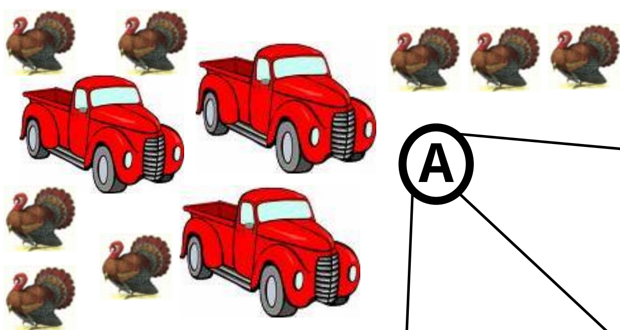
CSE 326: Data Structures
Network Flow

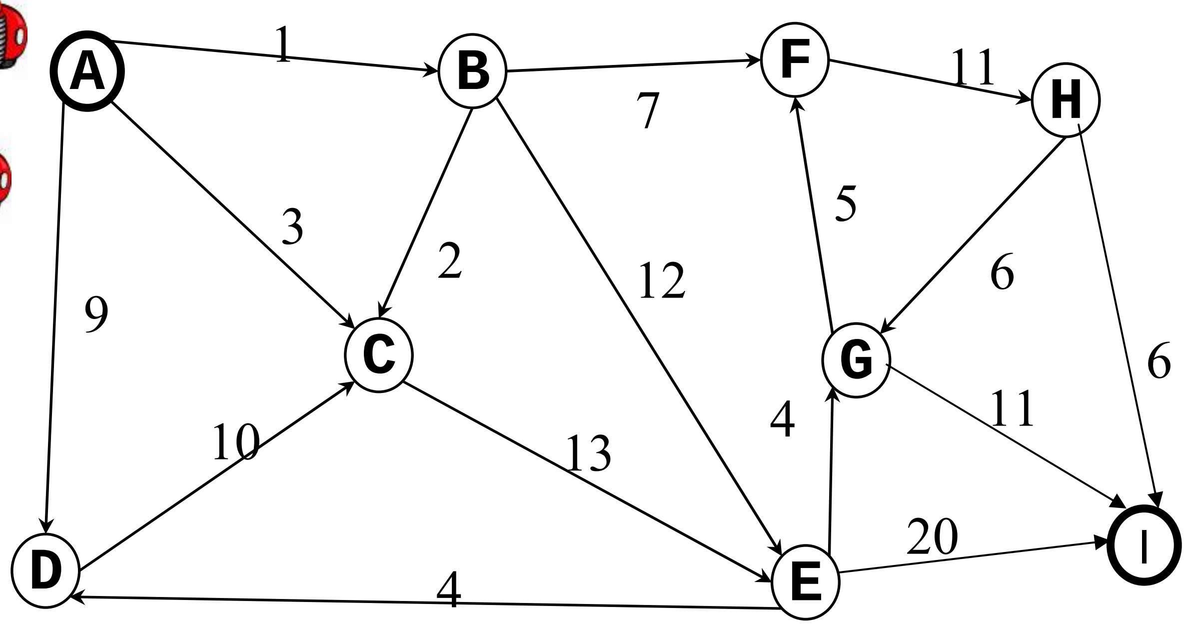
James Fogarty

Autumn 2007

Network Flows

- Given a weighted, directed graph G=(V,E)
- Treat the edge weights as *capacities*
- How much can we flow through the graph?





Network flow: definitions

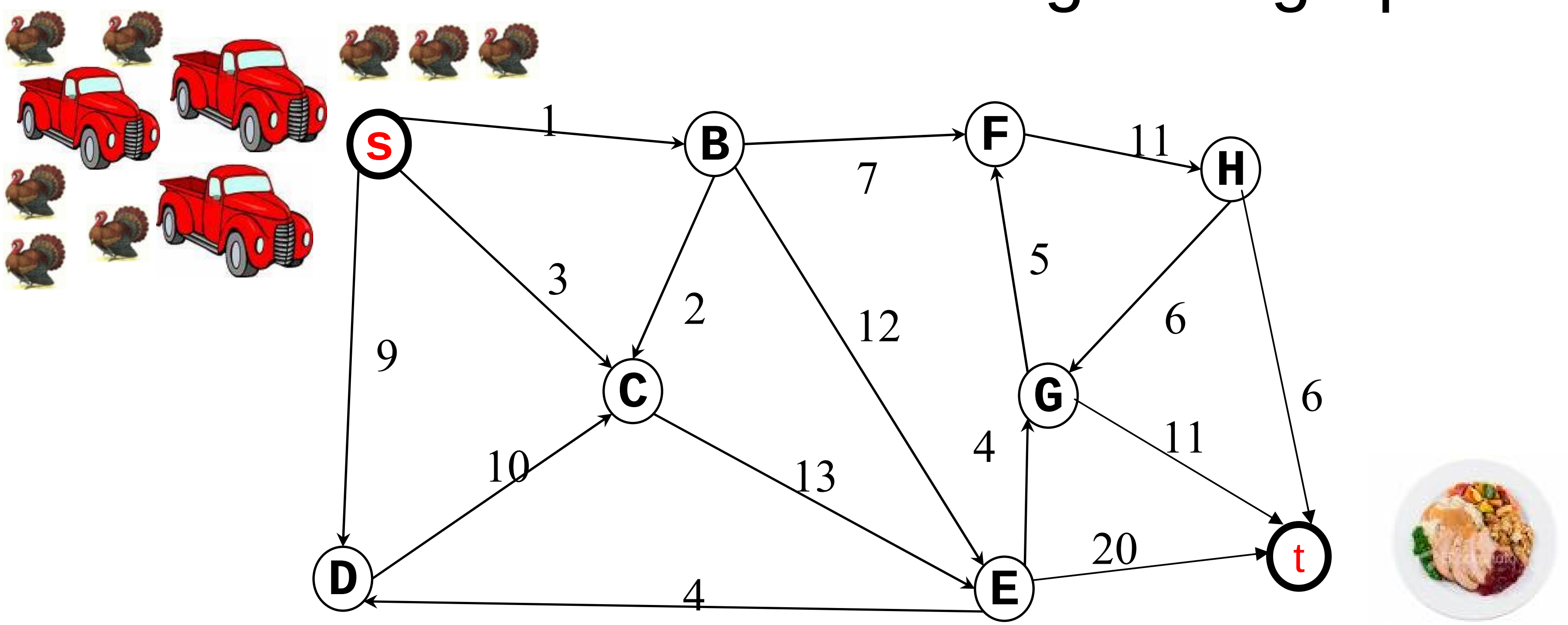
- Define special *source* s and *sink* t vertices
- Define a *flow* as a function on edges:
  - **Capacity**: *f(v,w) <= c(v,w)*
  - **Conservation**: $\sum_{v \in V} f(u,v) = 0$ for all u except source, sink
  - **Value of a flow**: $|f| = \sum_{v} f(s,v)$
  - Saturated edge: when f(v,w) = c(v,w)

Network flow: definitions

- **Capacity:** you can’t overload an edge
- **Conservation:** Flow entering any vertex must equal flow leaving that vertex
- We want to maximize the value of a flow, subject to the above constraints

Network Flows

- Given a weighted, directed graph G=(V,E)
- Treat the edge weights as *capacities*
- How much can we flow through the graph?



