Network Flow

CS 4104: Data and Algorithm Analysis

Yoseph Berhanu Alebachew

May 11, 2025

Virginia Tech

Table of contents

Introduction

Maximum Flow and Minimum Cut

- Two rich algorithmic problems.
- Fundamental problems in combinatorial optimization.
- Beautiful mathematical duality between flows and cuts.
- Numerous non-trivial applications:
  - Bipartite matching.
  - Data mining.
  - Project selection.
  - Airline scheduling.
  - Baseball elimination.
  - Image segmentation.
  - Network connectivity.
  - Network reliability.
  - Distributed computing.
  - Egalitarian stable matching.
  - Security of statistical data.
  - Network intrusion detection.
  - Multi-camera scene reconstruction.
  - Gene function prediction.

Flow Networks

Flow Networks

- Use directed graphs to model transportation networks:
  - Edges carry traffic and have capacities.
  - Nodes act as switches.
  - Source nodes generate traffic, sink nodes absorb traffic.
- A flow network is a directed graph $G = (V, E)$
  - Each edge $e \in E$ has a capacity $c_e > 0$.
  - There is a single source node $s \in V$.
  - There is a single sink node $t \in V$.
  - Nodes other than s and t are internal.
- In a flow network $G = (V, E)$, an s-t flow is a function $f : E \to \mathbb{R}^+$ such that:
  - **Capacity conditions**: For each $e \in E$, $0 \leq f(e) \leq c(e)$.
  - **Conservation conditions**: For each internal node v,

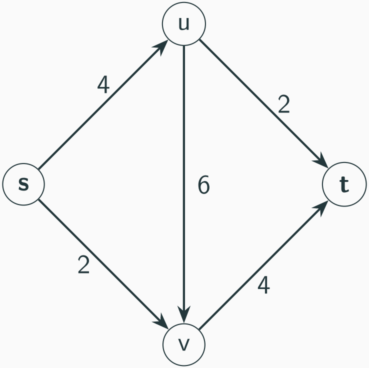
$$\sum_{\text{e into v}} f(e) = \sum_{\text{e out of v}} f(e)$$

- The value of a flow is $\nu(f) = \sum_{\text{e out of s}} f(e)$.

Maximum-Flow Problem

- **Input**: A flow network G
- **Solution**: The flow with the largest value in G, where the maximum is taken over all possible flows on G.
- Output should assign a flow value to each edge in the graph.
- The flow on each edge should satisfy the capacity condition.
- The flow into and out of each internal node should satisfy the conservation conditions.
- The value of the output flow, i.e., the total flow out of the source node in the output flow, must be the largest over all possible flows on G.
- Assumptions:
    1. No edges enter s, no edges leave t.
    2. There is at least one edge incident on each node.
    3. All edge capacities are integers.

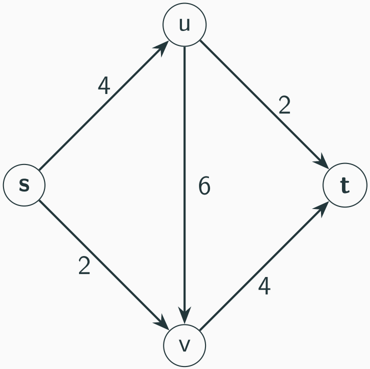
Examples of Flows



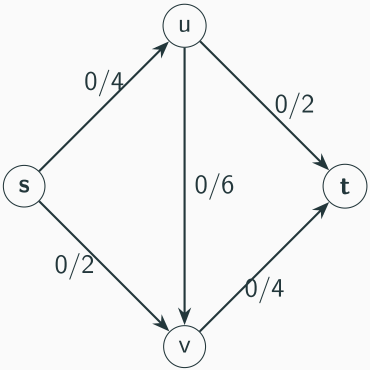
Greedy Algorithm

- No known dynamic programming algorithm.
- Let us take a greedy approach.
    1. Start with zero flow along all edges.
    2. Find an s-t path and push as much flow along it as possible.
    3. Idea to increase flow:
        - Push flow along edges with leftover capacity.
        - If needed, undo flow on edges already carrying flow.

