Week 10 - Monday

COMP 4500

Last time

- What did we talk about last time?
- Sequence alignment

Questions?

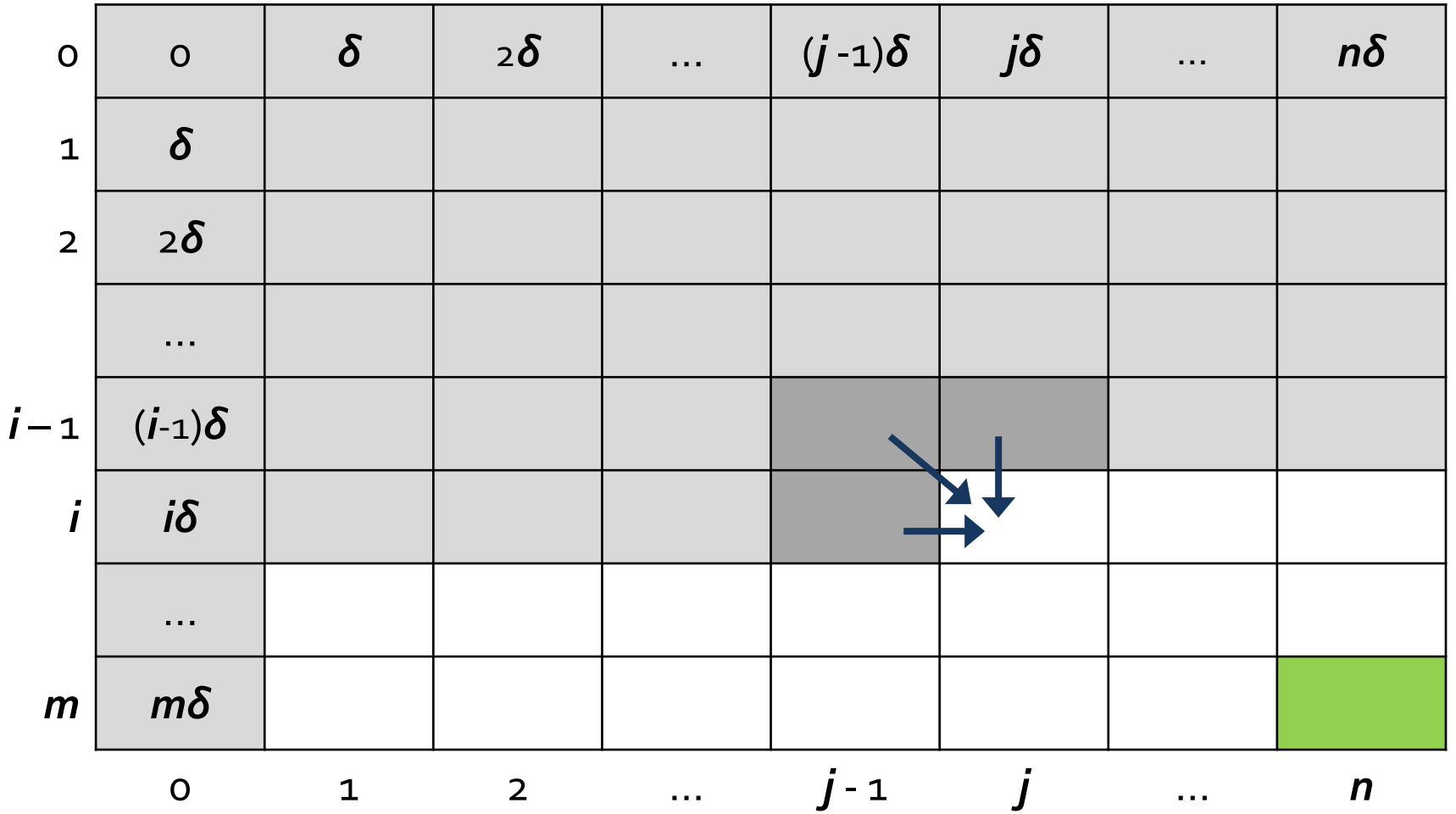
Assignment 5

Logical warmup

- A butcher goes to the market with $100
- He has to buy exactly 100 animals (for no especially good reason)
- There are cows, geese and chicken for sale
  - Cows are $15 each
  - Geese are $1 each
  - Chickens are $0.25 each
- He has to buy at least one of each animal and has to spend all his money
- What does the butcher buy?

