Faculty of Information Technology,
Monash University

FIT2004: Algorithms and Data Structures

Week 11: Network Flow

These slides are prepared by [M. A. Cheema] and are based on the material developed by [Arun Konagurthu] and [Lloyd Allison].

Announcements

- SETU Feedback
  - Link on Moodle (on the right)
  - Closes 6th June 2021

Recommended reading

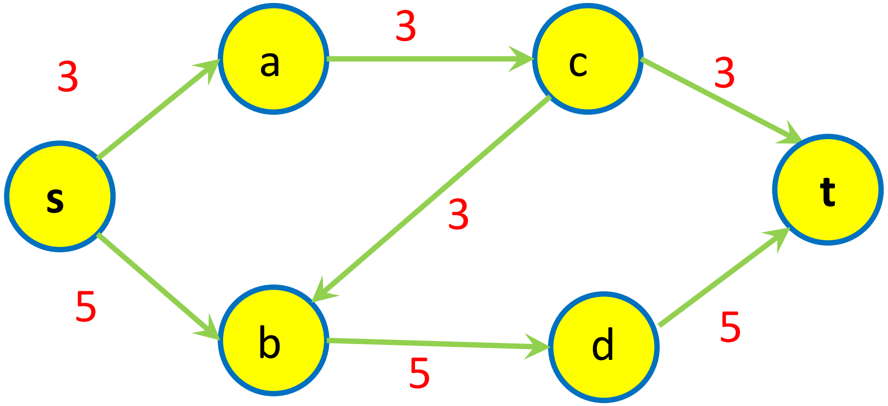
- Lecture Notes: Chapters 17
- Cormen et al. Introduction to Algorithms: Chapter 26.1 and 26.2.

Outline

1. Maximum Flow Problem
2. Ford-Fulkerson Algorithm
3. Min-cut Max-flow Theorem

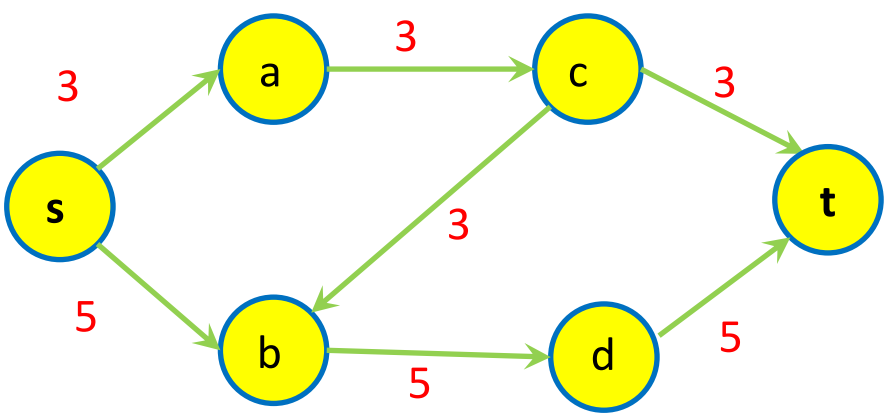
Flow Networks

- A flow network is a connected directed graph where:
  - there is a single source vertex and a single sink/target vertex;
  - each edge has a given (non-negative) capacity (usually integers).
    - giving the maximum amount/rate of flow that the edge can carry.
- Flow networks model many real-world problems:
  - water flowing through an assembly of pipes;
  - electric current flowing through electrical circuits;
  - information flowing through communication networks.
- Can be applied to solve a large range of other combinatorial problems (unrelated to physical flows).



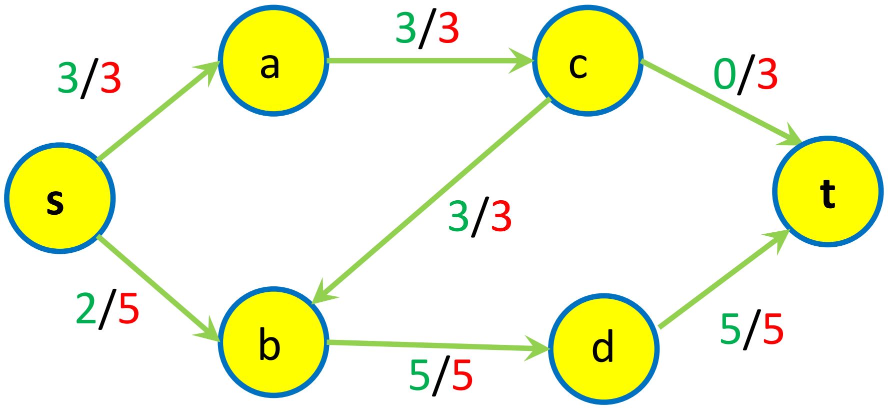
Some basic notations

- Set of all incoming edges to a vertex v: denoted as $E_{in}(v)$
  - $E_{in}(b) = s \rightarrow b, c \rightarrow b$
  - $E_{in}(a) = ?$
- Set of all outgoing edges from a vertex v: denoted as $E_{out}(v)$
  - $E_{out}(b) = b \rightarrow d$
  - $E_{out}(a) = ?$
- Source vertex: denoted as s (has no incoming edges)
- Sink/target vertex: denoted as t (has no outgoing edges)



Flow

- Flow is an **assignment** of how much material is flowing through each edge in the flow network given its stated edge capacity.



Green numbers indicate flow and red indicate capacity.
Note that we are only handling integers here…

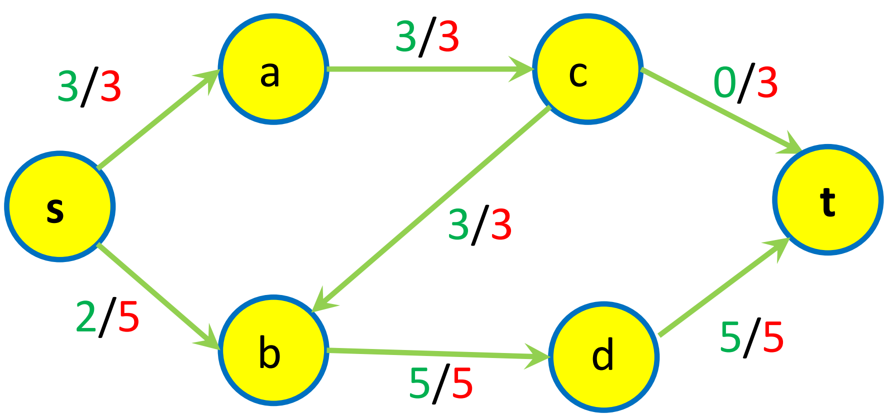
Conservation

- All vertices (except source and sink) conserve their flow. That is
  - The total amount flowing into any vertex (through incoming edges)

    IS EQUAL TO

    the total amount flowing out of that vertex (through outgoing edges).

  - This key property is called flow conservation.



Quiz time!
https://flux.qa/YTJMAZ

Properties of a Flow Network

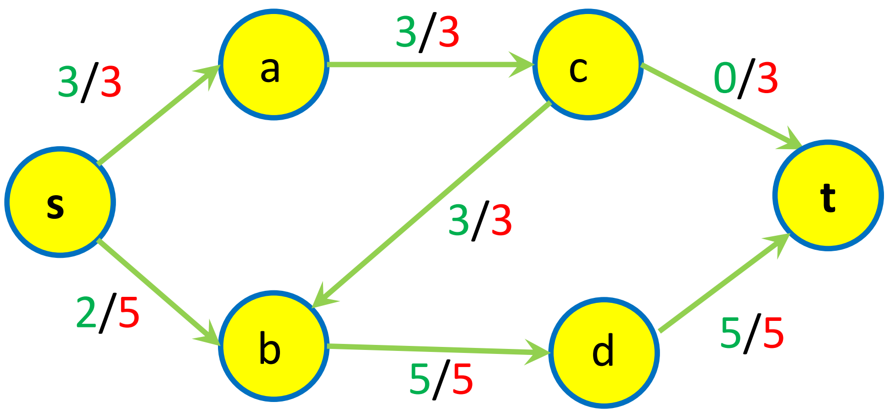
A flow network must satisfy the following two properties.

Property 1: Capacity Constraint

- For each edge e, its flow, denoted as f(e), is bounded by the capacity of its edge, i.e., $0 \le f(e) \le c(e)$ where c(e) is the capacity of the edge.

Property 2: Flow Conservation

- For any vertex v (except source and sink), the total flow coming into the vertex must be equal to the total flow going out from this vertex – formally

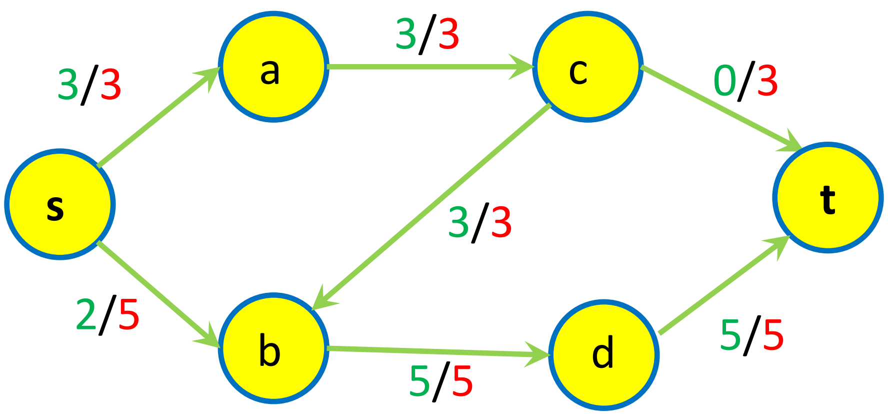


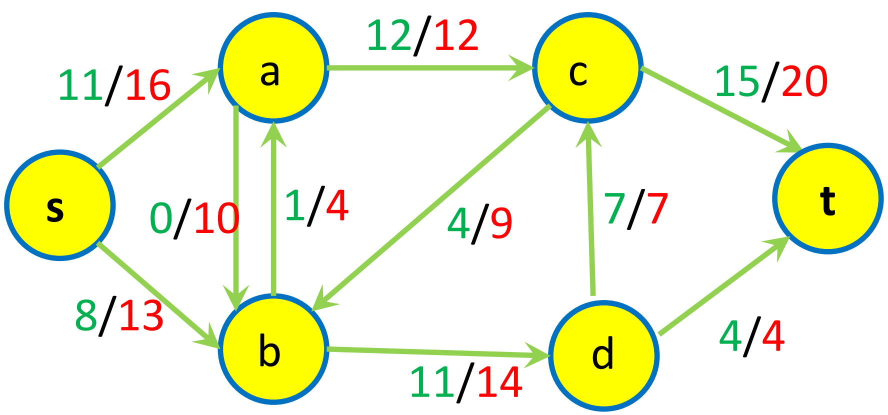
$$\sum_{\forall e_{in} \in E_{in}(v)} f(e_{in}) = \sum_{\forall e_{out} \in E_{out}(v)} f(e_{out}).$$

Maximum-flow Problem

Value of a flow in a network:

- Given that flow network satisfies the capacity constraint and flow conservation properties, flow of a network is the total flow out of the source vertex.
- Equivalently, this is the same as the total flow into sink vertex.
  - What is the flow value in the flow network at bottom left?
  - What is the flow value in the flow network at bottom right?



