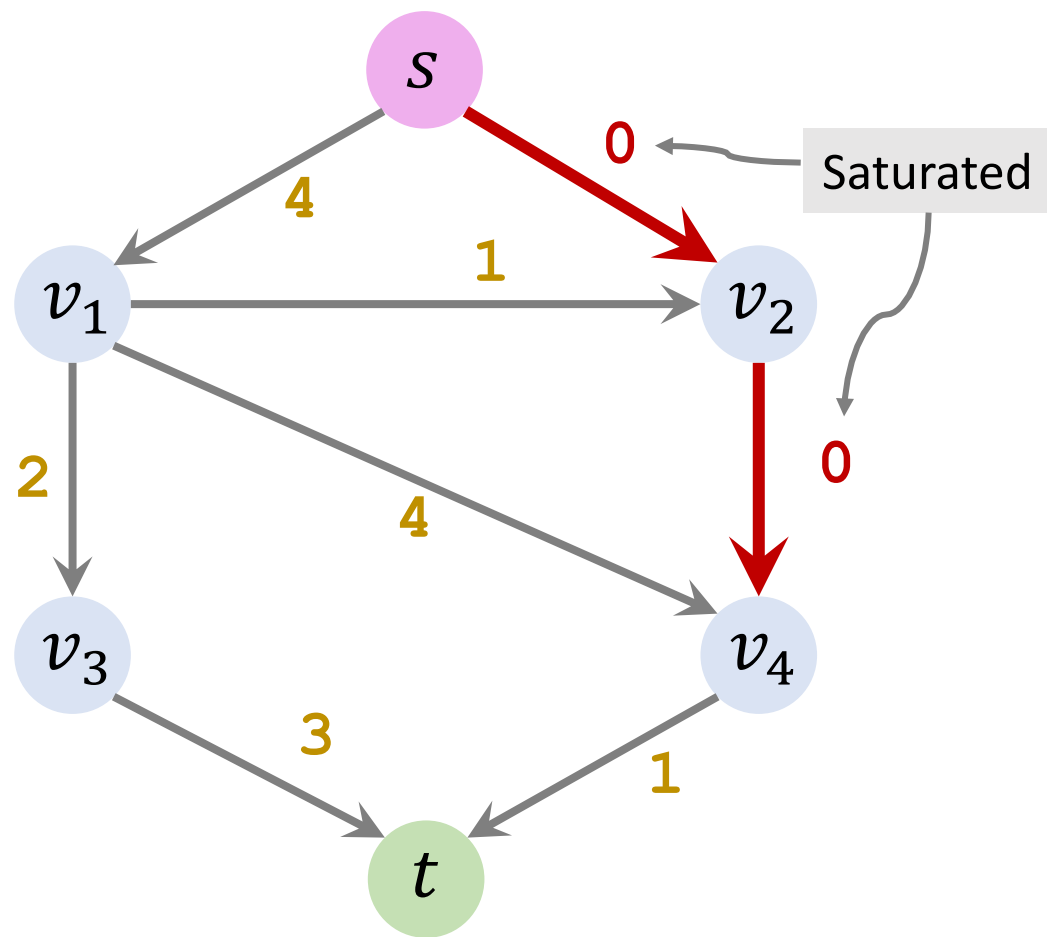
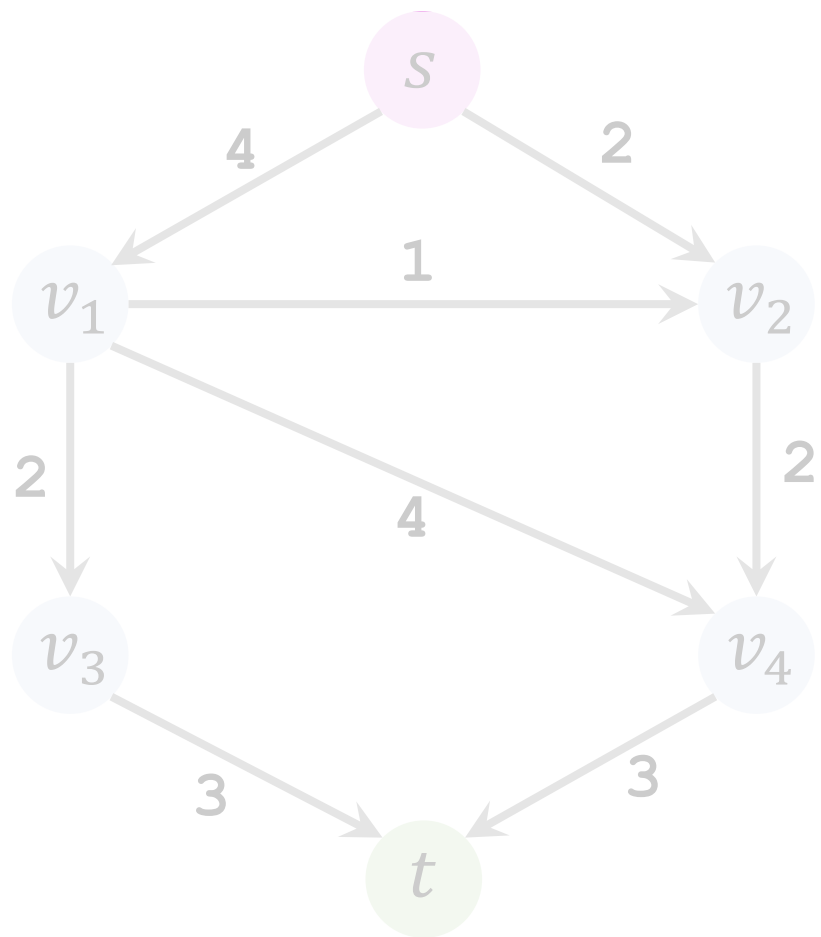


Found path  $s \rightarrow v_2 \rightarrow v_4 \rightarrow t$ . (Bottleneck capacity = 2.)

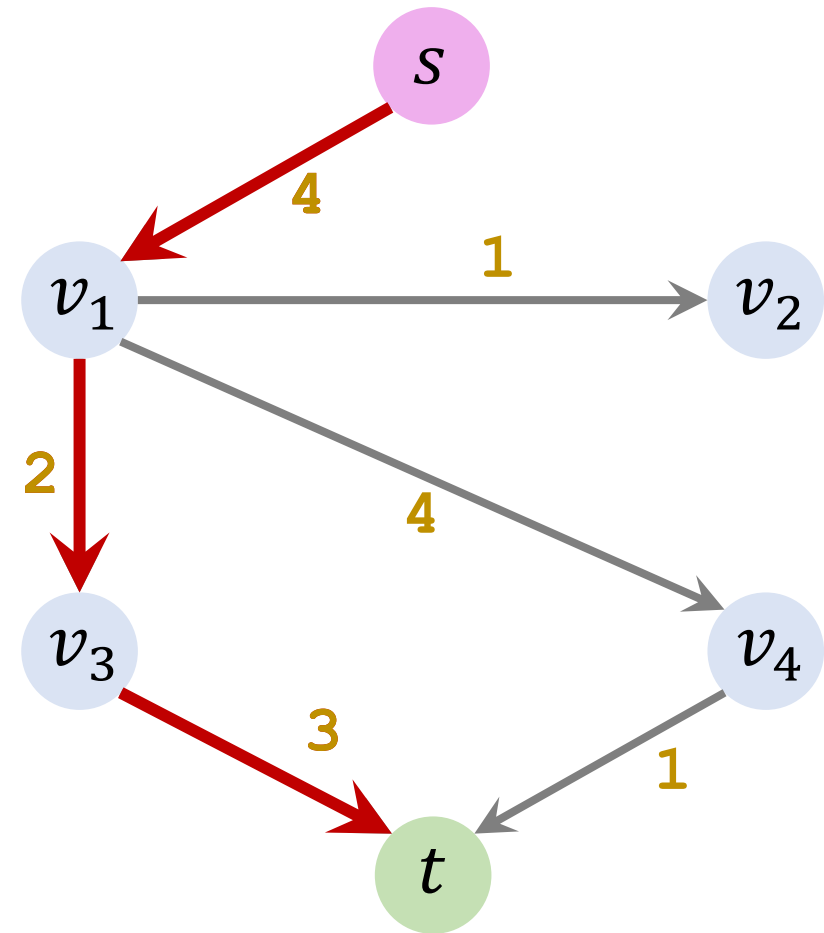
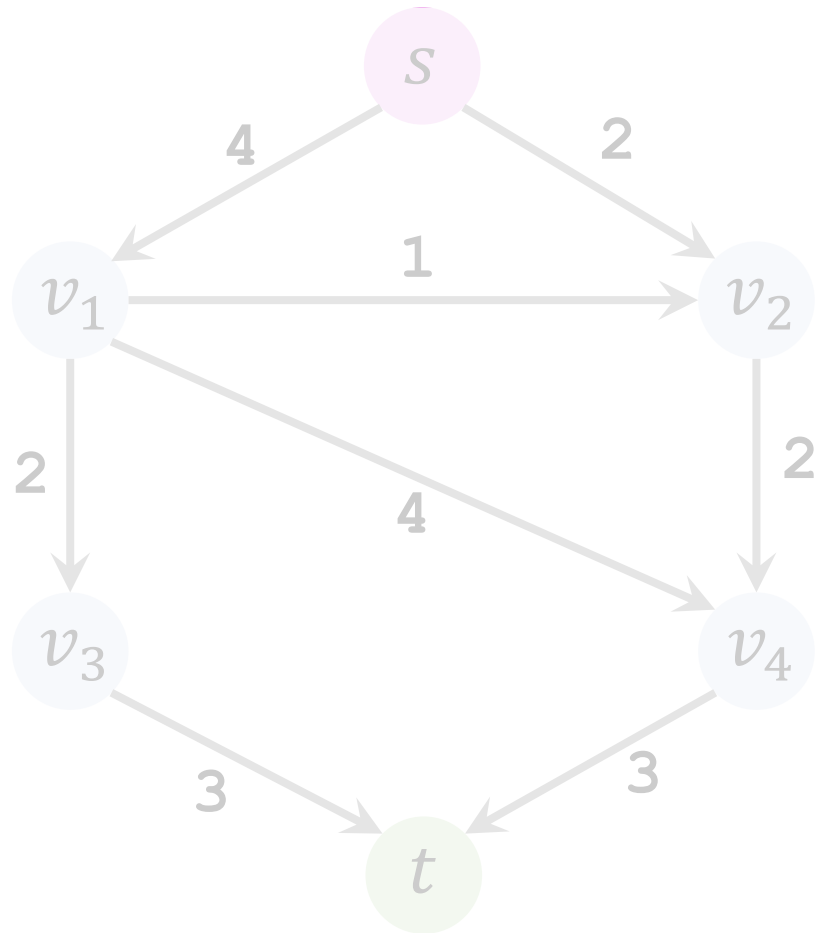
# Iteration 1: Update residuals



# Iteration 1: Remove saturated edges



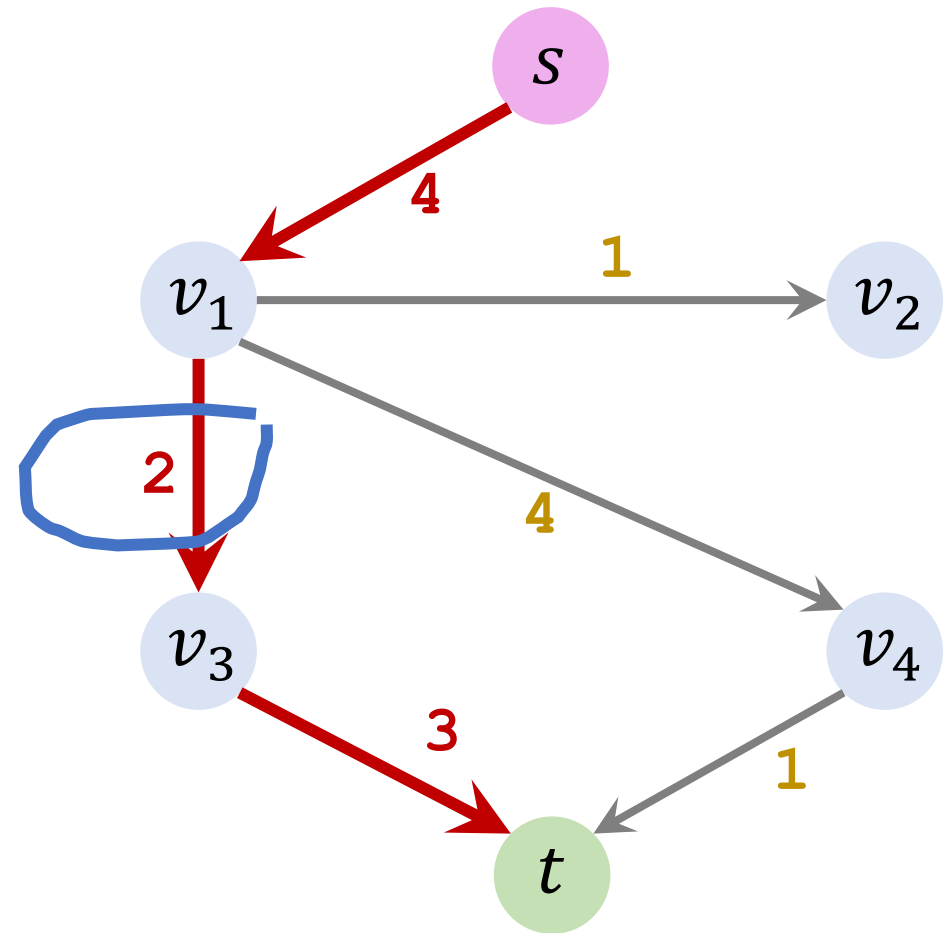
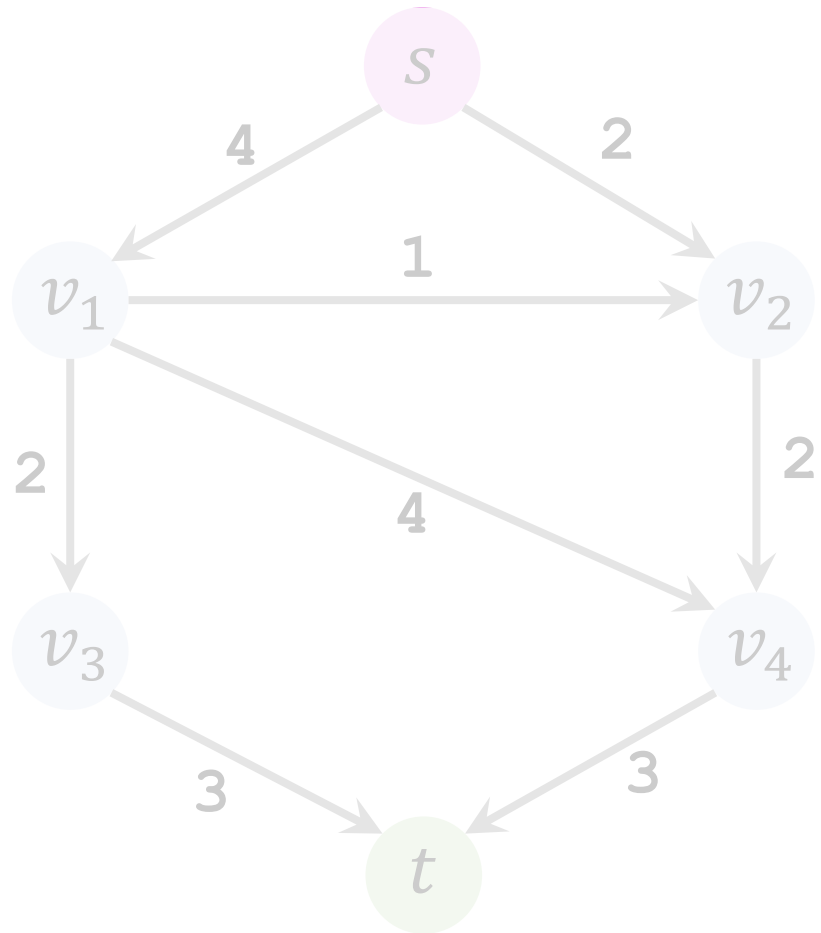
# Iteration 2: Find an augmenting path



Found path  $s \rightarrow v_1 \rightarrow v_3 \rightarrow t$ .

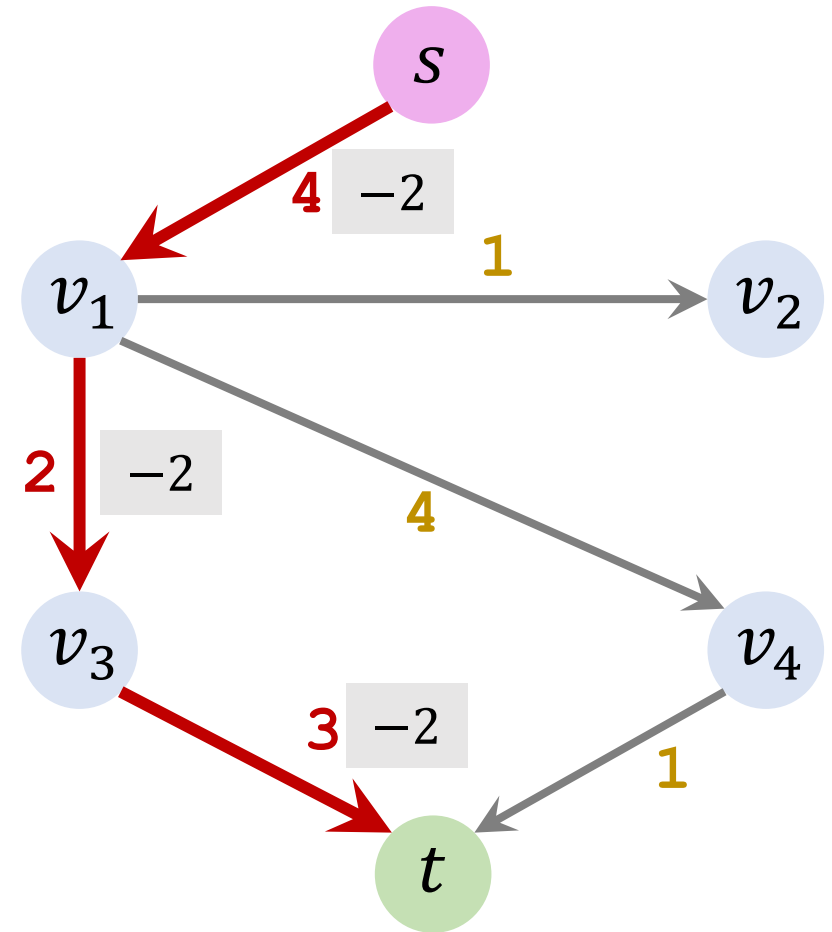
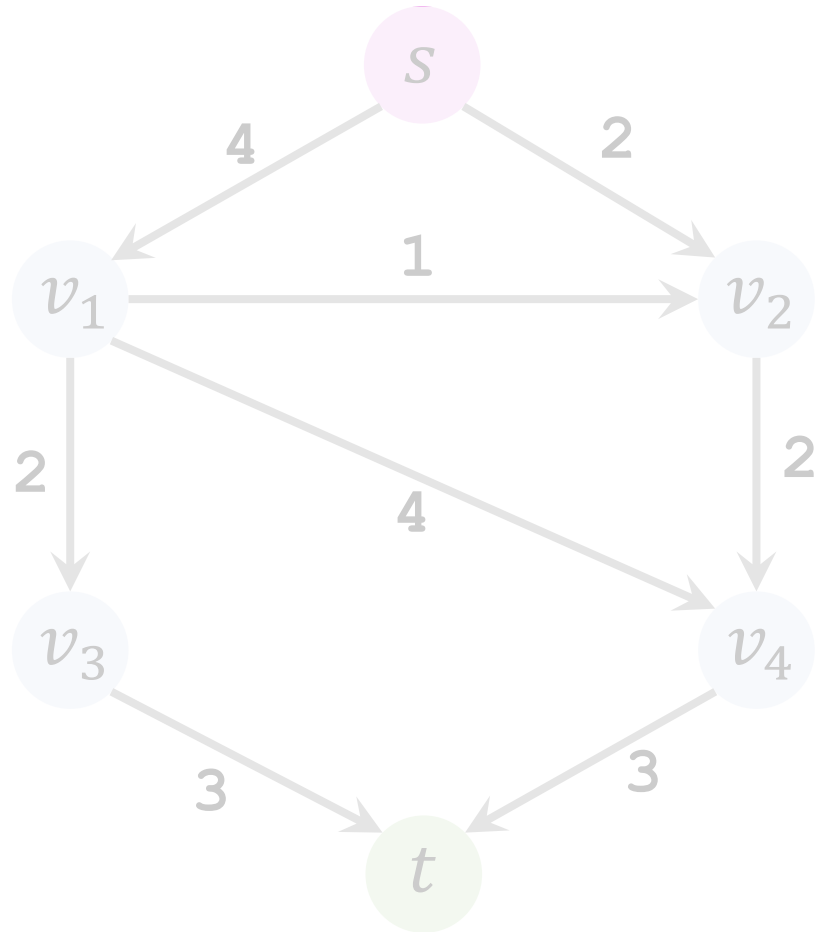


# Iteration 2: Find an augmenting path

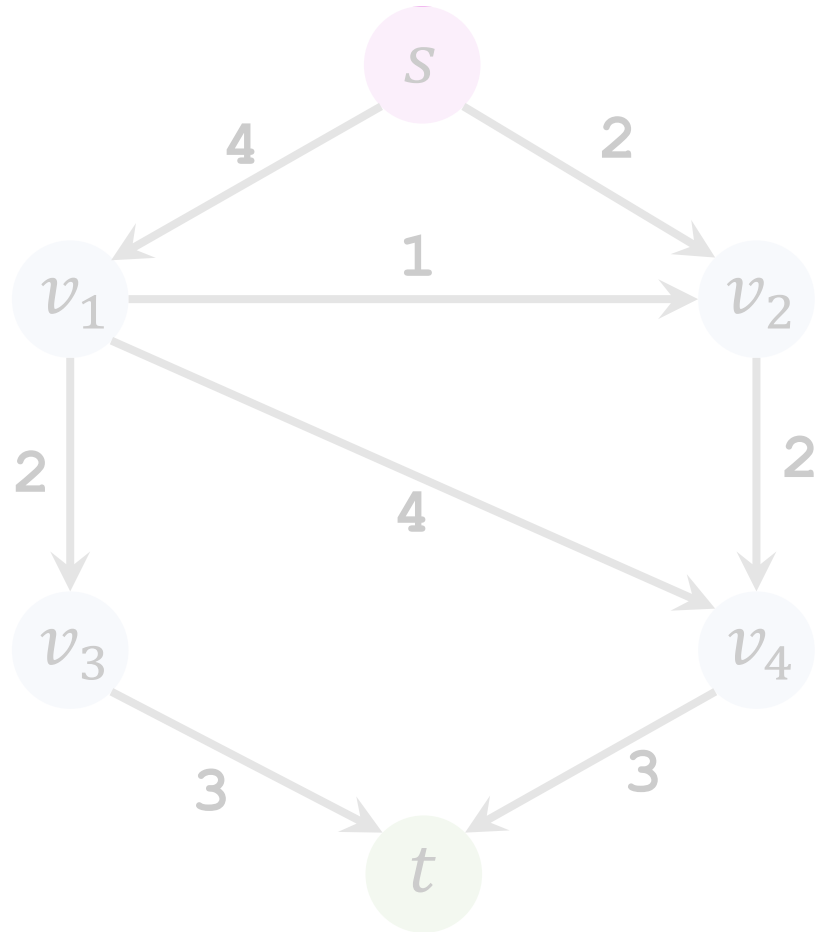


Found path  $s \rightarrow v_1 \rightarrow v_3 \rightarrow t$ . (Bottleneck capacity = 2.)

