Flow networks, flow, maximum flow

Can interpret directed graph as “flow network”.

Material courses through some system from some source to some sink.

Source produces material at some steady rate, sink consumes at same rate.

Examples:

- water through system of pipes
- current through electrical network
- data through network
- trucks from factory to warehouse

Each edge has given capacity (pipe: x gallons of water per minute; street: y trucks per hour; network: z kB per second).

Vertices are “junctions”, for those other than source and sink, material flows through them without collecting in them (i.e., rate at which material enters vertex equals rate at which it leaves it; “flow conservation”).

In **maximum-flow problem**, we wish to compute greatest possible rate of transportation from source to sink.

Some definitions

Flow network $G = (V, E)$ is directed graph, each edge $(u, v) \in E$ has **non-negative capacity** $c(u, v) \geq 0$. If $(u, v) \notin E$, we assume $c(u, v) = 0$.

Two distinguished vertices: **source** s and **sink** t.

We assume that every vertex lies on some path from s to t, i.e., for each $v \in V$, there is a path $s \leadsto v \leadsto t$ (implies that G is connected, $|E| \geq |V| - 1$).

A **flow** in G is a real-valued function $f : V \times V \to \mathbb{R}$, that satisfies these three properties:

Capacity constraint:

For all $u, v \in V$, we require $f(u, v) \leq c(u, v)$.

Flow from one vertex to another must not exceed given capacity.

Skew symmetry:

For all $u, v \in V$, we require $f(u, v) = -f(v, u)$.

Flow from vertex u to vertex v is negative of flow in reverse direction.

Flow conservation:

For all $u \in V - \{s, t\}$, we require

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

Total flow *out of* a vertex (other than source or sink) is 0. Can rewrite as $\forall v \in V - \{s, t\} : \sum_{u \in V} f(u, v) = 0$, i.e., total flow *into* a vertex is 0.

$f(u,v)$ — can be positive, zero, or negative — is called **flow** from u to v.

The **value** of flow f is defined as the total flow leaving the source (and thus entering the sink):

$$|f| = \sum_{v \in V} f(s,v)$$

Note: $|\cdot|$ does not mean "absolute value" or "cardinality").