A Good Idea that Doesn't Work

- Start flow at 0
- "While there's room for more flow, push more flow across the network!"
  - While there's some path from s to t, none of whose edges are saturated
  - Push more flow along the path until some edge is saturated

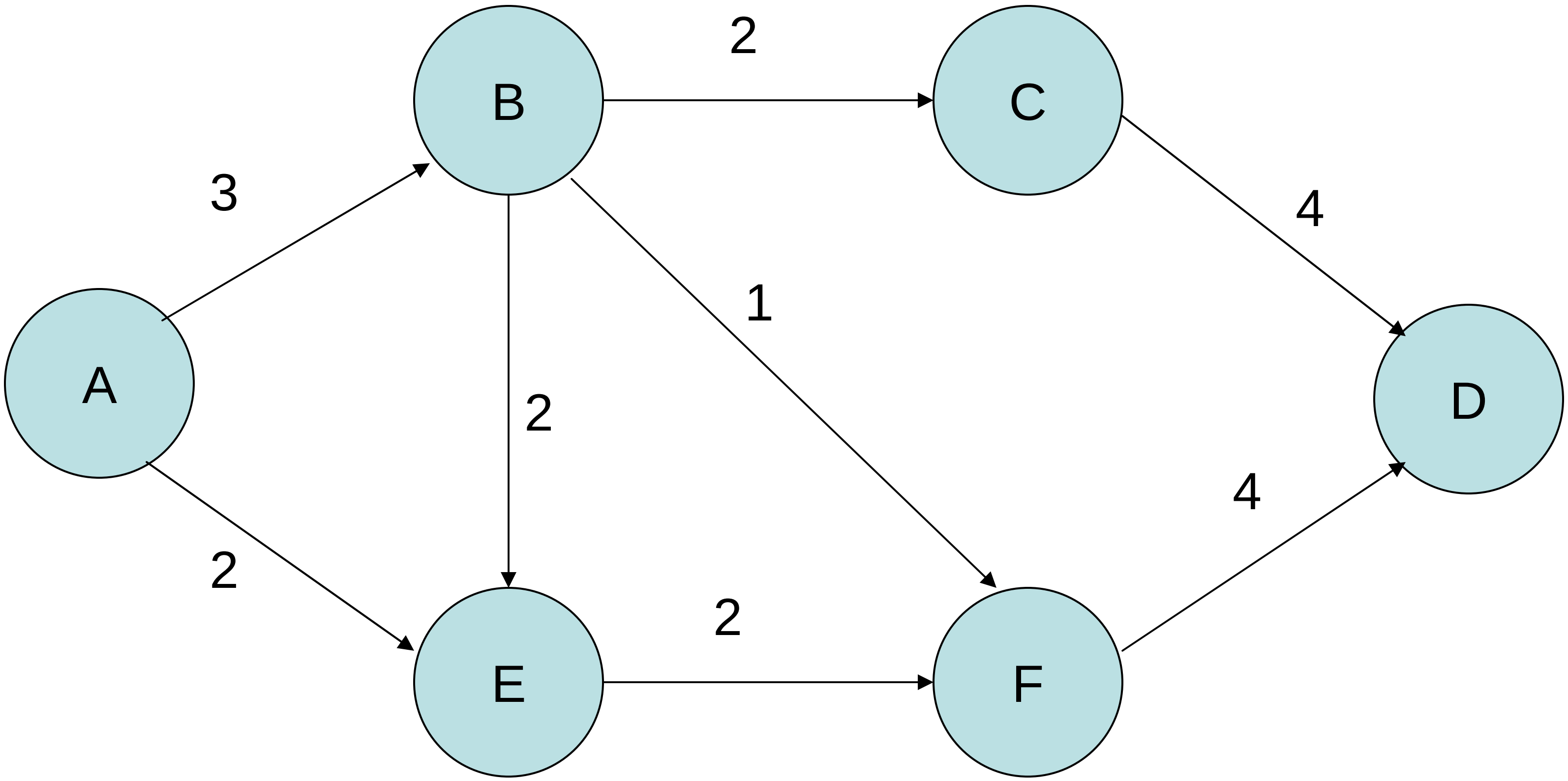
  - Called an "augmenting path"

How do we know there's still room?

- Construct a residual graph:
  - Same vertices
  - Edge weights are the “leftover” capacity on the edges
  - If there is a path s→t at all, then there is still room

Example (1)

