Flows

Proseminar Algorithmen auf Graphen

Nils Wagner

RWTH Aachen University

May 21, 2024

Table of Contents

Consider the following problem:

- production facility s

Consider the following problem:

- production facility s
- rail network through cities A, B, C, D, s

Consider the following problem:

- production facility s
- rail network through cities A, B, C, D, s, and t
- destination t

Consider the following problem:

- production facility s
- rail network through cities A, B, C, D, s, and t
- destination t
- limited amount of goods per day from one city to another

Consider the following problem:

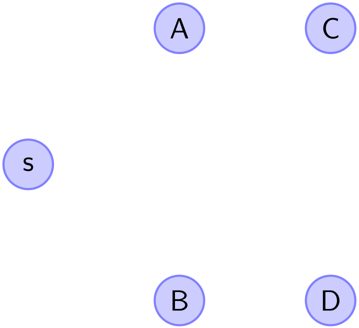
- production facility s
- rail network through cities A, B, C, D, s, and t
- destination t
- limited amount of goods per day from one city to another
- How many goods can be transported from s to t per day?

Consider the following problem:

- production facility s
- rail network through cities A, B, C, D, s, and t
- destination t
- limited amount of goods per day from one city to another
- How many goods can be transported from s to t per day?

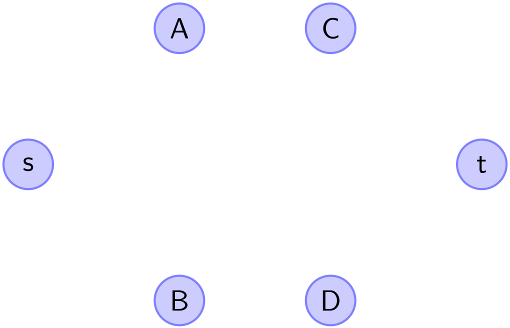
Consider the following problem:

- production facility s
- rail network through cities A, B, C, D, s, and t
- destination t
- limited amount of goods per day from one city to another
- How many goods can be transported from s to t per day?



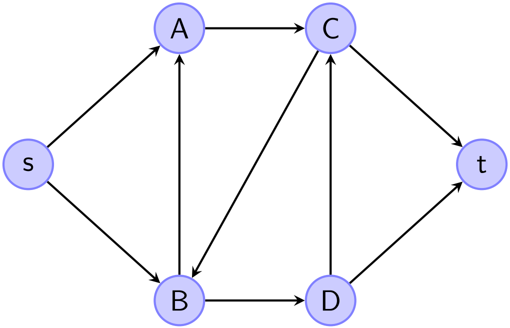
Consider the following problem:

- production facility s
- rail network through cities A, B, C, D, s, and t
- destination t
- limited amount of goods per day from one city to another
- How many goods can be transported from s to t per day?



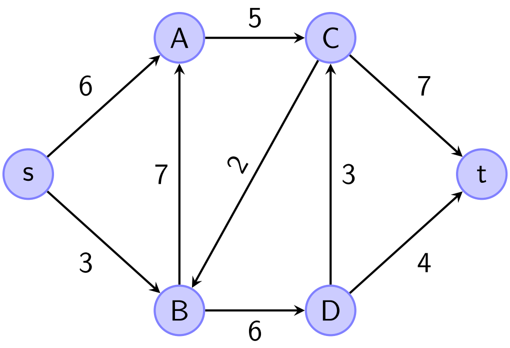
Consider the following problem:

- production facility s
- rail network through cities A, B, C, D, s, and t
- destination t
- limited amount of goods per day from one city to another
- How many goods can be transported from s to t per day?



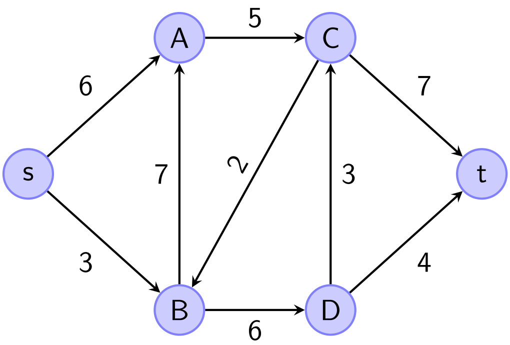
Consider the following problem:

- production facility s
- rail network through cities A, B, C, D, s, and t
- destination t
- limited amount of goods per day from one city to another
- How many goods can be transported from s to t per day?



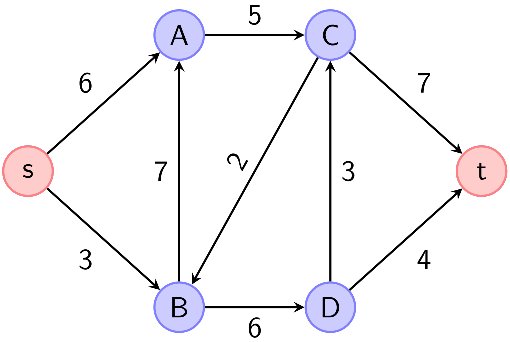
Consider the following problem:

- production facility s
- rail network through cities A, B, C, D, s, and t (flow network)
- destination t
- limited amount of goods per day from one city to another
- How many goods can be transported from s to t per day?



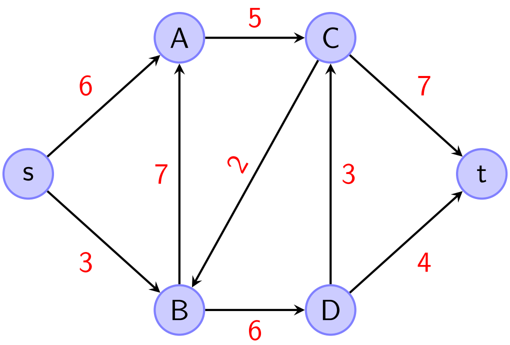
Consider the following problem:

- production facility s (source)
- rail network through cities A, B, C, D, s, and t *(flow network)*
- destination t (sink)
- limited amount of goods per day from one city to another
- How many goods can be transported from s to t per day?



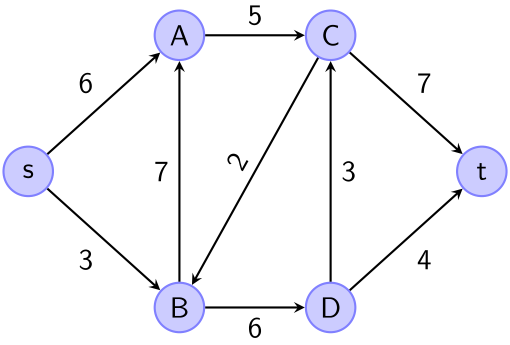
Consider the following problem:

- production facility s *(source)*
- rail network through cities A, B, C, D, s, and t *(flow network)*
- destination t *(sink)*
- limited amount of goods per day from one city to another (capacity)
- How many goods can be transported from s to t per day?



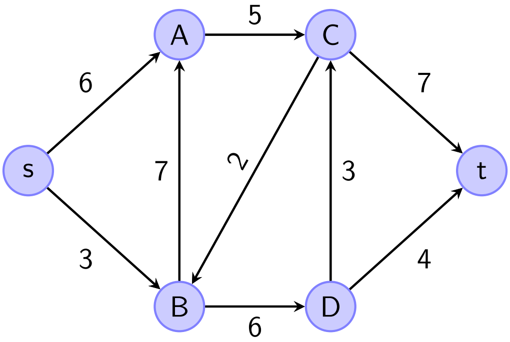
Consider the following problem:

- production facility s *(source)*
- rail network through cities A, B, C, D, s, and t *(flow network)*
- destination t *(sink)*
- limited amount of goods per day from one city to another *(capacity)*
- How many goods can be transported from s to t per day? (maximum flow)



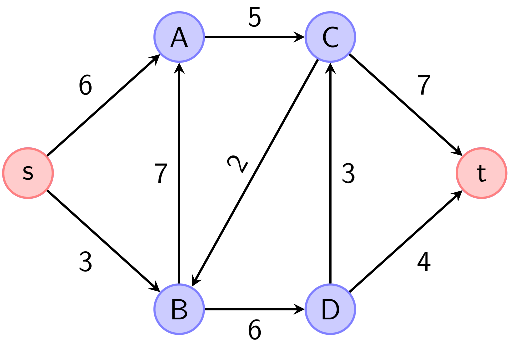
Definition (Flow Network)

A flow network $G = (V, E)$ is a directed graph.



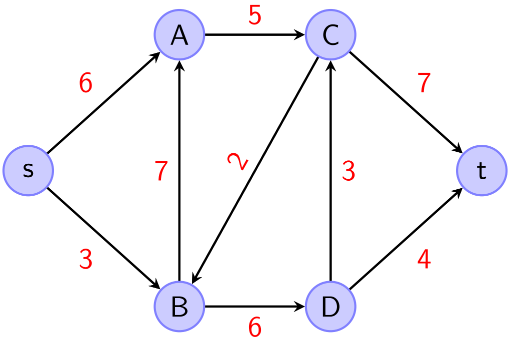
Definition (Flow Network)

A flow network $G = (V, E)$ is a directed graph.
The vertices $s, t \in V$ are called the source and the sink of G.

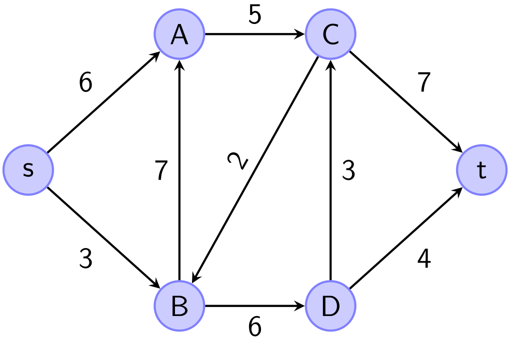


Definition (Flow Network)

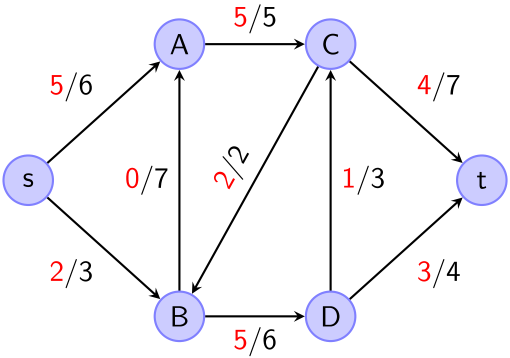
A flow network $G = (V, E)$ is a directed graph.
The vertices $s, t \in V$ are called the source and the sink of G.
The capacity function $c : V \times V \to \mathbb{R}_{\geq 0}$ assigns capacities to the edges of G where $c(u, v) = 0$ for all $(u, v) \notin E$.



Definition (Flow and Value of Flow)

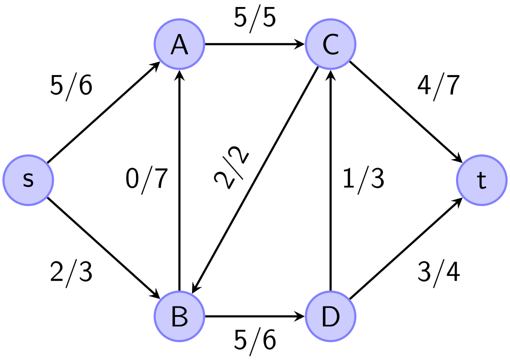


Definition (Flow and Value of Flow)



Definition (Flow and Value of Flow)

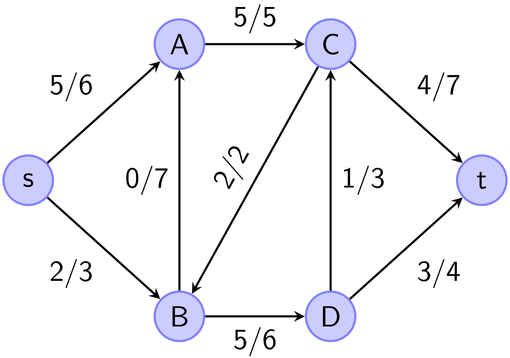
A function $f : V \times V \to \mathbb{R}$ is called a flow in the network $G = (V, E)$ if it satisfies the following properties:



Definition (Flow and Value of Flow)

A function $f : V \times V \to \mathbb{R}$ is called a flow in the network $G = (V, E)$ if it satisfies the following properties:

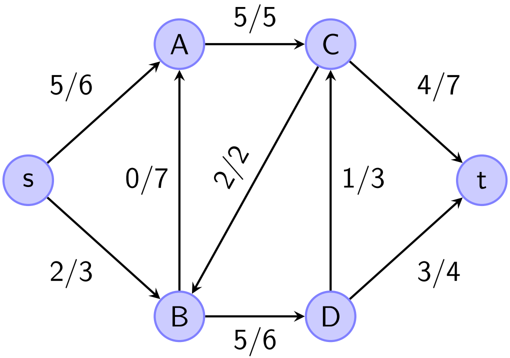
1. $0 \leq f(u, v) \leq c(u, v) \forall u, v, \in V$ (capacity constraint)



Definition (Flow and Value of Flow)

A function $f : V \times V \to \mathbb{R}$ is called a flow in the network $G = (V, E)$ if it satisfies the following properties:

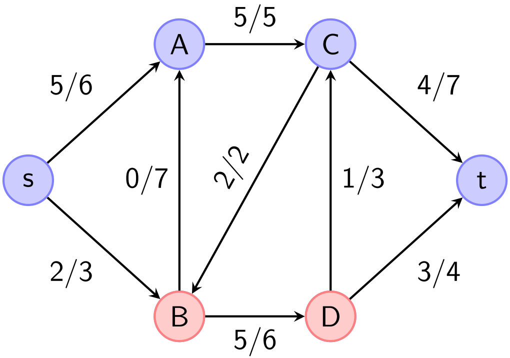
1. $0 \leq f(u, v) \leq c(u, v) \forall u, v, \in V$ (capacity constraint)
2. $\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u) \forall u \in V \setminus \{s, t\}$ (flow conservation)



Definition (Flow and Value of Flow)

A function $f : V \times V \to \mathbb{R}$ is called a flow in the network $G = (V, E)$ if it satisfies the following properties:

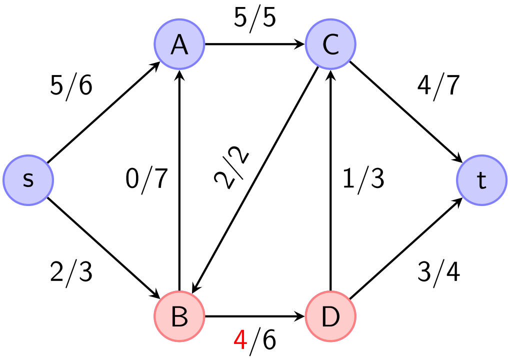
1. $0 \leq f(u, v) \leq c(u, v) \forall u, v, \in V$ (capacity constraint)
2. $\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u) \forall u \in V \setminus \{s, t\}$ (flow conservation)



Definition (Flow and Value of Flow)

A function $f : V \times V \to \mathbb{R}$ is called a flow in the network $G = (V, E)$ if it satisfies the following properties:

1. $0 \leq f(u, v) \leq c(u, v) \forall u, v, \in V$ (capacity constraint)
2. $\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u) \forall u \in V \backslash \{s, t\}$ (flow conservation)



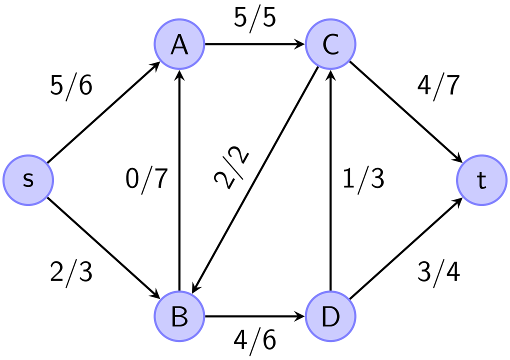
Definition (Flow and Value of Flow)

A function $f : V \times V \to \mathbb{R}$ is called a flow in the network $G = (V, E)$ if it satisfies the following properties:

1. $0 \leq f(u, v) \leq c(u, v) \forall u, v, \in V$ (capacity constraint)
2. $\sum\limits_{v \in V} f(u, v) = \sum\limits_{v \in V} f(v, u) \forall u \in V \setminus \{s, t\}$ (flow conservation)

The value of a flow $|f|$ is given by

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s).$$



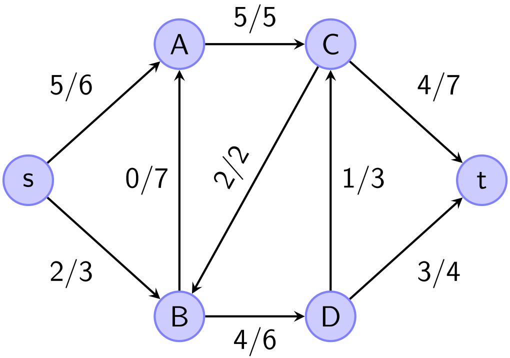
Definition (Flow and Value of Flow)

A function $f : V \times V \to \mathbb{R}$ is called a flow in the network $G = (V, E)$ if it satisfies the following properties:

1. $0 \le f(u, v) \le c(u, v) \forall u, v, \in V$ (capacity constraint)
2. $\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u) \forall u \in V \setminus \{s, t\}$ (flow conservation)

The value of a flow $|f|$ is given by

$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s).$$



$$\begin{aligned} |f| &= \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) \\ &= \quad 5 + 2 \quad - \quad 0 \quad = 7 \end{aligned}$$

Maximum-Flow Problem

Problem

Given a flow network G with source s, sink t, and a capacity function c.
Find a flow f in G of maximum value.