Back to Sequence Alignment

Table *A* of OPT values



Sequence alignment example

- Find the minimum cost to align:
  - "machine"
  - "catching"
- The cost of an insertion (or deletion) δ is 1
- The cost of replacing any letter with a different letter is 1
- The cost of "replacing" any letter with itself is 0

Fill in the table

		m	a	c	h	i	n	e
	0	1	2	3	4	5	6	7
c	1							
a	2							
t	3							
c	4							
h	5							
i	6							
n	7							
g	8							

Three-sentence Summary of the Maximum-Flow Problem and the Ford-Fulkerson Algorithm and Minimum Cuts

Maximum Flow

Flow networks

- A **flow network** is a weighted, directed graph with positive edge weights
  - Think of the weights as **capacities**, representing the maximum units that can flow across an edge
  - It has a **source *s*** (where everything comes from)
  - And a **sink *t*** (where everything goes to)
- Some books refer to this kind of flow network specifically as an ***st***-flow network

Flow

- The definition of ***s*-*t*** flow is a function ***f*(*e*)** that maps a non-negative real number to each edge ***e*** meeting the following conditions:
  - **Capacity condition:** No edge has more flow mapped to it than its capacity ***c*(*e*)**, more formally $0 \leq f(e) \leq c(e)$
  - **Conservation condition:** Except for ***s*** and ***t***, the flow coming into a node ***v*** is the same as the flow going out
- These conditions are intuitive: You can't pump more through a pipe than its capacity and material doesn't accumulate at a pipe joint