The **total positive flow entering** vertex v is

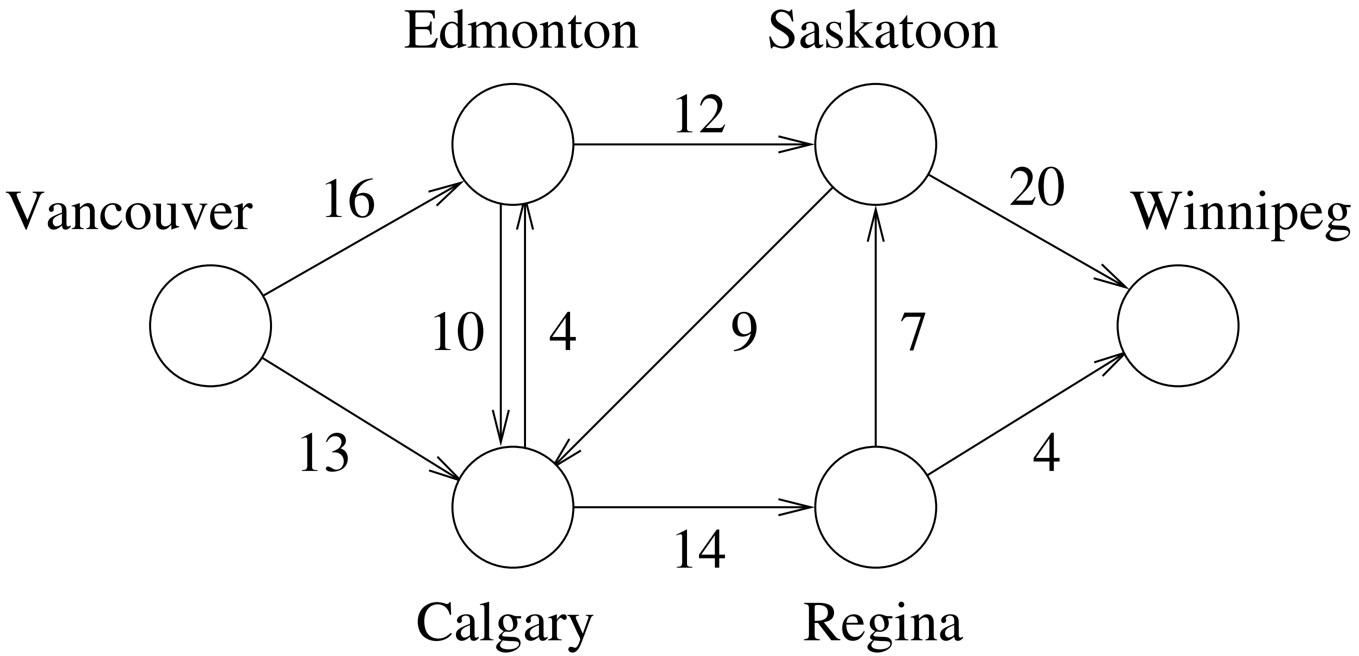
$$\sum_{u \in V:\ f(u,v)>0} f(u,v)$$

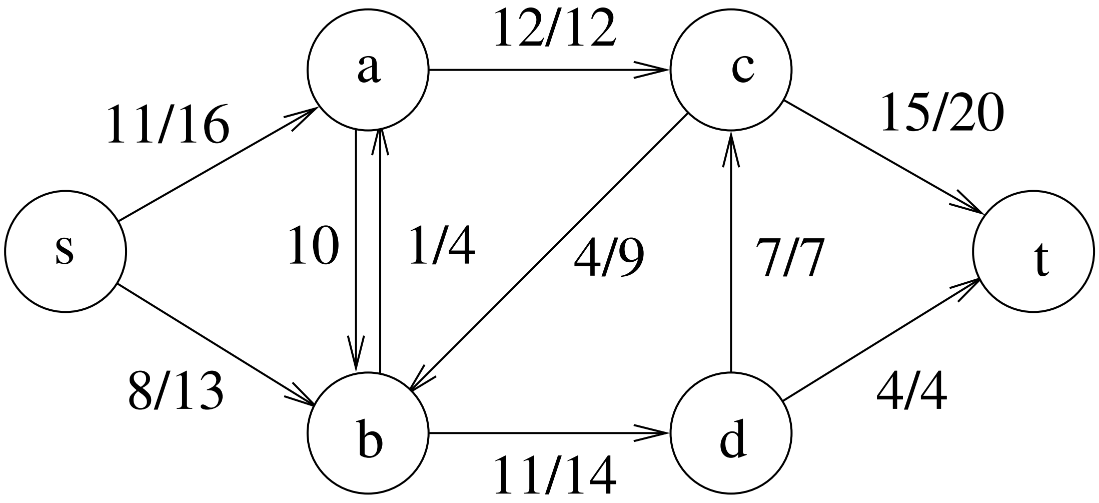
Also, **total positive flow leaving** vertex u is

$$\sum_{v \in V:\ f(u,v)>0} f(u,v)$$

The **total net flow** at a vertex v is

$$\begin{array}{rl} & \text{total positive flow leaving } v \\ - & \text{total positive flow entering } v \end{array}$$





Second figure shows flow f in G with value $|f| = 19$.

Cancellation

Suppose a company ships 8 crates per day from Edmonton to Calgary, and 3 crates per day from Calgary to Edmonton.

Natural enough, but we can't represent these shipments directly by flows: **skey-symmetry is violated** [must have $f(u,v) = -f(v,u)$].

However, the strategy is pointless anyway: why ship 8 crates from Edmonton to Calgary, and 3 creates back, instead of just shipping 5 crates from Edmonton to Calgary?

In effect, 3 of the 8 crates from Edmonton to Calgary are **cancelled** by the 3 crates from Calgary to Edmonton.

We represent this with a flow: we have $f(E,C) = 5$ and $f(C,E) = -5$.

In general, cancellation allows to represent shipments between two cities by a flow that is **positive along at most one of the two edges**.

Cancellation will arise implicitly in the algorithms.

- Suppose (u, v) has flow of $f(u, v)$.
- We may increase flow on edge (v, u) by some amount d.
- Mathematically, this must decrease $f(u, v)$ by d.
- We think of these d units as cancelling d units that are already on (u, v) (increase in one direction cancels flow in other).

Technical tools

Implicit summation. Let $X, Y \subseteq V$. Then

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

For example, flow-conservation

$$\sum_{v \in V} f(u,v) = 0 \quad \forall u \in V - \{s,t\}$$

can be written simply as

$$f(u,V) = 0 \quad \forall u \in V - \{s,t\}$$

Commonly occuring identities

1. For all $X \subseteq V$, we have $f(X,X) = 0$
   Because each $f(u,v)$ cancels $f(v,u)$, which $= -f(u,v)$ by skew symmetry.

2. For all $X, Y \subseteq V$, we have $f(X,Y) = -f(Y,X)$
   Generalisation of $f(X,X) = 0$, same reasoning.

3. For all $X, Y, Z \subseteq V$ with $X \cap Y = \emptyset$, we have

$$f(X \cup Y, Z) = f(X,Z) + f(Y,Z)$$

$$f(Z, X \cup Y) = f(Z,X) + f(Z,Y)$$

   Split summation into sum of two summations: one over X, one over Y.

