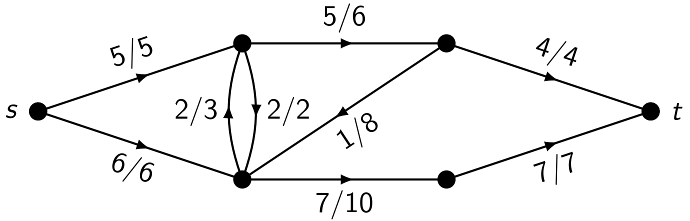
Why the Ford-Fulkerson algorithm looks so familiar
COMS20010 (Algorithms II)

John Lapinskas, University of Bristol

Recap of last lecture

A **flow network** (G, c, s, t) is a directed graph $G = (V, E)$, a **capacity** $c\colon E \to \mathbb{N}$, a **source** $s \in V$, and a **sink** $t \in V$, with $N^-(s) = N^+(t) = \emptyset$.



A **flow** is a function $f\colon E \to \mathbb{R}$ such that for all $e \in E$ and $v \in V \setminus \{s, t\}$:

- $0 \le f(e) \le c(e)$;
- $f^+(v) := \sum_{w \in N^+(v)} f(v, w) = \sum_{u \in N^-(v)} f(u, v) =: f^-(v)$.

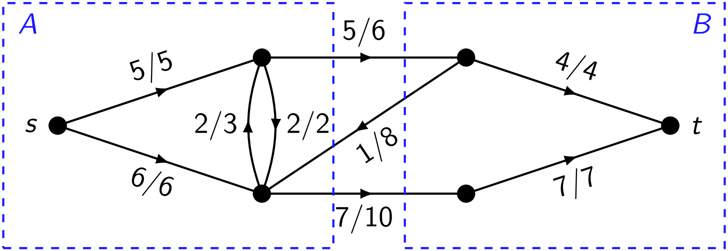
The **value** of f, denoted $v(f)$, is $f^+(s)$.

The problem: Find a **maximum flow**: a flow f maximising $v(f)$.

Theorem: The Ford-Fulkerson algorithm returns a maximum flow. It runs in time $O(v(f^*)|E|)$, where f^* is a maximum flow.

Theorem: There is always a maximum flow with integer values.

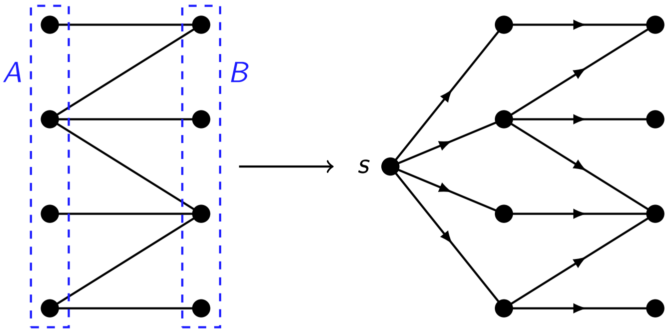
A **cut** is any pair of disjoint sets $A, B \subseteq V$ with $A \cup B = V$, $s \in A$ and $t \in V$. (So A and B partition V, the source is in A and the sink is in B.)



Max-flow min-cut theorem: The value of a maximum flow is equal to the minimum possible flow across a cut.

Matchings in bipartite graphs

Recall that a **matching** in a graph is a collection of disjoint edges.

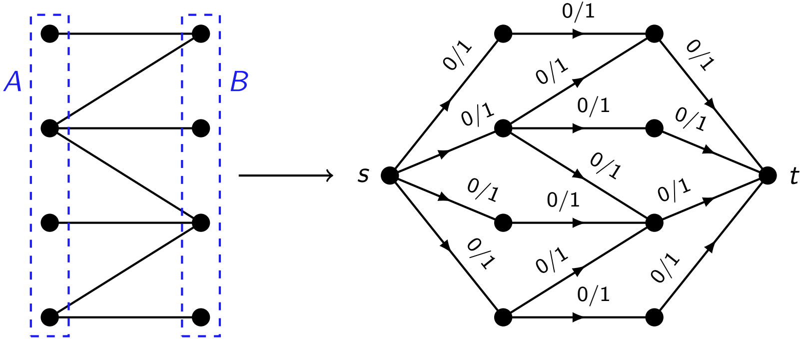


We can turn a graph G with bipartition (A, B) into a flow network:

- direct all G's edges from A to B;
- add a new vertex s and add every possible edge $s \to A$;

Matchings in bipartite graphs

Recall that a **matching** in a graph is a collection of disjoint edges.

