Maximum Flow

Ford-Fulkerson Algorithm

Flow Networks

- A ***flow network*** G = (V, E)
  - directed graph in which each edge $(u, v) \in E$
  - has a nonnegative ***capacity*** $c(u, v) \geq 0$.
  - if E contains an edge (u, v), then there is no edge (v, u) in the reverse direction.
  - two vertices in a flow network:

    a source s and **a sink t** .

Flows

- A ***flow*** in G is a real-valued function $f : V \times V \rightarrow R$ that satisfies the following two properties:
  - **Capacity constraint:** For all u, v $\in$ V , we require

    $0 \leq f(u, v) \leq c(u, v)$.
  - **Flow conservation:** For all u $\in$ V – {s, t}, we require

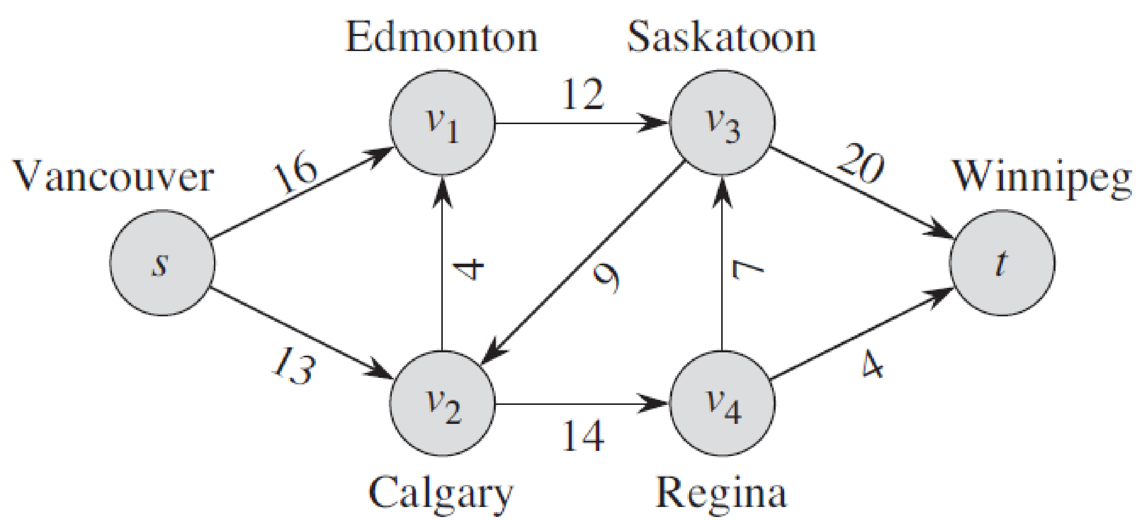
$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$

- The ***value*** $|f|$ of a flow f is defined as

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

Applications

- Fluid in pipes
- Current in electrical circuits
- Traffic on roads
- Data flow in computer networks
- Money flow in economy



Ford-Fulkerson Method

- It is called “method” rather than an “algorithm” because it encompasses several implementations with differing running times.

FORD-FULKERSON-METHOD(G, s, t)

1. initialize flow f to 0
2. while there exists an **augmenting path** p in the **residual network** G_f
3. augment flow f along p
4. return f

Some Terms

Residual Network:

