CSC 373: Algorithm Design and Analysis
Lecture 9

Allan Borodin

January 28, 2013

Lecture 9: Announcements and Outline

Announcements

- Problem set 1 due this Friday.
- Term Test 1 will be held next Monday, Feb 4 in the tutorials.
- Two announcements to follow about choosing a CS Focus and applying to Graduate School.

Today's outline

- Start flow networks
- Ford Fulkerson and augmenting paths
- Ford Fulkerson as a local search algorithm

Choosing a focus in CS

Computer Science
UNIVERSITY OF TORONTO

CHOOSE YOUR OWN ADVENTURE:
UNDERSTANDING THE ~~CONCENTRATIONS~~ FOCUSES

THURSDAY, FEBRUARY 7, 2013
11:00 am—1:00pm
ROOM: BA1180

HUMAN-COMPUTER INTERACTION,
COMPUTER SYSTEMS, GAME DESIGN...
So many choices!

Find out what each Focus is about— and where it might take you.
Get your questions answered by faculty experts in each Focus area.

For questions, contact ugliaison@cs.toronto.edu.

Thinking about Graduate School?

Computer Science
UNIVERSITY OF TORONTO

GRAD SCHOOL INFO SESSION

TUESDAY, FEBRUARY 12, 2013
11:00 am—1:00pm
ROOM: BA2135

THINKING ABOUT GRAD SCHOOL?

Get some answers!

- What is grad school is like?
  - How do I apply?
- How can I prepare as an undergraduate student?
  - How can I give myself the best edge—
    and get into the school of my dreams!

For questions, contact ugliaison@cs.toronto.edu.

Flow networks

- I will be following our old CSC364 lecture notes for the basic definitions and results concerning the computation of max flows.
- We follow the convention of allowing negative flows. While intuitively this may not seem so natural, it does simplify the development.
- The DPV and KT texts use the perhaps more standard convention of just having non-negative flows.

Definition

A flow network (more suggestive to say a capacity network) is a tuple $F = (G, s, t, c)$ where

- $G = (V, E)$ is a "bidirectional graph"
- the source s and the terminal t are nodes in V
- the capacity $c : E \to \mathbb{R}^{\geq 0}$

What is a flow?

A flow is a function $f : E \to \mathbb{R}$ satisfying the following properties:

1. Capacity constraint: for all $(u, v) \in E$,

$$f(u, v) \leq c(u, v)$$

2. Skew symmetry: for all $(u, v) \in E$,

$$f(u, v) = -f(v, u)$$

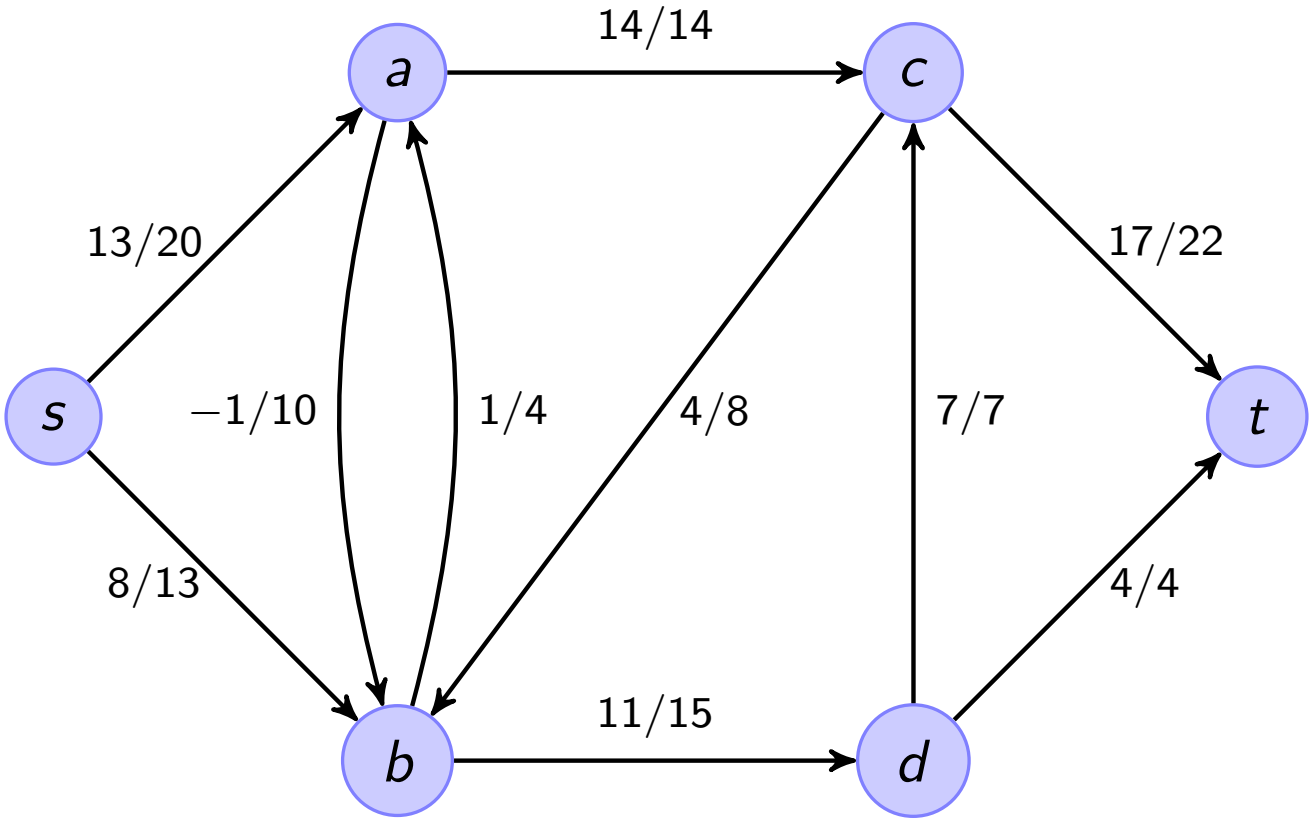
3. Flow conservation: for all nodes u (except for s and t),

