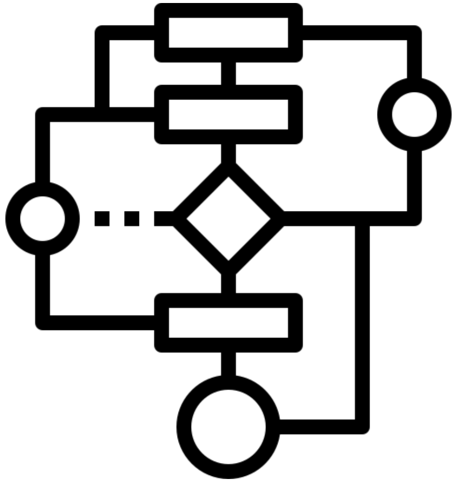


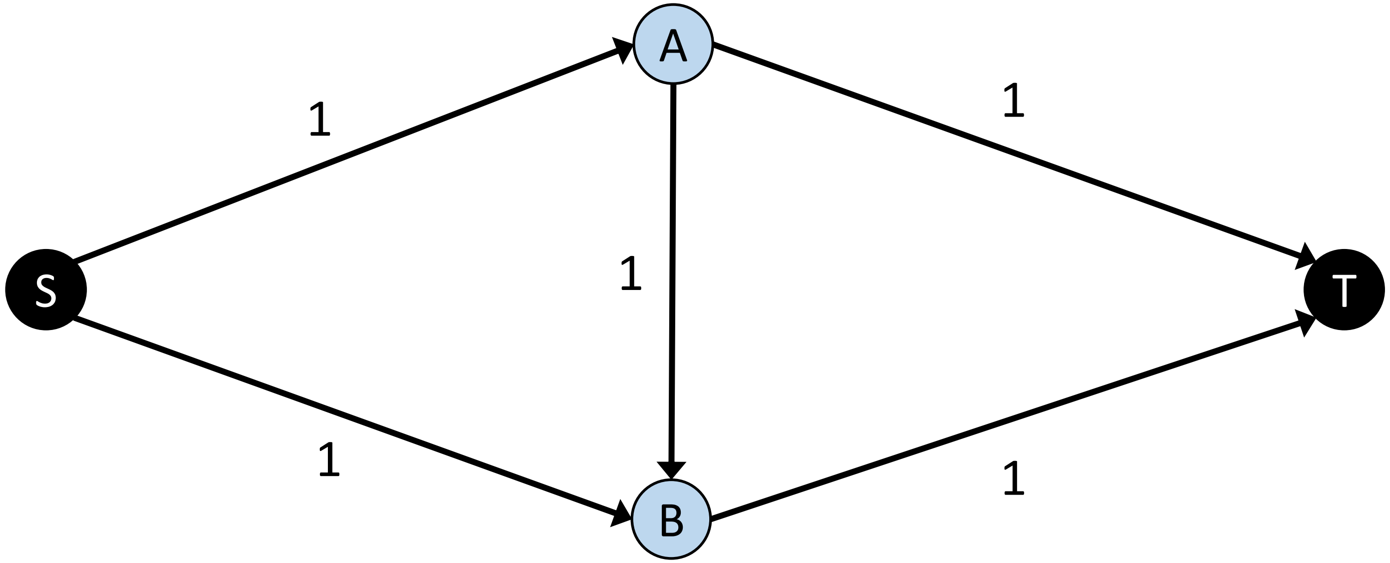
CS 310: Algorithms

Lecture 22

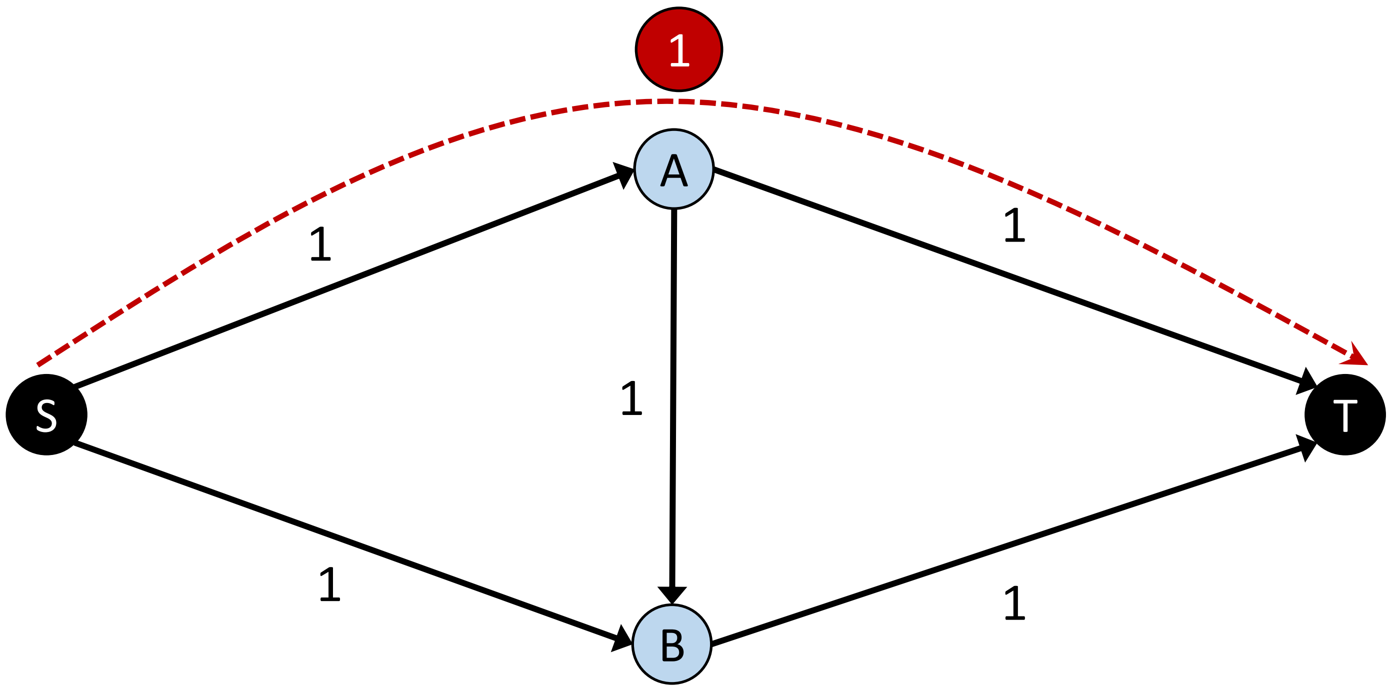
Instructor: Naveed Anwar Bhatti

Few Slides taken from Dr. Imdad's CS 510 course

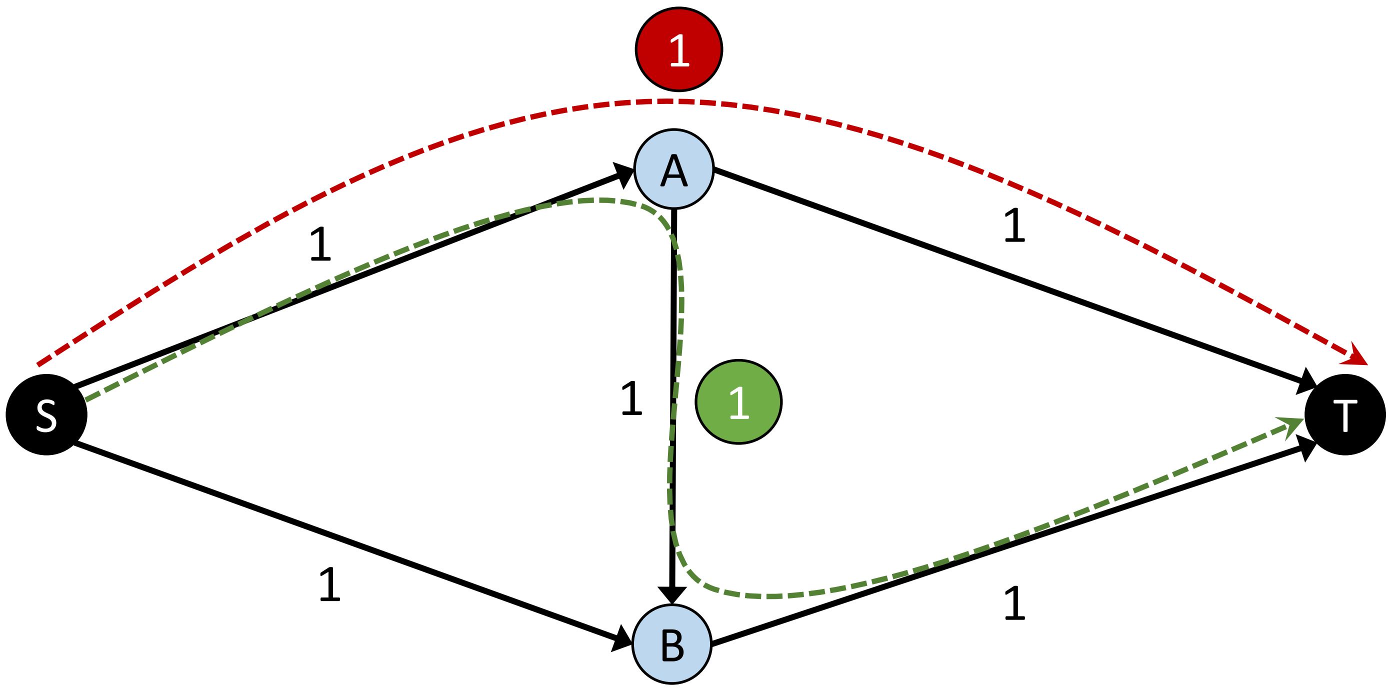
Max Flow – Problem with the Algorithm



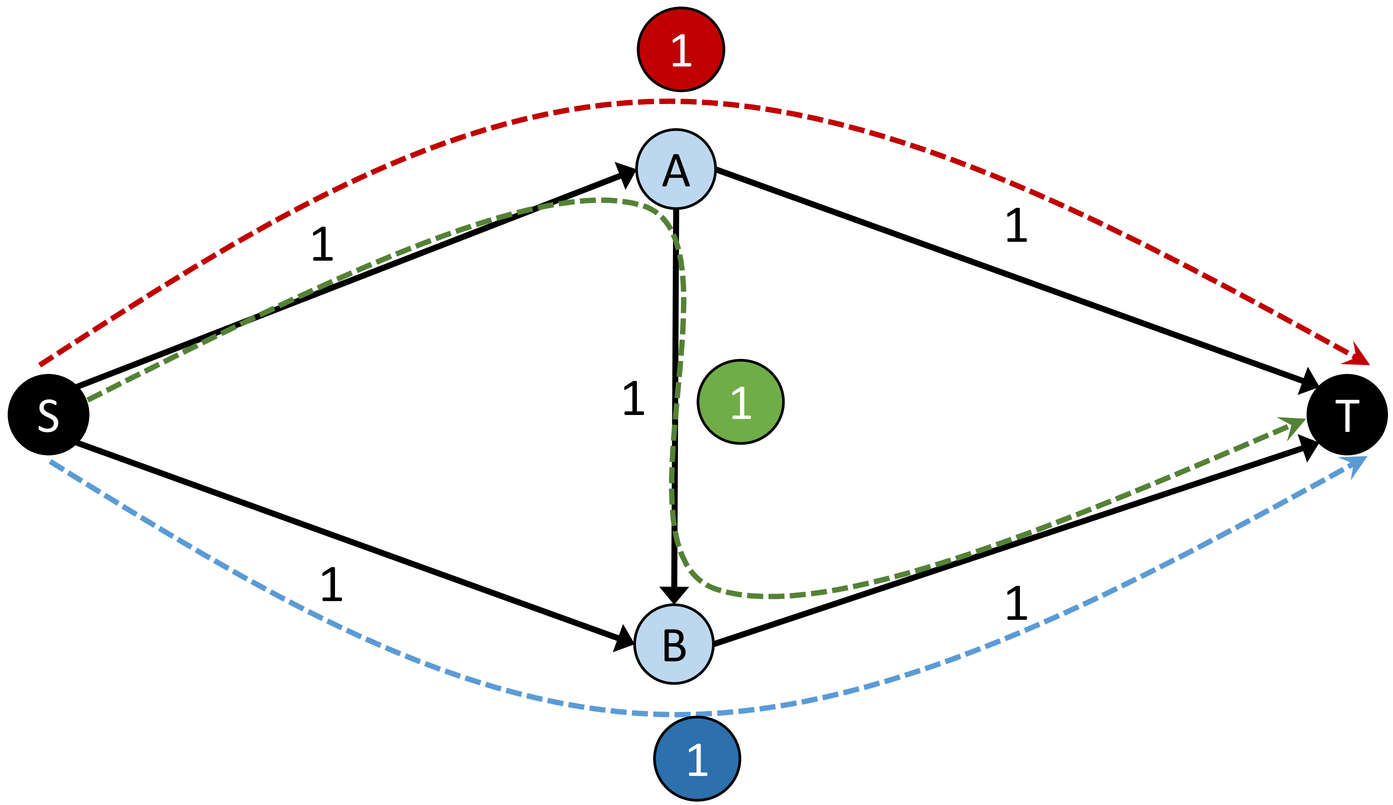
Max Flow – Problem with the Algorithm



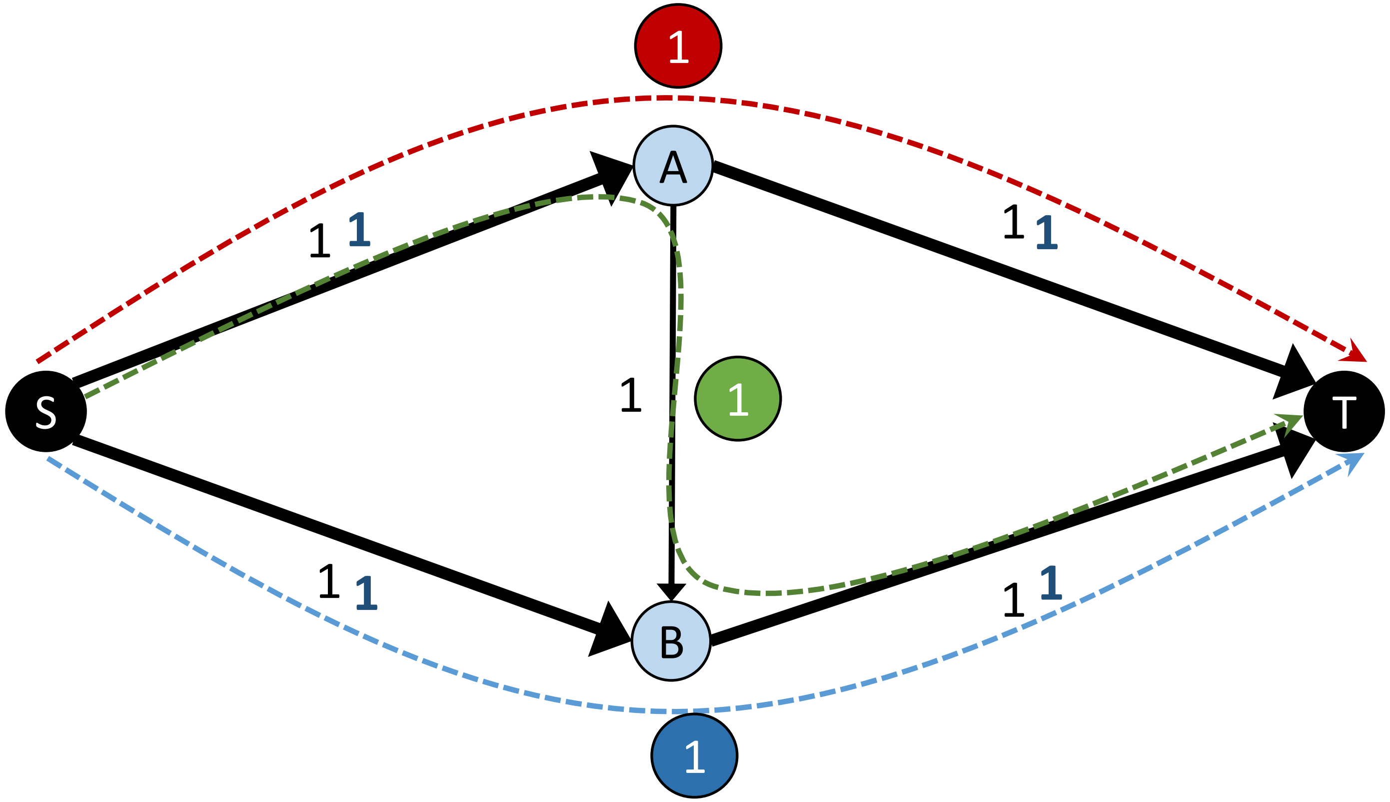
Max Flow – Problem with the Algorithm



Max Flow – Problem with the Algorithm



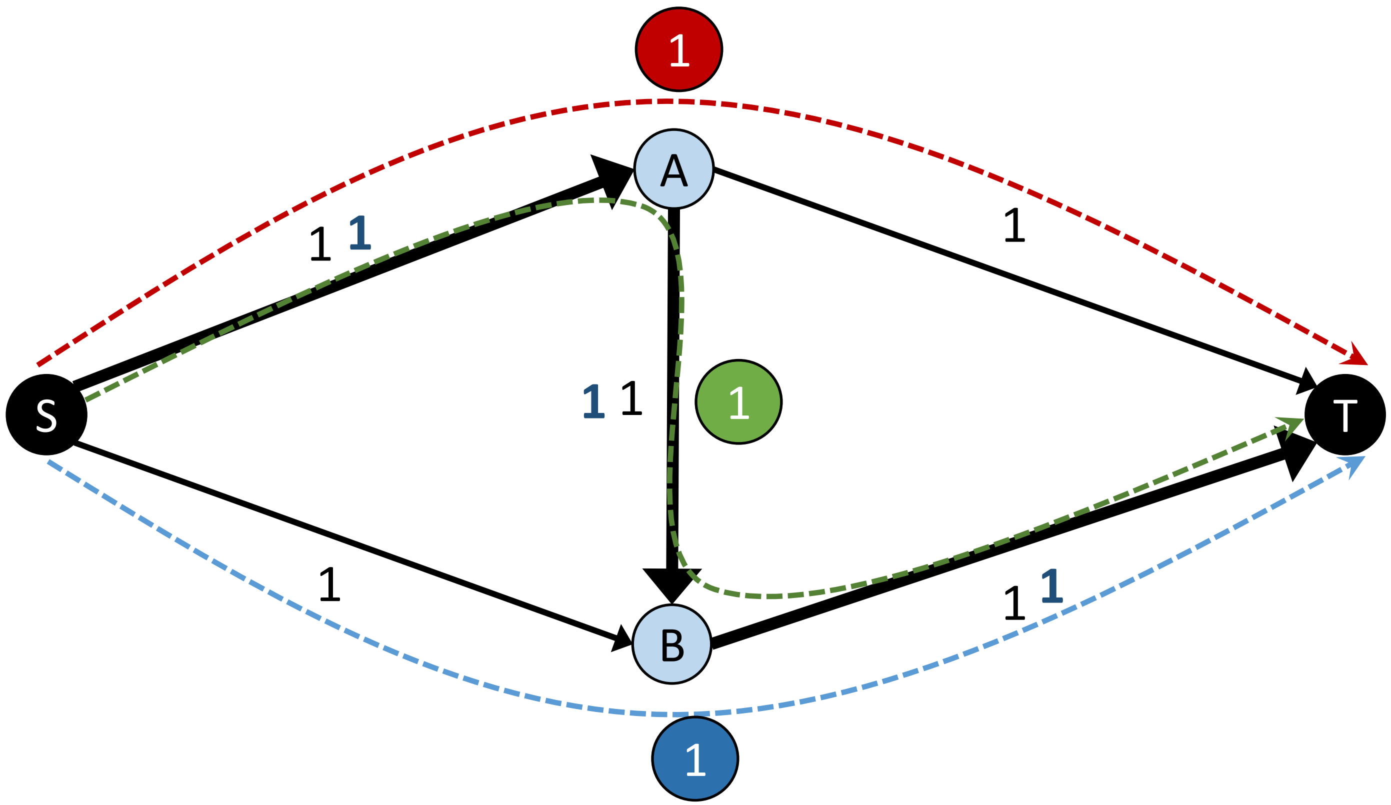
Max Flow – Problem with the Algorithm



1 + 1 = 2

The max flow clearly is of **size 2**

Max Flow – Problem with the Algorithm

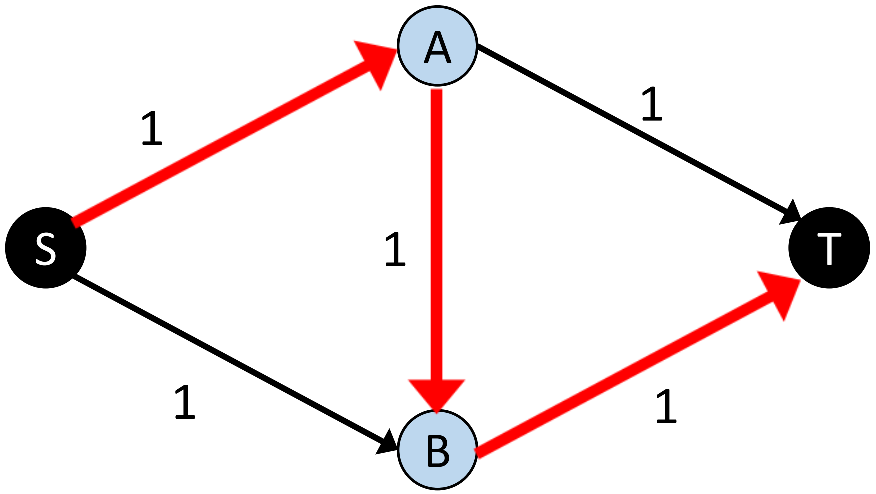


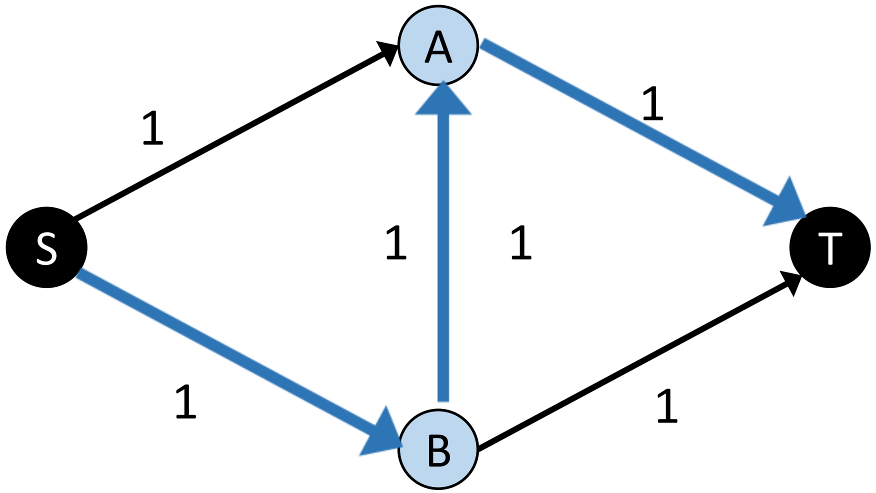
If the greedy algorithm adds a flow of size 1 via the ***s* - *t*** path **s, a, b, t**

No ***s* - *t*** path in the remaining graph

Max Flow – Fix for the Algorithm

- A more general way of pushing further flow is to push forward flow on edges where some capacity is remaining
- Cancel existing flow on the edges already carrying some flow
- Think of it as pushing flow backward