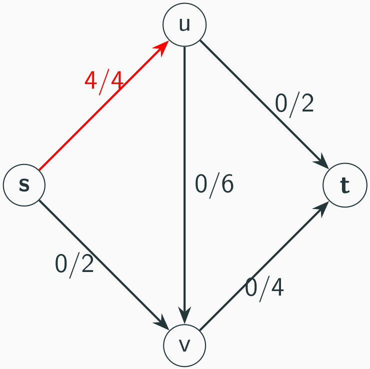
Examples of Flows



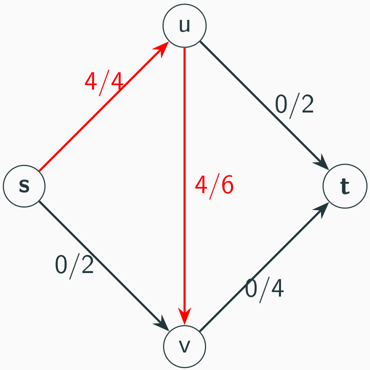
Examples of Flows



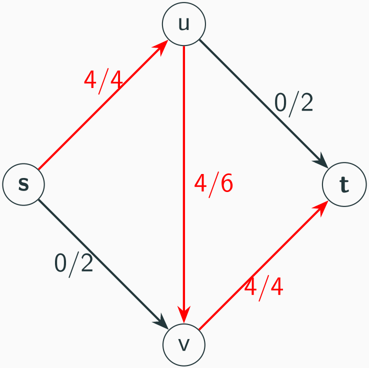
Examples of Flows



Examples of Flows



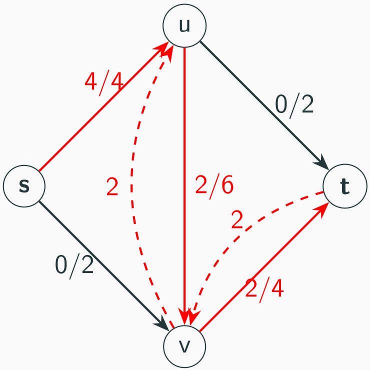
Examples of Flows



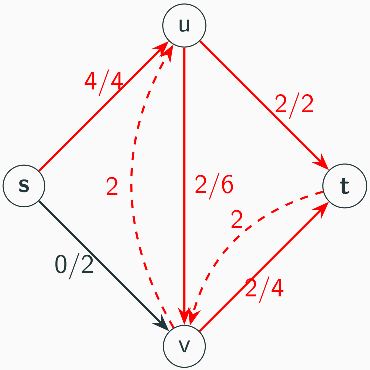
$v(f) = 4$

s-v has not used

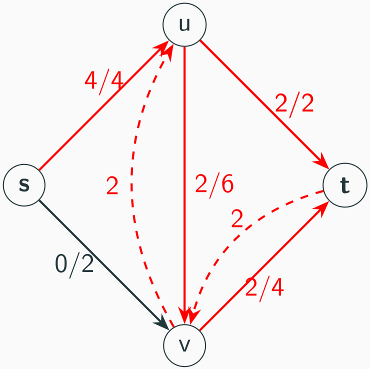
Examples of Flows



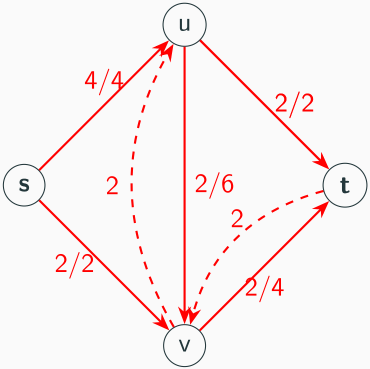
Examples of Flows



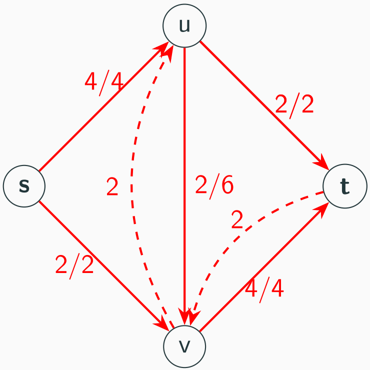
Examples of Flows



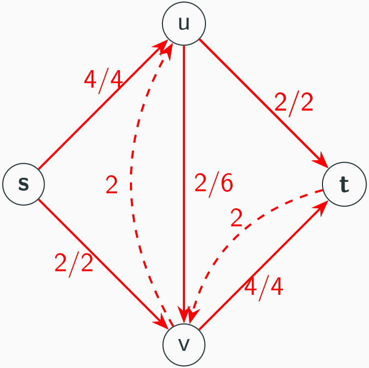
Examples of Flows



Examples of Flows



Examples of Flows



$v(f) = 6$

Ford-Fulkerson Algorithm

Residual Graph

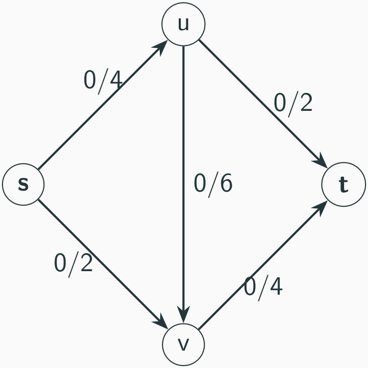
- Given a flow network $G = (V, E)$ and a flow f on G, the residual graph G_f of G with respect to f is a directed graph such that:
    1. (Nodes) G_f has the same nodes as G.
    2. (Forward edges) For each edge $e = (u, v) \in E$ such that $f(e) < c(e)$, G_f contains the edge (u, v) with a residual capacity $c(e) - f(e)$.
    3. (Backward edges) For each edge $e \in E$ such that $f(e) > 0$, G_f contains the edge $e' = (v, u)$ with a residual capacity $f(e)$.

Augmenting Paths in a Residual Graph

- Let P be a simple s-t path in G_f.
- $b = \text{bottleneck}(P, f)$ is the minimum residual capacity of any edge in P.
- The following operation augment(f, P) yields a new flow f' in G:
  - e is forward edge in $G_f \Rightarrow$ flow increases along e in G.
  - $e = (u, v)$ is backward edge in $G_f \Rightarrow$ flow decreases along (v, u) in G.

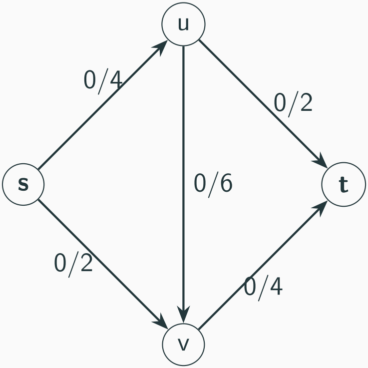
Running Example for Ford-Fulkerson Algorithm