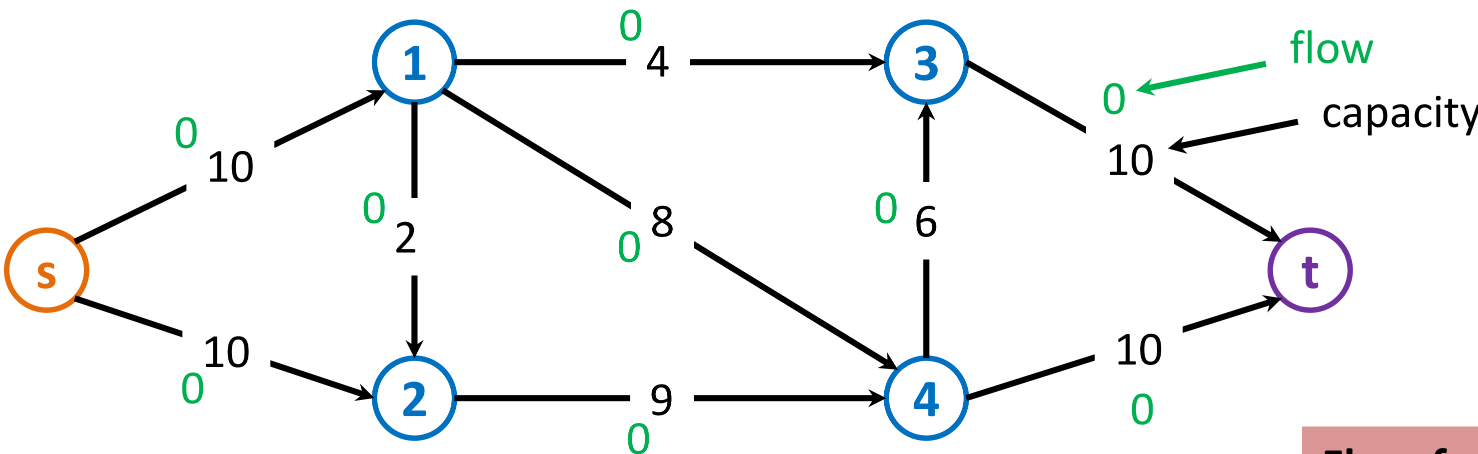
- Only edges from G that can still have more flow.
- Considering a pair of vertices u, v ϵ V, the ***residual capacity*** $c_f(u, v)$:

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

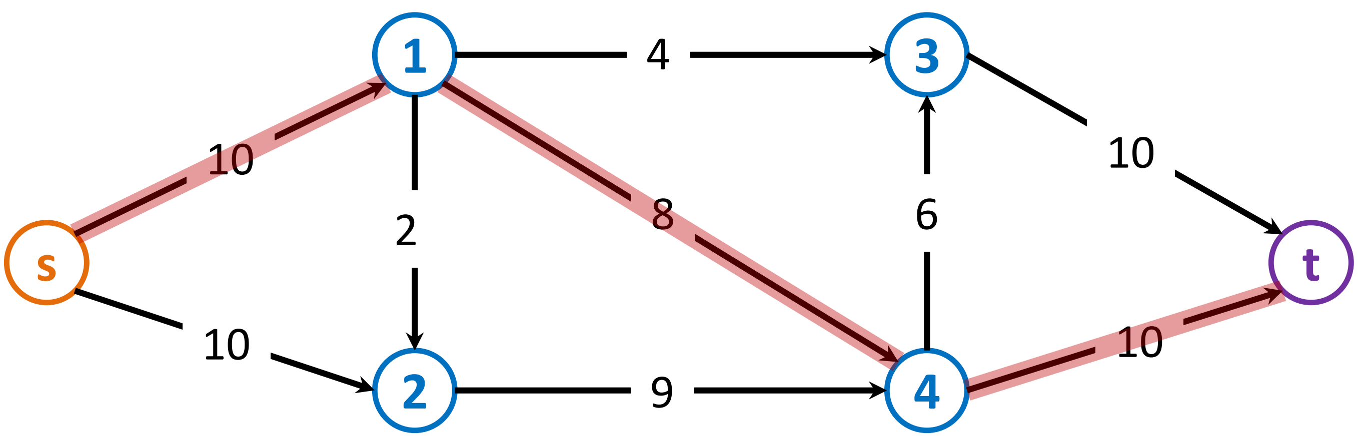
Augmenting Path:

