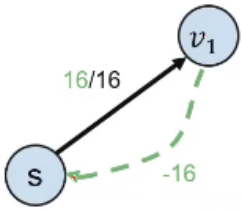
Definition

A **flow network** (G, c) consists of a directed graph $G = (V, E)$ together with
- a capacity function $c: E \to \mathbb{N}_{\geq 0}$ **(set to $c(u, v) = 0$ for $(u, v) \notin E$)**
- a source s, a sink t: in-degree$(s) = 0$, out-degree$(t) = 0$

- a flow is a function which returns the throughput as real number for each edge
  - capacity constraints
    - flow $f(u, v) <$ capacity $c(u, v)$
  - skew symmetry
    - $f(u, v) = -f(v, u)$



We only note down positive flows.
The other direction of flow is "implicit"

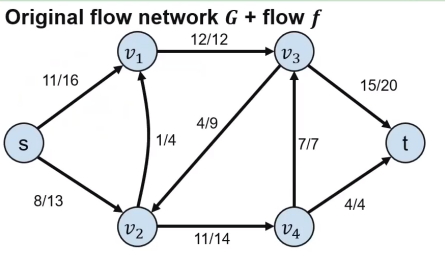
    -
  - conservation of flow
    - incoming flow = outcoming flow
    - except for source and sink

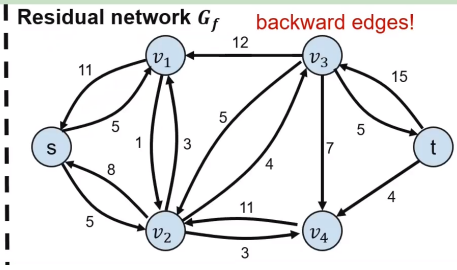
Ford-Fulkerson-Method

Input: Given a flow network (G, c) with source node s, and sink node t
Output: Compute a flow f from s to t of maximum value

- residual capacity:
  - $c_f(u, v) = c(u, v) - f(u, v)$
- residual network

The **residual network** $G_f(V, E_f)$ is given via the edge set
$$E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}$$
G_f is a flow network with capacities c_f.





  -
- augmenting path
  - path from s to t in G_f
  - $c_f(p)$ = minimal residual capacity
- augmenting a flow f along a path p
  - for each flow along the path
    - add/subtract min residual capacity

$$f_p(u,v) = \begin{cases} c_p(u,v) & \text{if } (u,v) \in p \\ -c_p(u,v) & \text{if } (v,u) \in p \\ 0 & \text{otherwise} \end{cases} \quad \Rightarrow f_p \text{ is a flow in } G_f$$