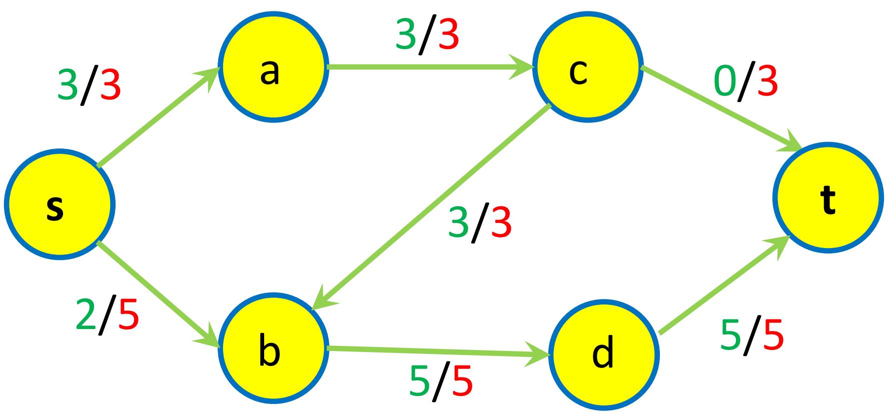


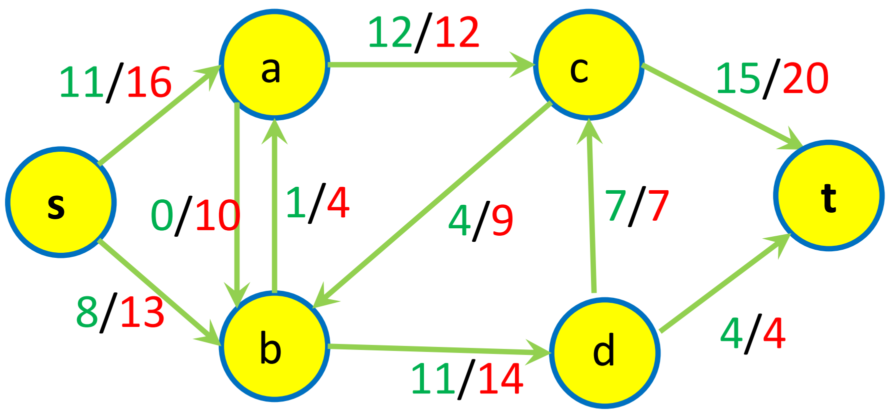
Maximum-flow Problem

Maximum-flow problem

- Given a flow network, determine the maximum value of the flow that can be sent from source s to sink t without violating the flow network properties.

Quiz time!
https://flux.qa/YTJMAZ

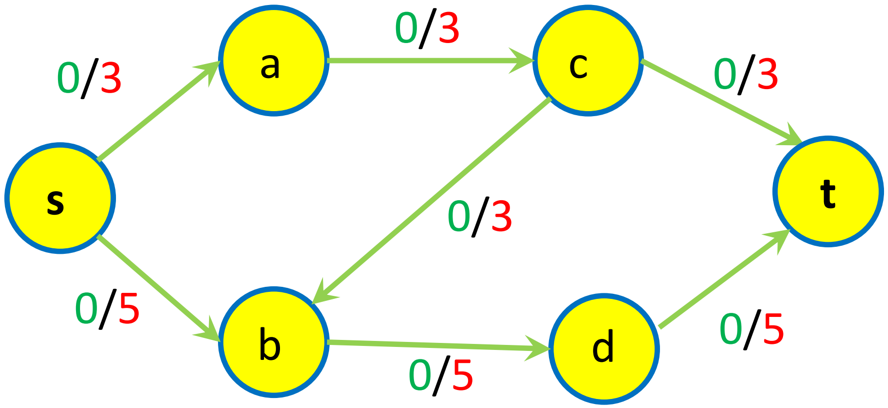




Outline

1. Maximum Flow Problem
2. Ford-Fulkerson Algorithm
3. Min-cut Max-flow Theorem

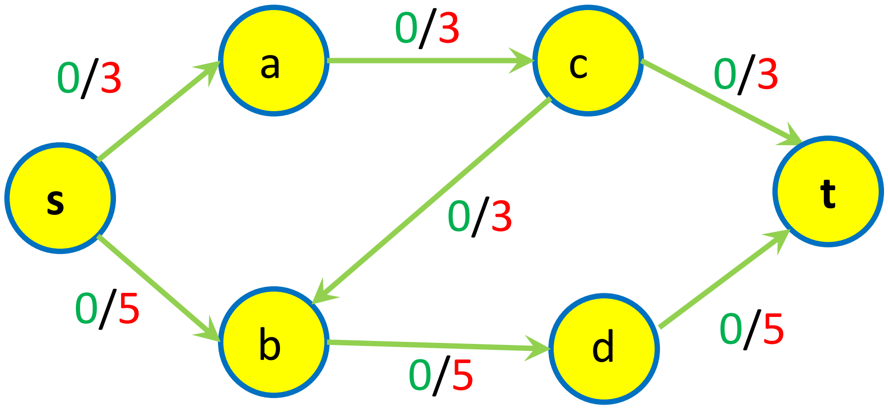
Ford Fulkerson Intuition



How can we increase the flow in the graph above?

Quiz time!
https://flux.qa/YTJMAZ

Ford Fulkerson Intuition

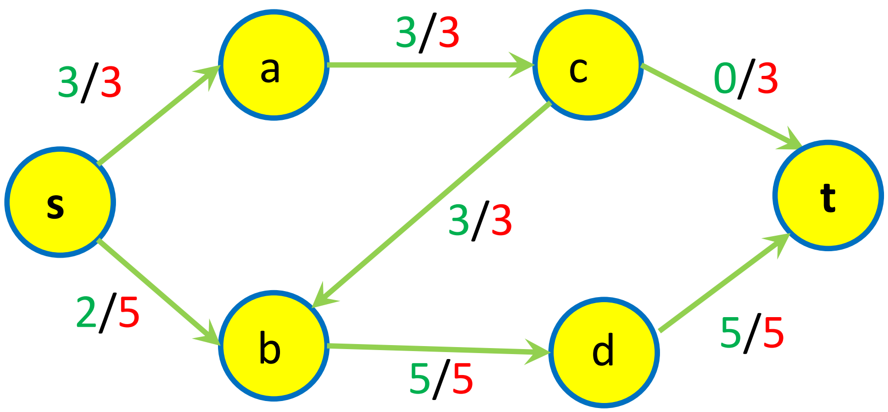


How can we increase the flow in the graph above?

1. Choose a path from source to sink
2. Increase flow along it

Seems easy enough!

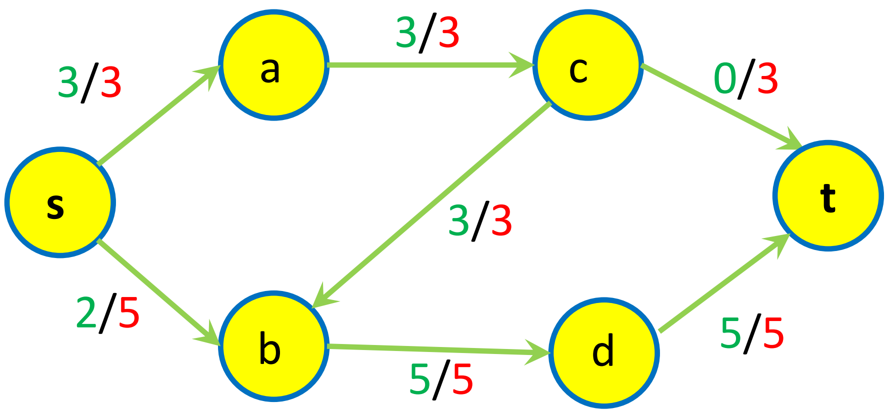
Ford Fulkerson Intuition



Can we increase the flow in the above network?

Quiz time!
https://flux.qa/YTJMAZ

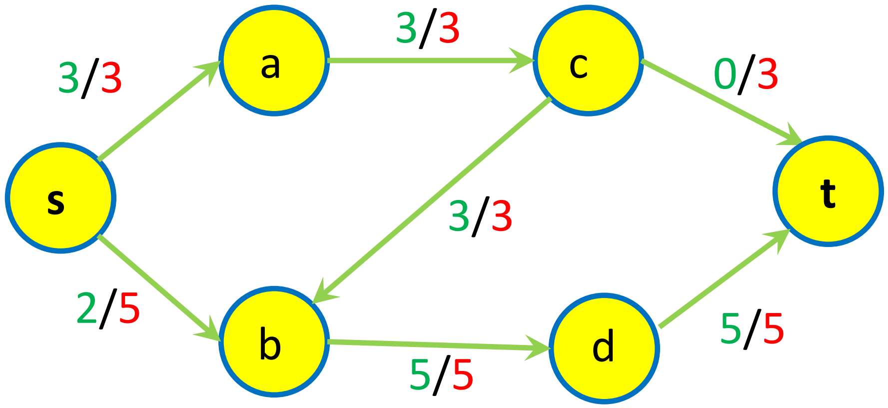
Ford Fulkerson Intuition



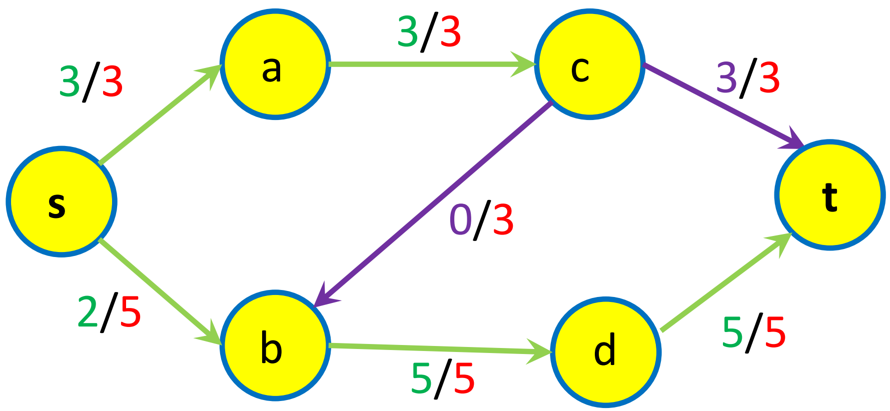
Can we increase the flow in the above network?

We can!
But there is no path from source to sink with spare capacity...

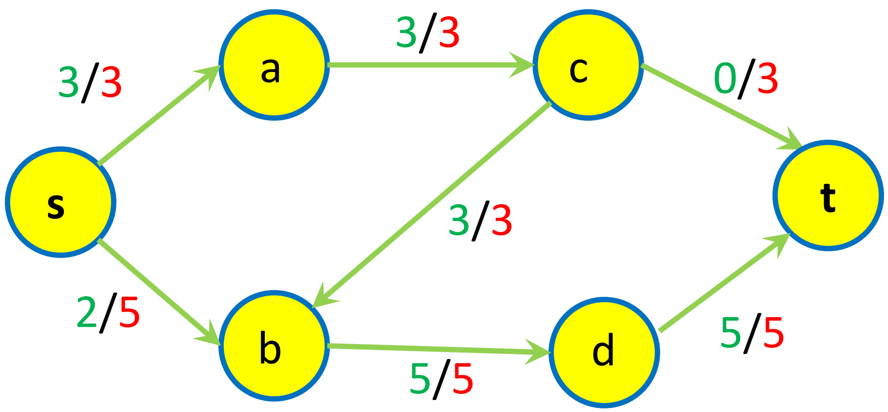
Ford Fulkerson Intuition

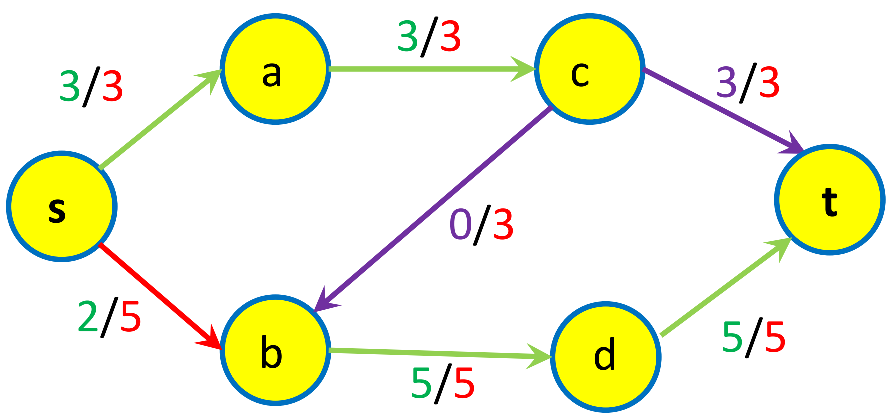


- There is no path from s to t with spare capacity.
- Instead, we “redirect” the 3 units on the edge c->b.
- They are now going to t.