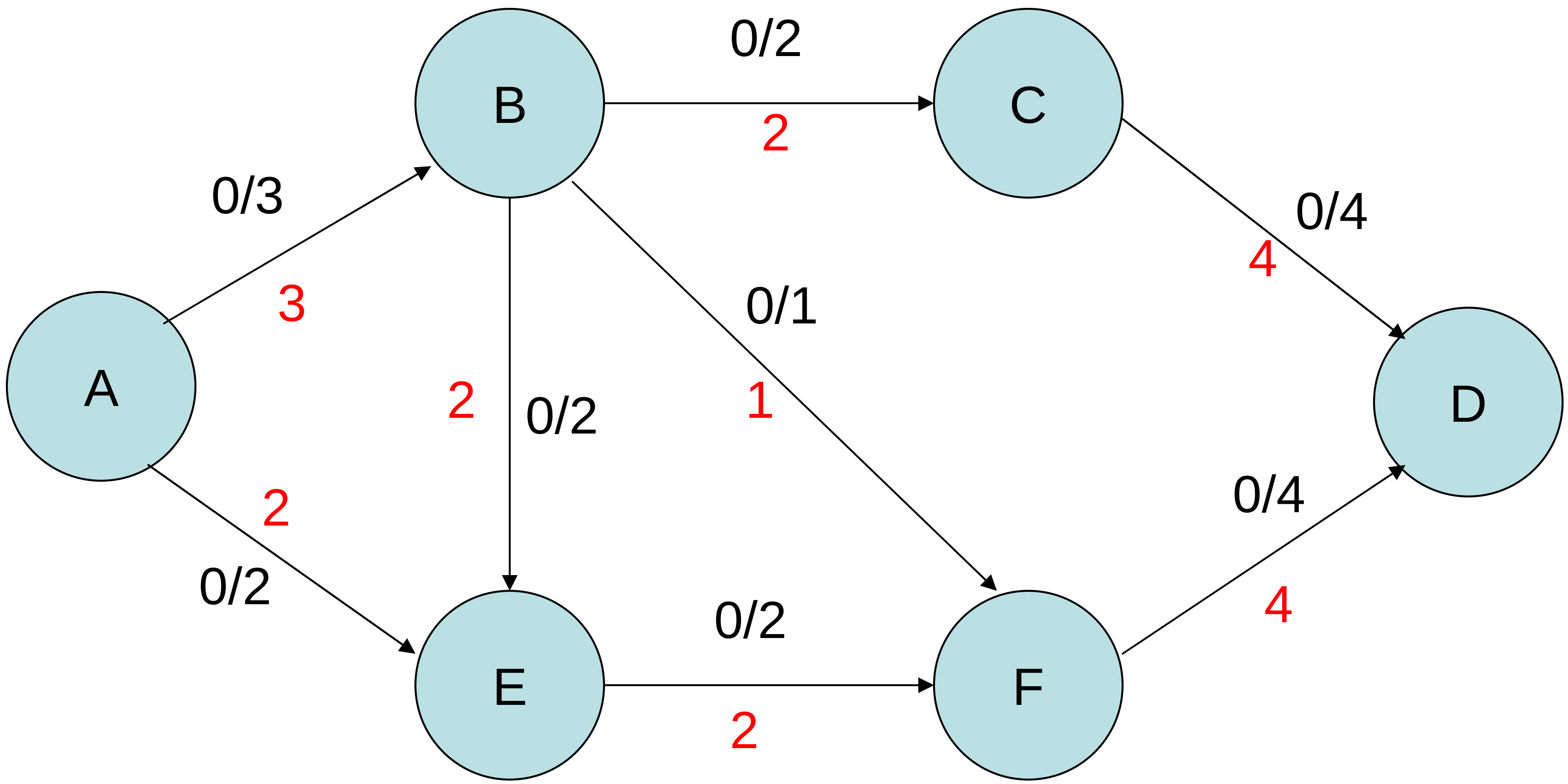
Initial graph – no flow



Flow / Capacity

Example (2)

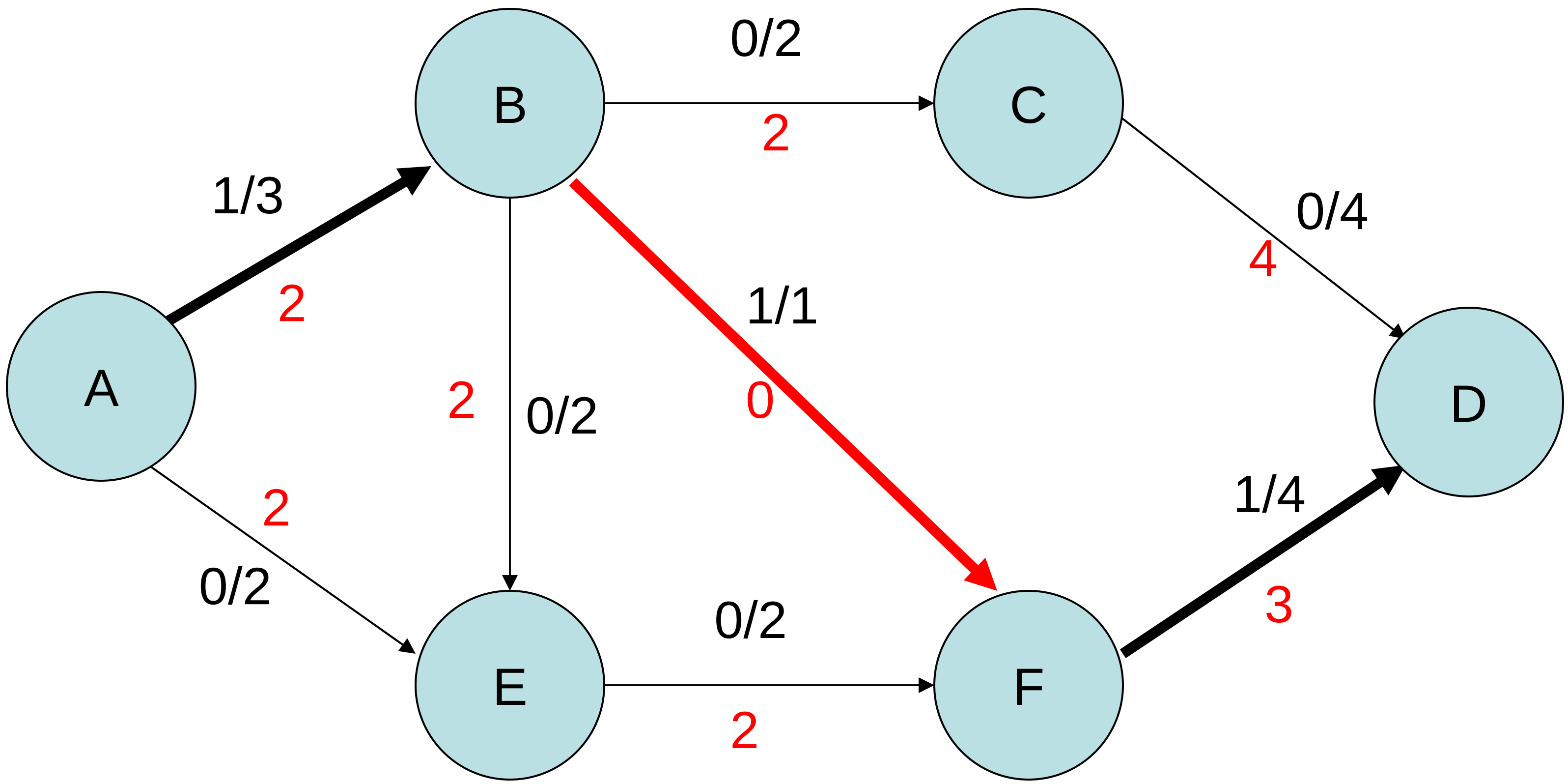
Include the residual capacities



Flow / Capacity
Residual Capacity

Example (3)

