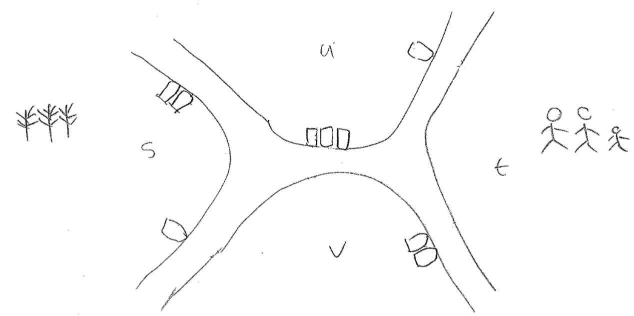
COMPSCI 311: Introduction to Algorithms

Lecture 18: Network Flow

Dan Sheldon

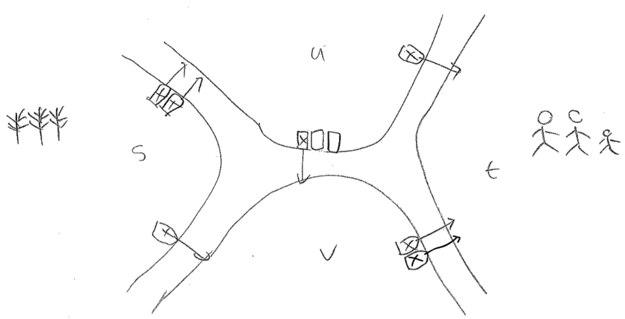
University of Massachusetts Amherst

A Puzzle

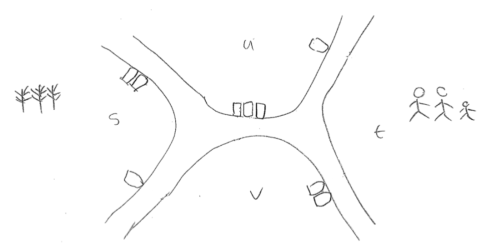


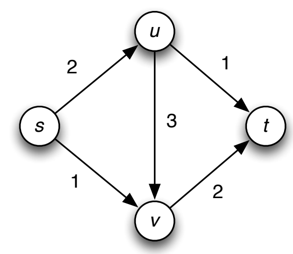
How many loads of grain can you ship from s to t? Which boats are used?

A Puzzle

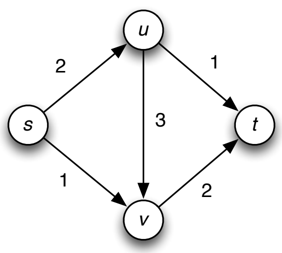


Flow Network





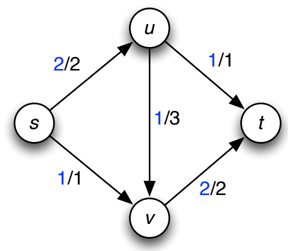
Max-Flow Problem



Problem input is a **flow network**

- Directed graph
- Source node s
- Target node or *sink* t
- Edge capacities $c(e) \geq 0$

Solution: A Flow



A **network flow** is an assignment of values $f(e)$ to each edge e, which satisfy:

- Capacity constraints: $0 \leq f(e) \leq c(e)$ for all e
- Flow conservation:

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$

for all $v \notin \{s, t\}$.

Value $v(f)$ of flow f = total flow on edges leaving source

Max flow problem: find a flow of maximum value

Algorithm Design Techniques

- Greedy
- Divide and Conquer
- Dynamic Programming
- Network Flows

Network Flow

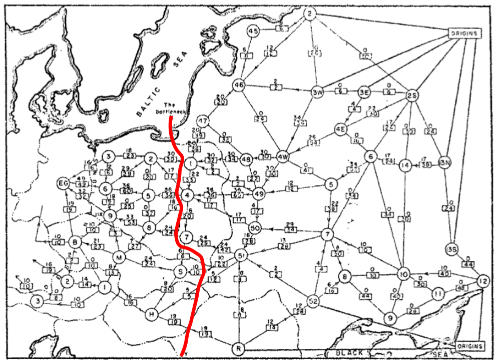
- Previous topics were design techniques (Greedy, Divide-and-Conquer, Dynamic Programming)
- Network flow: a specific class of problems with many applications

- Direct applications: commodities in networks
  - transporting goods on the rail network
  - packets on the internet
  - gas through pipes
- Indirect applications:
  - Matching in graphs
  - Airline scheduling
  - Baseball elimination

Plan: design and analyze algorithms for max-flow problem, then apply to solve other problems