$$\sum_{v \in N(u)} f(u, v) = 0$$

Note

Condition 3 is the "flow in = flow out" constraint if we were using the convention of only having non-negative flows.

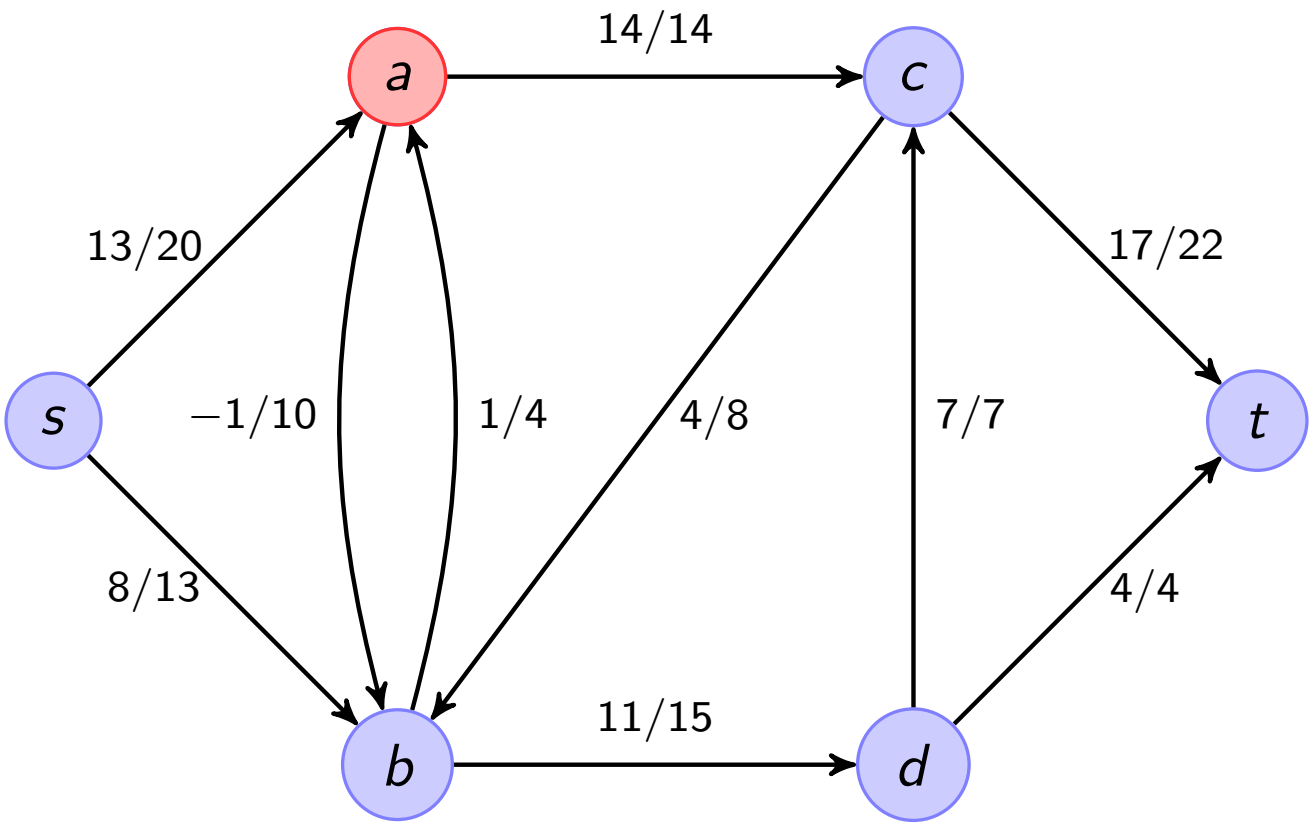