Maximum-Flow Problem

Problem

Given a flow network G with source s, sink t, and a capacity function c. Find a flow f in G of maximum value.

Applications:

- water flowing through network of pipes/rivers
- current in electrical network
- traffic: cars in road network
- messages in telecommunication network

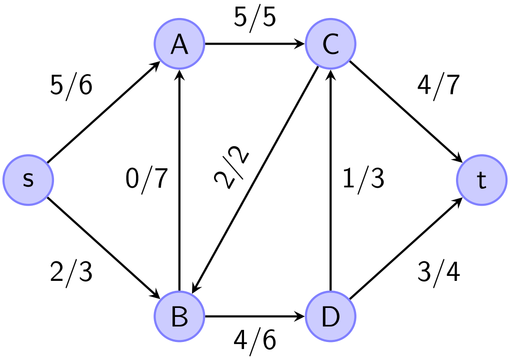
Maximum-Flow Problem

Problem

Given a flow network G with source s, sink t, and a capacity function c.
Find a flow f in G of maximum value.

Applications:

- water flowing through network of pipes/rivers
- current in electrical network
- traffic: cars in road network
- messages in telecommunication network



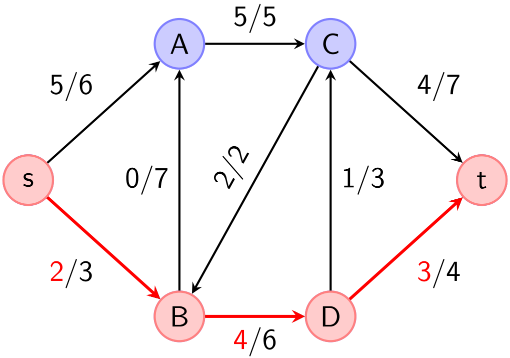
Maximum-Flow Problem

Problem

Given a flow network G with source s, sink t, and a capacity function c.
Find a flow f in G of maximum value.

Applications:

- water flowing through network of pipes/rivers
- current in electrical network
- traffic: cars in road network
- messages in telecommunication network



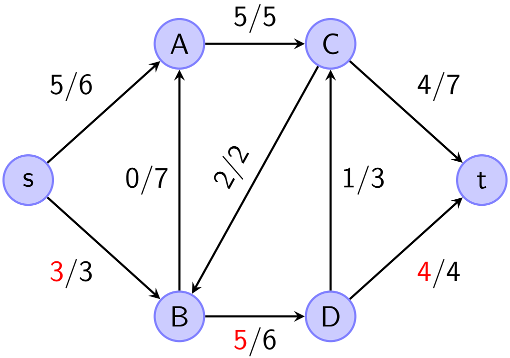
Maximum-Flow Problem

Problem

Given a flow network G with source s, sink t, and a capacity function c. Find a flow f in G of maximum value.

Applications:

- water flowing through network of pipes/rivers
- current in electrical network
- traffic: cars in road network
- messages in telecommunication network



Ford-Fulkerson Method

- method to solve the maximum-flow problem, i.e. find a maximum flow

Ford-Fulkerson Method

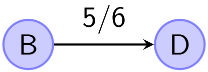
- method to solve the maximum-flow problem, i.e. find a maximum flow

Ford-Fulkerson Method

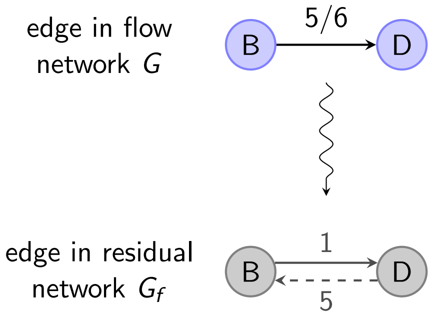
initialize flow f with 0
while there exists an augmenting path p in the residual network G_f **do**
  augment flow f along p
end while
return f

Residual Networks

edge in flow network G

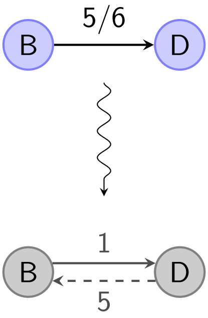


Residual Networks



Residual Networks

edge in flow network G



edge in residual network G_f

Definition (Residual Network)

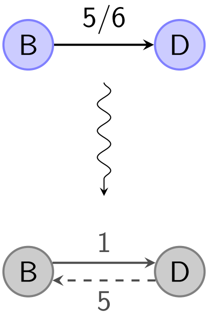
the residual capacity c_f is given by

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{otherwise} \end{cases}$$

Residual Networks

edge in flow network G

edge in residual network G_f



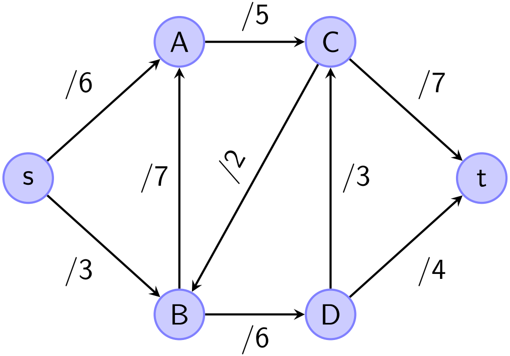
Definition (Residual Network)

A residual network of G induced by f is $G_f = (V, E_f)$ with

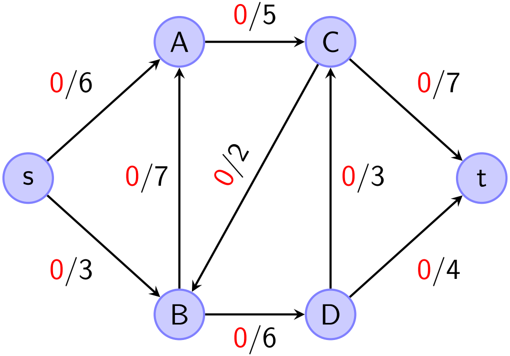
$$E_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\},$$

where the residual capacity c_f is given by

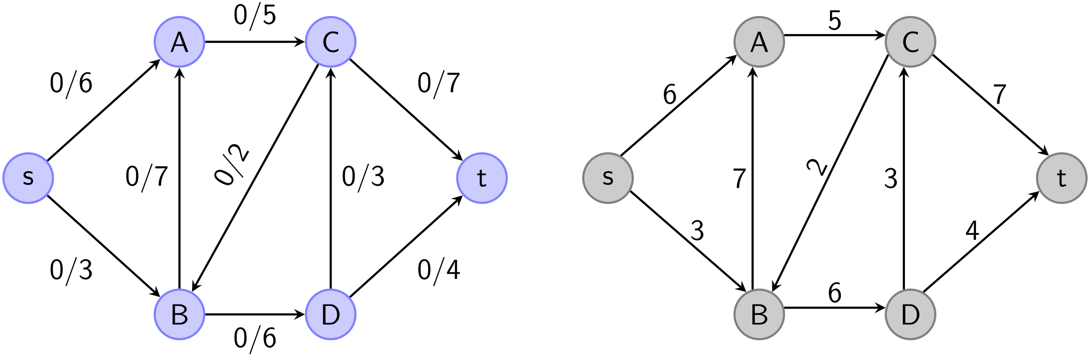
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E \\ f(v,u) & \text{if } (v,u) \in E \\ 0 & \text{otherwise} \end{cases}$$