First, a Story About Flow and Cuts

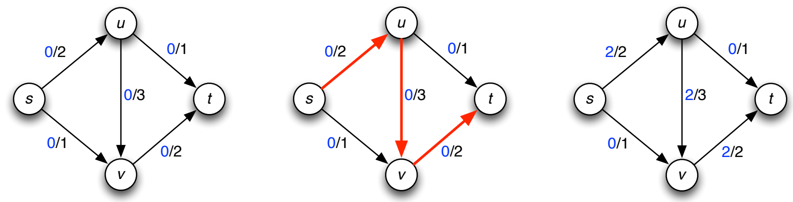
Key theme: flows in a network are intimately related to cuts   Soviet rail network (Harris & Ross, RAND report, 1955)



On the history of the transportation and maximum flow problems. Alexander Schrijver, Math Programming, 2002.

Designing a Max-Flow Algorithm

First idea: initialize to zero flow and then repeatedly "augment" flow on paths from s to t until we can no longer do so.

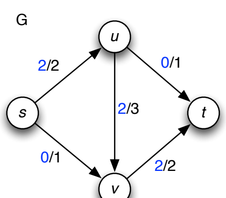


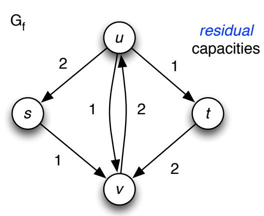
Problem: we are now stuck. All $s \to t$ paths have a *saturated* edge.

We would like to "augment" $s \xrightarrow{+1} v \xleftarrow{-1} u \xrightarrow{+1} t$, but this is not a real $s \to t$ path. How can we identify such an opportunity?

Residual Graph (Key Idea!!)

The residual graph G_f identifies ways to increase flow on edges with leftover capacity, or decrease flow on edges already carrying flow:





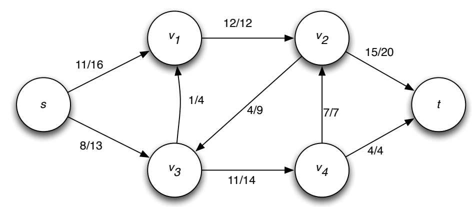
For each original edge $e = (u, v)$ in G, it has:

- A forward edge $e = (u, v)$ with *residual capacity* $c(e) - f(e)$
- A reverse edge $e' = (v, u)$ with *residual capacity* $f(e)$

Edges with zero residual capacity are omitted

Exercise: residual graph

G



Let G and f be as depicted above. What is the residual capacity of edge (v_1, v_3) in G_f?

A. 3

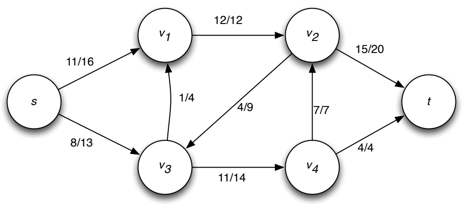
B. 1

C. 4

D. The edge is not present in G_f.

Exercise: residual graph

G



Let G and f be as depicted above. What is the residual capacity of edge (v_2, v_3) in G_f?

A. 5

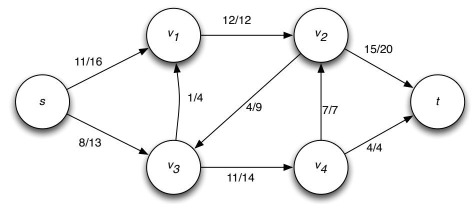
B. 4

C. 9

D. The edge is not present in G_f.

Exercise: residual graph

G



Let G and f be as depicted above. What is the residual capacity of edge (v_4, v_2) in G_f?

A. 0

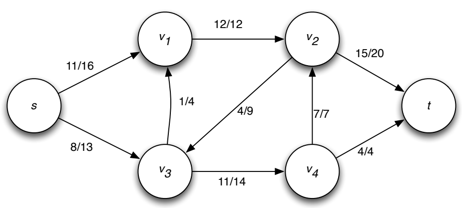
B. 7

C. 4

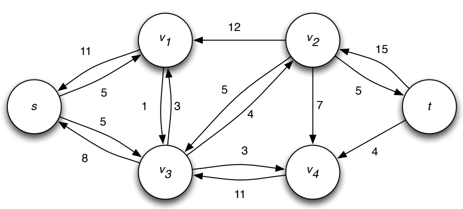
D. The edge is not present in G_f.