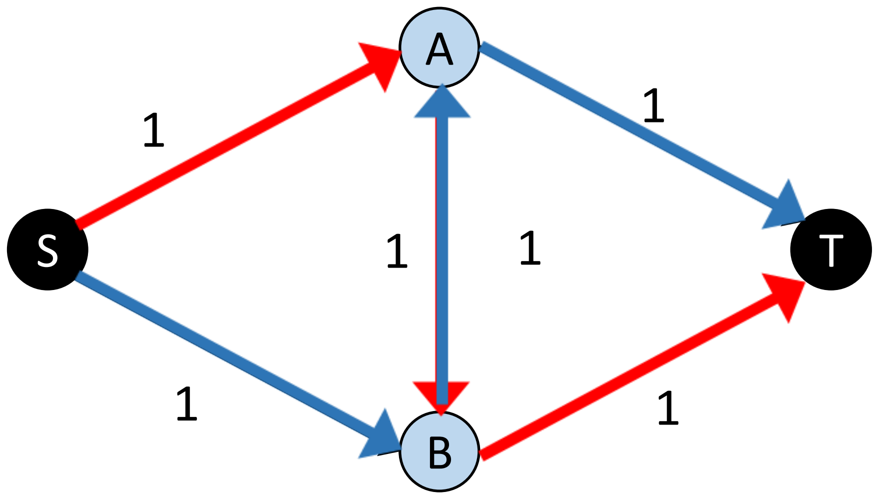




- Add one unit of flow via the **s, b, a, t path**
- **ba ∉ E**, but we can cancel the existing flow on the **ab ∈ E**

Max Flow – Fix for the Algorithm

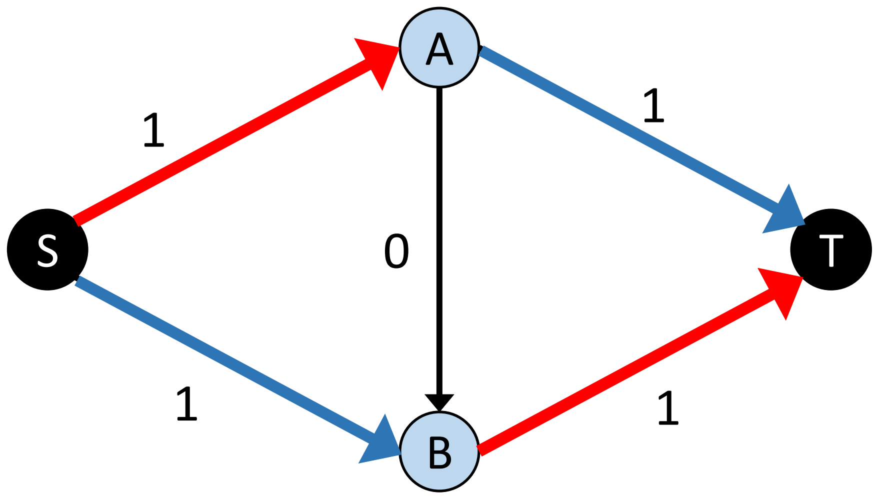
- A more general way of pushing further flow is to push forward flow on edges where some capacity is remaining
- Cancel existing flow on the edges already carrying some flow
- Think of it as pushing flow backward



- Add one unit of flow via the **s, b, a, t path**
- **ba ∉ E**, but we can cancel the existing flow on the **ab ∈ E**

Max Flow – Fix for the Algorithm

- A more general way of pushing further flow is to push forward flow on edges where some capacity is remaining
- Cancel existing flow on the edges already carrying some flow
- Think of it as pushing flow backward



- Add one unit of flow via the **s, b, a, t path**
- **ba ∉ E**, but we can cancel the existing flow on the **ab ∈ E**

Max Flow – Residual Network

- Cancellation of existing flows on edges (if need be) is the right framework to add more flow

- A systematic way to search for the right place to cancel flow and adding more flow is to use the **residual network**

Max Flow – Residual Network

- Given a network G and a flow f on G, the residual graph G_f of G with respect to f is defined as follows:

Max Flow – Residual Network

- Given a network G and a flow f on G, the residual graph G_f of G with respect to f is defined as follows:
- Vertex set of G_f is the same as that of G

Max Flow – Residual Network

- Given a network G and a flow f on G, the residual graph G_f of G with respect to f is defined as follows:
- Vertex set of G_f is the same as that of G
- forward edges: For each $e = uv$ of G on which $f_e < c_e$, there is an edge $e = uv$ in G_f with a capacity $c_e - f_e > 0$
  - we can push forward $c_e - f_e$ residual capacity units of flow on e

Max Flow – Residual Network

- Given a network G and a flow f on G, the residual graph G_f of G with respect to f is defined as follows:
- Vertex set of G_f is the same as that of G
- forward edges: For each $e = uv$ of G on which $f_e < c_e$, there is an edge $e = uv$ in G_f with a capacity $c_e - f_e > 0$
  - we can push forward $c_e - f_e$ residual capacity units of flow on e
- backward edges: For each edge $e = uv$ of G on which $f_e > 0$, there is an edge $e' = vu$ in G_f with a capacity of f_e

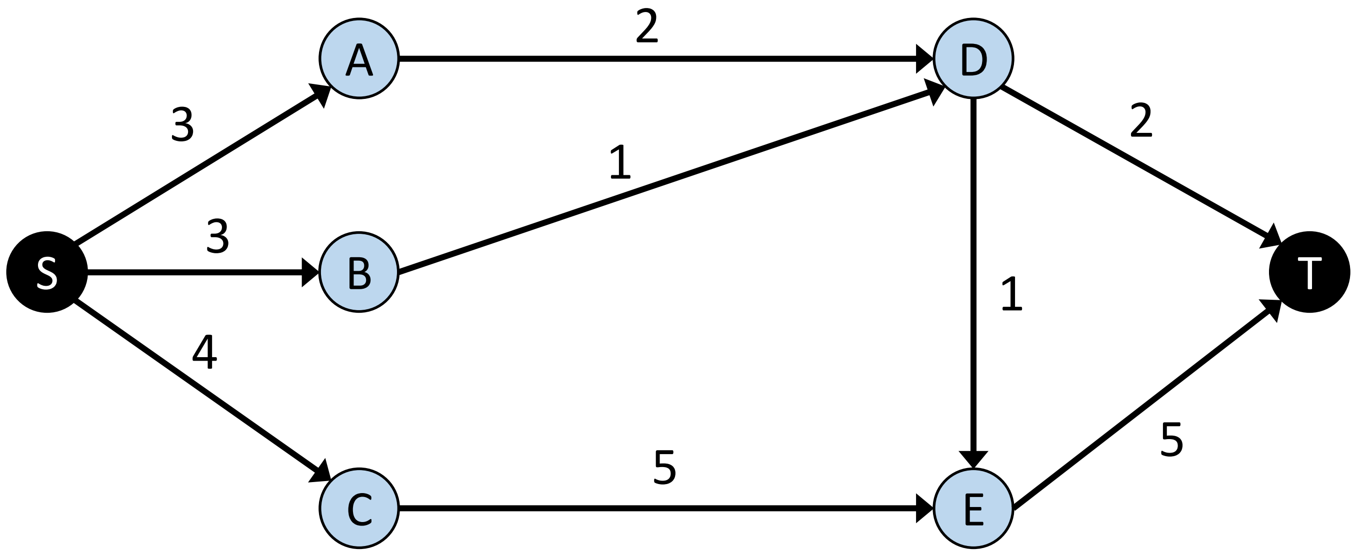
Max Flow – Residual Network

- Given a network G and a flow f on G, the residual graph G_f of G with respect to f is defined as follows:
- Vertex set of G_f is the same as that of G
- forward edges: For each $e = uv$ of G on which $f_e < c_e$, there is an edge $e = uv$ in G_f with a capacity $c_e - f_e > 0$
  - we can push forward $c_e - f_e$ residual capacity units of flow on e
- backward edges: For each edge $e = uv$ of G on which $f_e > 0$, there is an edge $e' = vu$ in G_f with a capacity of f_e
  - we can cancel or push backward f_e units of flow $c_{e'} = f_e$ on e

Max Flow – Residual Network

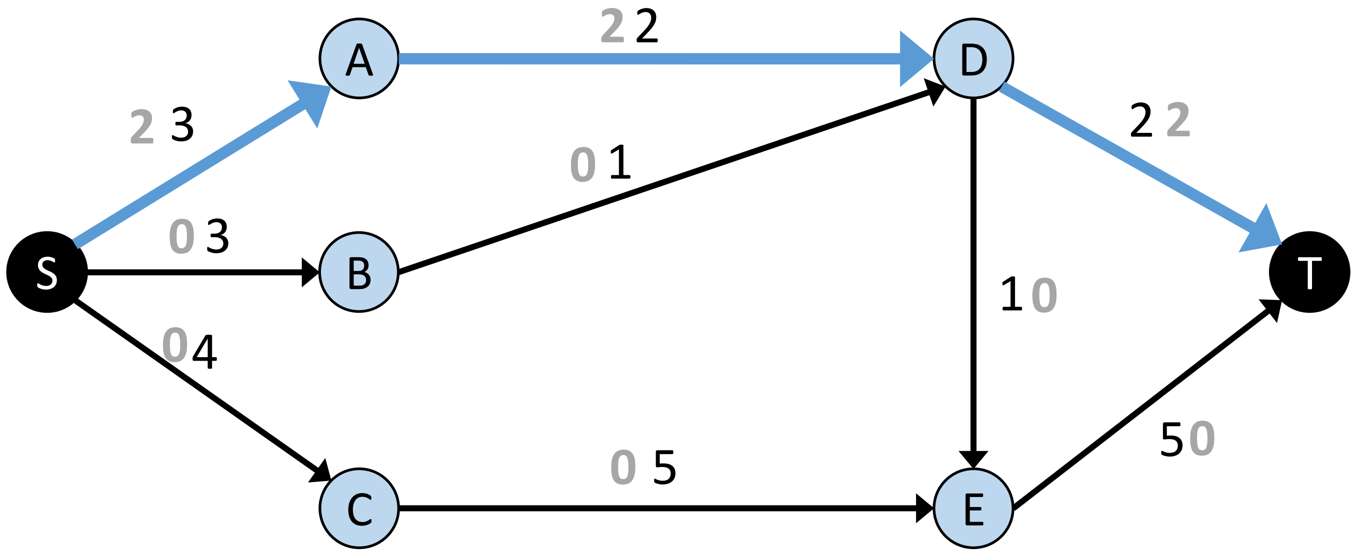
- Given a network G and a flow f on G, the residual graph G_f of G with respect to f is defined as follows:
- Vertex set of G_f is the same as that of G
- forward edges: For each $e = uv$ of G on which $f_e < c_e$, there is an edge $e = uv$ in G_f with a capacity $c_e - f_e > 0$
  - we can push forward $c_e - f_e$ residual capacity units of flow on e
- backward edges: For each edge $e = uv$ of G on which $f_e > 0$, there is an edge $e' = vu$ in G_f with a capacity of f_e
  - we can cancel or push backward f_e units of flow $c_{e'} = f_e$ on e
- For any G and f, G_f has at most twice as many edges as G

Max Flow – Residual Network

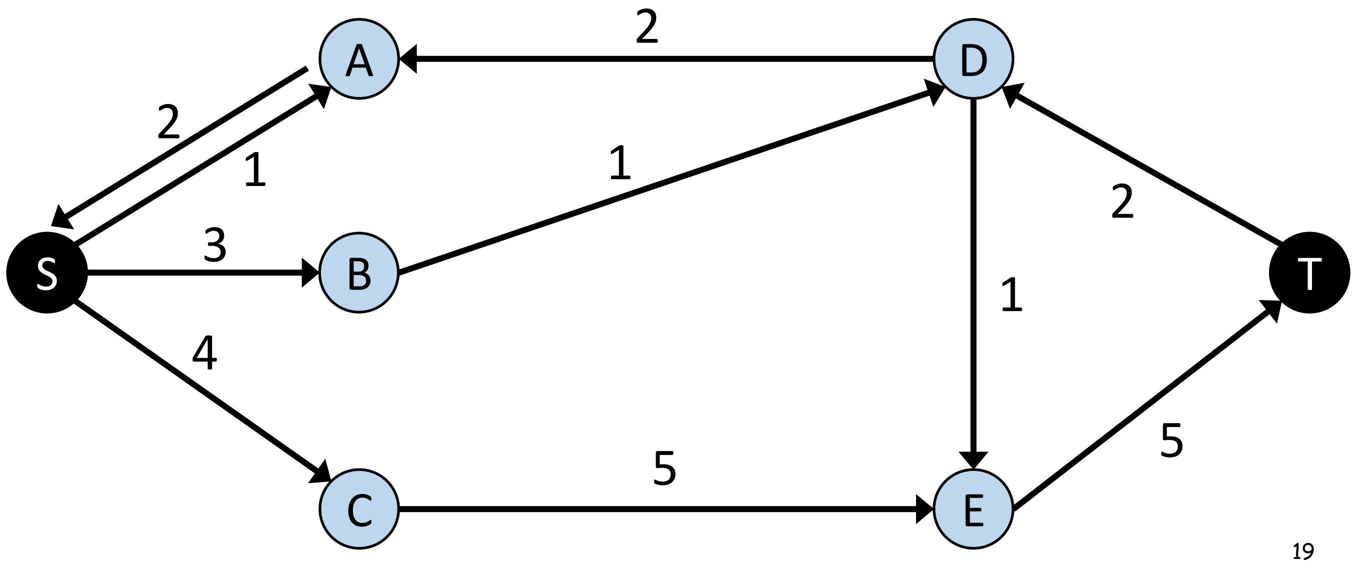


Max Flow – Residual Network

Flow network with flow shown in **blue**

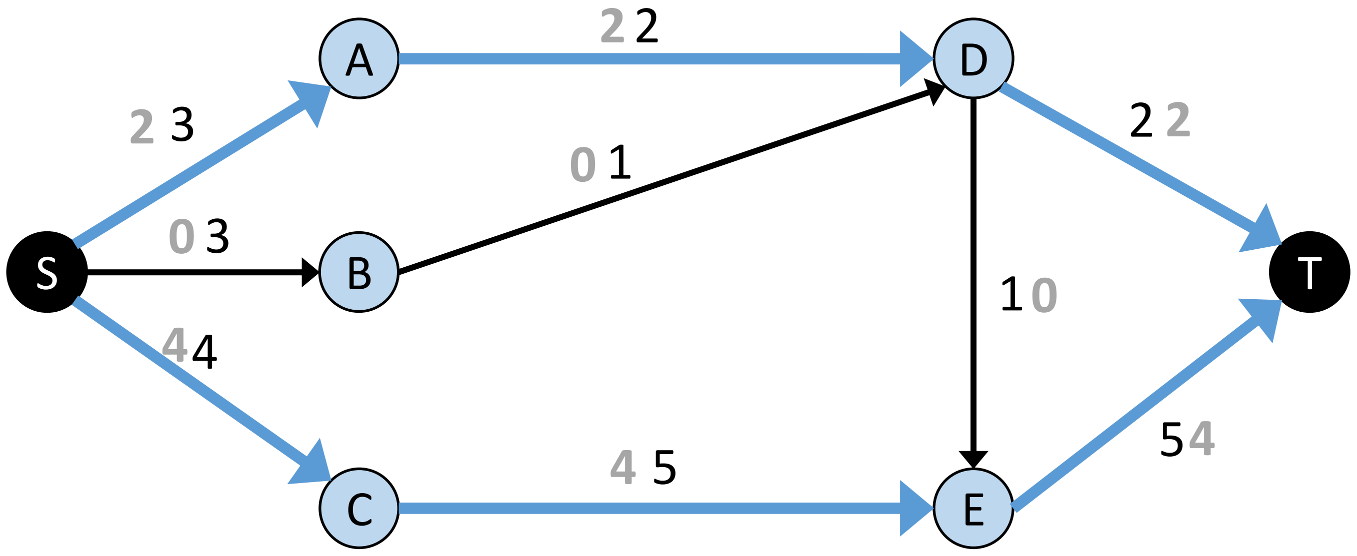


The corresponding **residual network**

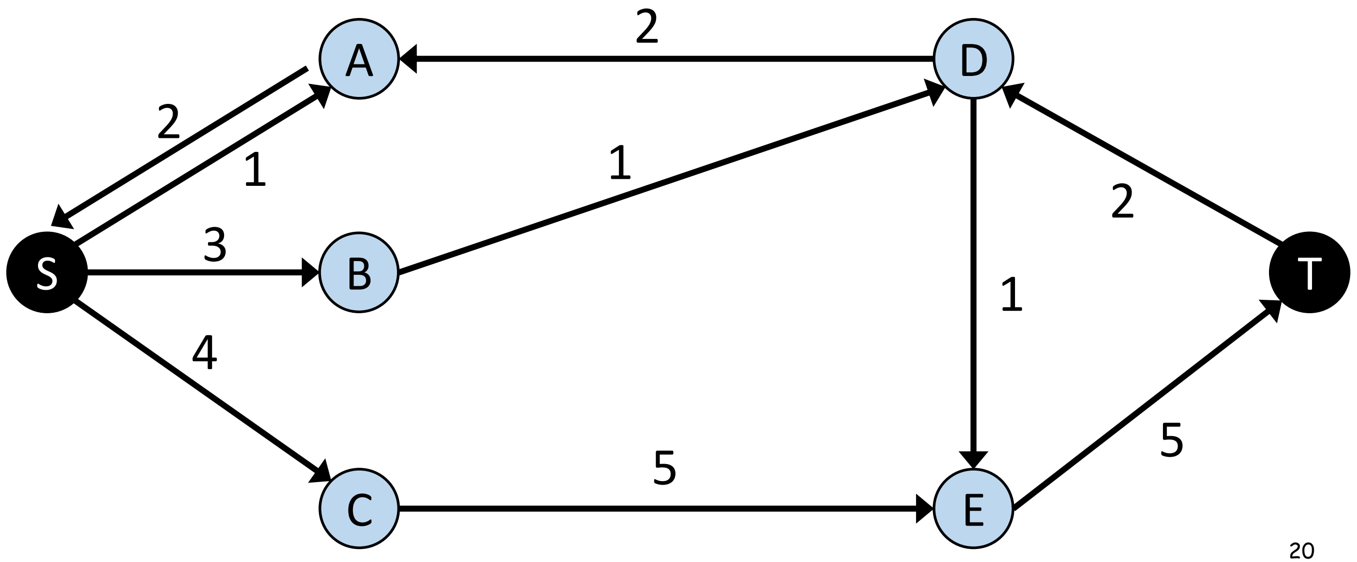


Max Flow – Residual Network

Flow network with flow shown in **blue**

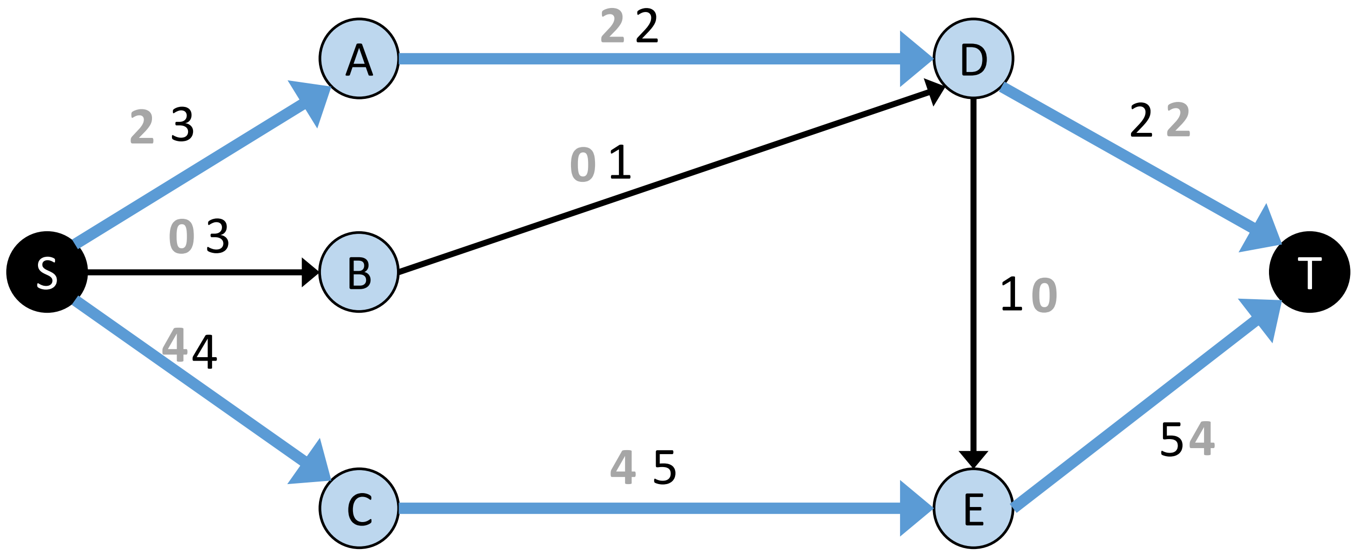


The corresponding **residual network**

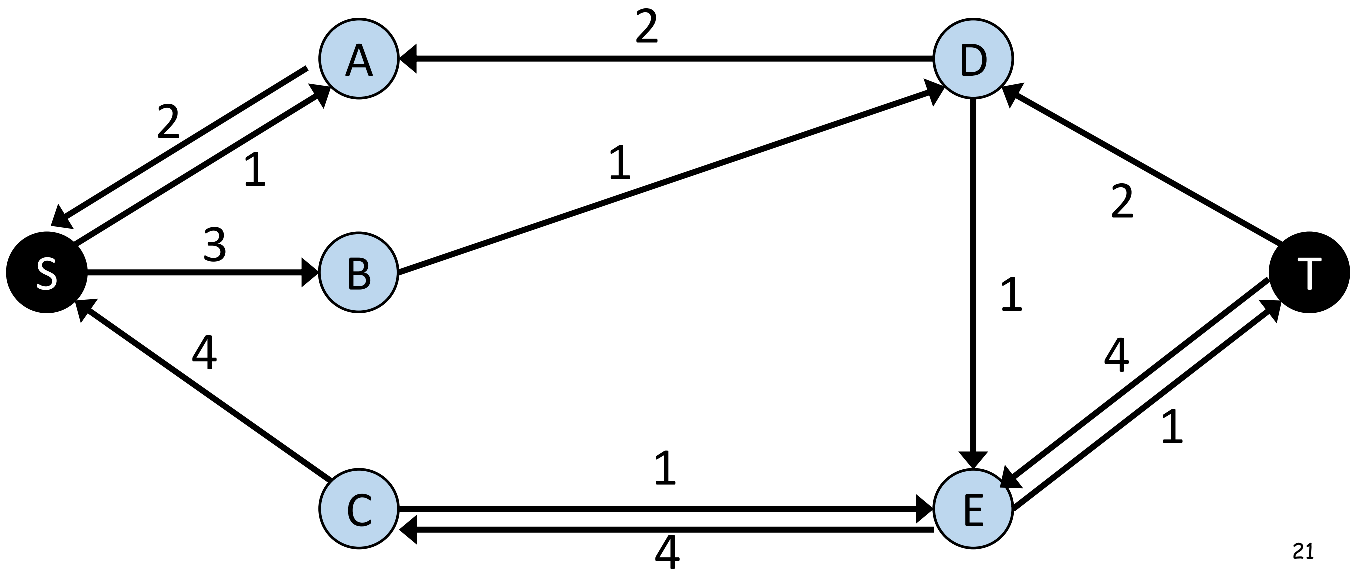


Max Flow – Residual Network

Flow network with flow shown in **blue**



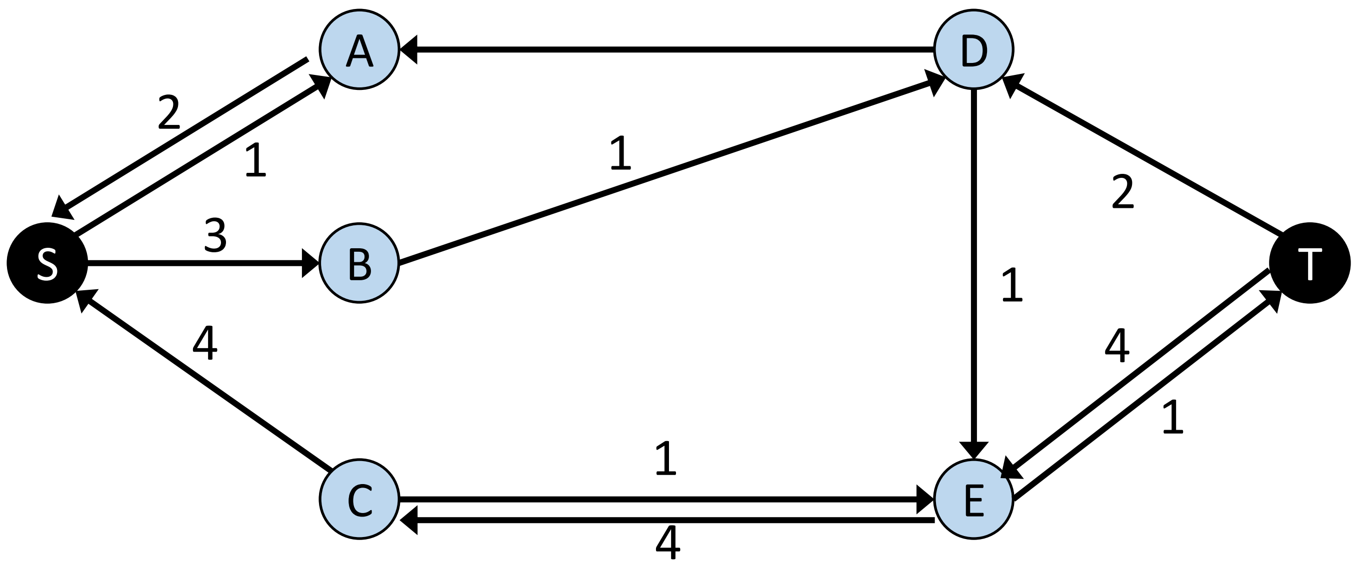
The corresponding **residual network**



Max Flow – Augmenting Path

An **augmenting path** is a simple ***s - t*** path in the residual graph $\mathbf{G_f}$