Flow Graph

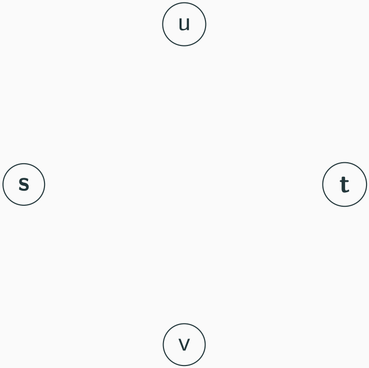


Running Example for Ford-Fulkerson Algorithm

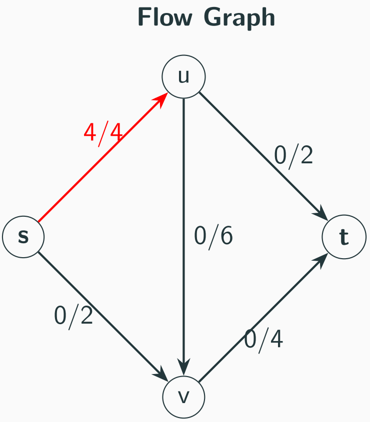
Flow Graph

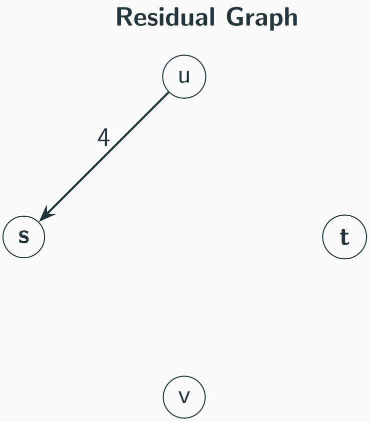


Residual Graph

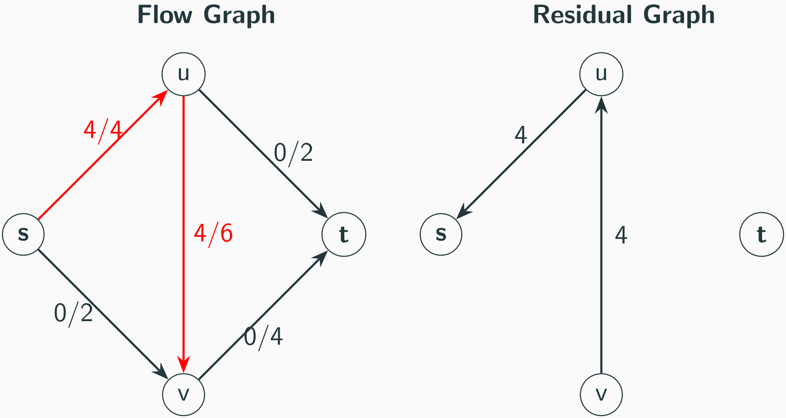


Running Example for Ford-Fulkerson Algorithm

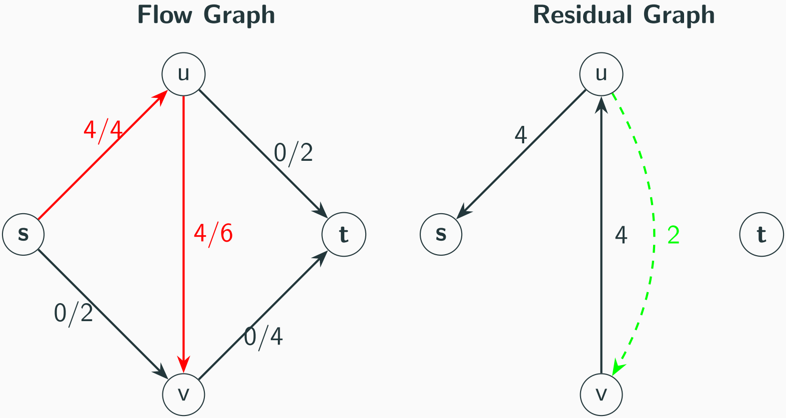




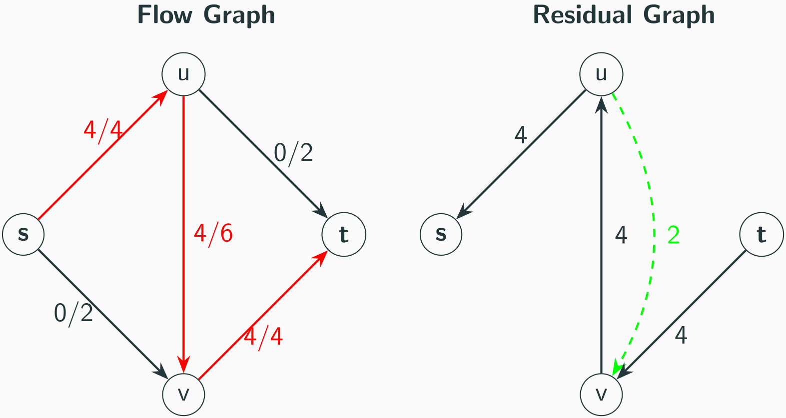
Running Example for Ford-Fulkerson Algorithm