The Ford-Fulkerson method

Three important ideas:

1. residual networks
2. augmenting paths
3. cuts

Method is iterative:

1. Start with $f(u, v) = 0$ for all $u, v \in V$
2. At each iteration, increase flow value by finding "augmenting path" — a path from source to sink along which we can increase flow — and then augment flow along this path.
3. Repeat until no augmenting path can be found.

Pseudocode:

```
initialise flow $f$ to 0
while there exists an augmenting path $p$ do
    augment flow $f$ along $p$
end while
```

Residual networks

Idea: Residual network consists of edges that can admit more flow.

Formally: Consider vertices u and v. Amount of *additional* flow we can push from u to v before exceeding capacity $c(u, v)$ is **residual capacity** of (u, v):

$$c_f(u, v) = c(u, v) - f(u, v).$$

Note: When flow is negative, then residual capacity $c_f(u, v)$ is greater than $c(u, v)$.

Interpretation:

- flow of $-x$ from u to v
- implies flow of x from v to u
- can be cancelled by pushing x units from u to v
- can then push another $c(u, v)$ from u to v
- can thus push total of $x + c(u, v) > c(u, v)$ from u to v

Given flow network $G = (V, E)$ and flow f, the **residual network** of G induced by f is $G_f = (V, E_f)$ with

$$E_f = \{(u, v) \in V \times V : \ c_f(u, v) > 0\}$$

i.e. each residual edge that can admit flow that is strictly positive.

$$G \quad u \underset{1}{\overset{3/5}{\rightleftarrows}} v \qquad \Longrightarrow \qquad G_f \quad u \underset{4}{\overset{2}{\rightleftarrows}} v$$

Note: $|E_f| \leq 2|E|$.

Relationship between flow in residual network and flow in original network.

Lemma 1. Let $G = (V, E)$ be a flow network, f be a flow in G, G_f be the residual network of G induced by f, and let f' be a flow in G_f.

Then flow sum $f + f'$ with

$$(f + f')(u, v) = f(u, v) + f'(u, v)$$

is a flow in G with value

$$|f + f'| = |f| + |f'|.$$

Proof. Must verify that skew symmetry, capacity constraints, and conservation are obeyed.

Skew symmetry:

$$\begin{aligned} (f + f')(u, v) &= f(u, v) + f'(u, v) \\ &= -f(v, u) - f'(v, u) \\ &= -(f(v, u) + f'(v, u)) \\ &= -(f + f')(v, u) \end{aligned}$$

Capacity constraints: Note f' is flow in G_f, so $f'(u,v) \le c_f(u,v)$. Since by def. $c_f(u,v) = c(u,v) - f(u,v)$,

$$\begin{aligned}(f+f')(u,v) &= f(u,v) + f'(u,v) \\ &\le f(u,v) + c_f(u,v) \\ &= f(u,v) + (c(u,v) - f(u,v)) \\ &= c(u,v)\end{aligned}$$

Flow conservation: For all $u \in V - \{s,t\}$,

$$\begin{aligned}\sum_{v\in V}(f+f')(u,v) &= \sum_{v\in V}(f(u,v) + f'(u,v)) \\ &= \sum_{v\in V} f(u,v) + \sum_{v\in V} f'(u,v) \\ &= 0 + 0 \\ &= 0\end{aligned}$$

Value:

$$\begin{aligned}|f+f'| &= \sum_{v\in V}(f+f')(s,v) \\ &= \sum_{v\in V}(f(s,v) + f'(s,v)) \\ &= \sum_{v\in V} f(s,v) + \sum_{v\in V} f'(s,v) \\ &= |f| + |f'|\end{aligned}$$

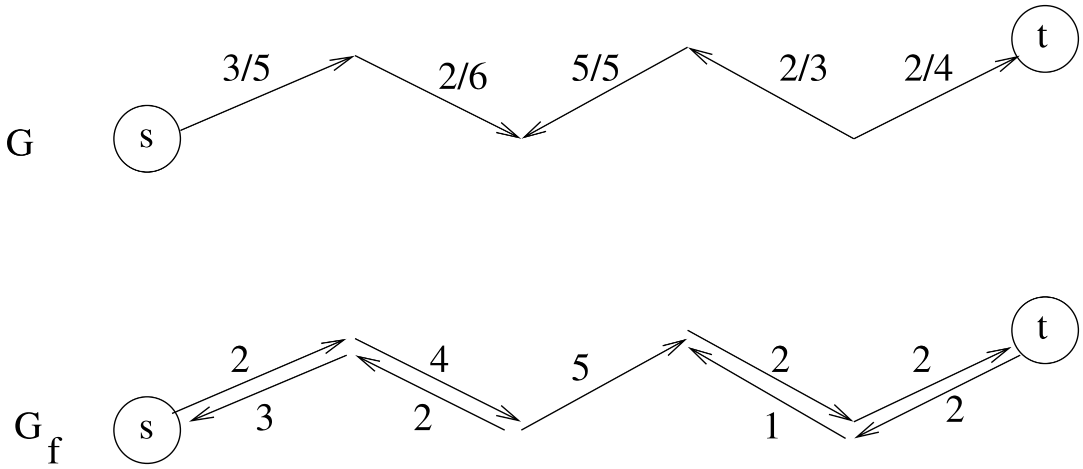
Augmenting paths

Given flow network $G = (V, E)$ and flow f, an **augmenting path** p is a simple path in residual network G_f.