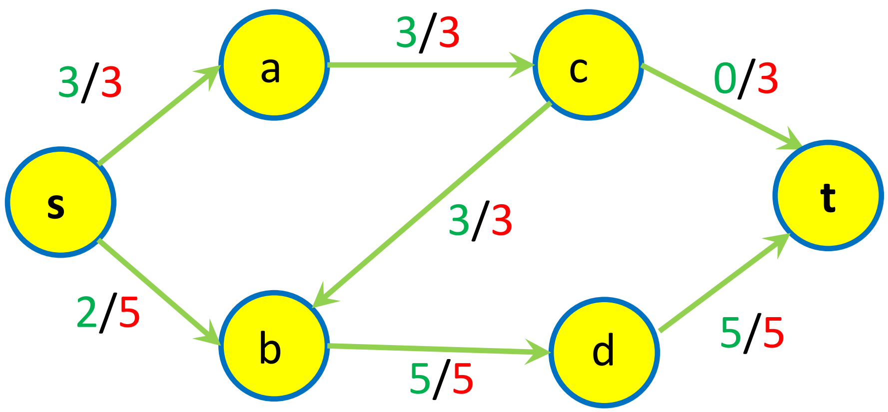
Ford Fulkerson Intuition

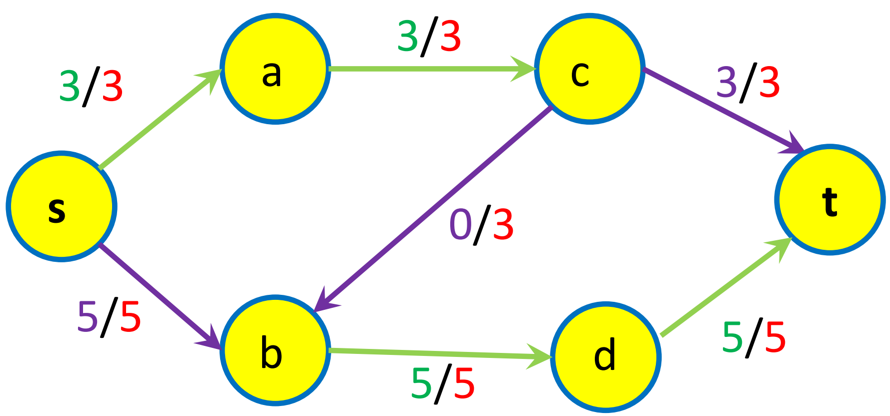




- There is no path from s to t with spare capacity.
- Instead, we "redirect" the 3 units on the edge c->b.
- They are now going to t.
- In the second diagram, the flow through b is not conserved.
- We can send 3 more units along s->b to both increase total flow and conserve flow through b.

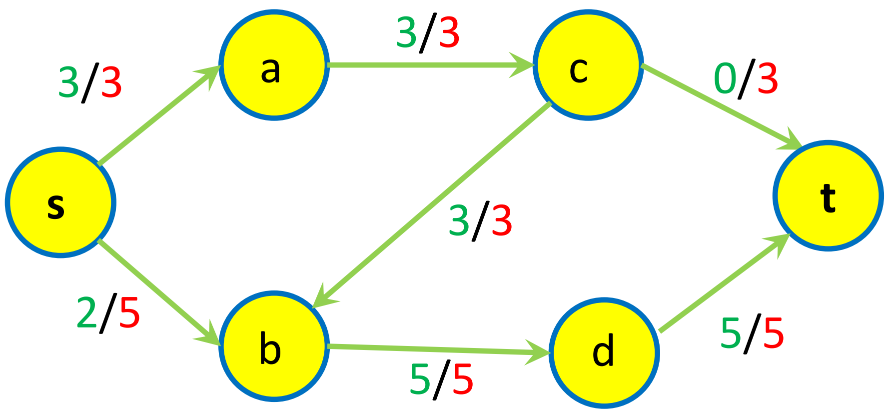
Ford Fulkerson Intuition

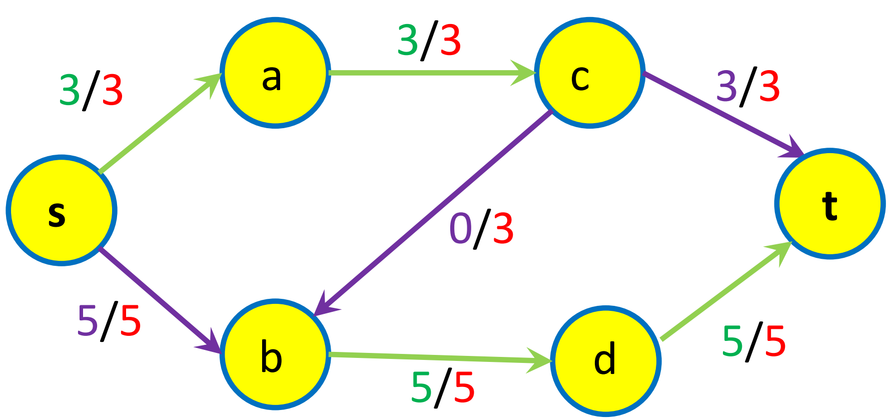




- There is not path from s to t with spare capacity.
- Instead, we "redirect" the 3 units on the edge c->b.
- They are now going to t.
- This means that we need to send 3 more units into b (because of flow conservation).
- In this case, the extra flow comes from s.

Ford Fulkerson Intuition

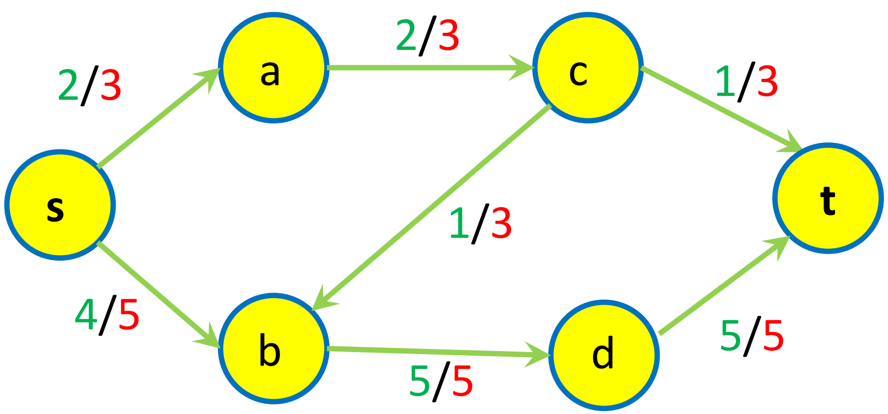




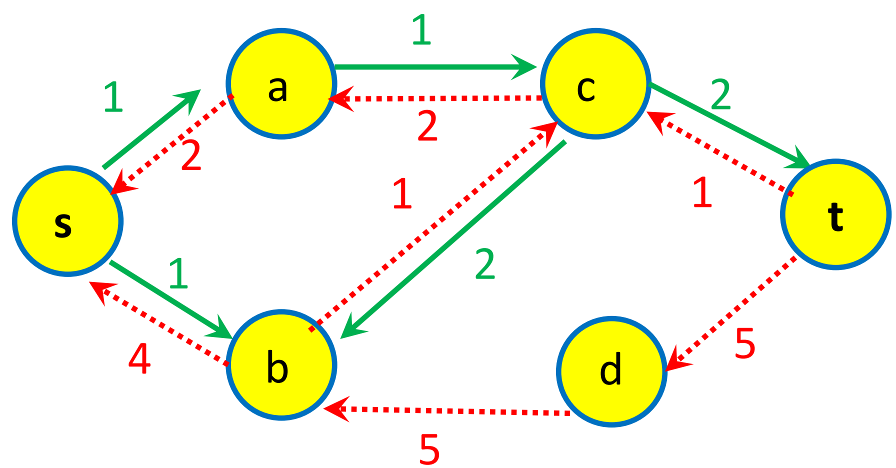
- What actually happened here?
- We increased the total flow by 3:
  - We sent 3 units along s->b, c->t.
  - We "removed" 3 units of flow from c->b.
- Our path was s->b->c->t, but we had a backwards edge…

Residual Network

- Residual network has the same vertices as the original network.
- For every directed edge u→v in flow network, we add two edges in the residual network:
  - Forward edge/Residual edge: An edge in the same direction as u→v with the residual/remaining capacity.
  - Backward edge/Reversible flow edge: An edge in the direction opposite to u→v (i.e., v→u) with weight equal to the current flow of u→v in the flow network.



Flow Network: Green numbers indicate flow and red indicate capacity. Flow is not shown if 0.

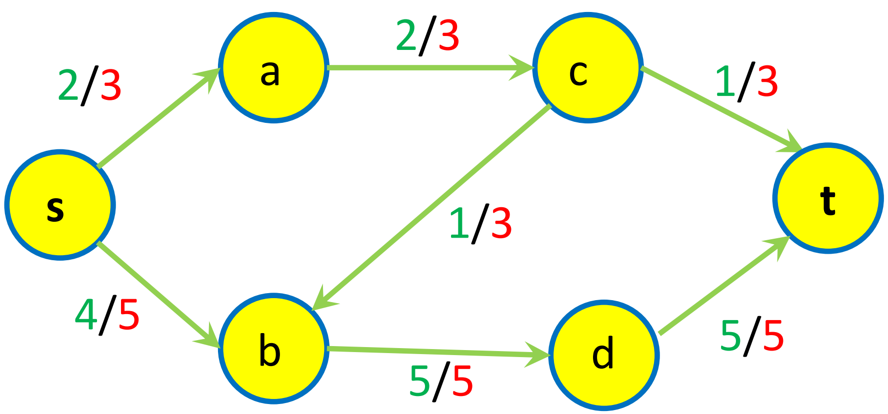


Residual Network: Where possible red edges indicate reversible flow and green indicate residual capacity.

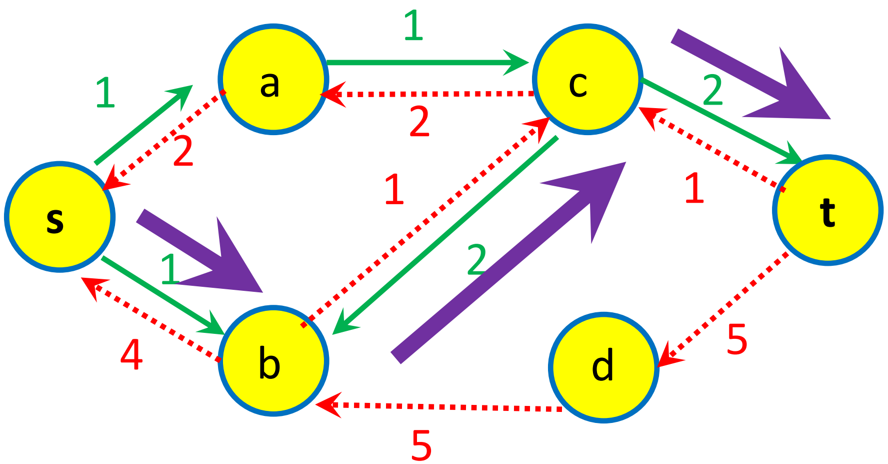
Augmenting Path

Augmenting path is any simple path (a path without repeating vertices) from source s to target t using both edge types.

- E.g., s → b → c → t (shown in purple edges).
- **Residual capacity** of a path is the minimum edge weight on this path (e.g., 1 in the example).
- For each edge along the augmenting path, we can push additional flow up to the residual capacity, e.g., 1 along each edge on s → b → c →t.