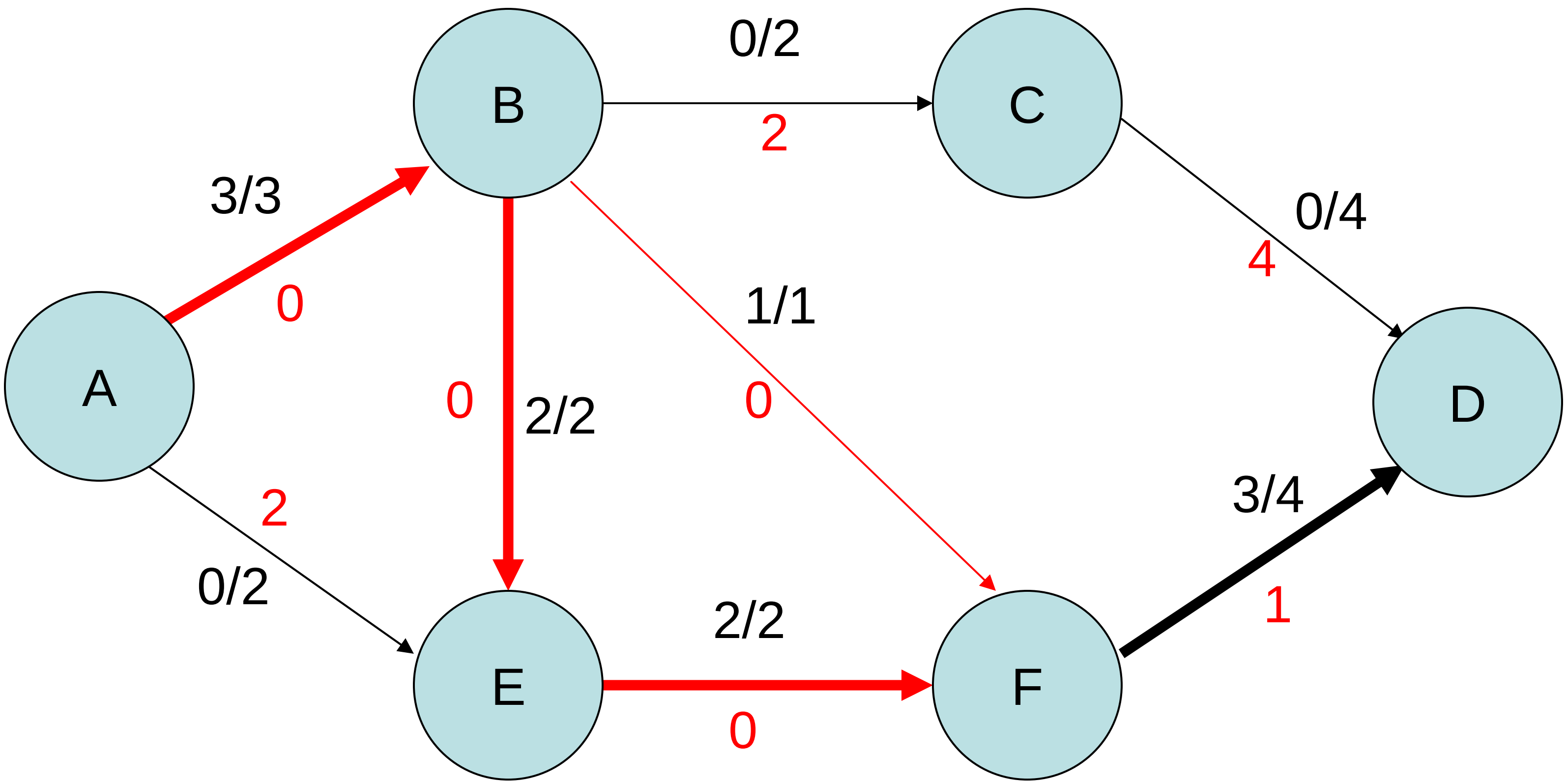
Augment along ABFD by 1 unit (which saturates BF)



Flow / Capacity
Residual Capacity

Example (4)

Augment along ABEFD (which saturates BE and EF)



Flow / Capacity
Residual Capacity

Now what?

- There's more capacity in the network…
- …but there's no more augmenting paths

Network flow: definitions

- Define special *source* s and *sink* t vertices
- Define a *flow* as a function on edges:
  - **Capacity**: *f(v,w) <= c(v,w)*
  - **Skew symmetry**: *f(v,w) = -f(w,v)*
  - **Conservation**: $\sum_{v \in V} f(u,v) = 0$ for all u except source, sink
  - **Value of a flow**: $|f| = \sum_{v} f(s,v)$
  - Saturated edge: when f(v,w) = c(v,w)

Network flow: definitions

- **Capacity:** you can't overload an edge
- **Skew symmetry:** sending *f* from *u*→*v* implies you're "sending *-f*", or you could "return f" from *v*→*u*
- **Conservation:** Flow entering any vertex must equal flow leaving that vertex
- We want to maximize the value of a flow, subject to the above constraints

Main idea: Ford-Fulkerson method

- Start flow at 0
- “While there’s room for more flow, push more flow across the network!”
  - While there’s some path from s to t, none of whose edges are saturated
  - Push more flow along the path until some edge is saturated
  - Called an “augmenting path”

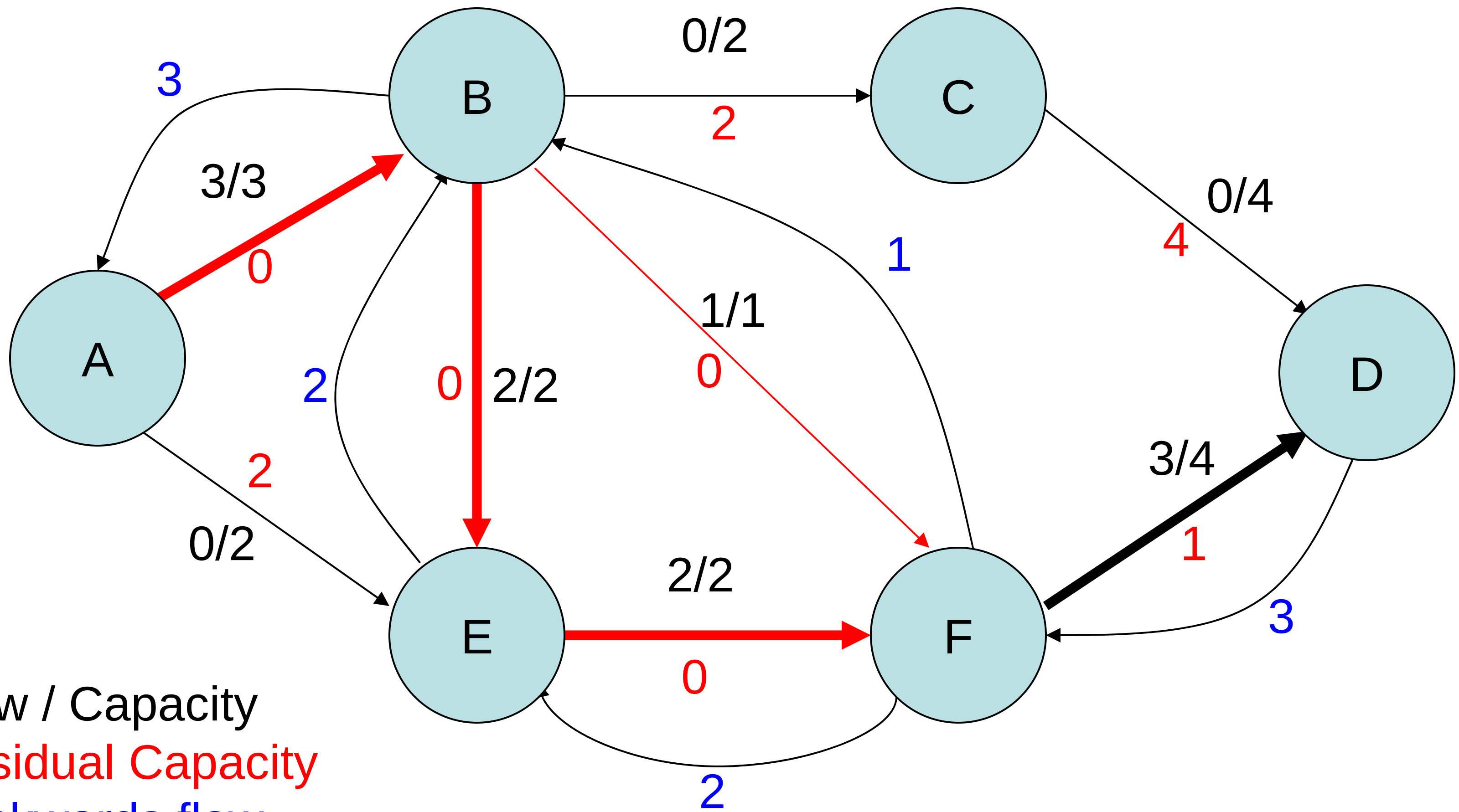
How do we know there's still room?