Exercise: residual graph

G



G_f

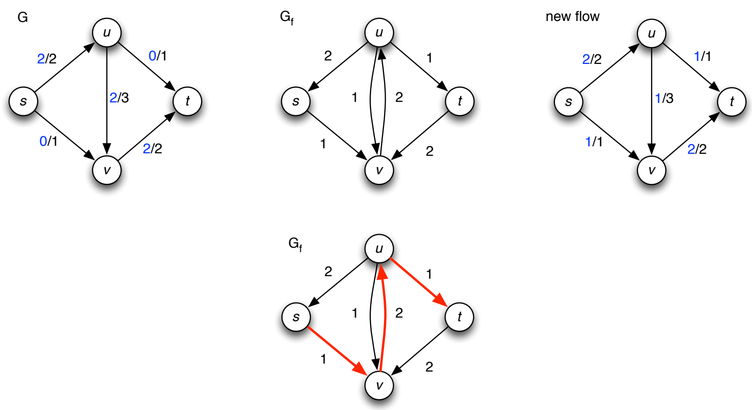


Emphasis: Residual Graph

- The residual graph is **the key data structure used for network flows**
- If you have a graph G and flow f, **construct the residual graph G_f**

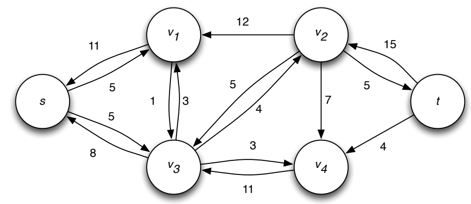
Augment Operation

Revised Idea: use s-t paths in the *residual* graph ("augmenting paths") to augment flow



Clicker Question

What is the largest bottleneck capacity of any augmenting path?



A. 1
B. 4
C. 5
D. 11

Augment Operation

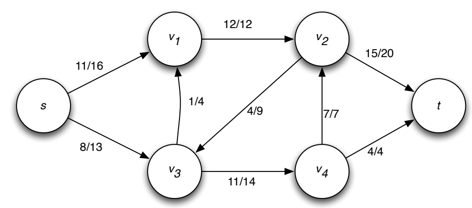
Revised Idea: use paths in the *residual* graph to augment flow

f = flow in G
P = augmenting path = $s \to t$ path in G_f

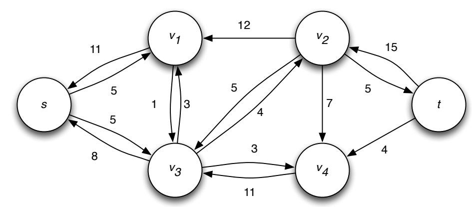
Augment(f, P)
  Let b = bottleneck(P, f) ▷ least residual capacity in P
  for each edge e in P **do**
    if e is a forward edge **then**
      $f(e) = f(e) + b$ ▷ increase flow on forward edges
    else e is a backward edge
      Let e' be opposite edge in G
      $f(e') = f(e') - b$ ▷ decrease flow on backward edges

Augment Example

G

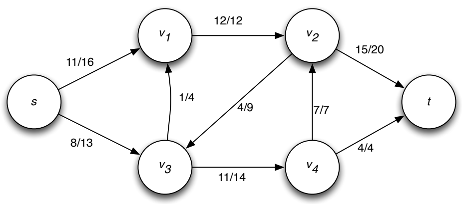


G_f

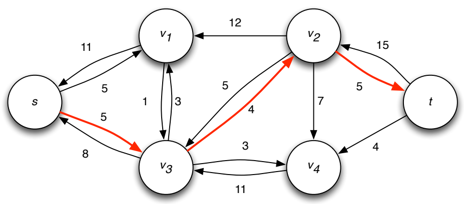


Augmenting Path

G

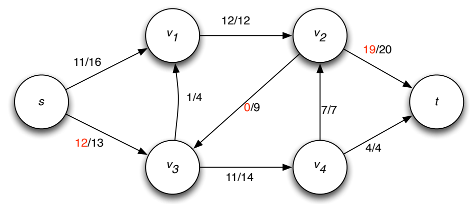


G_f

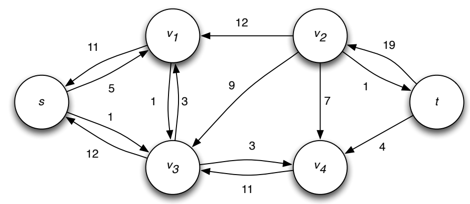


New Flow

G



G_f



Ford-Fulkerson Algorithm

Repeatedly find augmenting paths in the residual graph and use them to augment flow!

Ford-Fulkerson(G, s, t)

- ▷ Initially, no flow
- Initialize $f(e) = 0$ for all edges e
- Initialize $G_f = G$
- ▷ Augment flow as long as it is possible
- **while** there exists an s-t path P in G_f **do**
  - f = Augment(f, P)
  - update G_f
- return f

Clicker

Given a graph G and a flow f, how can you test if f is a maximum flow?