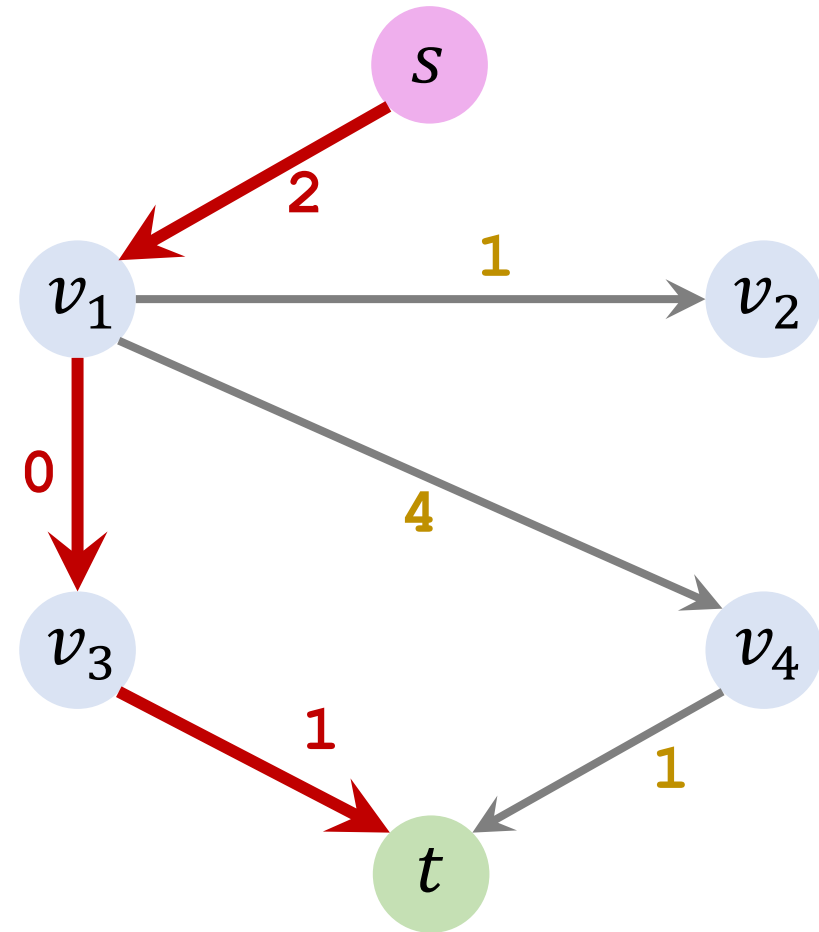
# Iteration 2: Update residuals



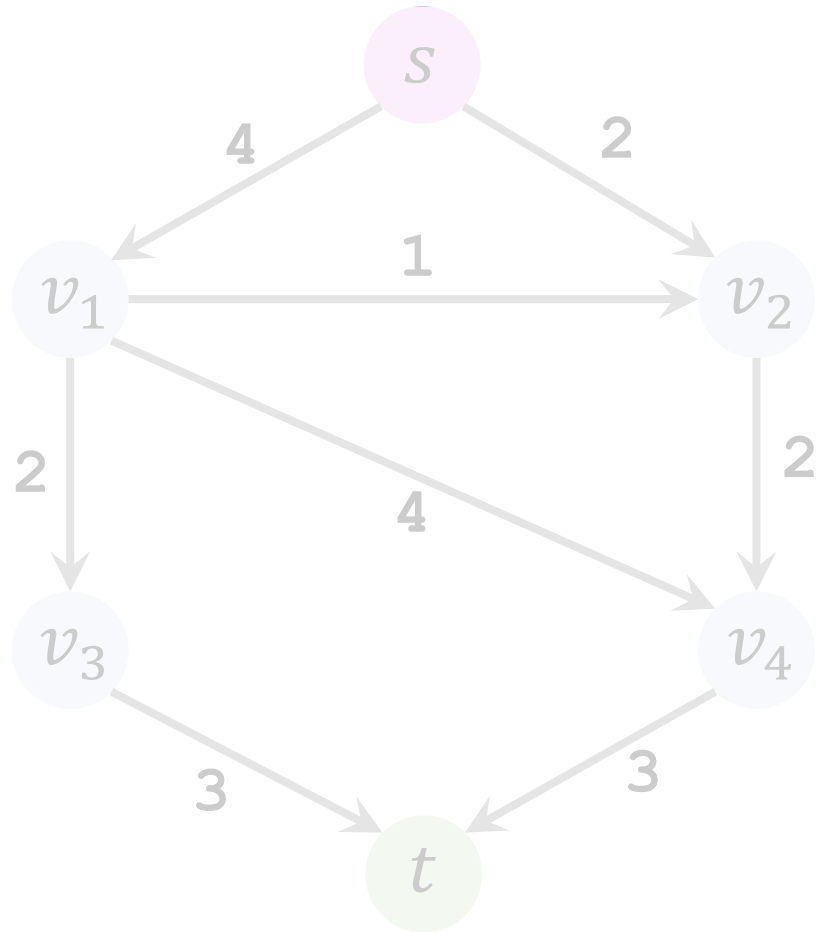
# Iteration 2: Update residuals



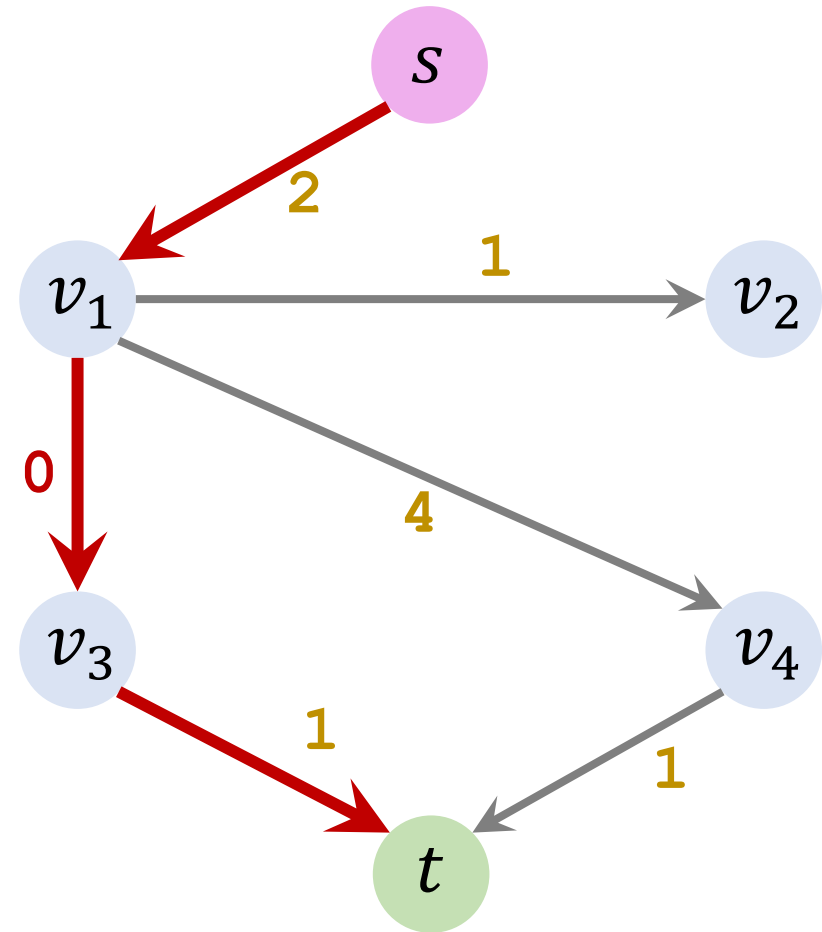
Saturated



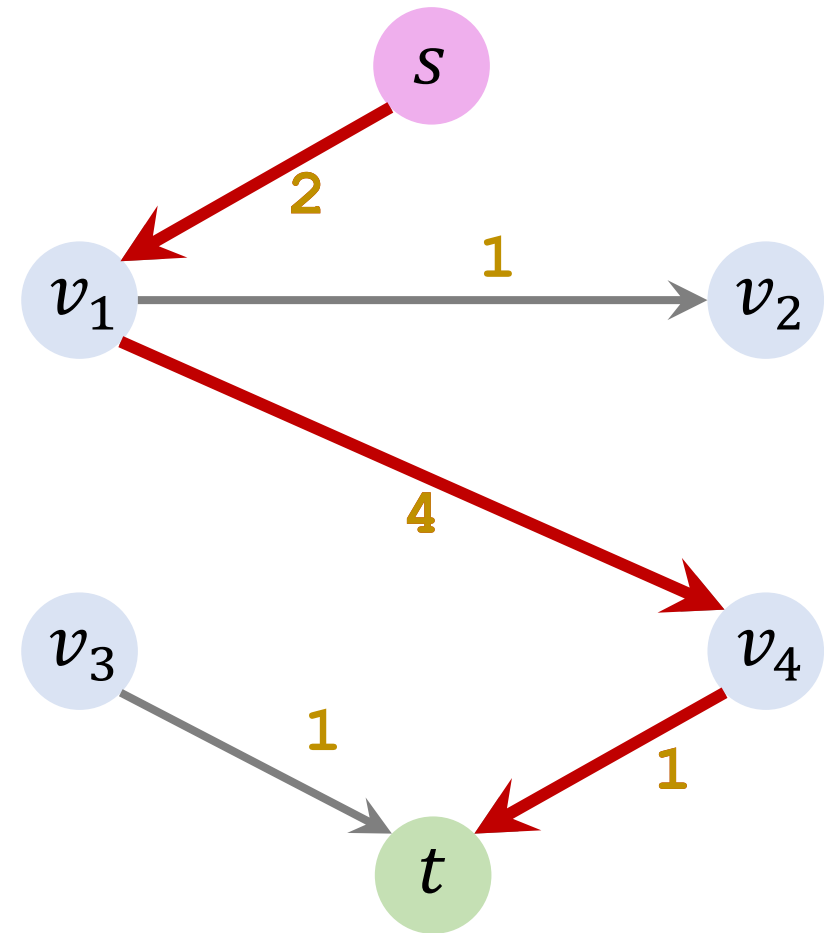
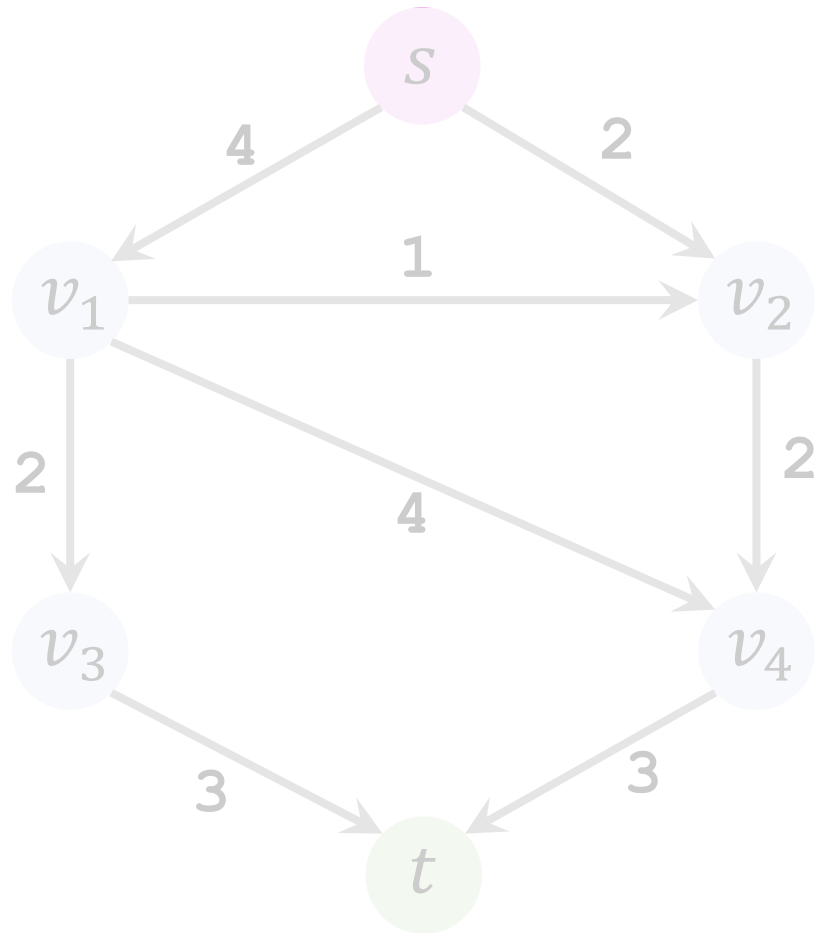
# Iteration 2: Remove saturated edges



Saturated

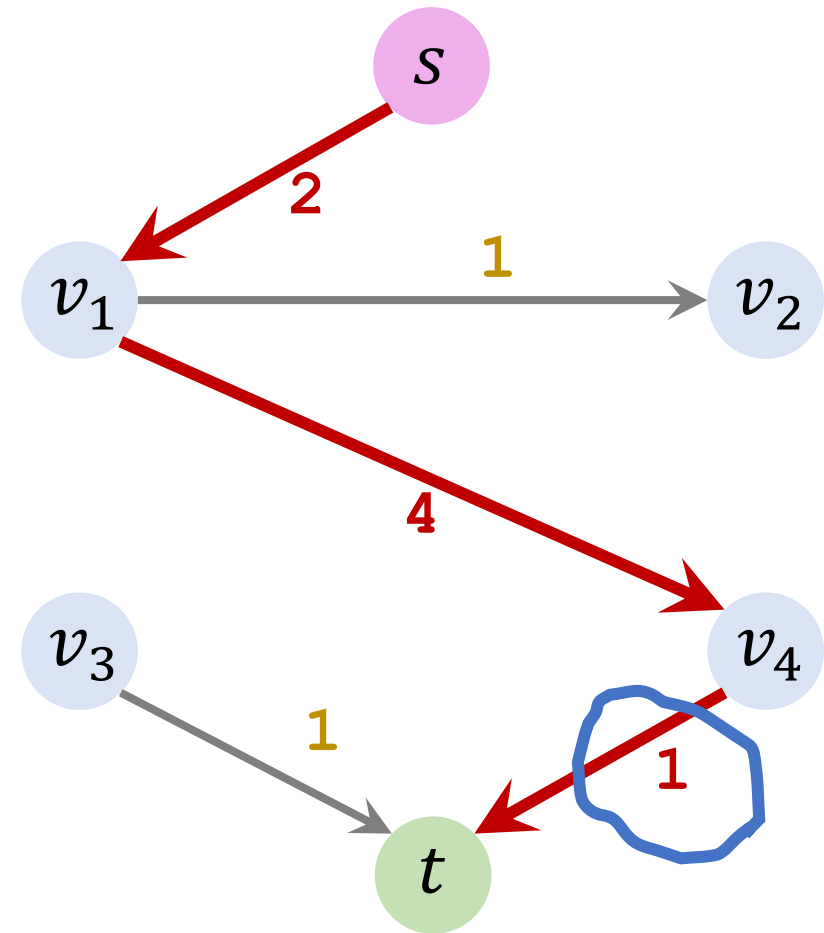
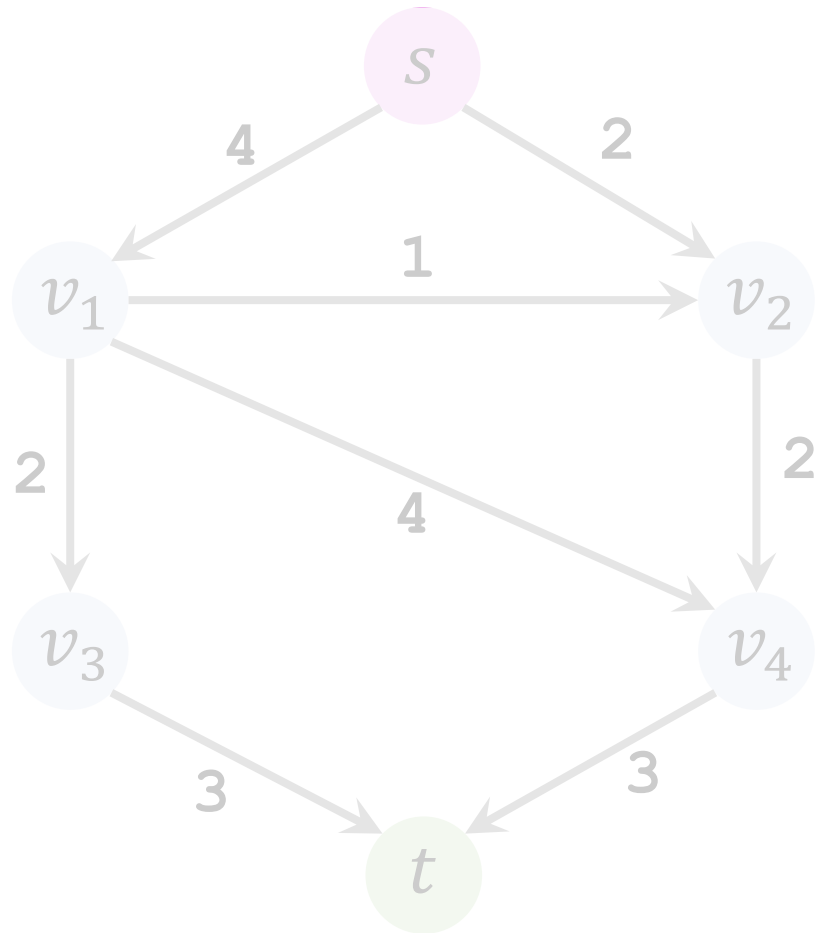


# Iteration 3: Find an augmenting path



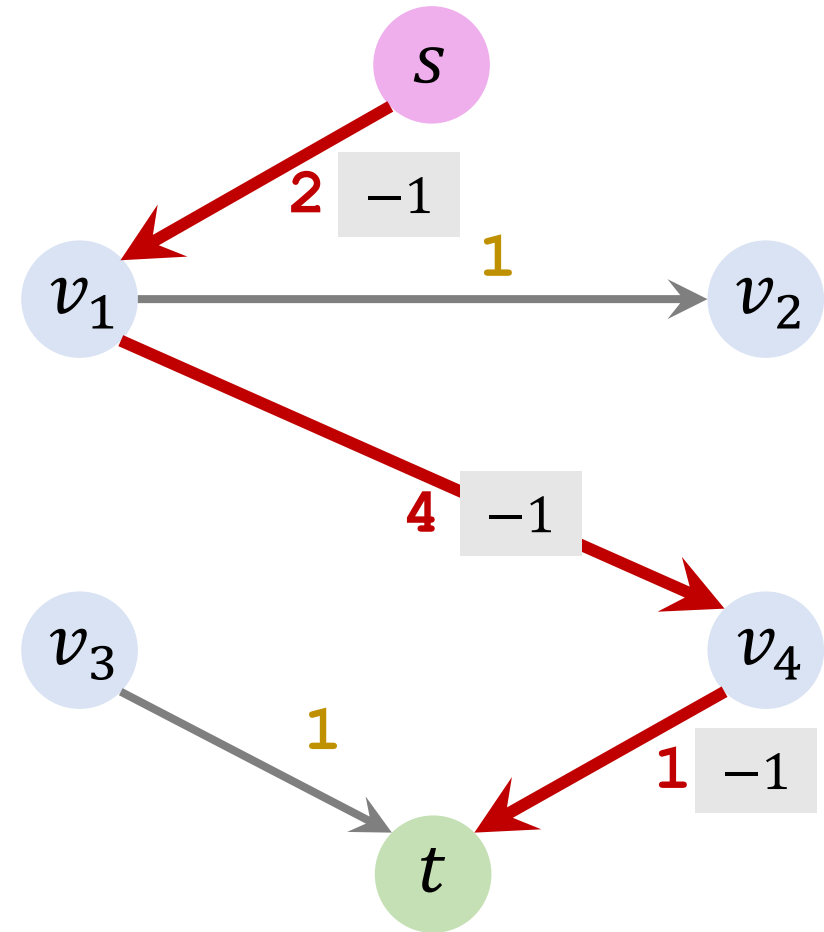
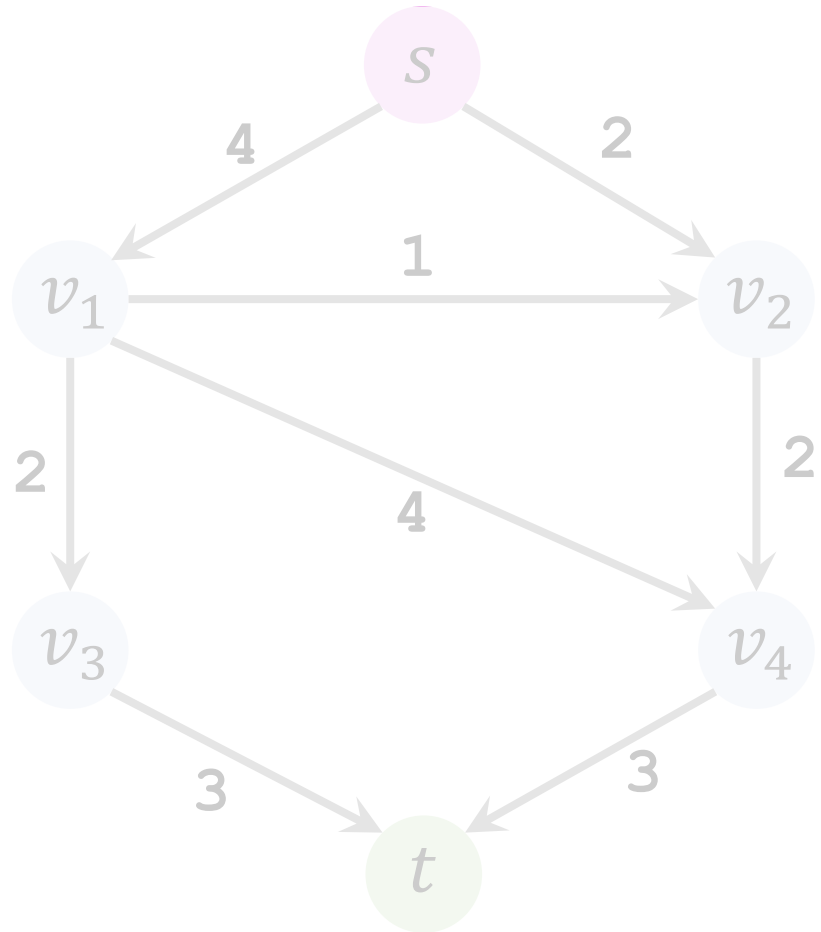
Found path  $s \rightarrow v_1 \rightarrow v_4 \rightarrow t$ .