- Construct a residual graph:
  - Same vertices
  - Edge weights are the "leftover" capacity on the edges
  - Add extra edges for backwards-capacity too!

  - If there is a path s→t at all, then there is still room

Example (5)

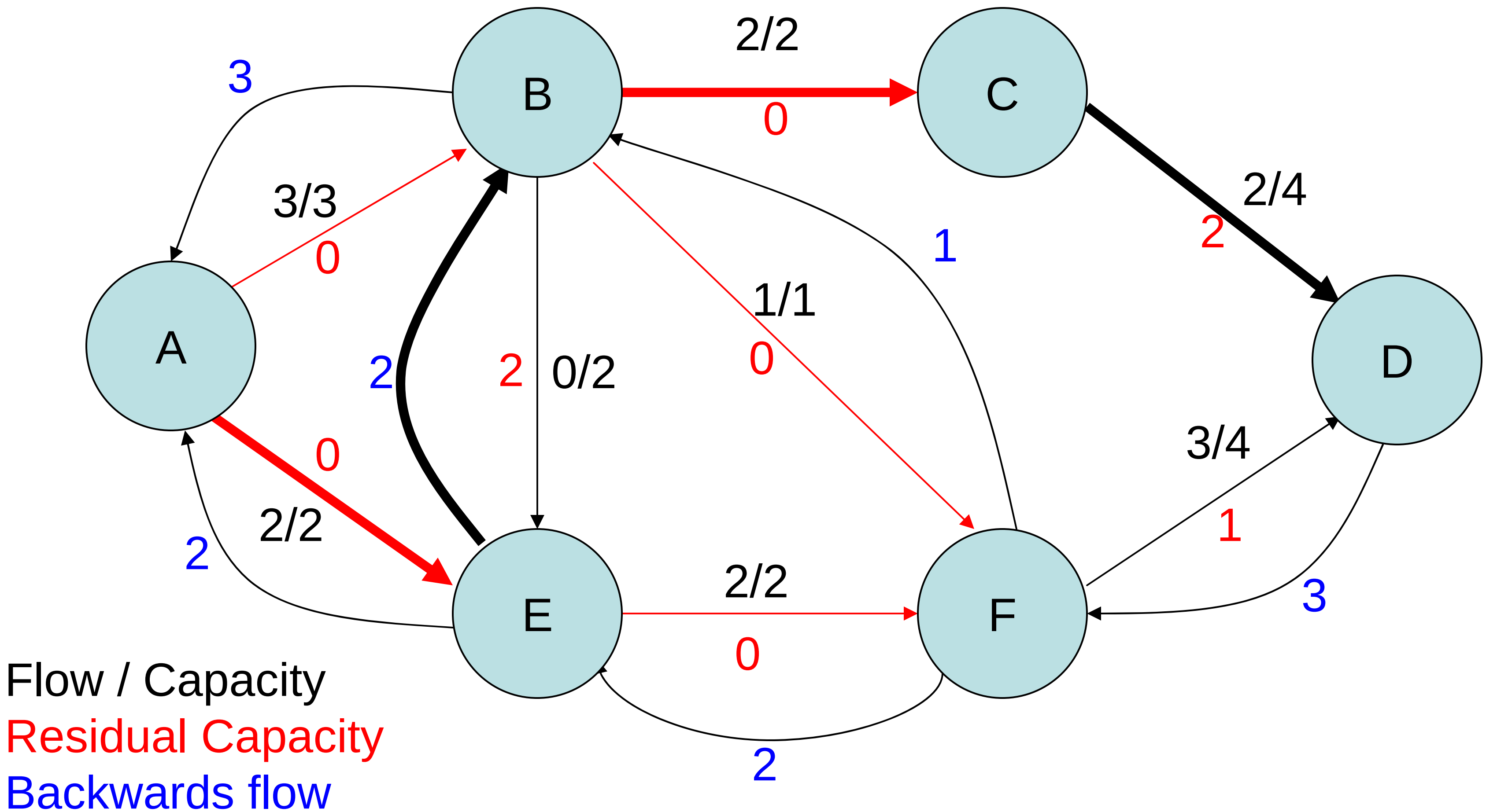
Add the backwards edges, to show we can "undo" some flow



