University of Pittsburgh

Algorithms and Data Structures 2
CS 1501

Pitt SCI 5 YEARS

Fall 2022

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

Announcements

- Upcoming Deadlines
  - Lab 9 and Homework 9:
    - reopened till this Friday @ 11:59 pm
  - Homework 10: this Friday 12/2 @ 11:59 pm
  - Lab 11: Monday 12/5 @ 11:59 pm
  - Homework 11: Friday 12/9 @ 11:59 pm
  - Assignment 3: ~~Monday 11/28~~ Friday 12/9 @ 11:59 pm
  - Assignment 4: Friday 12/9 @ 11:59 pm

Previous Lecture

- Dynamic Programming Examples
  - Subset Sum
  - Edit Distance
  - Longest Common Subsequence

This Lecture

- Back to Graph Algorithms
- Network Flow Problem

Muddiest Points

- **Q: I would like another example of the edit distance algorithm. I did not quite understand what was going on too well**
- Sure!

Muddiest Points

- **Q: Subset Sum Problem**
- Let’s have another example!

Muddiest Points

- **Q: LCS**
- Let’s have another example!

Problem of the Day: Finding Bottlenecks

- Let's assume that we want to send a large file from point A to point B over a computer network as fast as possible over multiple network links if needed
- Input:
  - A computer network
    - Network nodes and links
    - Links are labeled by link capacity in Mbps
  - Starting node and destination node
- Output:
  - The maximum network speed possible for sending a file from source to destination

Defining network flow

- Consider a directed, weighted graph G(V, E)
  - Weights are applied to edges to state their *capacity*
    - c(u, w) is the capacity of edge (u, w)
    - if there is no edge from u to w, c(u, w) = 0
- Consider two vertices, a *source s* and a *sink t*
  - Let's determine the maximum flow that can run from s to t in the graph G