An example



The notation x/y on an edge (u, v) means

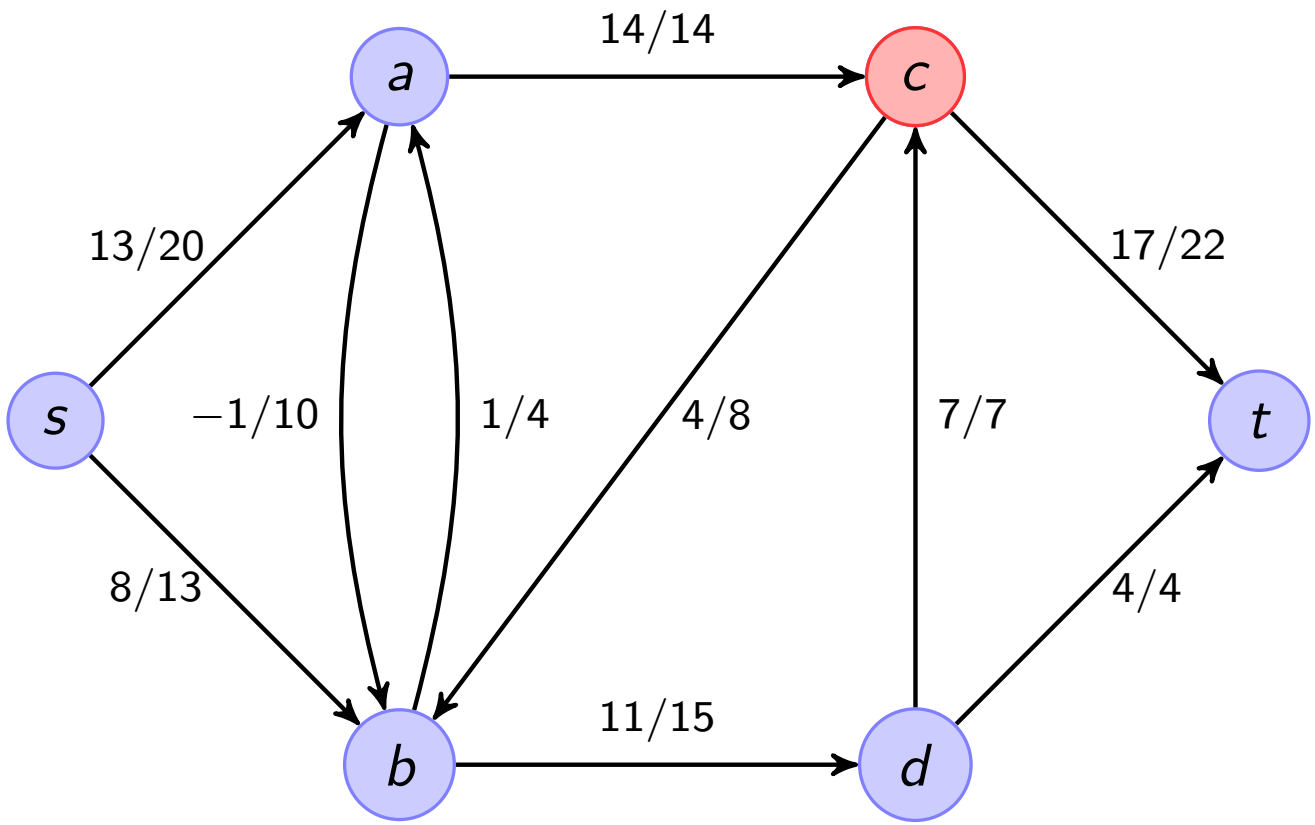
- x is the flow, i.e. $x = f(u, v)$
- y is the capacity, i.e. $x = c(u, v)$