/5
/6
/7
/2
/7
/3
/3
/4
/6
A
C
s
t
B
D

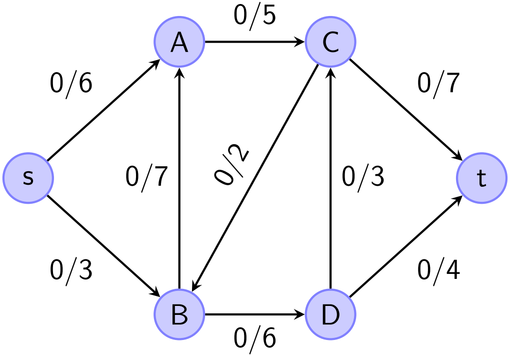
Demonstration of Ford-Fulkerson Method

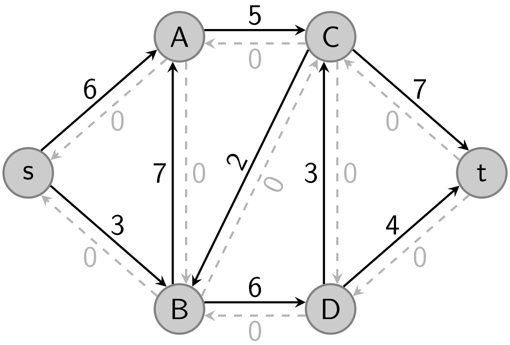


Demonstration of Ford-Fulkerson Method

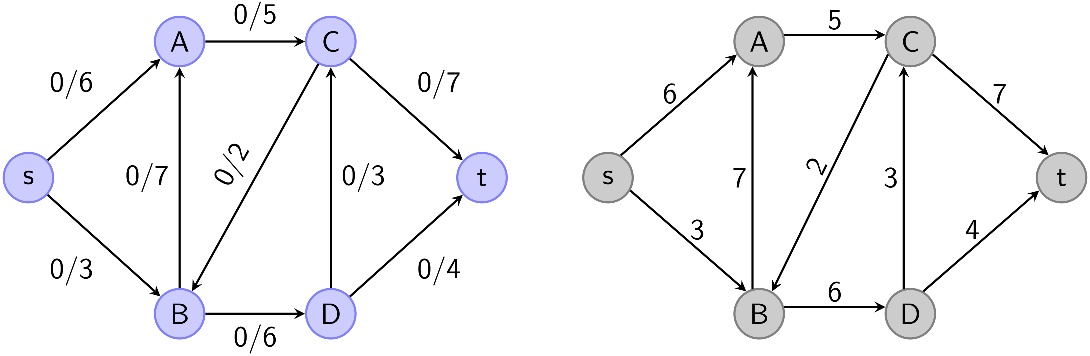


Demonstration of Ford-Fulkerson Method

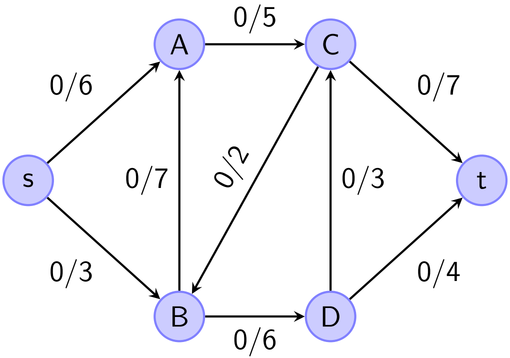


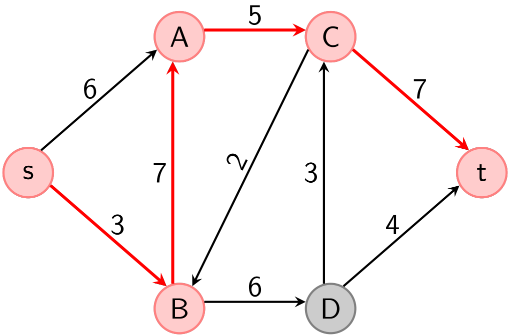


Demonstration of Ford-Fulkerson Method

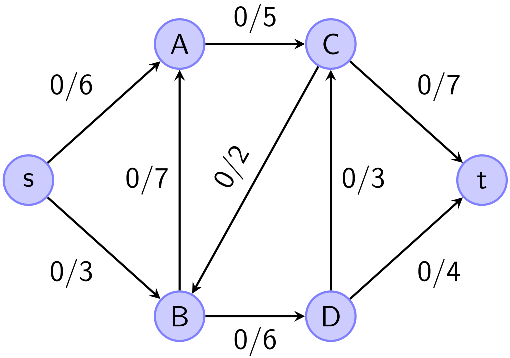


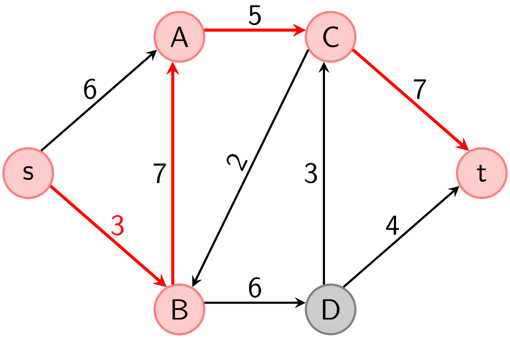
Demonstration of Ford-Fulkerson Method

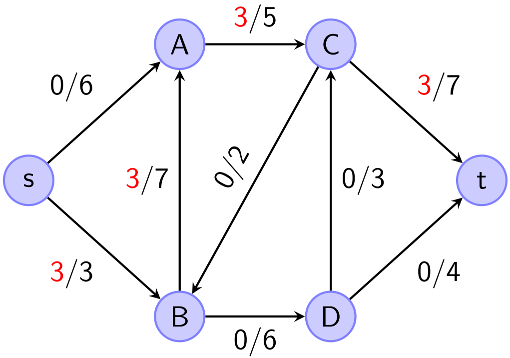


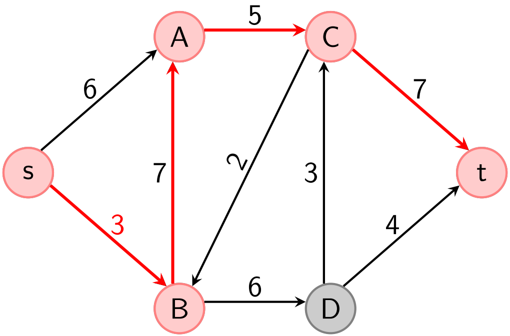


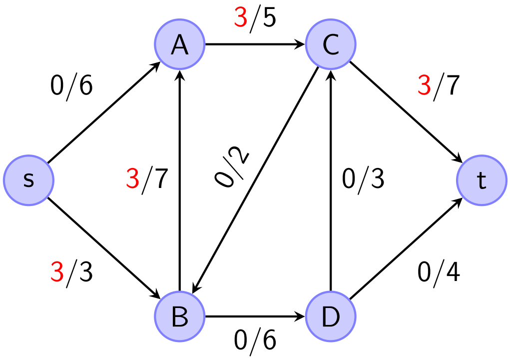
Demonstration of Ford-Fulkerson Method

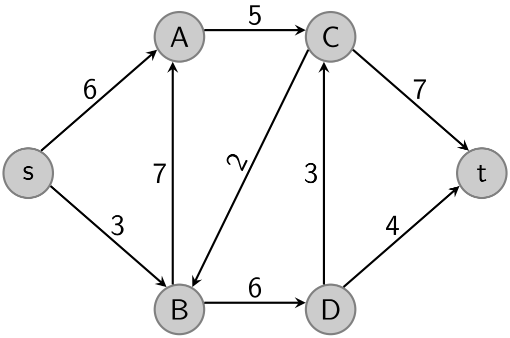




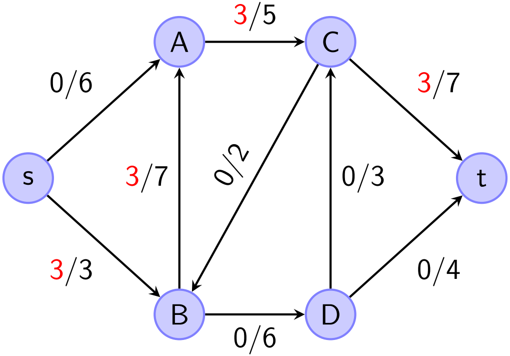

3/5
0/6
3/7
0/2
0/3
3/7
3/3
0/4
0/6
s
A
B
C
D
t

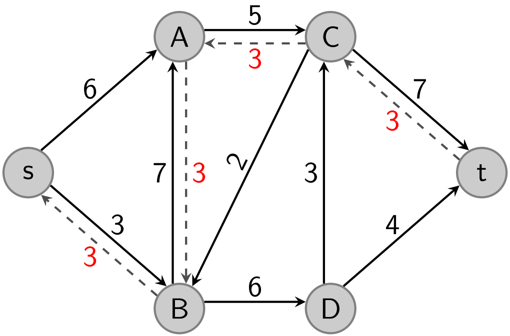

5
6
7
7
2
3
3
4
6
s
A
B
C
D
t


3/5
A
C
0/6
3/7
s
3/7
0/2
0/3
t
3/3
0/4
B
D
0/6

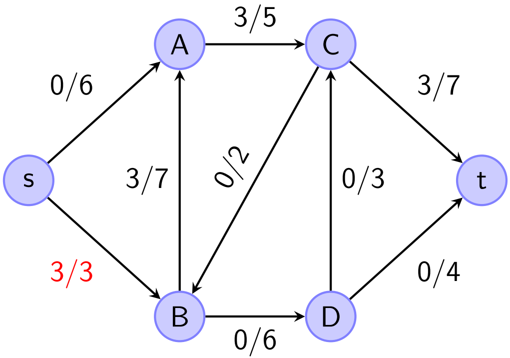

A
5
C
6
7
s
7
2
3
t
3
4
B
6
D

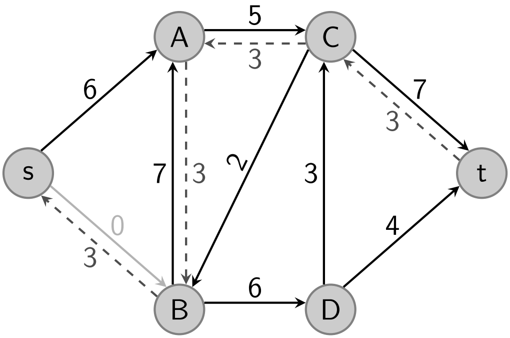
Demonstration of Ford-Fulkerson Method



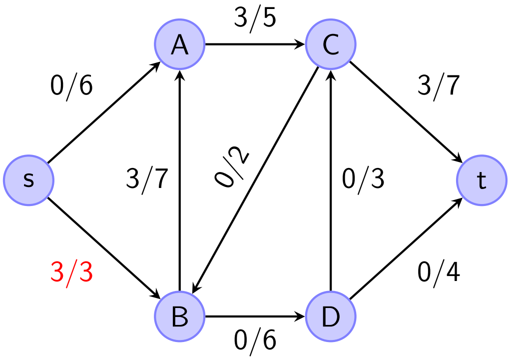


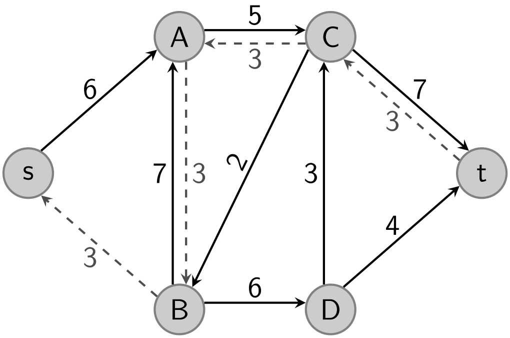
Demonstration of Ford-Fulkerson Method



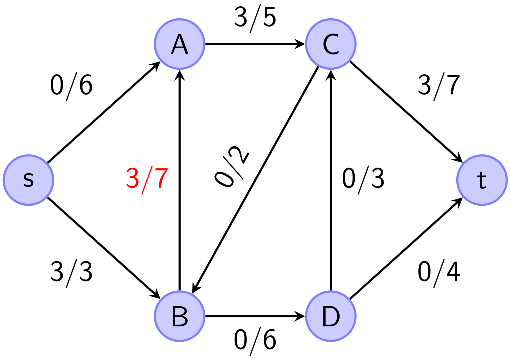


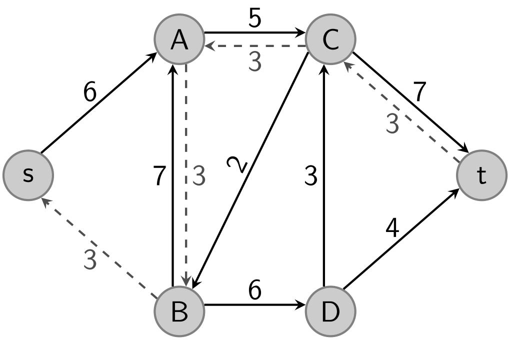
Demonstration of Ford-Fulkerson Method



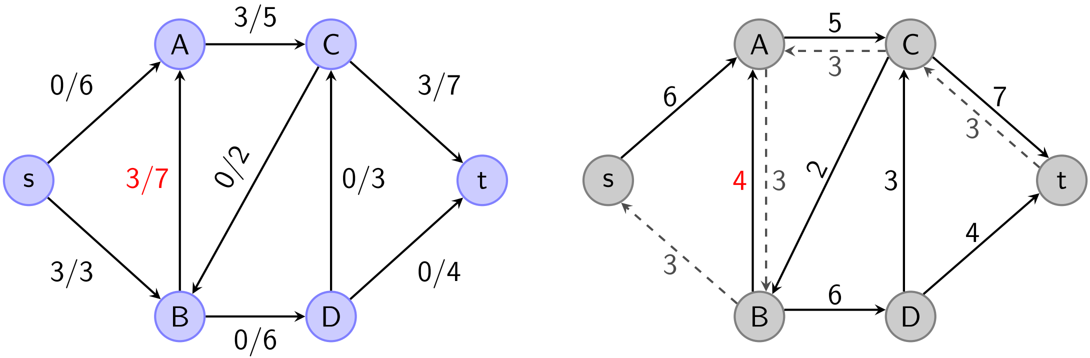


Demonstration of Ford-Fulkerson Method

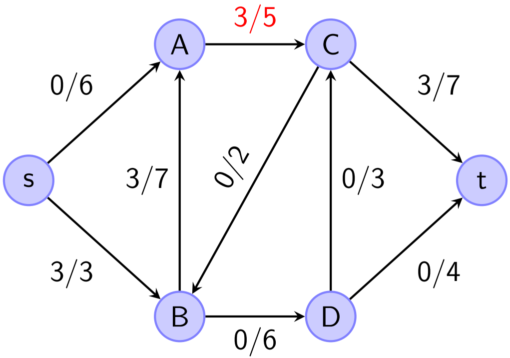


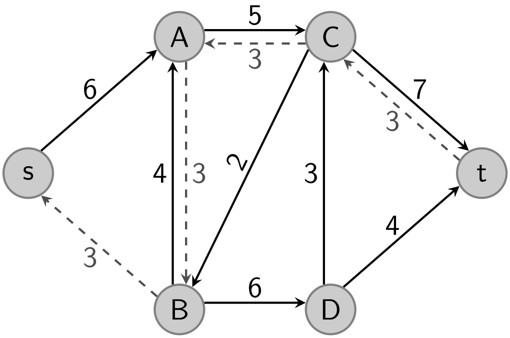


Demonstration of Ford-Fulkerson Method

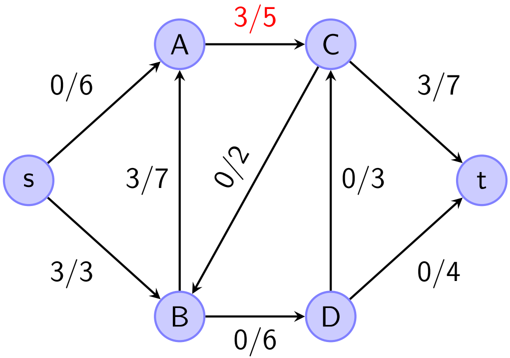


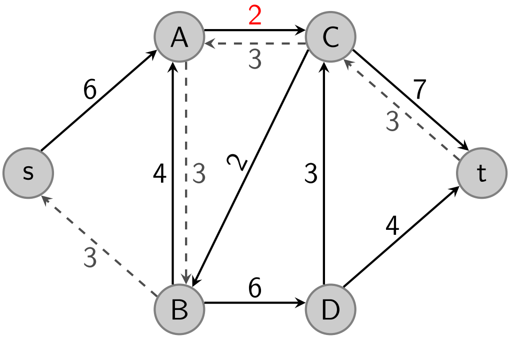
Demonstration of Ford-Fulkerson Method



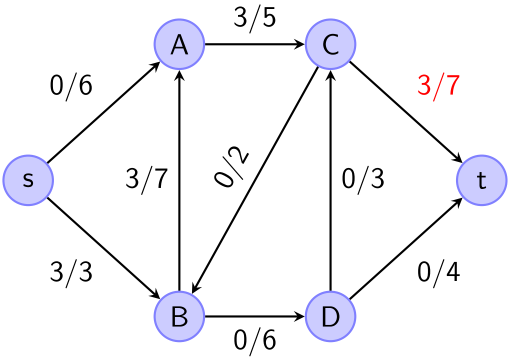


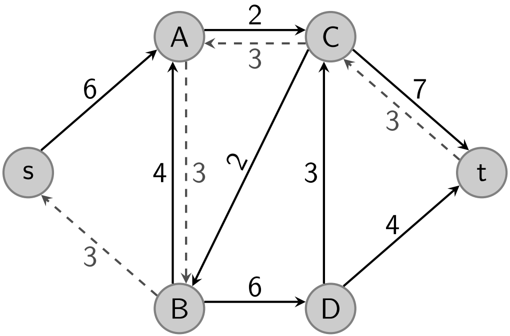
Demonstration of Ford-Fulkerson Method



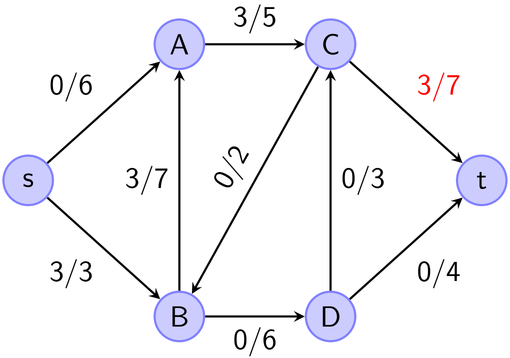