Recall: each edge (u, v) in G_f admits some additional positive flow, obeying capacity constraint.

Flow value can be increased by

$$c_f(p) = \min_{(u,v)\in p} c_f(u, v).$$



In this example, augmentation by 2.

Lemma. Let $G = (V, E)$ be a flow network, f be a flow in G, and let p be an augmenting path in G_f. Define $f_p : V \times V \to \mathbb{R}$ by

$$f_p(u,v) = \begin{cases} c_f(p) & \text{if } (u,v) \text{ is on } p \\ -c_f(p) & \text{if } (v,u) \text{ is on } p \\ 0 & \text{otherwise} \end{cases}$$

Then f_p is a flow in G_f with value $|f_p| = c_f(p) > 0$.

Corollary. Let G, f, p, f_p be as above. Define $f' : V \times V \to \mathbb{R}$ by $f' = f + f_p$. Then f' is a flow in G with value $|f'| = |f| + |f_p| > |f|$.

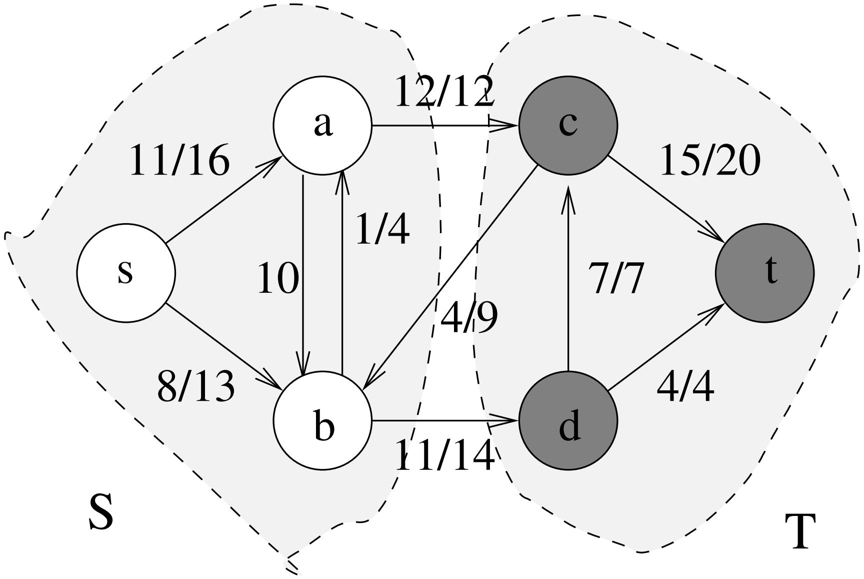
Proof. From last two lemmas.

Cuts of flow networks

A **cut** of flow network $G = (V, E)$ is partition of V into S and $T = V - S$ such that $s \in S$ and $t \in T$.

If f is a flow in G, then **net flow** across cut (S, T) is defined to be $f(S, T)$, capacity is $c(S, T)$.

A **minimum cut** is a cut with minimum capacity over all cuts.



Flow: $f(a, c) + f(b, c) + f(b, d) = 12 + (-4) + 11 = 19$
Note: can include negative flows.

Capacity: $c(a, c) + c(b, d) = 12 + 14 = 26$
Note: only non-negative values, only edges going from S to T are being accounted for.

Net flow across any cut is the same:

Lemma. Let f be a flow in a flow network G with source s and sink t, let (S,T) be a cut of G. Then the flow across (S,T) is $f(S,T) = |f|$.

Proof. Note $T = V - S$.

$$\begin{aligned} f(S,T) &= f(S,V) - f(S,S) \\ &= f(S,V) - 0 \\ &= f(s,V) + f(S-s,V) \\ &= f(s,V) + 0 \text{ by flow cons.} \\ &= |f| \text{ by def.} \end{aligned}$$

Corollary. The value of a flow f in a network G is upper-bounded by the capacity of any cut (S,T) in G.