- Path from s to t in residual network, G_f

Example

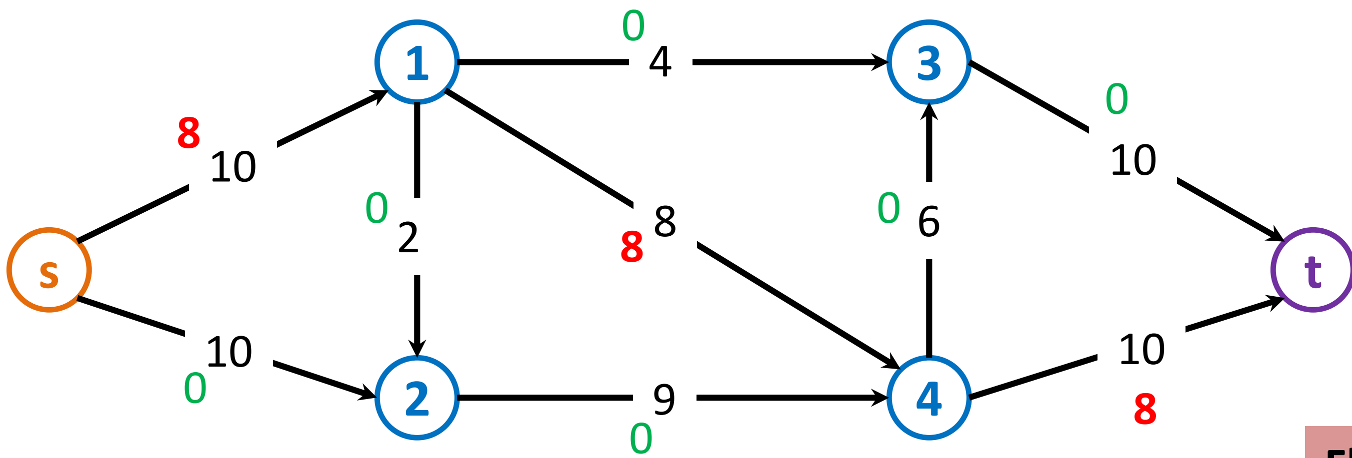


Flow *f* = 0

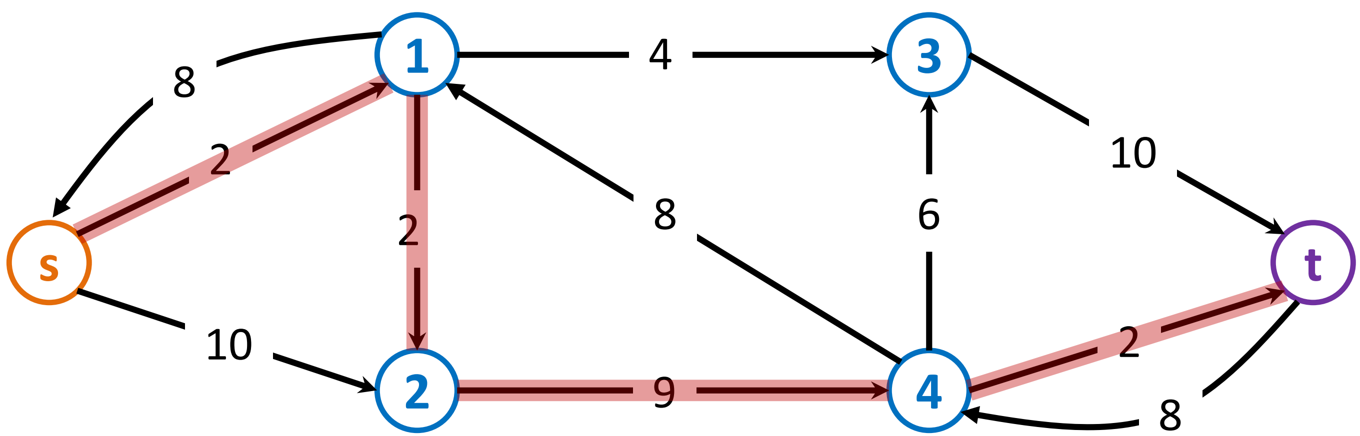


Residual Network

Example

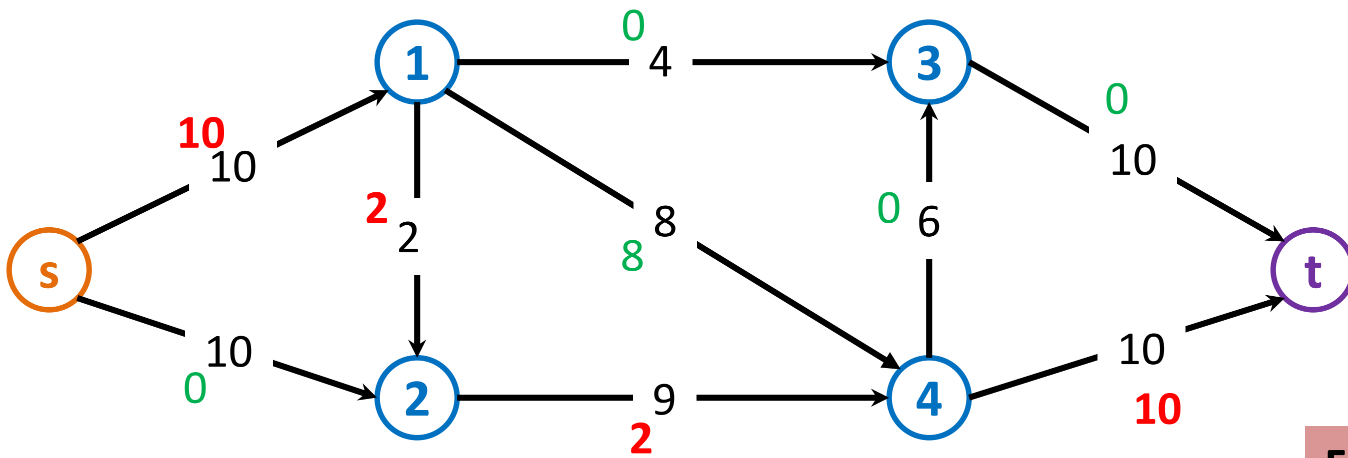


Flow *f* = 8

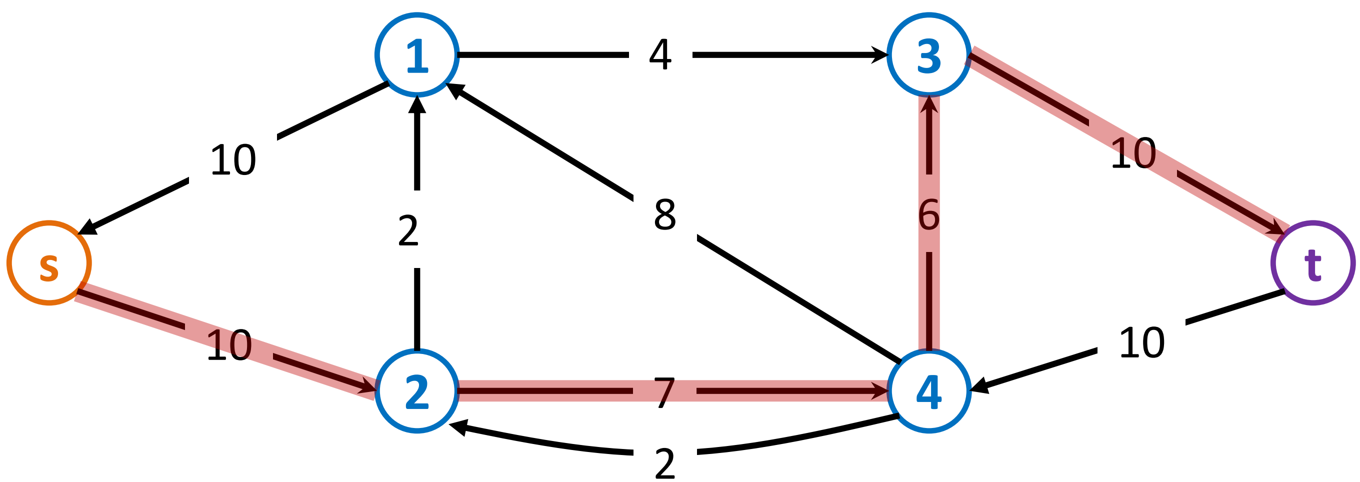


Residual Network