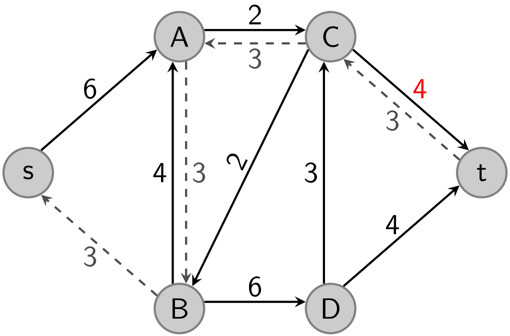
Demonstration of Ford-Fulkerson Method



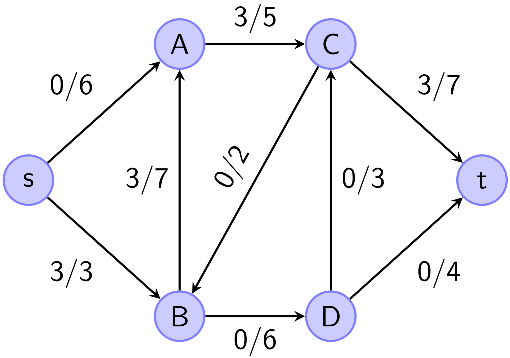


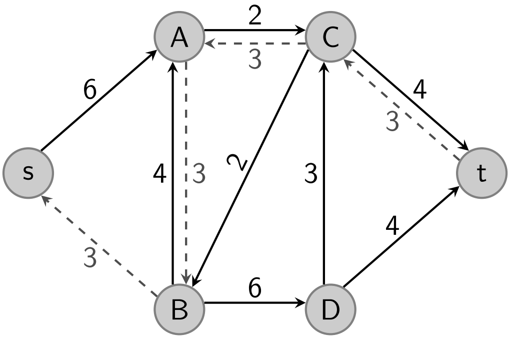
Demonstration of Ford-Fulkerson Method



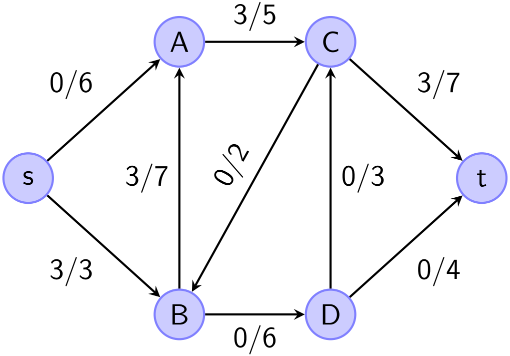


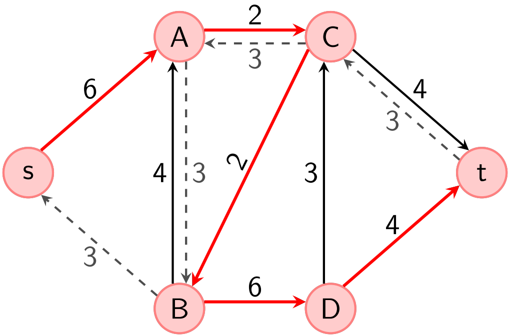
Demonstration of Ford-Fulkerson Method



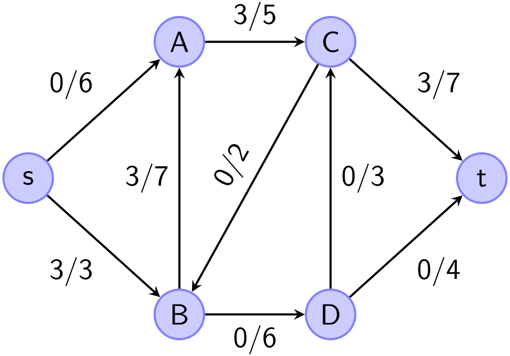


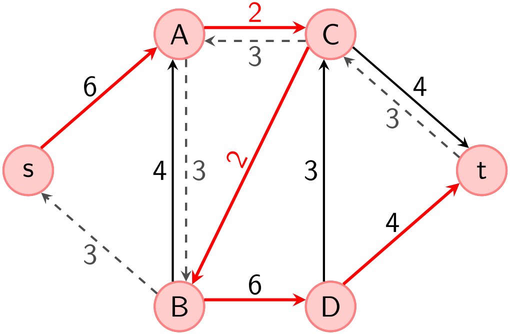
Demonstration of Ford-Fulkerson Method



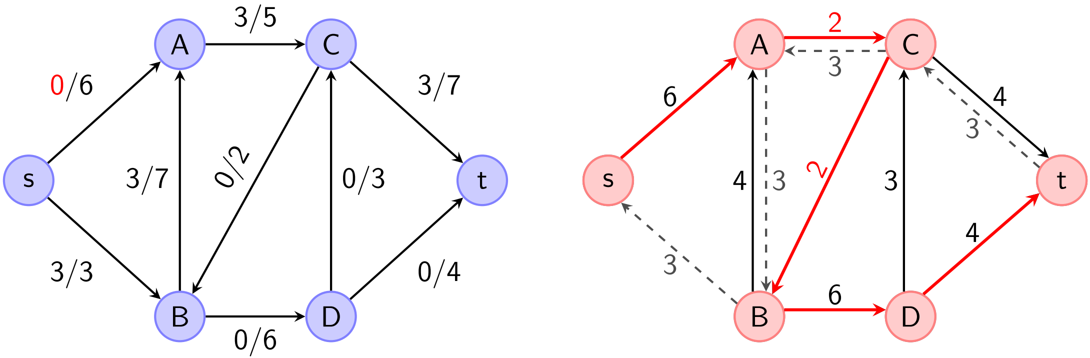


Demonstration of Ford-Fulkerson Method

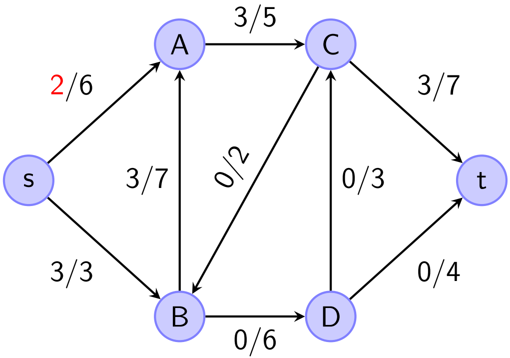


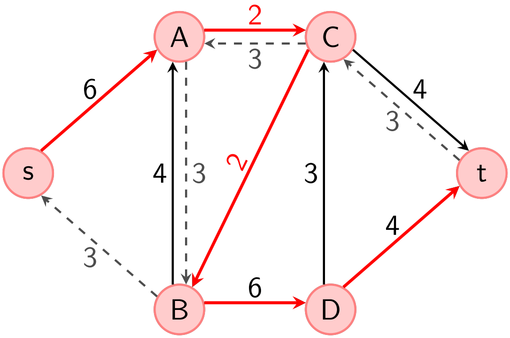


Demonstration of Ford-Fulkerson Method

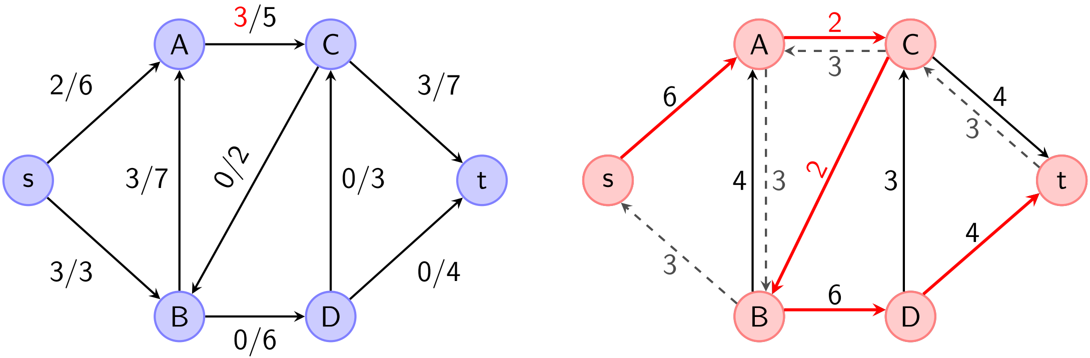


Demonstration of Ford-Fulkerson Method

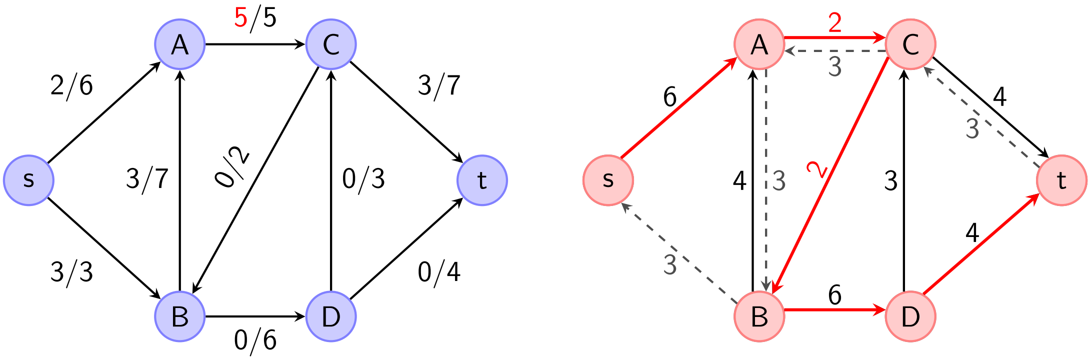




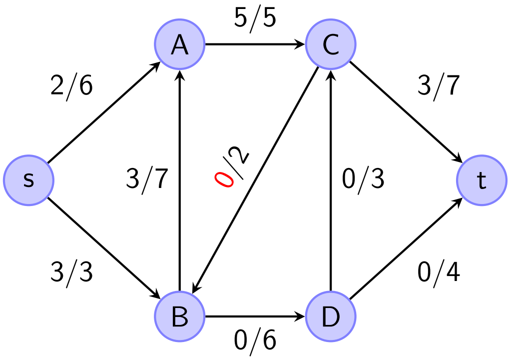
Demonstration of Ford-Fulkerson Method

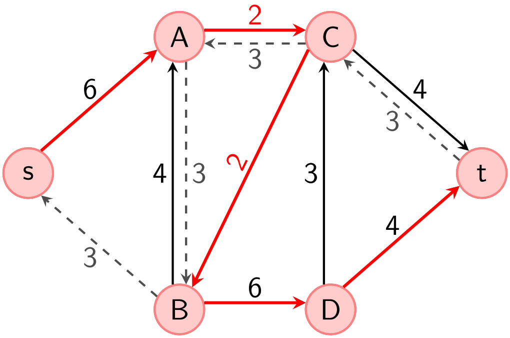


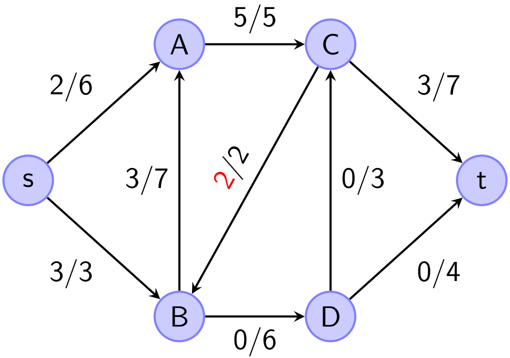
Demonstration of Ford-Fulkerson Method

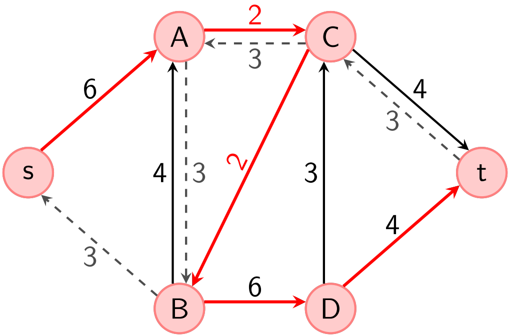


Demonstration of Ford-Fulkerson Method

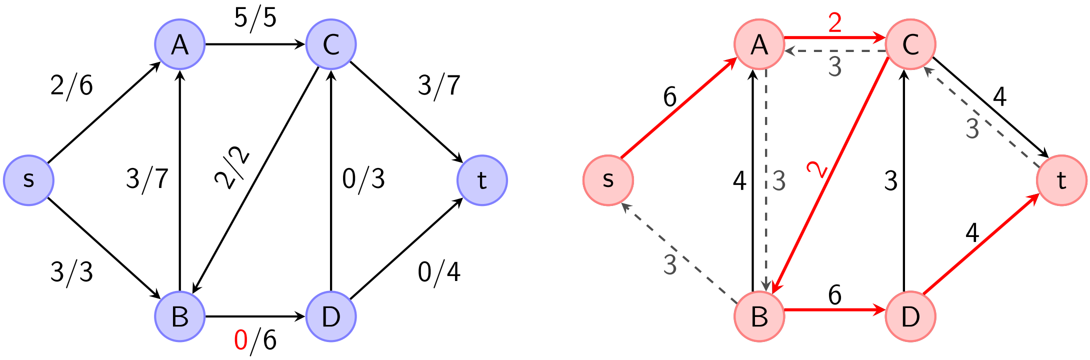




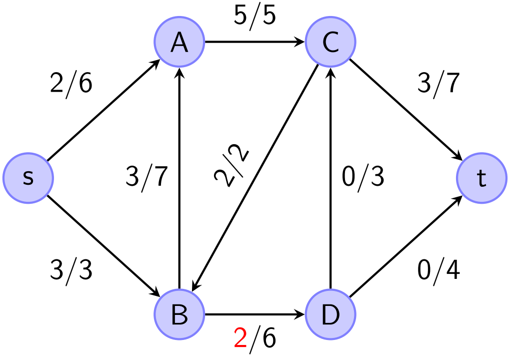

5/5
2/6
3/7
2/2
0/3
3/7
3/3
0/6
0/4
A
C
s
t
B
D

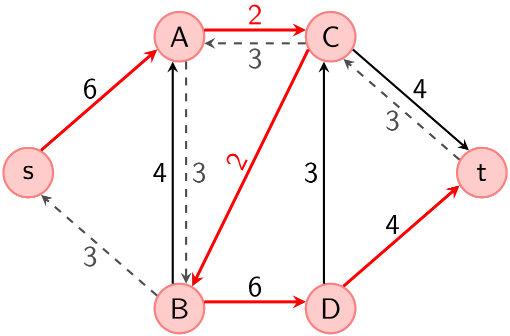

2
3
6
4
3
2
3
4
3
3
6
4
A
C
s
t
B
D

Demonstration of Ford-Fulkerson Method

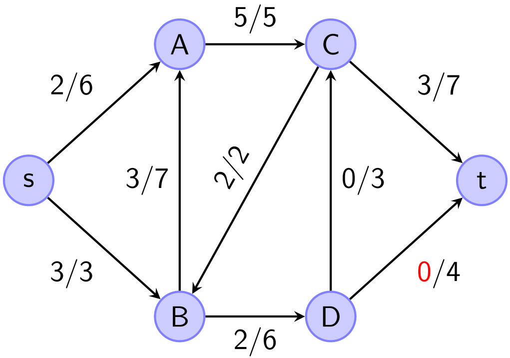


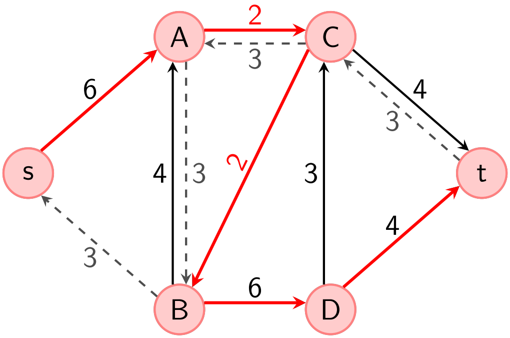
Demonstration of Ford-Fulkerson Method



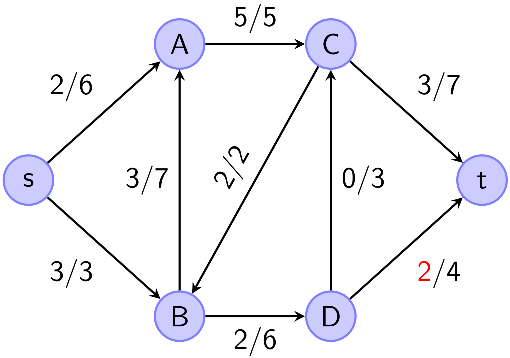


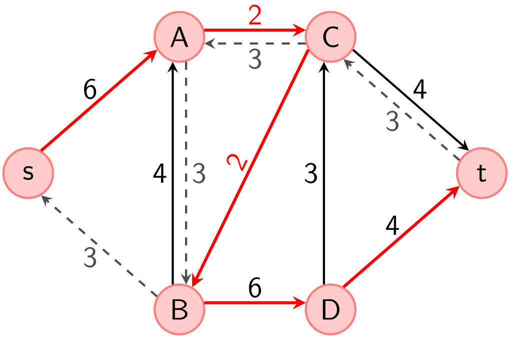
Demonstration of Ford-Fulkerson Method



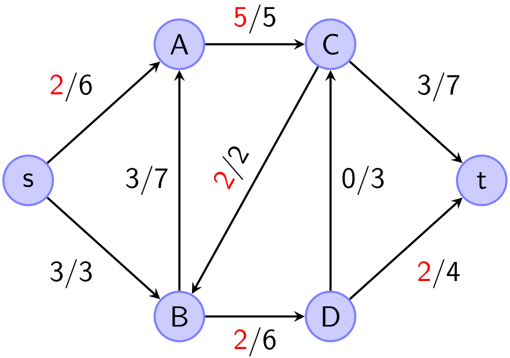


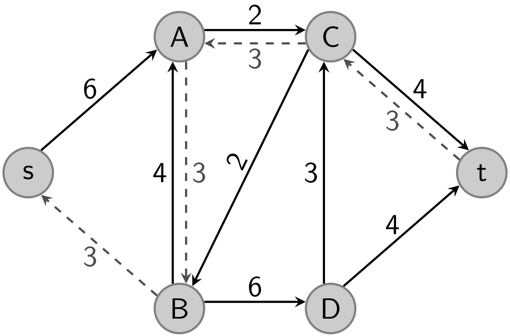
Demonstration of Ford-Fulkerson Method



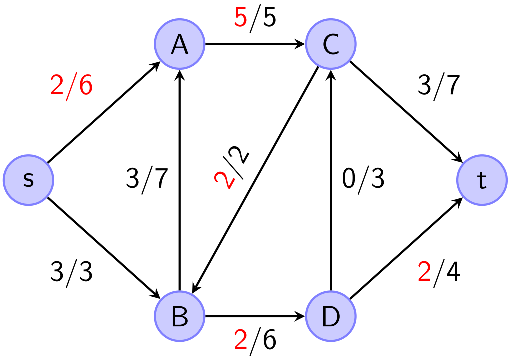


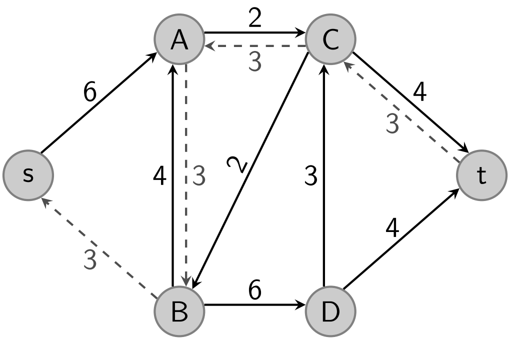
Demonstration of Ford-Fulkerson Method