A. Check for an $s \to t$ path in the residual graph G_f.

B. Check for an $s \to t$ path in the residual graph G_f.

C. Check for an $s \to t$ path in the residual graph G_f.

D. Check for an $s \to t$ path in the residual graph G_f.

Ford-Fulkerson Example

See Pearson slides

Ford-Fulkerson Analysis

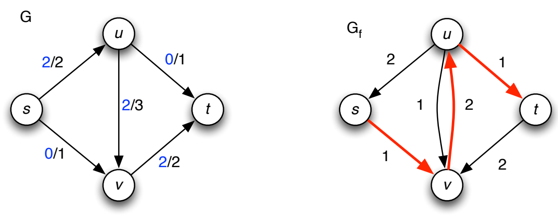
- Step 1: argue that F-F returns a flow
- Step 2: analyze termination and running time
- Step 3: argue that F-F returns a maximum flow

Step 1: F-F returns a flow

Claim: If f is a flow then $f' = \text{Augment}(f, P)$ is also a flow.

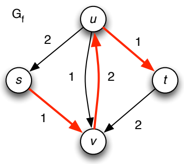
Proof idea. Verify two conditions for f' to be a flow: capacity and flow conservation.

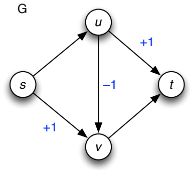
Capacity



- Suppose original edge is $e = (x, y)$
- If forward edge (x, y) appears in P, then flow on e increases by bottleneck capacity b, which is at most $c(e) - f(e)$, so does not exceed $c(e)$
- If reverse edge (y, x) appears in P, then flow decreases by bottleneck capacity b, which is at most $f(e)$, so is at least 0

Flow Conservation





Consider any node v in augmenting path, do case analysis on edge types:

residual graph: $P = s \leadsto u \longrightarrow v \longrightarrow w \leadsto t$

original graph:

$$u \xrightarrow{+b} v \xrightarrow{+b} w$$
$$u \xrightarrow{+b} v \xleftarrow{-b} w$$
$$u \xleftarrow{-b} v \xrightarrow{+b} w$$
$$u \xleftarrow{-b} v \xleftarrow{-b} w$$

In all cases, change in incoming flow at v is equal to the change in outgoing flow.

Step 2: Termination and Running Time

Assumption: All capacities are integers. By nature of F-F, all flow values and residual capacities remain integers during the algorithm.

Running time:

- In each F-F iteration, flow increases by at least 1. Therefore, number of iterations is at most $v(f^*)$, where f^* is the final flow.
- Let C be the total capacity of edges leaving source s.
- Then $v(f^*) \leq C$.
- So F-F terminates in at most C iterations

Running time per iteration? $O(m + n)$ to find an augmenting path

Step 3: F-F returns a maximum flow

We will prove this by establishing a deep connection between flows and cuts in graphs: the max-flow min-cut theorem.

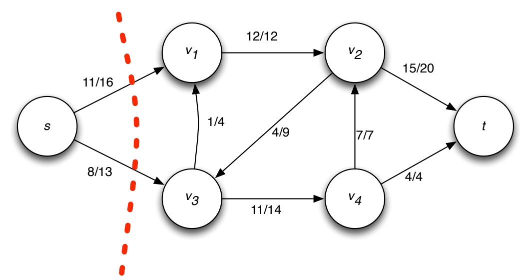
- An s-t cut (A, B) is a partition of the nodes into sets A and B where $s \in A$, $t \in B$
- Capacity of cut (A, B) equals

$$c(A, B) = \sum_{e \text{ from } A \text{ to } B} c(e)$$

- Flow across a cut (A, B) equals

$$f(A, B) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$

Example of Cut



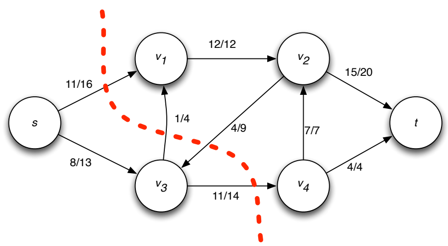
Exercise: write capacity of cut and flow across cut.

Capacity is 29 and flow across cut is 19.

Clicker Question

What is the capacity of the cut and the flow across the cut?

	Capacity	Flow
A.	16+4+9+14	11+1+3+11
B.	16+4 -9+14	11+1 -4+11
C.	16+4+14	11+1 -4+11
D.	16+4+14	11+1+11



Flow Value Lemma

First relationship between cuts and flows

Lemma: let f be any flow and (A, B) be any s-t cut. Then

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$

Proof: see book. Basic idea is to use conservation of flow: all the flow out of s must leave A eventually.