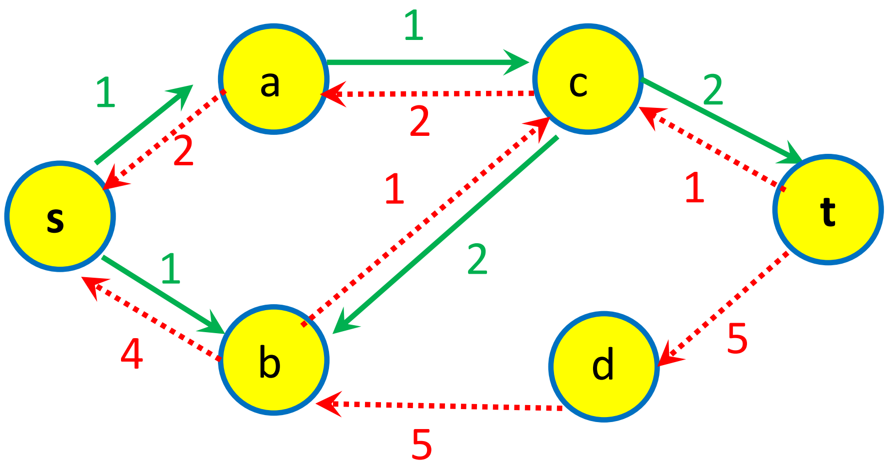
Flow Network: Green numbers indicate flow and red indicate capacity. Flow is not shown if 0

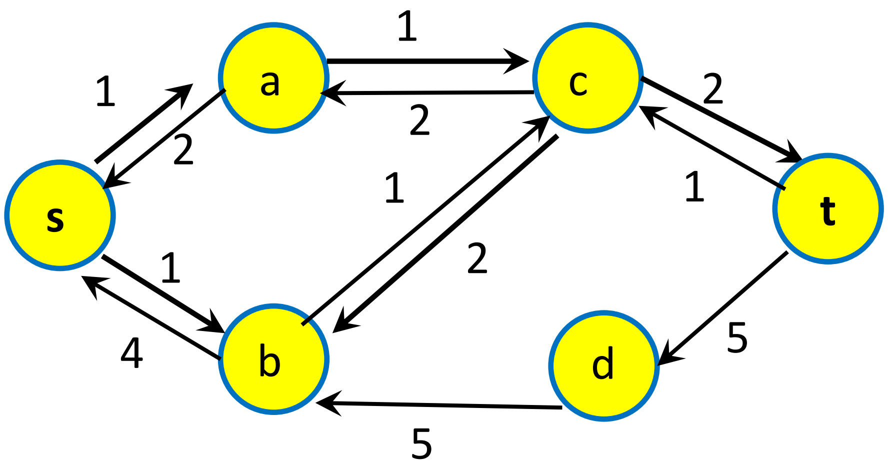


Residual Network: Where possible red edges indicate reversible flow and green indicate residual capacity

Augmenting Path Details

- Edge “type” in the residual does not matter.
- We can send flow along any edge in the residual, so they are all equivalent.
- Spare capacity edges can have flow sent along them because they have spare capacity.
- Existing flow edges (which are in the opposite direction to the flow) can have flow sent along them because reducing a flow in one direction is the same as increasing the flow in the opposite direction.



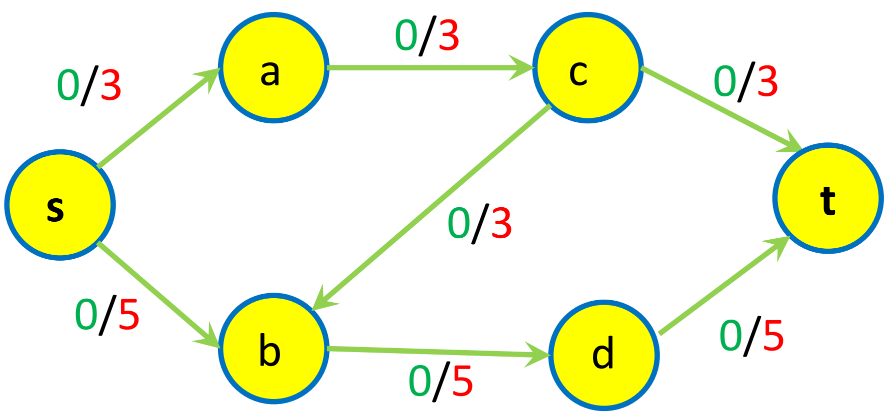


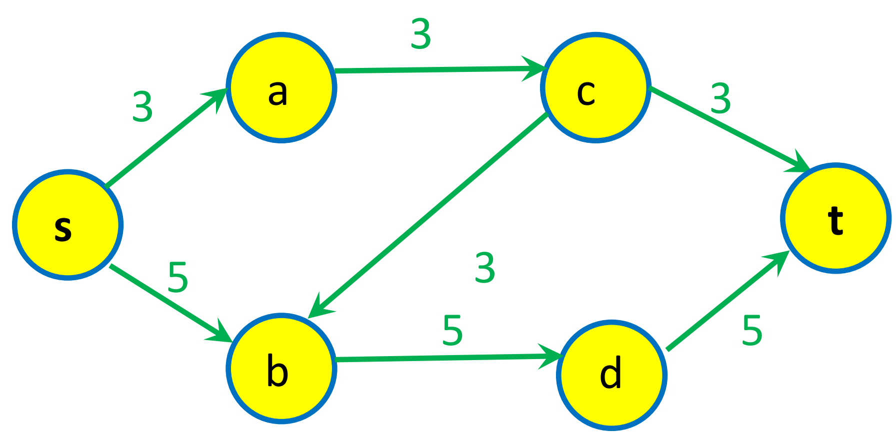
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
  Set initial flow f to 0 on all edges
  while there exists an augmenting path p in the residual network G_f **do**
    Augment the flow f along the augmenting path p
  return f

Total flow: 0



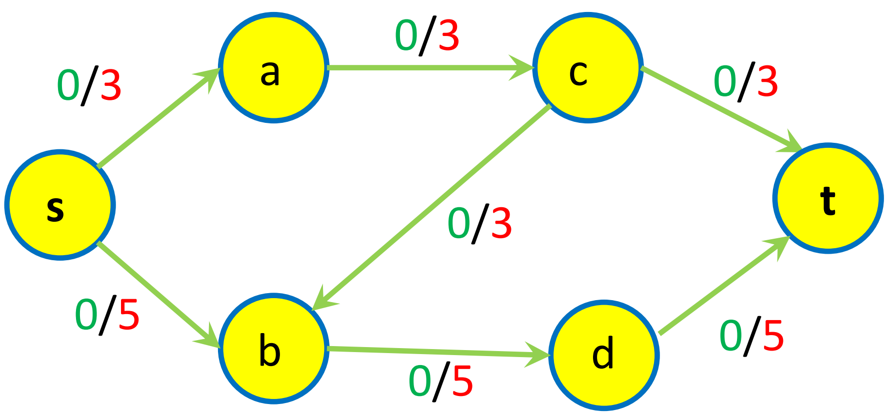


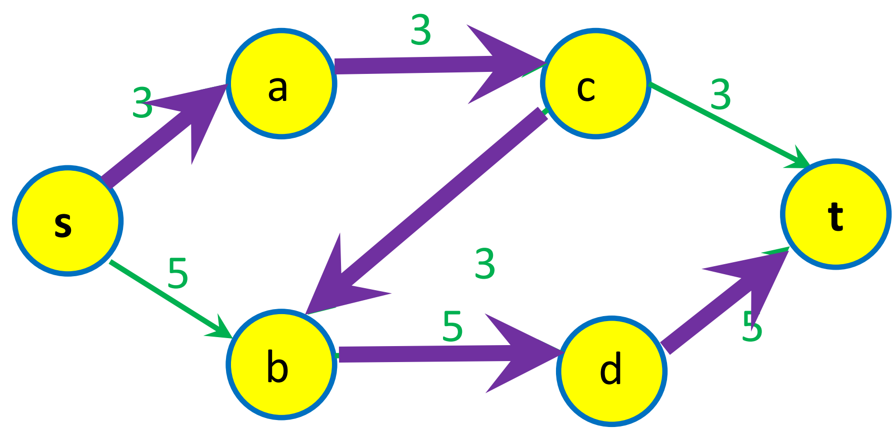
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
  Set initial flow f to 0 on all edges
  while there exists an augmenting path p in the residual network G_f **do**
    Augment the flow f along the augmenting path p
  return f

Total flow: 0



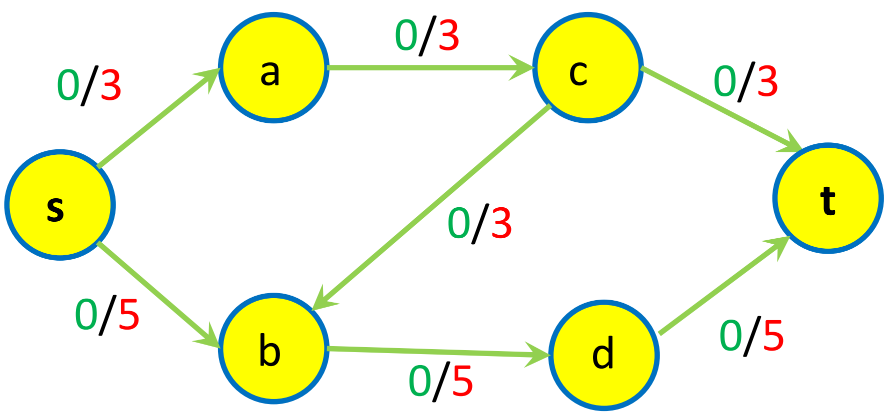


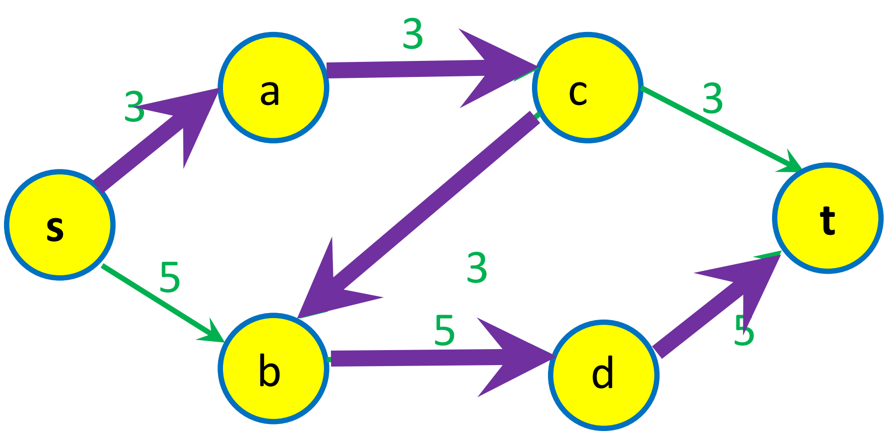
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
  Set initial flow f to 0 on all edges
  while there exists an augmenting path p in the residual network G_f **do**
    Augment the flow f along the augmenting path p
  return f

Total flow: 0





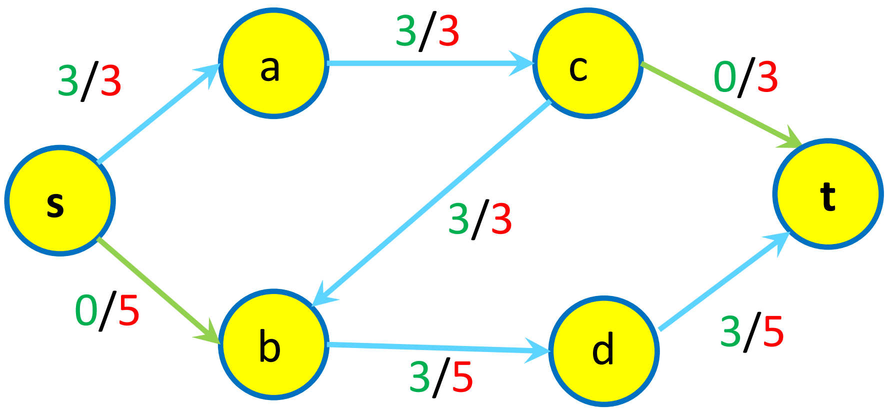
Path found: capacity = 3.