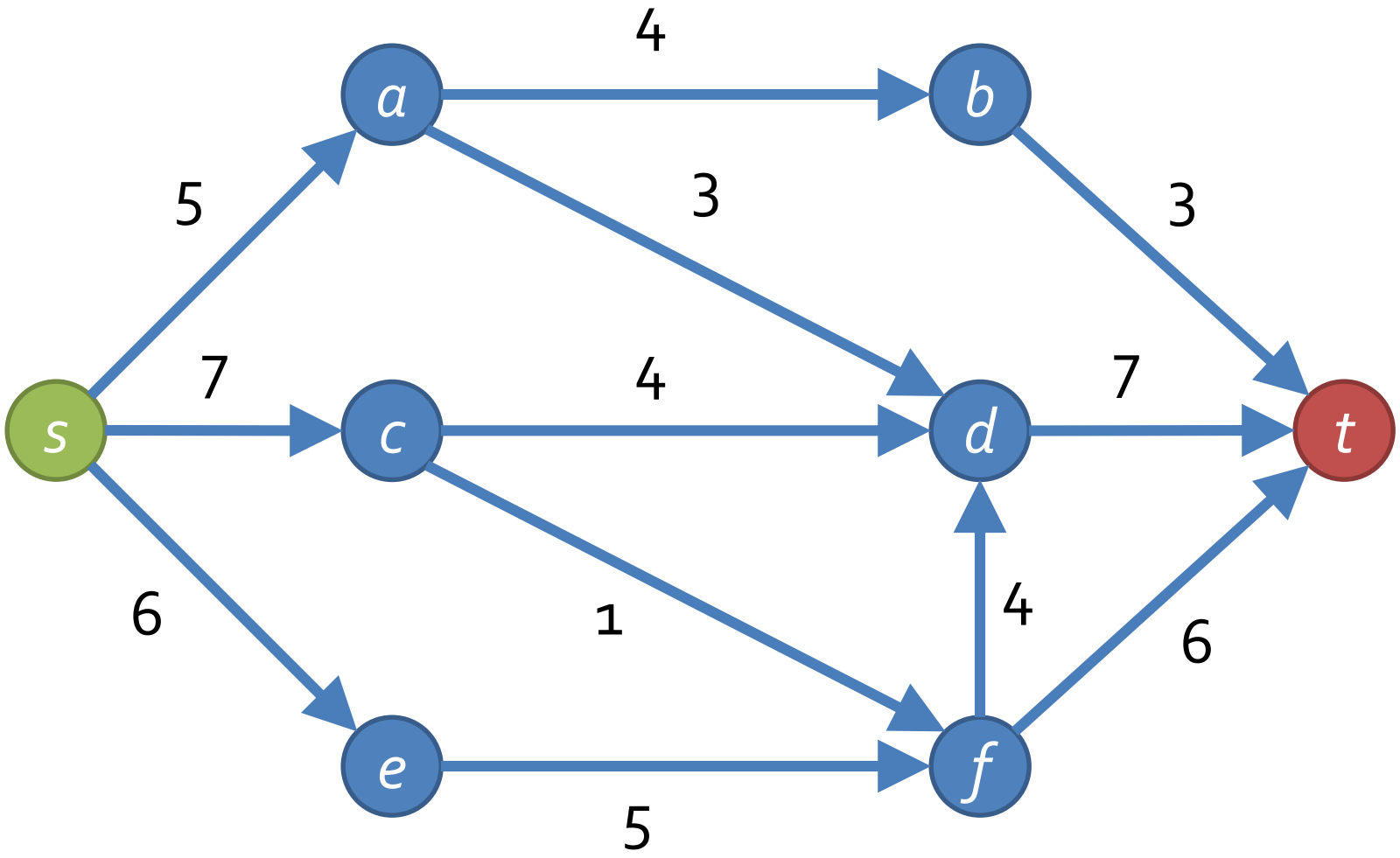
Applications of flow problems

- Oil flowing from a start to a destination
- Airline crews needed to man aircraft, moving from city to city
- Goods being produced by factories and consumed by cities, with roads that can accommodate a certain amount of traffic

Maximum flow

- A common flow problem is to find the **maximum flow**
- A maximum flow is a flow such that the amount leaving ***s*** and the amount going into ***t*** is as large as possible
- In other words:
  - The maximum amount of flow gets from ***s*** to ***t***
  - No edge has more flow than its capacity
  - The flow going into every node (except ***s*** and ***t***) is equal to the flow going out

Flow network



Augmenting path

- Our algorithm involves adding **augmenting paths** to our graph
- A flow augmenting path:
  - Starts at ***s*** and ends at ***t***
  - May cross some edges in the direction of the edge (forward edges)
  - May cross some edges in the opposite direction (backwards edges)
  - Increases the flow by the minimum of the unused capacity in the forward edges or the maximum of the flow in the backwards edges