# Iteration 3: Find an augmenting path

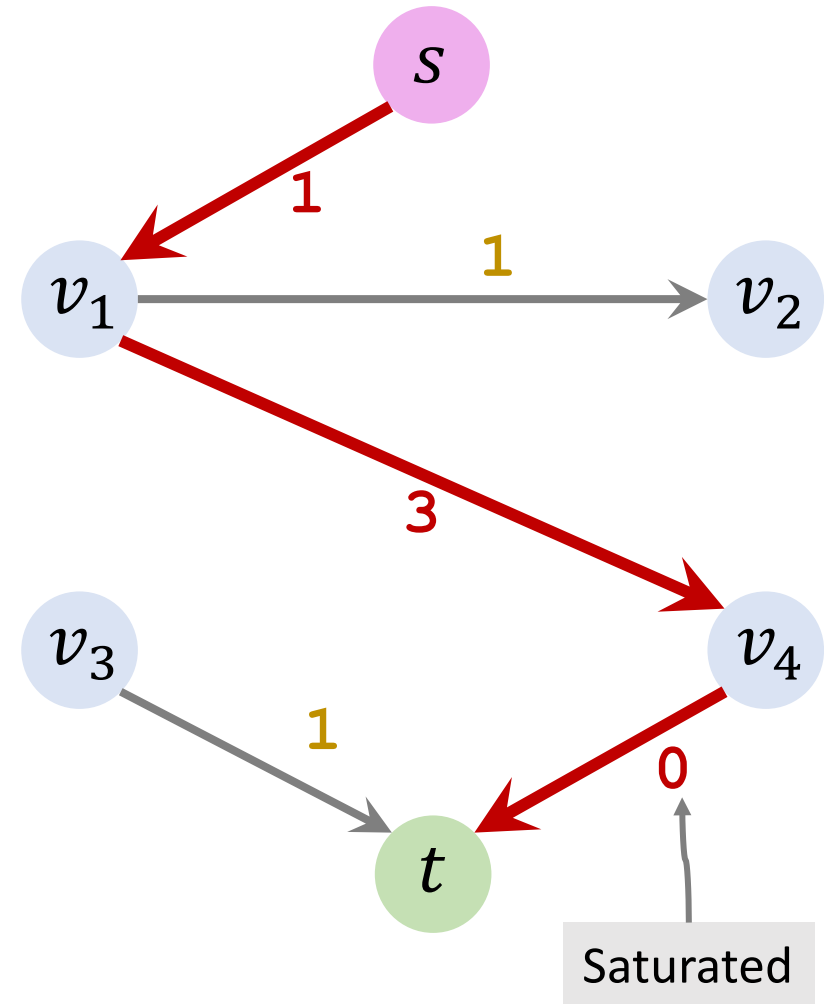
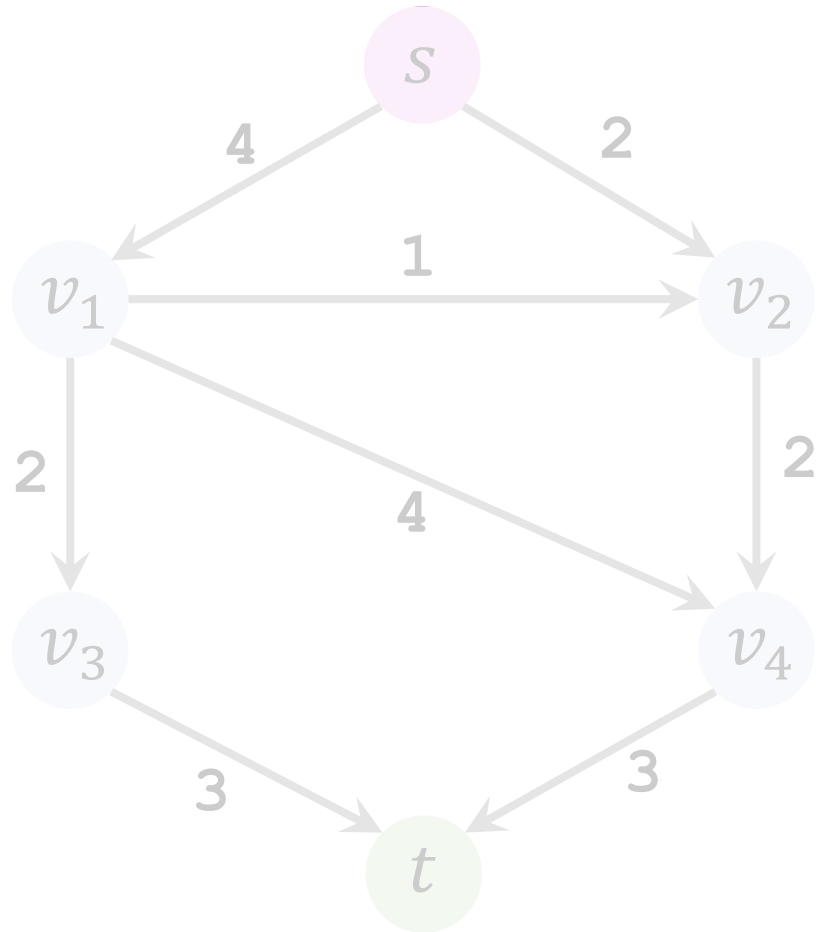


Found path  $s \rightarrow v_1 \rightarrow v_4 \rightarrow t$ . (Bottleneck capacity = 1.)

# Iteration 3: Update residuals

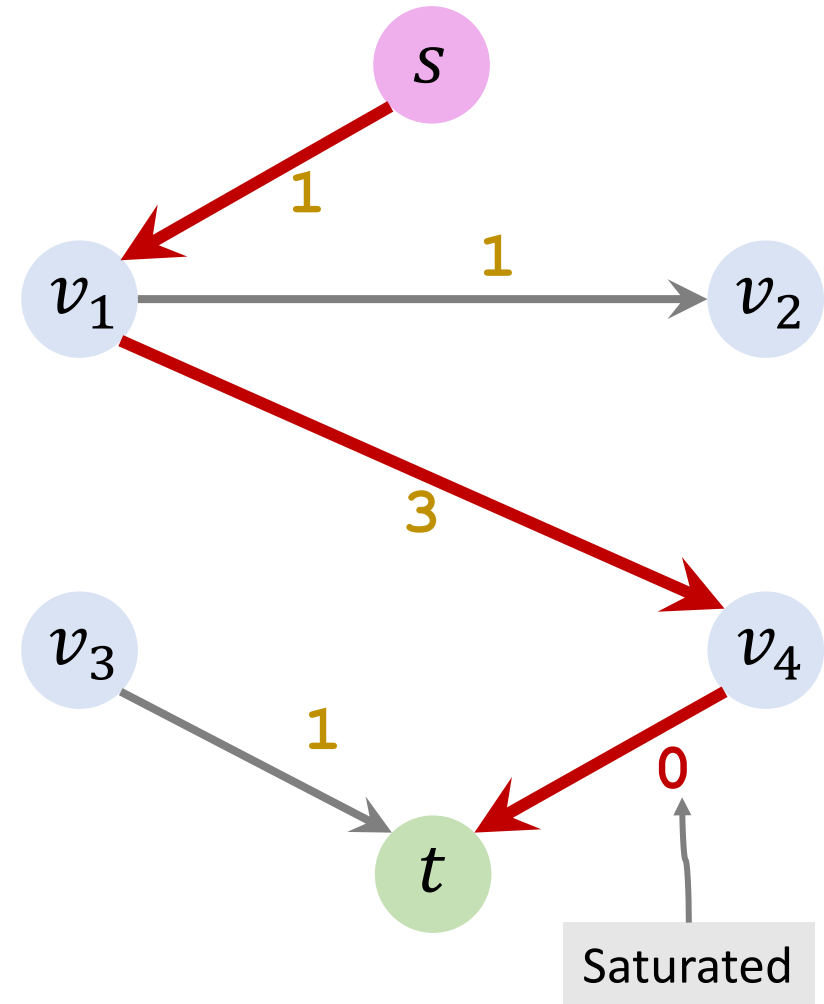
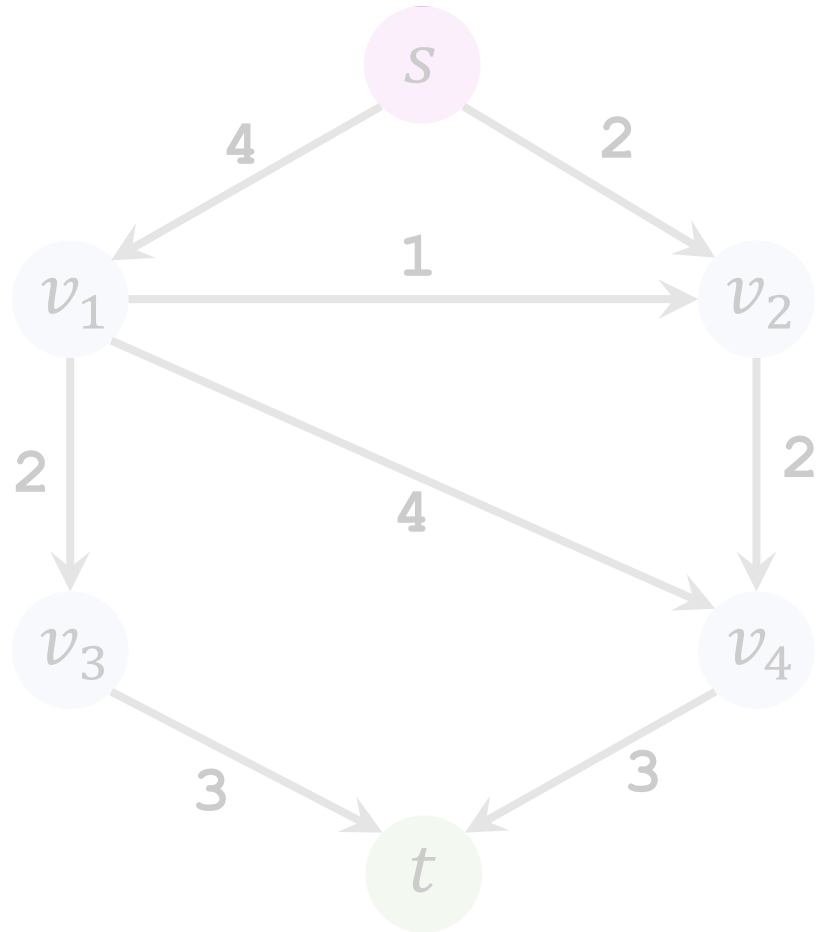


# Iteration 3: Update residuals

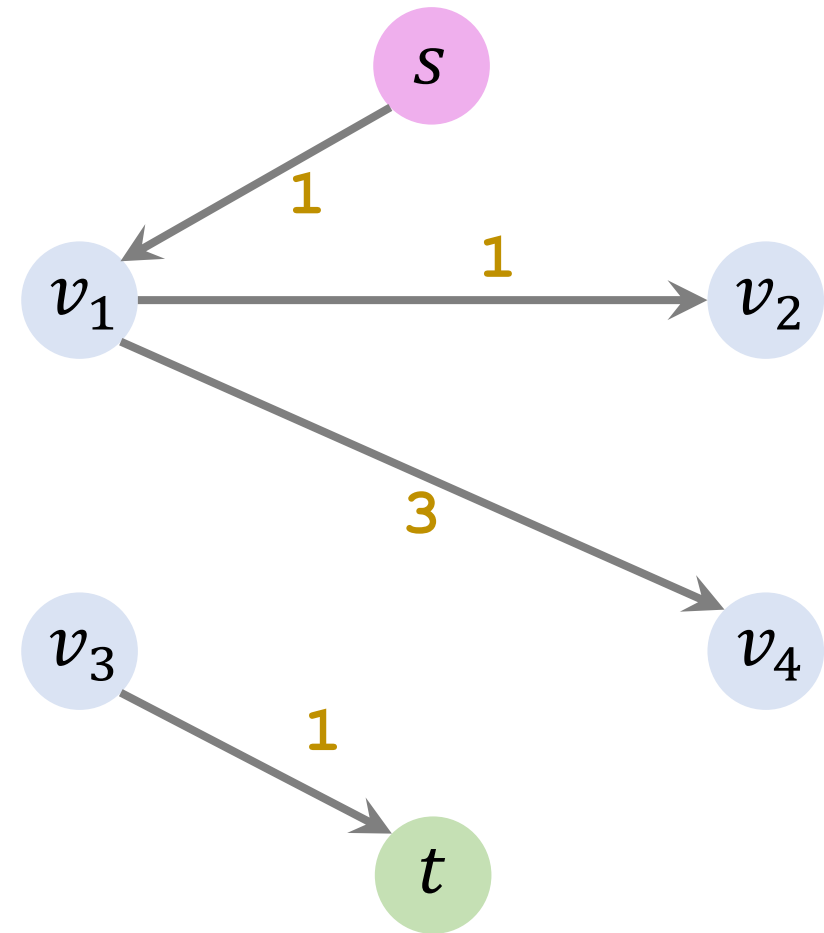
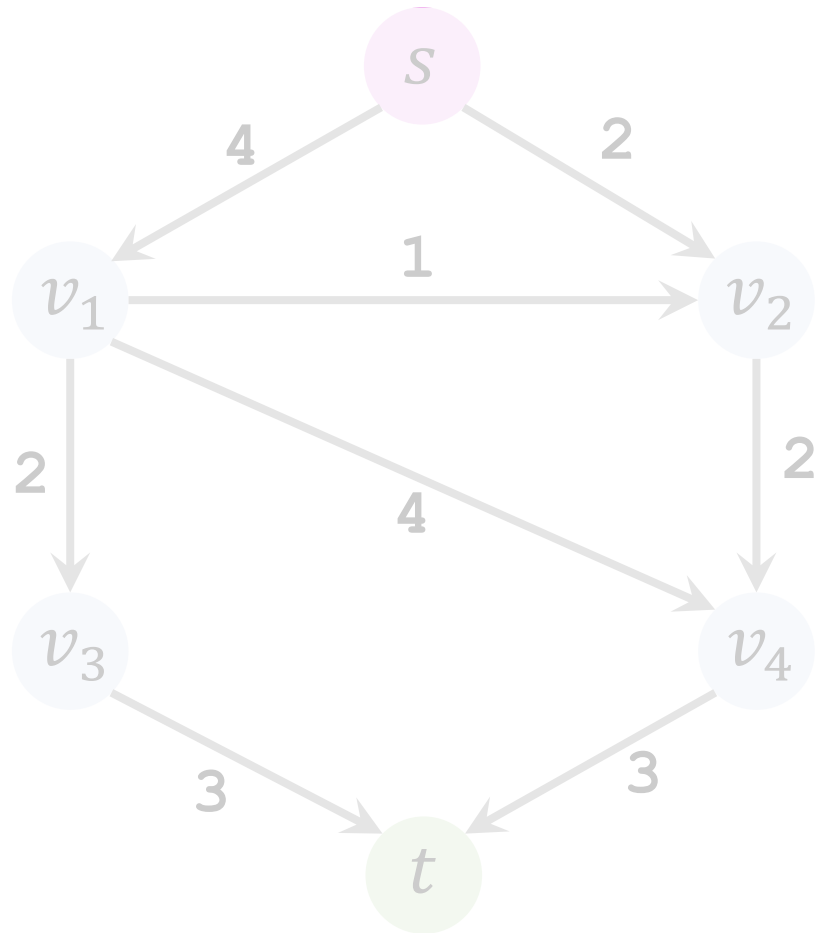




# Iteration 3: Remove saturated edges

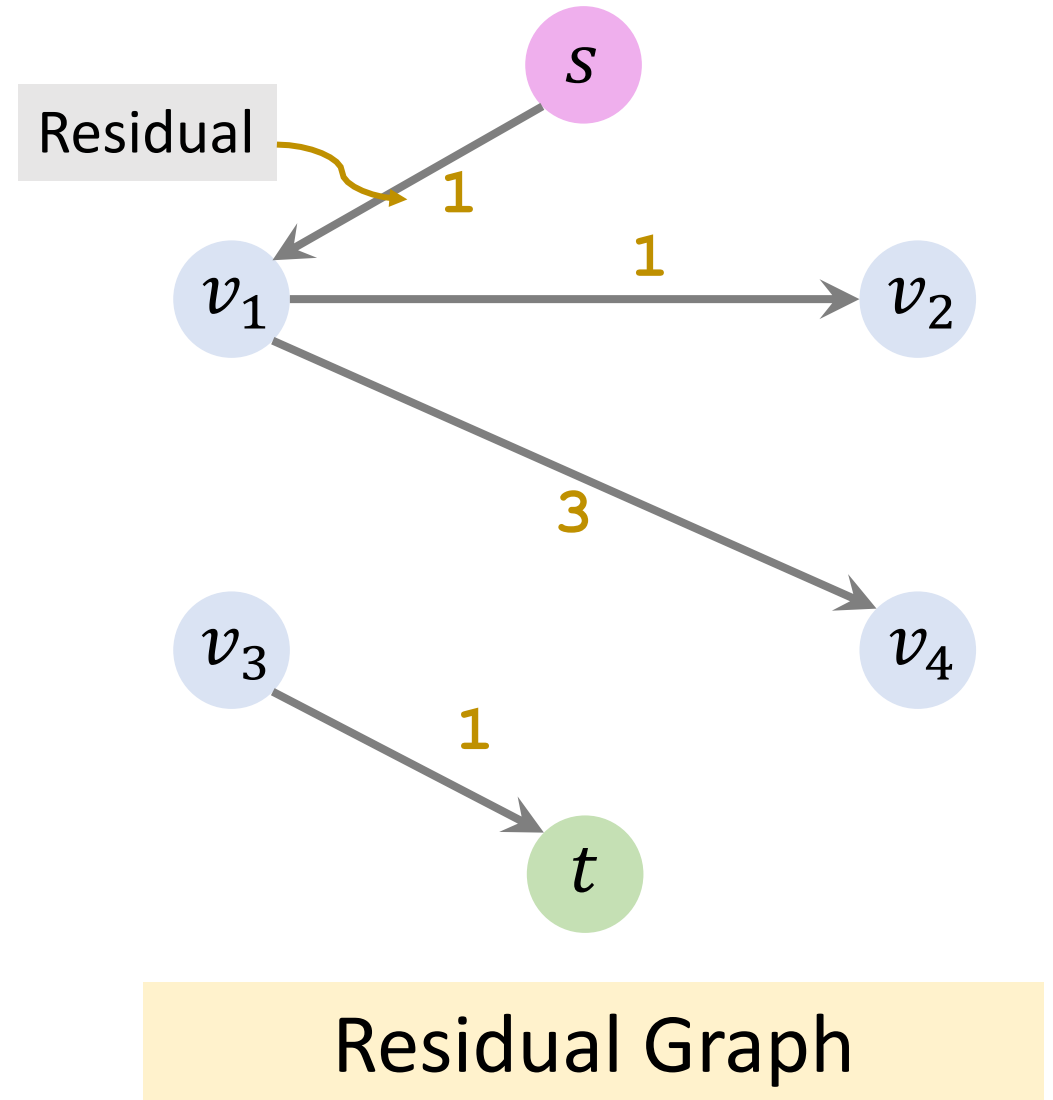
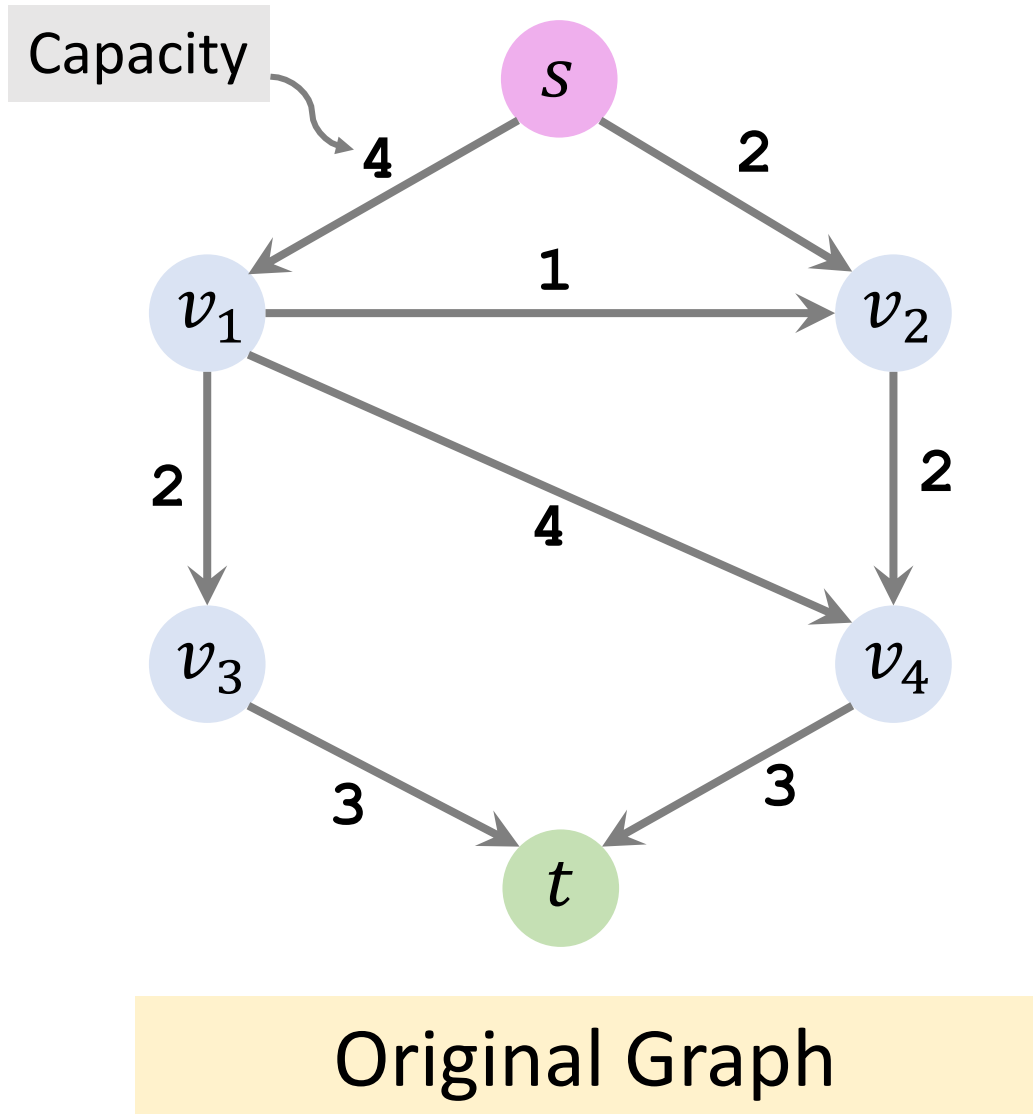