  - *
  - * $f + f_p$
- pseudocode
  - ```
    Initialize flow f with 0
    while there exists an augmenting path p in G_f do
        augment f along p (by f_p)
    return f
    ```
- correctness
  - augmented version is a flow

Capacity constraint:

$$(f + f_p)(u,v) = f(u,v) + f_p(u,v) \le f(u,v) + c_f(u,v) \le f(u,v) + \big(c(u,v) - f(u,v)\big) = c(u,v)$$

Conservation of flow:

Let $u \in V \setminus \{s,t\}$. $(f + f_p)(u,V) = f(u,V) + f_p(u,V) = 0 + 0 = 0$

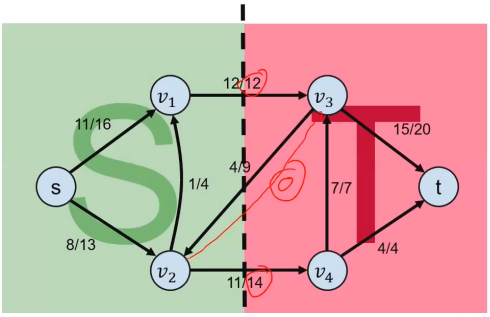
Skew Symmetry:

$$(f + f_p)(u,v) = f(u,v) + f_p(u,v) = -f(v,u) - f_p(v,u) = -(f(v,u) + f_p(v,u)) = -(f + f_p)(v,u)$$

    - *
  - min-cut = maximum flow

Max-Flow = Min-Cut Theorem

An s-t-cut (S,T) in G is a partition of V ($V = S \cup T$, $S \cap T = \emptyset$) with $s \in S, t \in T$.

- 

The **capacity** of a cut is

$$c(S,T) = \sum_{u \in S, v \in T} c(u,v)$$

The **flow** over a cut is

$$f(S,T) = \sum_{u \in S, v \in T} f(u,v)$$

Lemma A: For each flow f and each cut (S,T) we have $|f| = f(S,T)$.

Proof:

$$f(S,T) = f(S,V) - f(S,V \setminus T) = f(S,V) - f(S,S) = f(S,V) = f(s,V) + f(S \setminus \{s\}, V) = f(s,V) = |f|$$

- insert definition
- (omitted simple proof, using symmetry) $f(S,S) = 0$
- $V \setminus T = S$
- $f(S \setminus \{s\}, V) = 0$ as $f(u,V) = 0$ for all $u \neq s,t$ due to flow conservation

Notation:

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

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

Theorem: Let f be an s-t-flow in a flow network $(G = (V, E), c)$.
The following are equivalent

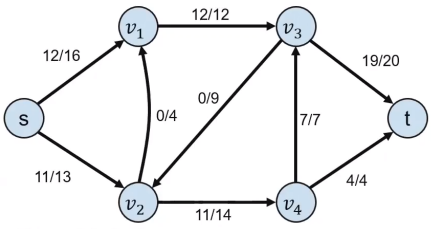
1. flow f is a maximum flow
2. The residual network G_f contains **no** augmenting path
3. $|f| = c(S, T)$ for a cut (S, T)

- **Proof 1→ 2:**
  Let f be a maximum flow.
  Assume there is an augmenting path p in G_f.
  Then $f + f_p$ is a flow with $|f + f_p| > |f|$, a contradiction.
- **Proof 2→ 3:**
  Assume that G_f has no s-t path. Define:
  $S := \{ v \in V \mid$ there is an s-v path in $G_f \}$, $T := V \setminus S$ (note $T \neq \emptyset$)
  Then (S, T) is a cut and we have $f(u, v) = c(u, v)$ for all $u \in S, v \in T$.

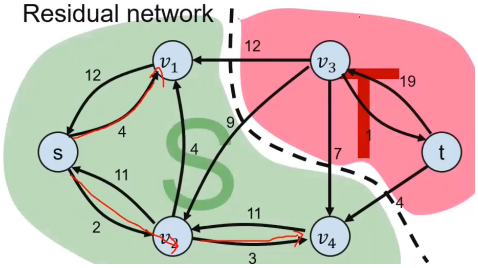
$$|f| = f(S,T) = \sum_{u \in S, v \in T} f(u,v) = \sum_{u \in S, v \in T} c(u,v) = c(S,T).$$

Lemma A

Flow network



Residual network



- **Proof 3→ 1:**
  Pick the cut (S, T) from 3 and an arbitrary flow f:

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

Lemma A

Theorem: If all capacities are integral, then the Ford-Fulkerson-method finds a maximum flow f after at most at most f^* (value of a maximum flow) augmentations. Further, all values of $f(u, v)$ are integral.

Proof:

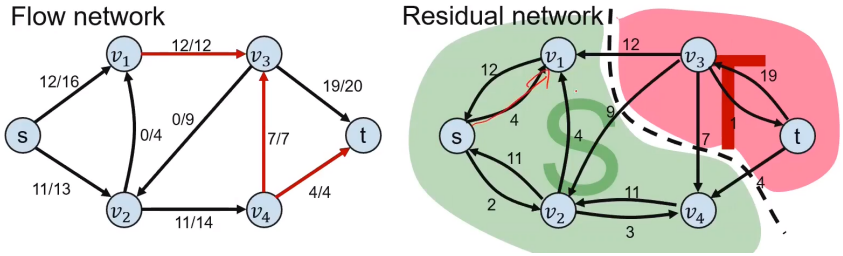
Integrality sounds innocent but it is crucial for many applications.

- $|f|$ increases by at least one in each iteration → at most f^* iterations.
- By induction: The capacity of each augmenting path is integral, and $f(u, v)$ remains integral.

- does not hold for real-valued capacities

- finding a min-cut

1. Compute a max-flow
2. Let S consist of the vertices that one can reach from s via non-critical edges (An edge (u, v) is critical if $c(u, v) = f(u, v)$)
3. Let $T = V \setminus S$.



Edmonds-Karp-Algorithm

- faster version of the Ford-Fulkerson-Method
  - FF can be very slow even for small networks
  - runtime depends on min capacity

- pseudo-code