An example of flow conservation



- For node a: $f(a,s) + f(a,b) + f(a,c) = -13 + (-1) + 14 = 0$

An example of flow conservation



- For node a: $f(a,s) + f(a,b) + f(a,c) = -13 + (-1) + 14 = 0$
- For node c:

$$f(c,a) + f(c,b) + f(c,d) + f(c,t) = -14 + 4 + (-7) + 17 = 0$$

The max flow problem

The max flow problem

Given a network flow, the goal is to find a valid flow that maximizes the flow out of the source node s.

- As we will see this is also equivalent to maximizing the flow in to the terminal node t. (This should not be surprising as flow conservation dictates that no flow is being stored in the other nodes.)
- We let $val(f)$ denote the flow out of the source s for a given flow f.
- We will study the Ford-Fulkerson augmenting path scheme for computing an optimal flow.
- I am calling it a "scheme" as there are many ways to instantiate this scheme although I dont view it as a general "paradigm" in the way I view (say) greedy and DP algorithms.

So why study Ford-Fulkerson?

- Why do we study the Ford-Fulkerson scheme if it is not a very generic algorithmic approach?
- As in DPV text, max flow problem can also be represented as a linear program (LP) and all LPs can be solved in polynomial time.
- I view Ford-Fulkerson and augmenting paths as an important example of a local search algorithm although unlike most local search algorithms we obtain an optimal solution.
- The topic of max flow (and various generalizations) is important because of its immediate application and many applications of max flow type problems to other problems (e.g. max bipartite matching).
  - That is many problems can be polynomial time transformed/reduced to max flow (or one of its generalizations).
  - One might refer to all these applications as "flow based methods".

A flow f and its residual graph

- Given any flow f for a flow network $F = (G, s, t, c)$, we define the residual graph $G_f = (V, E_f)$, where
  - V is the set of vertices of the original flow network F
  - E_f is the set of all edges e having positive residual capacity

$$c_f(e) = c(e) - f(e) > 0.$$

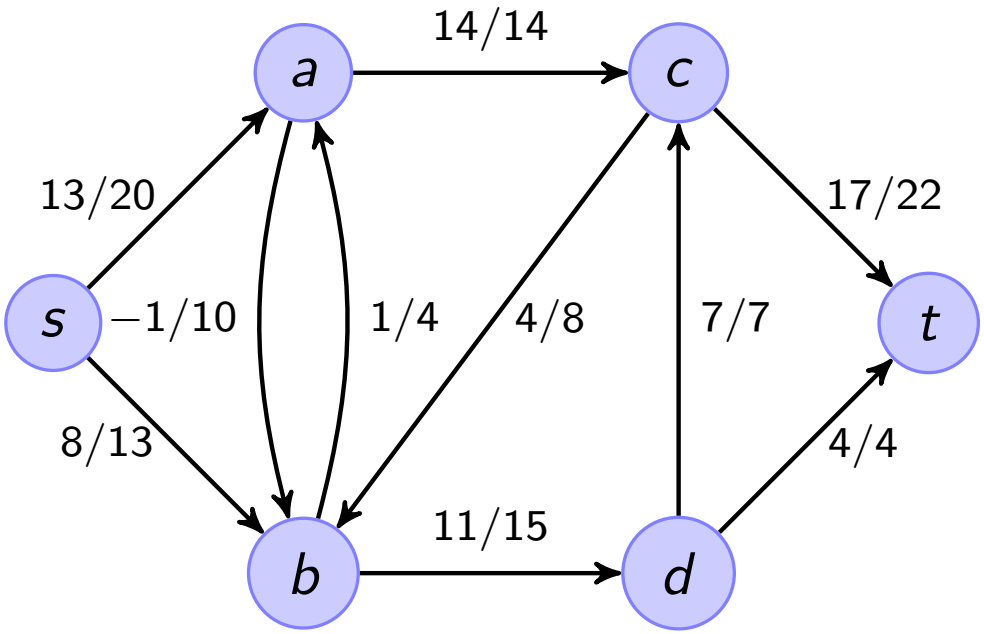
- Note that $c(e) - f(e) \geq 0$ for all edges by the capacity constraint.