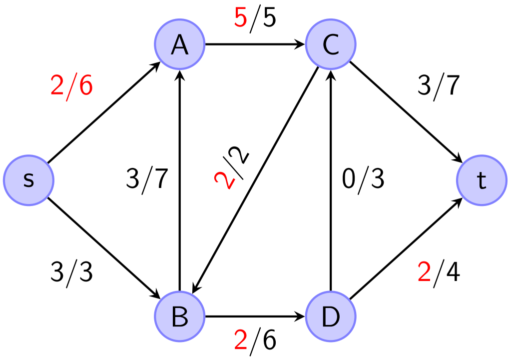


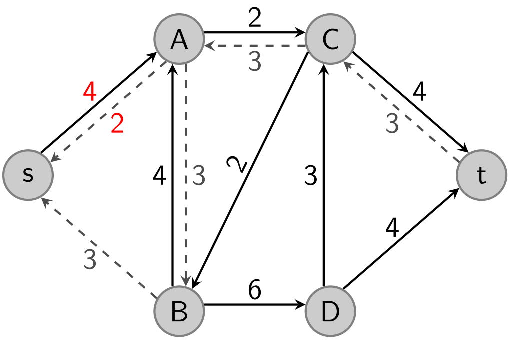
Demonstration of Ford-Fulkerson Method



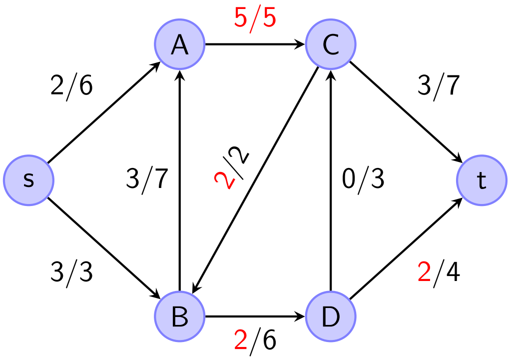


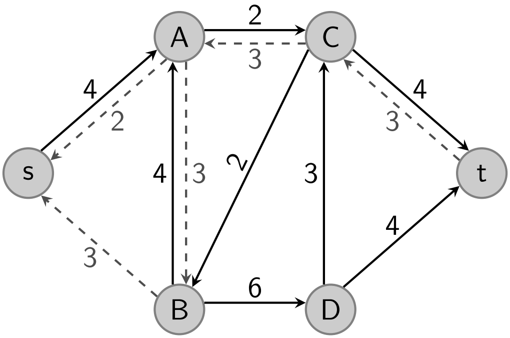
Demonstration of Ford-Fulkerson Method



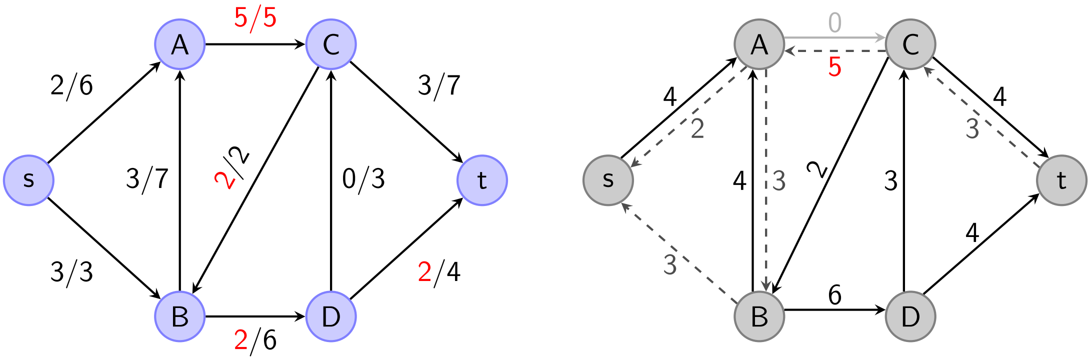


Demonstration of Ford-Fulkerson Method

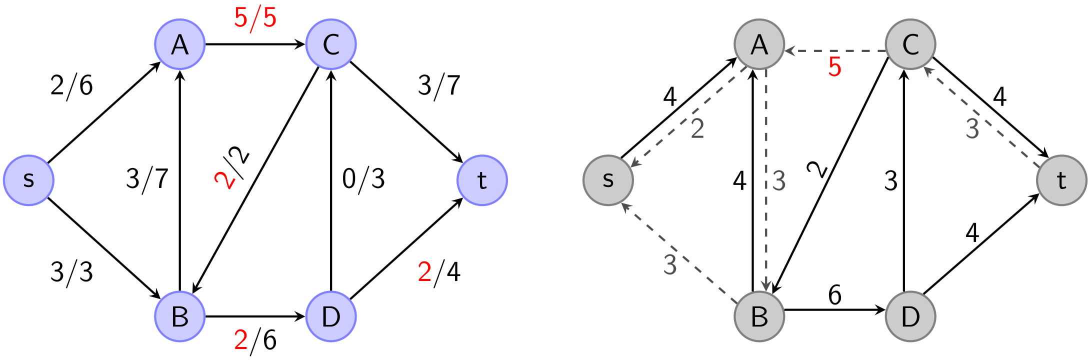




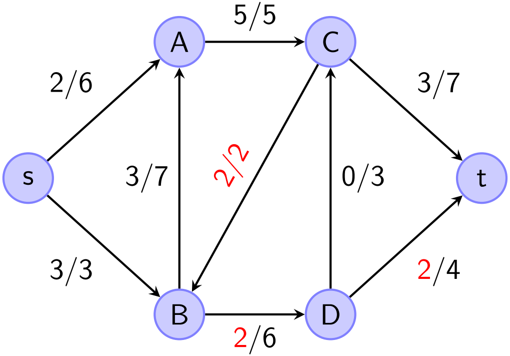
Demonstration of Ford-Fulkerson Method

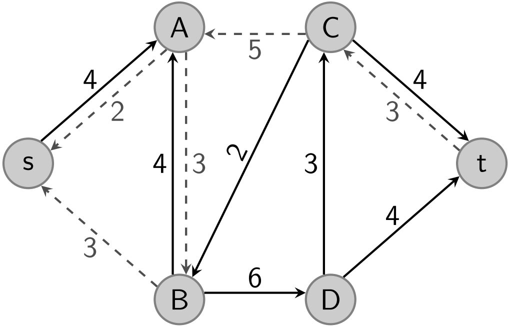


Demonstration of Ford-Fulkerson Method

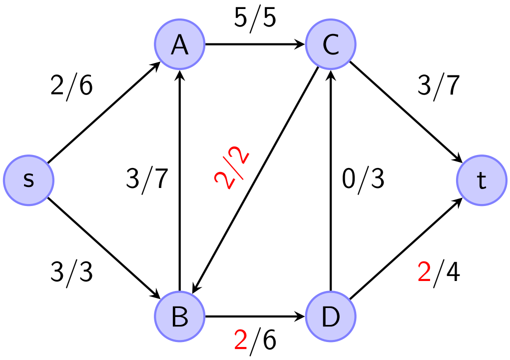


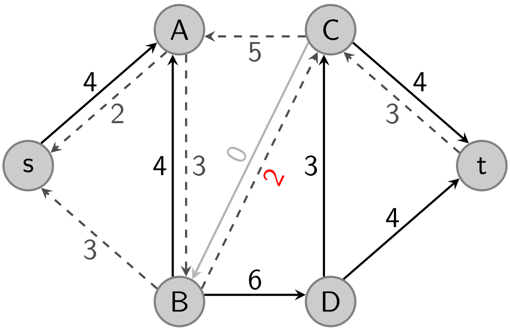
Demonstration of Ford-Fulkerson Method



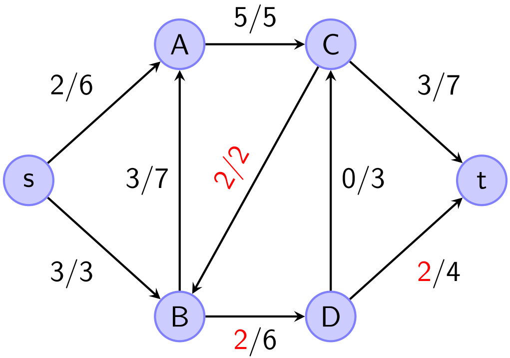


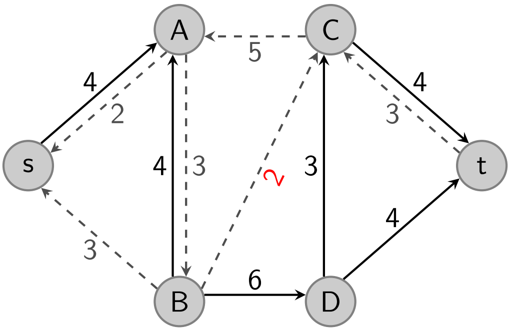
Demonstration of Ford-Fulkerson Method



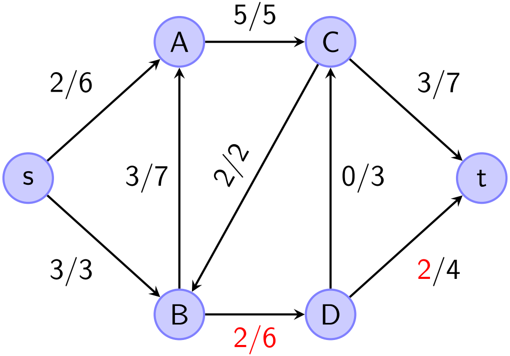


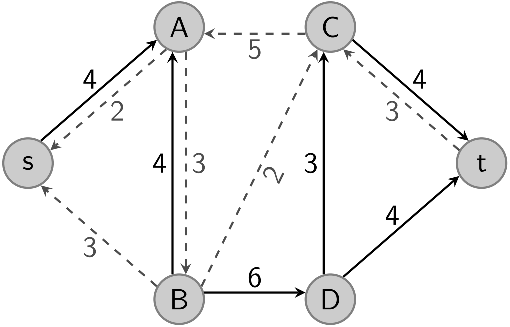
Demonstration of Ford-Fulkerson Method



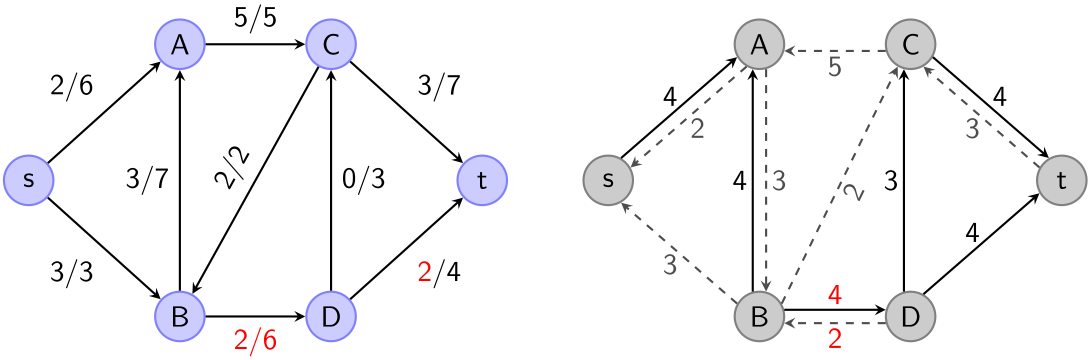


Demonstration of Ford-Fulkerson Method

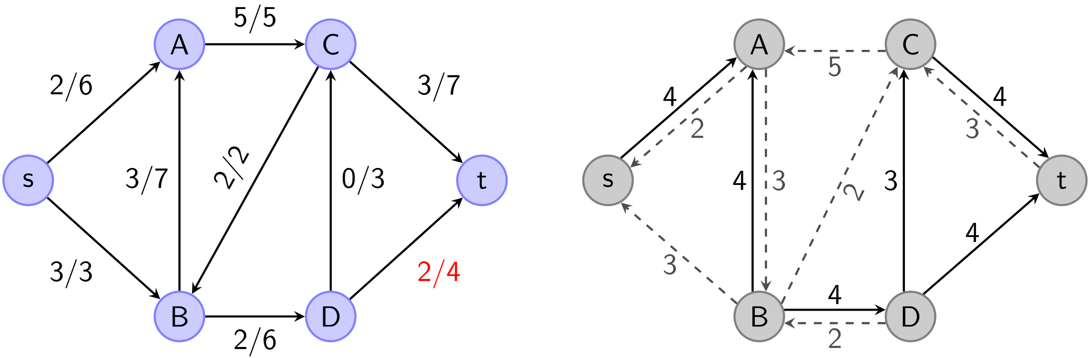




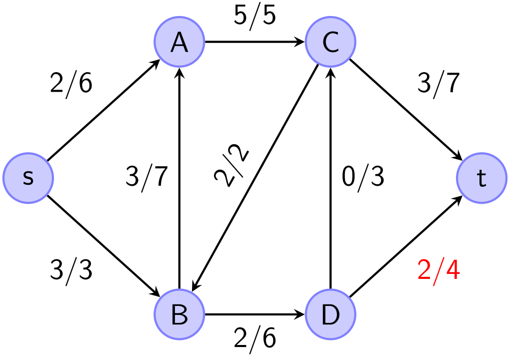
Demonstration of Ford-Fulkerson Method

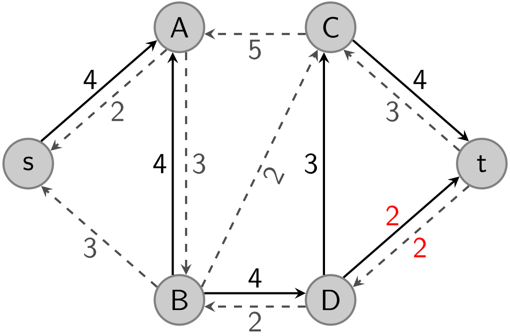


Demonstration of Ford-Fulkerson Method

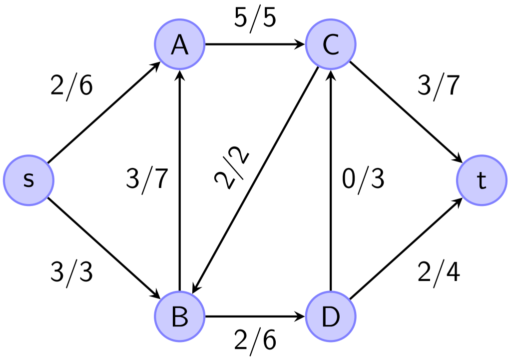


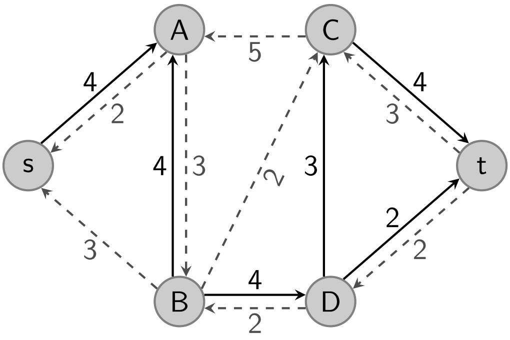
Demonstration of Ford-Fulkerson Method



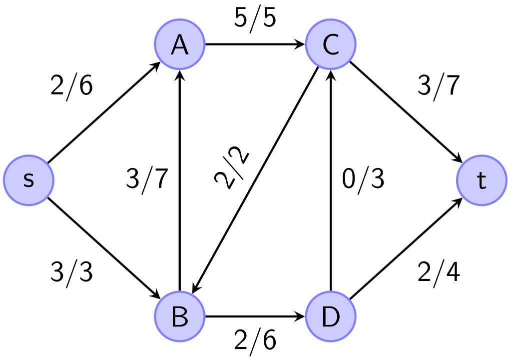


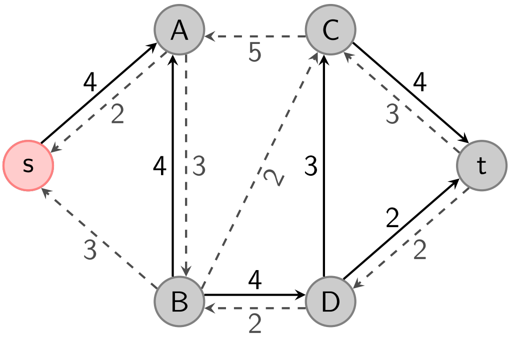
Demonstration of Ford-Fulkerson Method



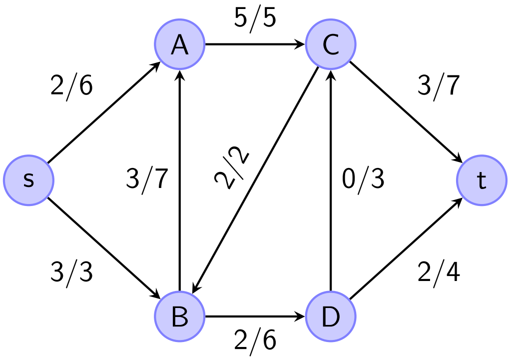


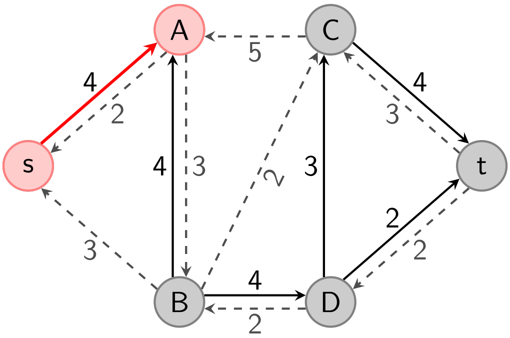
Demonstration of Ford-Fulkerson Method

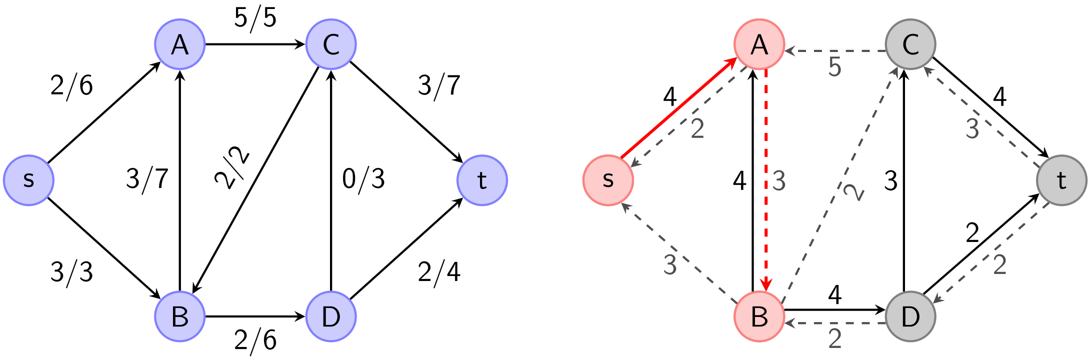




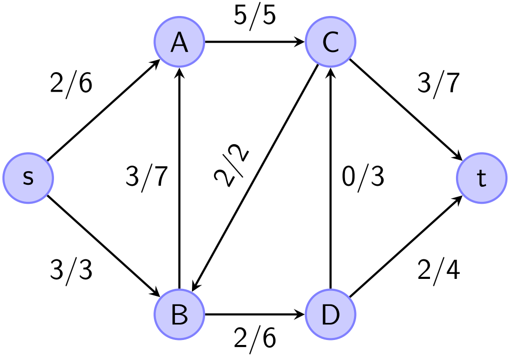
Demonstration of Ford-Fulkerson Method

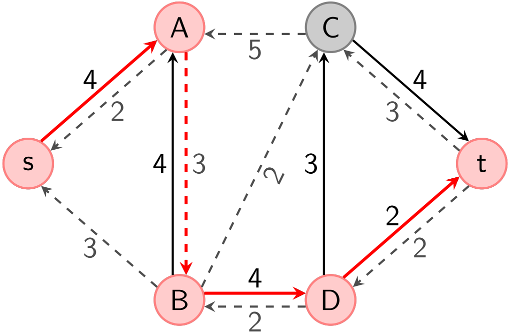




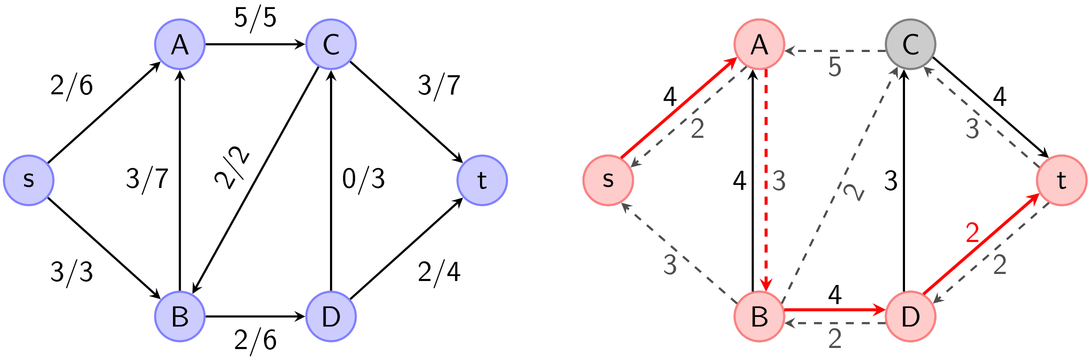

5/5
2/6
3/7
2/2
0/3
3/7
3/3
2/4
2/6
A
C
s
t
B
D
5
4
2
4
3
2
3
4
3
3
2
2
4
2
A
C
s
t
B
D