Example

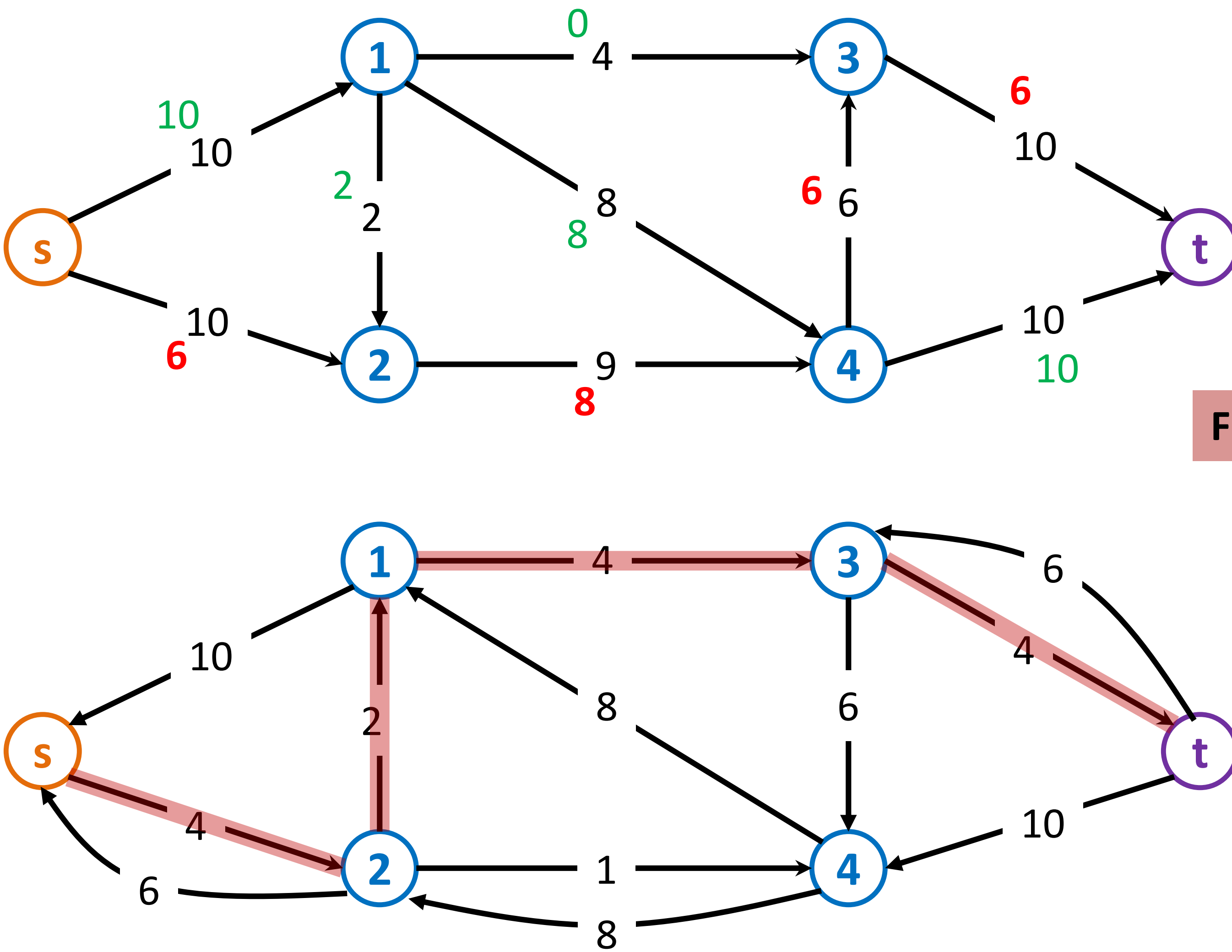


Flow f = 10



Residual Network

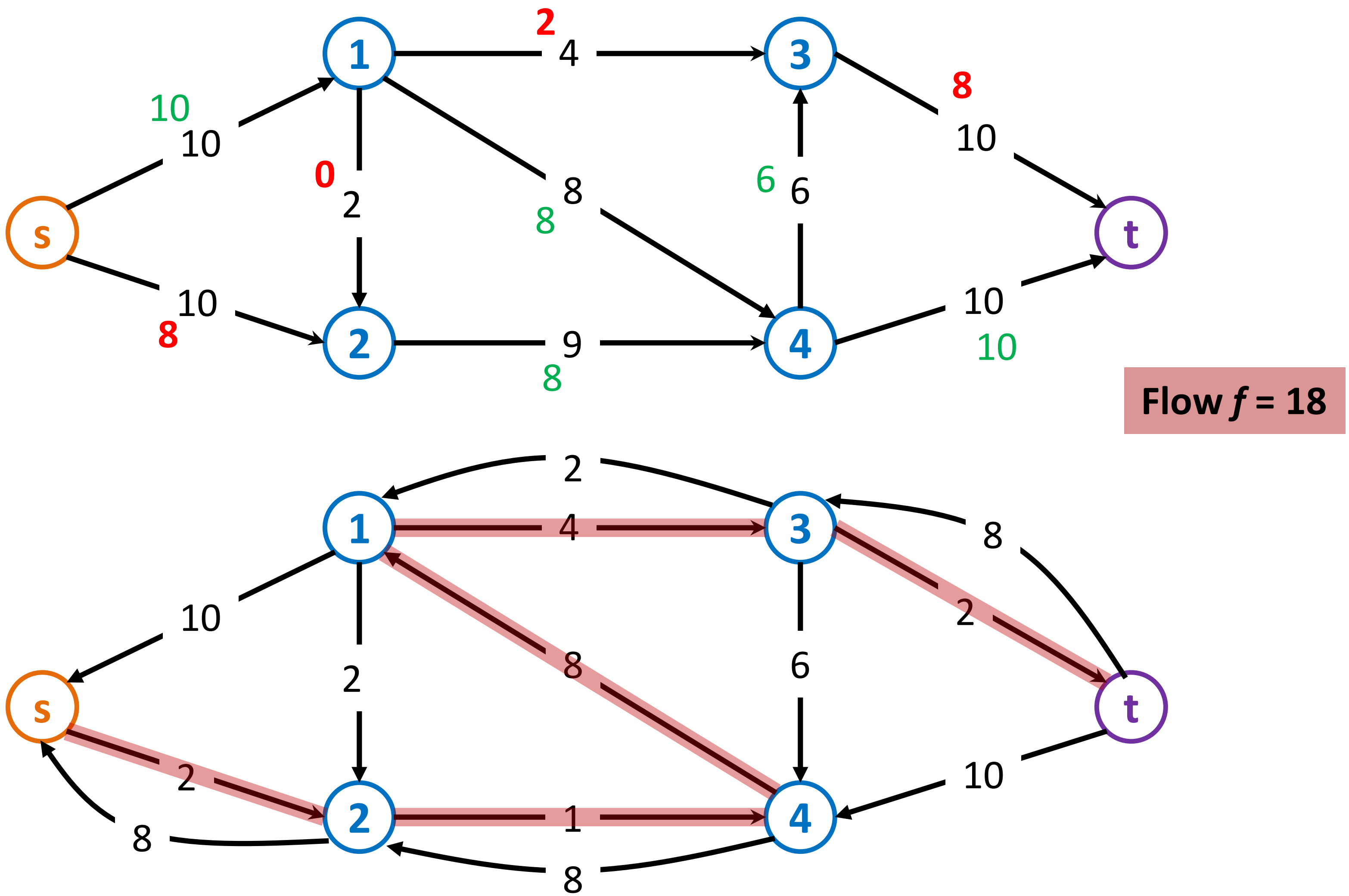
Example



Flow f = 16

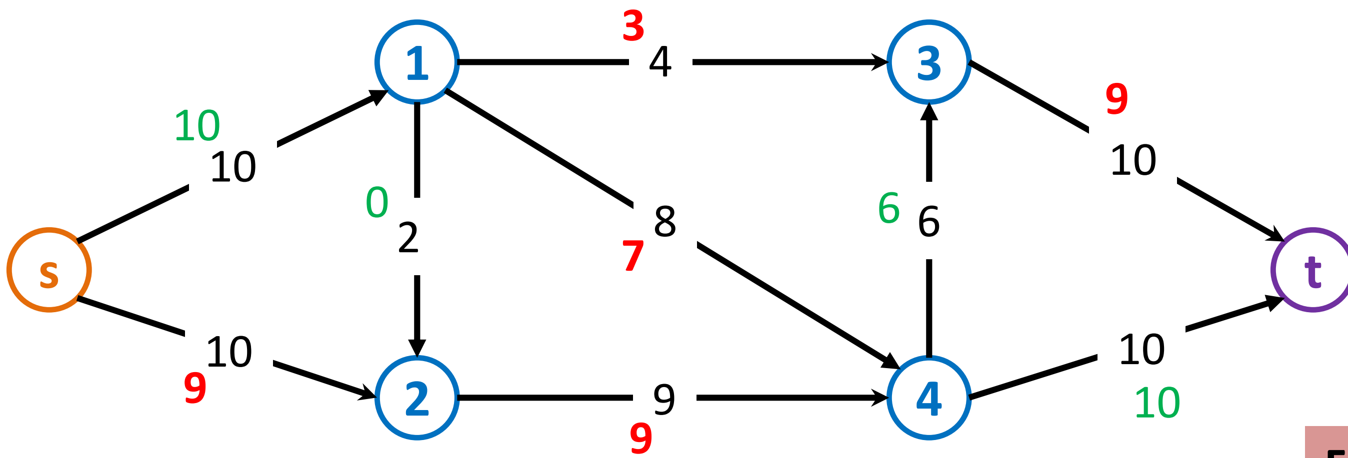
Residual Network

Example

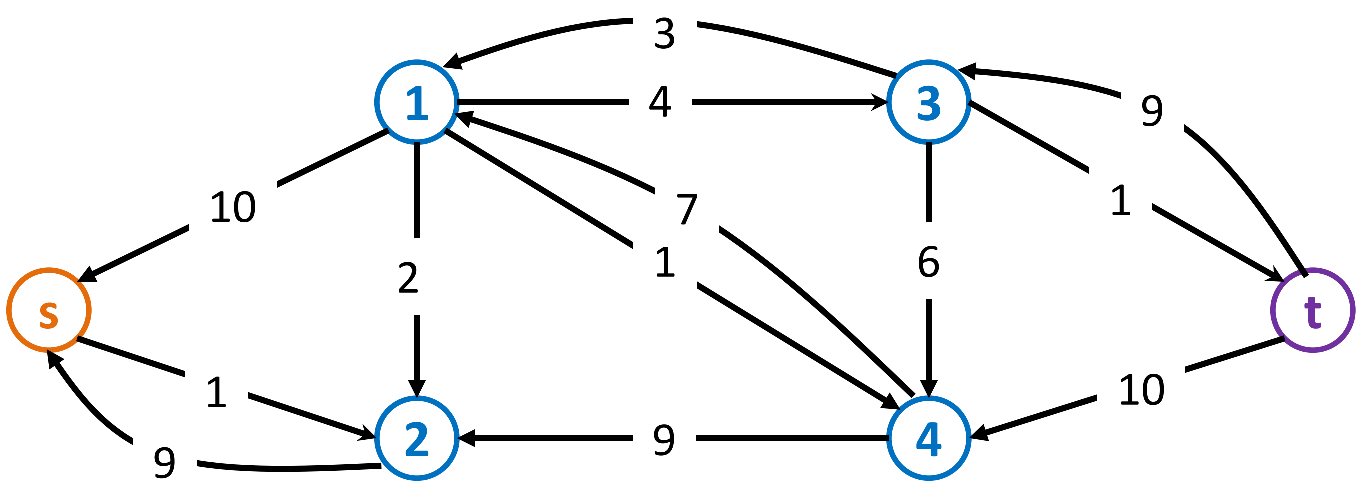


Residual Network

Example



Flow *f* = 19



Residual Network

Ford-Fulkerson Algorithm

```
FORD-FULKERSON(G, s, t)
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min{c_f(u, v) : (u, v) is in p}
    for each edge (u, v) in p
        if (u, v) ∈ E
            (u, v).f = (u, v).f + c_f(p)
        else (v, u).f = (v, u).f − c_f(p)
```

Analysis

- If f^* denotes a maximum flow in the transformed network, then a straightforward implementation of FORD-FULKERSON executes
  - the while loop of lines 3–8 at most $|f^*|$ times, since the flow value increases by at least one unit in each iteration.
- The time to find a path in a residual network is $O(V + E') = O(E)$ if we use either depth-first search or breadth-first search.
- Each iteration of the **while** loop thus takes O(E) time, as does the initialization in lines 1–2, making the total running time of the FORD-FULKERSON algorithm $O(E\ |f^*|)$.