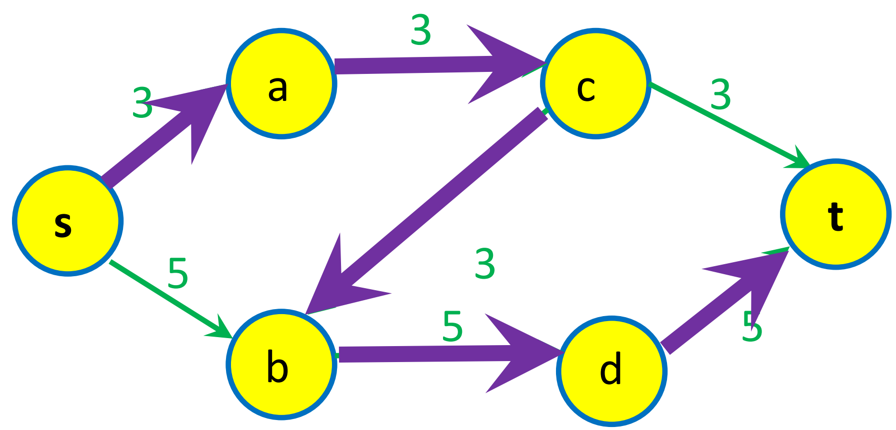
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
  Set initial flow f to 0 on all edges
  while there exists an augmenting path p in the residual network G_f **do**
    Augment the flow f along the augmenting path p
  return f

Total flow: 3





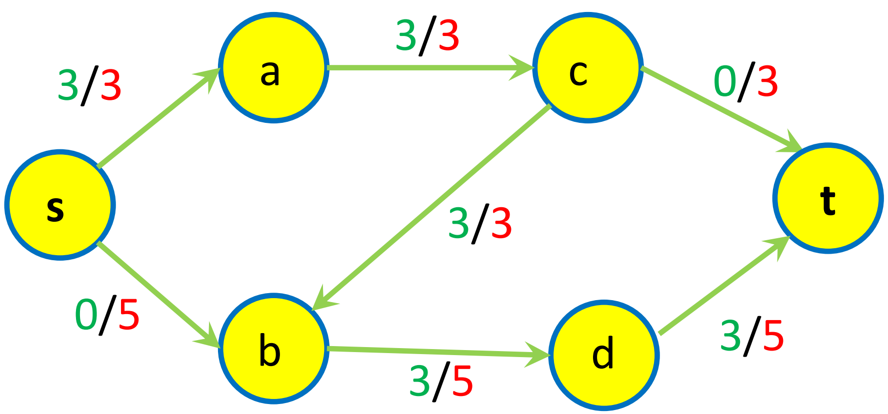
Path found: capacity = 3.

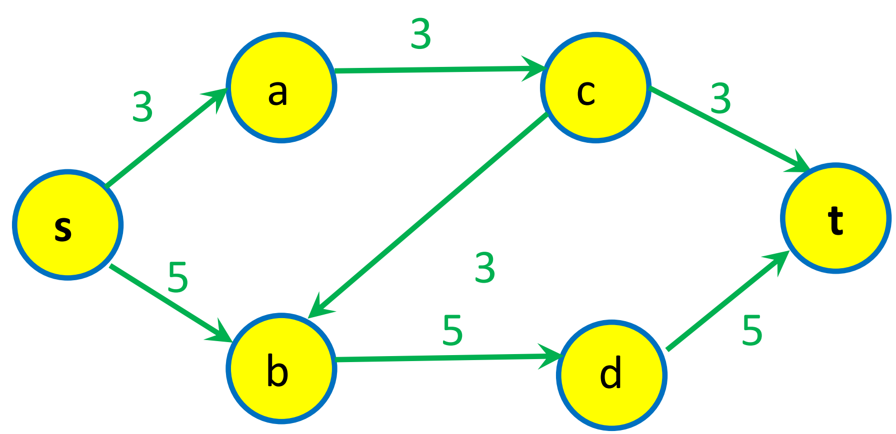
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
  Set initial flow f to 0 on all edges
  while there exists an augmenting path p in the residual network G_f **do**
    Augment the flow f along the augmenting path p
  return f

Total flow: 3





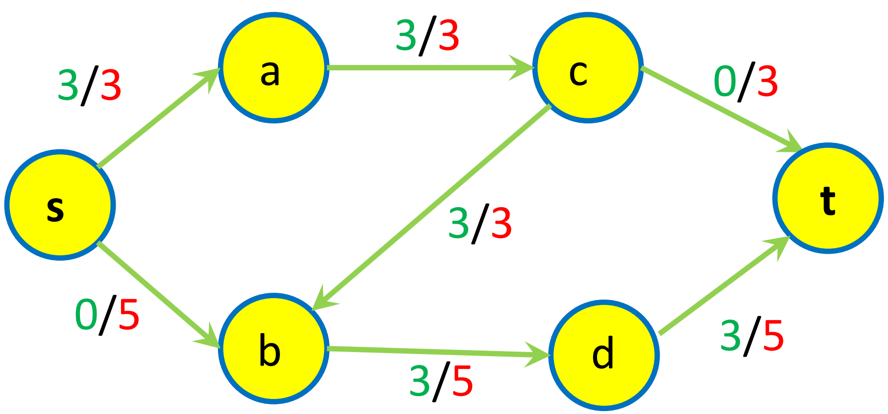
Update residual to match new flows.

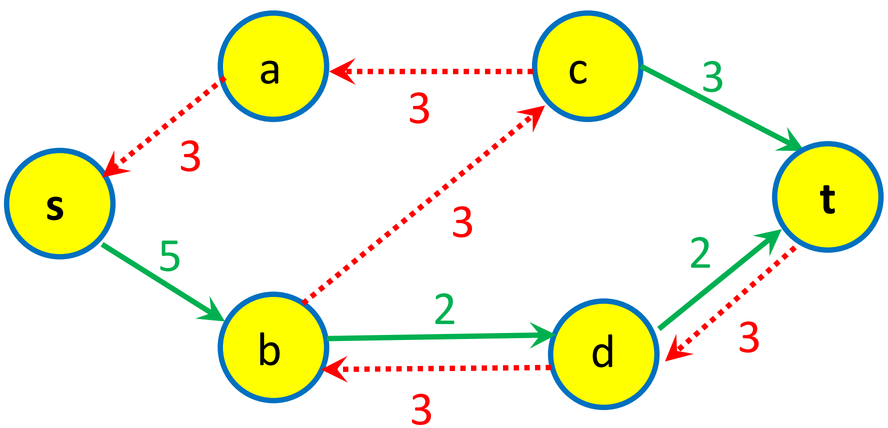
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

```
function MAX_FLOW(G = (V, E), s, t)
    Set initial flow f to 0 on all edges
    while there exists an augmenting path p in the residual network G_f do
        Augment the flow f along the augmenting path p
    return f
```

Total flow: 3





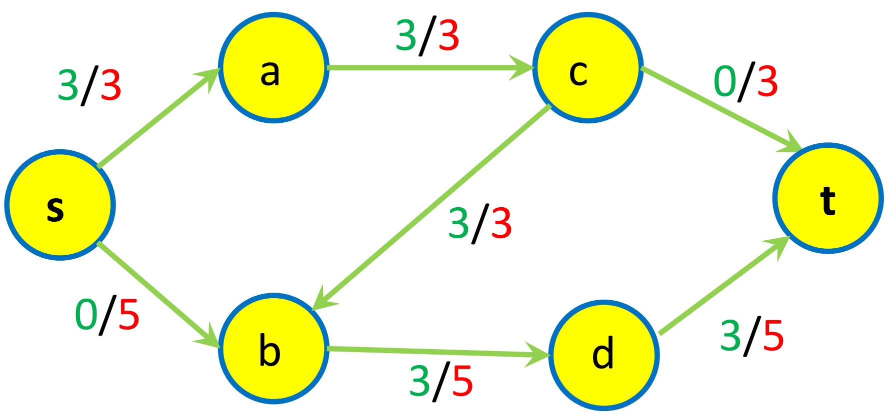
Update residual to match new flows.

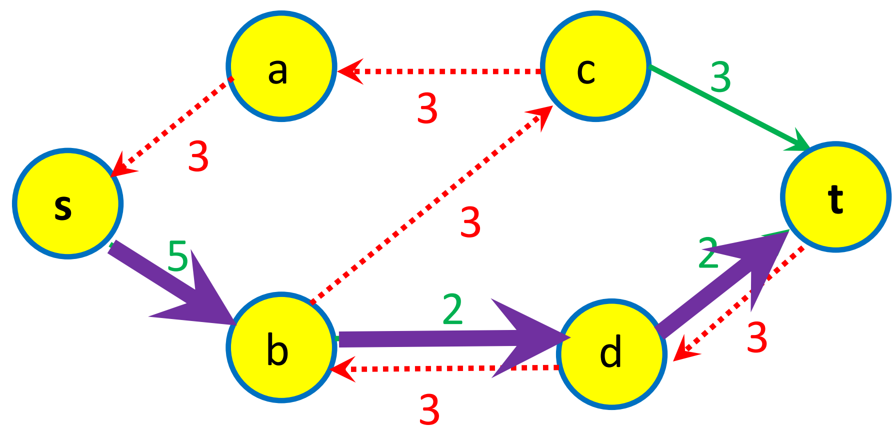
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
  Set initial flow f to 0 on all edges
  while there exists an augmenting path p in the residual network G_f **do**
    Augment the flow f along the augmenting path p
  return f

Total flow: 3





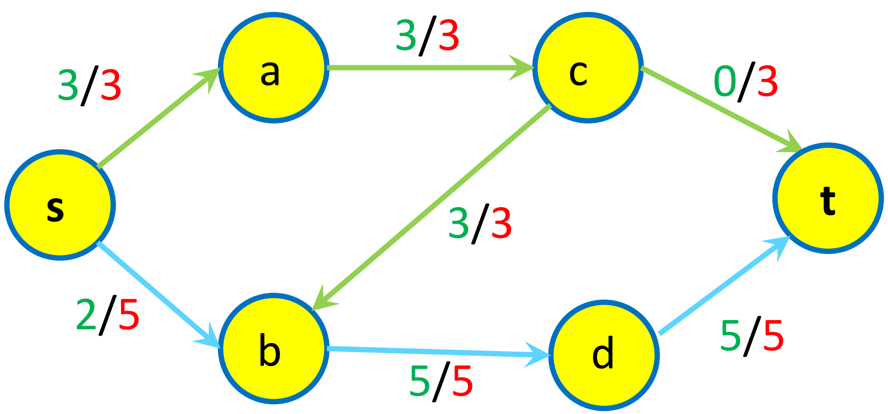
Path found: capacity = 2.