# Iteration 4: Find an augmenting path

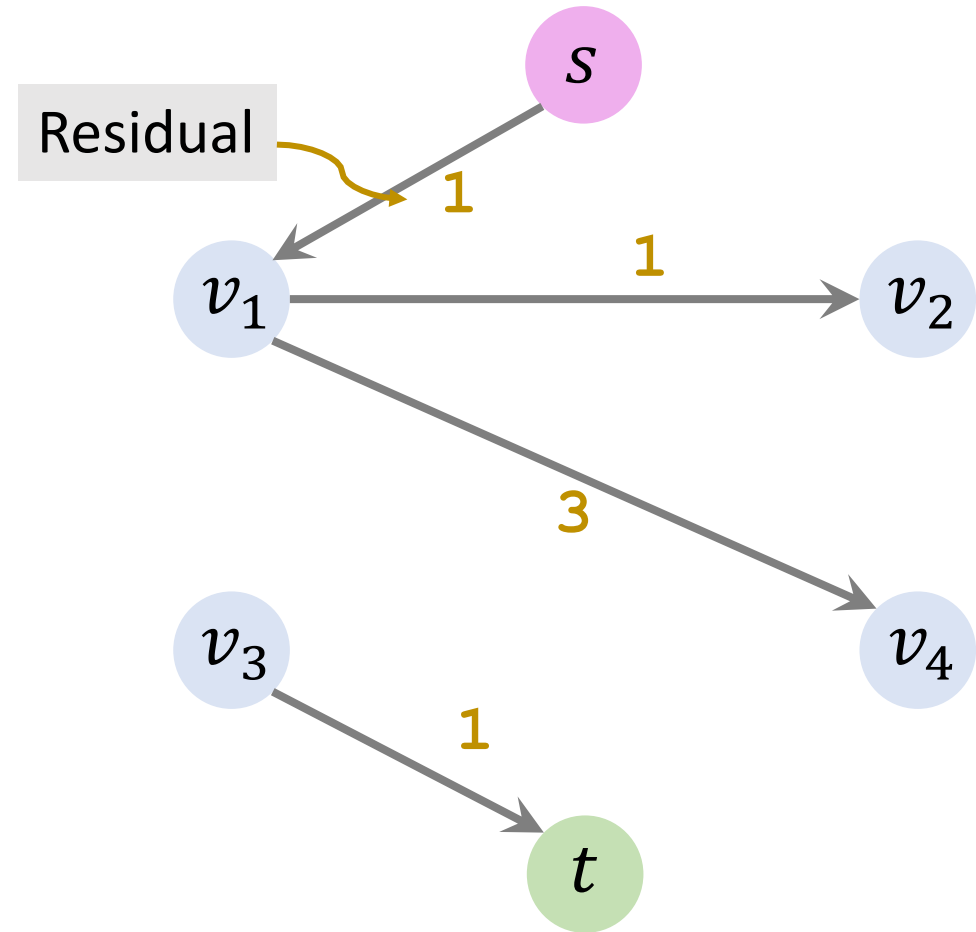
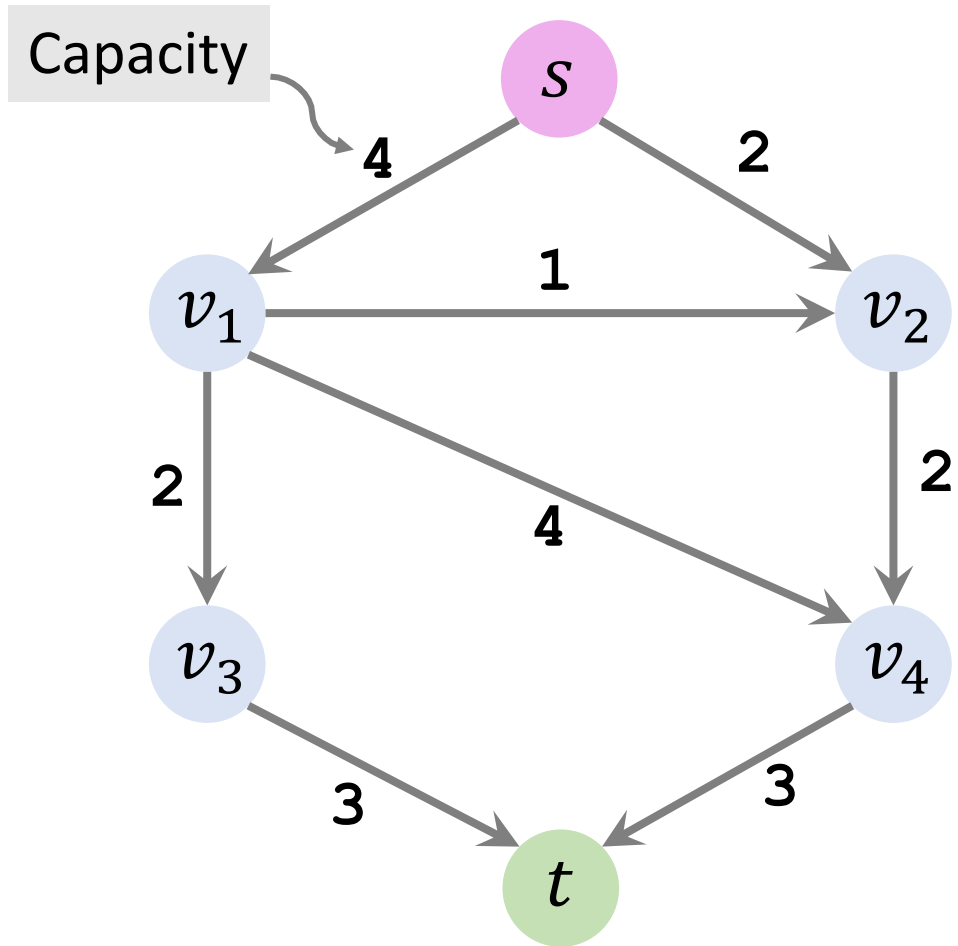


Cannot find any path from source to sink.

# End of Procedure

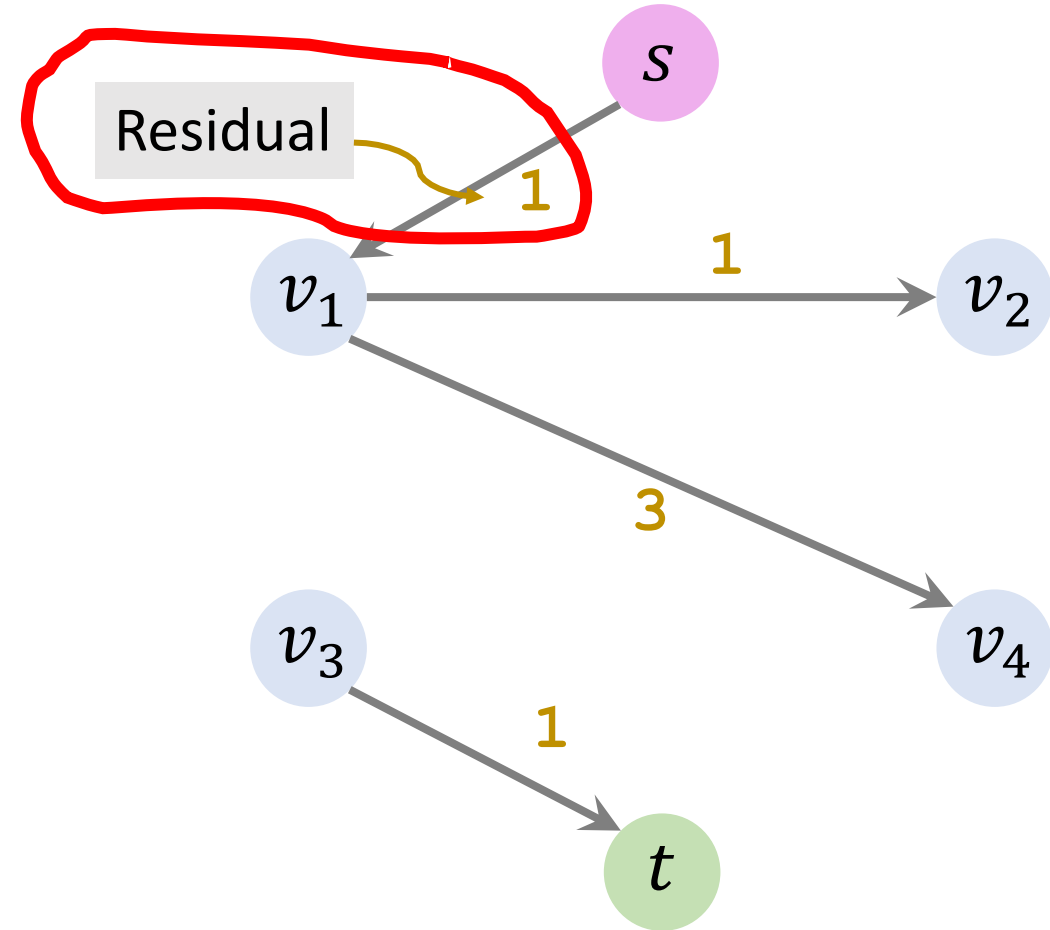
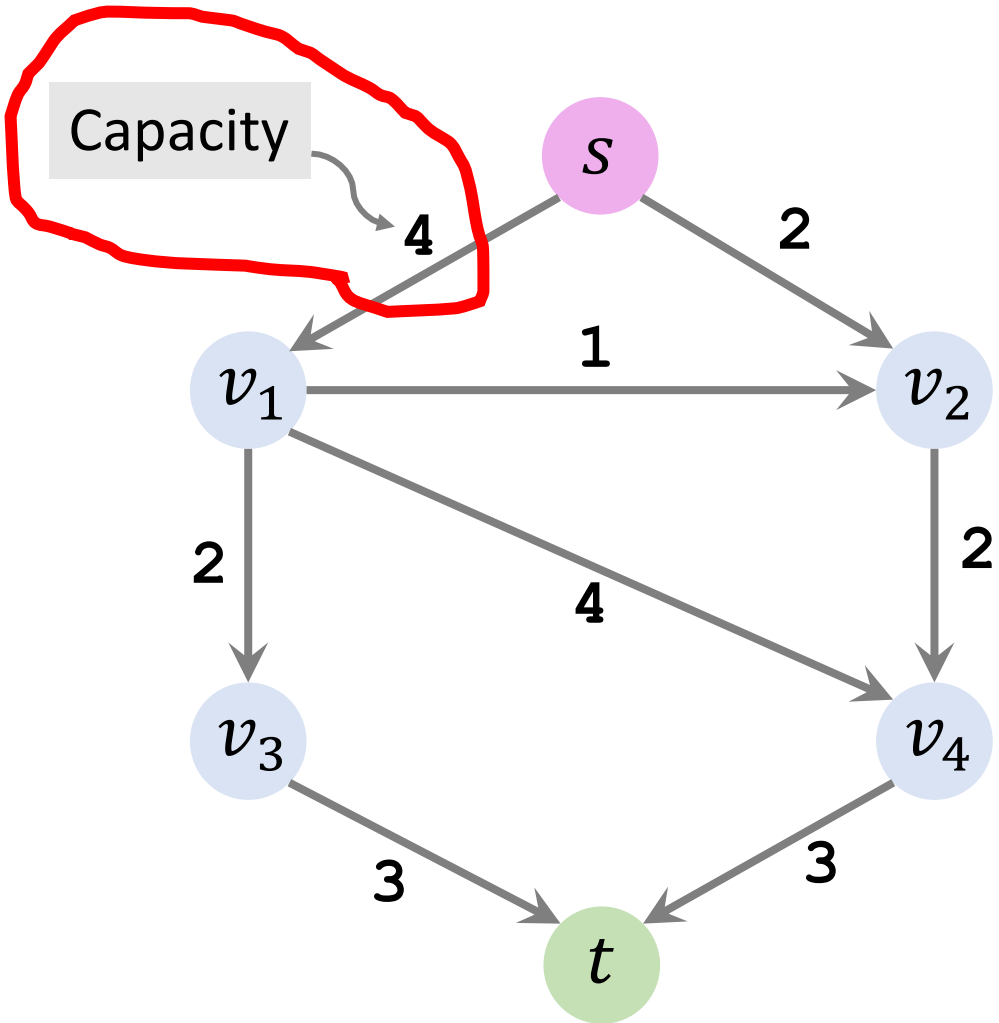


# End of Procedure



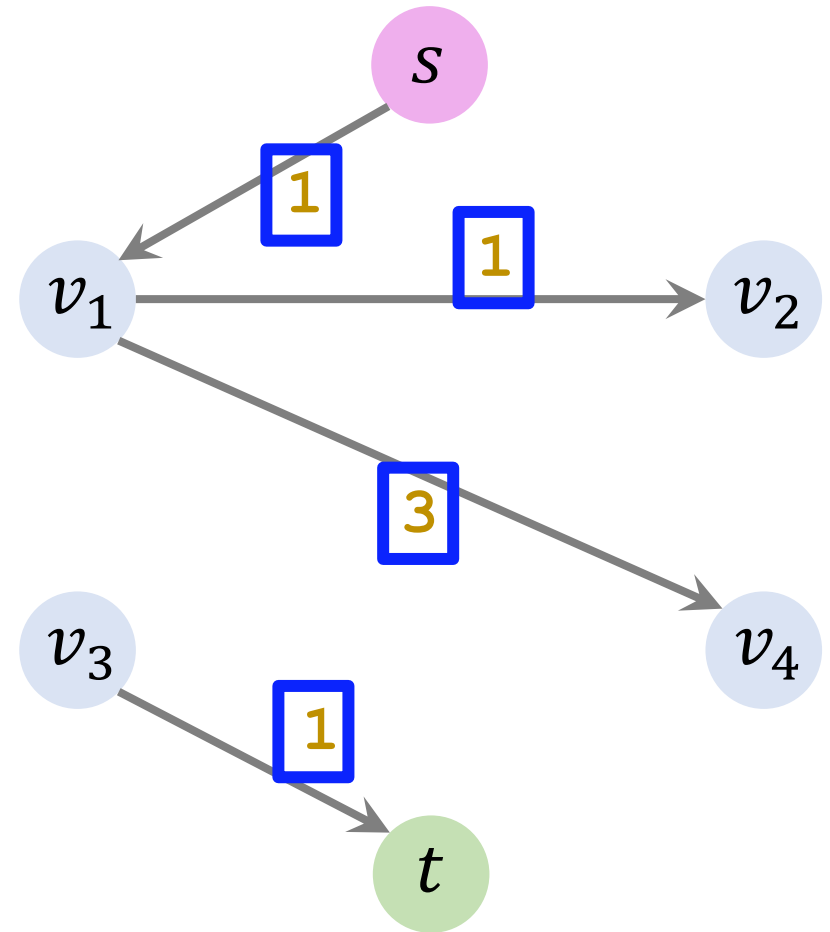
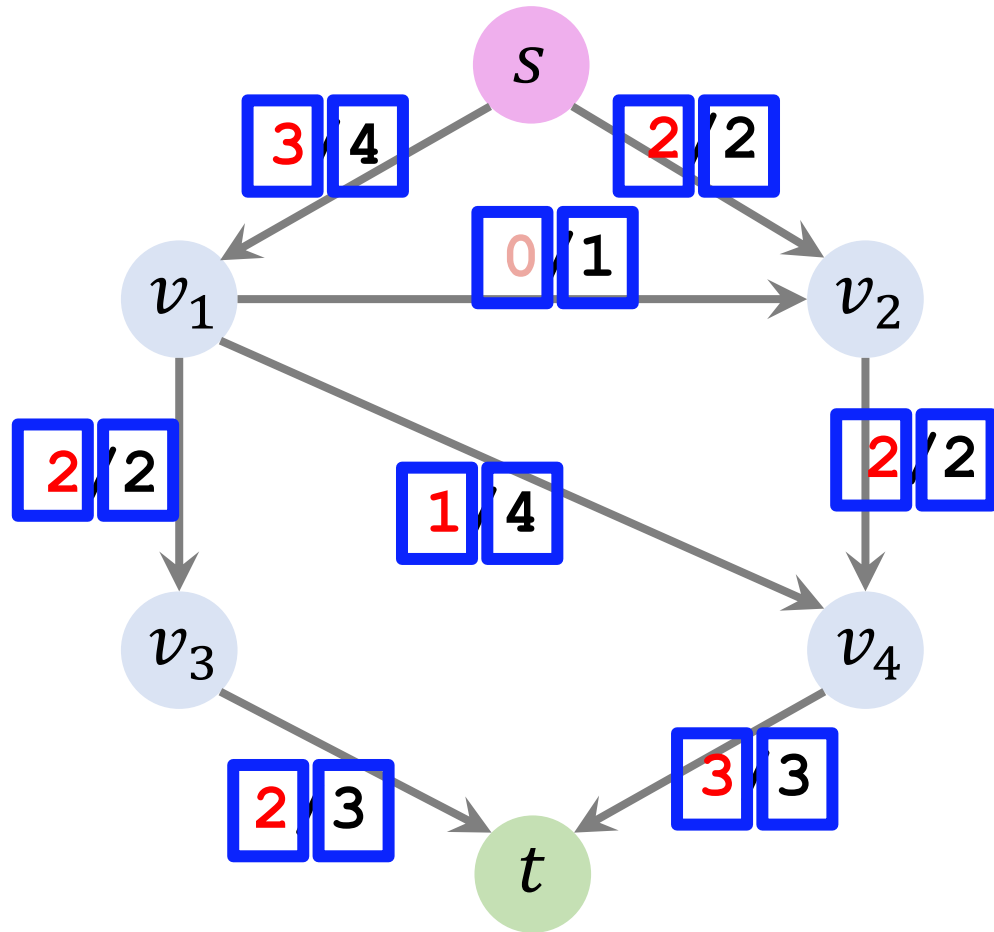
$$\text{Flow} = \text{Capacity} - \text{Residual}.$$

# End of Procedure



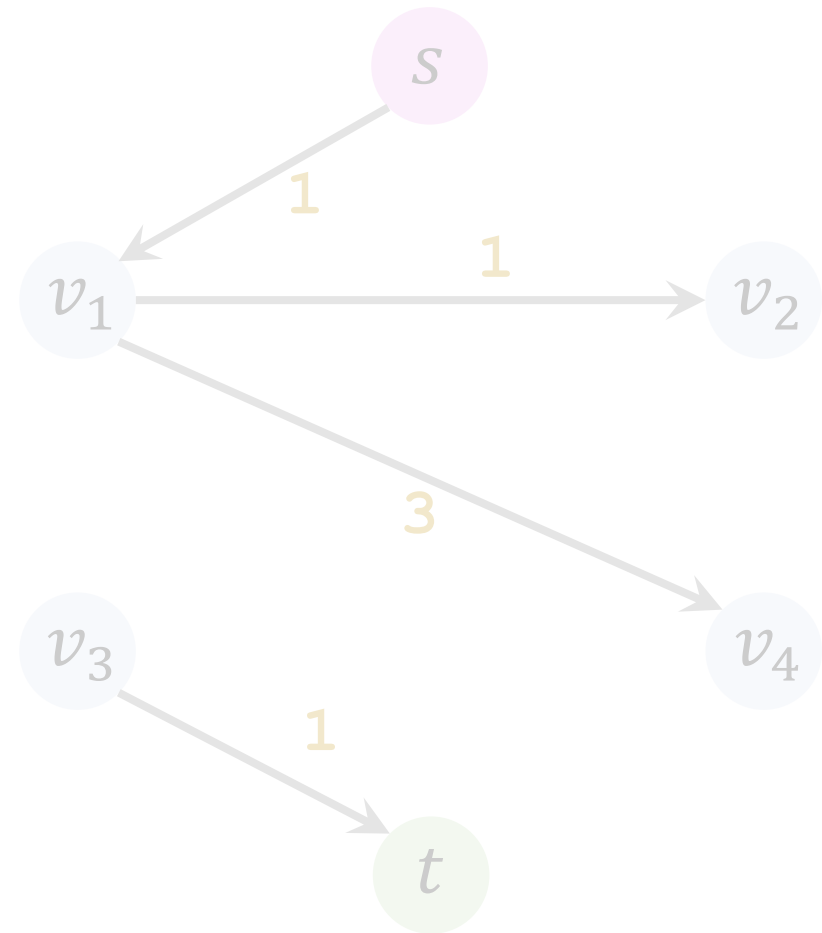
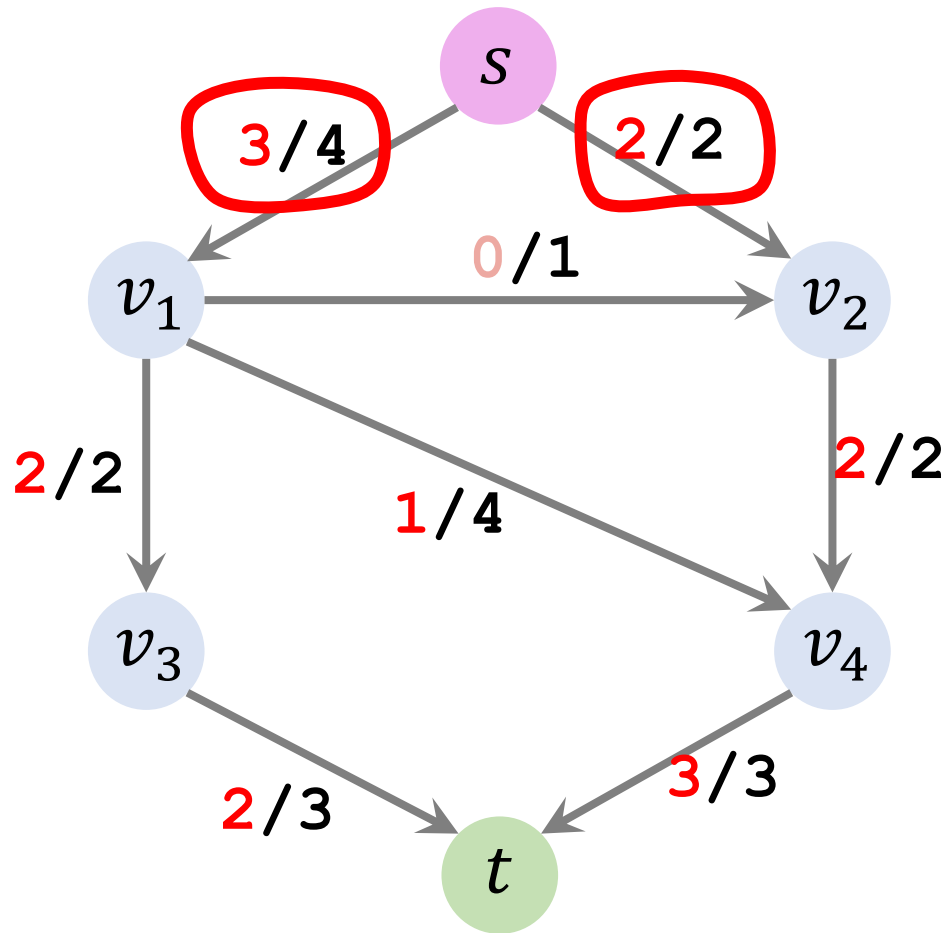
$$\text{Flow} = \text{Capacity} - \text{Residual}.$$