Flow / Capacity
Residual Capacity
Backwards flow

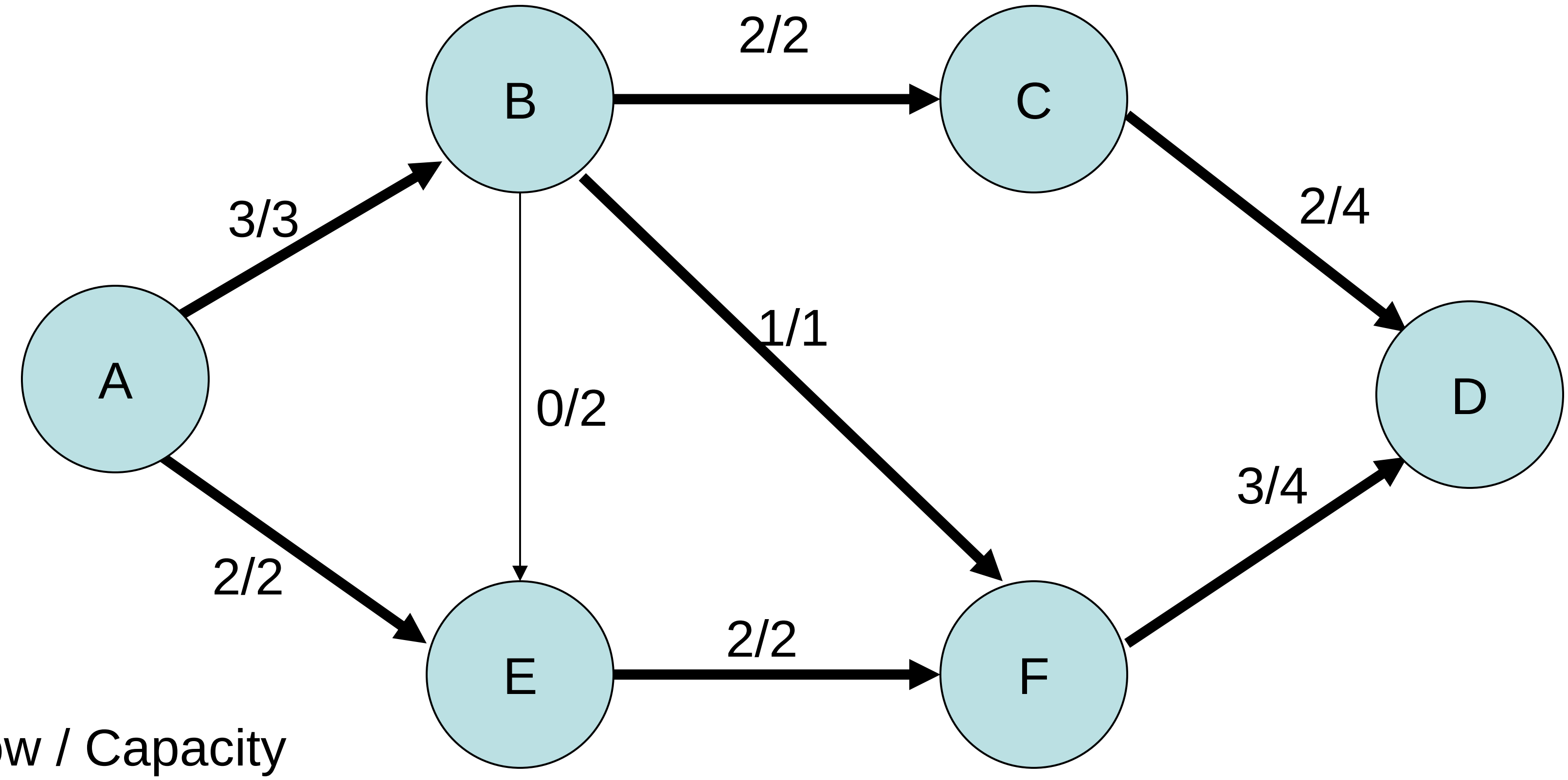
Example (6)

Augment along AEBCD (which saturates AE and EB, and empties BE)



Example (7)

Final, maximum flow

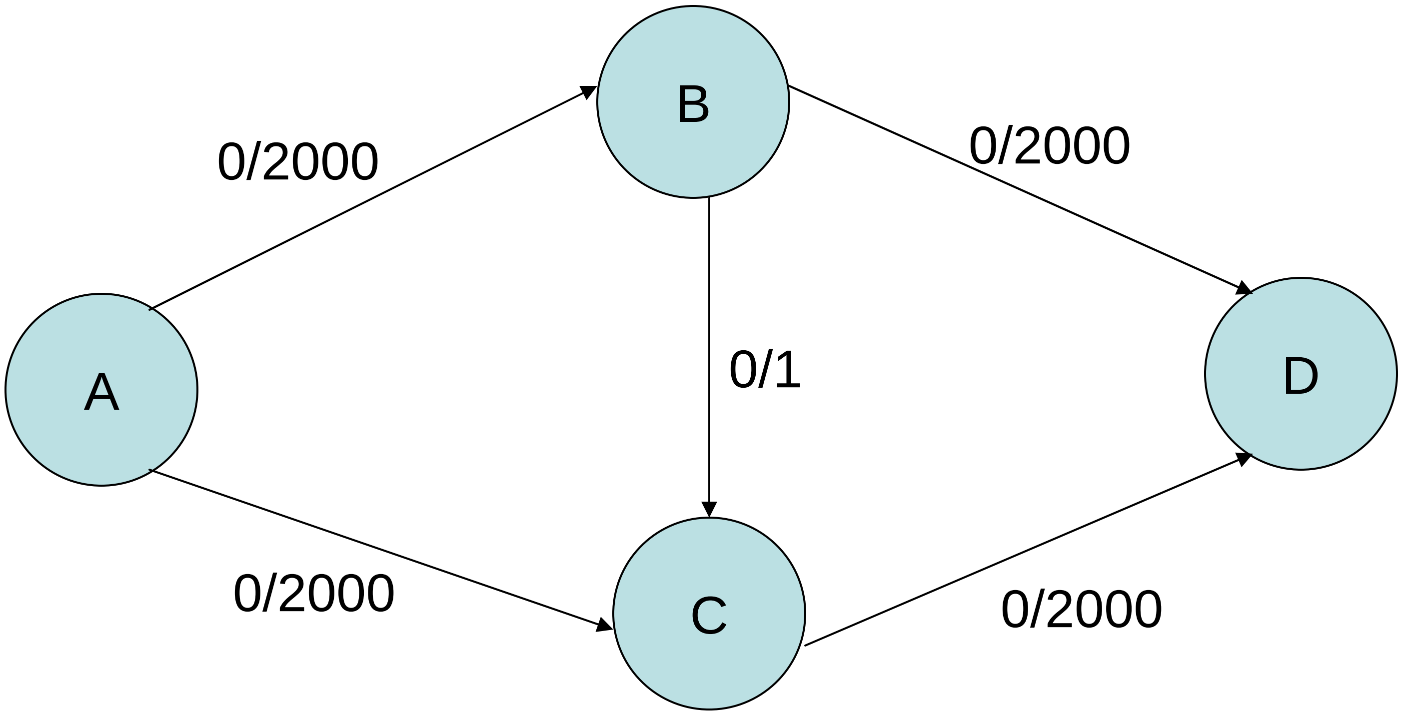


Flow / Capacity
Residual Capacity
Backwards flow

How should we pick paths?