Proof.

$$\begin{aligned} |f| &= f(S,T) \\ &= \sum_{u\in S}\sum_{v\in T} f(u,v) \\ &\leq \sum_{u\in S}\sum_{v\in T} c(u,v) \\ &= c(S,T) \end{aligned}$$

We know: maximum flow in a network is **bounded** by capacity of minimum cut. Following theorem shows **equality.**

Theorem. (Max-flow min-cut theorem)
If f is a flow in a flow network $G = (V, E)$ with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting path.
3. $|f| = c(S, T)$ for some cut (S, T) of G.

Remark. By last corollary, value of *any* flow is at most capacity of *any* cut.
In particular, value of *maximum* flow is at most capacity of *minimum* cut. Thus, if value of maximum flow equals capacity of *some* cut, then this must a minimum cut (can't be greater).

Proof of theorem.

(1) $\Longrightarrow$ (2)
Suppose that f is a maximum flow in G but G_f has an augmenting path p. Then by earlier corollary today (slide 4), flow sum $f + f_p$ is flow in G w/ value strictly greater than $|f|$, contradiction!

(2) $\Longrightarrow$ (3)
Suppose G_f doesn't have augmenting path, i.e., G_f contains no path from s to t. Define

$$\begin{array}{rcl} S & = & \{v \in V : \ \exists \text{ path from } s \text{ to } v \text{ in } G_f\} \\ T & = & V - S \end{array}$$

Partition (S, T) is a cut because $s \in S$ and $t \notin S$. For each $u \in S, v \in T$ we have $f(u, v) = c(u, v)$, because otherwise $(u, v) \in E_f$, v would be in S ($s \leadsto u \to v$). By earlier Lemma, $|f| = f(S, T) = c(S, T)$.

(3) $\Longrightarrow$ (1)
By earlier corollary, $|f| \leq c(S, T)$ for all cuts (S, T). Condition $|f| = c(S, T)$ implies that f is maximum flow.

Basic Ford-Fulkerson algorithm

```
Input: $G, s, t$
for each edge $(u, v) \in E$ do
    $f[u, v] \leftarrow 0$
    $f[v, u] \leftarrow 0$
end for
while $\exists$ path $p = s \rightsquigarrow t$ in res. network $G_f$ do
    $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \in p\}$
    for each edge $(u, v) \in p$ do
        $f[u, v] \leftarrow f[u, v] + c_f(p)$
        $f[v, u] \leftarrow -f[u, v]$
    end for
end while
```

An example: book page 659, figure 26.5

Running time

Depends on how augmenting paths in line 5 are determined.

If chosen poorly, algorithm might not even terminate:

- value of flow increases with each iteration, but
- need not even converge to maximum flow value (theoretically; problems arise with irrational capacities — can't be stored on finite-precision computers).

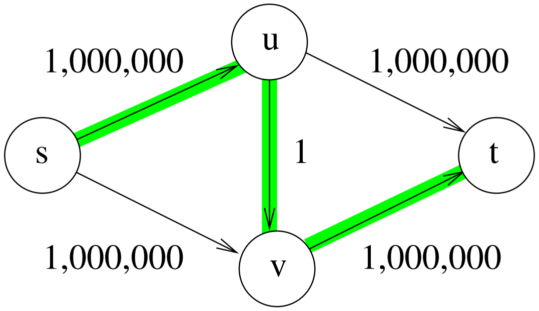
If chosen by using BFS, running time is polynomial. (We'll see later.)

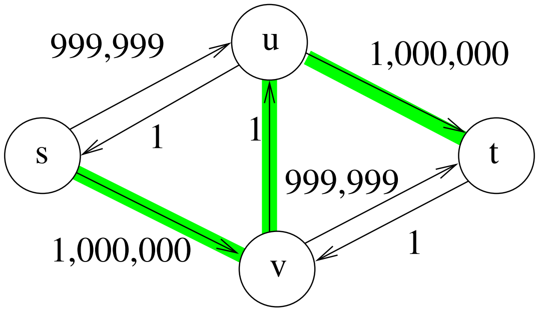
A simple analysis

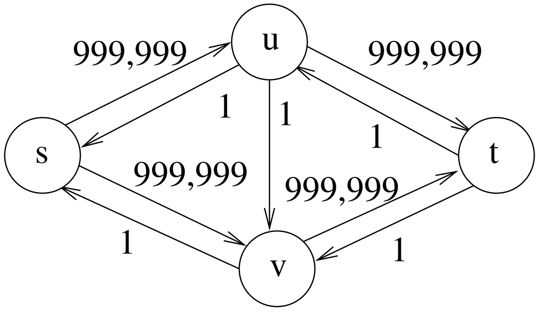
Assume **integral capacities**. Simple bound $O(E|f^*|)$ for running time when chosing paths **arbitrarily**, $|f^*|$ being **value of maximum flow**:

- **initialization** in lines 1–4 take $O(E)$
- **while** loop in lines 5–11 is executed at most $|f^*|$ times (value of flow increases by at least one unit in each iteration)
- work done **within while loop** takes $O(V + E) = O(E)$ if we use DFS or BFS

Problem case: Alg. takes $\Omega(E|f^*|)$ because (u, v) is always chosen to be part of augmenting path.
Here: $f^*| = 2,000,000$







An improvement: Edmonds-Karp

Idea: compute augmenting paths with BFS, picking **shortest** paths from source s to sink t (shortest w.r.t. # edges)

This variant has running time $O(VE^2)$, no longer dependent on value of maximum flow, $|f^*|$.