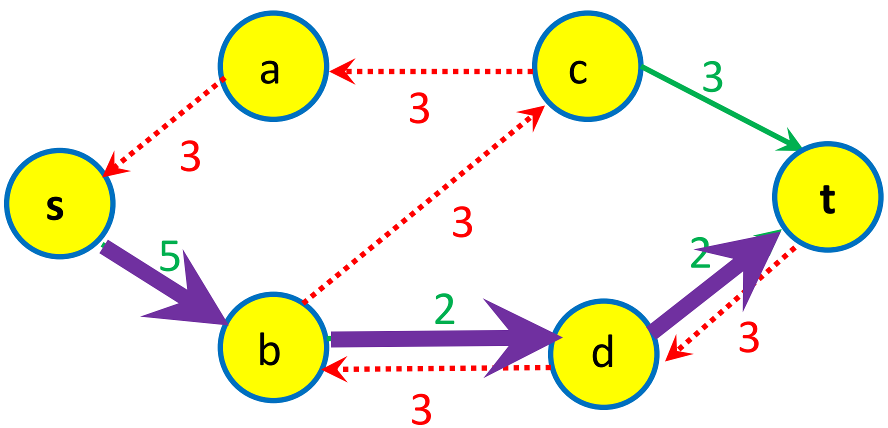
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
  Set initial flow f to 0 on all edges
  while there exists an augmenting path p in the residual network G_f **do**
    Augment the flow f along the augmenting path p
  return f

Total flow: 5





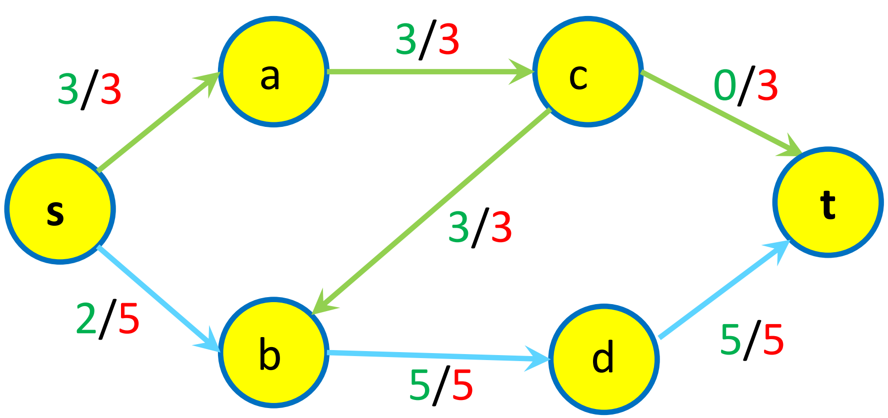
Path found: capacity = 2.

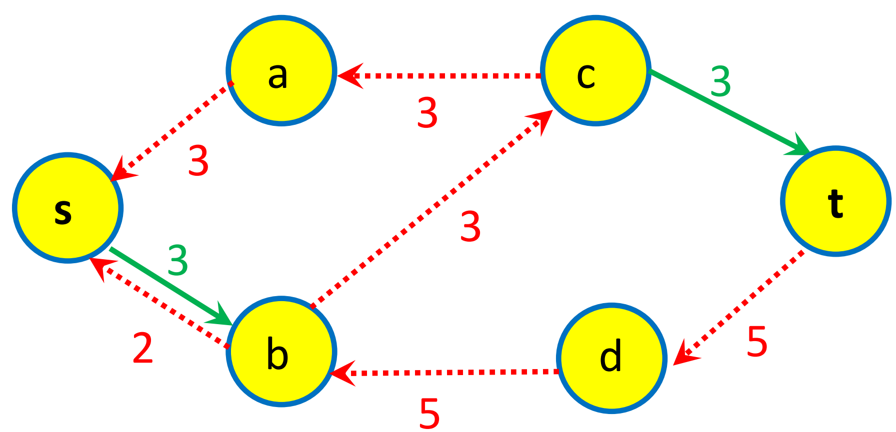
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
    Set initial flow f to 0 on all edges
    while there exists an augmenting path p in the residual network G_f **do**
        Augment the flow f along the augmenting path p
    return f

Total flow: 5





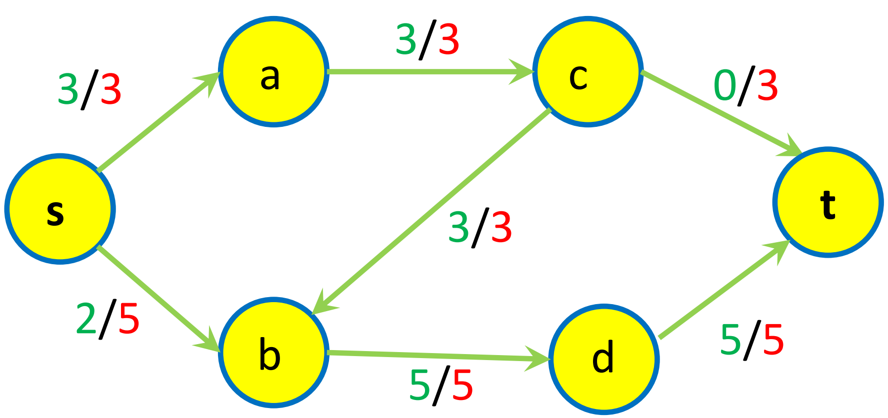
Update residual to match new flows.

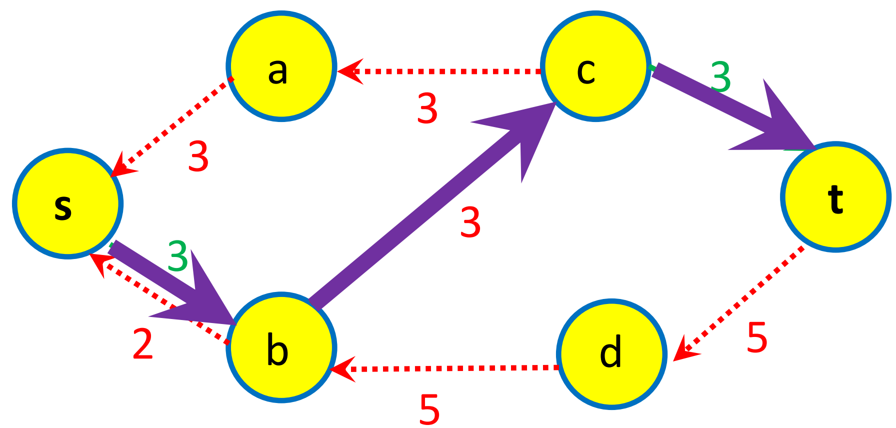
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
    Set initial flow f to 0 on all edges
    while there exists an augmenting path p in the residual network G_f **do**
        Augment the flow f along the augmenting path p
    return f

Total flow: 5





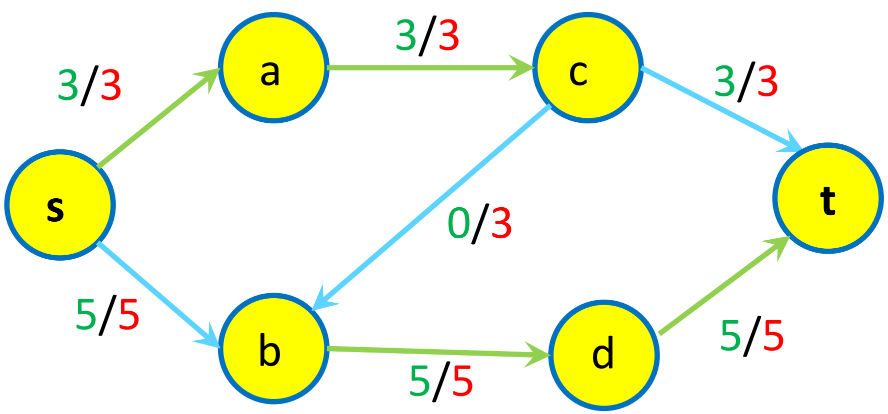
Path found: capacity = 3.

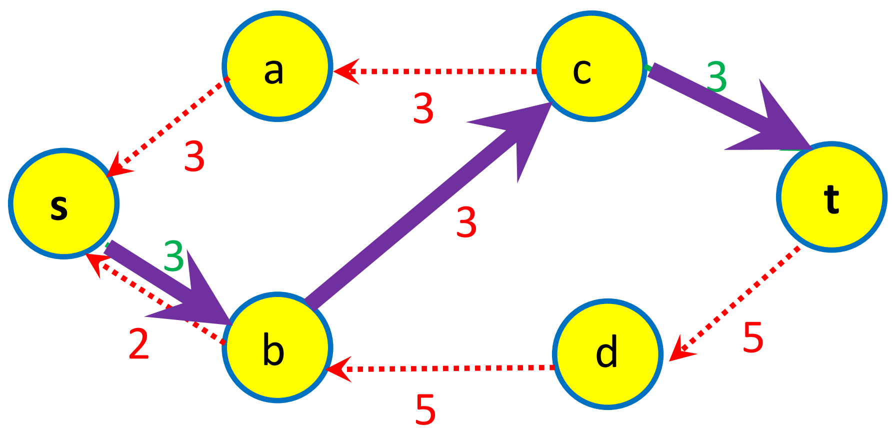
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW$(G = (V, E), s, t)$
    Set initial flow f to 0 on all edges
    while there exists an augmenting path p in the residual network G_f **do**
        Augment the flow f along the augmenting path p
    return f

Total flow: 8





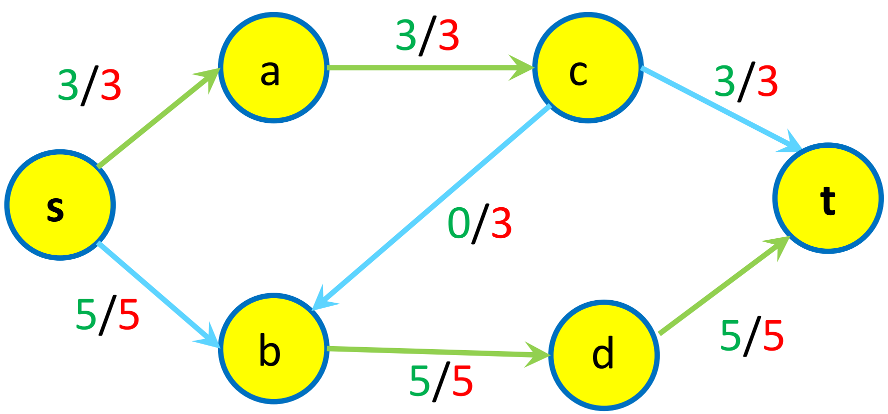
Path found: capacity = 3.

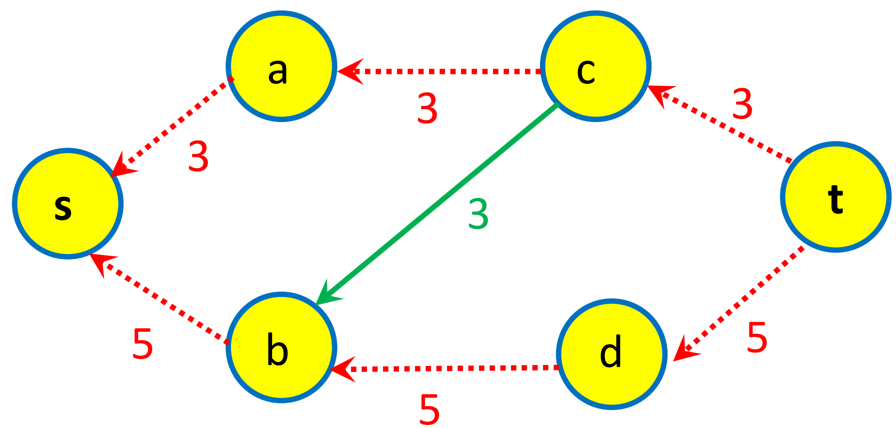
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
    Set initial flow f to 0 on all edges
    while there exists an augmenting path p in the residual network G_f **do**
        Augment the flow f along the augmenting path p
    return f

Total flow: 8





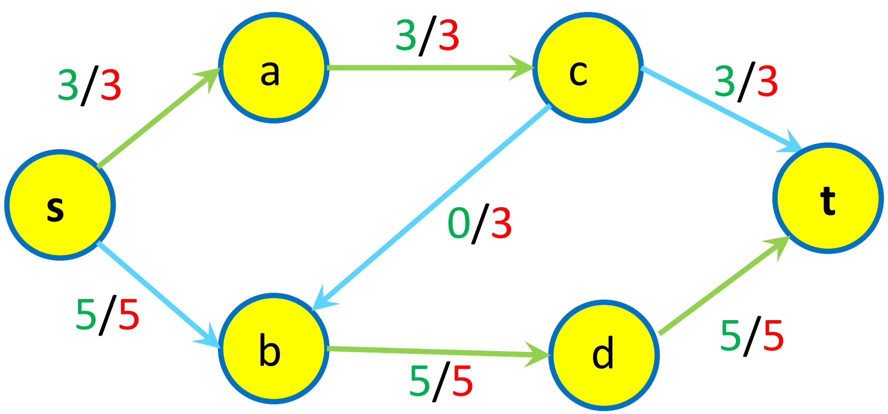
Update residual to match new flows.