- Two very good heuristics (Edmonds-Karp):
  - Pick the largest-capacity path available
    - Otherwise, you'll just come back to it later…so may as well pick it up now
  - Pick the shortest augmenting path available
    - For a good example why…

Don't Mess this One Up



Augment along ABCD, then ACBD, then ABCD, then ACBD…

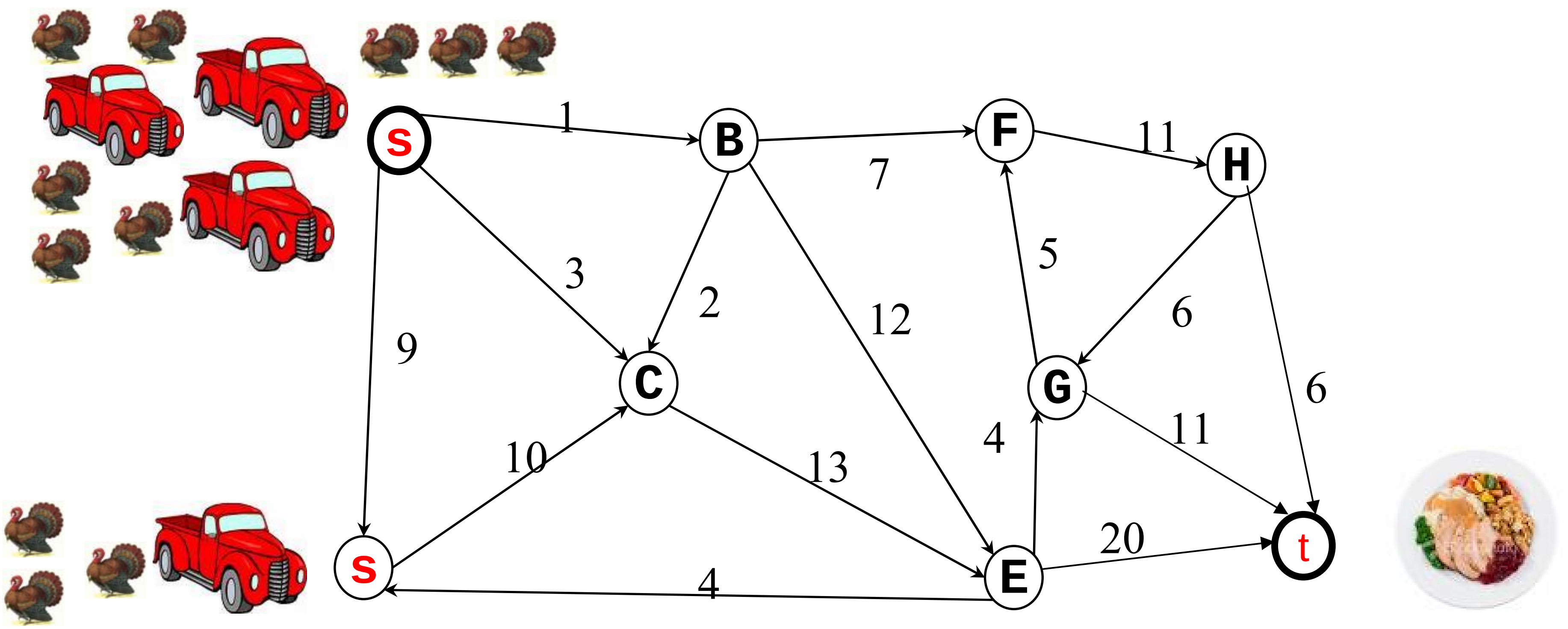
Should just augment along ACD, and ABD, and be finished

Running time?

- Each augmenting path can't get shorter…and it can't always stay the same length
  - So we have at most O(E) augmenting paths to compute for each possible length, and there are only O(V) possible lengths.
  - Each path takes O(E) time to compute
- Total time = $O(E^2V)$

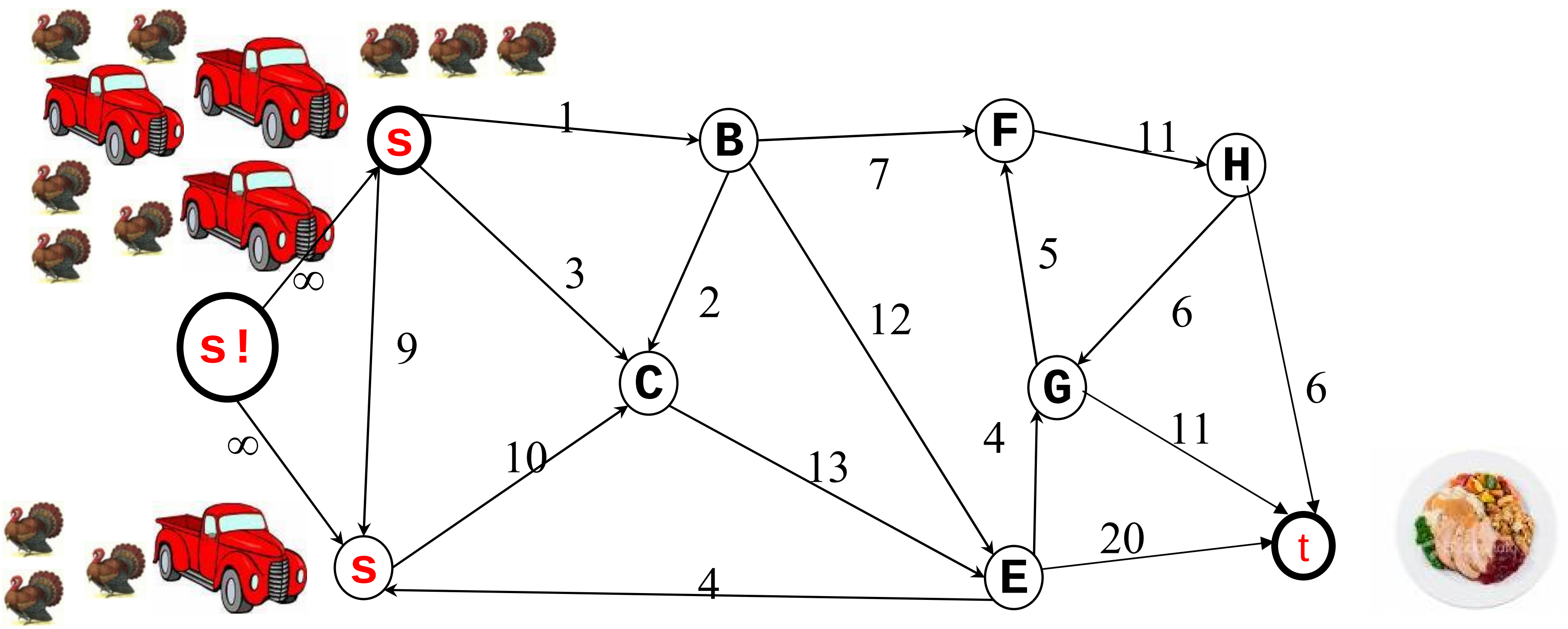
Network Flows

- What about multiple turkey farms?