Def.: $\delta_f(u, v)$ is **shortest-path distance** from u to v in G_f (each edge has **unit distance**).

Lemma. If the Edmonds-Karp (EK) algorithm is run in flow network $G = (V, E)$ with source s and sink t, then $\forall v \in V - \{s, t\}$, $\delta_f(s, v)$ in residual network **increases monotonically** with each flow augmentation.

Proof (by contradiction). Assume for some $v \in V - \{s,t\}$, there is a first augmentation that causes shortest-path distance to **decrease**.

Let f be flow **before** this augmentation,
    f' be flow **afterward**.

Let v be vertex with **minimum** $\delta_{f'}(s,v)$ whose distance was decreased, so $\delta_{f'}(s,v) < \delta_f(s,v)$.

Let $p = s \rightsquigarrow u \to v$ be a shortest path from s to v in $G_{f'}$ so that $(u,v) \in E_{f'}$ and $\delta_{f'}(s,u) = \delta_{f'}(s,v) - 1$.

By choice of v, distance of u **did not decrease** (v has smallest $\delta_{f'}(s,\cdot)$), that is, $\delta_{f'}(s,u) \geq \delta_f(s,u)$.

Claim: $(u,v) \notin E_f$ (note: $(u,v) \in E_{f'}$ by constr.)
If we **had** $(u,v) \in E_f$, then also

$$\delta_f(s,v) \overset{(*)}{\leq} \delta_f(s,u) + 1 \leq \delta_{f'}(s,u) + 1 = \delta_{f'}(s,v)$$

Contradiction!

$(*)$ Consider a shortest path p from s to v, among cheapest such paths. In particular, p is not more expensive than a shortest path from s to v plus edge (u,v), which comes with “weight” one.

So far, so good. . .
. . . but how come $(u, v) \notin E_f$ and $(u, v) \in E_{f'}$???
. . . there was **no** residual capacity **before**,
but there **is** afterward???

Must be because algorithm has **increased** flow from v to u.

EK always augments flow along **shortest paths**, thus shortest path from s to u in G_f has (v, u) as last edge, and

$$\begin{aligned} \delta_f(s,v) &= \delta_f(s,u) - 1 \\ &\leq \delta_{f'}(s,u) - 1 \\ &= \delta_{f'}(s,v) - 2 \end{aligned}$$

Contradiction to assumption $\delta_{f'}(s, v) < \delta_f(s, v)$, so **no such vertex** v **exists**.

q.e.d.

We now use this lemma to prove. . .

Theorem. If EK is run on a flow network $G = (V, E)$ with source s and sink t, then the total # of flow augmentations is $O(VE)$.

Proof. We call edge (u, v) in G_f **critical** on augmenting path (AP) p if $c_f(p) = c_f(u, v)$.

Note:

- after augmenting along AP, all critical edges disappear from residual network
- at least one edge on any AP must be critical

We show that each edge can become critical at most $|V|/2 - 1$ times.

Let u, v be connected by some edge in E.

APs are **shortest paths**, thus when (u, v) is critical for the first time, we have $\delta_f(s, v) = \delta_f(s, u) + 1$.

Once flow is augmented, (u, v) **disappears** from residual network.

It cannot **reappear** on another AP until flow from u to v is **decreased**, i.e., (v, u) appears on an AP.

With f' as flow in G at this moment, $\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$.

With $\delta_f(s, v) \leq \delta_{f'}(s, v)$ by Lemma (shortest-path distances increase monotonically),

$$\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1 \geq \delta_f(s, v) + 1 = \delta_f(s, u) + 2$$

Thus, from time (u, v) becomes critical to time when it **next** becomes critical, **distance** of u from source (in residual network!) **increases** by ≥ 2.

Initially, distance is ≥ 0.

Intermediate vertices on a shortest path $s \rightsquigarrow u$ can't contain s, u, t [(u, v) on AP implies $u \neq t$].

Thus, until u becomes **unreachable** (if ever), distance is $\leq |V| - 2$, and (u, v) can become critical at most $(|V| - 2)/2 = |V|/2 - 1 = O(V)$ times.

$O(E)$ pairs of vertices with edge between them in residual network, thus **total # of critical edges** during entire execution of KE is $O(VE)$.