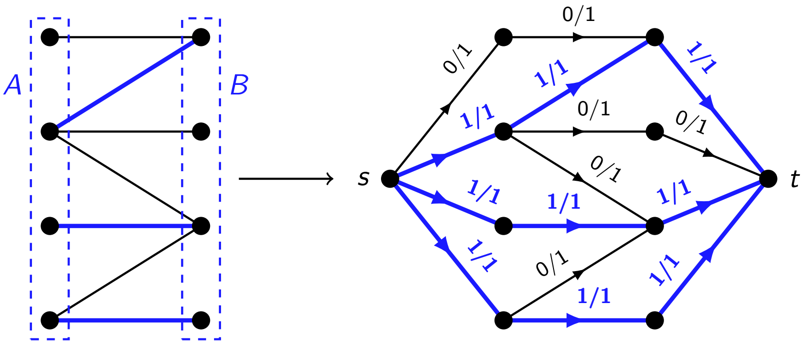


We can turn a graph G with bipartition (A, B) into a flow network:

- add a new vertex t and add every possible edge $t \to B$;
- give every edge capacity 1.

Matchings in bipartite graphs

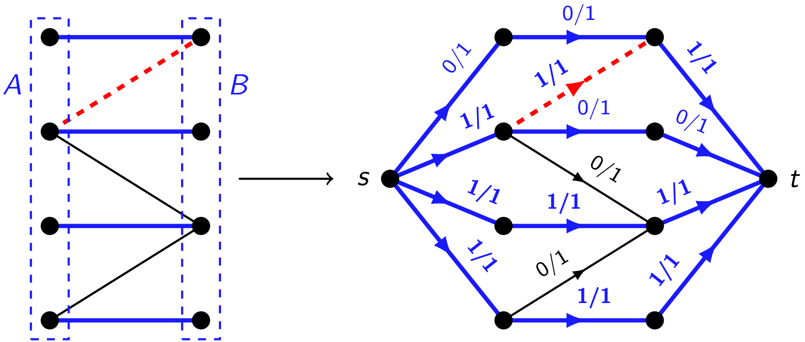
Recall that a **matching** in a graph is a collection of disjoint edges.



Then integer-valued maximum flows correspond to maximum matchings, and maximum matchings correspond to integer-valued maximum flows.

Matchings in bipartite graphs

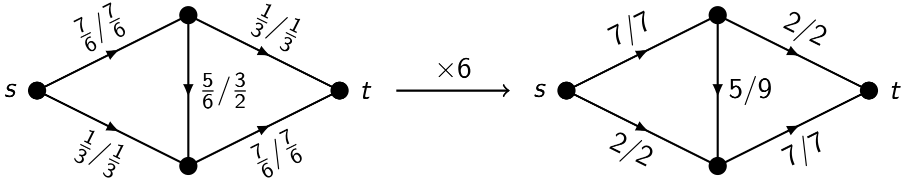
Recall that a **matching** in a graph is a collection of disjoint edges.



The augmenting paths are (essentially) the same for each.

In a real flow network, the capacities probably won't be integers...

How can we simulate rational weights?



Lemma 1: Let (G, c, s, t) be a flow network where c may take non-negative values in $\mathbb{Q}$ as well as $\mathbb{N}$. Then for all $k > 0$, f is a maximum flow in (G, c, s, t) if and only if kf is a maximum flow in (G, kc, s, t).

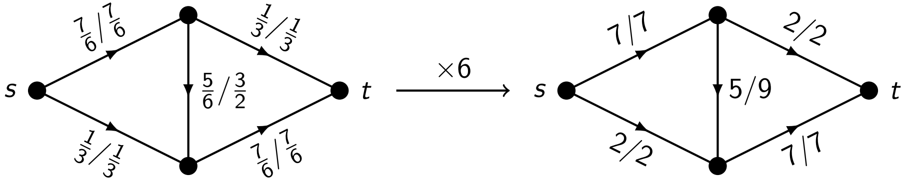
Proof: f is a flow in $(G, c, s, t) \Leftrightarrow kf$ is a flow in (G, kc, s, t). Moreover:

kf is maximum in $(G, kc, s, t) \Leftrightarrow \forall$ flows kg of (G, kc, s, t): $v(kf) \geq v(kg)$

$\Leftrightarrow \forall$ flows $\boldsymbol{g}$ of $(G, \boldsymbol{c}, s, t)$: $v(kf) \geq v(kg)$

In a real flow network, the capacities probably won't be integers...

How can we simulate rational weights?



Lemma 1: Let (G, c, s, t) be a flow network where c may take non-negative values in $\mathbb{Q}$ as well as $\mathbb{N}$. Then for all $k > 0$, f is a maximum flow in (G, c, s, t) if and only if kf is a maximum flow in (G, kc, s, t).

Proof: f is a flow in $(G, c, s, t) \Leftrightarrow kf$ is a flow in (G, kc, s, t). Moreover:

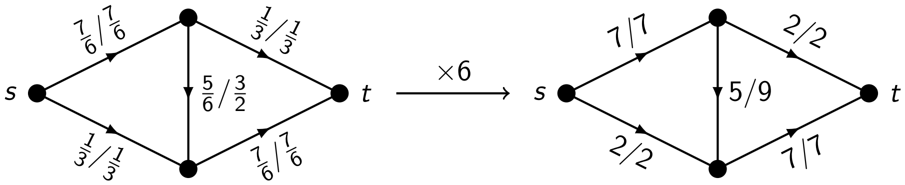
kf is maximum in $(G, kc, s, t) \Leftrightarrow \forall$ flows g of (G, c, s, t): $v(kf) \geq v(kg)$

$\Leftrightarrow \forall$ flows g of (G, c, s, t): $\boldsymbol{k}v(f) \geq \boldsymbol{k}v(g)$

Removing the simplifying assumptions: rational weights

In a real flow network, the capacities probably won't be integers...