Terminology

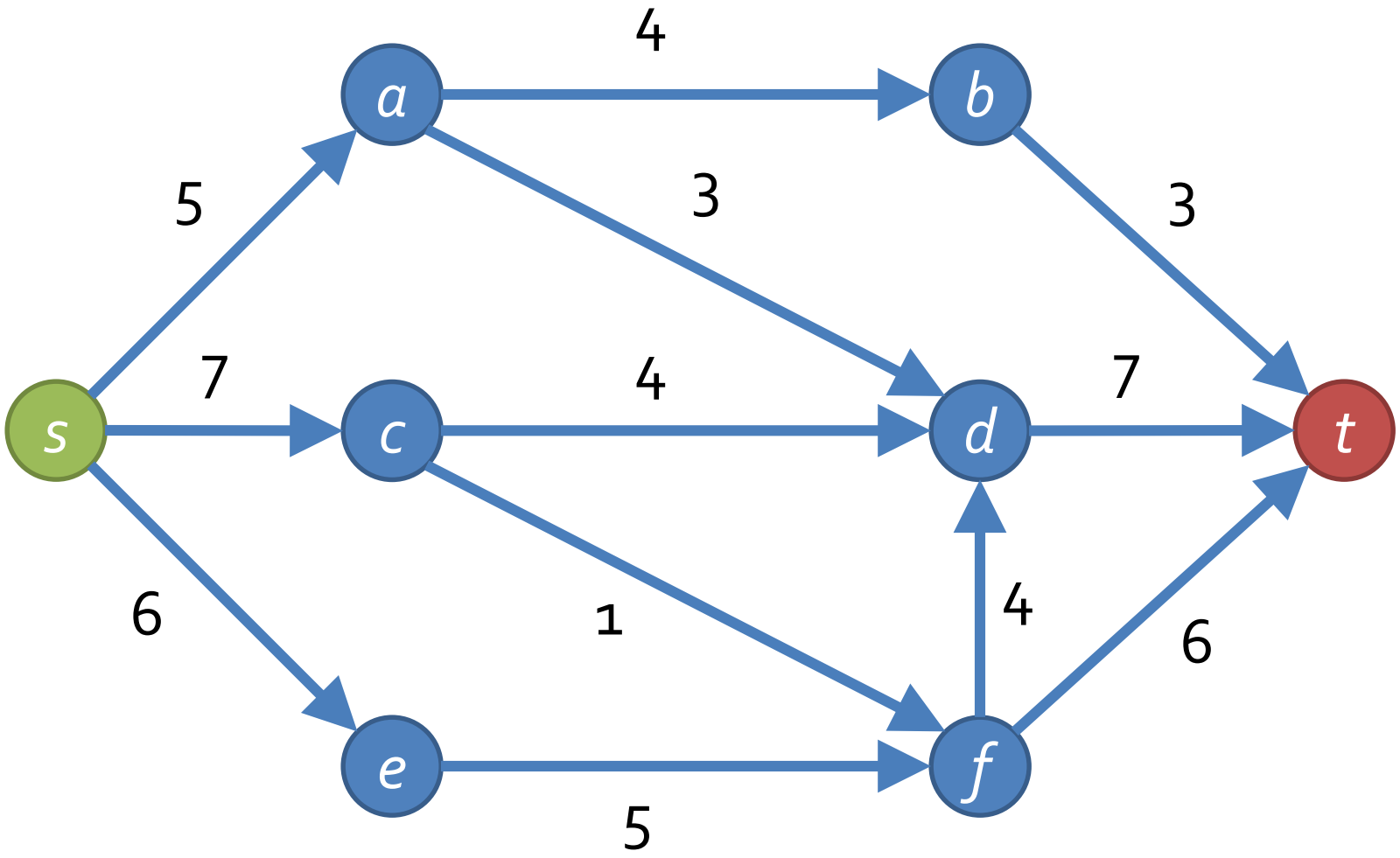
- When all the capacity of an edge is used, we say that edge is **saturated**
- After adding an augmenting path, we refer to the new graph (with updated flows) as the **residual graph**
- The capacity left on edges in the residual graph is its **residual capacity**

Ford-Fulkerson algorithm

- Ford-Fulkerson is a family of algorithms for finding the maximum flow

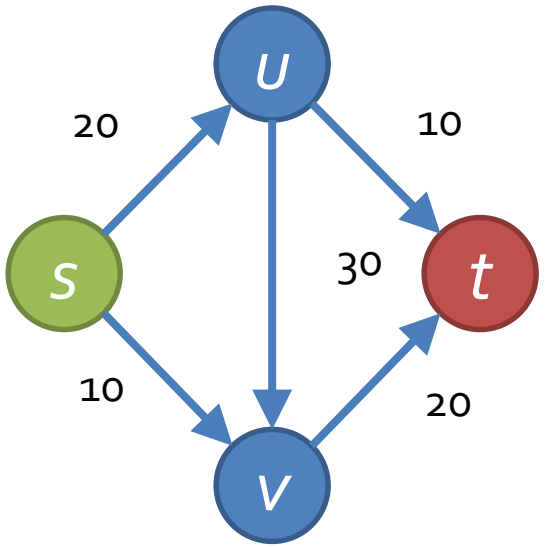
1. Start with zero flow on all edges
2. Find an augmenting path (increasing flow on forward edges and decreasing flow on backwards edges)
3. If you can still find an augmenting path in the residual graph, go back to Step 2