Each PA has **at least one critical edge** $\Longrightarrow$ *q.e.d.*

Each iteration takes $O(E)$ (find APs by BFS), thus **total running time** is $O(VE^2)$.

Although not bad, other algorithms (push-relabel method) are better, obtain $O(V^2E)$, or even $O(V^3)$.

An application of the FF method, other than flow

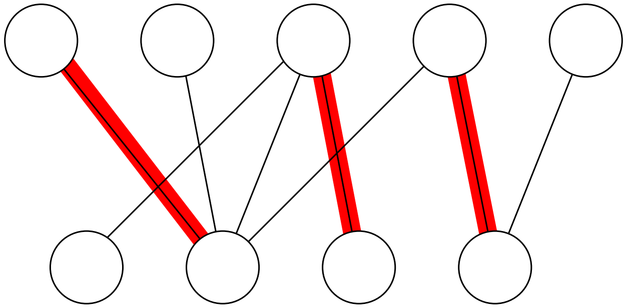
Many (combinatorial) problems can be reduced to max-flow problems, for instance, **maximum matchings in bipartite graphs**.

Given undirected, bipartite graph $G = (L \cup R, E)$ (all edges between L and R).

A **matching** is a subset $M \subseteq E$ s.t. $\forall v \in L \cup R$, at most one edge is incident on v.

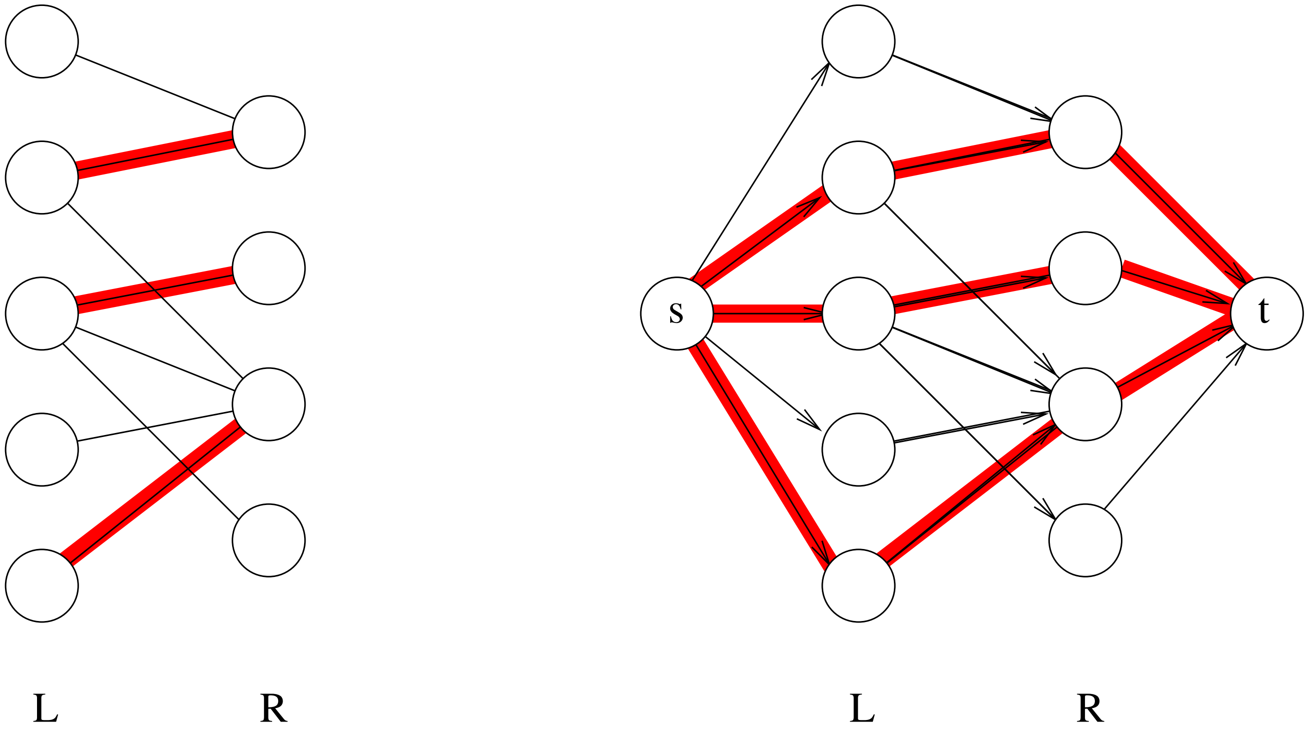
Vertex v is either **matched** (there *is* an edge incident), or **unmatched**.

A **maximum matching** is a matching of **maximum cardinality** (as opposed to a **maxim**<u>**al**</u> matching, which is not **extendable**).