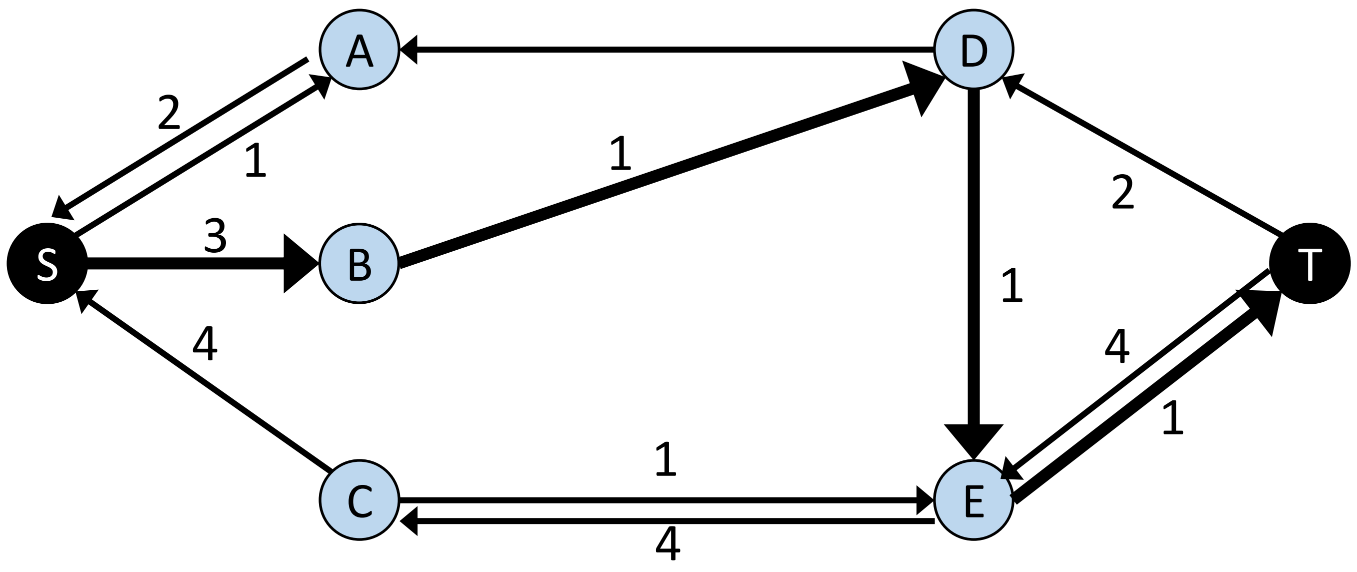
The corresponding **residual network**



Max Flow – Augmenting Path

An **augmenting path** is a simple ***s - t*** path in the residual graph $\mathbf{G_f}$

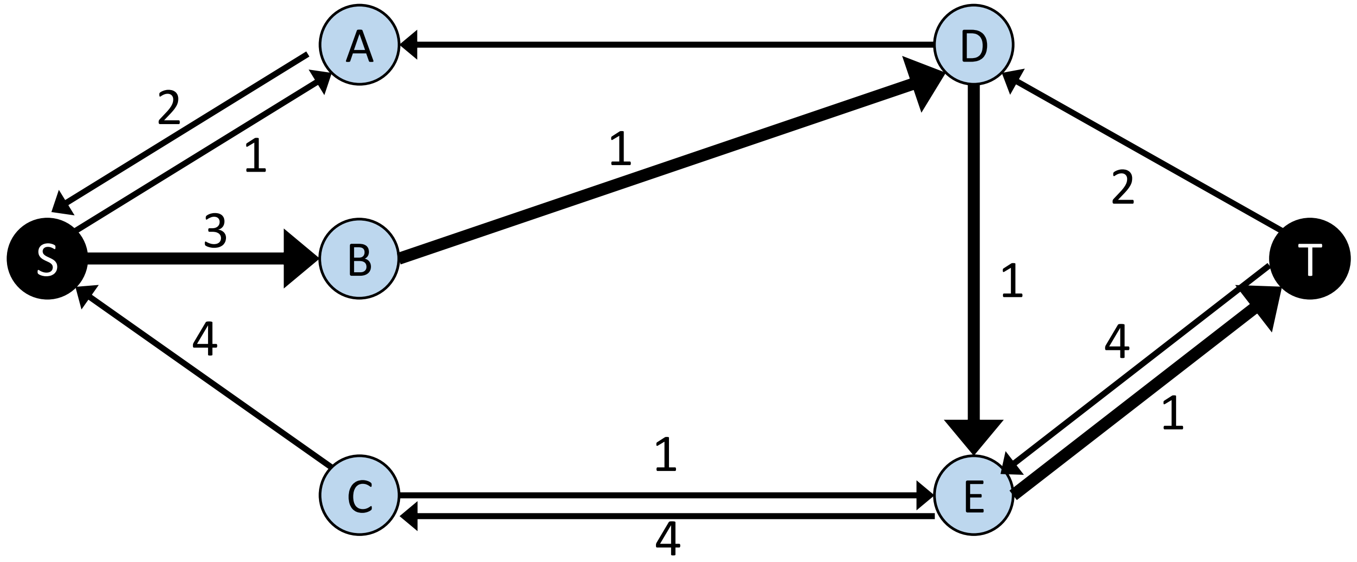
The corresponding **residual network**



Max Flow – Augmenting Path

An **augmenting path** is a simple ***s - t*** path in the residual graph $\mathbf{G_f}$

The corresponding **residual network**

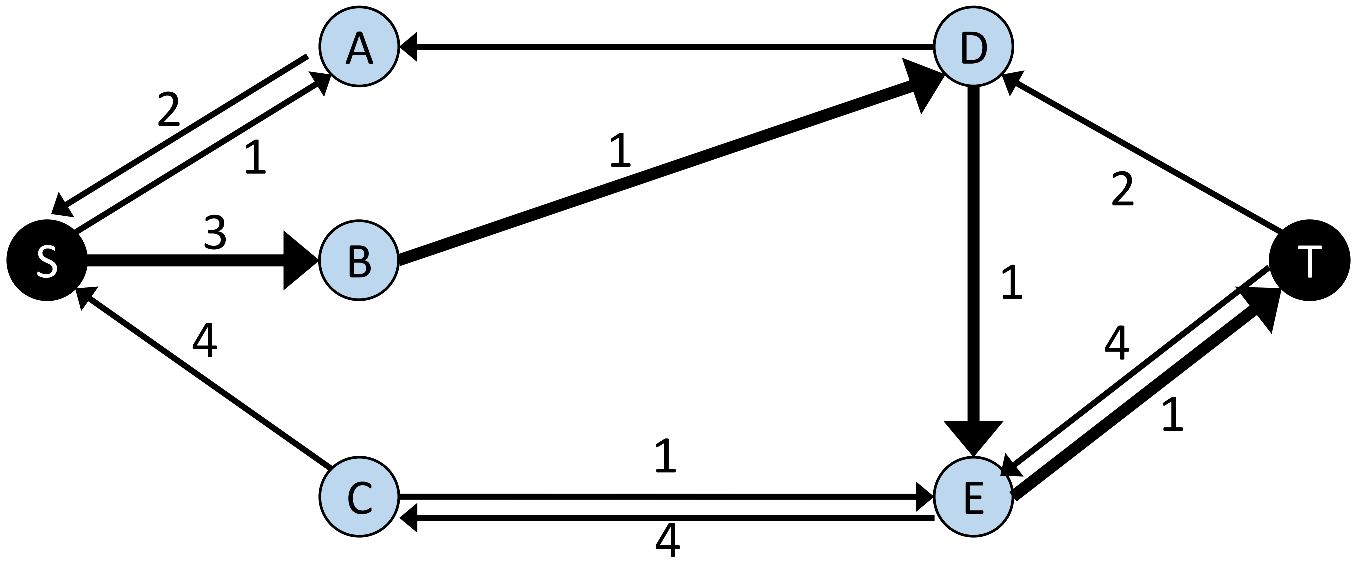


Augmenting path theorem: Flow *f* is a max flow ***iff*** there are no augmenting paths.

Max Flow – Augmenting Path

An **augmenting path** is a simple ***s* - *t*** path in the residual graph $\mathbf{G_f}$

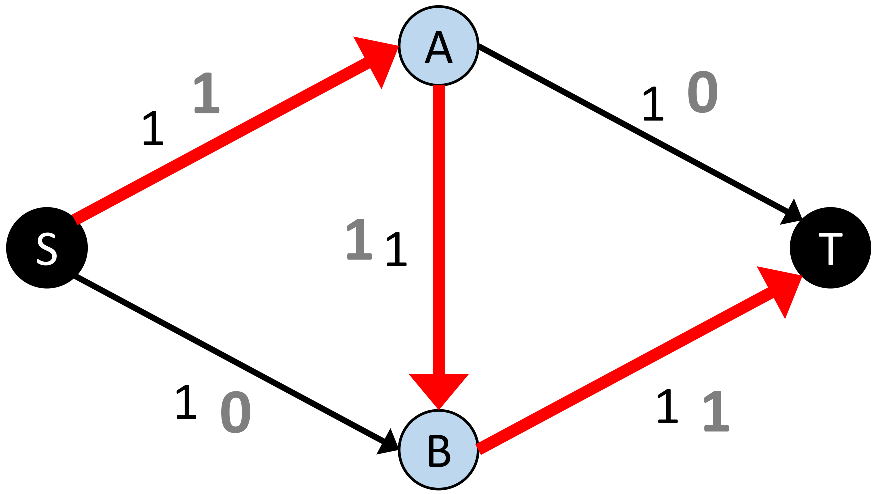
The corresponding **residual network**



Augmenting path theorem: Flow *f* is a max flow *iff* there are no augmenting paths.

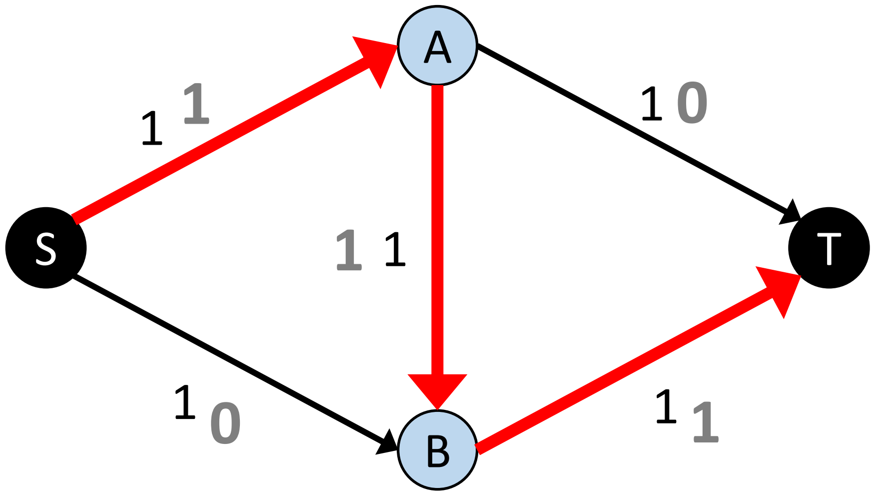
Max-flow min-cut theorem: [Elias-Feinstein-Shannon 1956, Ford-Fulkerson 1956]
The value of the max flow is equal to the value of the min cut.

Max Flow – Augmenting Path

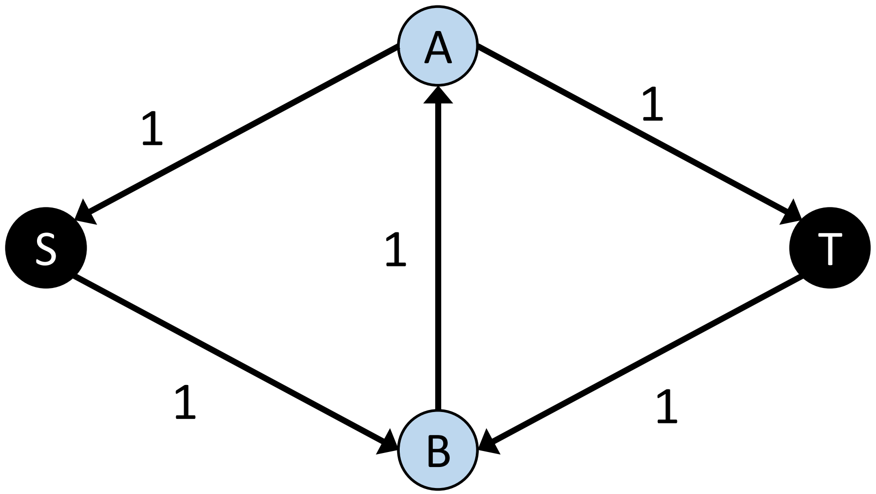


Flow network

Max Flow – Augmenting Path

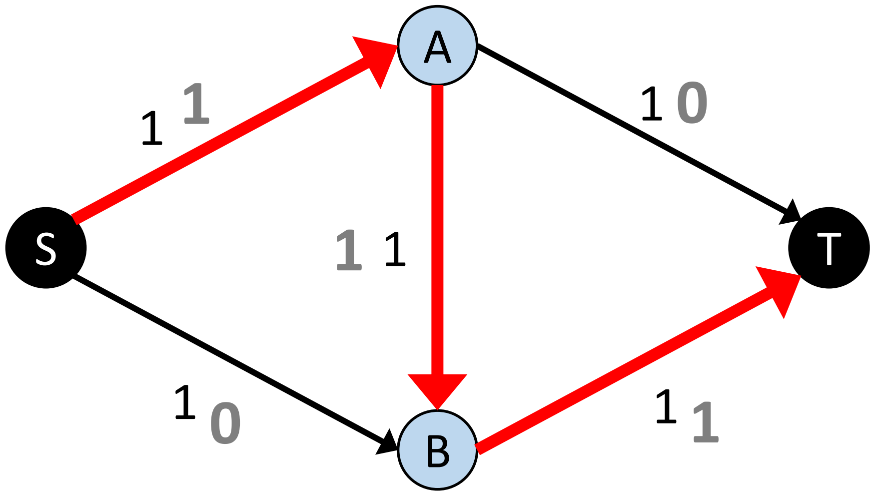


Flow network



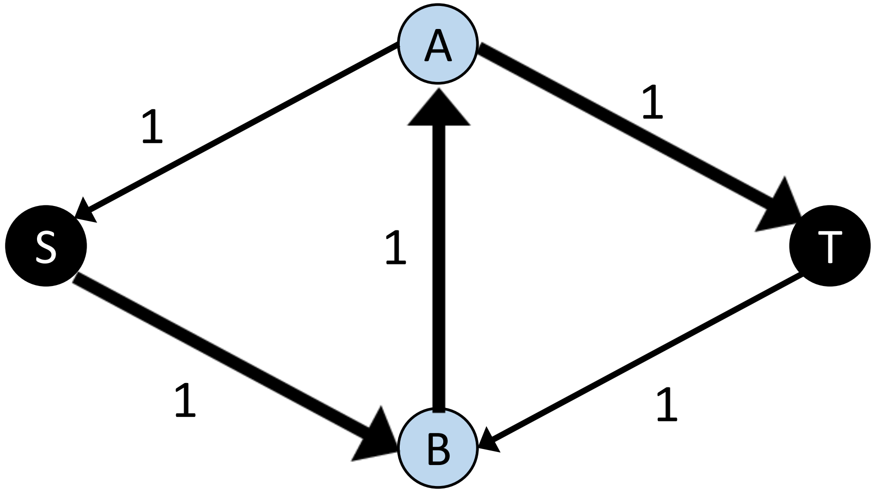
Residual network

Max Flow – Augmenting Path



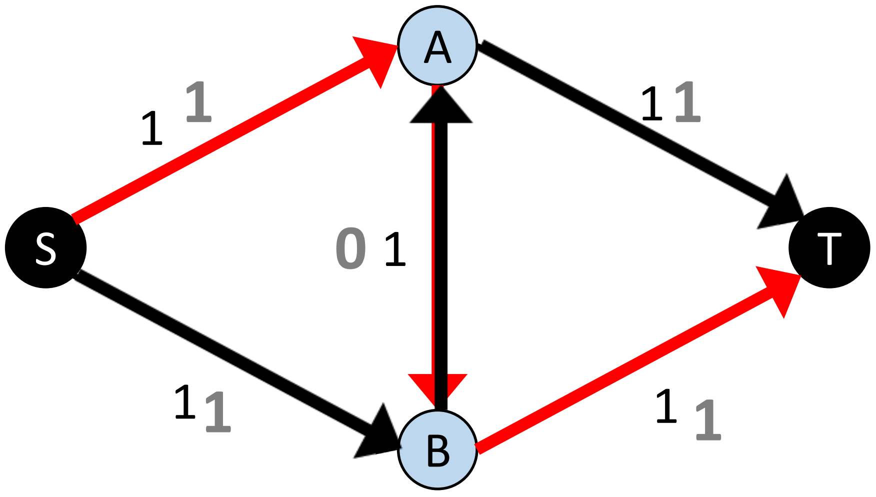
Flow network

Bottleneck (P) = 1



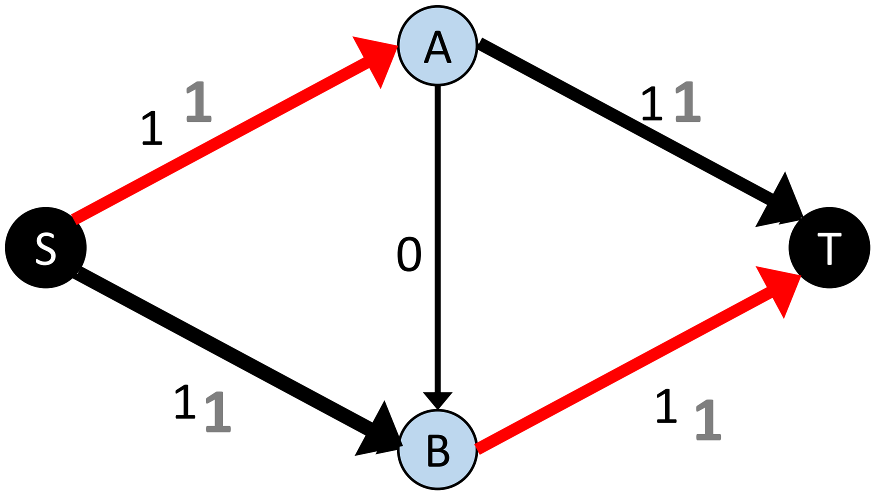
Residual network

Max Flow – Augmenting Path



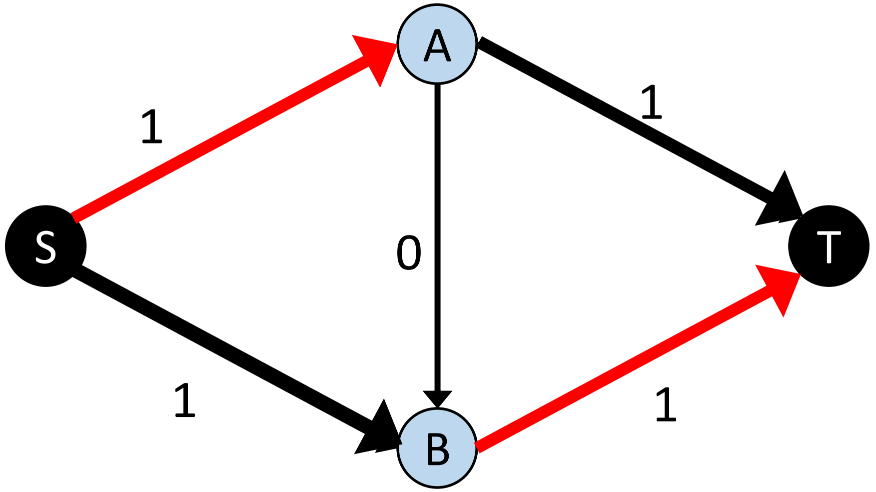
Flow network

Max Flow – Augmenting Path



Flow network

Max Flow – Augmenting Path



Algorithm $\textsc{Augment}(P, f)$

$b \leftarrow bottleneck(P, f)$
$f' \leftarrow f$
for each edge $e = uv \in P$ **do**
    if e is a forward edge **then**
        $f'_e \leftarrow f_e + b$
    else if e is a backward edge **then**
        $f'_{vu} \leftarrow f_{vu} - b$