# End of Procedure



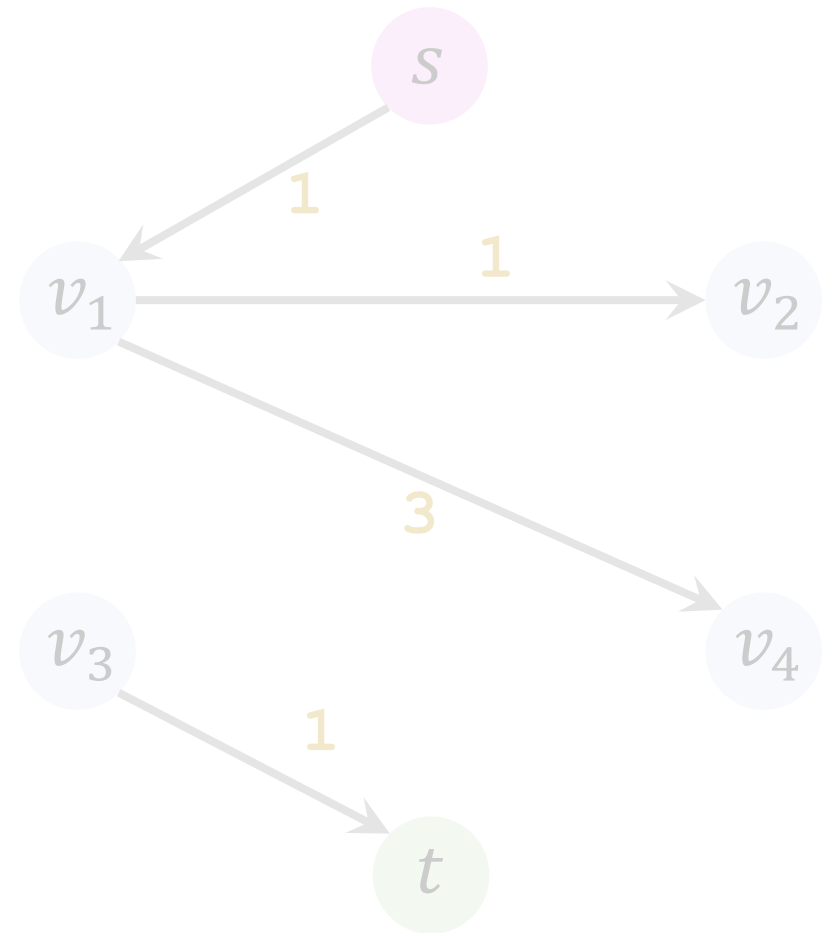
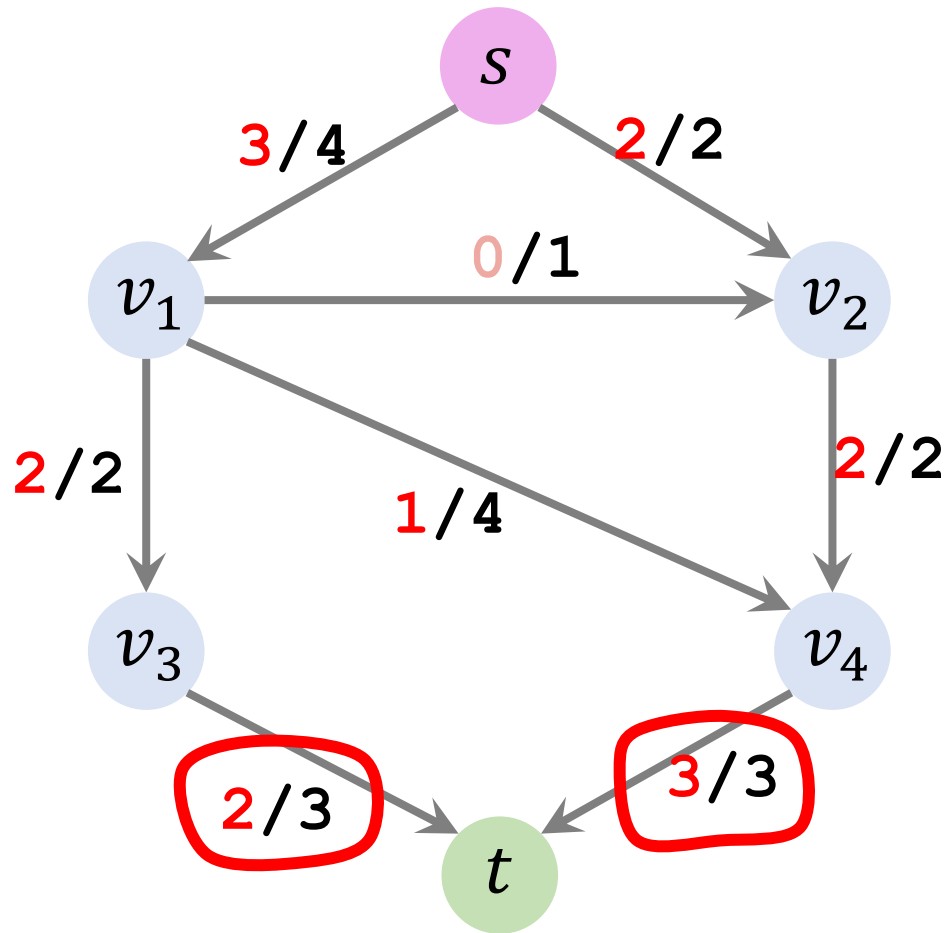
Flow = Capacity - Residual.

# End of Procedure



Amount of Flow = **5**.

# End of Procedure

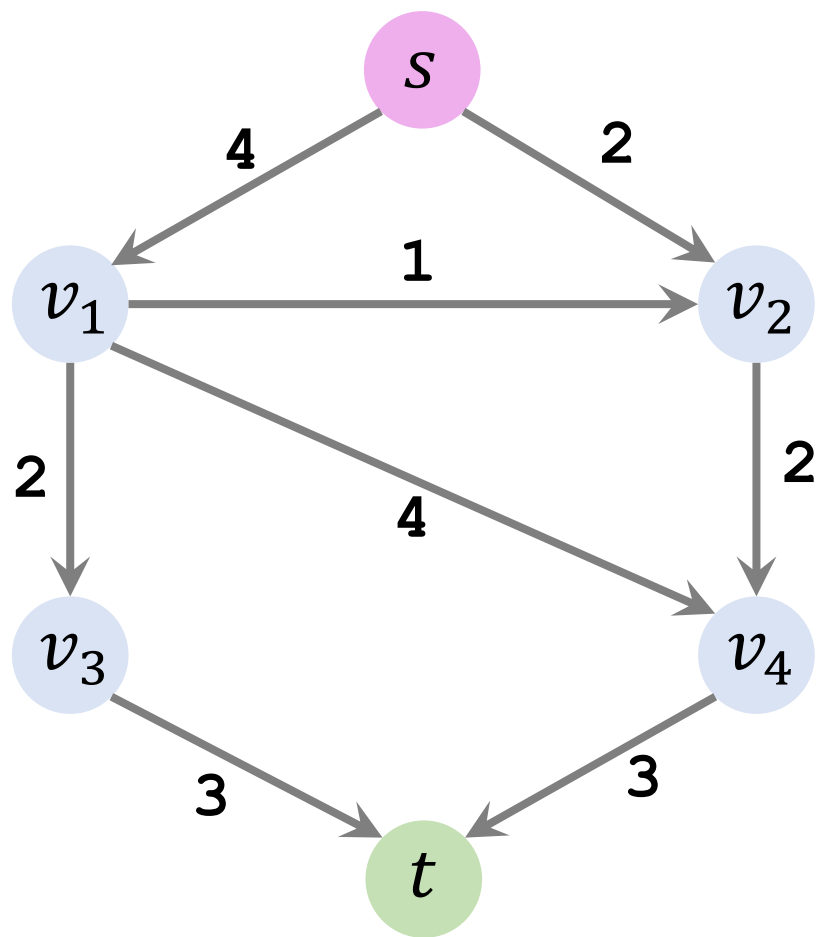


Amount of Flow = 5.

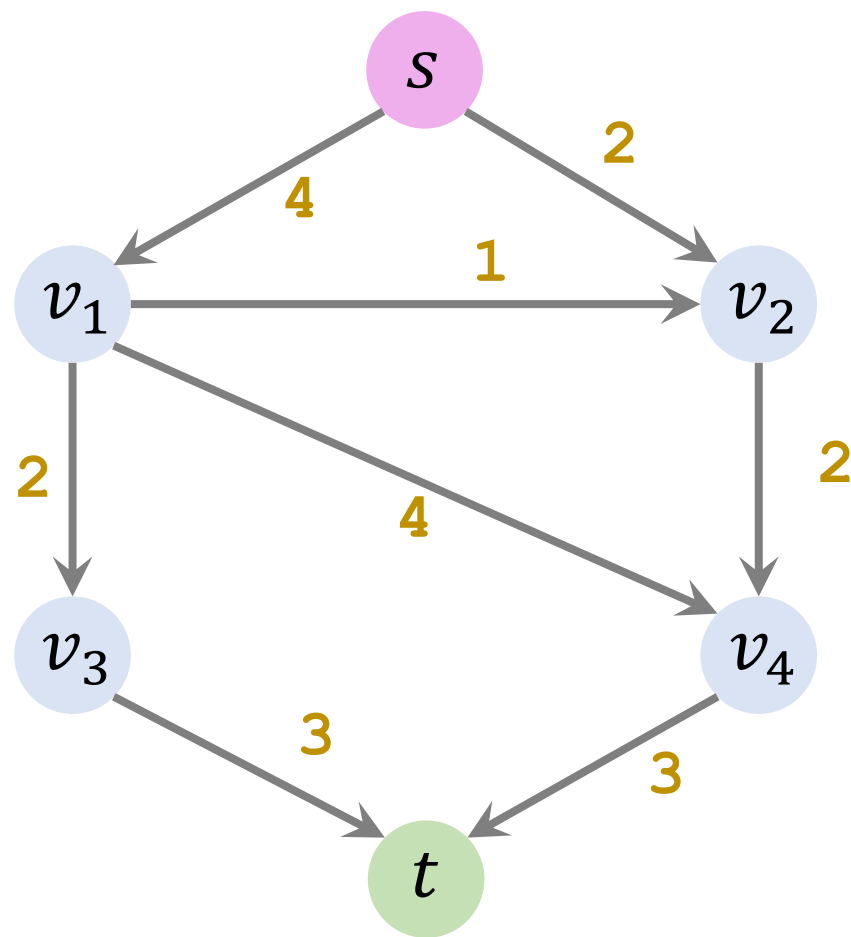


**The naïve algorithm can fail!**

# Initial State

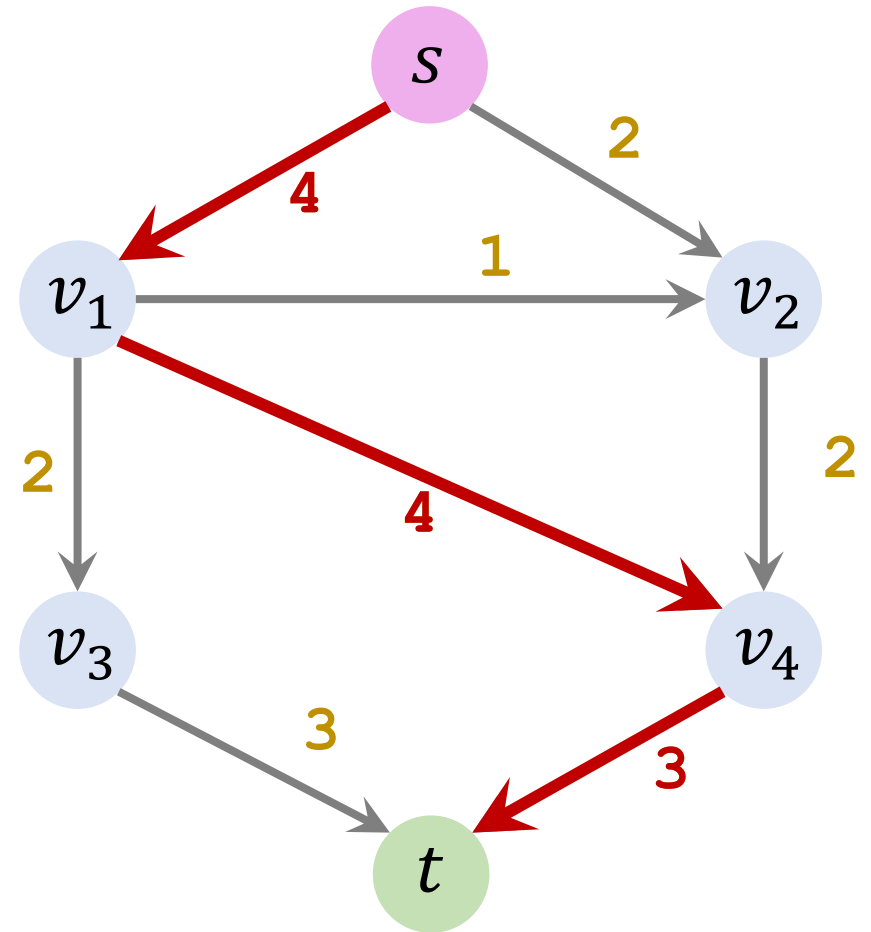


Original Graph



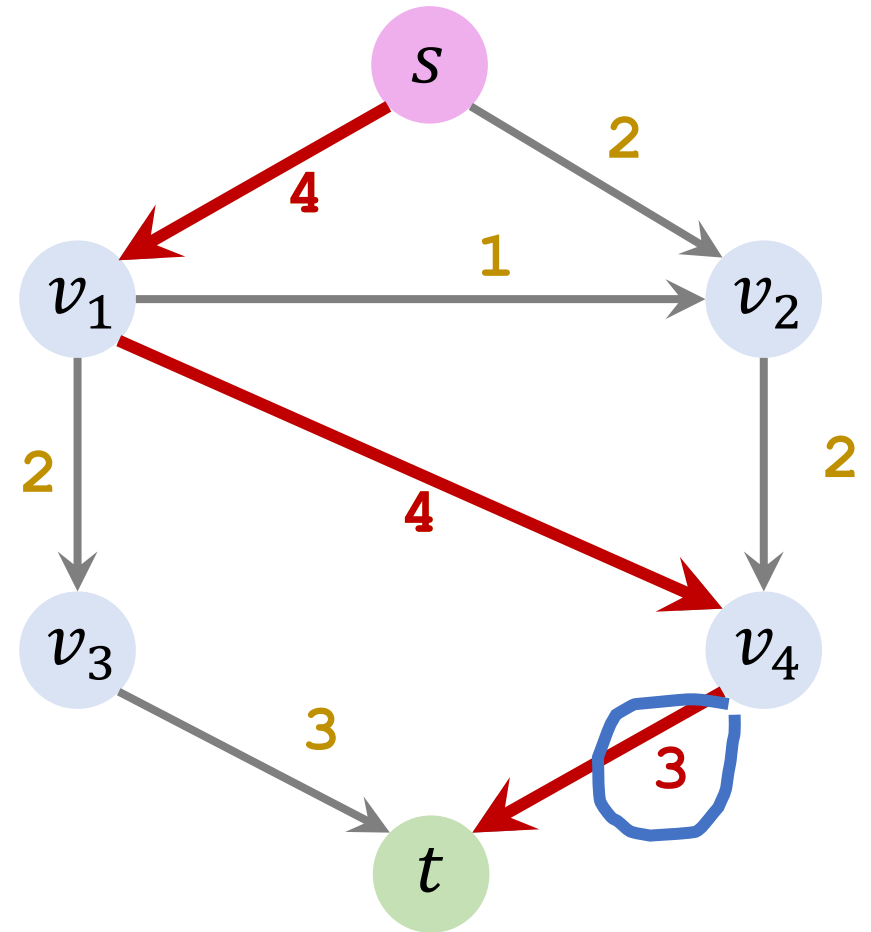
Residual Graph

# Iteration 1: Find an augmenting path



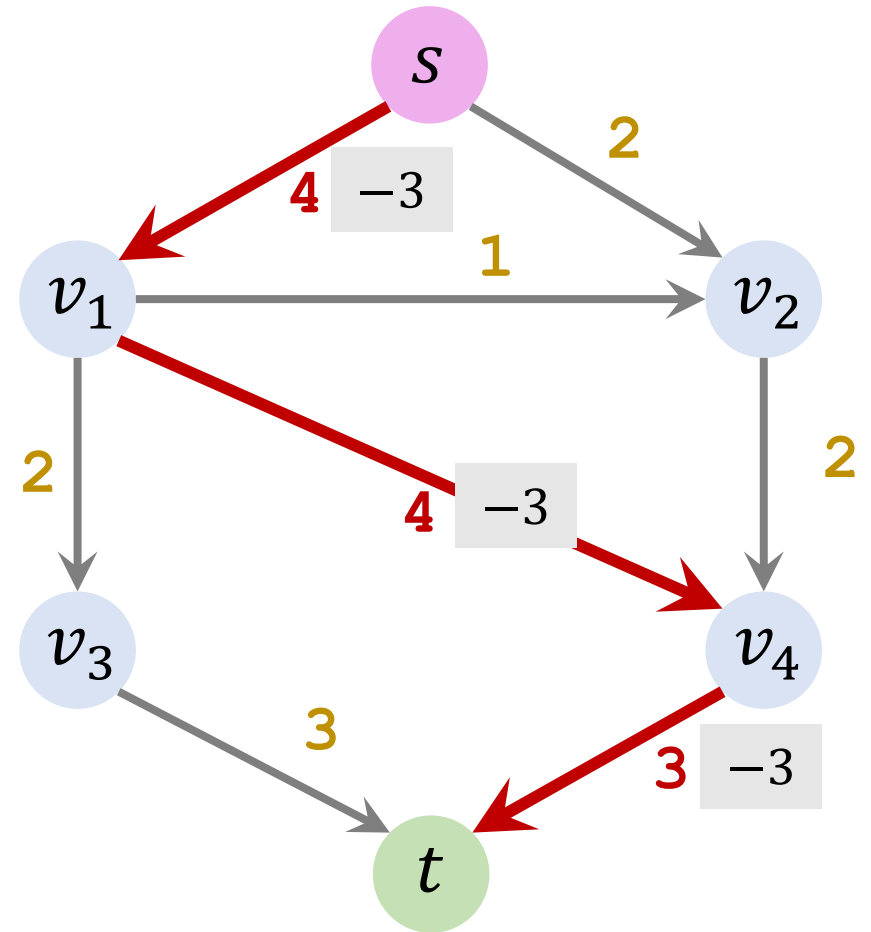
Found path  $s \rightarrow v_1 \rightarrow v_4 \rightarrow t$ .

# Iteration 1: Find an augmenting path

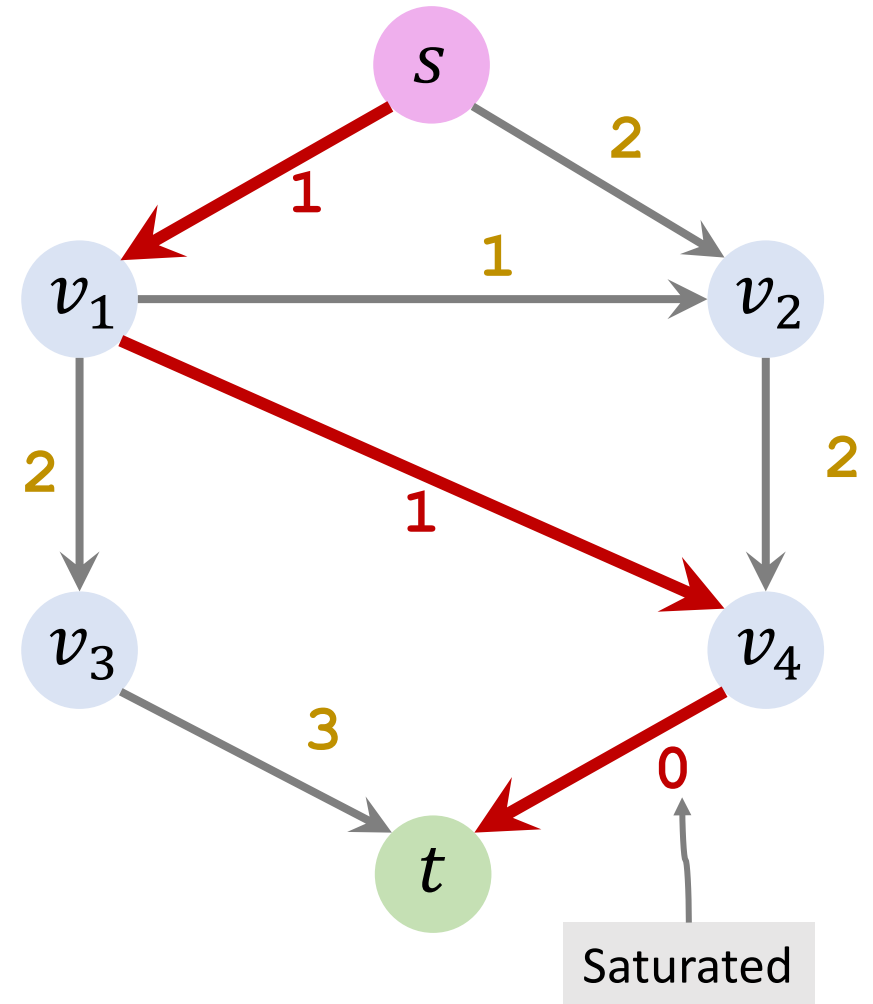


Found path  $s \rightarrow v_1 \rightarrow v_4 \rightarrow t$ . (Bottleneck capacity = 3.)