Note

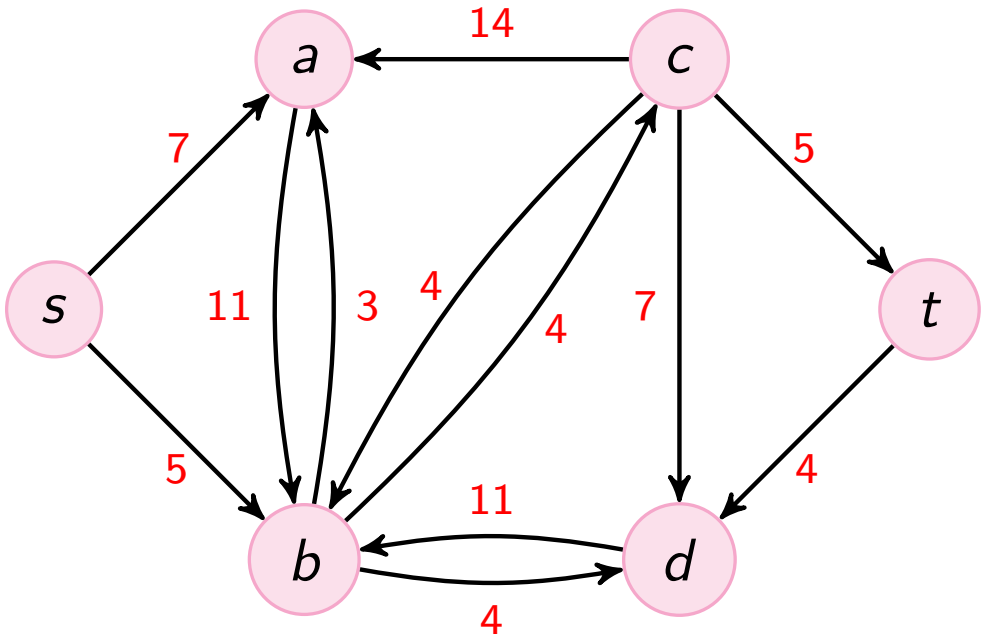
With our convention of negative flows, even a zero capacity edge (in G) can have residual capacity.

- The basic concept underlying the Ford-Fulkerson algorithm is an **augmenting path** which is an s-t path in G_f.
  - Such a path can be used to augment the current flow f to derive a better flow f'.

An example of a residual graph



The previous network flow



The residual graph

The residual capacity of an augmenting path

- Given an augmenting path π in G_f, we define its residual capacity $c_f(\pi)$ to be the
$$\min\{c(e) - c_f(e) \mid e \in \pi\}$$
- **Note:** the residual capacity of an augmenting path is itself is greater than 0 since every edge in the path has positive residual capacity.
- **Question:** How would we compute an augmenting path of maximum residual capacity?

Using an augmenting path to improve the flow

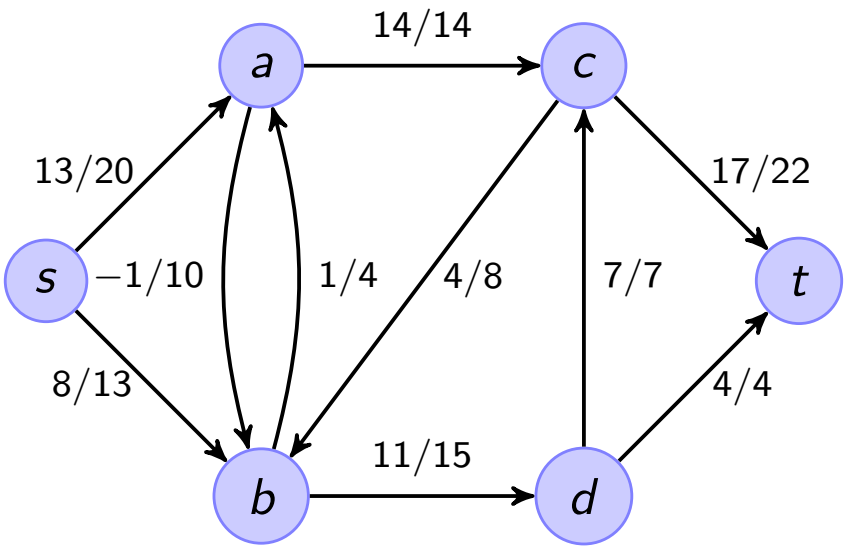
- We can think of an augmenting path as defining a flow f_π (in the "residual network"):