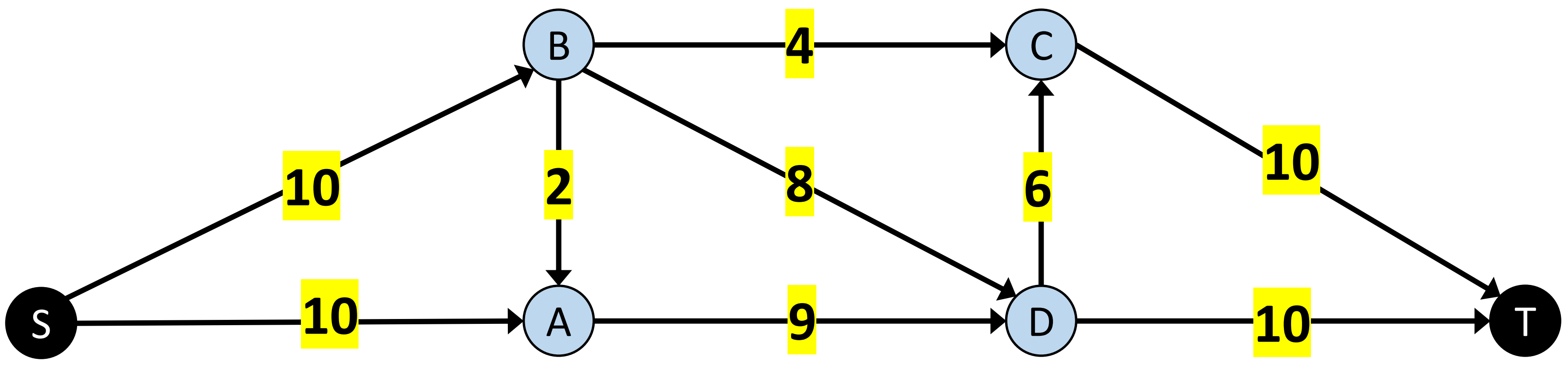
Max Flow – The Ford-Fulkerson Algorithm

Given a flow network G with source s and t

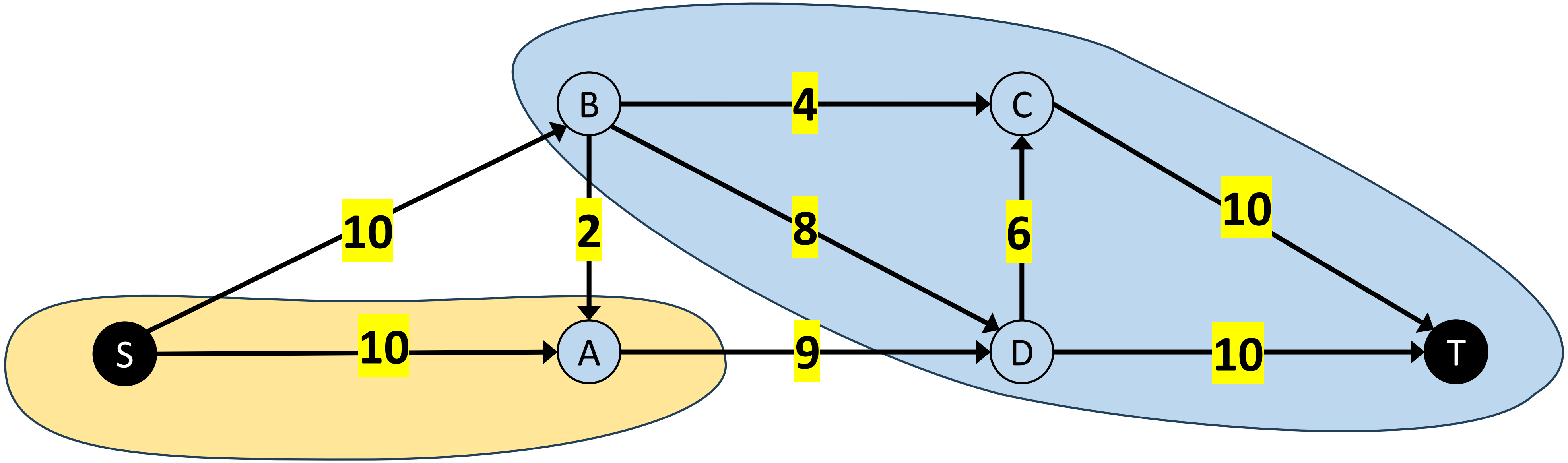
Algorithm Ford-Fulkerson Algorithm (G)

```
f ← 0                                  ▷ Initialize to a (valid) flow of size 0 (on every edge)
while TRUE do
    Compute G_f
    Find an s − t path P in G_f                                    ▷ Using e.g. DFS
    if no such path then
        return f
    else
        f ← AUGMENT(P, f)
```