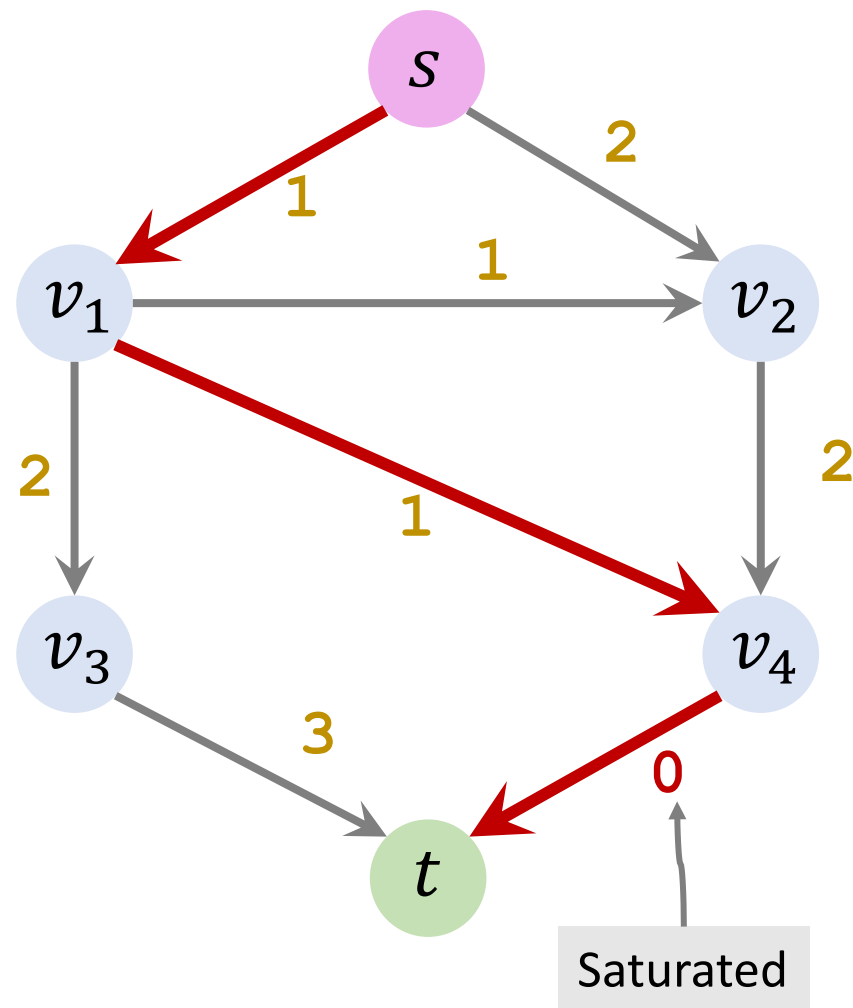
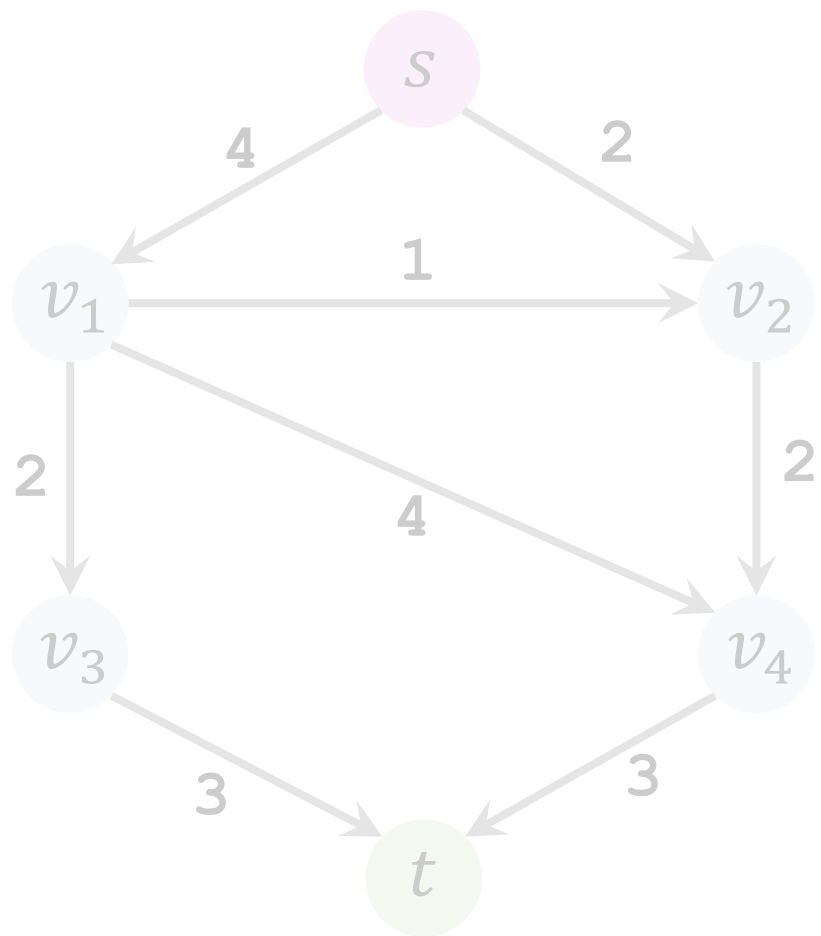
# Iteration 1: Update residuals



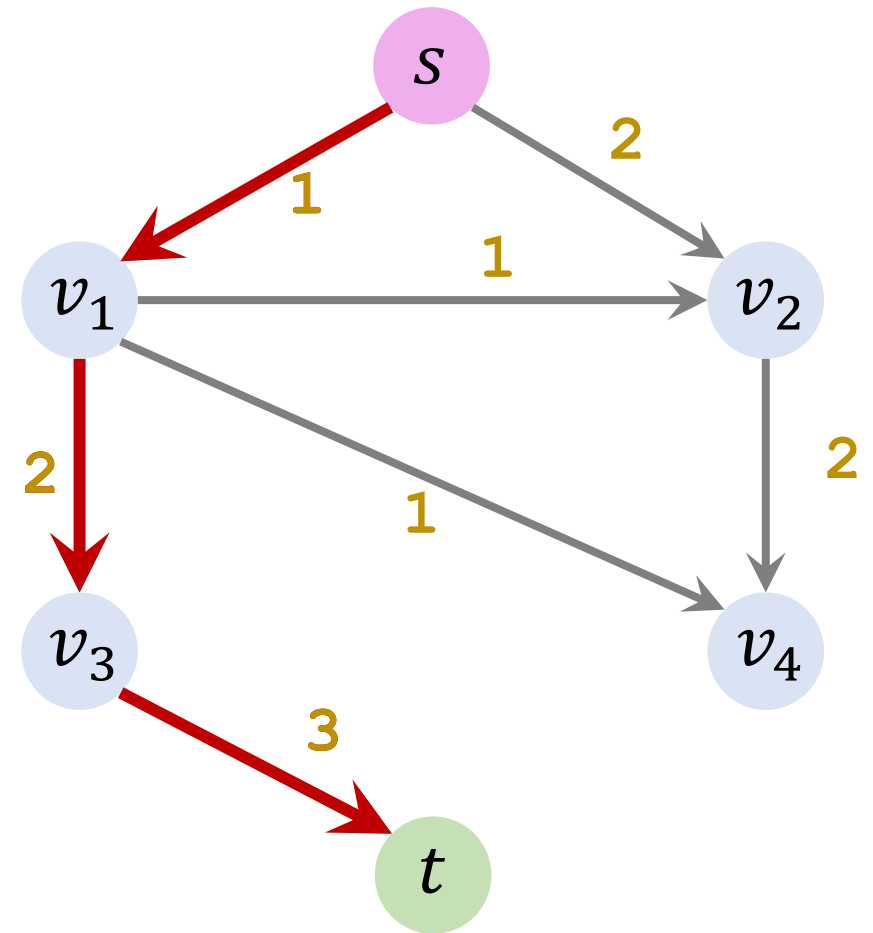
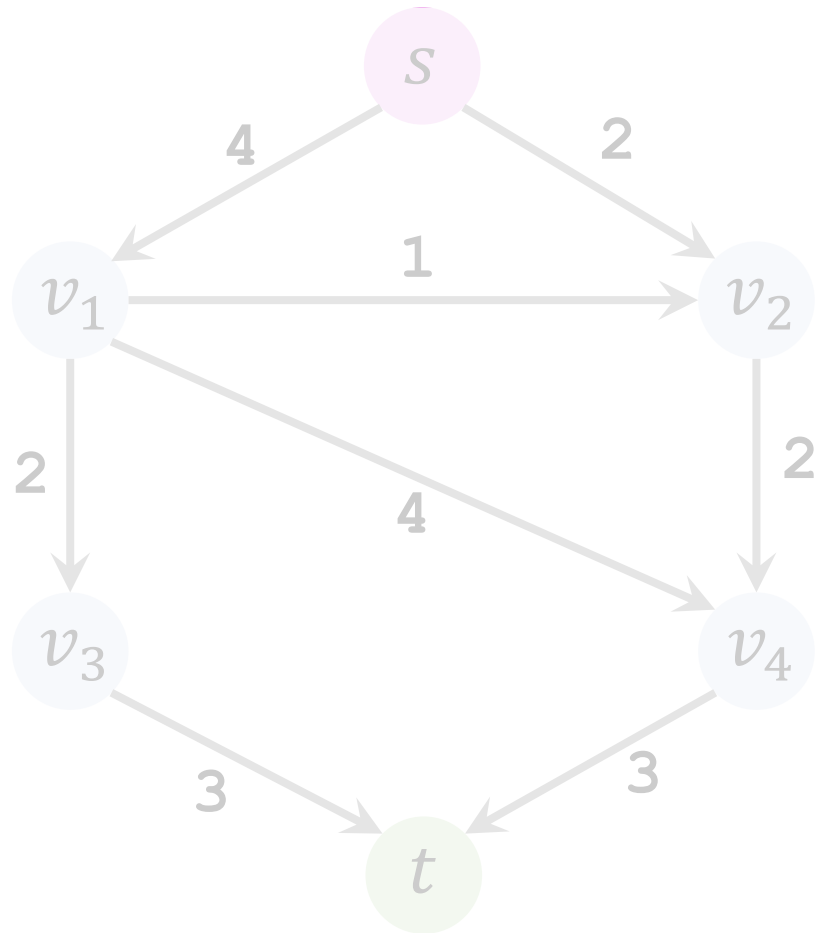
# Iteration 1: Update residuals



# Iteration 1: Remove saturated edges



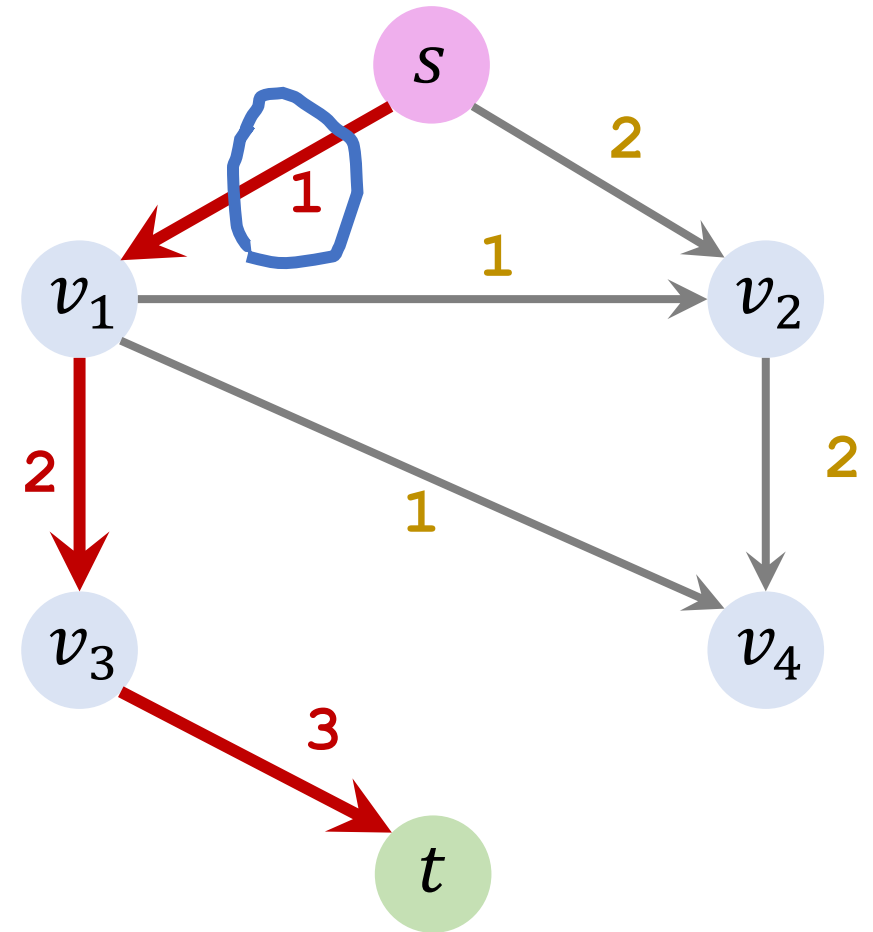
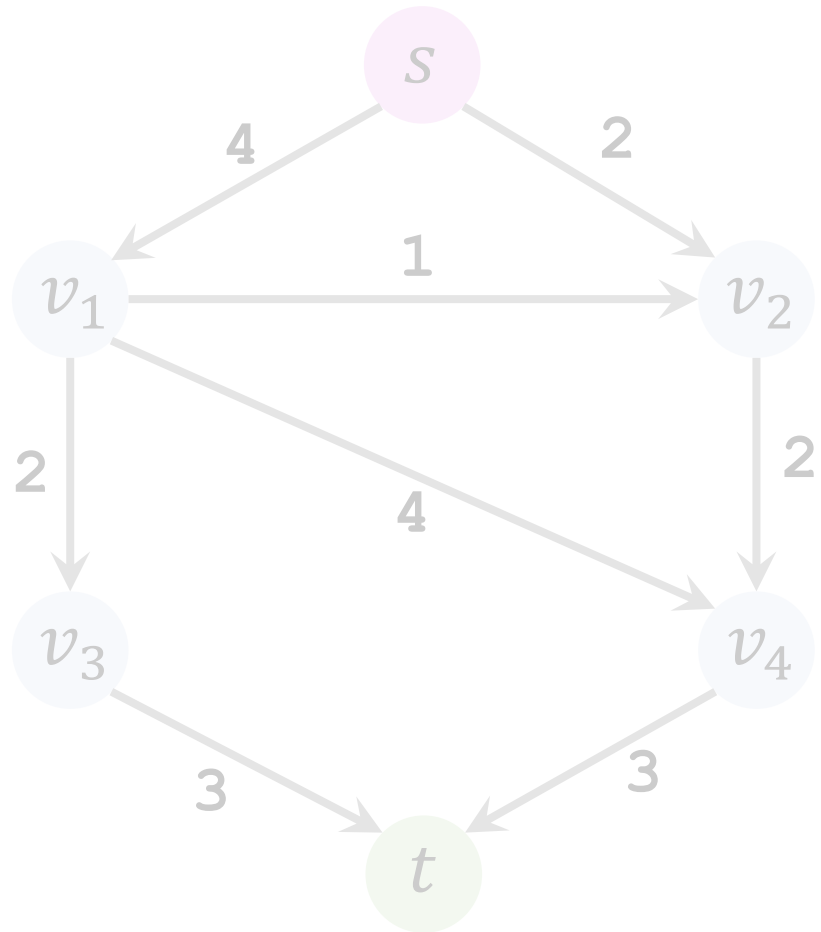
# Iteration 2: Find an augmenting path



Found path  $s \rightarrow v_1 \rightarrow v_3 \rightarrow t$ .

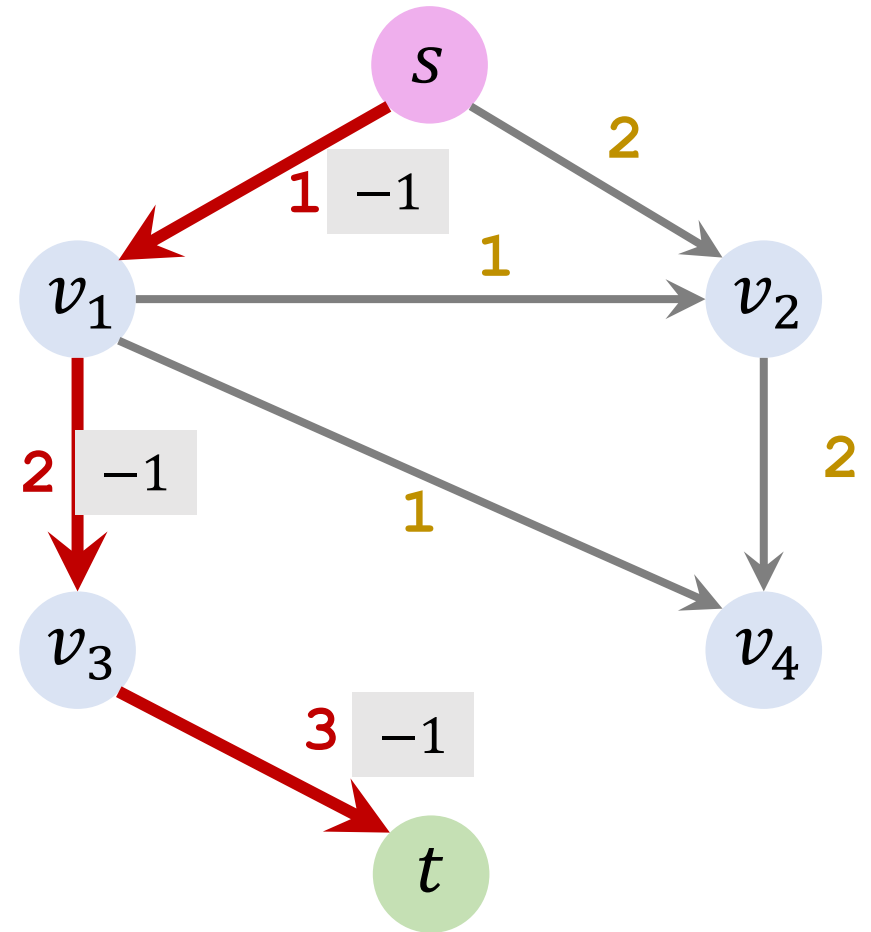
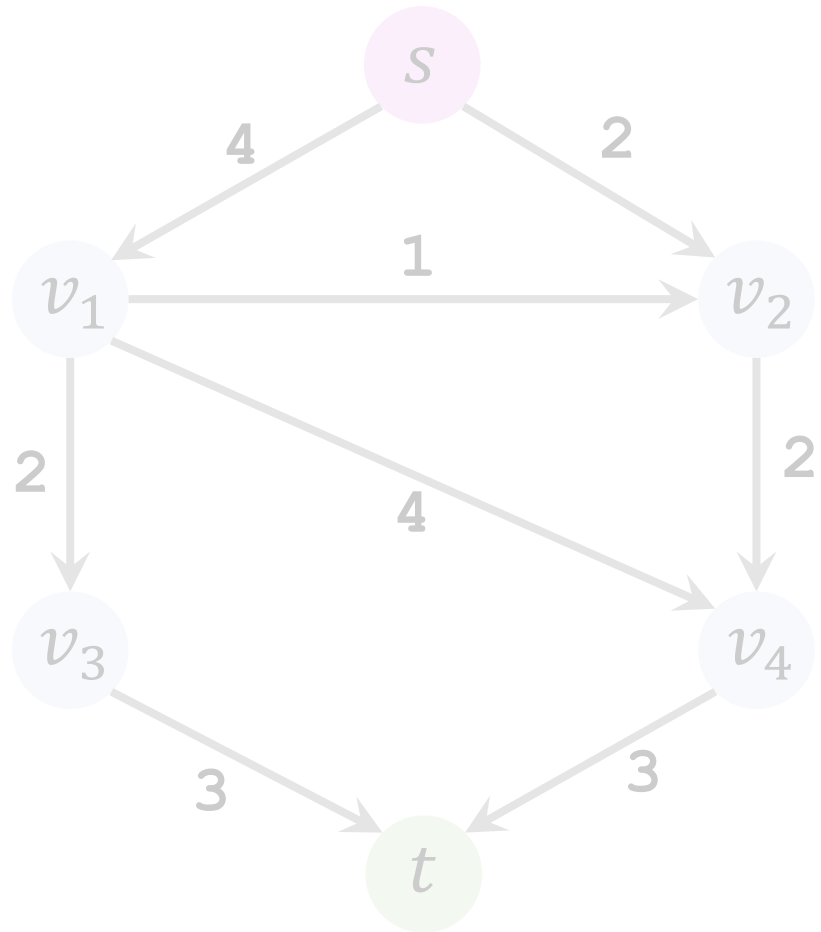


# Iteration 2: Find an augmenting path

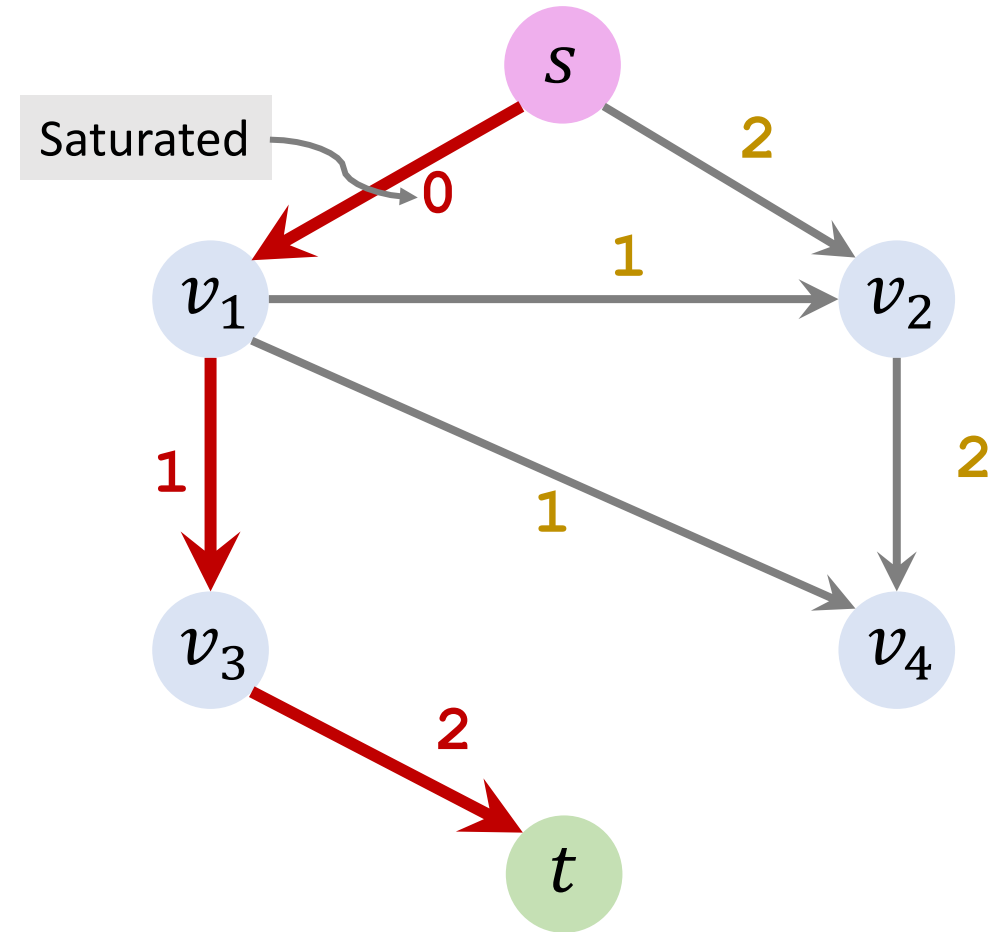


Found path  $s \rightarrow v_1 \rightarrow v_3 \rightarrow t$ . (Bottleneck capacity = 1.)