Demonstration of Ford-Fulkerson Method

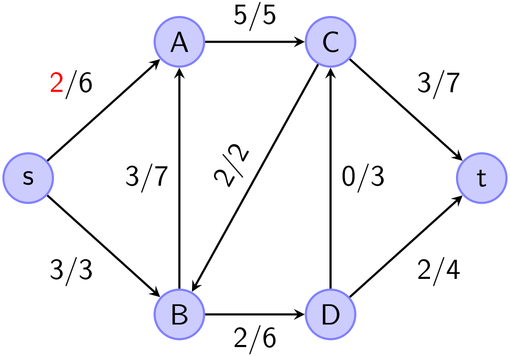


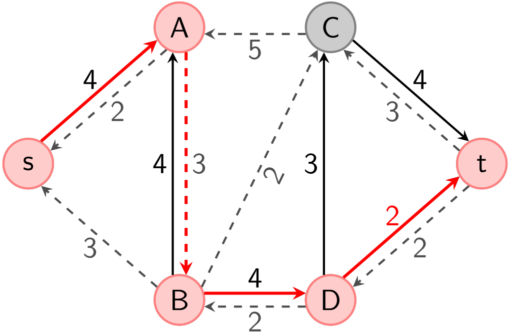


Demonstration of Ford-Fulkerson Method

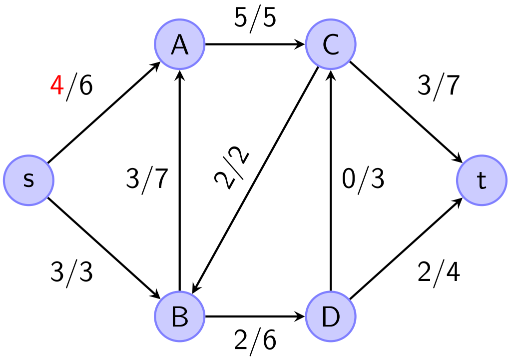


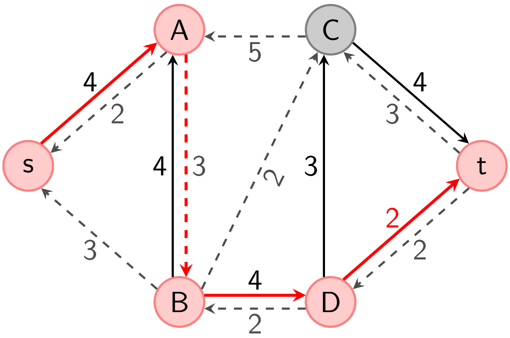
Demonstration of Ford-Fulkerson Method



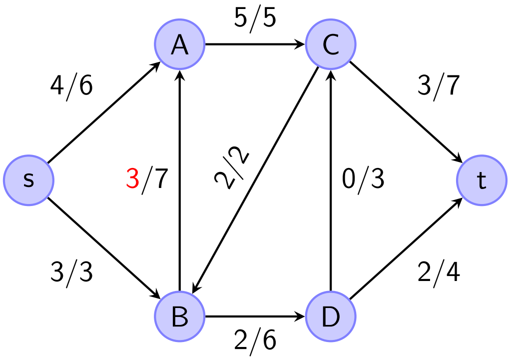


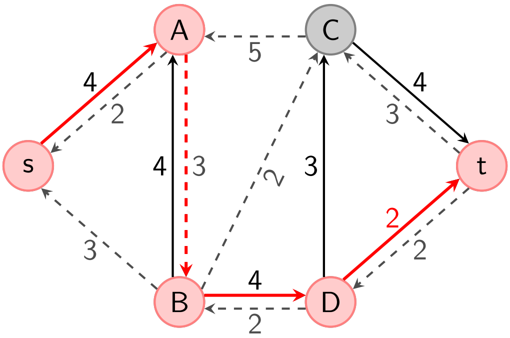
Demonstration of Ford-Fulkerson Method



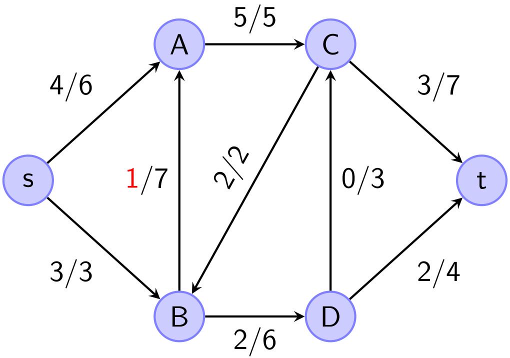


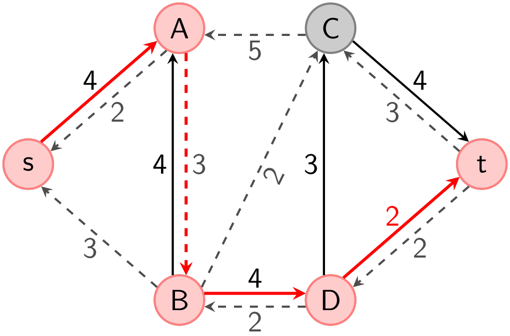
Demonstration of Ford-Fulkerson Method



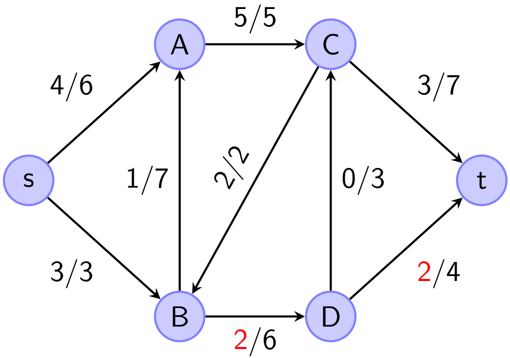


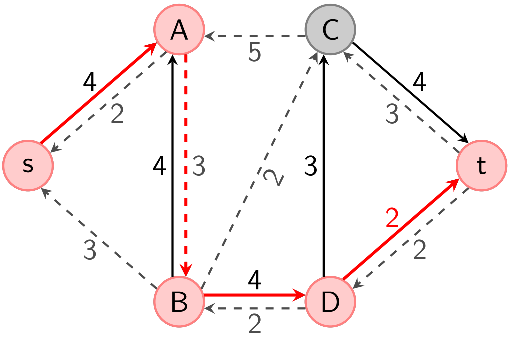
Demonstration of Ford-Fulkerson Method

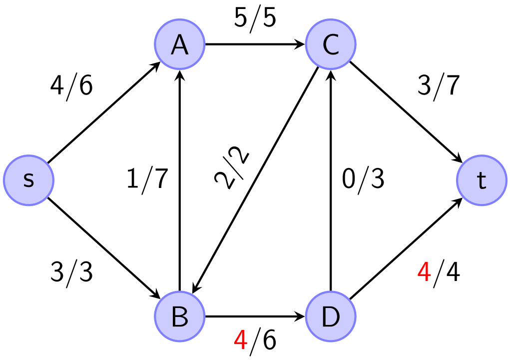


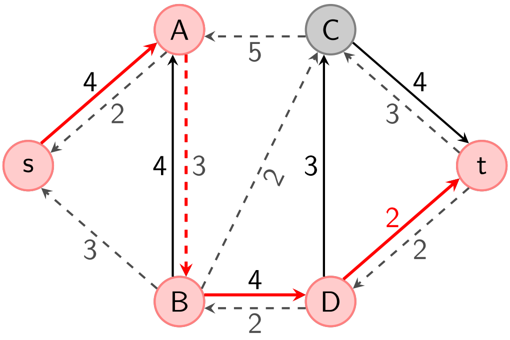


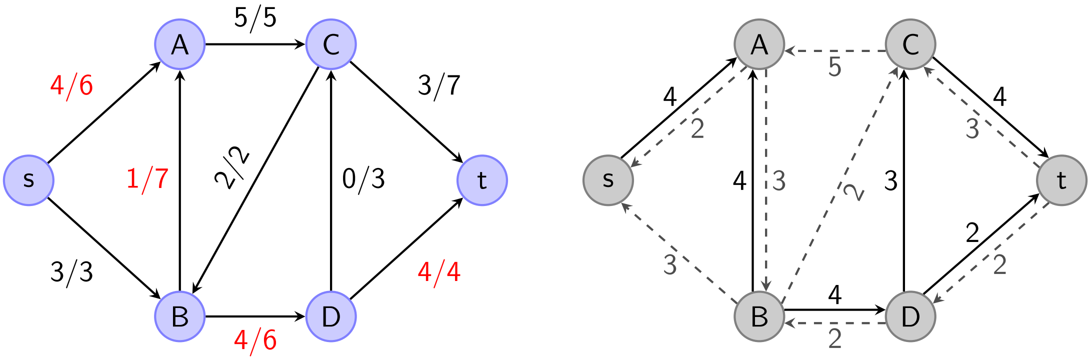
Demonstration of Ford-Fulkerson Method



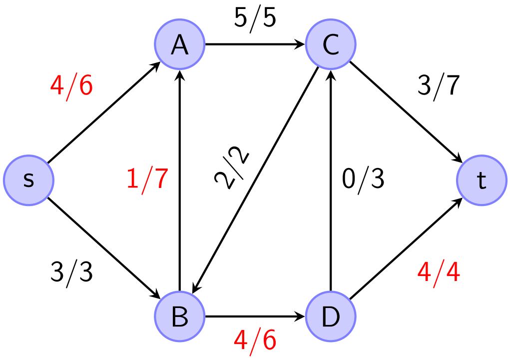


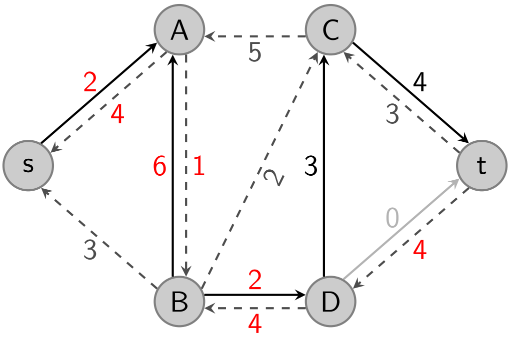

5/5
A
C
4/6
3/7
s
1/7
2/2
0/3
t
3/3
4/4
B
D
4/6


A
C
5
4
4
2
3
s
4
3
2
3
t
2
3
2
4
B
D
2

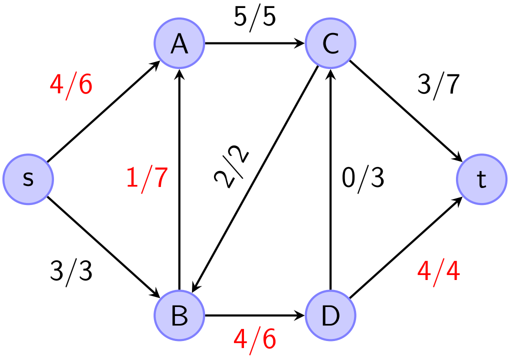

s
A
B
C
D
t
4/6
3/3
5/5
1/7
2/2
0/3
3/7
4/6
4/4
s
A
B
C
D
t
4
2
3
5
4
3
2
3
4
3
4
2
2
4
2