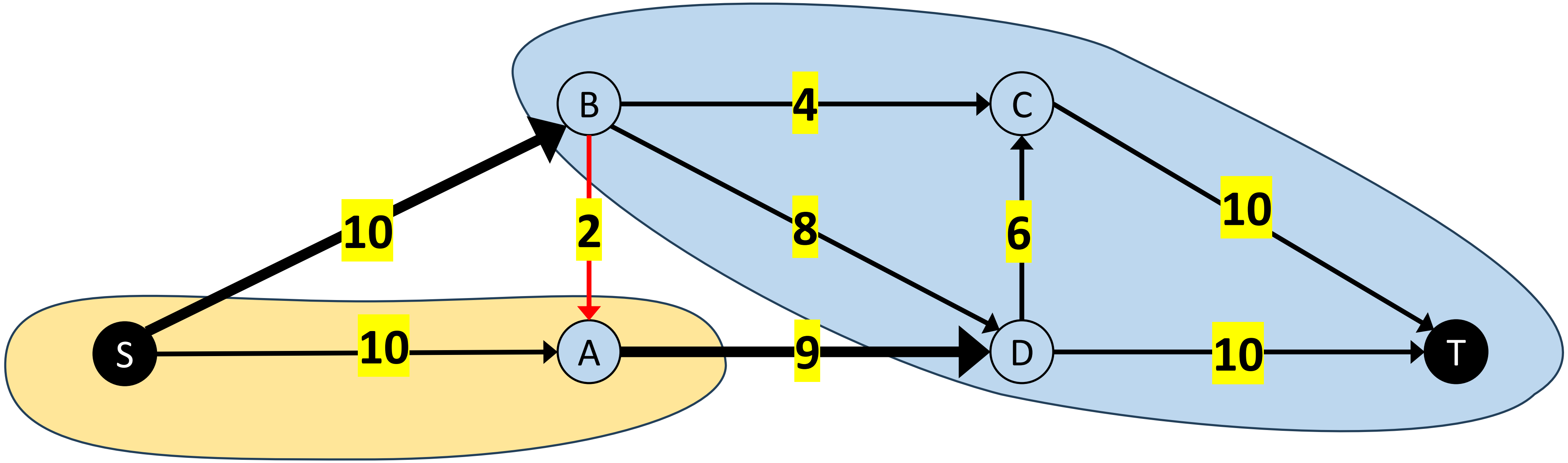
The Ford-Fulkerson - Demo



The Ford-Fulkerson - Demo



The Ford-Fulkerson - Demo

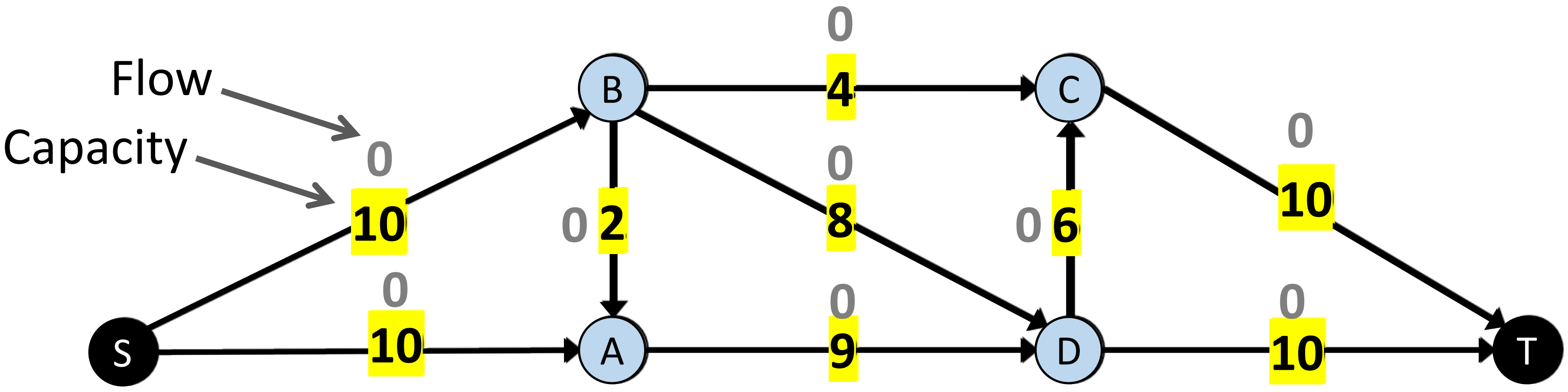


Min-Cut = 19

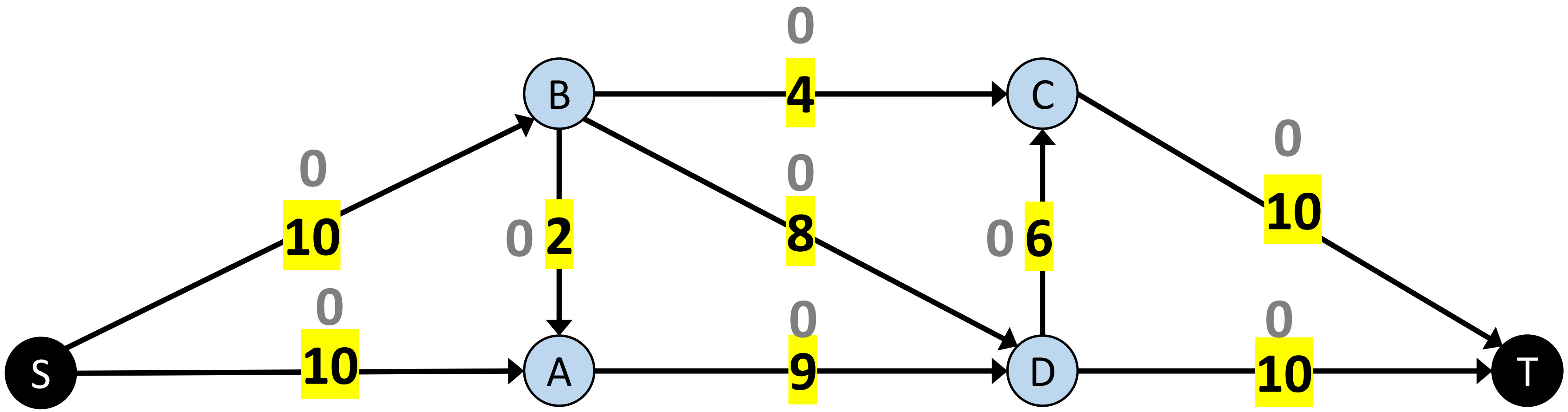
or

Max-flow = 19

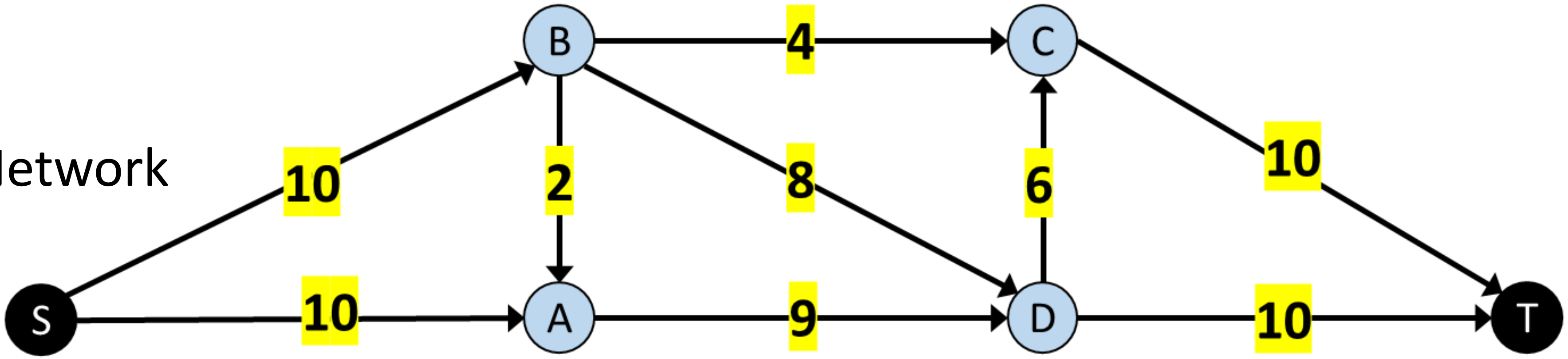
The Ford-Fulkerson - Demo



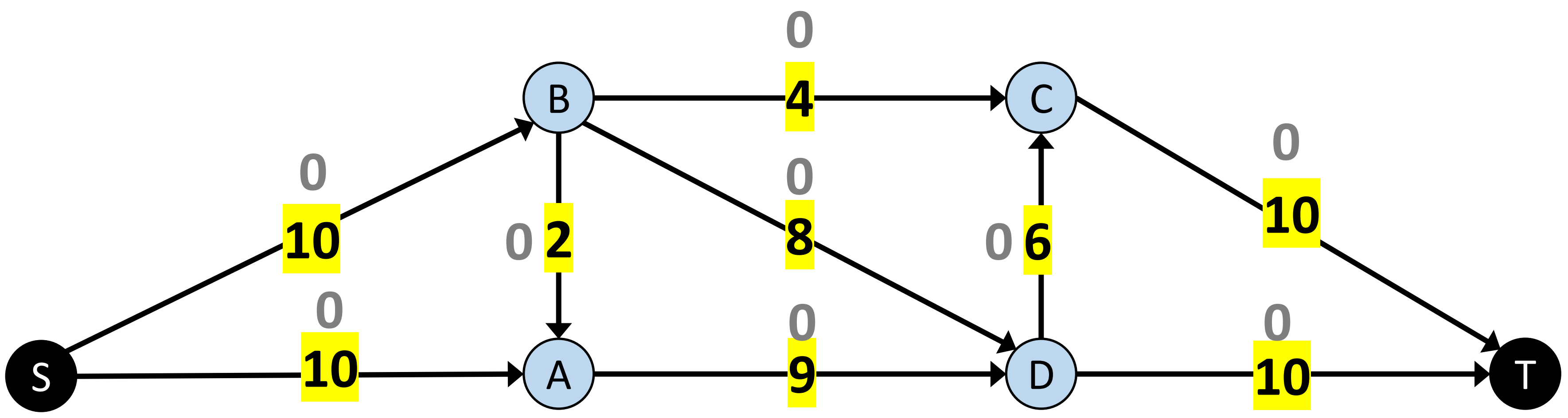
The Ford-Fulkerson - Demo



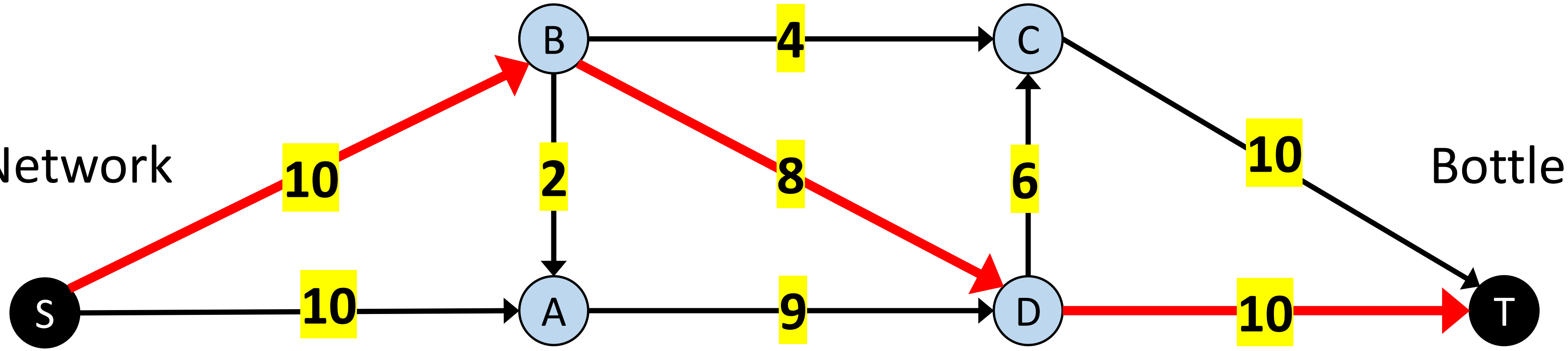
Residual Network



The Ford-Fulkerson - Demo

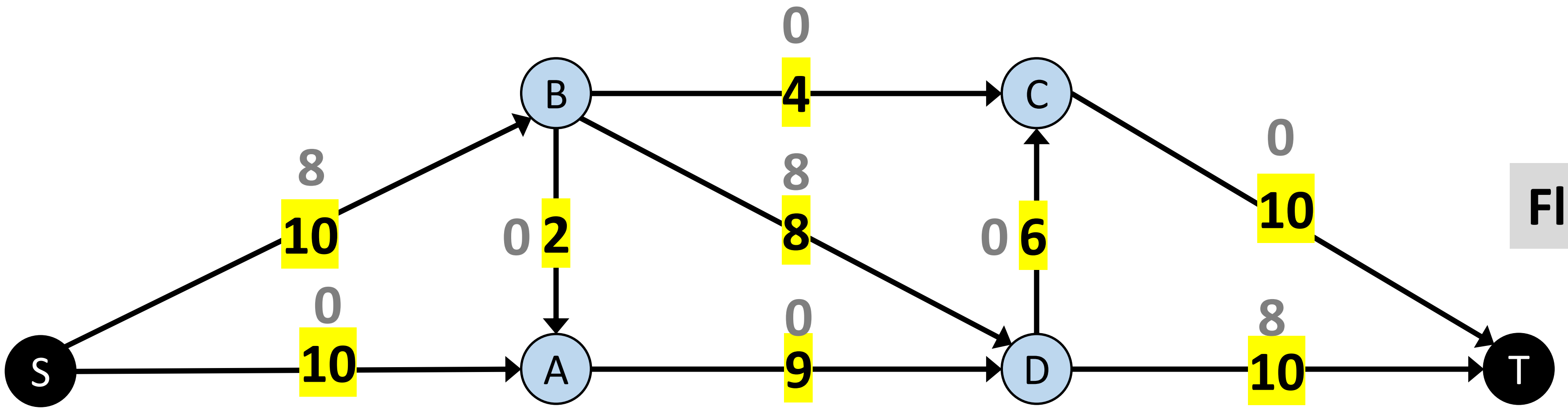


Residual Network



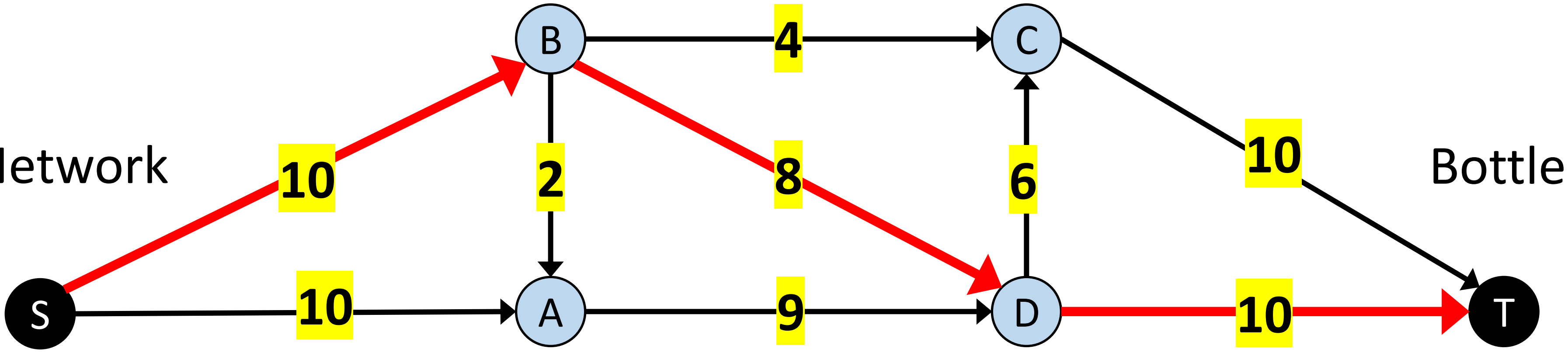
Bottleneck (P) = 8

The Ford-Fulkerson - Demo



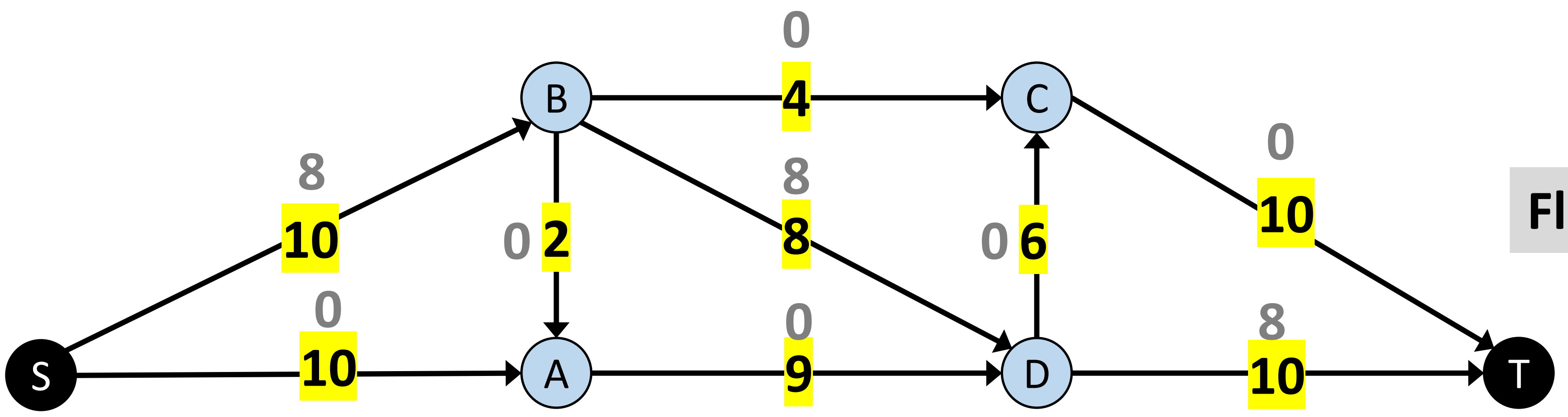
Flow Value = 8

Residual Network



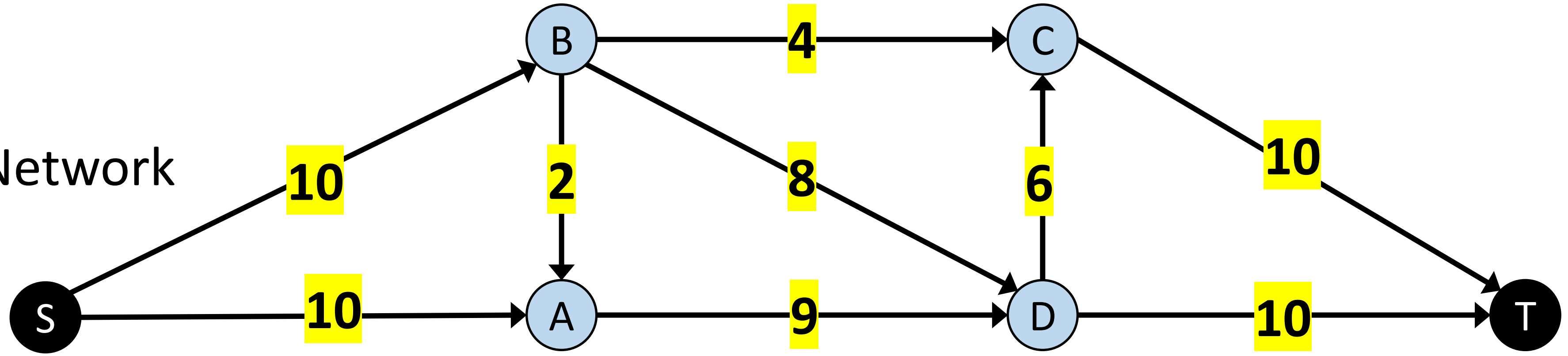
Bottleneck (P) = 8

The Ford-Fulkerson - Demo

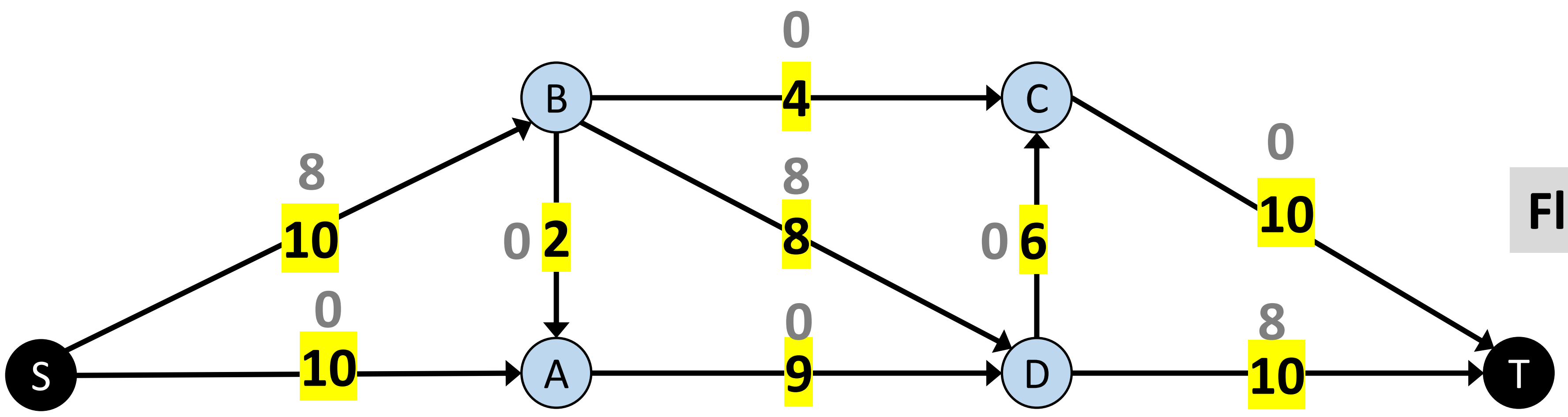


Flow Value = 8

Residual Network

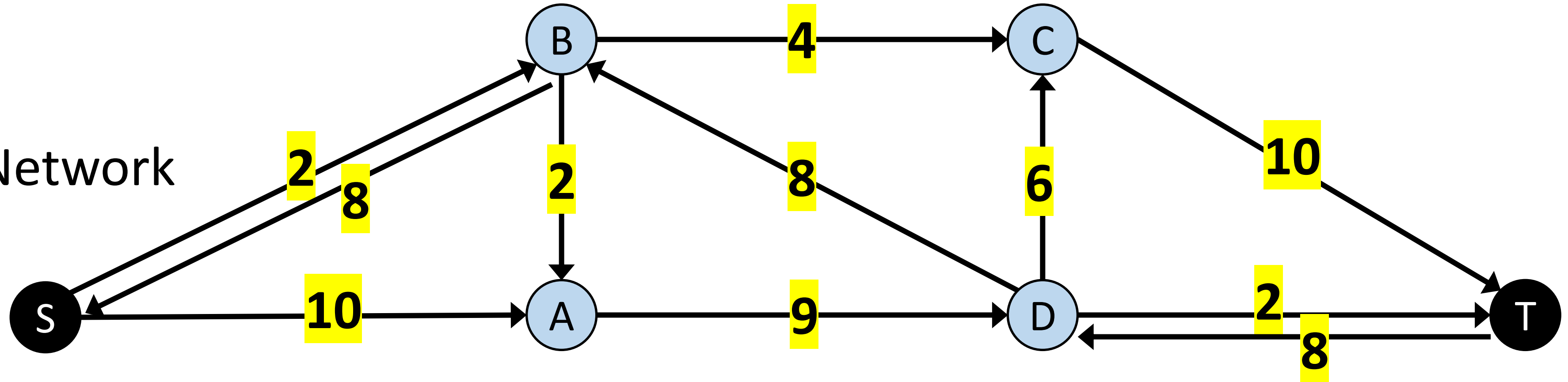


The Ford-Fulkerson - Demo

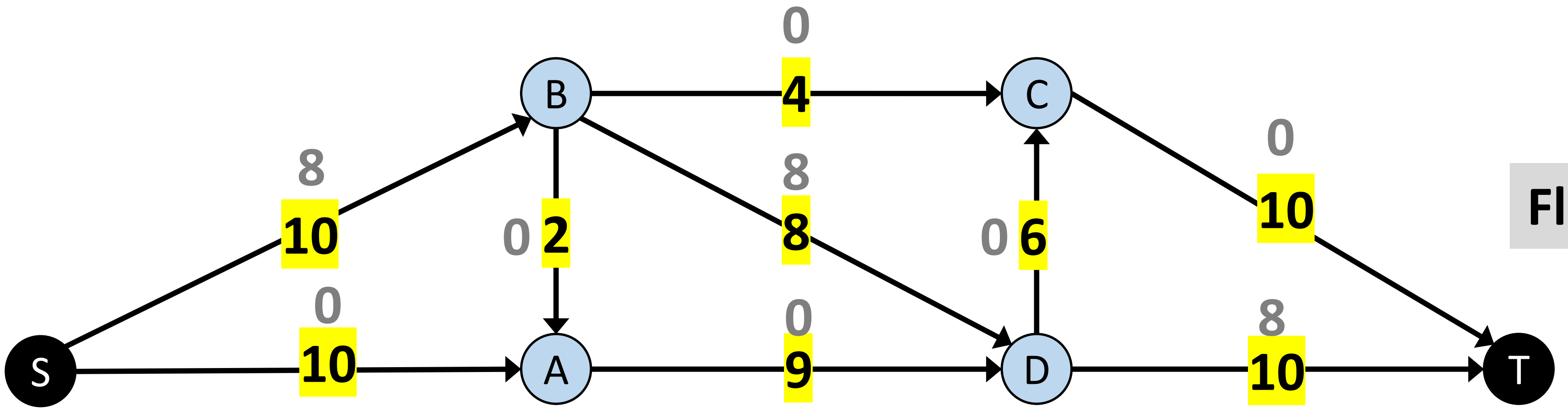


Flow Value = 8

Residual Network

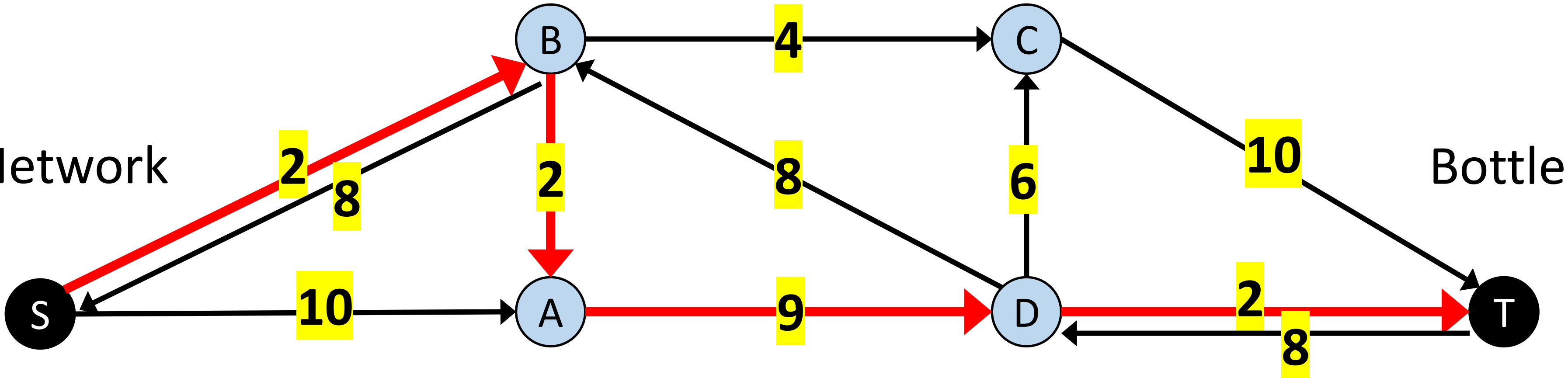


The Ford-Fulkerson - Demo



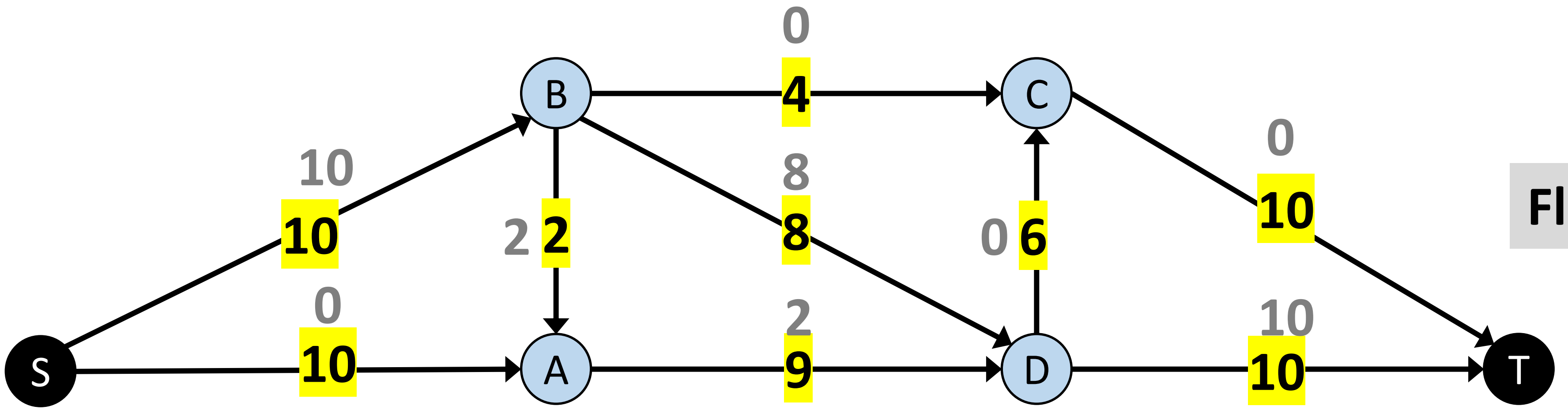
Flow Value = 8

Residual Network



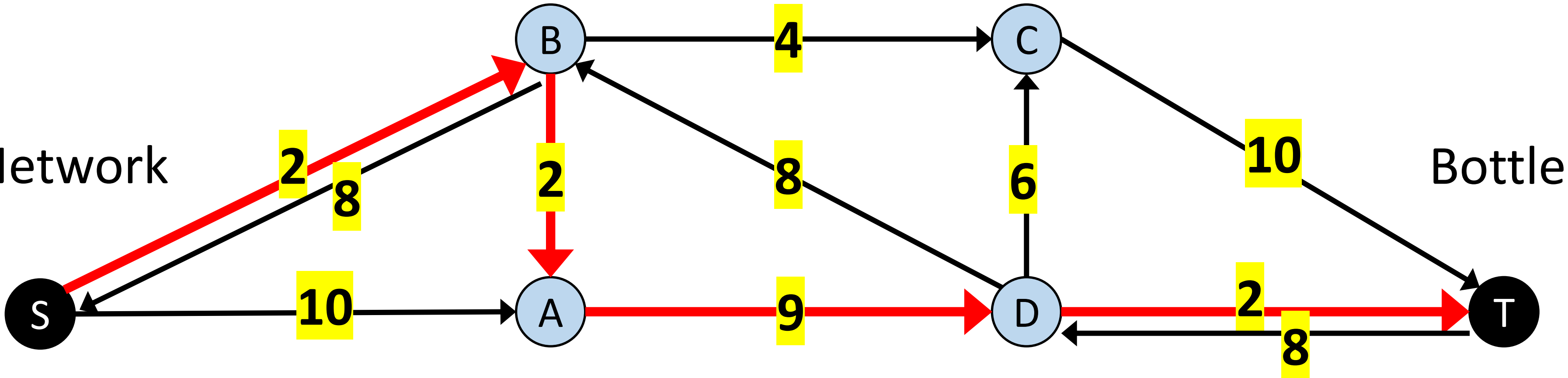
Bottleneck (P) = 2

The Ford-Fulkerson - Demo



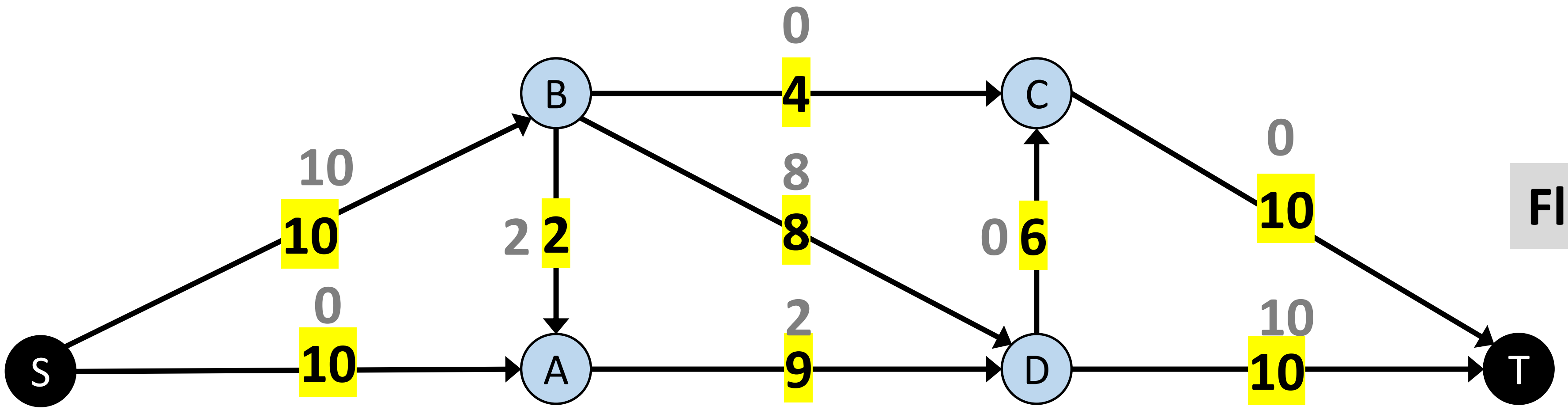
Flow Value = 10

Residual Network



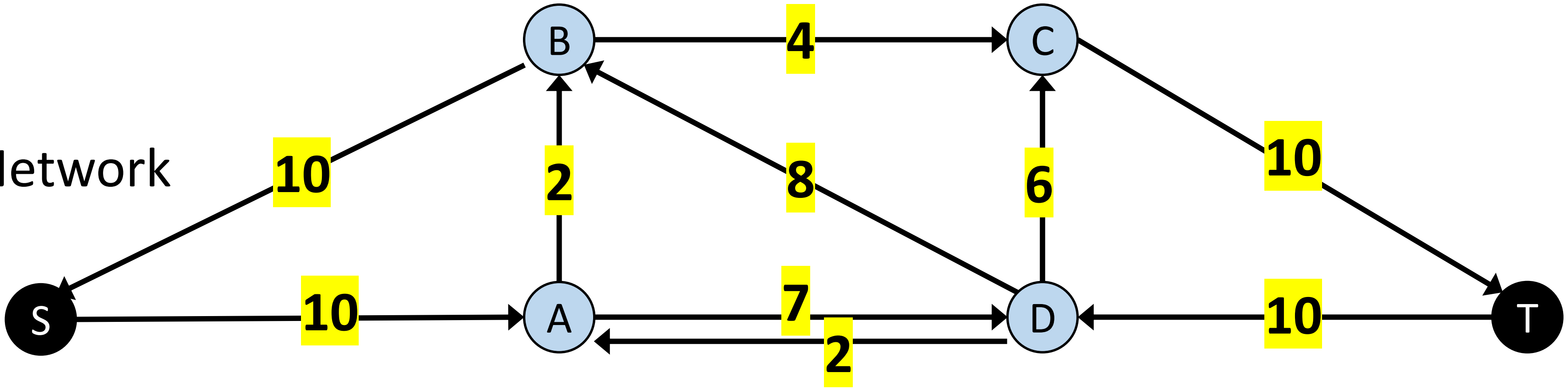