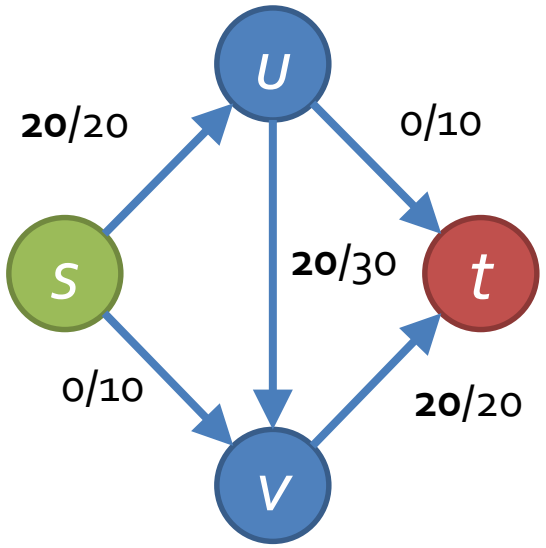
Find a max flow



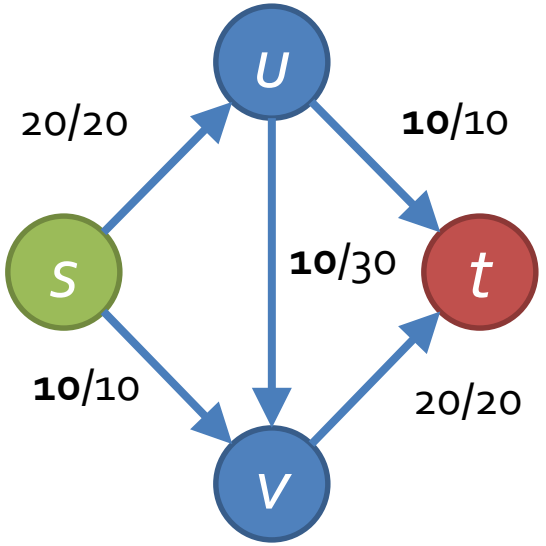
Why do we need the rule for backwards edges?

- If you don't have the rule for backwards edges, you can sometimes get stuck with more flow left unused
- Consider the following example:





If we add an augmenting path ***suvt*** with 20 units, we'll be stuck, unable to add more to ***v***



By adding an augmenting path ***svut*** with 10 units, we pull flow off ***uv***, splitting it back to ***ut***

Why does it work?

- An augmenting path is a flow
  - It never increases a forward edge beyond its maximum
  - It never decreases a backward edge below 0
  - The input to each node (other than ***s*** and ***t***) continues to be equal to the output
- Each augmenting path adds additional flow because more is coming out of ***s*** (and going into ***t***)
- When you can't find any more augmenting paths, there's no way to add more flow from ***s*** to ***t***
- Since all weights are integers, we will eventually reach the maximum flow, even if we're only increasing by 1 each time

Running time of Ford-Fulkerson

- Our definition of Ford-Fulkerson didn't say how you pick the augmenting path
- If the capacities are all integers, each flow value is an integer
- An augmenting path will increase the total flow by at least 1 each time
- Thus, the algorithm could take $O(|E|f)$, where $|E|$ is the number of edges in the graph and f is the maximum flow
  - That could be terrible if f has a large numerical value
- Edmonds-Karp is a variation of Ford-Fulkerson that uses a breadth-first search to find a shortest augmenting path
  - It runs in $O(|V||E|^2)$