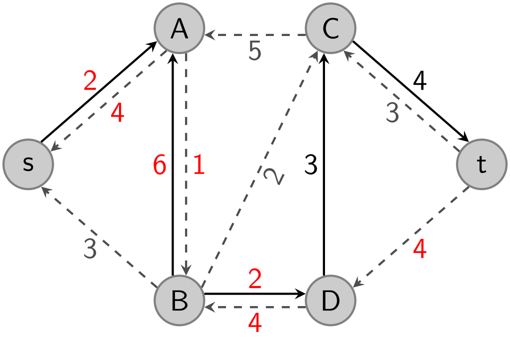
Demonstration of Ford-Fulkerson Method



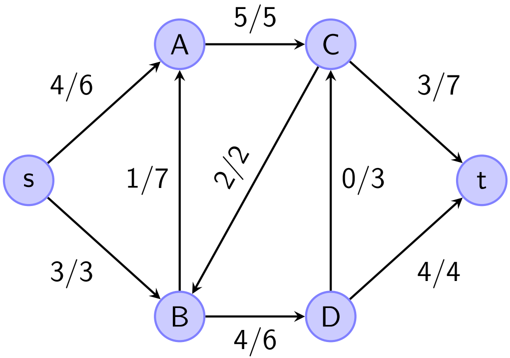


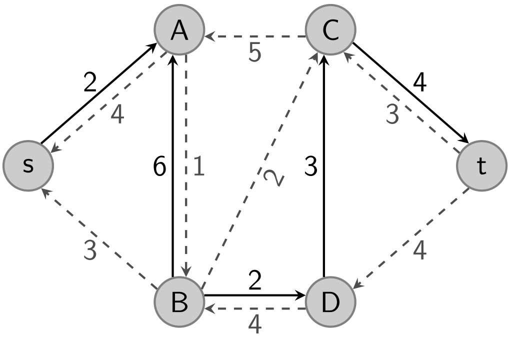
Demonstration of Ford-Fulkerson Method



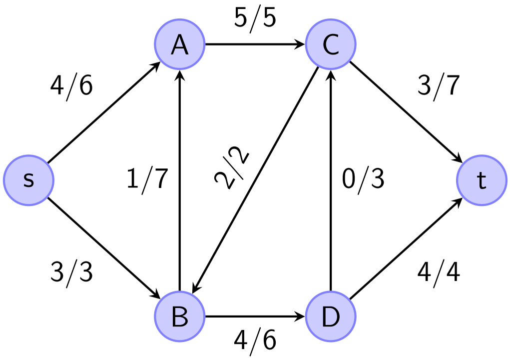


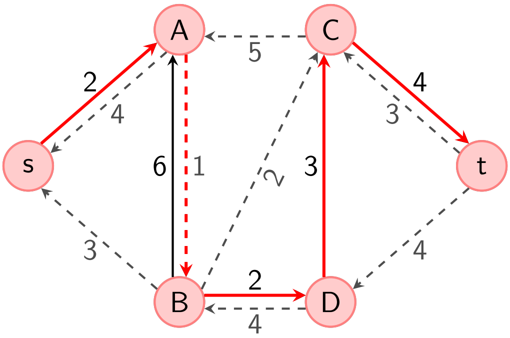
Demonstration of Ford-Fulkerson Method



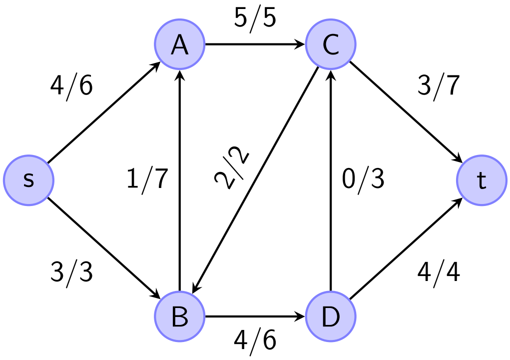


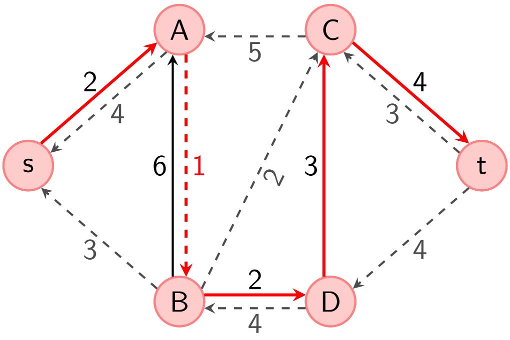
Demonstration of Ford-Fulkerson Method



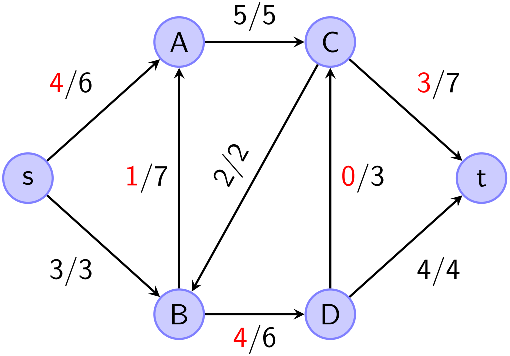


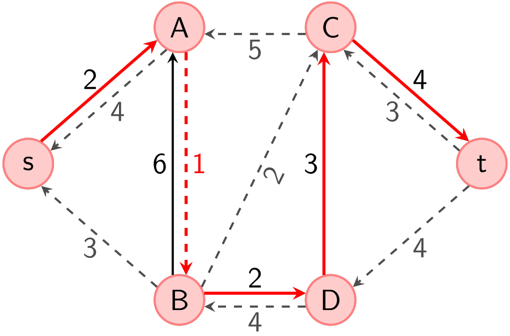
Demonstration of Ford-Fulkerson Method



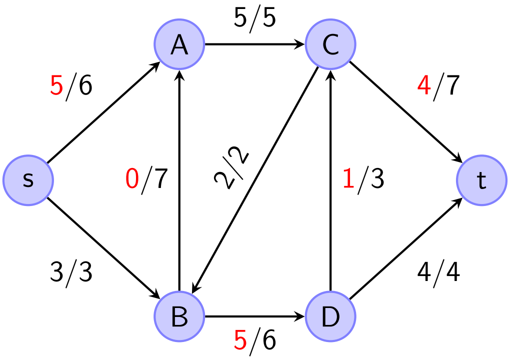


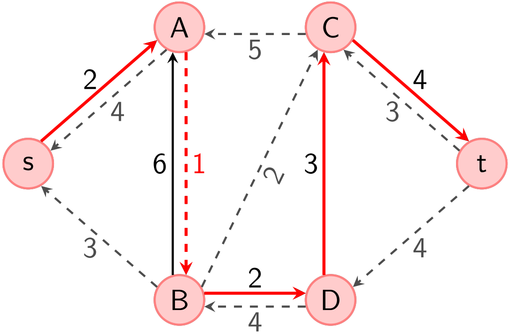
Demonstration of Ford-Fulkerson Method



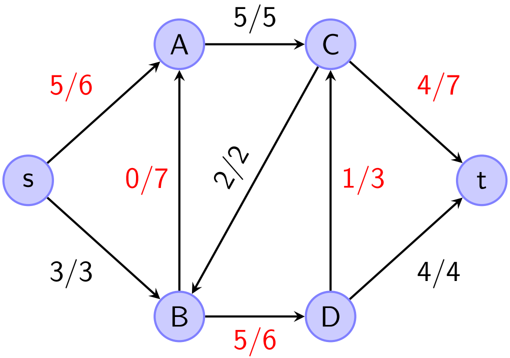


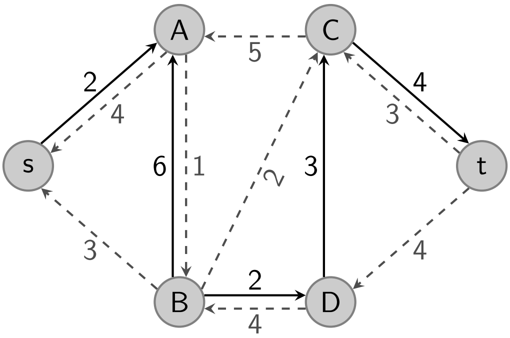
Demonstration of Ford-Fulkerson Method



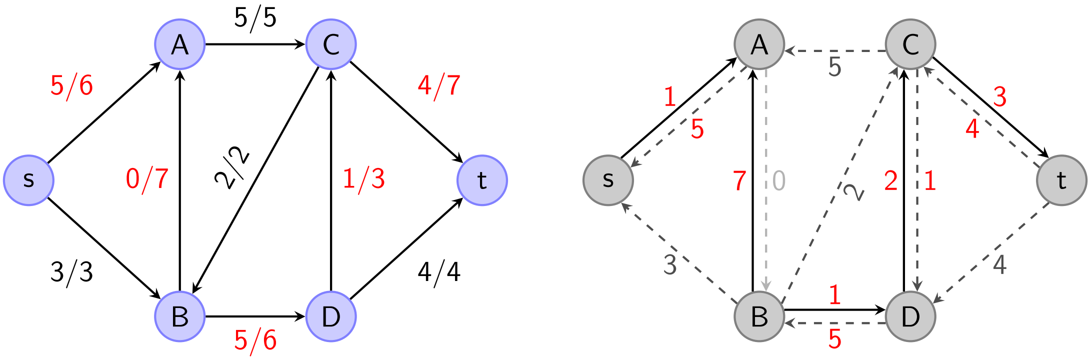


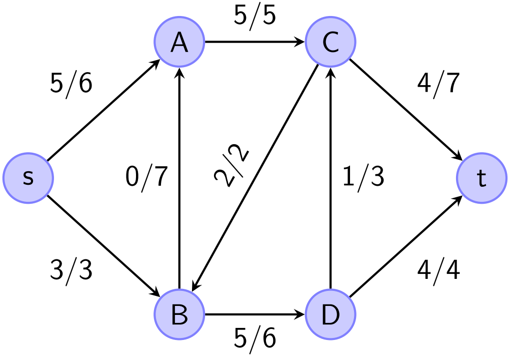
Demonstration of Ford-Fulkerson Method

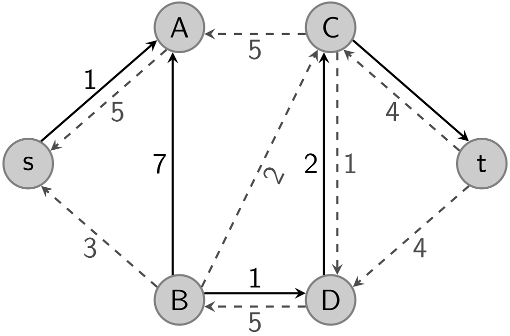




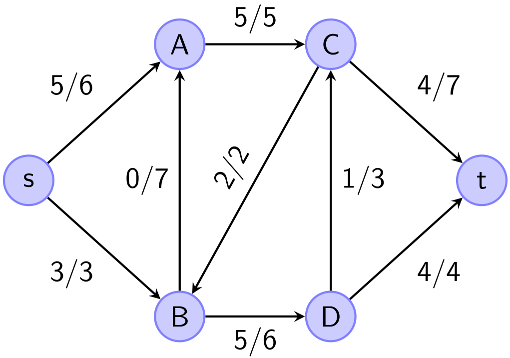
Demonstration of Ford-Fulkerson Method

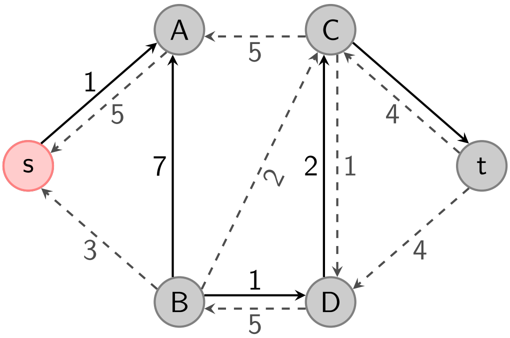


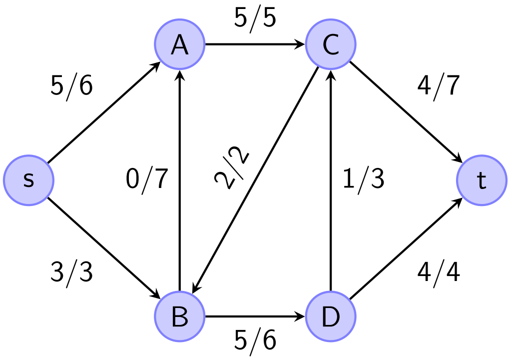

5/5
5/6
4/7
0/7
2/2
1/3
3/3
4/4
5/6
s
A
B
C
D
t

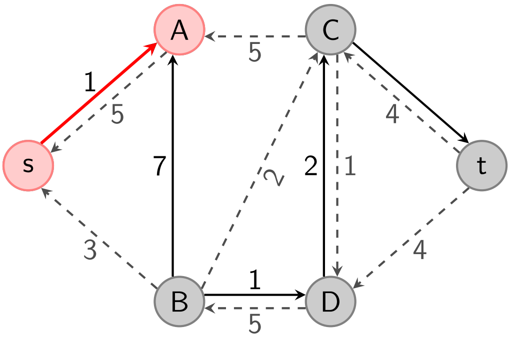

5
1
5
4
7
2
2
1
3
4
1
5
s
A
B
C
D
t

Demonstration of Ford-Fulkerson Method

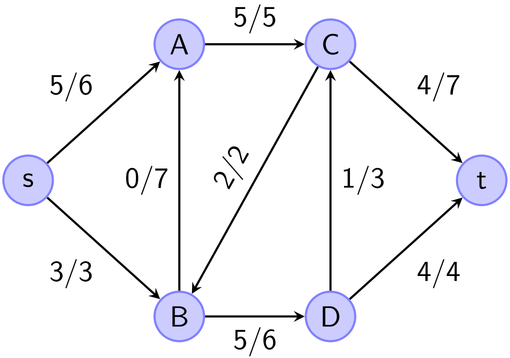




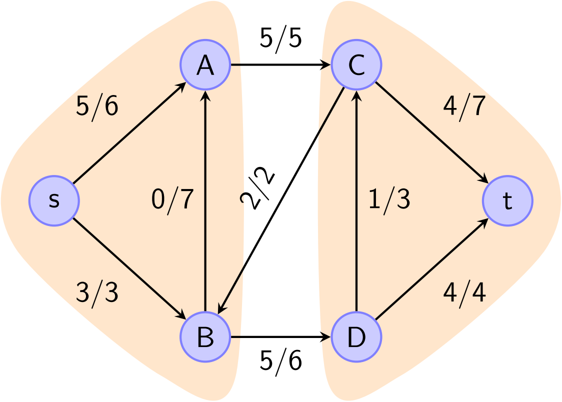

A
C
s
t
B
D
5/5
5/6
4/7
0/7
2/2
1/3
3/3
4/4
5/6


A
C
s
t
B
D
5
1
5
4
7
2
2
1
3
4
1
5

value of flow:
$8 = 5 + 3$



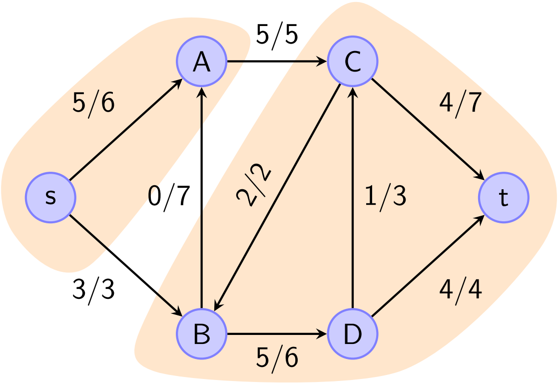
value of flow:
$8 = 5 + 3$



net flow across cut: $8 = 5 - 2 + 5$

capacity of cut: $11 = 5 + 6$

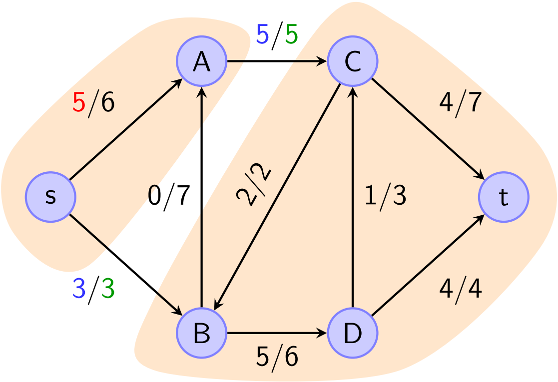
value of flow:
$8 = 5 + 3$



net flow across cut:

capacity of cut:

value of flow:
$8 = 5 + 3$



net flow across cut: $8 = 5 + 3$

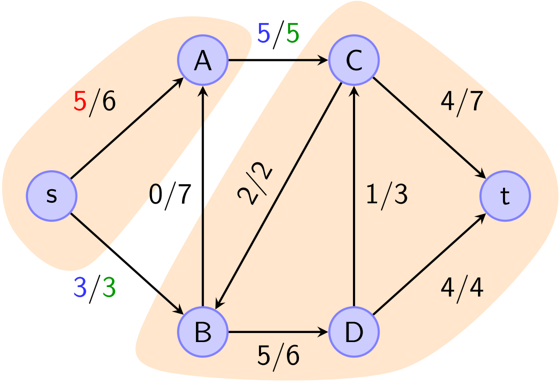
capacity of cut: $8 = 5 + 3$

Partial Correctness

Max-Flow Min-Cut Theorem

For a flow f in $G = (V, E)$ the following conditions are equivalent

1. f is a maximum flow
2. there is no augmenting path in G_f
3. the value of flow equals capacity of some cut of G



value of flow: $8 = 5 + 3$

net flow across cut: $8 = 5 + 3$

capacity of cut: $8 = 5 + 3$

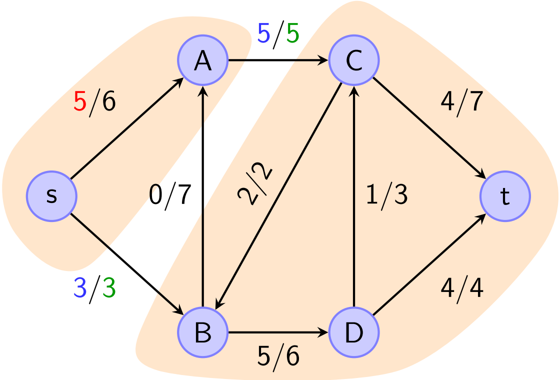
Partial Correctness and Termination

Max-Flow Min-Cut Theorem

For a flow f in $G = (V, E)$ the following conditions are equivalent

1. f is a maximum flow
2. there is no augmenting path in G_f
3. the value of flow equals capacity of some cut of G

Termination:
in each iteration the flow is increased by at least one unit



value of flow: $8 = 5 + 3$

net flow across cut: $8 = 5 + 3$

capacity of cut: $8 = 5 + 3$

Time Complexity

- method to find augmenting paths unspecified in Ford-Fulkerson Method

Time Complexity

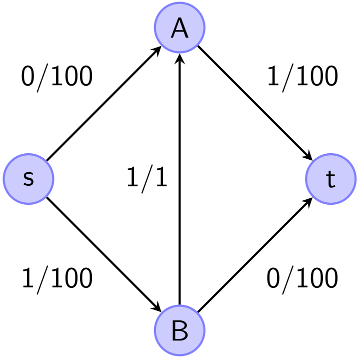
- method to find augmenting paths unspecified in Ford-Fulkerson Method
- use e.g. Depth-First Search (DFS) for finding augmenting paths (i.e. random choice), takes $O(|V| + 2|E|) = O(|E|)$ time

Time Complexity

- method to find augmenting paths unspecified in Ford-Fulkerson Method
- use e.g. Depth-First Search (DFS) for finding augmenting paths (i.e. random choice), takes $O(|V| + 2|E|) = O(|E|)$ time
- initializing and updating residual network takes $O(|E|)$ time as well

Time Complexity

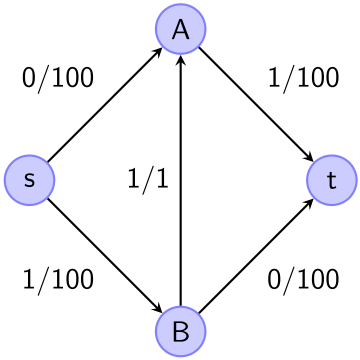
- method to find augmenting paths unspecified in Ford-Fulkerson Method
- use e.g. Depth-First Search (DFS) for finding augmenting paths (i.e. random choice), takes $O(|V| + 2|E|) = O(|E|)$ time
- initializing and updating residual network takes $O(|E|)$ time as well
- maximum number of iterations: value of flow $|f|$



Time Complexity

- method to find augmenting paths unspecified in Ford-Fulkerson Method
- use e.g. Depth-First Search (DFS) for finding augmenting paths (i.e. random choice), takes $O(|V| + 2|E|) = O(|E|)$ time
- initializing and updating residual network takes $O(|E|)$ time as well
- maximum number of iterations: value of flow $|f|$

$\Rightarrow$ $|f|$ iterations of $O(|E|)$ yields $O(|f| \cdot |E|)$ in total



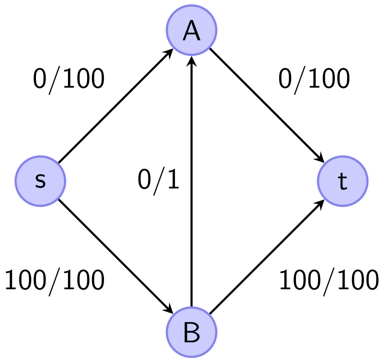
Optimizing Ford-Fulkerson Method

Idea: vary method of finding augmenting paths by

- doing Breadth-First Search (BFS)
- choosing paths of high bottle neck value first
- using so-called level graph to find shortest augmenting paths

Edmonds-Karp Algorithm

- use BFS to find augmenting paths
- shortest-path distance increases monotonically with flow augmentation
- maximum number of iterations: $O(VE)$
  - maximum number of augmenting paths of same length: $|E|$
  - maximum length of any path: $|V|$
- in total: $O(VE^2)$ which is independent of maximum flow value



Capacity Scaling

- heuristic to choose paths of high capacity first

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths

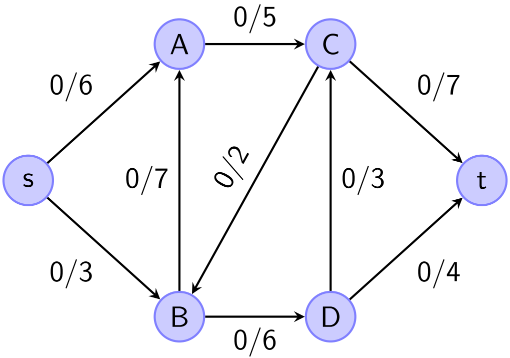
Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:

$$\Delta \leftarrow \Delta/2$$

Capacity Scaling

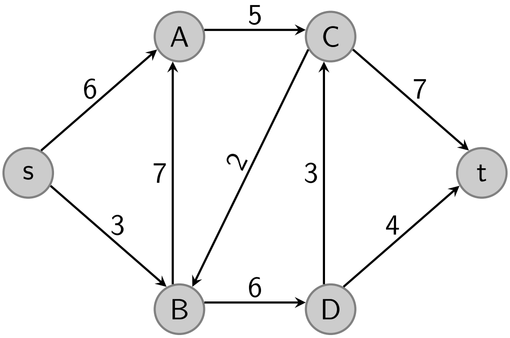
- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:
  $$\Delta \leftarrow \Delta/2$$



Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:

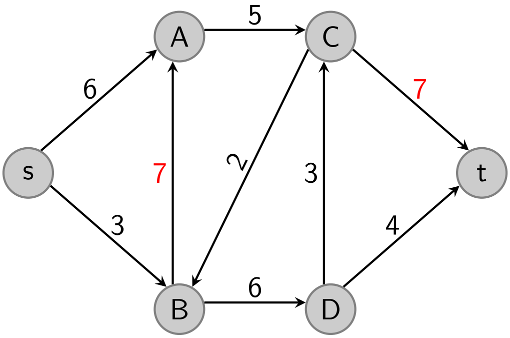
$$\Delta \leftarrow \Delta/2$$



Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:

$$\Delta \leftarrow \Delta/2$$

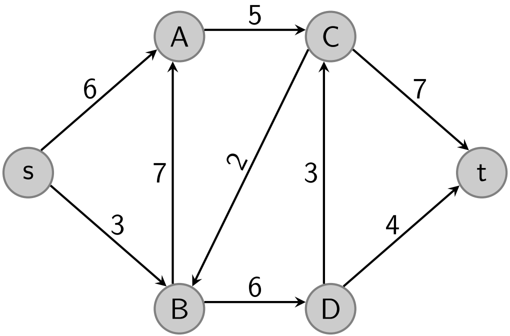


$C_{max} = 7$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:

$$\Delta \leftarrow \Delta/2$$

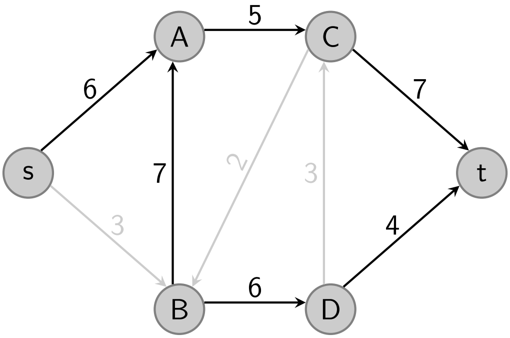


$\Delta = 4 = 2^2 \leq C_{max} < 2^3$

$C_{max} = 7$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:
  $$\Delta \leftarrow \Delta/2$$