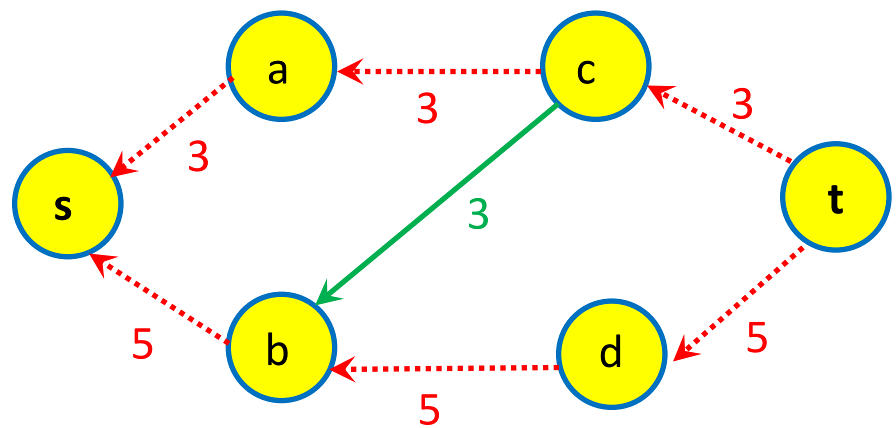
Ford-Fulkerson Method – Example

Algorithm 75 The Ford-Fulkerson method

function MAX_FLOW($G = (V, E), s, t$)
    Set initial flow f to 0 on all edges
    while there exists an augmenting path p in the residual network G_f **do**
        Augment the flow f along the augmenting path p
    return f

Total flow: 8





No more augmenting paths!

Complexity Analysis

Algorithm 75 The Ford-Fulkerson method

```
function MAX_FLOW(G = (V, E), s, t)
    Set initial flow f to 0 on all edges
    while there exists an augmenting path p in the residual network G_f do
        Augment the flow f along the augmenting path p
    return f
```

Cost of finding an augmenting path:

- Using BFS, cost is O(V+E) (or just O(E) since the graph is connected).

Augmenting flow along a path:

- length of path ≤ V, so cost is O(V).

Updating the residual:

- Same amount of work as updating the flows along the augmenting path, O(V).

Total work in one iteration of the loop:

- **O(V+E) = O(E).**

Quiz time!
https://flux.qa/YTJMAZ

Complexity Analysis

Algorithm 75 The Ford-Fulkerson method

```
function MAX_FLOW(G = (V, E), s, t)
    Set initial flow f to 0 on all edges
    while there exists an augmenting path p in the residual network G_f do
        Augment the flow f along the augmenting path p
    return f
```

Total work in one iteration of the loop:

- **O(V+E) = O(E).**

How many iterations?

- Assuming integer flows and capacities…
- Each iteration, flow grows by at least 1.
- If maximum flow in graph is F.
- Maximum number of iterations is also F.

Total work:

- **O(EF)**

Complexity Analysis

Algorithm 75 The Ford-Fulkerson method

```
function MAX_FLOW(G = (V, E), s, t)
    Set initial flow f to 0 on all edges
    while there exists an augmenting path p in the residual network G_f do
        Augment the flow f along the augmenting path p
    return f
```

Total work:

- **O(EF)**

Not examinable:

- This looks polynomial.
- But it isn't because F is a number, so its **value** is exponential in the space required to store it.
- The **Edmonds-Karp algorithm (1972)** it achieves **O(VE²)**. This is a refined version of Ford-Fulkerson, uses BFS to finds *shortest augmenting paths*.
- Another (earlier) refinement, **Dinic's Algorithm (1970)** achieves **O(V²E).**

Proof of Correctness

Algorithm 75 The Ford-Fulkerson method

```
function MAX_FLOW(G = (V, E), s, t)
    Set initial flow f to 0 on all edges
    while there exists an augmenting path p in the residual network G_f do
        Augment the flow f along the augmenting path p
    return f
```

- **Does the algorithm terminate?**
  - Yes (assuming all capacities are integers), because:
    - The flow always increases by at least 1 per iteration and there cannot be any augmenting path if all source's outgoing edges (similarly, if all sink's incoming edges) are saturated.

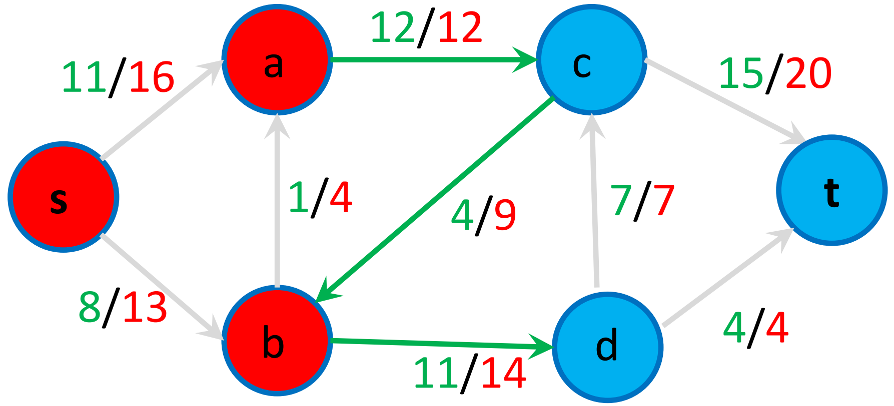
- **In order to show that the algorithm finds the *maximum* feasible flows, we will need to study the Min-cut Max-flow Theorem.**

Outline

1. Maximum Flow Problem
2. Ford-Fulkerson Algorithm
3. Min-cut Max-flow Theorem

Flow and Capacity of a Cut

- A cut (S,T) of a flow network partitions the vertices of the network into two disjoint partitions S and T such that source s is in S and target t is in T.
  - E.g., S = {s,a,b} and T = {t, c, d}.
- The cut-set of a cut (S,T) is the set of edges that "cross" the cut, i.e., each edge connects one vertex in S with another in T.
  - E.g., the cut-set for the example is a→c, b→d, c→b (green edges).
  - The edges that have direction from a vertex in S to a vertex in T are called outgoing edges of the cut.
    - E.g., a→c and b→d are the outgoing edges of the cut.
  - The edges that have direction from a vertex in T to a vertex in S are called incoming edges of the cut.
    - E.g., c→b is an incoming edge of the cut.



Flow and Capacity of a Cut

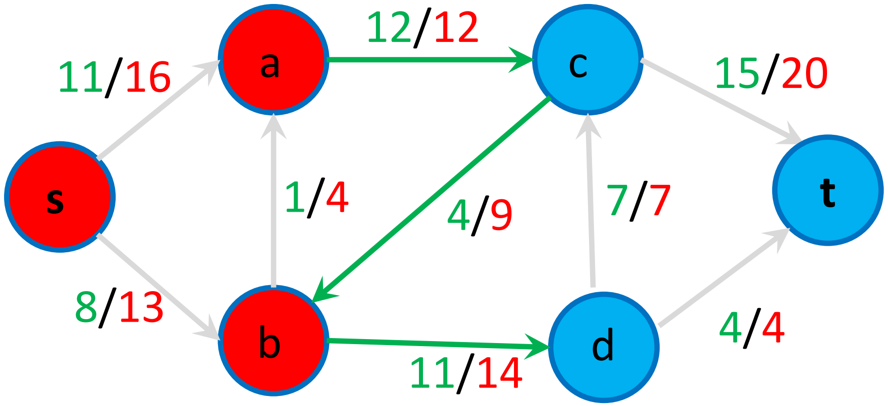
- Capacity of a cut (S,T) is **just** the total capacity of its **outgoing** edges.
  - E.g., capacity of the cut in the example is 12 + 14 = 26.

- Flow of a cut (S,T) is
  total flow of outgoing edges minus total flow of incoming edges.
  - E.g., flow in the example is 12 + 11 – 4 = 19.

cut flow <= cut capacity

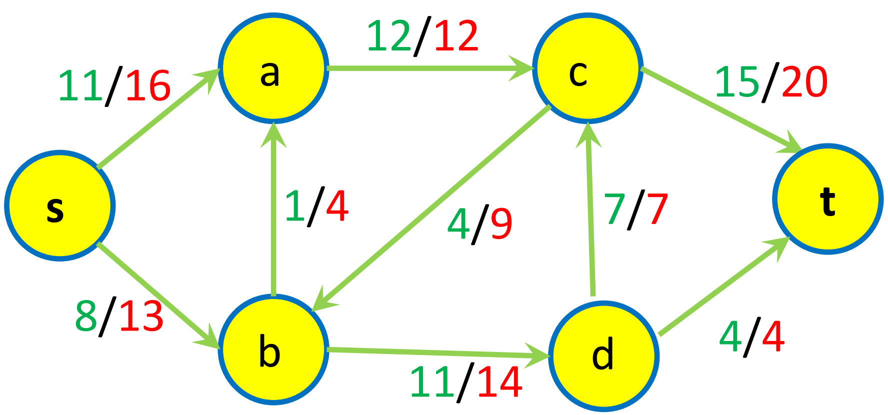
because:

- Flow of an edge ≤ capacity of an edge.
- Second term (flow of incoming edges) is negative.



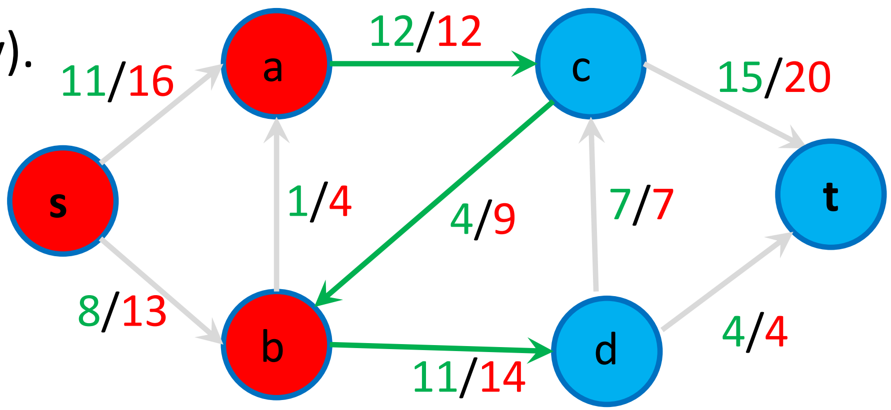
Flow and Capacity of a Cut

- Capacity of a cut (S,T): the total capacity of its outgoing edges.
- Flow of a cut (S,T): Total flow of its outgoing edges – total flow of its incoming edges.
- Assume S = {s, a,b,c,d} and T = {t}.
    - What is the capacity of this cut?
    - What is the flow of this cut?
- Assume S = {s, a, b,d} and T = {c,t}.
    - What is the capacity of this cut?
    - What is the flow of this cut?
- Assume S = {s, a} and T = {b,c,d,t}.
    - What is the capacity of this cut?
    - What is the flow of this cut?
- What is the flow value of this network?
- Note: flow of all of the above cuts is 19.
    - which is the same as flow of the network.
- I.e., flow of **every** cut = flow of the network.
- Let's prove this formally.