Minimum Cuts

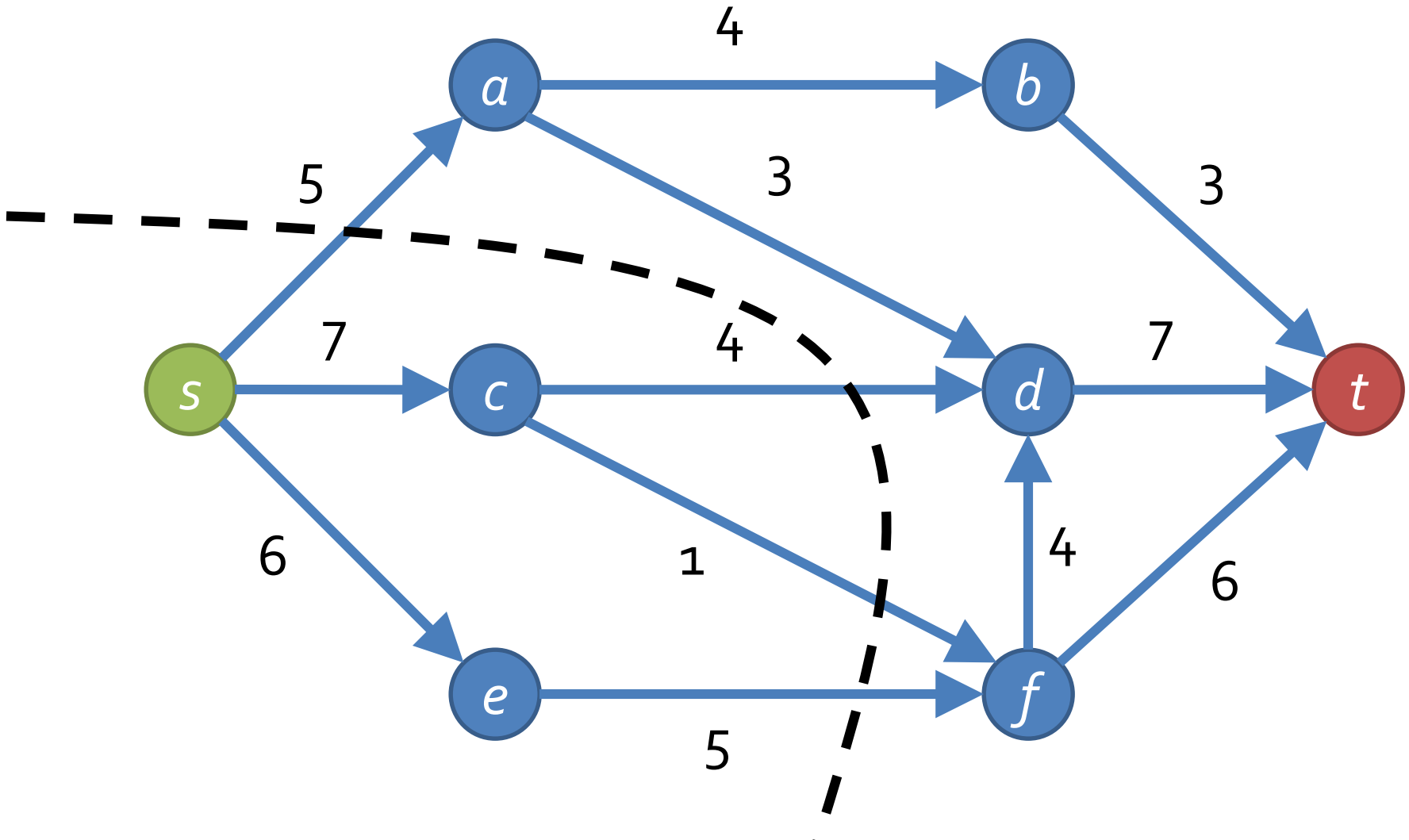
Cuts

- An **cut** in a graph partitions the graph into two disjoint sets
- An ***st*-cut** is a cut such that ***s*** is in one partition and ***t*** is in the other
- Think of it as a line that slices through the edges, putting ***s*** on one side and ***t*** on the other
- The **capacity of a cut** is the sum of the capacities of the edges that the cut divides

Maxflow-mincut theorem

- The smallest capacity ***st***-cut you can make has the same capacity as the largest possible ***st***-flow
- Intuitively, it's like that cut is a set of edges that most constricts the flow from ***s*** to ***t***

Minimum *st*-cut



Upcoming

Next time…

- Bipartite matching

Reminders

- Work on Homework 5
  - Due Friday by midnight
- Read section 7.5