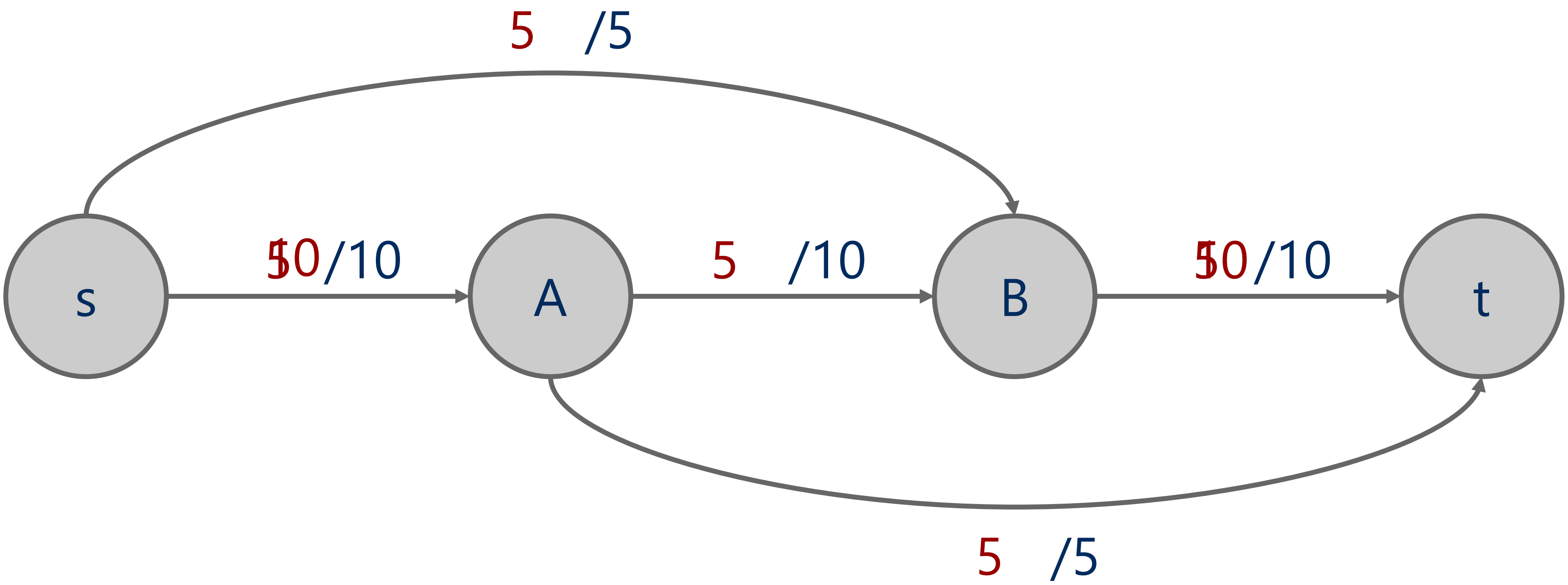
Flow

- Let the $f(u, w)$ be the amount of flow being carried along the edge (u, w)
- Some rules on the flow running through an edge:
  - $\forall (u, w) \in E \; f(u, w) <= c(u, w)$
  - $\forall u \in (V - \{s,t\}) \; (\Sigma_{w \in V} f(w, u) - \Sigma_{w \in V} f(u, w)) = 0$

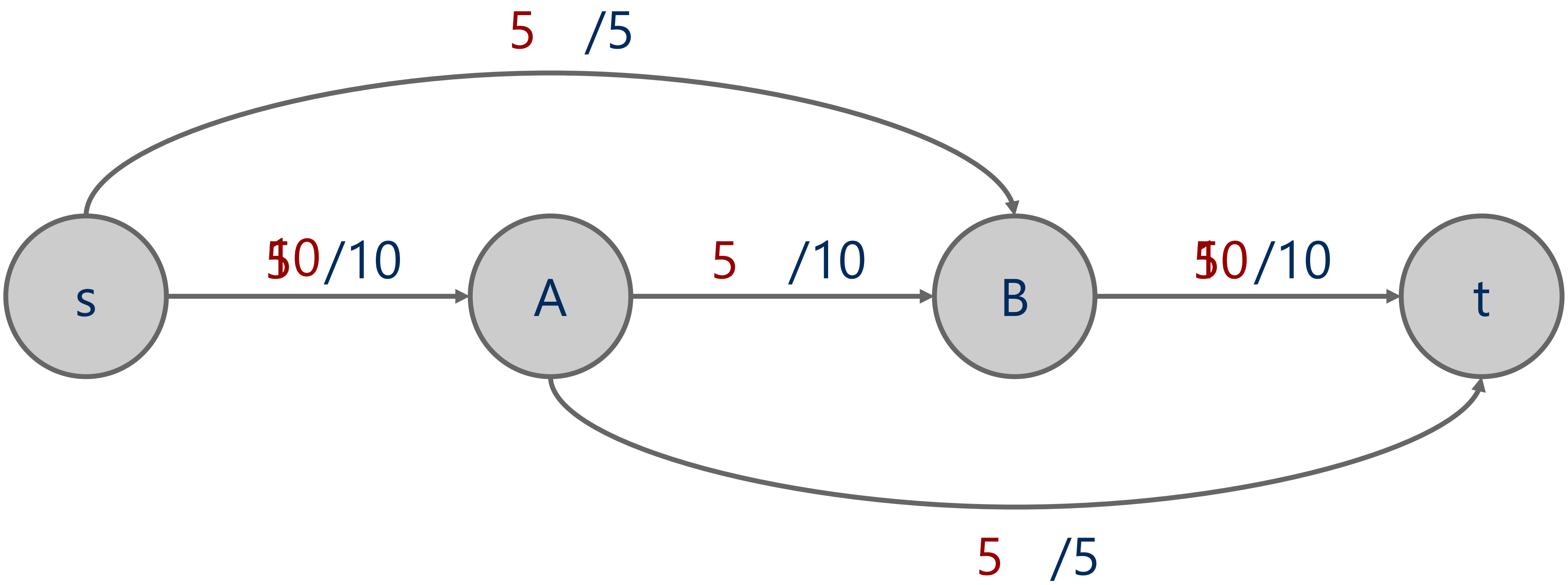
Ford Fulkerson

- Let all edges in G have an allocated flow of 0
- While there is path *p* from s to t in G s.t. all edges in p have some *residual capacity* (i.e., $\forall(u, w) \in p\ f(u, w) < c(u, w)$):
    - (Such a path is called an *augmenting path*)
    - Compute the residual capacity of each edge in p
        - Residual capacity of edge (u, w) is c(u, w) - f(u, w)
    - Find the edge with the minimum residual capacity in p
        - We'll call this residual capacity *new_flow*
    - Increment the flow on all edges in p by new_flow