Can use FF method to find a maximum matching (MM) in bipartite, undirected $G = (V, E)$ in time polynomial in $|V|$ and $|E|$.

Idea: construct flow network where flows correspond to matchings:



Basically,

- direct edges from L to R
- add source and sink + new edges
- each edge has capacity one
- $L - R$ edges used in matching have flow one
- other $L - R$ edges have flow zero

More formally,given undirected, bipartite $G = (V, E)$, construct **corresponding flow network** $G' = (V', E')$:

- let source s and sink t be new vertices not in V, let $V' = V \cup \{s, t\}$
- if vertex partition of G is $V = L \cup R$, the directed edges of G' are edges of G, directed from L to R, along with $|V|$ new edges

$$\begin{aligned} E' \;=\; & \{(s,u):\; u \in L\} \;\cup \\ & \{(u,v):\; u \in L, v \in R, (u,v) \in E\} \;\cup \\ & \{(v,t):\; v \in R\} \end{aligned}$$

- each edge in G' has unit capacity

If we assume that each edge in G has degree at least one, we have $|E| \geq |V|/2$ and thus $|E| \leq |E'| = |E| + |V| \leq 3|E|$, and so $|E'| = \Theta(E)$.

Def.: A flow f in flow network G is **integer-valued** if $f(u,v)$ is an integer for all $(u,v) \in V \times V$.

Lemma. Let $G = (V, E)$ be bipartite, directed graph, let G' be corresponding flow network.

1. If M is a matching in G, then there is an integer-valued flow f in G' with value $|f| = |M|$.
2. If f is an integer-valued flow in G', then there is a matching M in G with cardinality $|M| = |f|$.

Proof. Very simple.

(i) From matching M to IV flow in G'.

Define f:

- if $(u,v) \in M$, then
$$f(s,u) = f(u,v) = f(v,t) = 1$$
and
$$f(u,s) = f(v,u) = f(t,v) = -1$$
- other edges $(u,v) \in E'$, $f(u,v) = 0$

f satisfies *skew symmetry*, *capacity contstraints*, and *flow conservation* (obvious, but must check!).

Intuition: each edge $(u,v) \in M$ in the matching corresponds to one unit of flow in G' along
$$s \to u \to v \to t.$$

Paths induced by edges in M are **vertex-disjoint** (by def. of matching; except s&t)

Net flow across cut $(L \cup \{s\},\ R \cup \{t\})$ is equal to $|M|$

We know lemma "flow across any cut is equal to value of total flow", so $|f| = |M|$

(ii) From IV flow in G' to matching M.

Let f be an IV flow in G', let

$$M = \{(u,v):\ u \in L,\ v \in R,\ f(u,v) > 0\}$$

Each $u \in L$ has **exactly one entering edge**, (s,u)

At most one unit of pos. flow enters each $u \in L$

If so, by flow conservation, **one unit must leave**

Flow is IV, thus $\forall u \in L$, the one unit can **enter** on at most one edge, and can **leave** on at most one edge

Thus, one unit of flow enters $u \in L$ iff there is **exactly** one vertex $v \in R$ s.t. $f(u,v) = 1$

At most one edge leaving each $u \in L$ carries pos. flow

Analogous argument for for each $v \in R$

$\Longrightarrow$ our M is a matching

What about M's size?

$f(s,u) = 1$ for every matched $u \in L$

$f(u,v) = 0$ for every edge $(u,v) \in E - M$

Since $V' = L \cup R \cup \{s,t\} \;\Leftrightarrow\; R = V' - L - \{s,t\}$, we have

$$\begin{aligned} |M| &= f(L,R) \\ &= f(L,V') - f(L,L) - f(L,s) - f(L,t) \end{aligned}$$

Observe

$f(L,V') = 0$
flow conservation, $\sum_{v \in V'} f(u,v) = 0 \ \forall u \in V' - \{s,t\}$

$f(L,L) = 0$
skew-symmetry

$-f(L,s) = f(s,L)$
also skew-symmetry

$f(L,t) = 0$
no edges from L to t

Thus $|M| = 0 - 0 - (-f(s,L)) - 0 = f(s,L) = |f|$

q.e.d.

We would like to conclude that **maximum matching** in some bipartite graph G corresponds to **maximum flow** in corresponding flow network G'

and that we therefore can **compute the matching by solving the flow problem**

. . . but. . .

max flow algorithm **might** return flow with $f(u, v)$ not integral for some edge (u, v), even though $|f|$ must be integral (by lemma)

But not all is lost:

Theorem. If the capacity function c takes only integral values, then the maximum flow f produced by the Ford-Fulkerson method has the property that $|f|$ is integer-valued.
Moreover, for all vertices u and v, the value of $f(u, v)$ is an integer.

Proof. *Homework.*