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

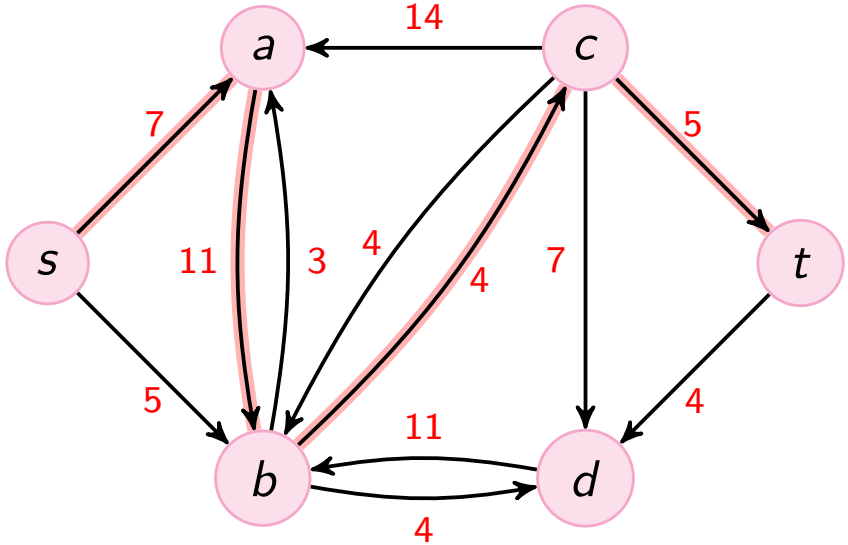
Claim

$f' = f + f_\pi$ is a flow in F and $val(f') > val(f)$

Deriving a better flow using an augmenting path

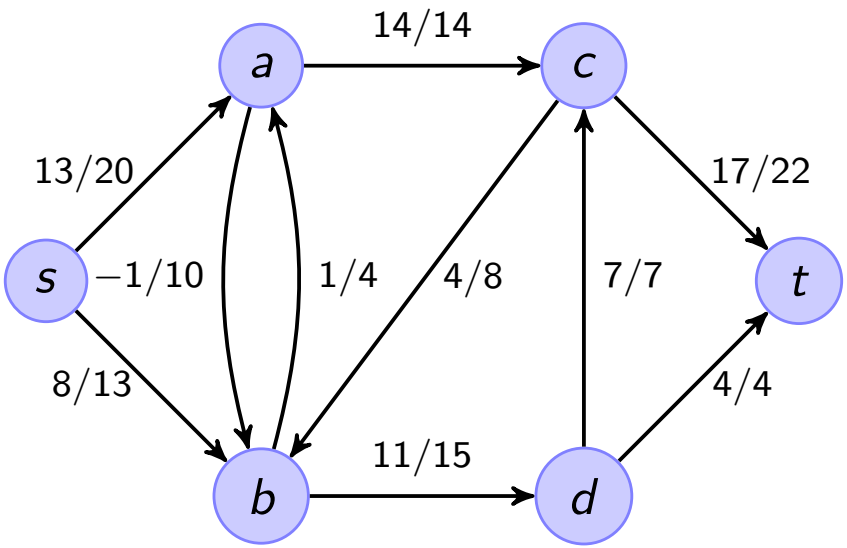


The original network flow

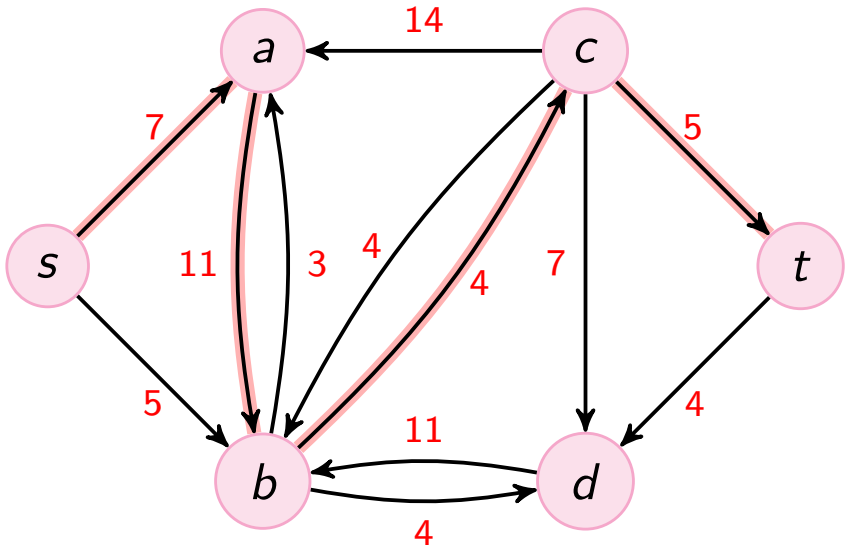


An augmenting path π with $c_f(\pi) = 4$

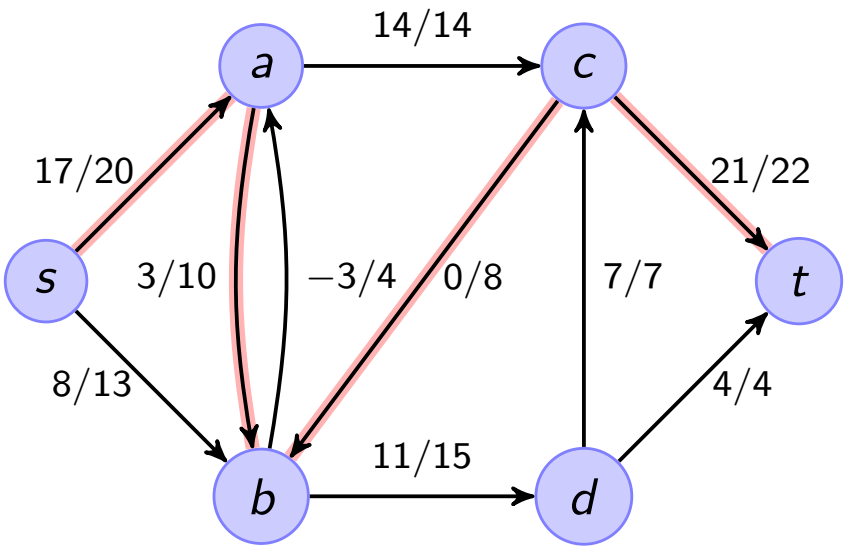
Deriving a better flow using an augmenting path