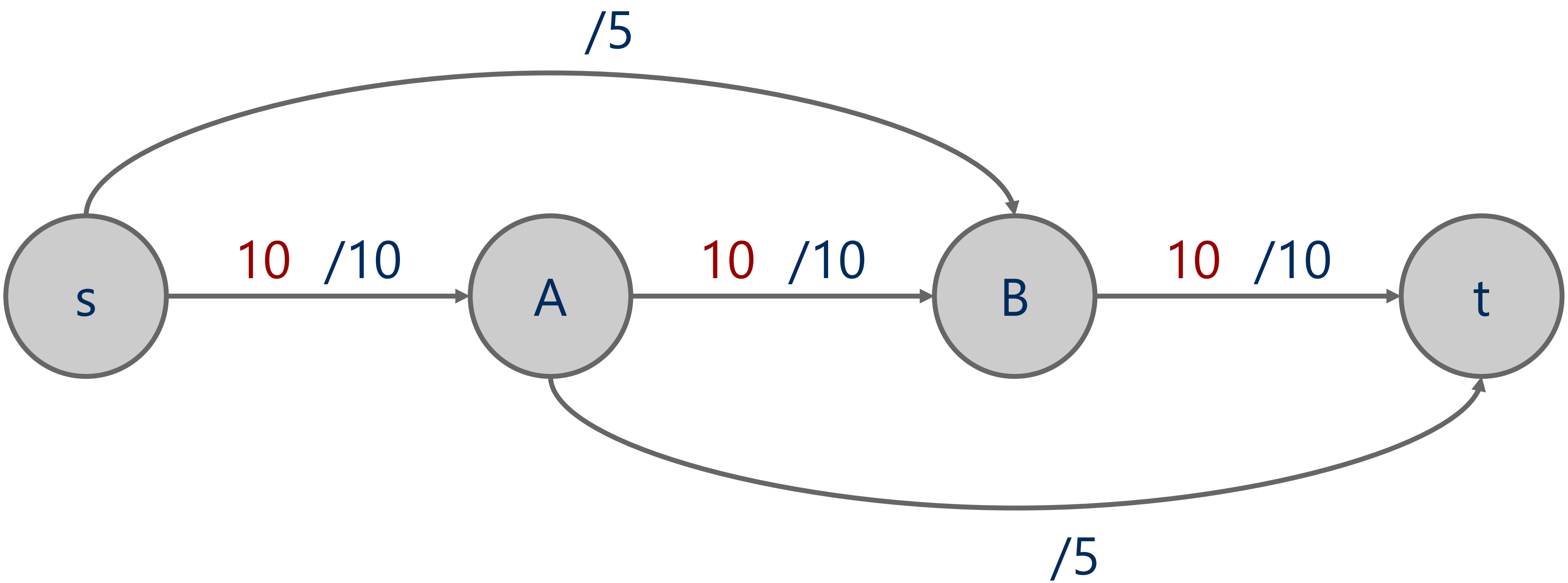
Ford Fulkerson example



Ford Fulkerson example



Another Ford Fulkerson example



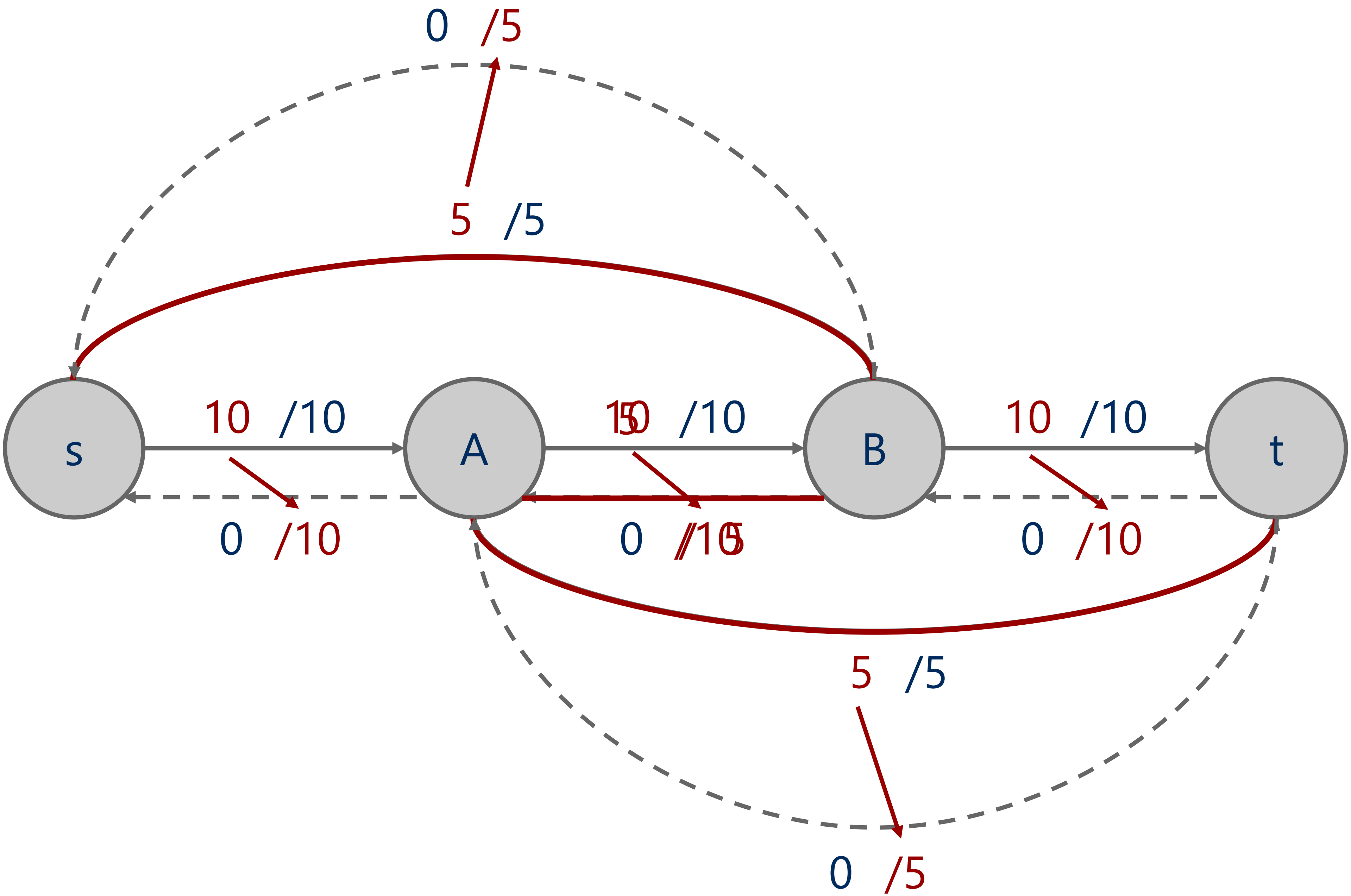
Expanding on residual capacity

- To find the max flow we will have to consider re-routing flow we had previously allocated
    - This means, when finding an augmenting path, we will need to look not only at the edges of G, but also at *backwards edges* that allow such re-routing
        - For each edge (u, w) ∈ E, a backwards edge (w, u) must be considered during pathfinding if f(u, w) > 0
            - The capacity of a backwards edge (w, u) is equal to f(u, w)