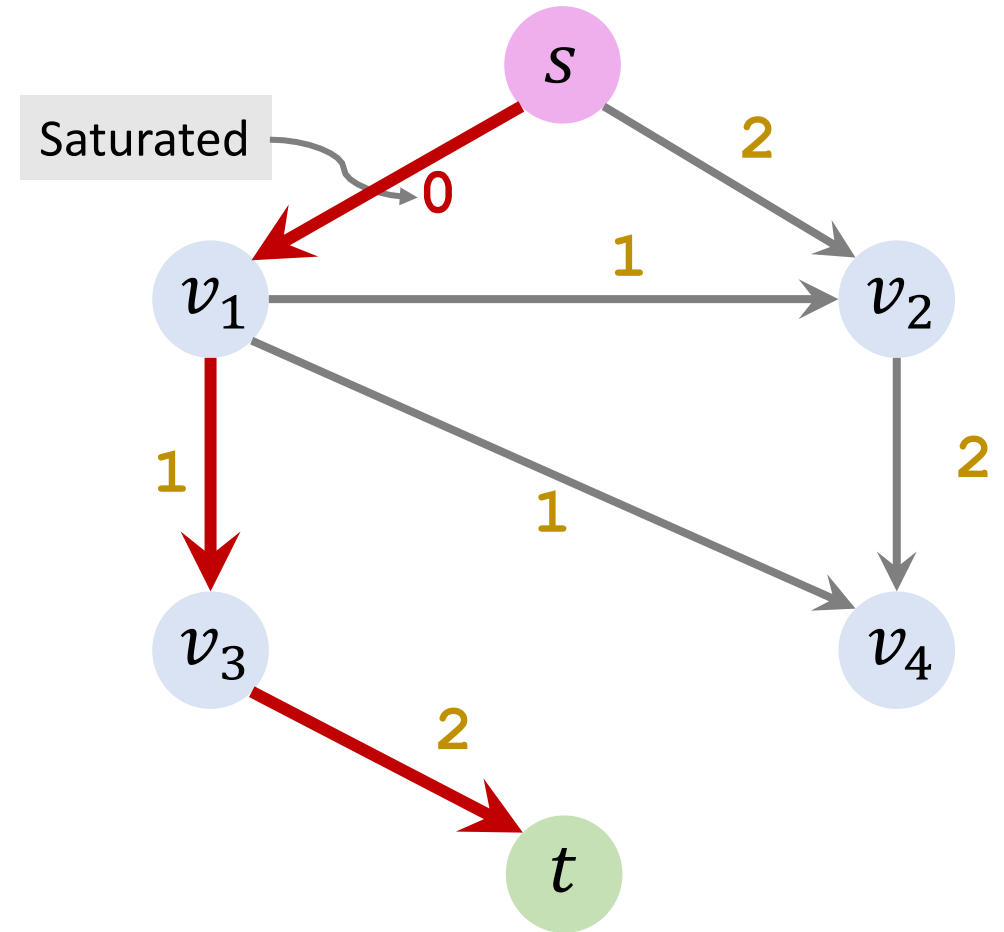
# Iteration 2: Update residuals



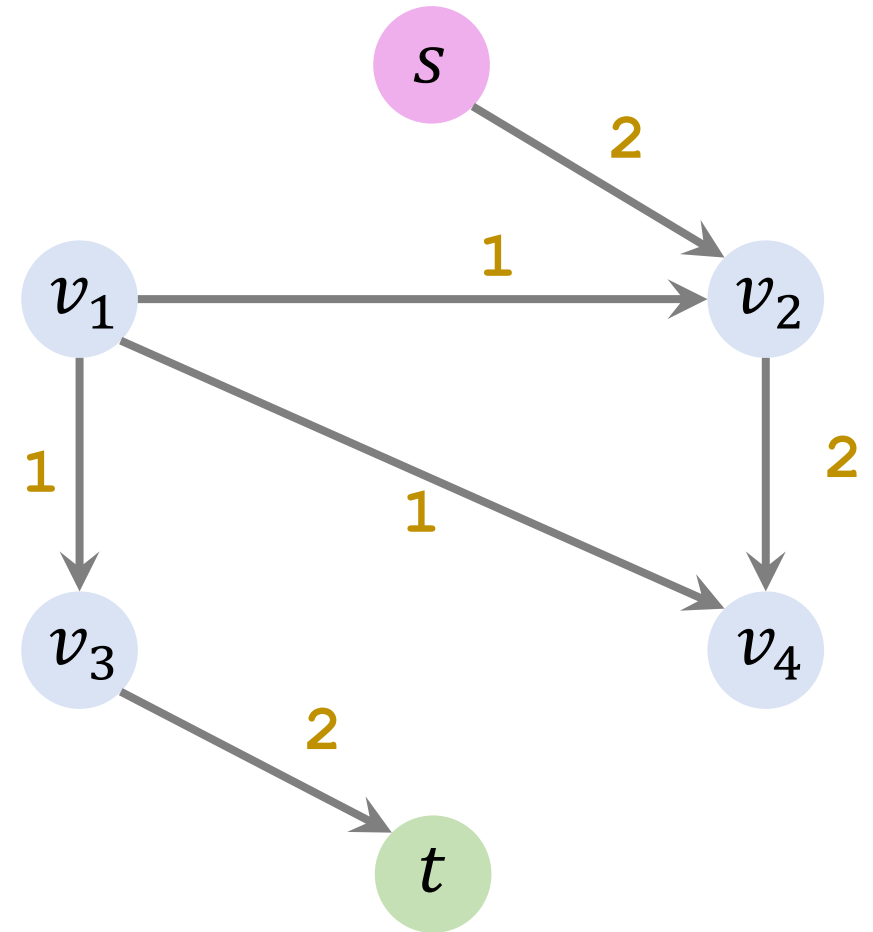
# Iteration 2: Update residuals



# Iteration 2: Remove saturated edges

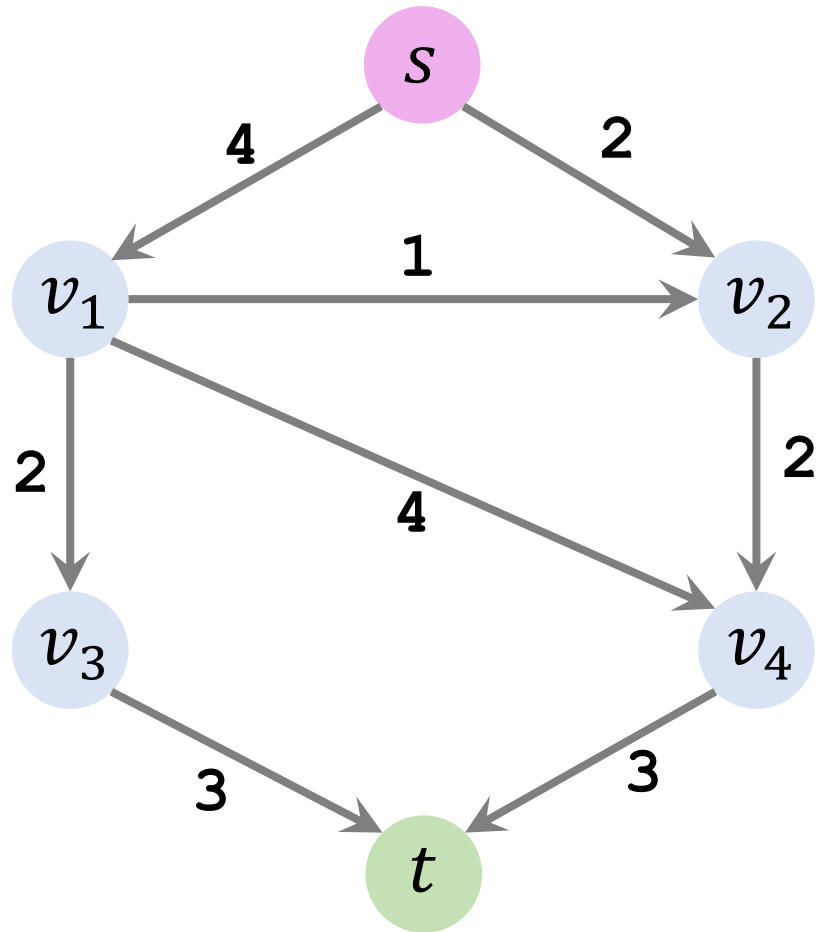


# Iteration 3: Find an augmenting path

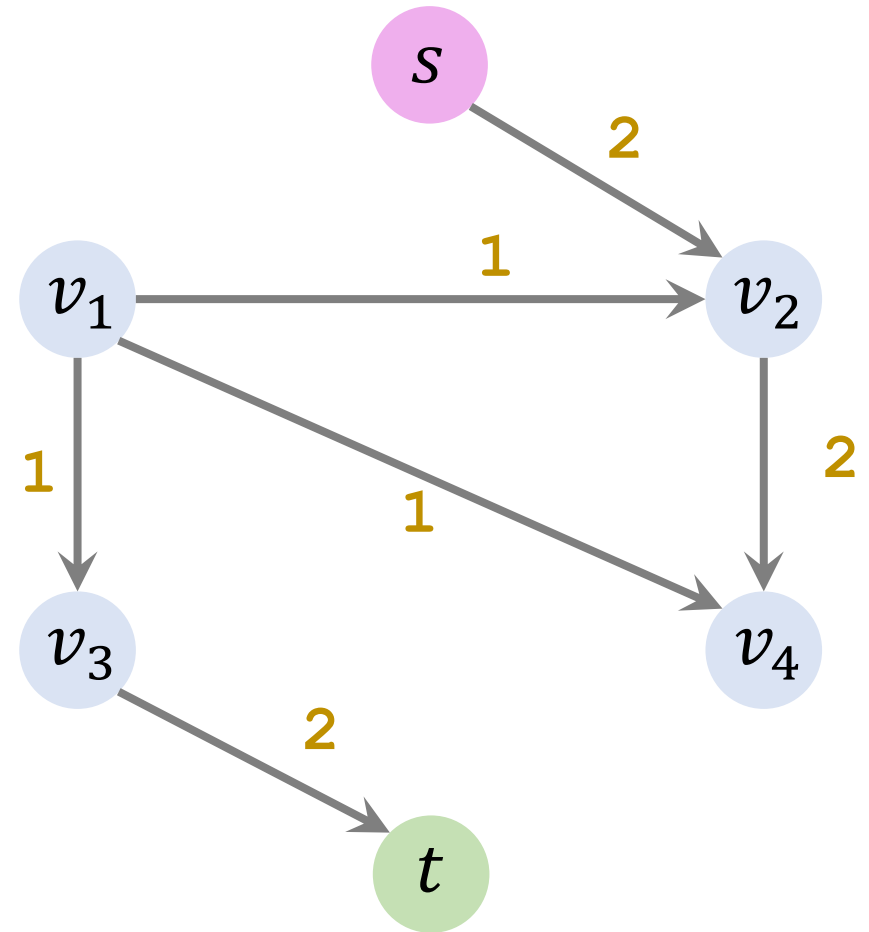


Cannot find any path from source to sink.

# End of Procedure

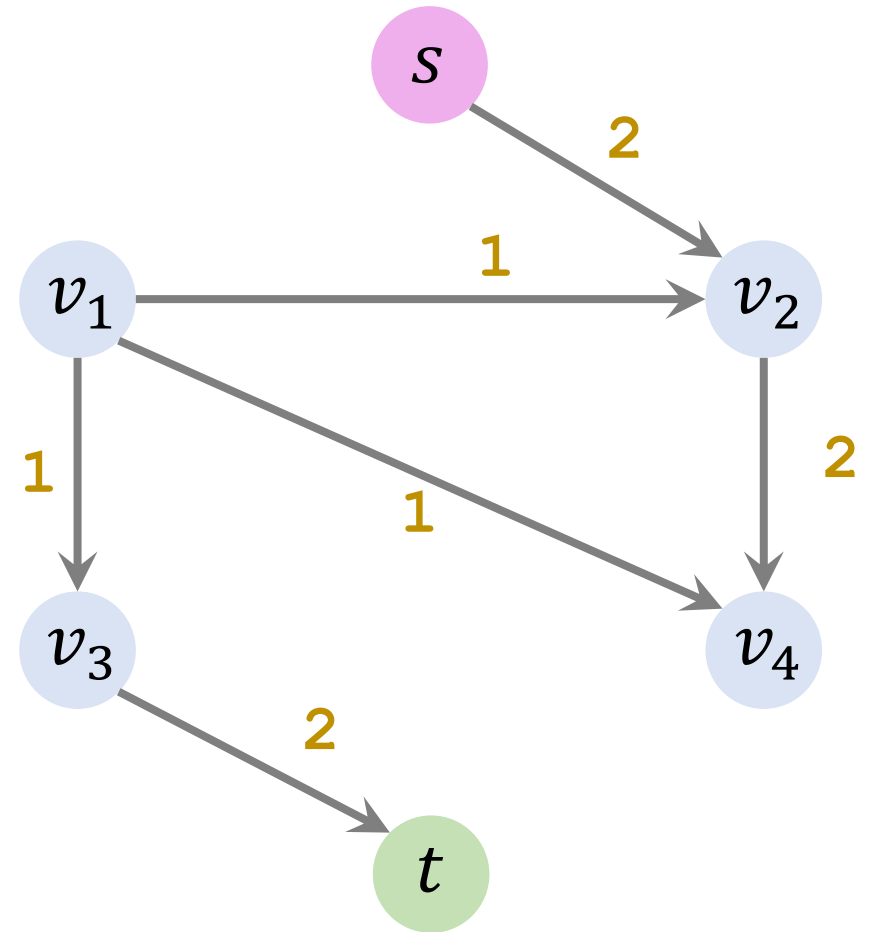
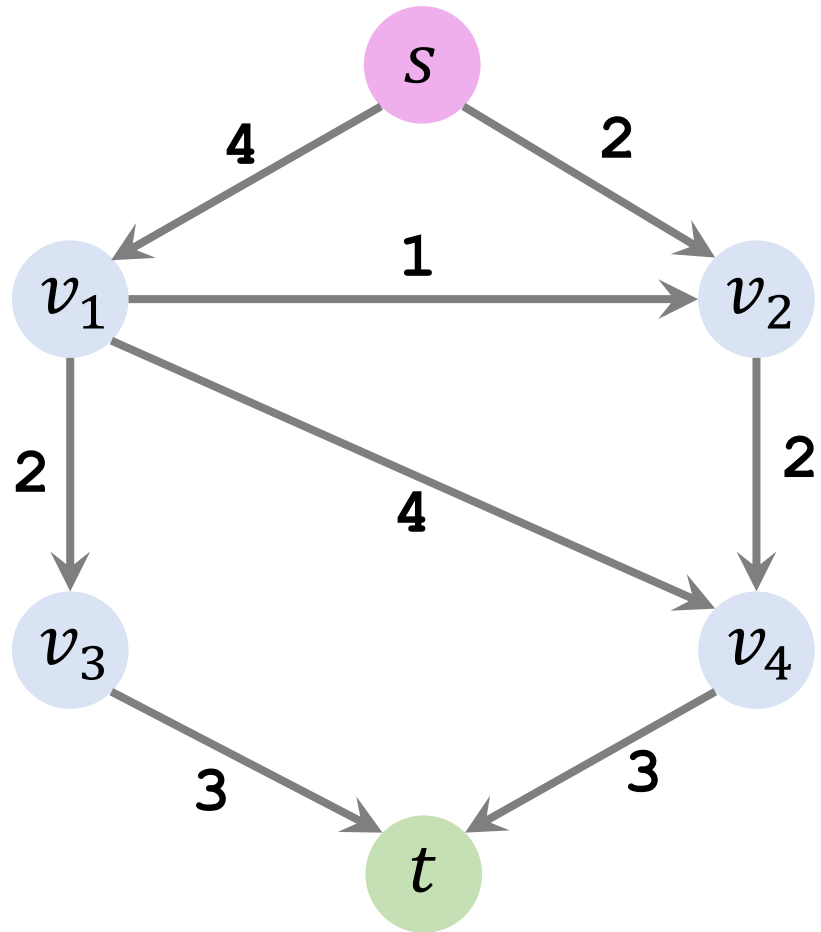


Original Graph



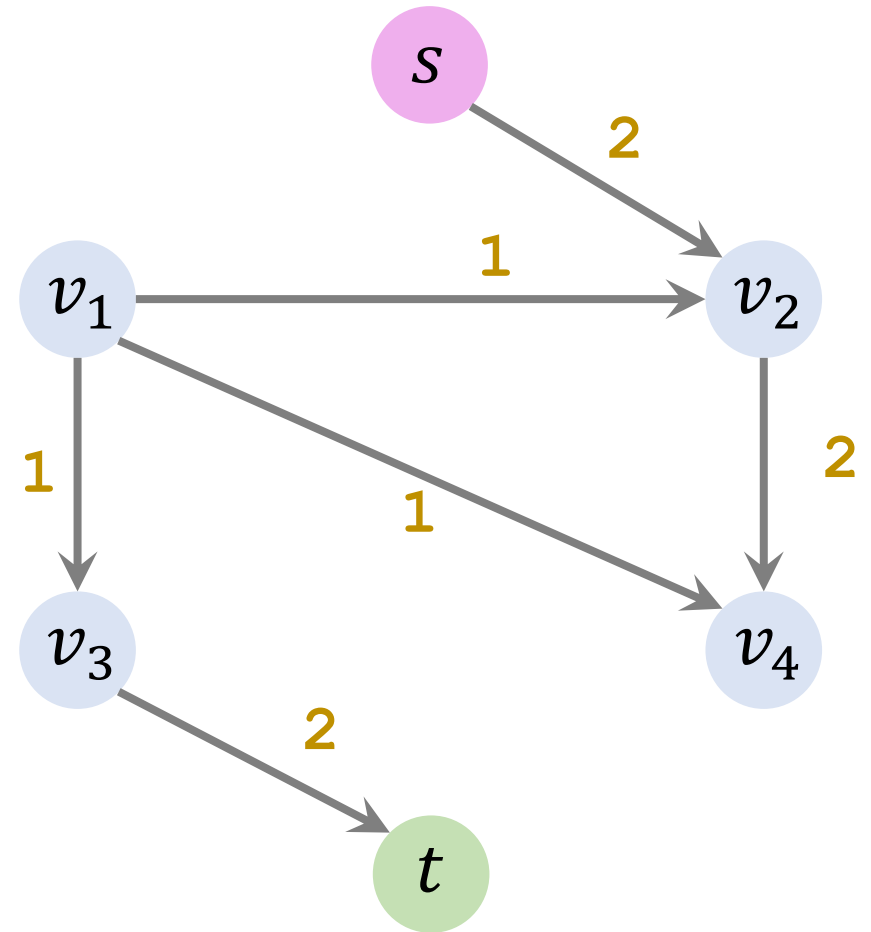
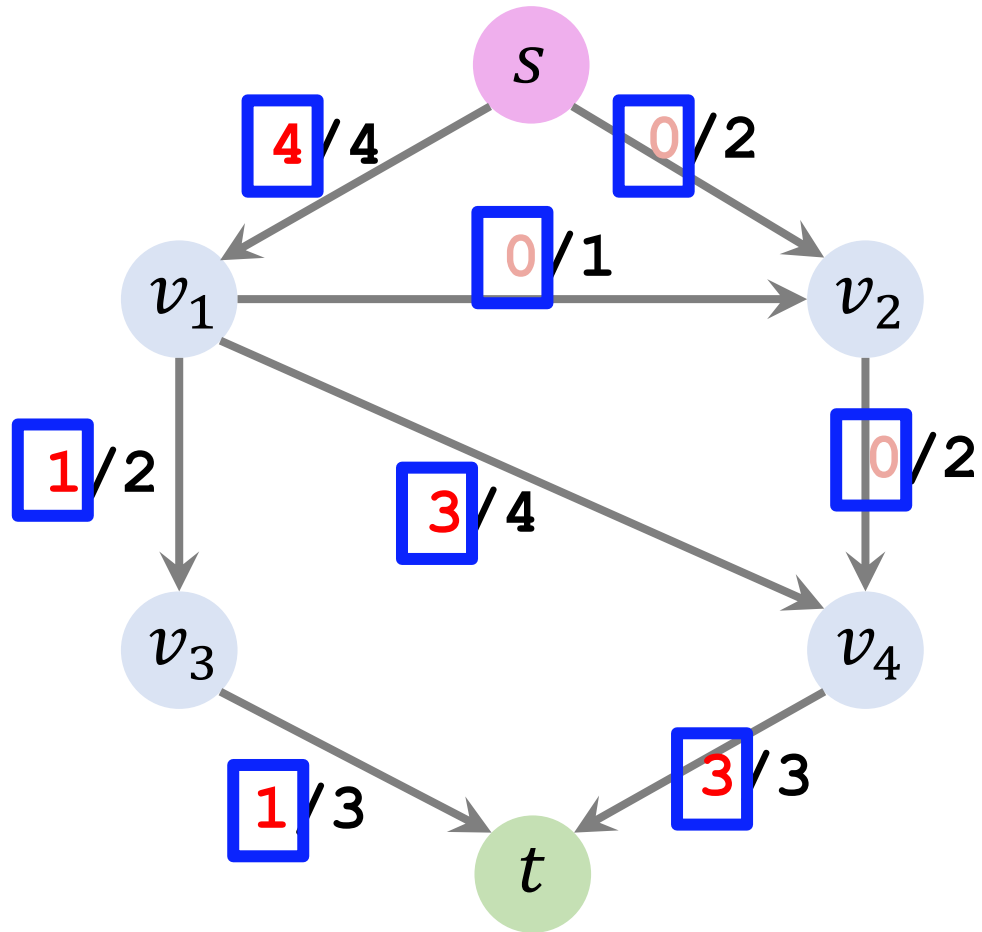
Residual Graph

# End of Procedure



Flow = Capacity - Residual.