The original network flow

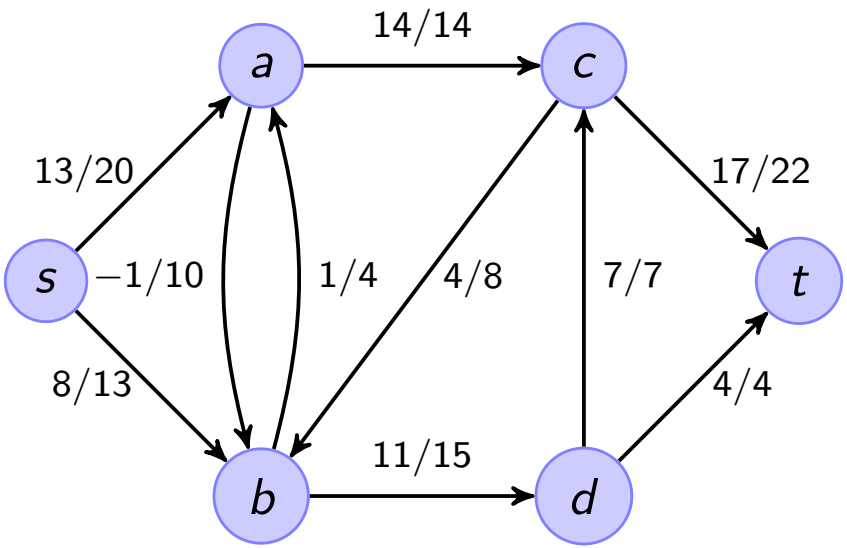


An augmenting path π with $c_f(\pi) = 4$

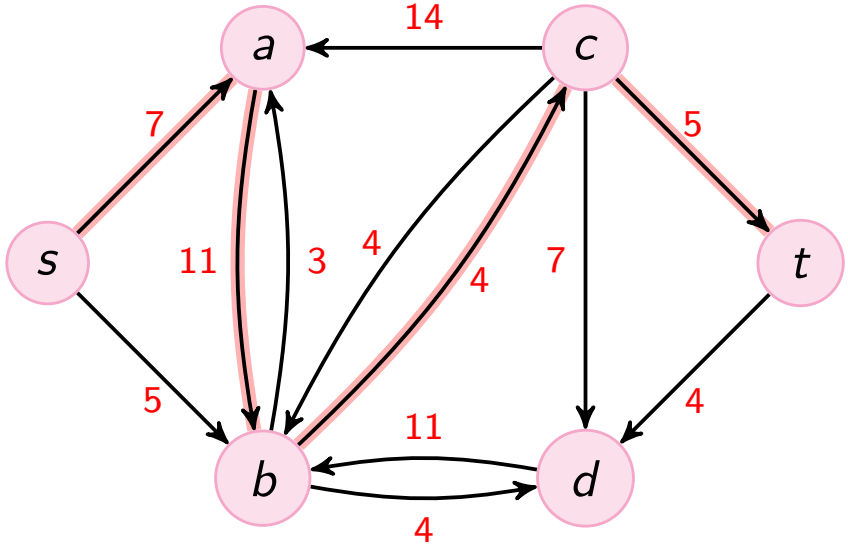


The updated flow whose value = 25

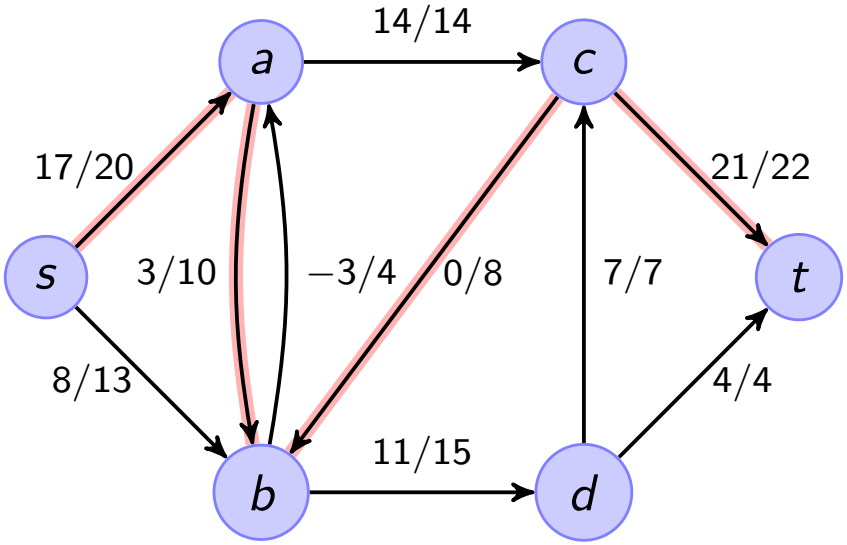
Deriving a better flow using an augmenting path



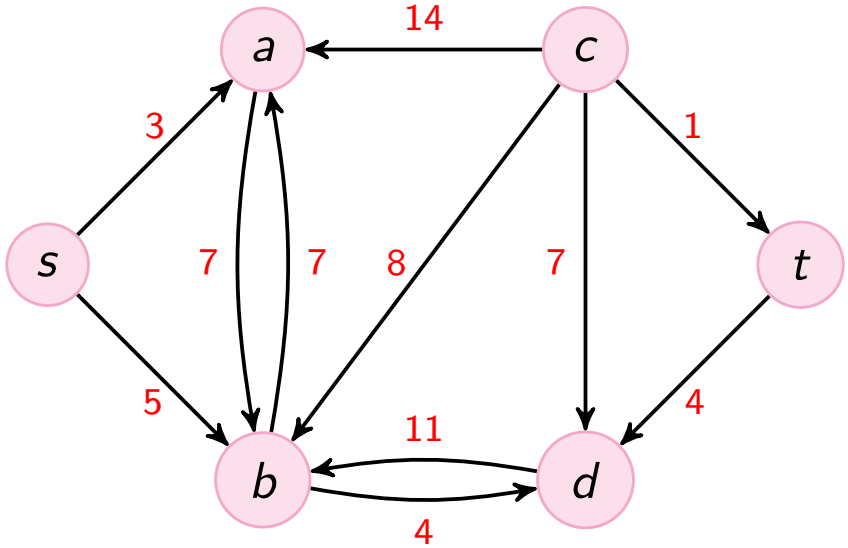
The original network flow



An augmenting path π with $c_f(\pi) = 4$



The updated flow whose value $= 25$



Updated res. graph with no aug. path

The Ford-Fulkerson scheme

The Ford-Fulkerson scheme

1: /* *Initialize* */
2: $f := 0$
3: $G_f := G$
4: **while** there is an augmenting path π in G_f **do**
5: $\quad f := f + f_\pi$ /* *Note this also changes* G_f */
6: **end while**

Note

I call this a "scheme" rather than an algorithm since we haven't said how one chooses an augmenting path (as there can be many such paths)