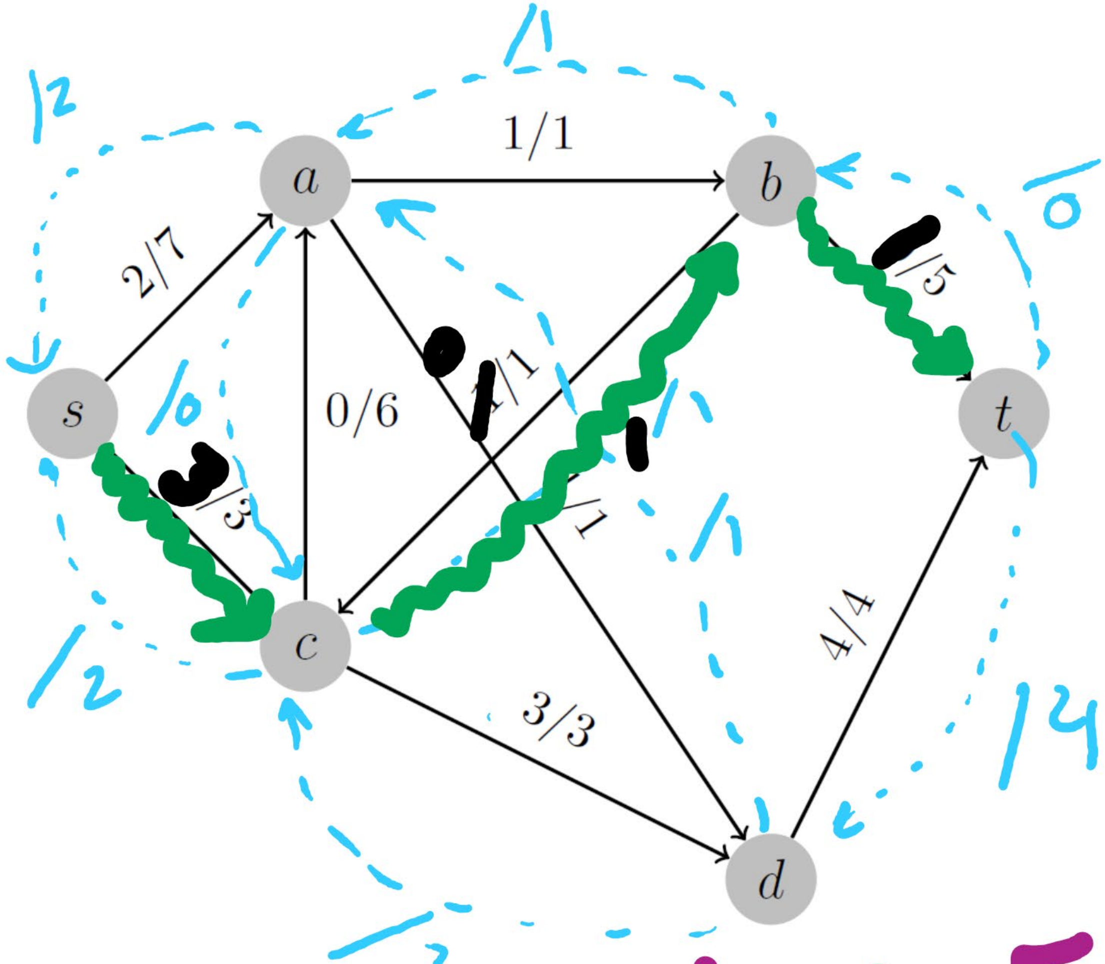
The residual graph

- We will perform searches for an augmenting path not on G, but on a residual graph built using the current state of flow allocation on G
- The residual graph is made up of:
    - V
    - An edge for each (u, w) ∈ E where f(u, w) < c(u, w)
        - (u, w)'s mirror in the residual graph will have 0 flow and a capacity of c(u, w) - f(u, w)
    - A backwards edge for each (u, w) ∈ E where f(u, w) > 0
        - (u, w)'s backwards edge has a capacity of f(u, w)
        - All backwards edges have 0 flow

Residual graph example



2nd Tophat Question



$s \xrightarrow{1} c \xrightarrow{1} b \xrightarrow{5} t$

Backwards edges

- Adding flow to a backwards edge means rerouting flow from the corresponding forward edge