```
Initialize flow f with 0
while there exists an augmenting path in G_f do
    find a shortest augmenting path p
    augment f along p
return f
```

- **Theorem:** The runtime of the Edmonds–Karp–Algorithm is polynomial in the size of the network, regardless of the value of the capacities.

f_i: the current flow before the i-th augmentation step.

$G_i = G_{f_i}$: the residual network of f_i (note that $G_1 = G$)

$level_i(v)$: the distance of s to v in G_i

Proof: $level_{i+1}(v) \geq level_i(v)$.

For $v = s$ the claim is trivial as $level_i(s) = 0$.

If there is no such path, $level_{i+1}(v) = \infty$, done!

Let $v \neq s$ with minimal $level_{i+1}(v)$, contradicting the claim.

consider a shortest path from s to v in G_{i+1}

(*) We have $level_{i+1}(v) = level_{i+1}(u) + 1 \geq level_i(u) + 1$

due to choice of v, u doesn't contradict the claim !

(*)

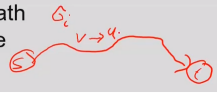
Case $(u, v) \in G_i$: $level_i(v) \leq level_i(u) + 1 \leq level_{i+1}(v)$

(levels of adjacent vertices can differ by at most one)

Case $(u, v) \notin G_i$: (v, u) must be an edge in the i-th augmenting path (v, u) must lie on the shortest s-t-path in G_i as (u, v) in G_{i+1}, hence

$level_i(v) = level_i(u) - 1 \leq level_i(u) + 1 \leq level_{i+1}(v)$

trivial (*)



During the execution of the Edmonds-Karp-Algorithm, any edge (u, v) disappears from the residual graph at most $V/2$ times.

Proof: Suppose $u \to v$ is in two residual graphs G_i and G_{j+1}, but not in any of the intermediate residual graphs G_{i+1}, G_j for some i < j.

- $u \to v$ must be on the i-th augmenting path: $level_i(v) = level_i(u) + 1$,
- $v \to u$ must be on the j-th augmenting path: $level_j(v) = level_j(u) - 1$.

By Lemma B, we have

$$level_j(u) = level_j(v) + 1 \geq level_i(v) + 1 = level_i(u) + 2.$$

The distance from s to u increased by at least 2 between the disappearance and reappearance of $u \to v$. Since every level is either less than V or infinite, the number of disappearances is at most $V/2$

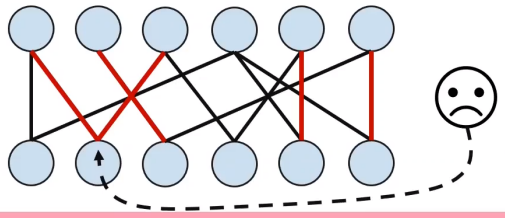
The Edmonds-Karp-Algorithm computes a maximum flow with runtime $O(|E|^2|V|)$.

Proof:

- As a special case of the Ford-Fulkerson Method, it computes a max-flow!
- Every augmentation requires $O(|E|)$ steps as a shortest augmenting paths can be found via BFS and also G_f can be built with $O(|E|)$ steps.
- There are at most $O(|E| \cdot |V|)$ augmentations ($\leq |V|/2$ augmentations per edge)

Application

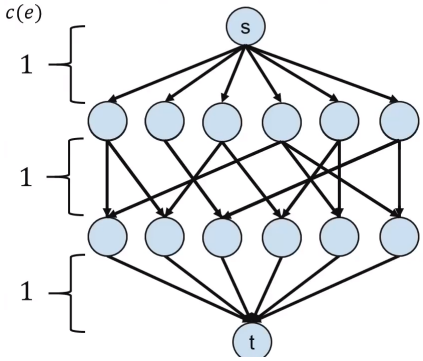
A matching of a graph $G = (V, E)$ is a subset $M \subseteq E$ of the edges such any vertex $v \in V$ has at most one adjacent edge in M.



Maximum Bipartite Cardinality Matching: Given a bipartite graph, compute a matching that contains as many edges as possible.

Here, important special case: The graph is bipartite!
(match customers to products, ads to customers, …)

Maximum Bipartite Cardinality Matching: Given a bipartite graph, compute a matching that contains as many edges as possible.



Reduction to Max-Flow

Given: Bipartite graph $B = (X \cup Y, E)$.

1. Build flow network
   $\widetilde{B} = (X \cup Y \cup \{s, t\}, E', c)$ with
   $E' = \{(s, x) | x \in X\} \cup \{(y, t) | y \in Y\}$
   $\cup \{(x, y) | \{x, y\} \in E\}$
   and $c(e') = 1$ for all $e' \in E'$.
2. Compute an **integral** max-flow of $(\tilde{B}, c)$
3. Return the edges (of the original graph) with flow 1