Cuts of Flow Networks

- A ***cut*** (S, T) of flow network G = (V, E) is a partition of V into S and T = V - S such that s ϵ S and t ϵ T.
- If *f* is a flow, then the ***net flow*** *f* (S, T) across the cut (S, T) is defined to be

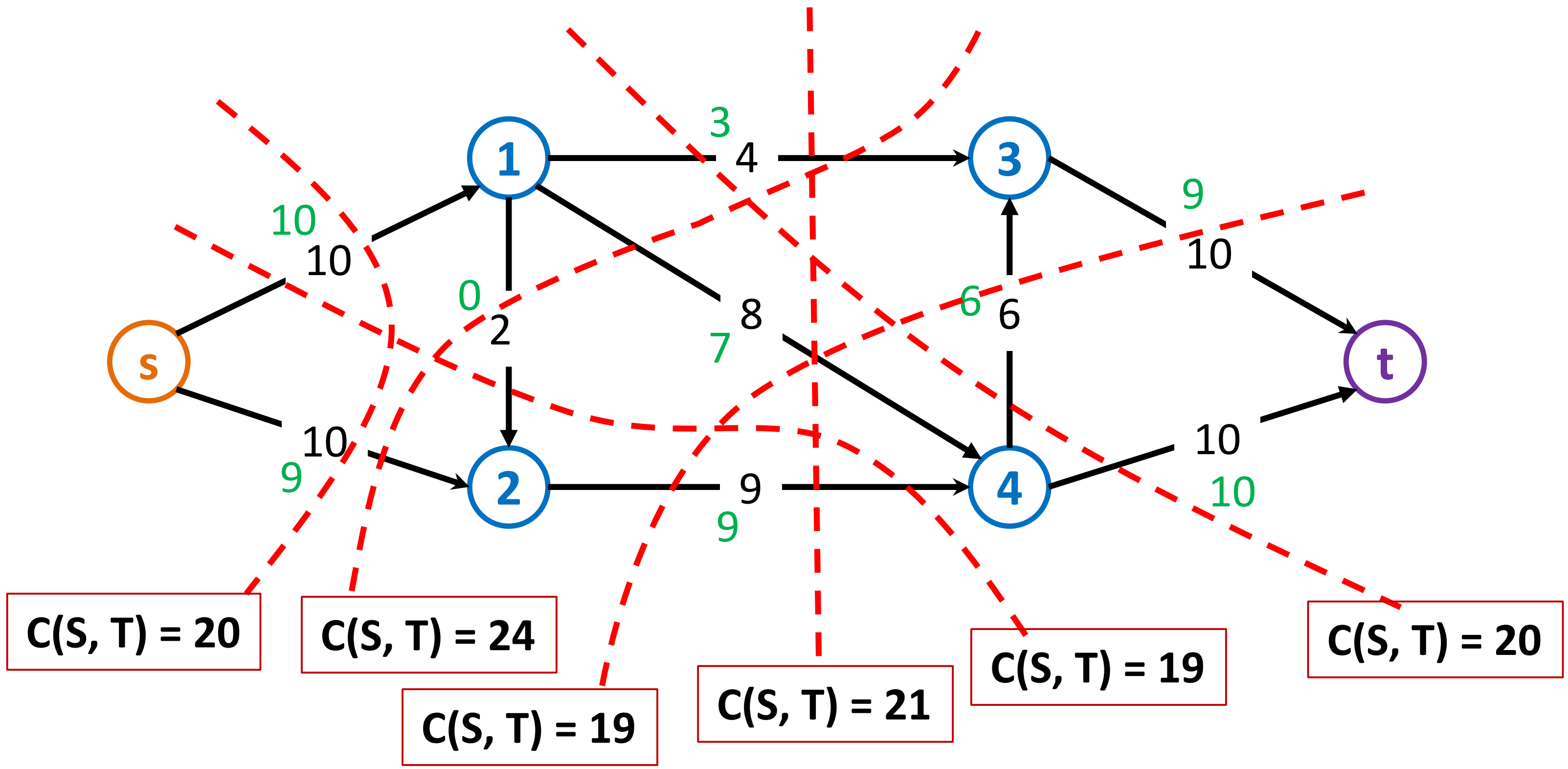
$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

- The ***capacity*** of the cut (S, T) is

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$$

Minimum Cut

- A ***minimum cut*** of a network is a cut whose capacity is minimum over all cuts of the network.



Max-flow Min-cut Theorem

If f is a flow in a flow network G = (V, E) with source **s** and sink **t** , then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. $|f| = c(S, T)$ for some cut (S, T) of G.

Proof: Self-study