Bottleneck (P) = 2

The Ford-Fulkerson - Demo

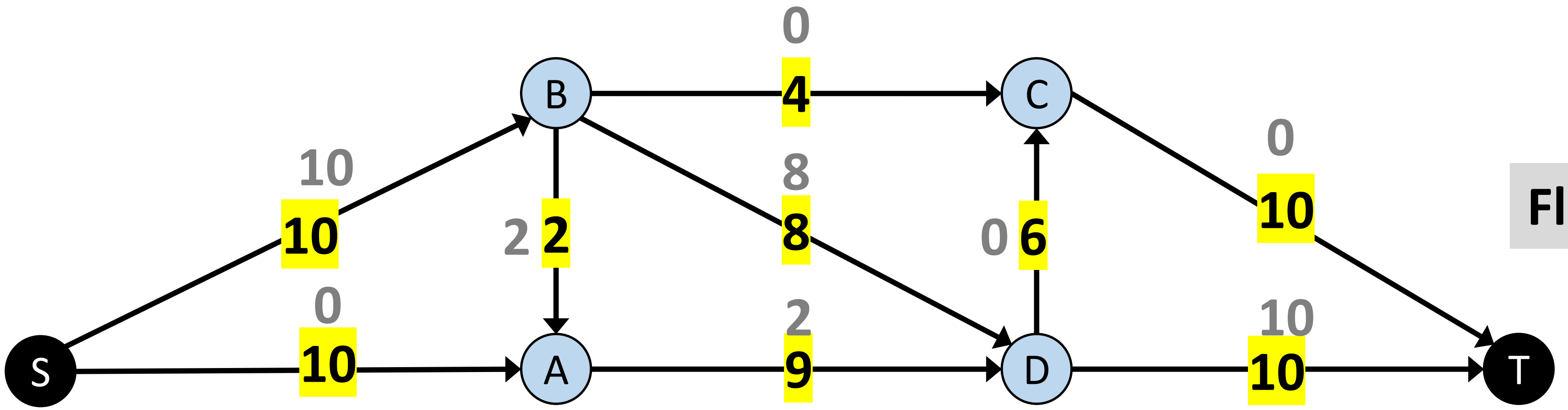


Flow Value = 10

Residual Network

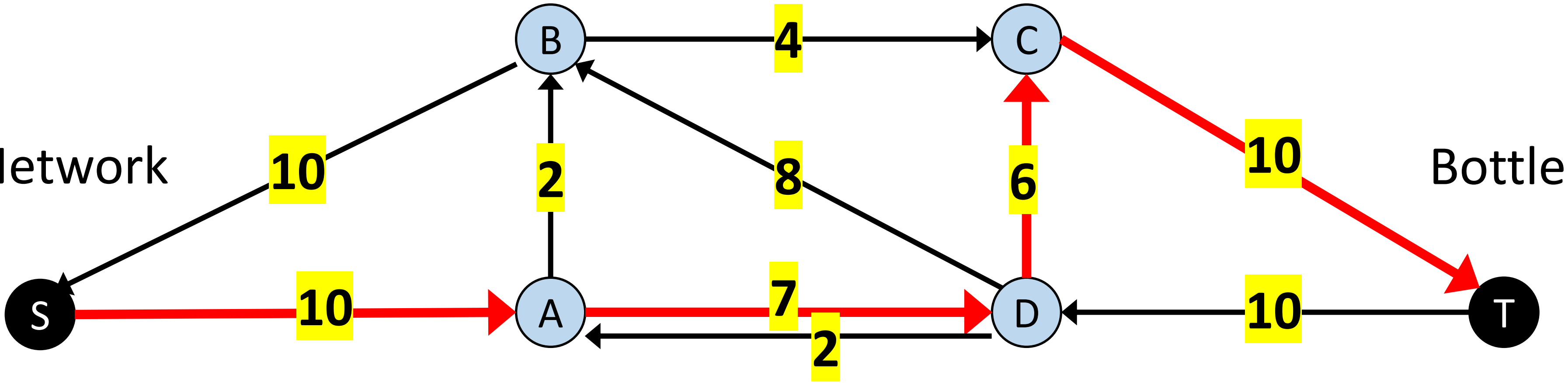


The Ford-Fulkerson - Demo



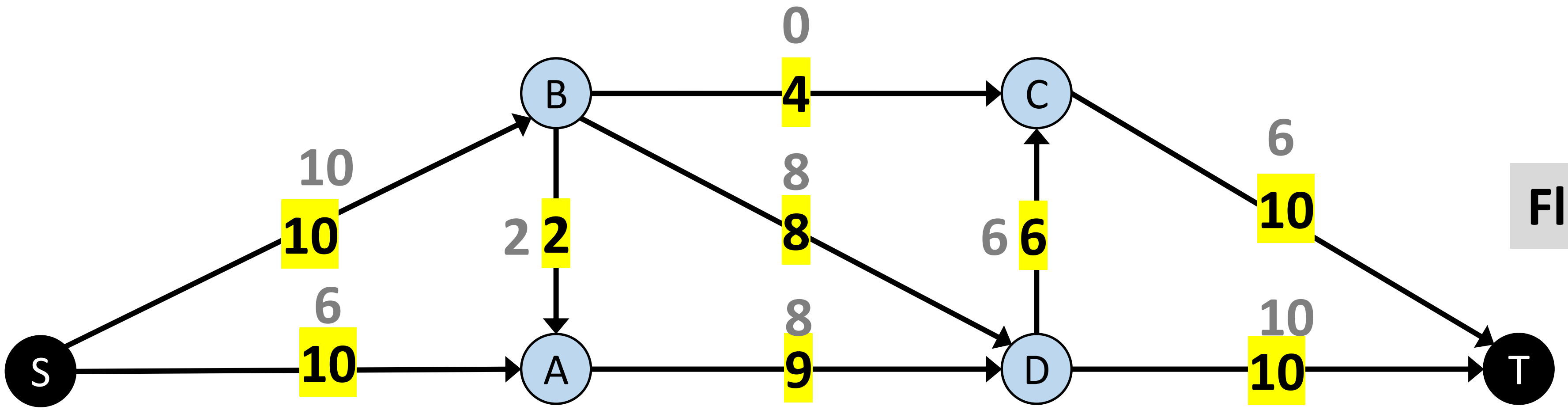
Flow Value = 10

Residual Network



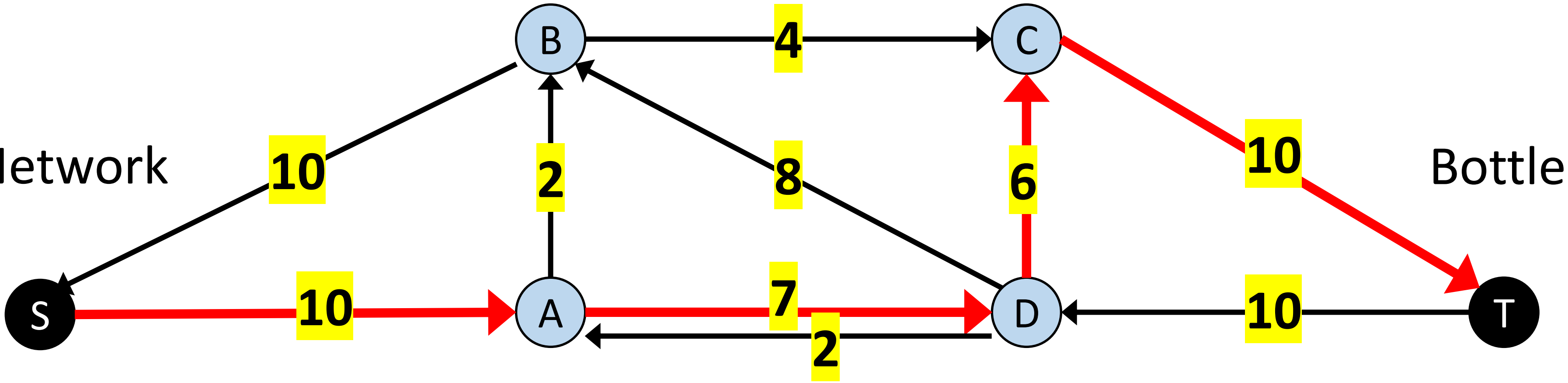
Bottleneck (P) = 6

The Ford-Fulkerson - Demo



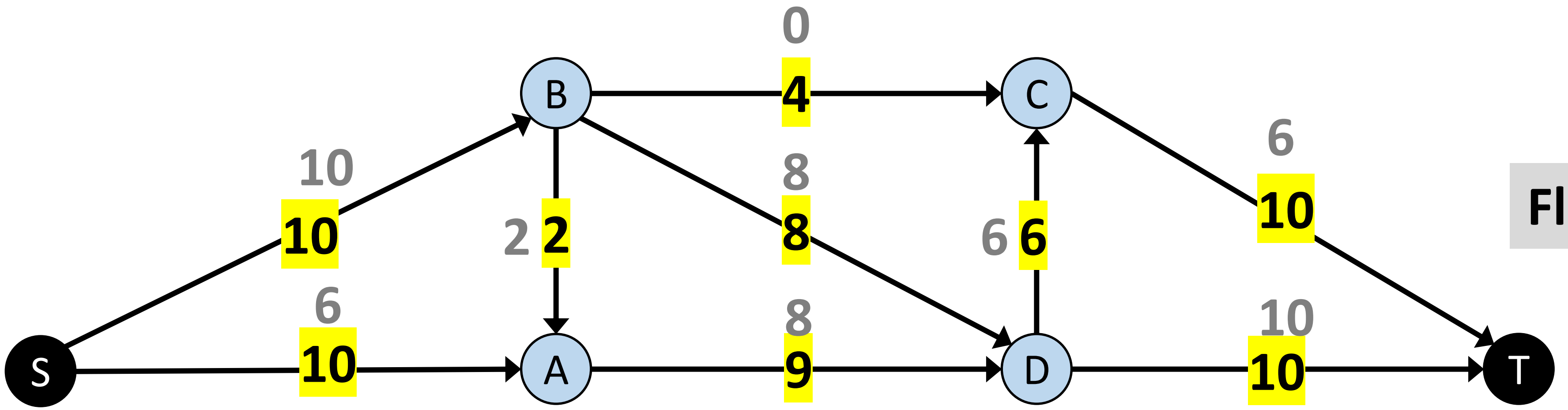
Flow Value = 16

Residual Network



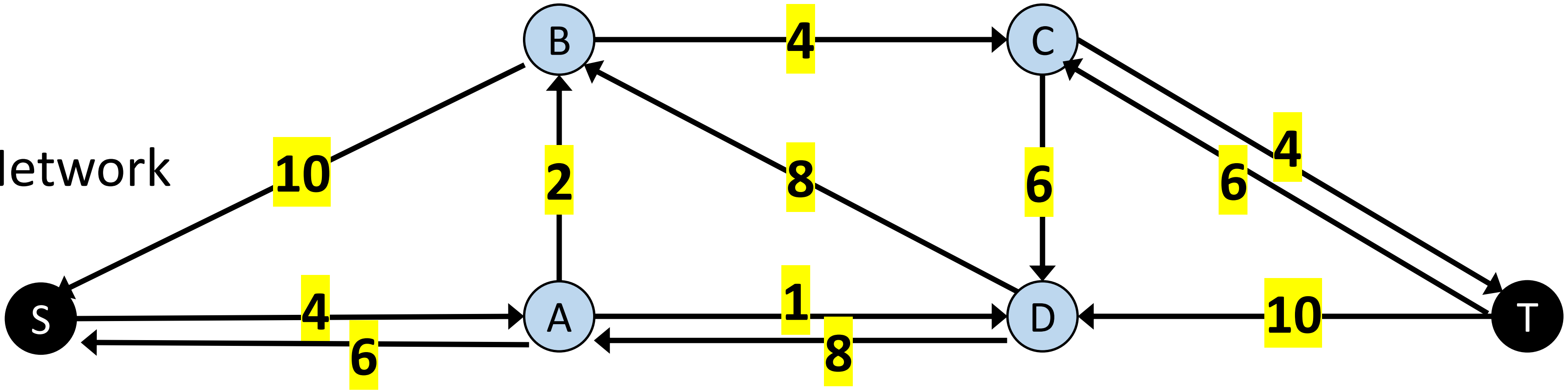
Bottleneck (P) = 6

The Ford-Fulkerson - Demo

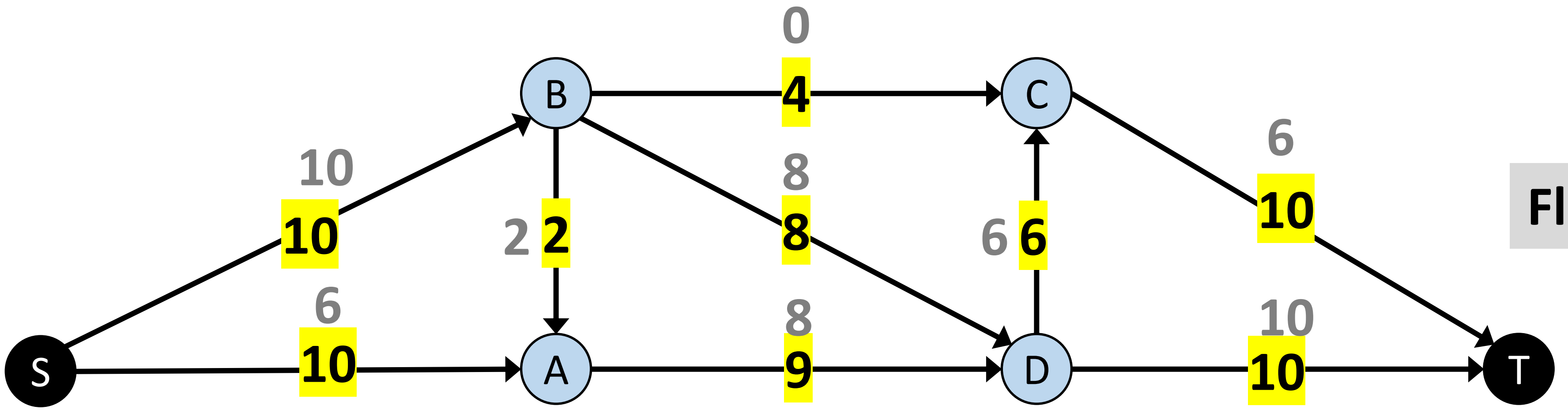


Flow Value = 16

Residual Network

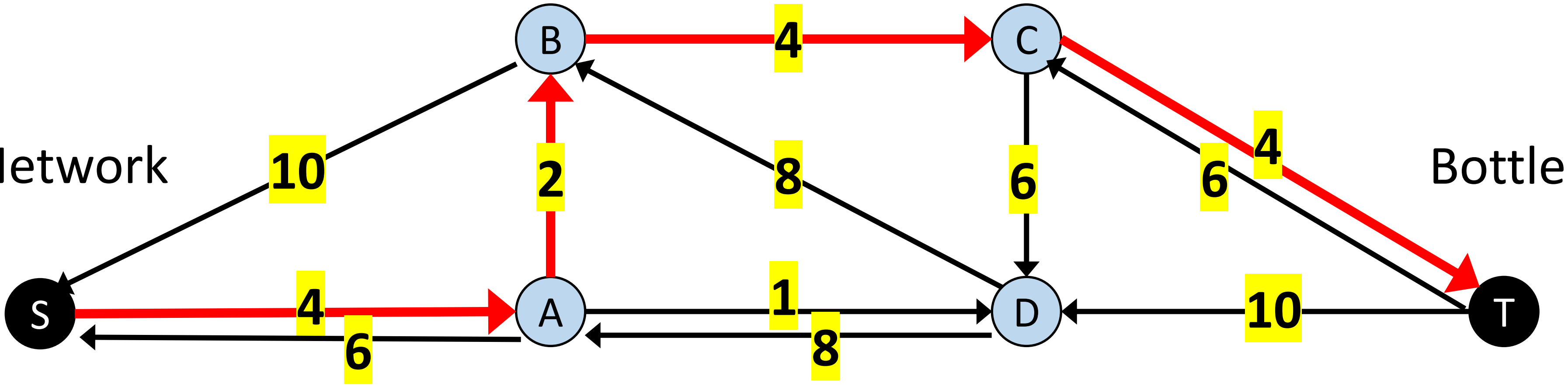


The Ford-Fulkerson - Demo



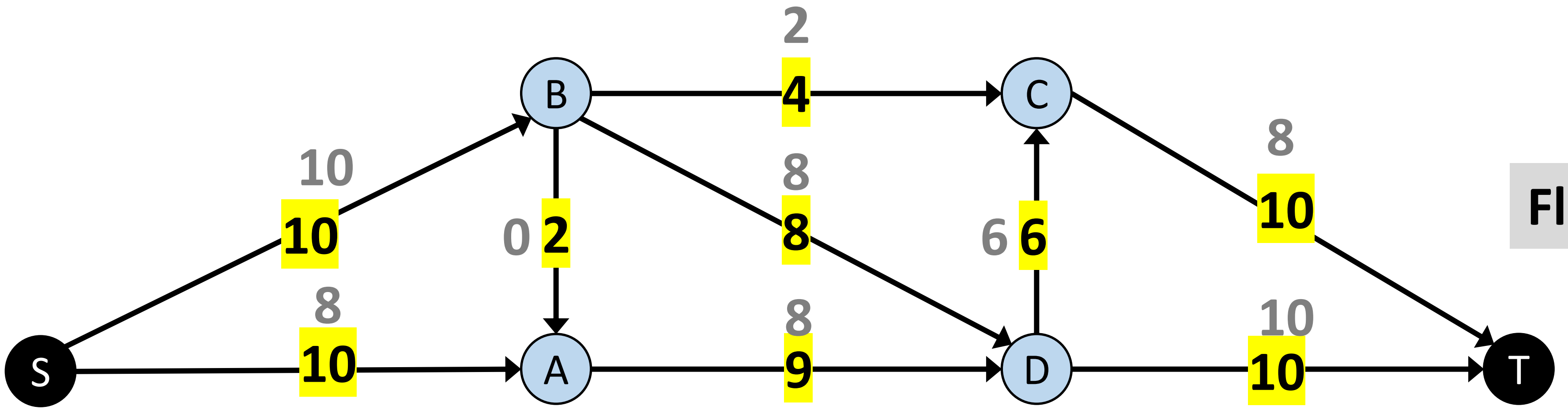
Flow Value = 16

Residual Network



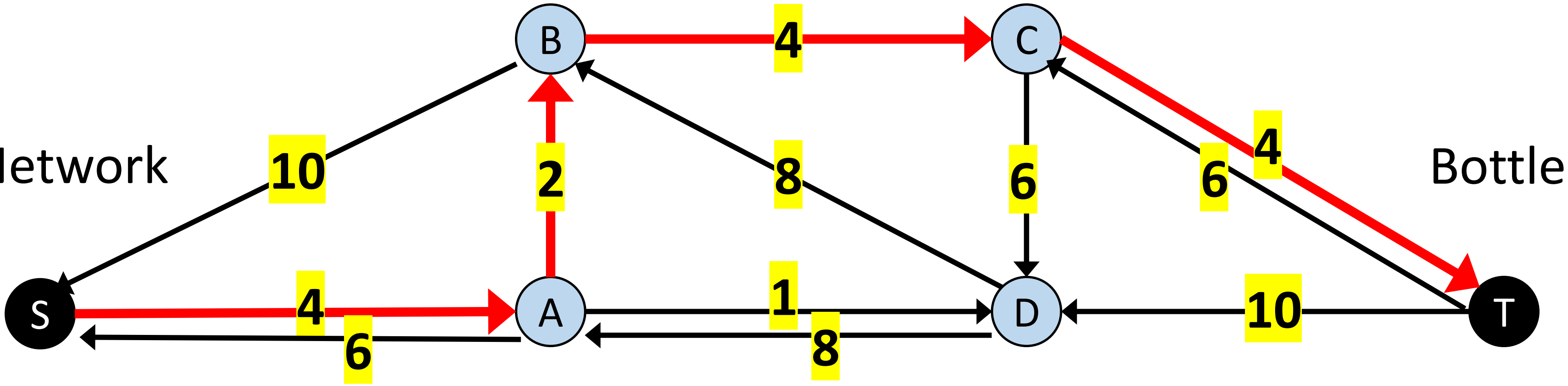
Bottleneck (P) = 2

The Ford-Fulkerson - Demo



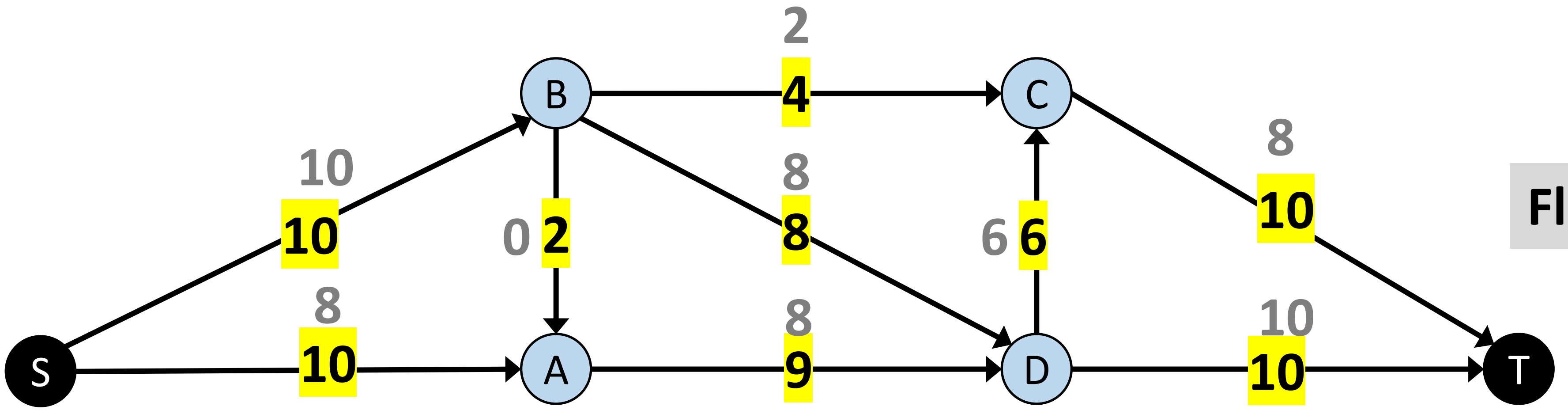
Flow Value = 18

Residual Network



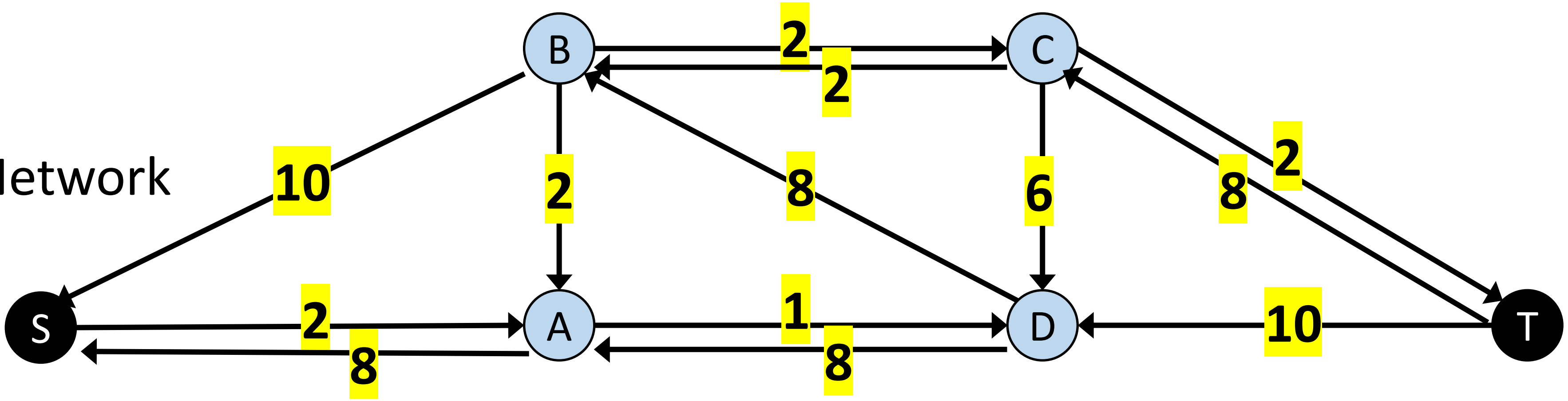
Bottleneck (P) = 2

The Ford-Fulkerson - Demo

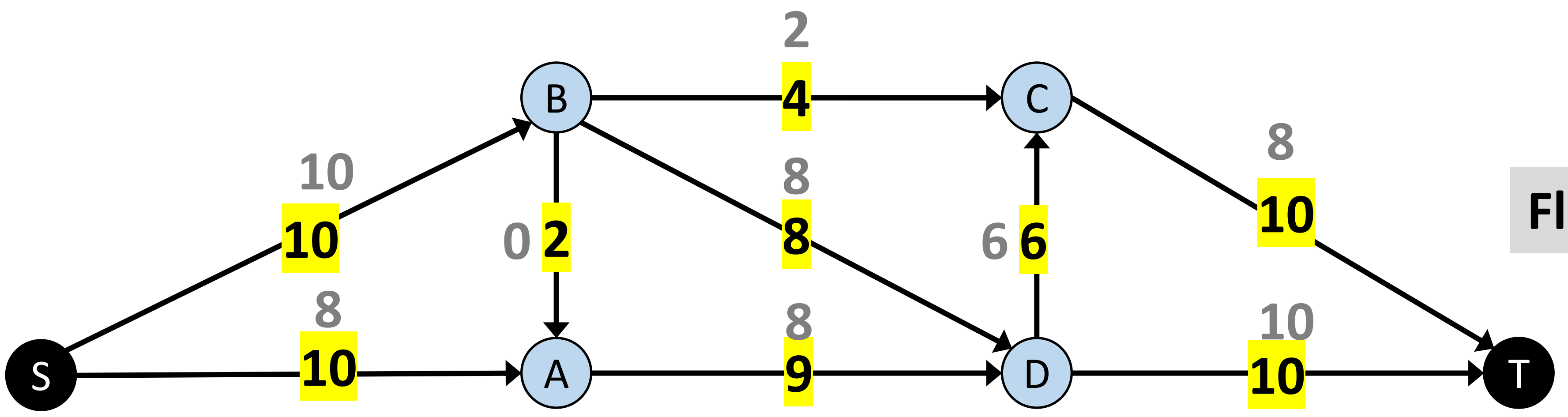


Flow Value = 18

Residual Network

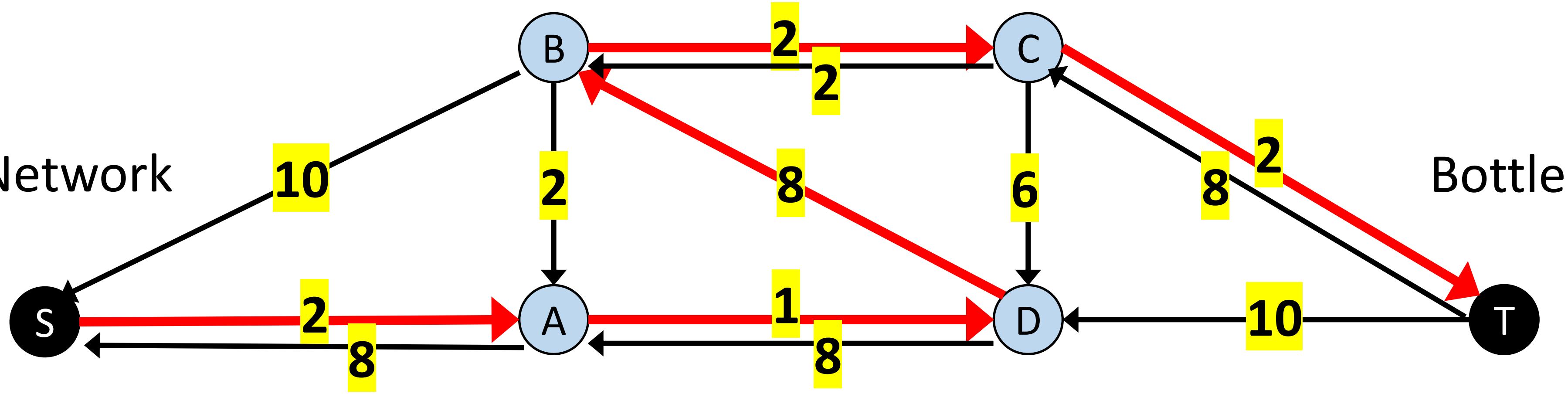


The Ford-Fulkerson - Demo



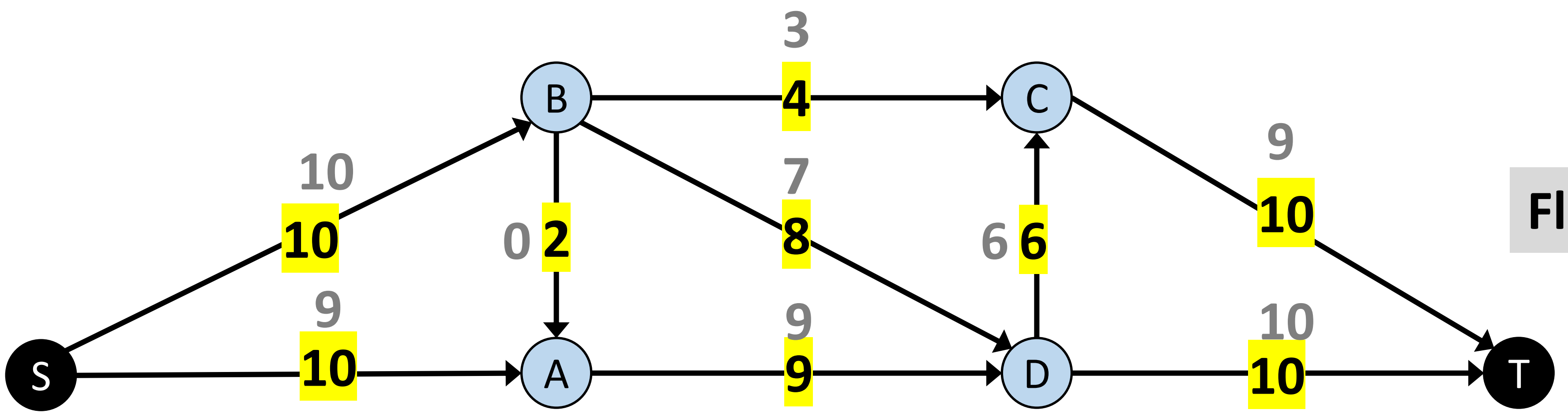
Flow Value = 18

Residual Network



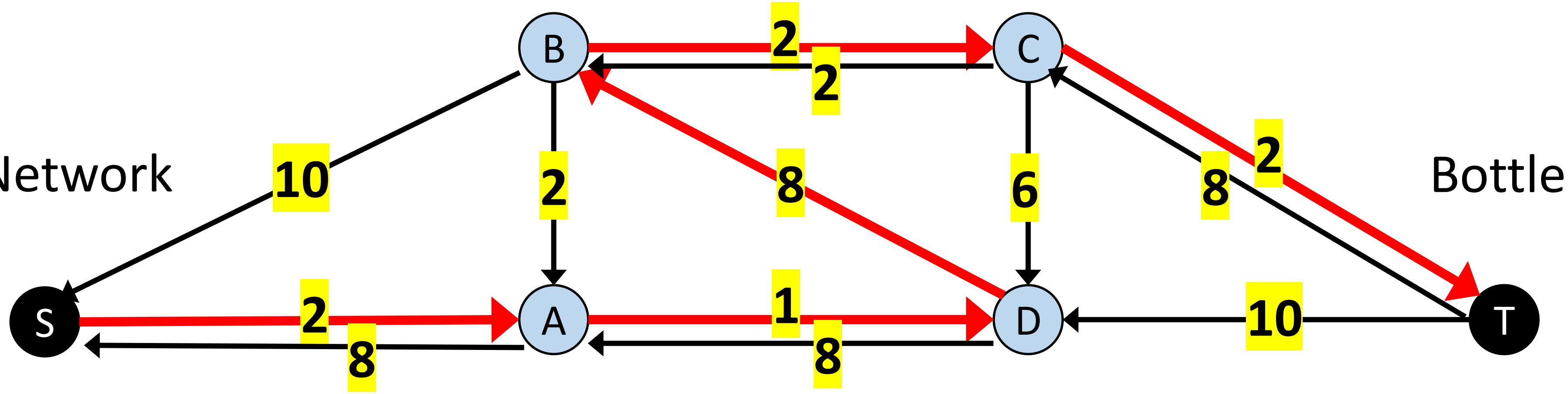
Bottleneck (P) = 1

The Ford-Fulkerson - Demo



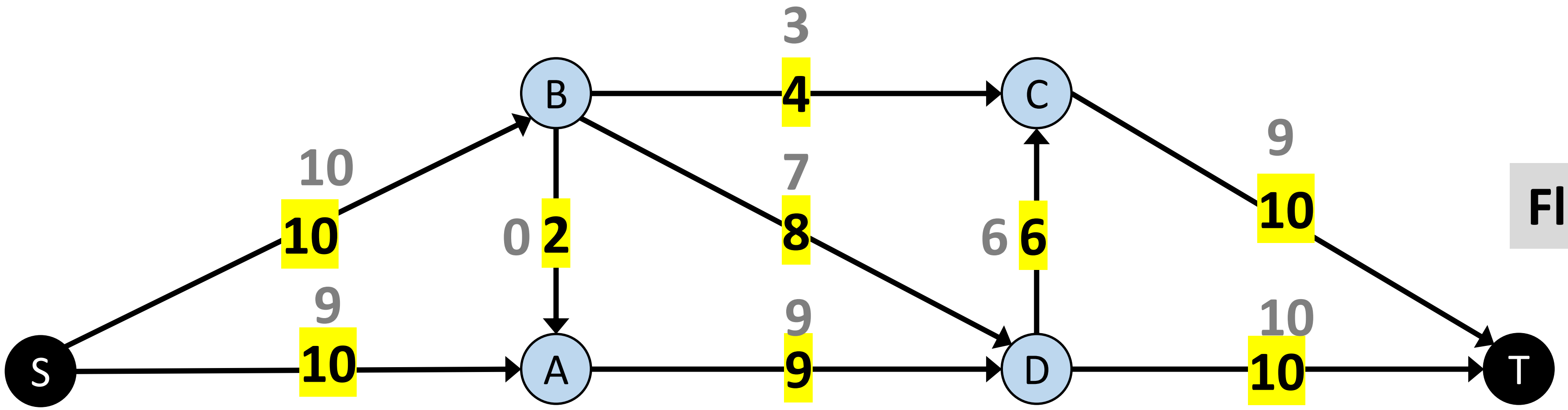
Flow Value = 19