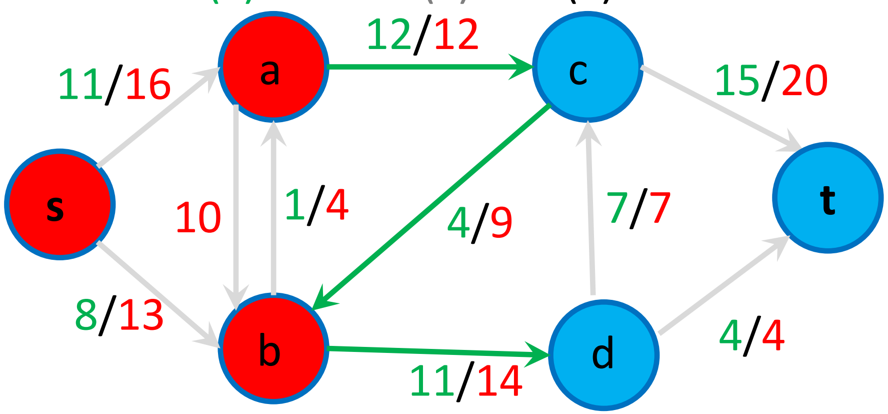
Flow of a Cut = Flow of the Network

- Let $F^{out}(v)$ be the total flow going out of a vertex and $F^{in}(v)$ be the total flow coming in the vertex.
- Recall that flow of a network is the total flow going out from the source s.
  - Flow of the network = $F^{out}(s)$.
- Flow conservation property: $F^{out}(v) - F^{in}(v) = 0$ for every vertex except s and t.
- Flow of the network = $F^{out}(s)$.
- Flow of the network = $F^{out}(s) + \sum_{v \in S \setminus s}(F^{out}(v) - Fin(v))$.
  - The second term is zero (conservation)
  - recall S is the cut containing s and excluding t.
- Since $F^{in}(s) = 0$, we can rewrite the flow as:
- Flow of the network = $\sum_{v \in S} F^{out}(v) - Fin(v)$.



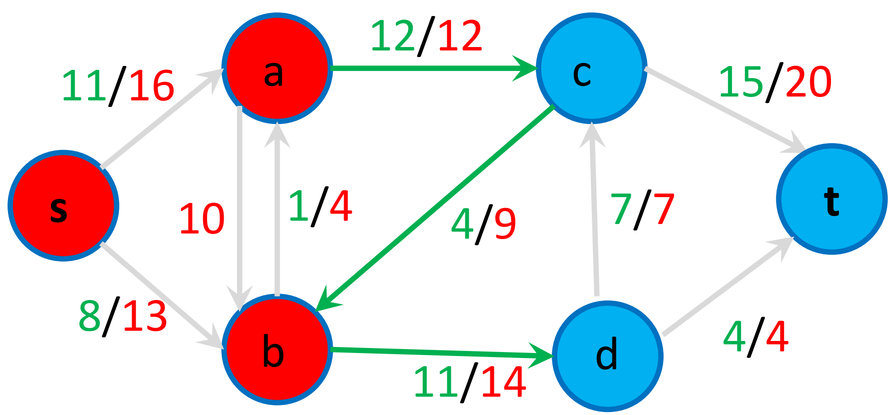
Flow of a Cut = Flow of the Network

- Flow of the network = $\sum_{v \in S} F^{out}(v) - F^{in}(v)$.
- We now check how the flow of S splits into internal flow and flow across the S-boundary
- Each vertex v in S (red vertices) can have two types of edges:
  - Grey edges (the edges that connect the vertex to another vertex in S).
  - Green edges (the edges that connect the vertex to a vertex in T).
  - Let $F^{out-grey}(v)$ be the total flow out from v via grey edges. Similarly, $F^{in-grey}(v)$ be the total flow coming to v via grey edges.
  - Let $F^{out-green}(v)$ be the total flow out from v via green edges. Similarly, $F^{in-green}(v)$ be the total flow coming to v via green edges.
  - We have $F^{out-green}(v) + F^{out-grey}(v) = F^{out}(v)$ and $F^{in-green}(v) + F^{in-grey}(v) = F^{in}(v)$.



Flow of a Cut = Flow of the Network

- We have $F^{out-green}(v) + F^{out-grey}(v) = F^{out}(v)$ and $F^{in-green}(v) + F^{in-grey}(v) = F^{in}(v)$.
- From before, flow of the network = $\sum_{v \in S} F^{out}(v) - Fin(v)$.
- Flow of the network = $\sum_{v \in S} F^{out-green}(v) + F^{out-grey}(v) - (F^{in-green}(v) + F^{in-grey}(v))$.
- Flow of the network = $\sum_{v \in S} F^{out-green}(v) - Fin^{-green}(v) + F^{out-grey}(v) - F^{in-grey}(v)$.
- Note that $\sum_{v \in S} F^{out-grey}(v) - F^{in-grey}(v) = 0$ because each grey edge appears once as an incoming edge for one vertex and once as an outgoing edge for another vertex.
- Flow of the network = $\sum_{v \in S} F^{out-green}(v) - Fin^{-green}(v)$.
- Flow of the network = Flow of any cut.



Min-Cut Max-Flow Theorem

- Min-cut of a flow network is the cut with the minimum capacity.

We know that:

1. Flow of any cut C ≤ capacity of C.
2. Flow of **every** cut (which are all the same) = Flow of the network.

- Therefore, maximum possible flow of the network ≤ capacity of any cut C.

- Specifically, maximum possible flow of the network ≤ capacity of min-cut.

Min-Cut Max-Flow Theorem

- Maximum possible flow of the network ≤ capacity of min-cut.

- What if we can find a pair of flow and cut such that the flow of the network = capacity of the cut?
  - We cannot increase the flow any further, because then we would violate the capacity of that cut.
  - We have found the maximum flow.
  - The cut is the min-cut of the flow network.

The Min-cut Max-flow Theorem states that:

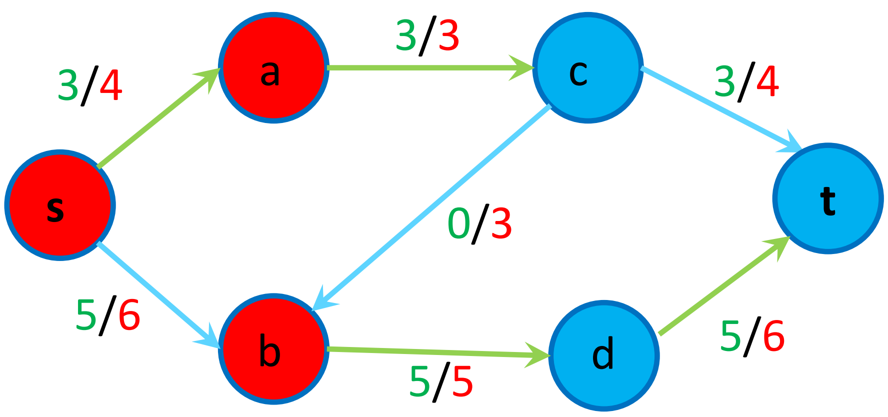
- Maximum possible flow of a network = capacity of the min-cut.

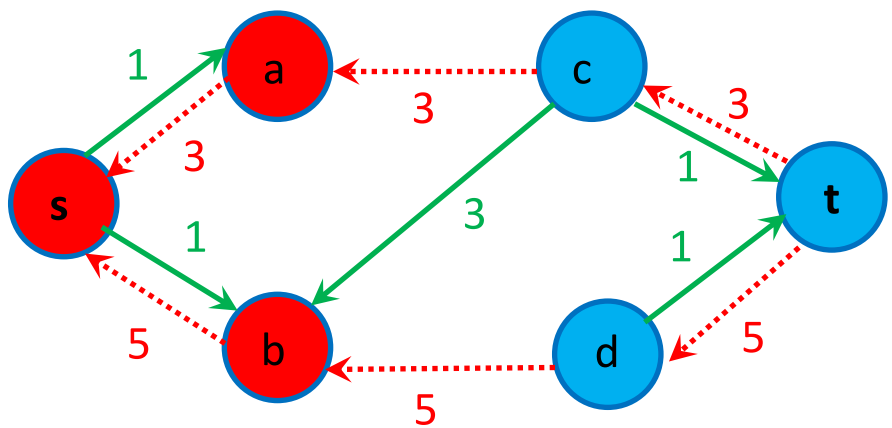
So far, we have only shown that it is not possible for the flow to exceed the capacity of the min-cut.

We still need to show that it is always possible to obtain a valid flow whose value matches the capacity of the min-cut. We will prove this by showing that the Ford-Fulkerson algorithm always terminates and outputs a flow with value equal to the capacity of the min-cut.

Proof of correctness

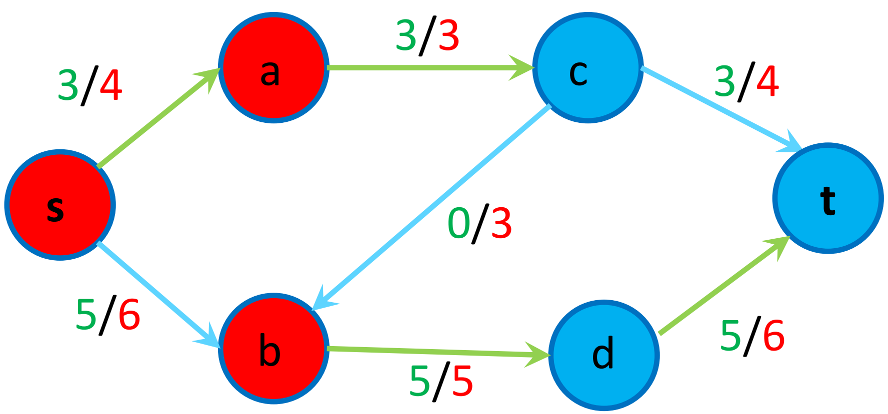
- Suppose the algorithm has terminated (there does not exist any augmenting path in the residual network).
- We define a cut (**S**,T) such that:
  - **S** contains every vertex v that is reachable from s in **the residual network** (*edge direction matters*!)
  - **T** contains every other vertex.
  - Note t cannot be in S because it is not reachable from S (no augmenting path).

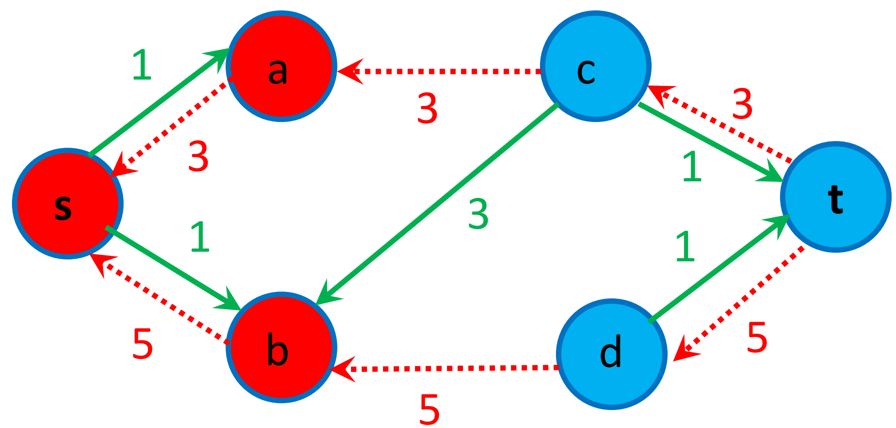




Proof of correctness

- Flow of this cut = Capacity of this cut because:
  - For each outgoing edge (eg, a→c), its flow is equal to the capacity of the edge.
    - Otherwise, we would have an edge (eg a→c) in the residual network which would mean t is reachable from s but we know this is not the case.
  - For each incoming edge (eg, c → b), its flow is zero.
    - Otherwise, there would be an edge (eg b→c) in the residual network implying t is reachable from s but we know this is not the case.
- Therefore, the flow is maximum when the algorithm terminates.





Summary

Take home message

- Maximum flow of a network is equal to its min-cut and can be found using Ford-Fulkerson.

Things to do (this list is not exhaustive)

- **Make sure you understand Ford-Fulkerson algorithm and why it is correct.**
- Start preparing for the final exam.

Coming Up Next

- Topological sorting and final exam.
- Revision – Enter your questions in Flux Q&A before the end of the week, I will leave this running