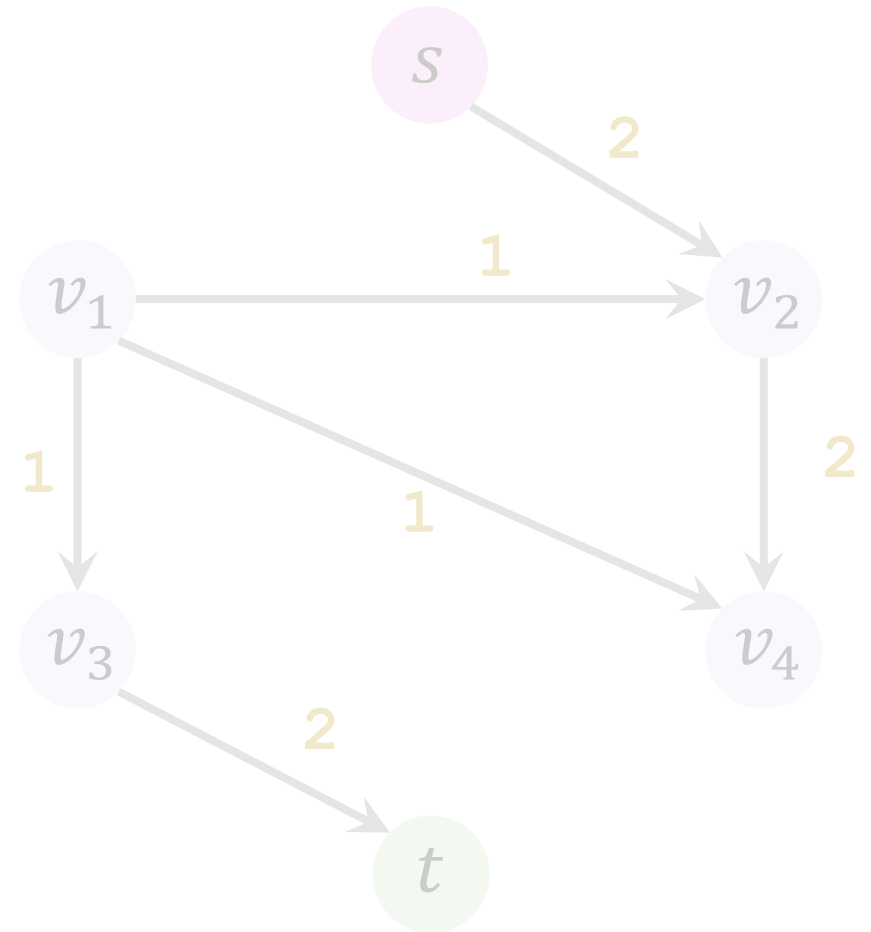
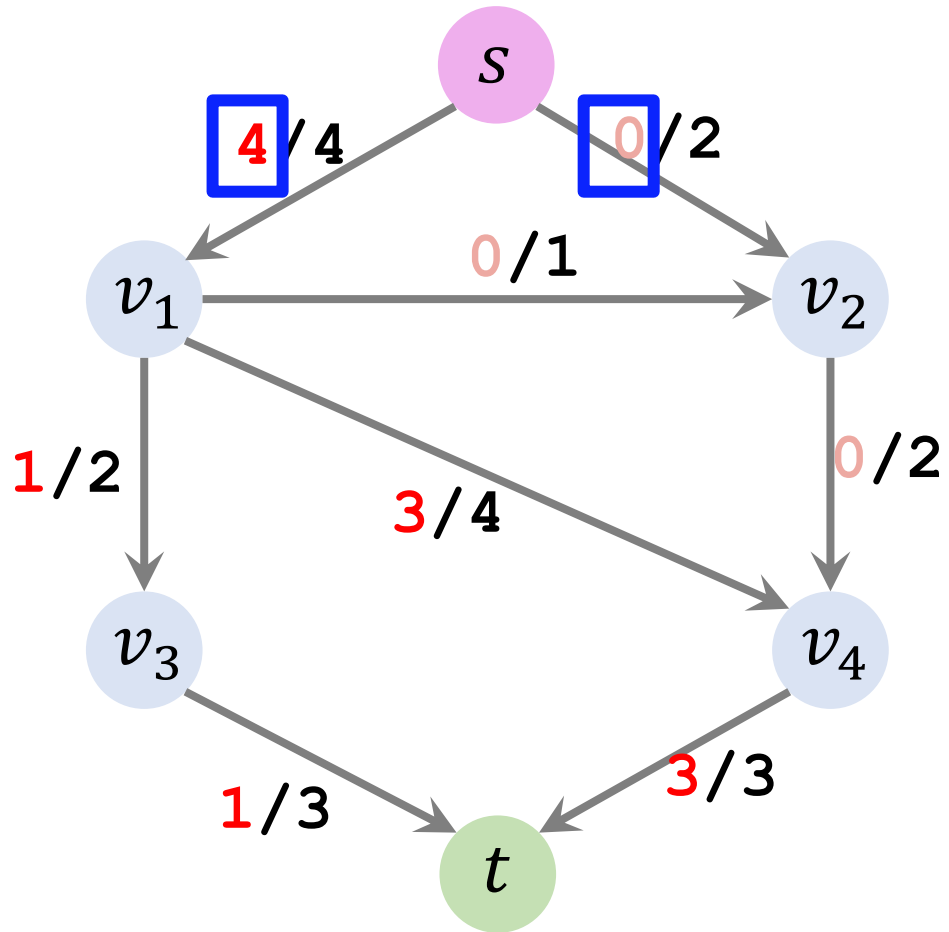
# End of Procedure



Flow = Capacity - Residual.

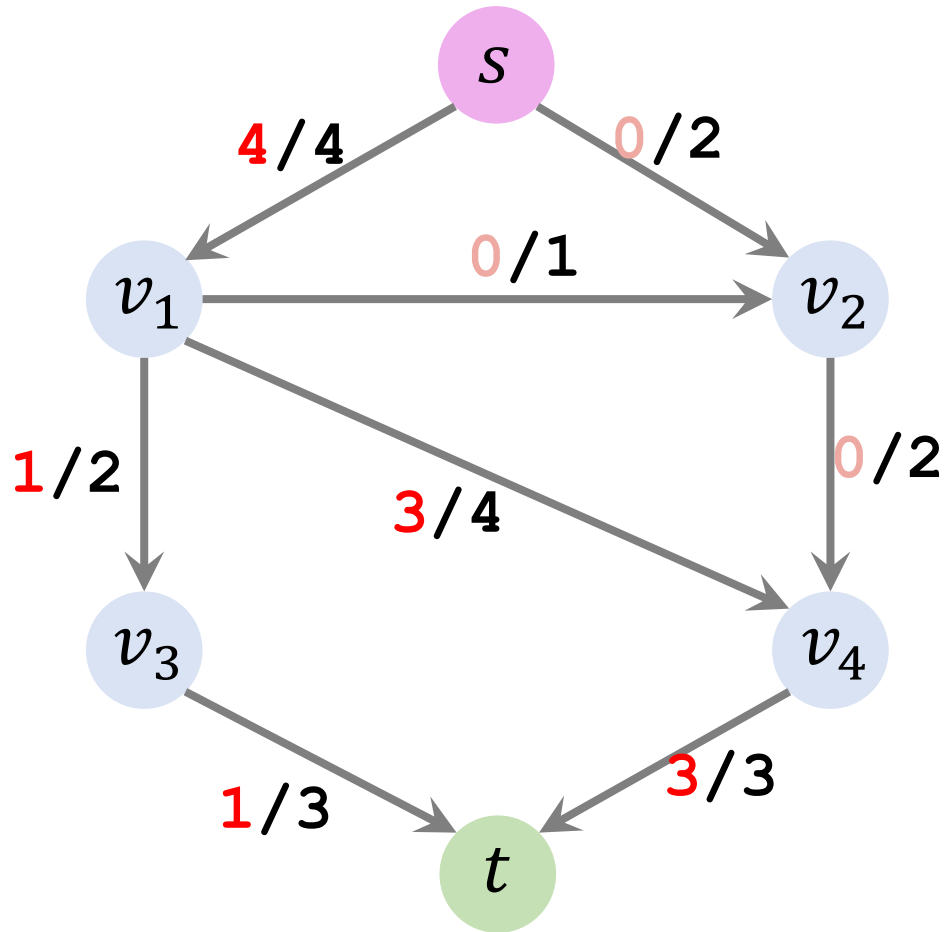


# End of Procedure

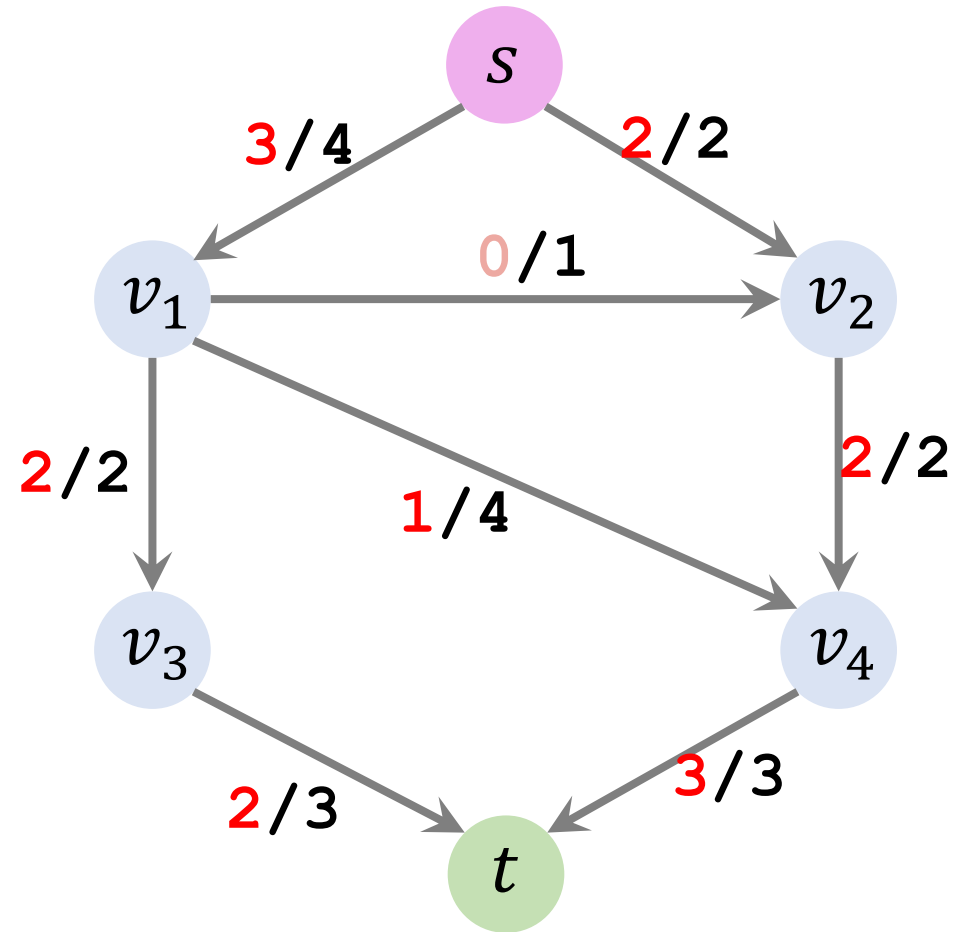


Amount of Flow = 4.

# The result is not maximum flow!

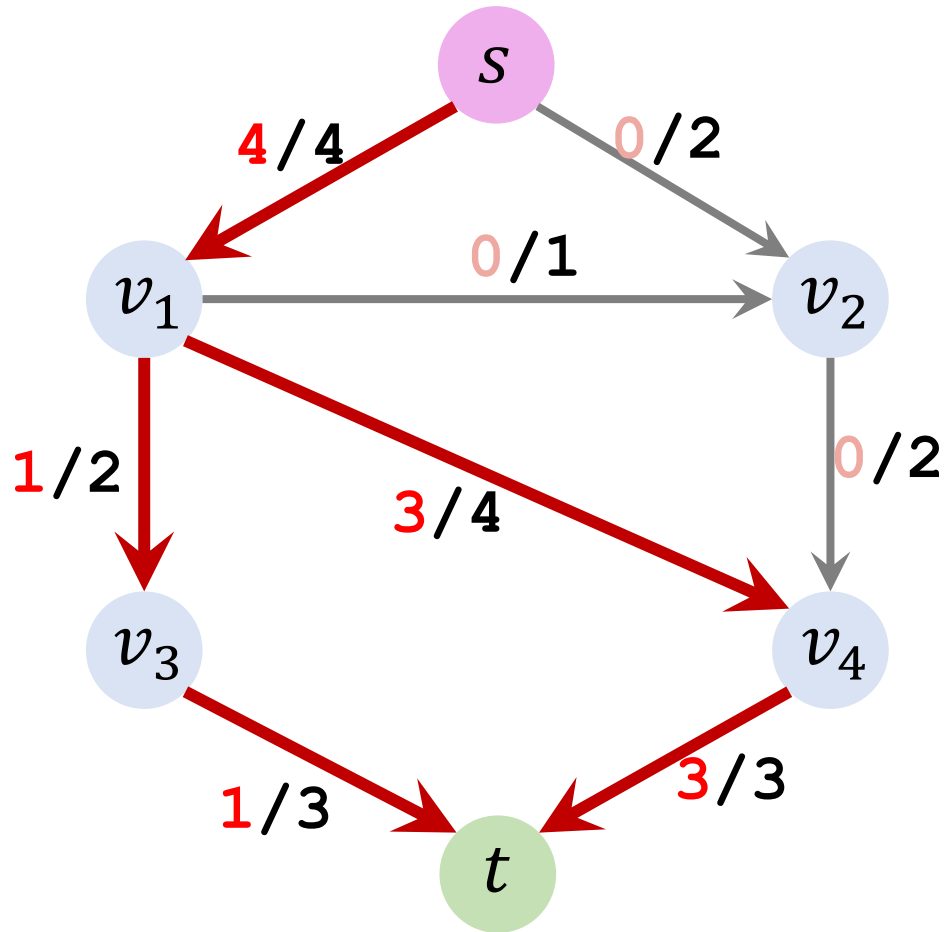


Flow = 4. (Not maximum!)



Flow = 5. (Maximum!)

# Blocking Flow



- A flow is **blocking flow** if no more flow from source to sink can be found.
- The “pipes” are blocked.
- **Maximum flow** is also **blocking flow**.

# Summary


# Maximum Flow Problem

- **Inputs:** a weighted directed graph, the source  $s$ , and the sink  $t$ .
- **Goal:** Send as much water as possible from  $s$  to  $t$ .
- **Constraints:**
  - Each edge has a weight (i.e., the capacity of the pipe).
  - The flow must not exceed the capacity.

# Naïve Algorithm

1. Build a **residual graph**; initialize the residuals to the capacity.

# Naïve Algorithm

1. Build a residual graph; initialize the residuals to the capacity.
2. While **augmenting path** can be found:
  -  a. Find an augmenting path (on the residual graph.)
  - b. Find the bottleneck capacity  $x$  in the augmenting path.
  - c. Update the residuals. ( $\text{residual} \leftarrow \text{residual} - x$ .)

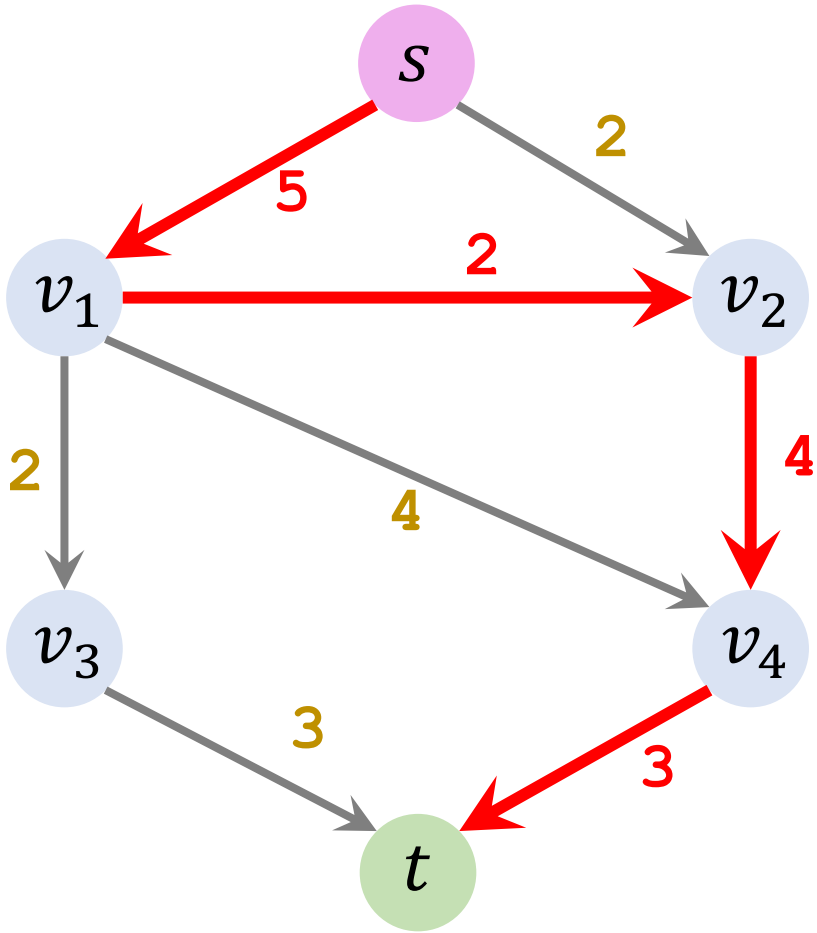
# The naïve algorithm can fail

- The naïve algorithm always finds the **blocking flow**.
- However, the outcome may not be the **maximum flow**.



# Questions

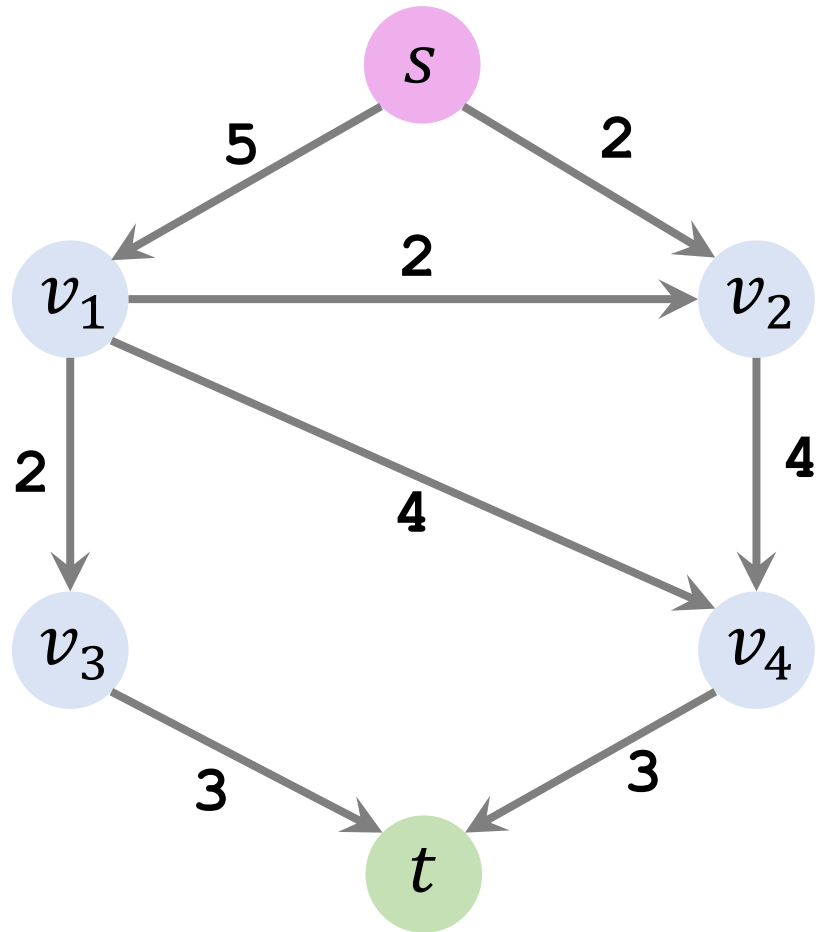
# Q1: Bottleneck Capacity



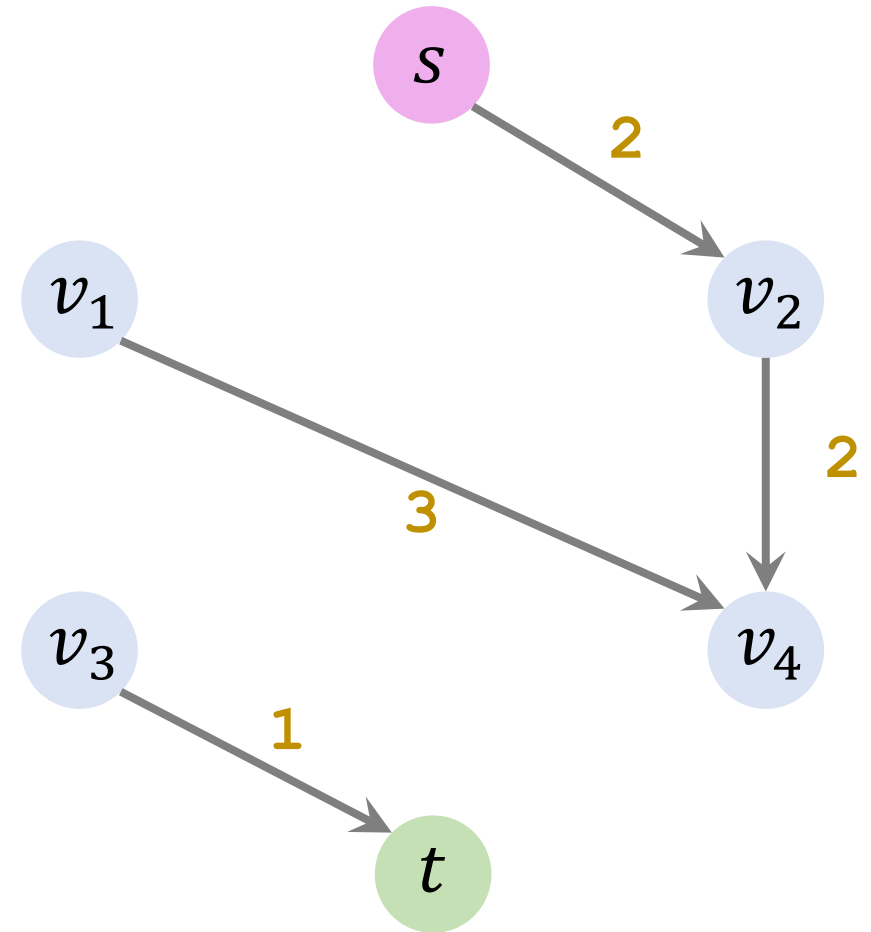
- **Question:** What is the bottleneck of the red path?

Residual Graph

**Q2: What is the amount of flow from  $s$  to  $t$ ?**



Original Graph



Final Residual Graph

**Thank You!**

<http://wangshusen.github.io/>