How can we simulate rational weights?



Lemma 1: Let (G, c, s, t) be a flow network where c may take non-negative values in $\mathbb{Q}$ as well as $\mathbb{N}$. Then for all $k > 0$, f is a maximum flow in (G, c, s, t) if and only if kf is a maximum flow in (G, kc, s, t).

Proof: f is a flow in $(G, c, s, t) \Leftrightarrow kf$ is a flow in (G, kc, s, t). Moreover:

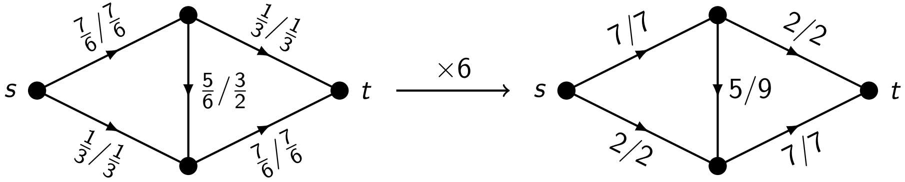
kf is maximum in $(G, kc, s, t) \Leftrightarrow \forall$ flows g of (G, c, s, t): $kv(f) \geq kv(g)$

$\Leftrightarrow \forall$ flows g of (G, c, s, t): $\boldsymbol{v(f)} \geq \boldsymbol{v(g)}$

Removing the simplifying assumptions: rational weights

In a real flow network, the capacities probably won't be integers...

How can we simulate rational weights?



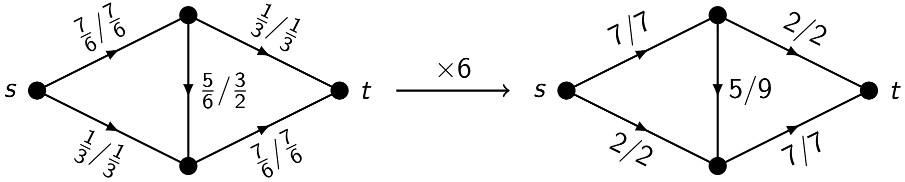
Lemma 1: Let (G, c, s, t) be a flow network where c may take non-negative values in $\mathbb{Q}$ as well as $\mathbb{N}$. Then for all $k > 0$, f is a maximum flow in (G, c, s, t) if and only if kf is a maximum flow in (G, kc, s, t).

Proof: f is a flow in $(G, c, s, t) \Leftrightarrow kf$ is a flow in (G, kc, s, t). Moreover:

$$kf \text{ is maximum in } (G, kc, s, t) \Leftrightarrow \forall \text{ flows } g \text{ of } (G, c, s, t)\text{: } v(f) \geq v(g)$$
$$\Leftrightarrow f \text{ is maximum in } (G, c, s, t).$$

In a real flow network, the capacities probably won't be integers...

How can we simulate rational weights?



Lemma 1: Let (G, c, s, t) be a flow network where c may take non-negative values in $\mathbb{Q}$ as well as $\mathbb{N}$. Then for all $k > 0$, f is a maximum flow in (G, c, s, t) if and only if kf is a maximum flow in (G, kc, s, t).

Proof: f is a flow in $(G, c, s, t) \Leftrightarrow kf$ is a flow in (G, kc, s, t). Moreover:

$$kf \text{ is maximum in } (G, kc, s, t) \Leftrightarrow f \text{ is maximum in } (G, c, s, t). \quad \square$$

So if the denominators of capacities in (G, c, s, t) are $b_1, \ldots, b_m$, then we find $L = \text{lcm}(b_1, \ldots, b_m)$, then find the max flow in (G, Lc, s, t).

A better algorithm: Edmonds-Karp

How can we simulate rational weights?

If the denominators of capacities in (G, c, s, t) are $b_1, \ldots, b_m$, then we find $L = \text{lcm}(b_1, \ldots, b_m)$, then find a maximum flow in (G, Lc, s, t). Then divide it by L to recover a maximum flow in (G, c, s, t).

Problem: Remember Ford-Fulkerson's running time depends on the value of a maximum flow — this could increase a lot!

In fact, if we allow **irrational** edge capacities, it may never terminate... We prove this on the problem sheet!

Solution: If we always pick an augmenting path with **as few edges as possible**, then we are guaranteed to terminate in $O(|V||E|^2)$ time, no matter how big the maximum flow is. (See CLRS 26.7 and 26.8.)

In other words, we just have to use breadth-first search on the residual graph G_f to find augmenting paths, rather than depth-first search! This is the **Edmonds-Karp** algorithm.