Network Flows

- Create a single source, with infinite capacity edges connected to sources
- Same idea for multiple sinks



One more definition on flows

- We can talk about the flow from a *set of vertices* to another set, instead of just from one vertex to another:

$$f(X,Y) = \sum_{x \in X} \sum_{y \in Y} f(x,y)$$

  - Should be clear that f(X,X) = 0
  - So the only thing that counts is flow between the two sets

Network cuts

- Intuitively, a cut separates a graph into two disconnected pieces

- Formally, a cut is a pair of sets (S, T), such that

$$V = S \cup T$$

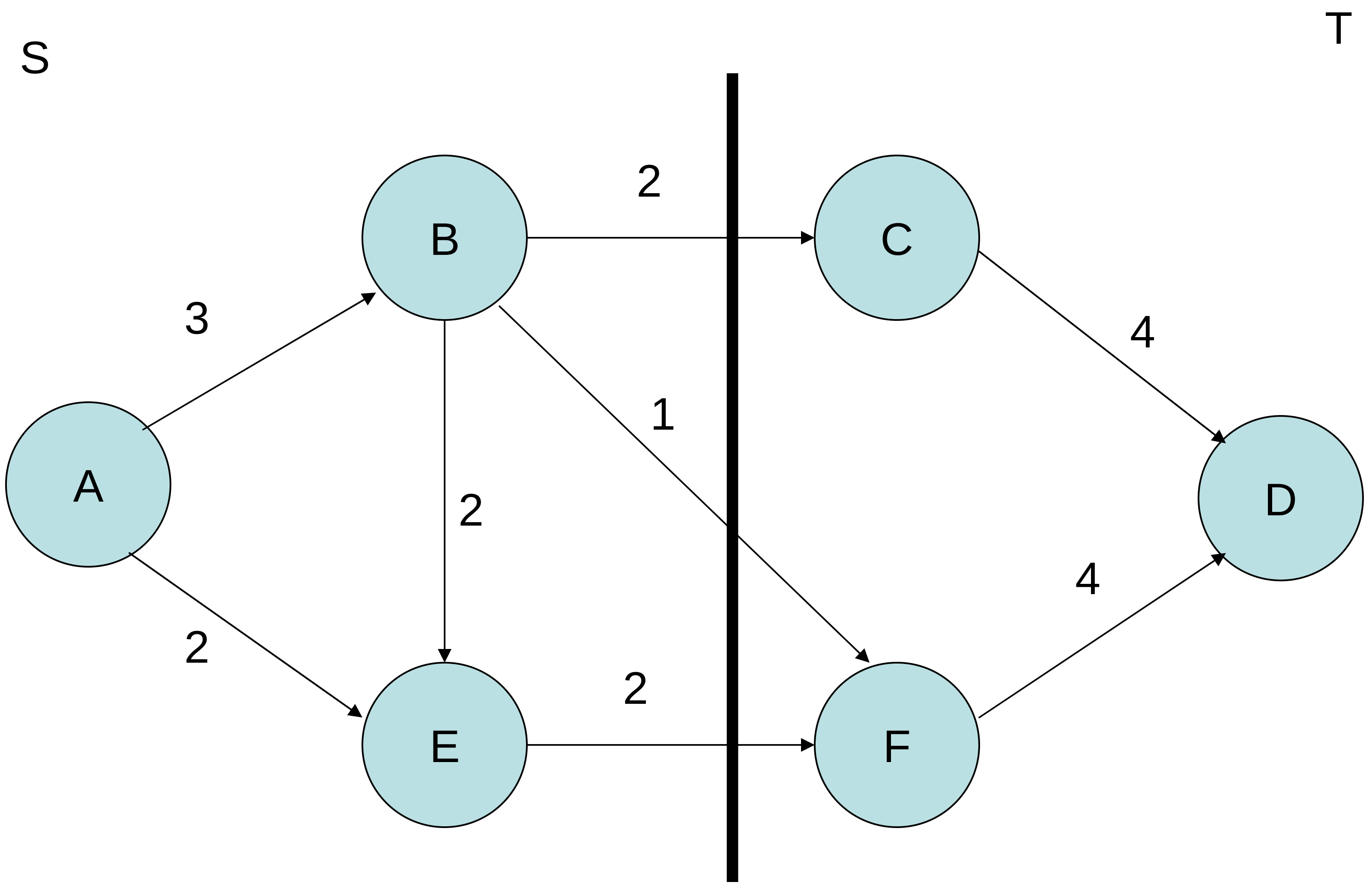
$$S \cap T = \{\}$$

and S and T are connected subgraphs of G

Minimum cuts

- If we cut G into (S, T), where S contains the source s and T contains the sink t,
- Of all the cuts (S, T) we could find, what is the smallest (max) flow f(S, T) we will find?

Min Cut - Example (8)

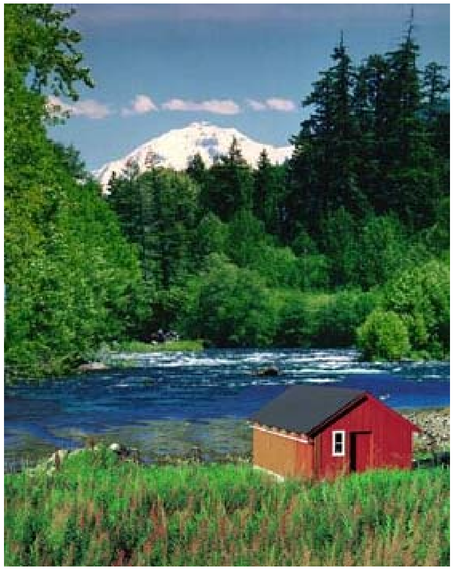


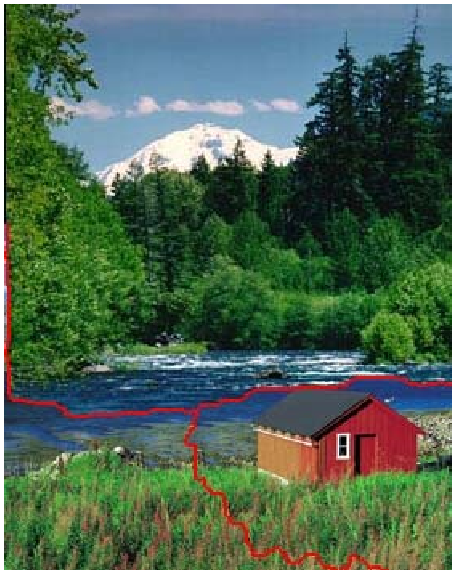
Capacity of cut = 5

Coincidence?

- NO! Max-flow always equals Min-cut
- Why?
  - If there is a cut with capacity equal to the flow, then we have a maxflow:
    - We can't have a flow that's bigger than the capacity cutting the graph! So any cut puts a bound on the maxflow, and if we have an equality, then we must have a maximum flow.
  - If we have a maxflow, then there are no augmenting paths left
    - Or else we could augment the flow along that path, which would yield a higher total flow.
  - If there are no augmenting paths, we have a cut of capacity equal to the maxflow
    - Pick a cut (S,T) where S contains all vertices reachable in the residual graph from s, and T is everything else. Then every edge from S to T must be saturated (or else there would be a path in the residual graph). So $c(S,T) = f(S,T) = f(s,t) = |f|$ and we're done.

GraphCut





http://www.cc.gatech.edu/cpl/projects/graphcuttextures/