Residual Network



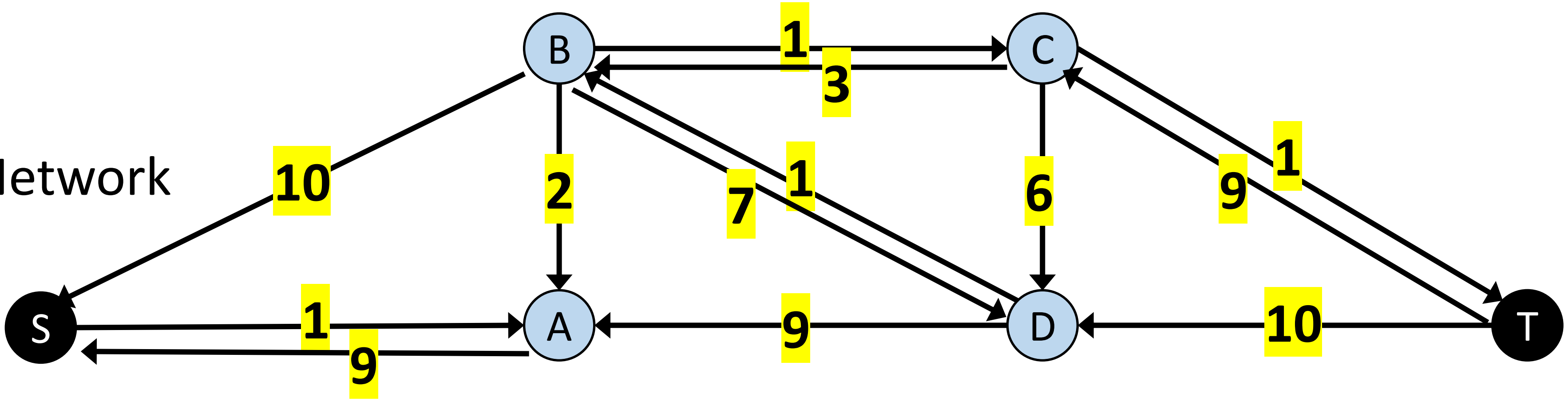
Bottleneck (P) = 1

The Ford-Fulkerson - Demo



Flow Value = 19

Residual Network



The Ford-Fulkerson Algorithm – Time Complexity

Given a flow network G with source s and t

Algorithm Ford-Fulkerson Algorithm (G)

$f \leftarrow 0$ $\triangleright$ Initialize to a (valid) flow of size 0 (on every edge)

while TRUE **do** O(f)

Compute G_f O(V+E)

Find an $s - t$ path P in G_f O(V+E) $\triangleright$ Using e.g. DFS

if no such path **then**

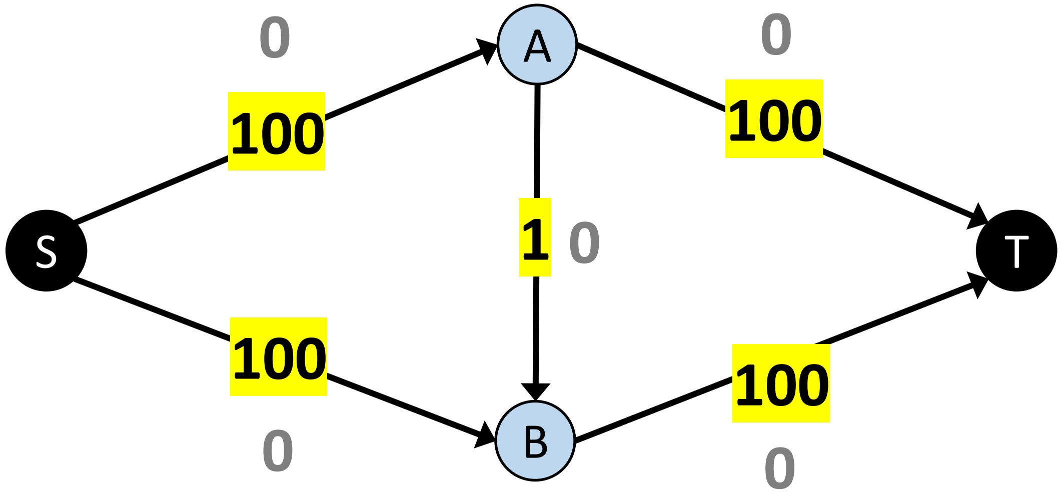
return f

else

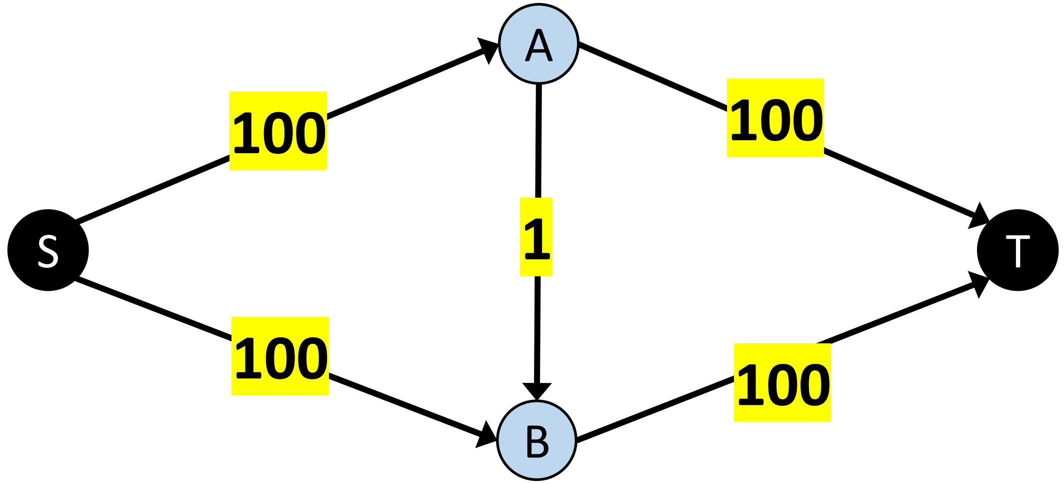
$f \leftarrow$ AUGMENT(P, f) O(E)

O(f E) when E >= V

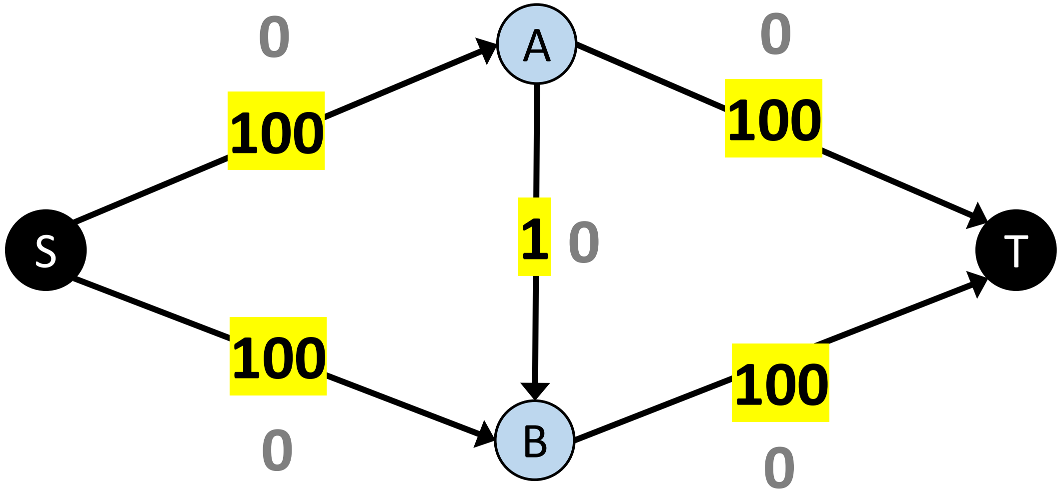
How O(*f* E)? Why O(*f* E)? When O(*f* E)?



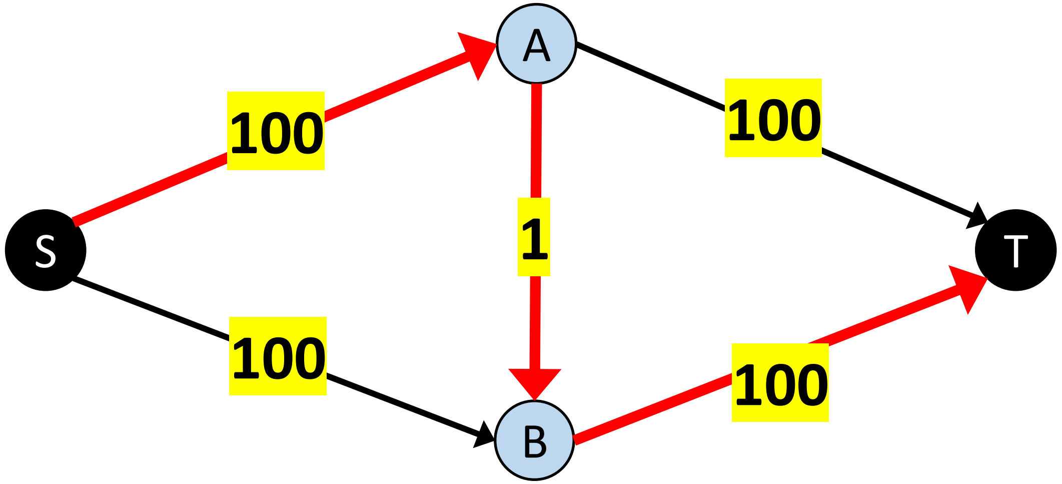
Residual Network



How O(*f* E)? Why O(*f* E)? When O(*f* E)?

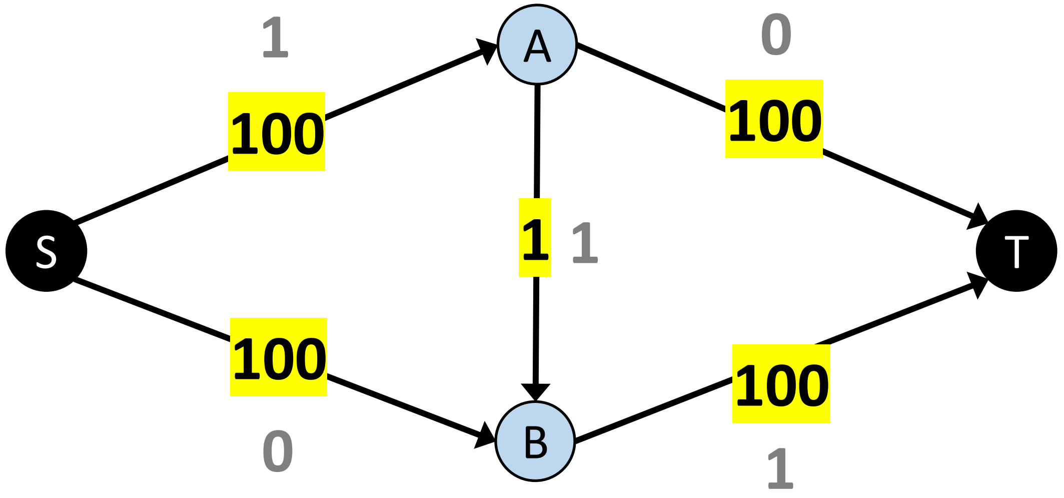


Residual Network



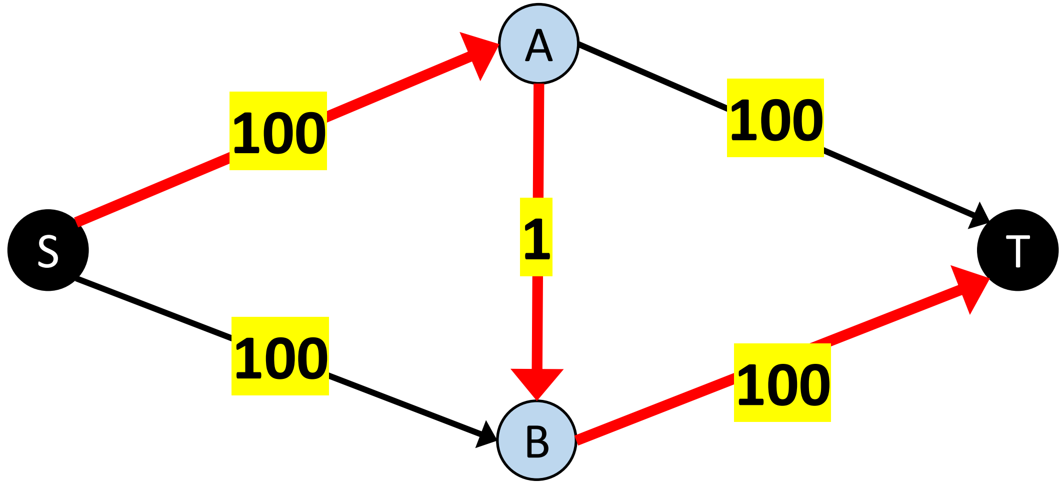
Bottleneck (P) = 1

How O(f E)?   Why O(f E)?   When O(f E)?



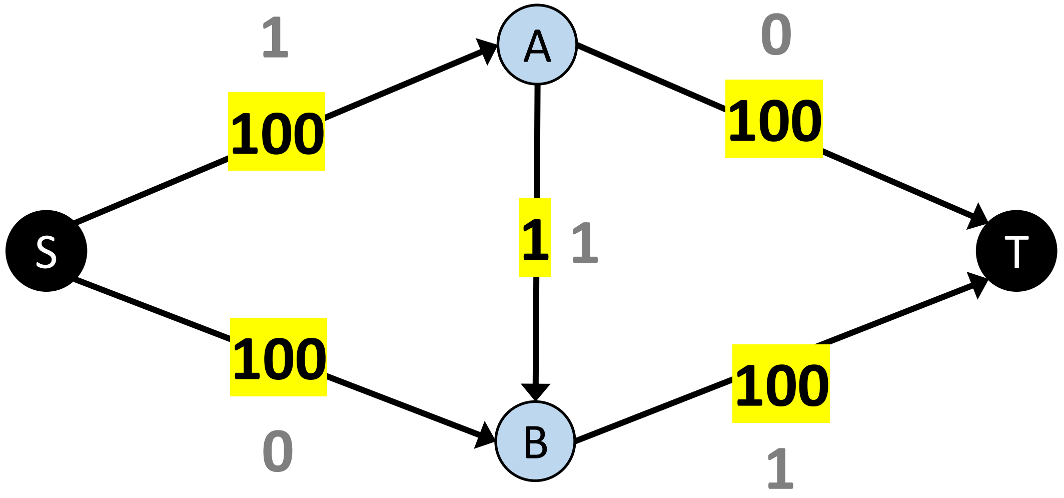
Flow Value = 1

Residual Network



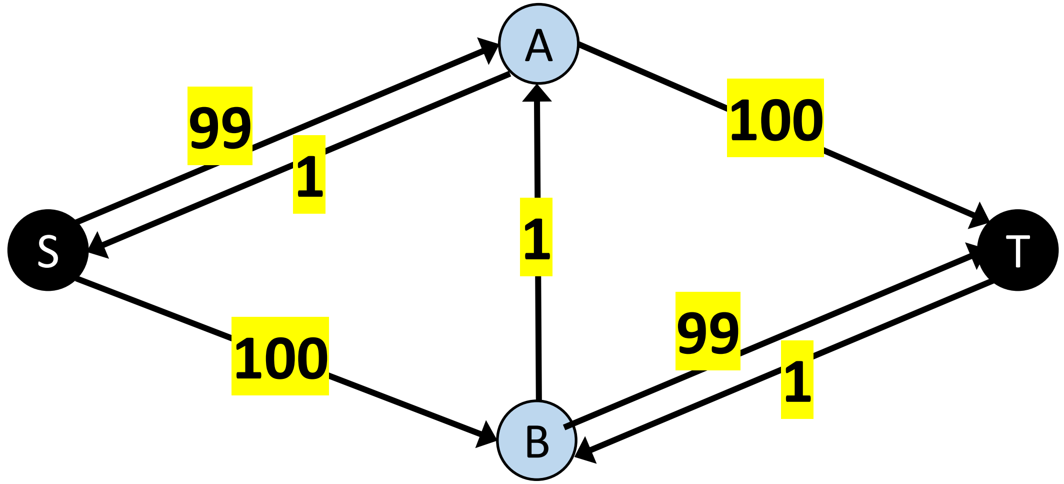
Bottleneck (P) = 1

How O(*f* E)? Why O(*f* E)? When O(*f* E)?

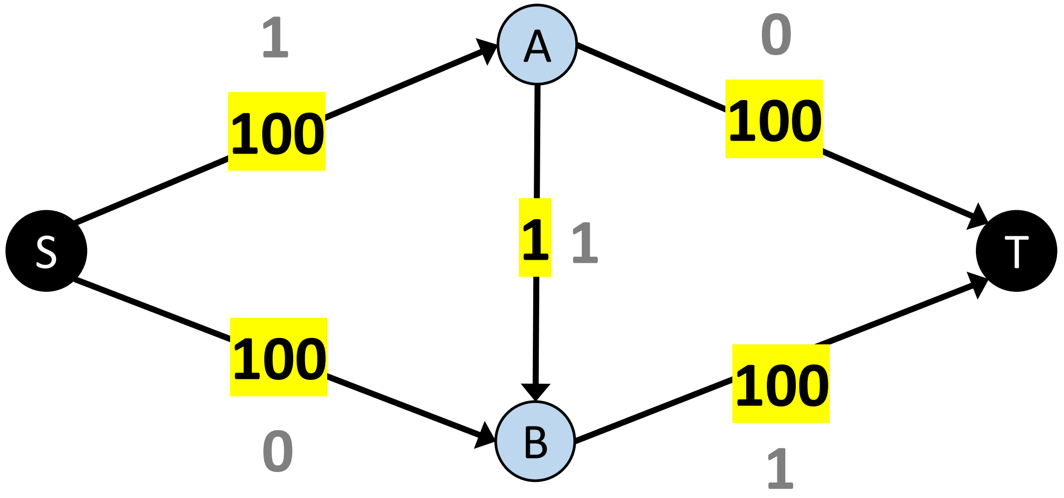


Flow Value = 1

Residual Network

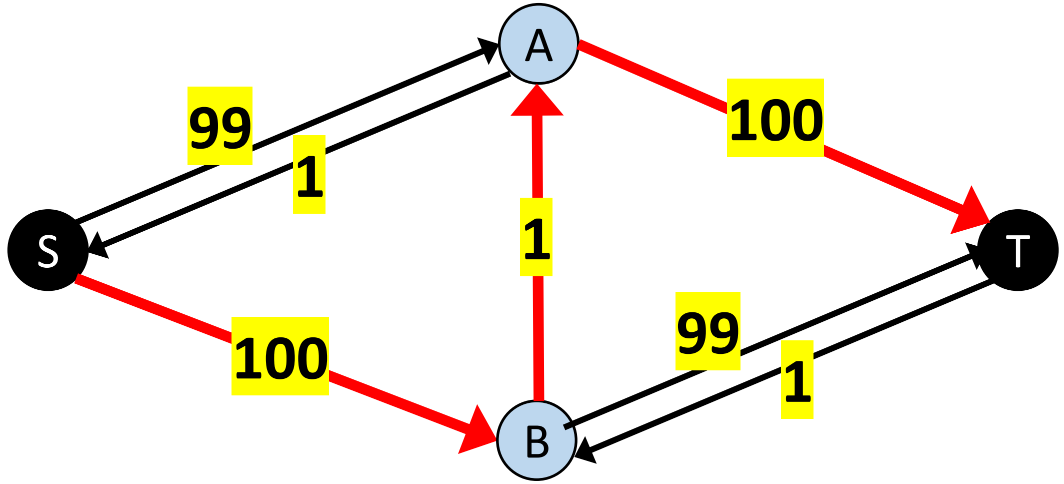


How O(*f* E)? Why O(*f* E)? When O(*f* E)?