Running Example for Ford-Fulkerson Algorithm

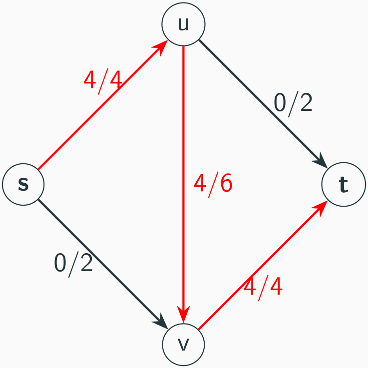


Running Example for Ford-Fulkerson Algorithm

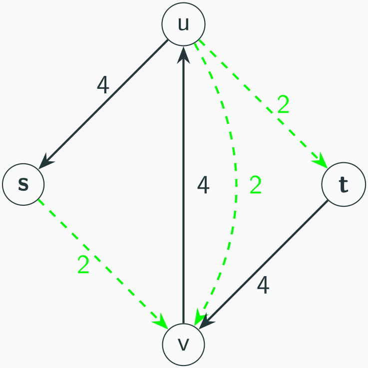


Running Example for Ford-Fulkerson Algorithm

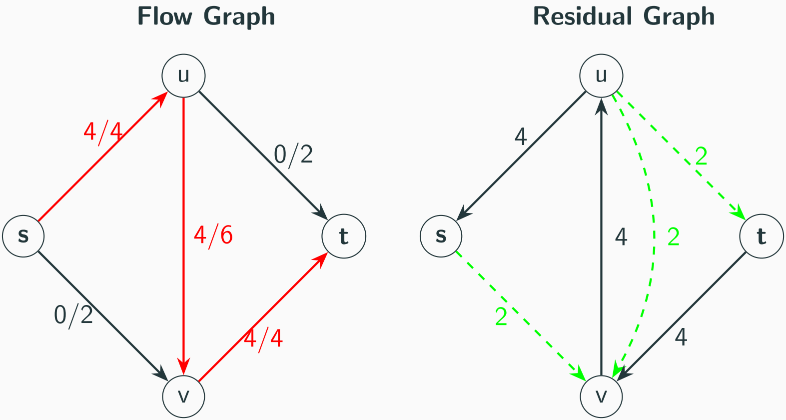
Flow Graph



Residual Graph

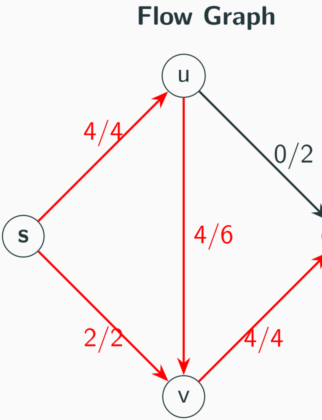


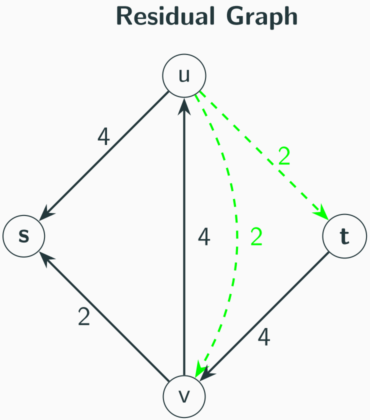
Running Example for Ford-Fulkerson Algorithm



- $v(f) = 4$
- Use the residual graph to find a path from s to t
- Let's say $P' = \{s, v, u, t\}$

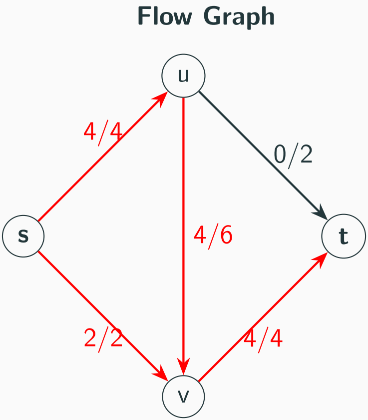
Running Example for Ford-Fulkerson Algorithm

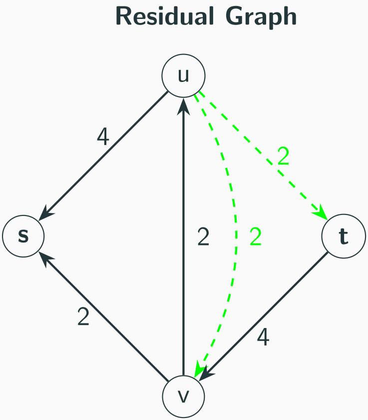




- $v(f) = 4$
- Use the residual graph to find a path from s to t
- Let's say $P' = \{s, v, u, t\}$