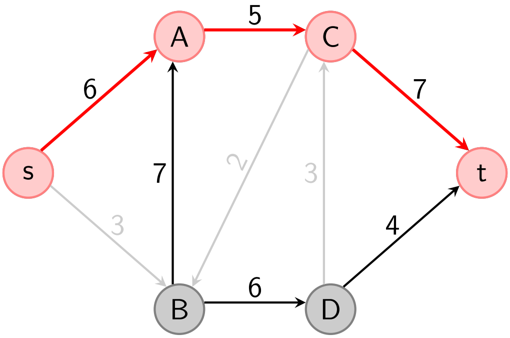


$\Delta = 4$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:

$$\Delta \leftarrow \Delta/2$$

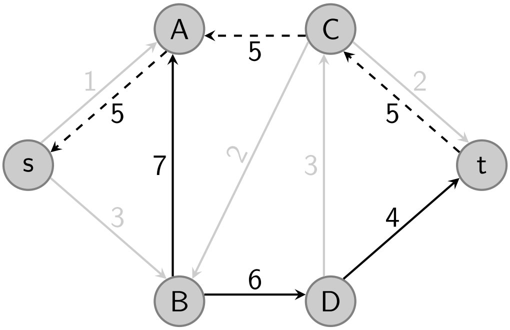


$\Delta = 4$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:

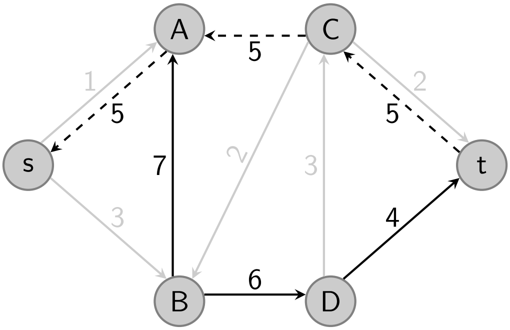
$$\Delta \leftarrow \Delta/2$$



$\Delta = 4$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:
  $$\Delta \leftarrow \Delta/2$$

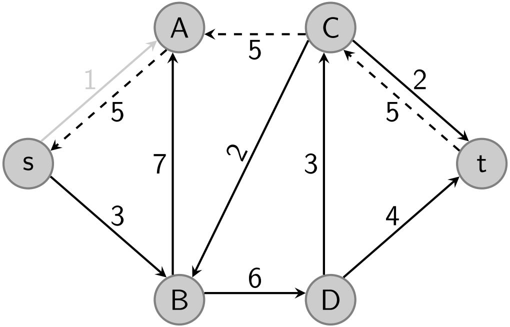


$\Delta = 2 = 4/2$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:

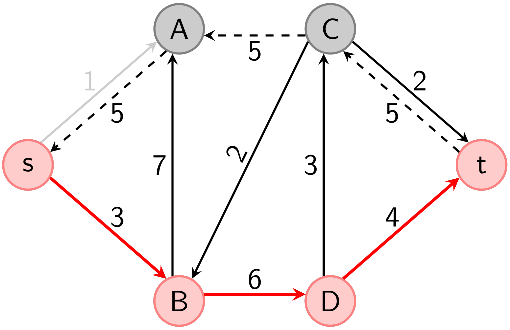
$$\Delta \leftarrow \Delta/2$$



$\Delta = 2$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:
$$\Delta \leftarrow \Delta/2$$

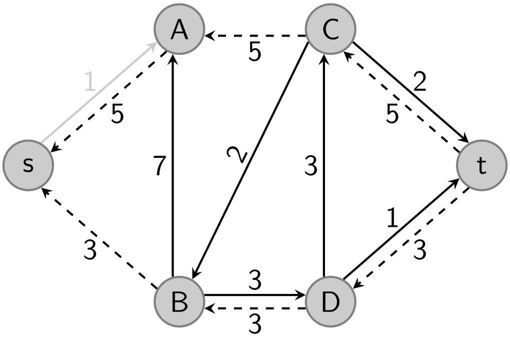


$\Delta = 2$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:

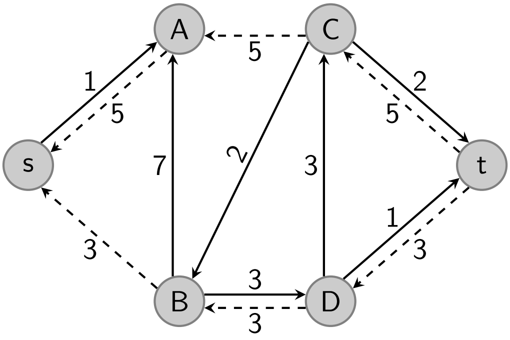
$$\Delta \leftarrow \Delta/2$$



$\Delta = 2$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:
  $\Delta \leftarrow \Delta/2$

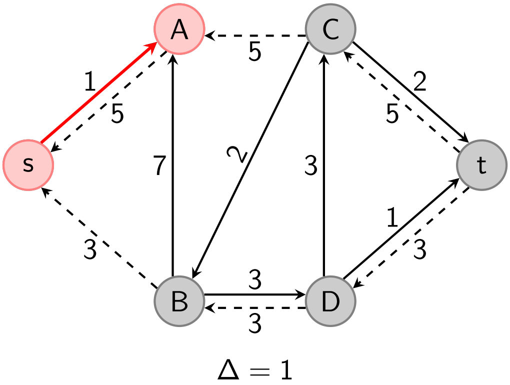


$\Delta = 1 = 2/2$

Capacity Scaling

- heuristic to choose paths of high capacity first
- highest capacity of all edges C_{max}
- lower capacity bound Δ
- start with $\Delta = 2^n \leq C_{max} < 2^{n+1}$
- only consider edges with capacities greater than or equal Δ to find paths
- decrease Δ if there are no more paths:

$$\Delta \leftarrow \Delta/2$$



Time Complexity of Capacity Scaling

Ford-Fulkerson Method (with BFS) / Edmonds-Karp Algorithm

$C_{max} \leftarrow$ max capacity edge; $\Delta \leftarrow 2^{\lfloor \log_2(C_{max}) \rfloor}$
initialize flow f with 0
while $\Delta \geq 1$ **do** $O(\log(C_{max}))$ iterations
    $G_f(\Delta) \leftarrow$ residual graph with edges of capacity $\geq \Delta$
    while there exists an augmenting path p in $G_f(\Delta)$ **do**
        augment flow f along p and update $G_f(\Delta)$
    end while
    $\Delta \leftarrow \Delta/2$
end while
return f

DFS: $O(E^2 \log(C_{max}))$ BFS: $O(EV \log(C_{max}))$

Dinic's Algorithm

- uses shortest augmenting paths (like Edmonds-Karp Algorithm)
- find augmenting flows in level graph created from BFS
- time complexity: $O(V^2E)$

Dinic's Algorithm

initialize flow f with 0
while there exists an s-t-path in the residual network G_f **do**
    $G_L \leftarrow$ level graph from BFS on G_f
    find blocking flow f' in G_L by DFS
    augment flow f by f'
end while
return f

Definition (Level Graph)

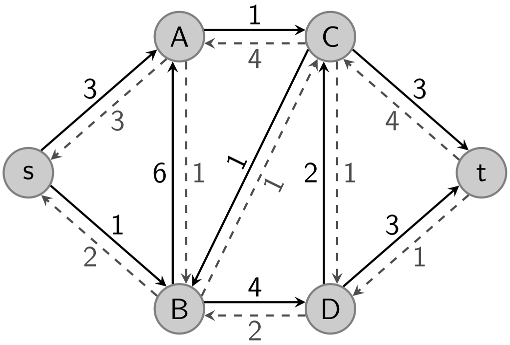
The level graph of a residual network $G_f = (V, E_f)$ is a graph $G_L = (V, E_L)$ with

$$E_L = \{(u, v) \in E_f \mid \mathrm{dist}(v) = \mathrm{dist}(u) + 1\}$$

Definition (Level Graph)

The level graph of a residual network $G_f = (V, E_f)$ is a graph $G_L = (V, E_L)$ with

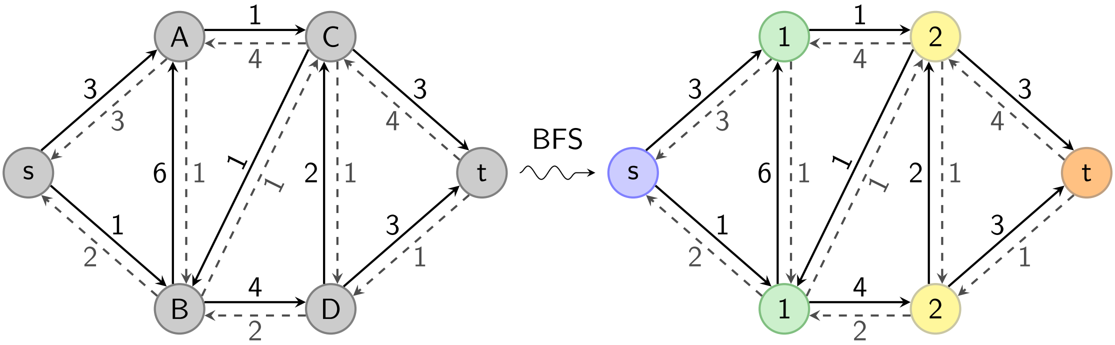
$$E_L = \{(u, v) \in E_f \mid \text{dist}(v) = \text{dist}(u) + 1\}$$



Definition (Level Graph)

The level graph of a residual network $G_f = (V, E_f)$ is a graph $G_L = (V, E_L)$ with

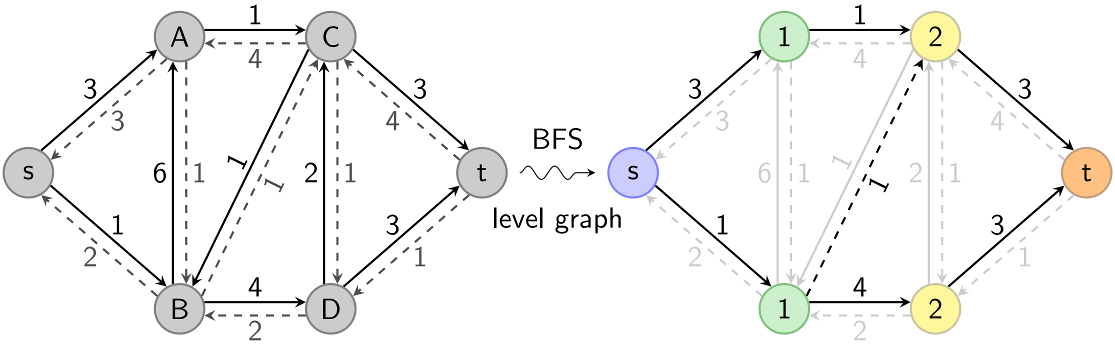
$$E_L = \{(u, v) \in E_f \mid \text{dist}(v) = \text{dist}(u) + 1\}$$



Definition (Level Graph)

The level graph of a residual network $G_f = (V, E_f)$ is a graph $G_L = (V, E_L)$ with

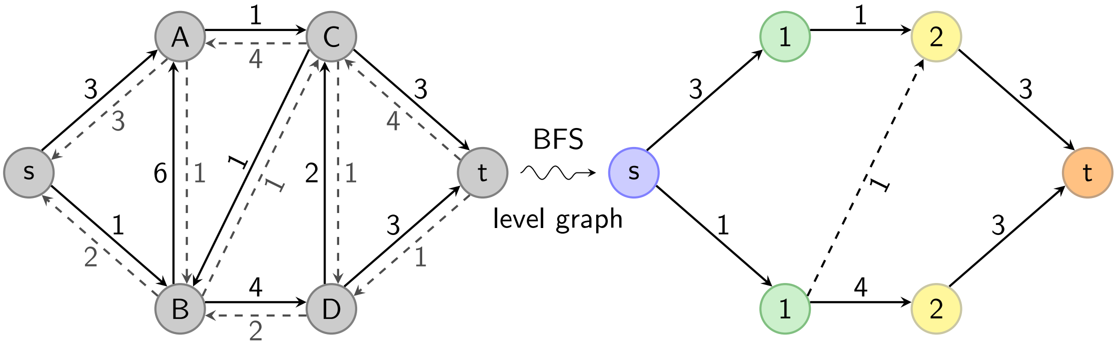
$$E_L = \{(u, v) \in E_f \mid \mathrm{dist}(v) = \mathrm{dist}(u) + 1\}$$



Definition (Level Graph)

The level graph of a residual network $G_f = (V, E_f)$ is a graph $G_L = (V, E_L)$ with

$$E_L = \{(u, v) \in E_f \mid \text{dist}(v) = \text{dist}(u) + 1\}$$



Definition

A blocking flow is a maximum flow in the level graph G_L.

Definition

A blocking flow is a maximum flow in the level graph G_L.

Dinic's Algorithm

initialize flow f with 0
while there exists an s-t-path in G_f **do**
  $G_L \leftarrow$ level graph from BFS on G_f
  find blocking flow f' in G_L by DFS
  augment flow f by f'
end while
return f

Definition

A blocking flow is a maximum flow in the level graph G_L.

Dinic's Algorithm

initialize flow f with 0
while there exists an s-t-path in G_f **do** $O(V)$
  $G_L \leftarrow$ level graph from BFS on G_f
  find blocking flow f' in G_L by DFS
  augment flow f by f'
end while
return f

- each iteration increases s-t distance by 1, so only $O(V)$ blocking flows

Definition

A blocking flow is a maximum flow in the level graph G_L.

Dinic's Algorithm

initialize flow f with 0
while there exists an s-t-path in G_f **do** $O(V)$
  $G_L \leftarrow$ level graph from BFS on G_f $O(E)$
  find blocking flow f' in G_L by DFS
  augment flow f by f' $O(E)$
end while
return f

- each iteration increases s-t distance by 1, so only $O(V)$ blocking flows

Definition

A blocking flow is a maximum flow in the level graph G_L.

Dinic's Algorithm

initialize flow f with 0
while there exists an s-t-path in G_f **do** $O(V)$
  $G_L \leftarrow$ level graph from BFS on G_f $O(E)$
  find blocking flow f' in G_L by DFS $O(EV)$
  augment flow f by f' $O(E)$
end while
return f

- each iteration increases s-t distance by 1, so only $O(V)$ blocking flows
- every augmenting path in G_L saturates one edge, reverse edge not part of level graph, so $O(E)$ iterations

Definition

A blocking flow is a maximum flow in the level graph G_L.

Dinic's Algorithm

initialize flow f with 0
while there exists an s-t-path in G_f **do** $O(V)$
  $G_L \leftarrow$ level graph from BFS on G_f $O(E)$
  find blocking flow f' in G_L by DFS $O(EV)$
  augment flow f by f' $O(E)$
end while
return f

- each iteration increases s-t distance by 1, so only $O(V)$ blocking flows
- every augmenting path in G_L saturates one edge, reverse edge not part of level graph, so $O(E)$ iterations
- DFS and augmentation in level graph G_L only costs $O(V)$, since augmenting paths are shortest paths

Definition

A blocking flow is a maximum flow in the level graph G_L.

Dinic's Algorithm

initialize flow f with 0
while there exists an s-t-path in G_f **do** $O(V)$
  $G_L \leftarrow$ level graph from BFS on G_f $O(E)$
  find blocking flow f' in G_L by DFS $O(EV)$
  augment flow f by f' $O(E)$
end while
return f

- each iteration increases s-t distance by 1, so only $O(V)$ blocking flows
- every augmenting path in G_L saturates one edge, reverse edge not part of level graph, so $O(E)$ iterations

- DFS and augmentation in level graph G_L only costs $O(V)$, since augmenting paths are shortest paths

$\Rightarrow$ time complexity of $O(V(E + EV)) = O(V^2E)$

Algorithms & Time Complexity (Summary)

Algorithm	Time Complexity	Comment
Ford-Fulkerson Method	$O(\lvert f \rvert E)$	using DFS
Edmonds-Karp	$O(VE^2)$	Ford-Fulkerson + BFS
Capacity Scaling + DFS	$O(E^2 \log(C_{max}))$	lower capacity bound Δ
Capacity Scaling + BFS	$O(EV \log(C_{max}))$	lower capacity bound Δ
Dinic's Algorithm	$O(V^2 E)$	level graph + DFS
Modification of Dinic's	$O(V^3)$	new approach to find blocking flows

Bibliography

1. Abolfathe, S., Quinonez, J. *Blocking Flows.* October 4, 2006.
2. Chaudhuri, K. *CSE 202: Design and Analysis of Algorithms. Lecture 8.* University of California. 2011.
3. Cormen, Leiserson, Rivest, Stein. *Introduction to Algorithms. Fourth Edition.* Cambridge, Massachusetts: The MIT Press (2022).
4. Devadas S. *Incremental Improvement: Max Flow, Min Cut.* MIT OpenCourseWare. March 4, 2016.
5. Fiset, W. *Capacity Scaling |Network Flow |Graph Theory.* October 22, 1018.
6. Fiset, W. *Dinic's Algorithm |Network Flow |Graph Theory.* November 17, 2018.
7. Wikipedia. *Flow networks, Maximum flow problem, Ford-Fulkerson algorithm, Dinic's algorithm.* 2024.