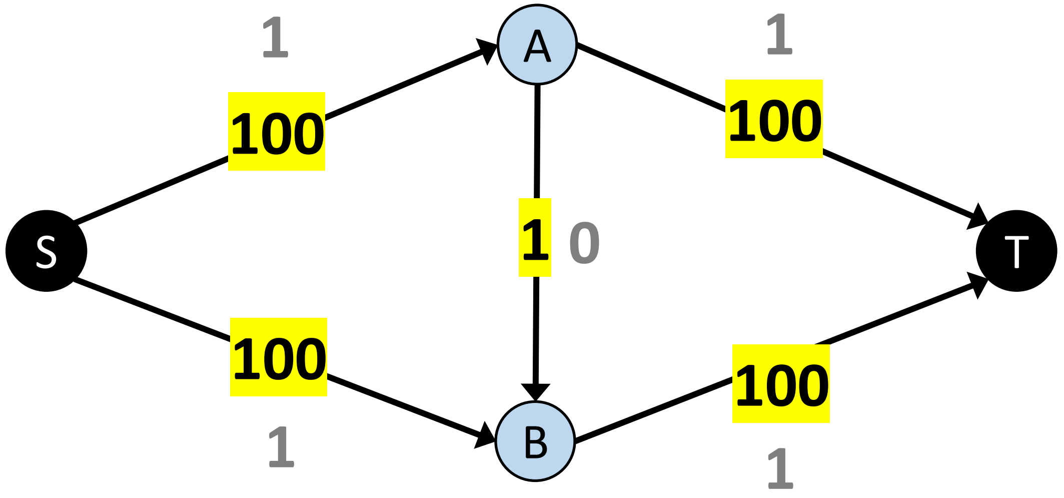
Flow Value = 1

Residual Network



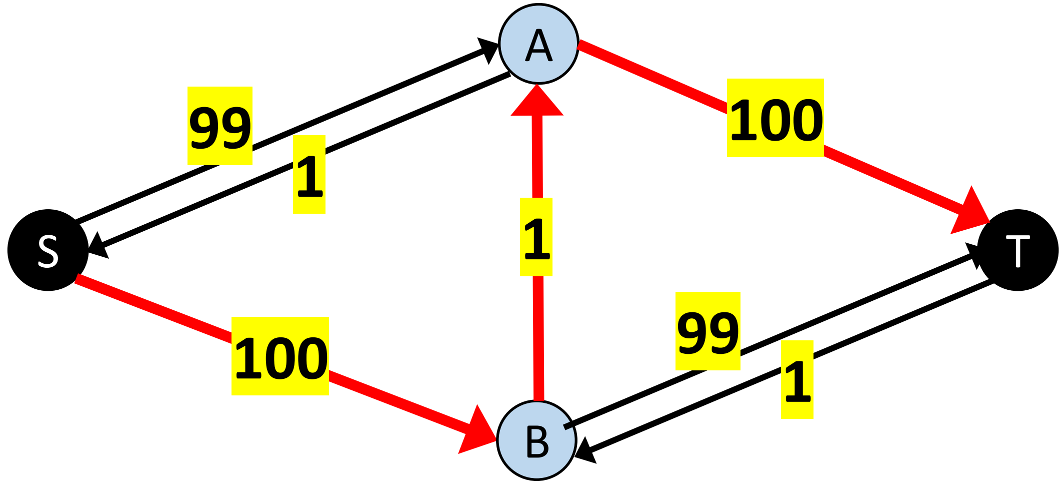
Bottleneck (P) = 1

How O(*f* E)? Why O(*f* E)? When O(*f* E)?



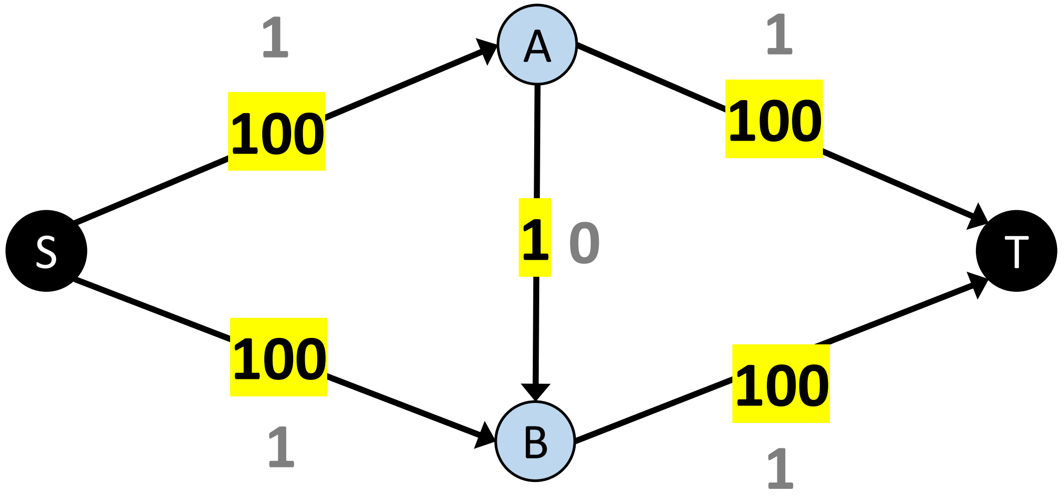
Flow Value = 2

Residual Network



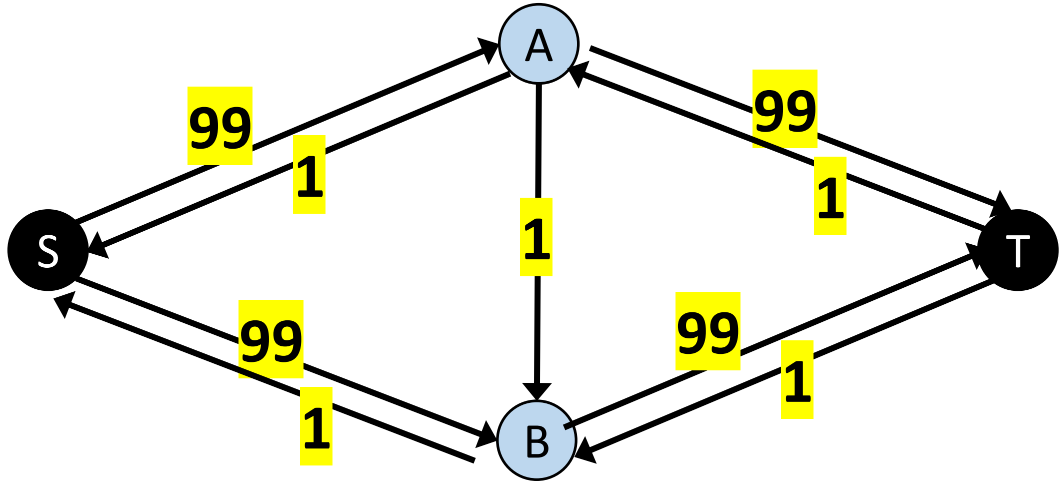
Bottleneck (P) = 1

How O(*f* E)? Why O(*f* E)? When O(*f* E)?

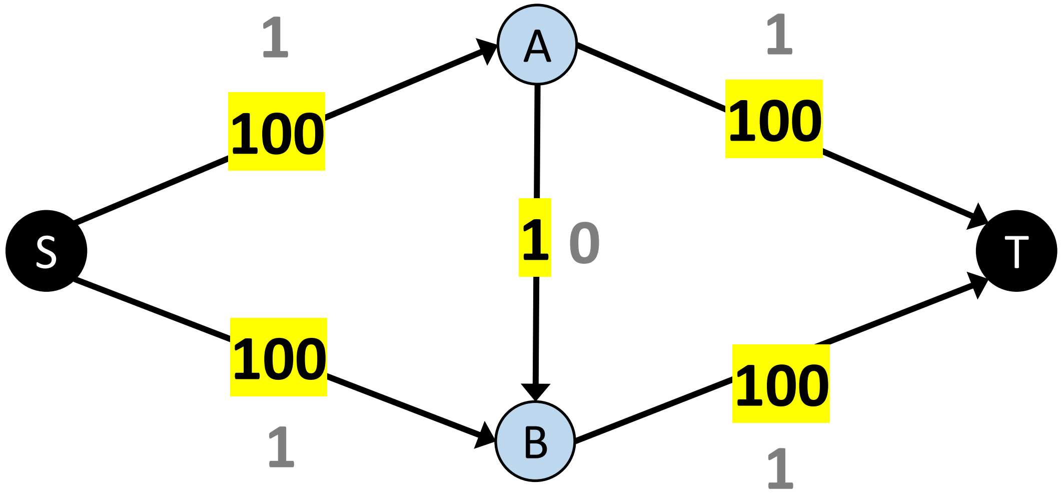


Flow Value = 2

Residual Network

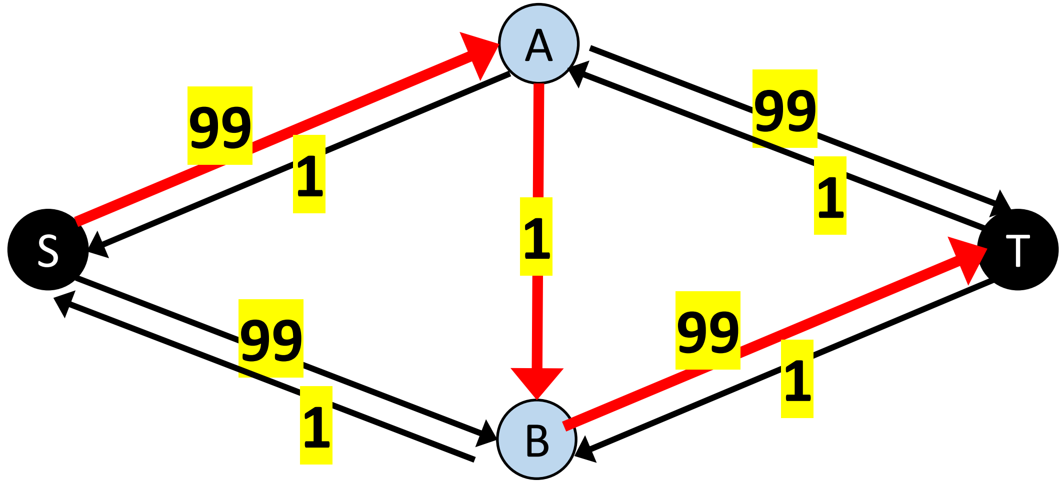


How O(*f* E)? Why O(*f* E)? When O(*f* E)?



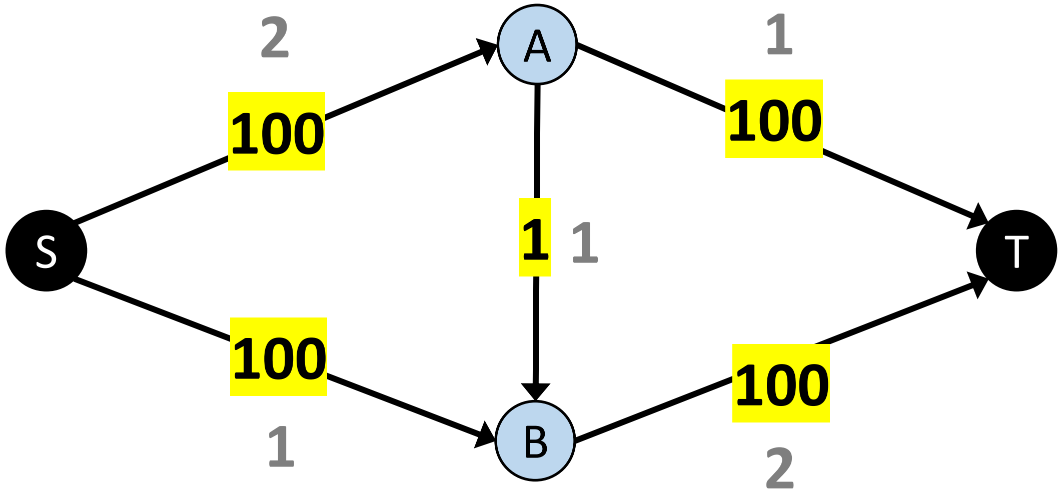
Flow Value = 2

Residual Network



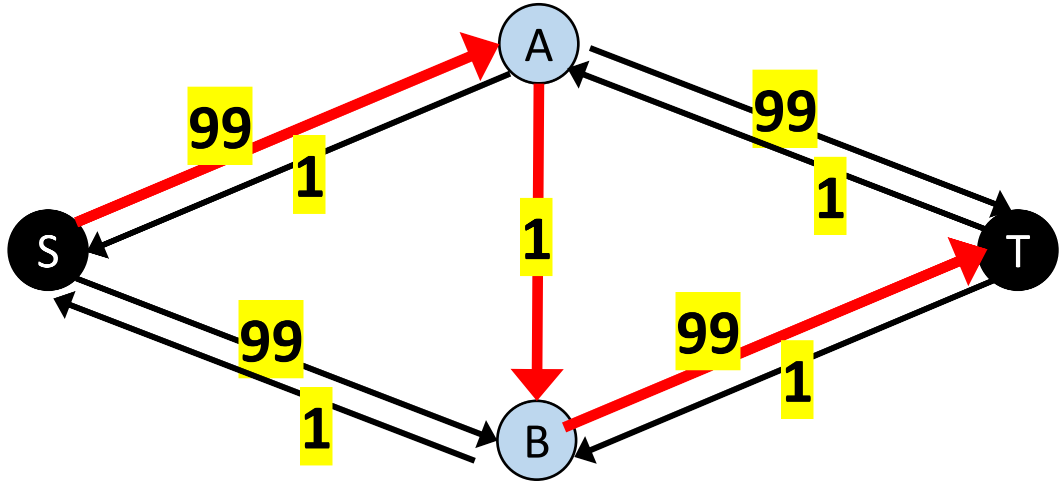
Bottleneck (P) = 1

How O(f E)? Why O(f E)? When O(f E)?



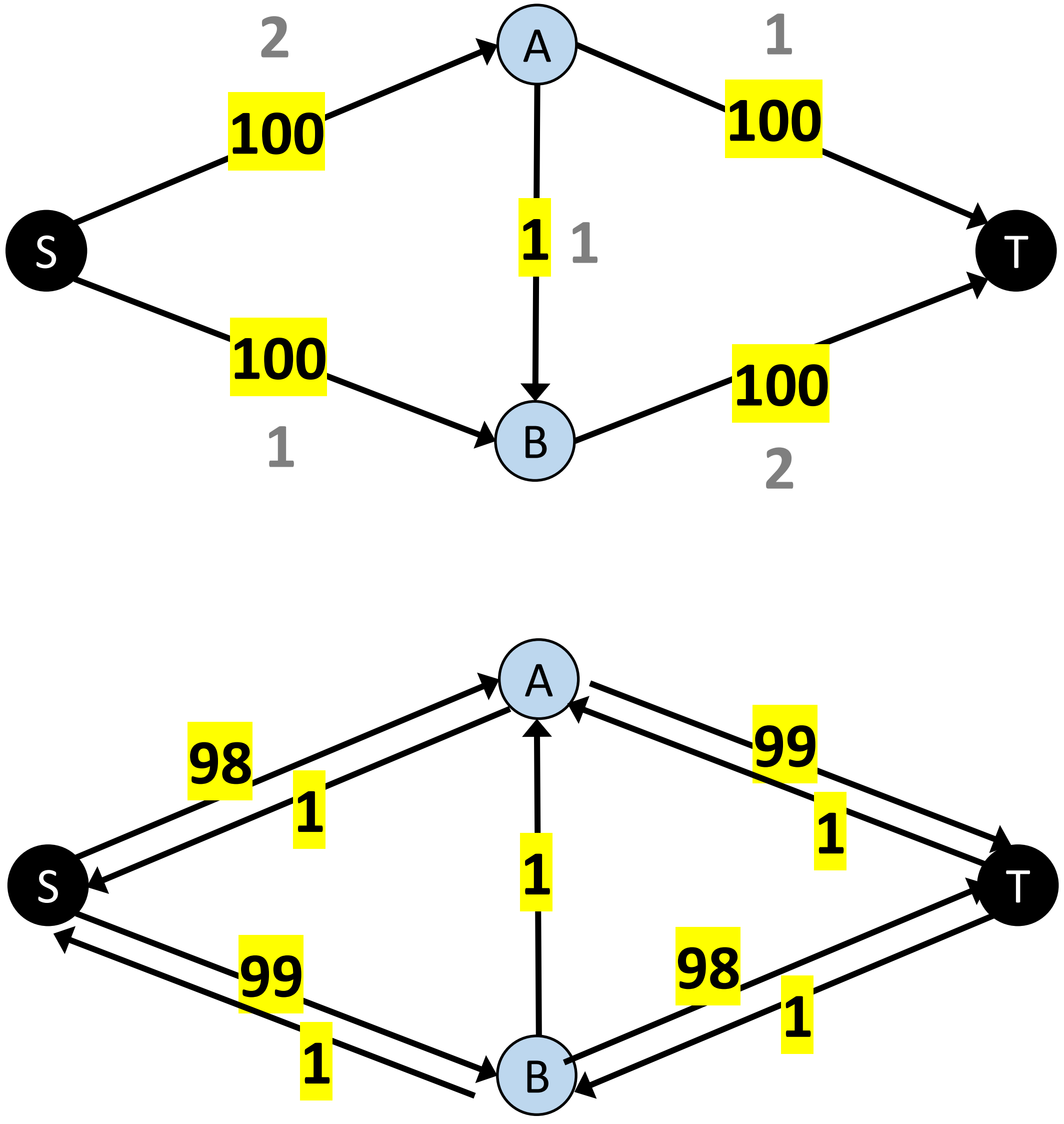
Flow Value = 3

Residual Network



Bottleneck (P) = 1

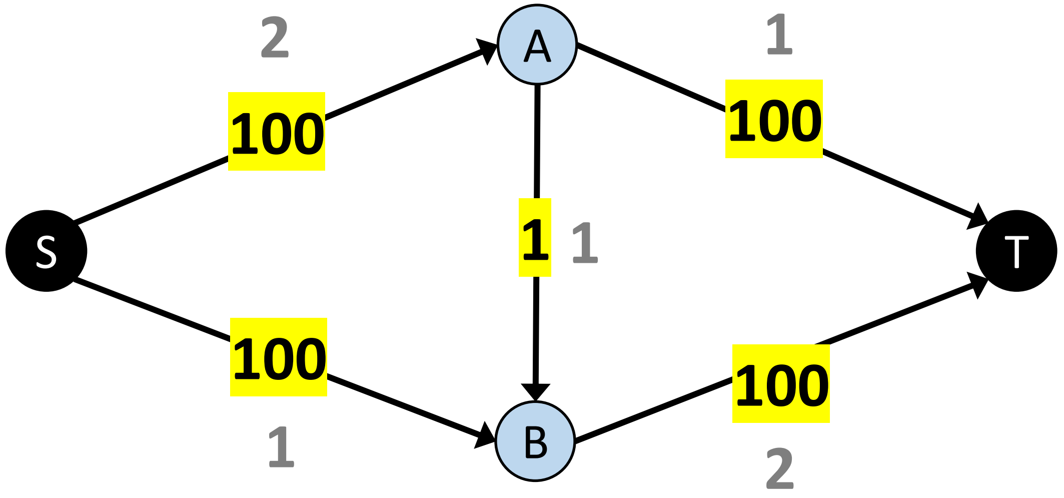
How O(*f* E)? Why O(*f* E)? When O(*f* E)?



Flow Value = 3

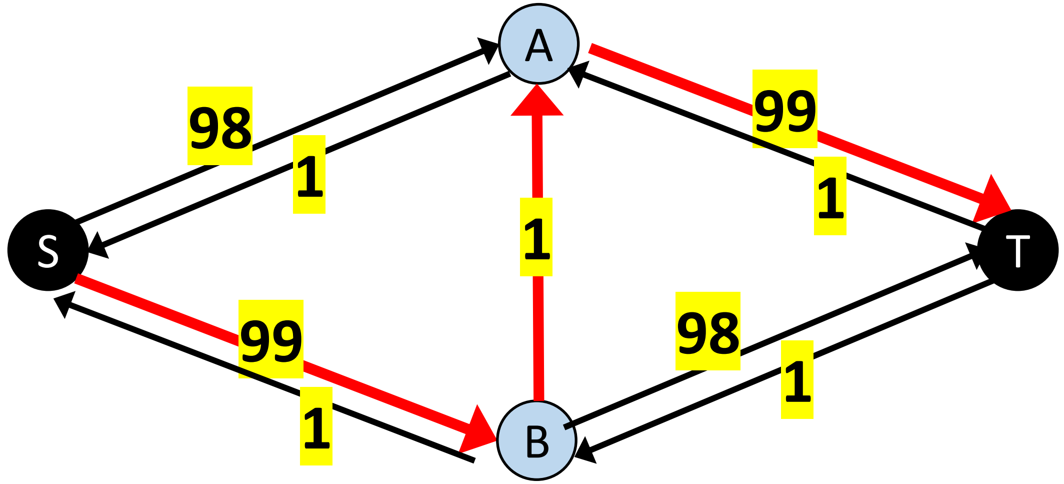
Residual Network

How O(*f* E)? Why O(*f* E)? When O(*f* E)?



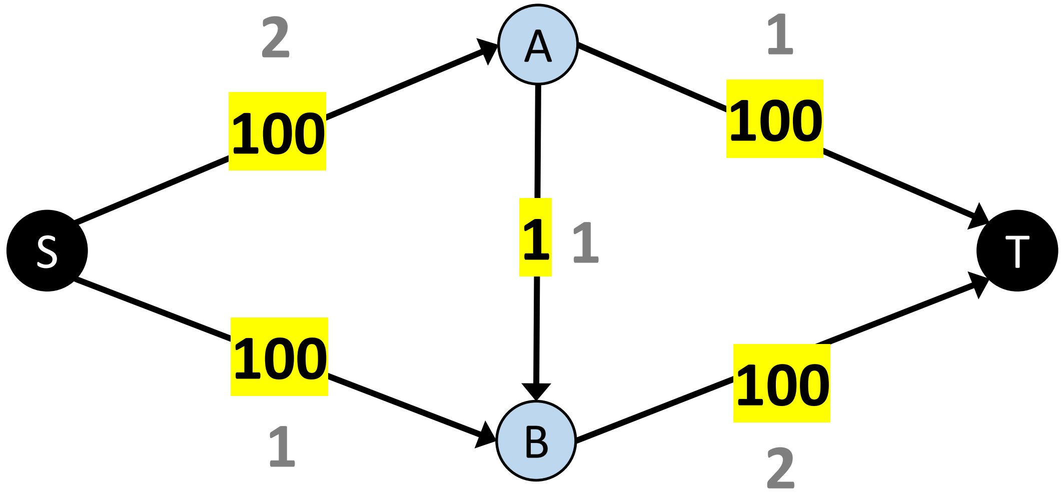
Flow Value = 3

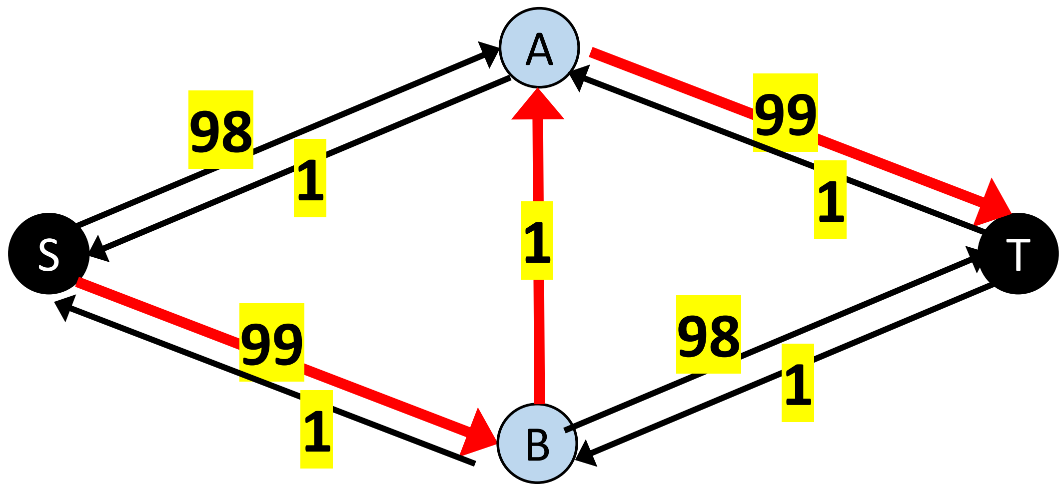
Residual Network



Bottleneck (P) = 1

How O(*f* E)? Why O(*f* E)? When O(*f* E)?





Waiting for Ford Fulkerson algorithm to complete on 4 Vertices and 5 Edges

Thanks a lot

If you are taking a Nap, **wake up**.......Lecture Over