Running Example for Ford-Fulkerson Algorithm

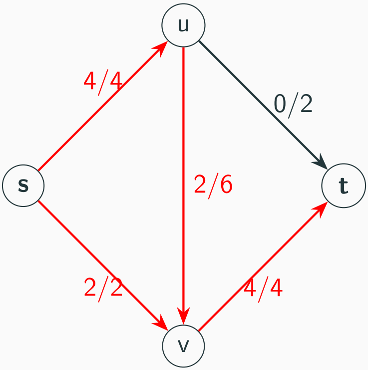




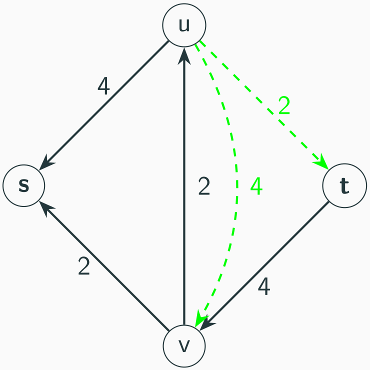
- $v(f) = 4$
- Use the residual graph to find a path from s to t
- Let's say $P' = \{s, v, u, t\}$

Running Example for Ford-Fulkerson Algorithm

Flow Graph



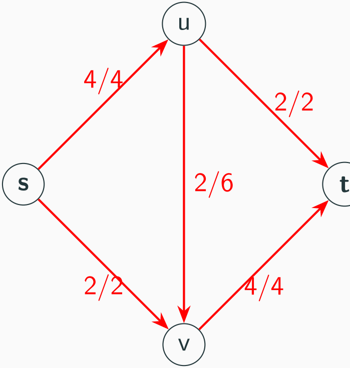
Residual Graph



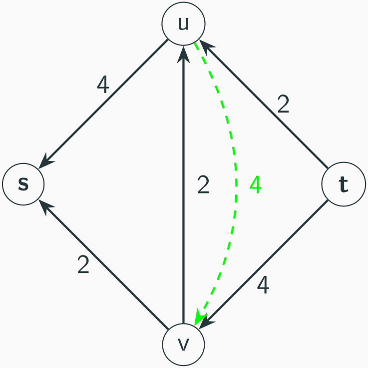
- $v(f) = 4$
- Use the residual graph to find a path from s to t
- Let's say $P' = \{s, v, u, t\}$

Running Example for Ford-Fulkerson Algorithm

Flow Graph

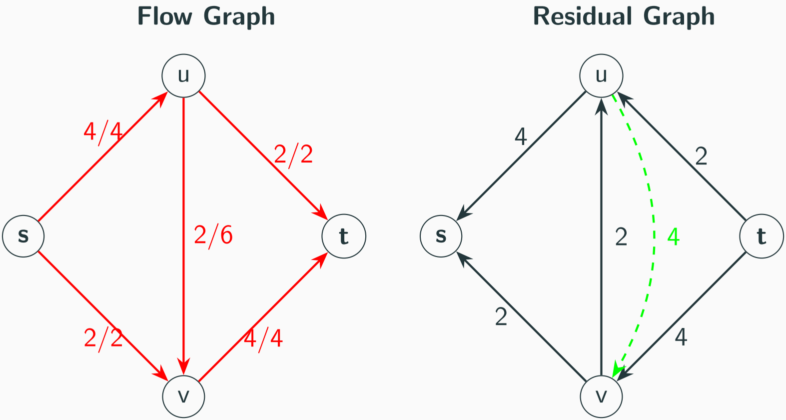


Residual Graph



- $v(f) = 4$
- Use the residual graph to find a path from s to t
- Let's say $P' = \{s, v, u, t\}$

Running Example for Ford-Fulkerson Algorithm



- $v(f) = 6$
- We don't have a path from s to t in the residual graph
- This indicates that our max flow is 6

Ford-Fulkerson Algorithm Pseudocode

Algorithm 1 Ford-Fulkerson Algorithm

Require: Graph G with capacities $c(u, v)$, source s, sink t

Ensure: Maximum flow f from s to t

```
initialize flow f(u,v) = 0 for all edges (u,v) ∈ G
while there is a path p from s to t in the residual graph G_f do
    find residual capacity c_f(p) = min{c_f(u,v) | (u,v) ∈ p}
    for all edges (u,v) in p do
        increase flow f(u,v) by c_f(p)
        decrease flow f(v,u) by c_f(p)
    end for
end while
return f
```

s-t Cuts and Capacities

s-t Cuts and Capacities

- An s-t cut is a partition of V into sets A and B such that $s \in A$ and $t \in B$.
- Capacity of the cut (A, B) is:

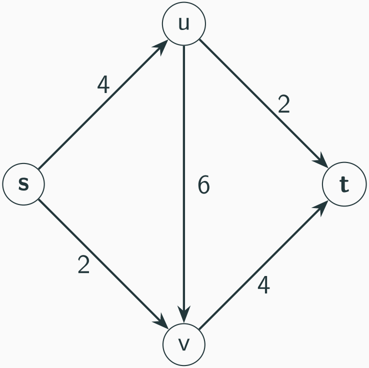
$$c(A, B) = \sum_{\text{e out of A}} c(e) = \sum_{\mathrm{e}=(u,v), u \in A, v \in B} c(e)$$

- Intuition: For every flow f and for every s-t cut (A, B), $\nu(f) \leq c(A, B)$.

Flows and Cuts

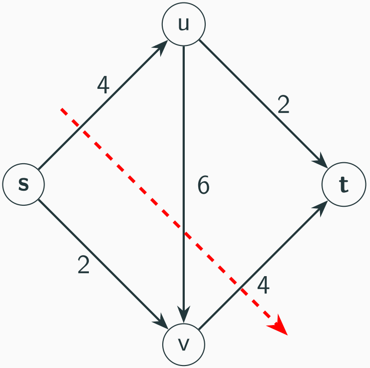
- Let $\overline{f}$ denote the flow computed by the Ford-Fulkerson algorithm.
- Enough to show $\exists s$-t cut (A^*, B^*) such that $\nu(\overline{f}) = c(A^*, B^*)$.
- When the algorithm terminates, the residual graph has no s-t path.
- Claim: If f is an s-t flow such that G_f has no s-t path, then there is an s-t cut (A^*, B^*) such that $\nu(f) = c(A^*, B^*)$.

Examples of Min Cut



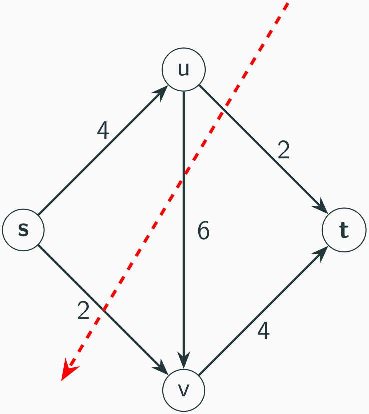
Min cut value:

Examples of Min Cut



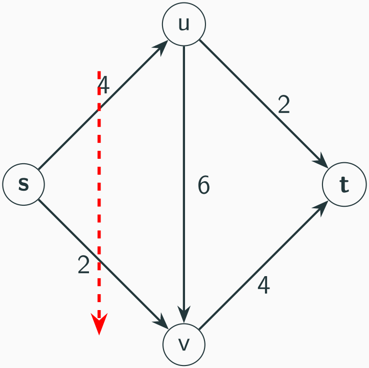
Min cut value: 8

Examples of Min Cut



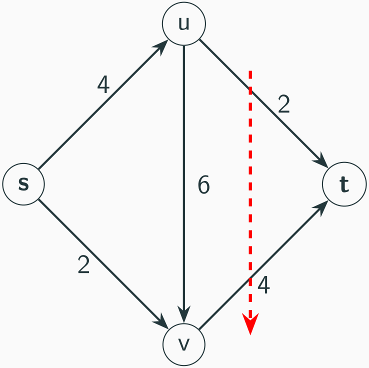
Min cut value: 10

Examples of Min Cut



Min cut value: 6

Examples of Min Cut



Min cut value: 6

Max-Flow Min-Cut Theorem

- The flow $\overline{f}$ computed by the Ford-Fulkerson algorithm is a maximum flow.
- Given a flow of maximum value, we can compute a minimum s-t cut in $O(m)$ time.
- In every flow network, there is a flow f and a cut (A, B) such that $\nu(f) = c(A, B)$.
- Max-Flow Min-Cut Theorem: In every flow network, the maximum value of an s-t flow is equal to the minimum capacity of an s-t cut.
- Corollary: If all capacities in a flow network are integers, then there is a maximum flow f where $f(e)$, the value of the flow on edge e, is an integer for every edge e in G.