IN4344 Advanced Algorithms

Lecture 8 – Max flow, Approximation Algorithms, TSP

Yuki Murakami

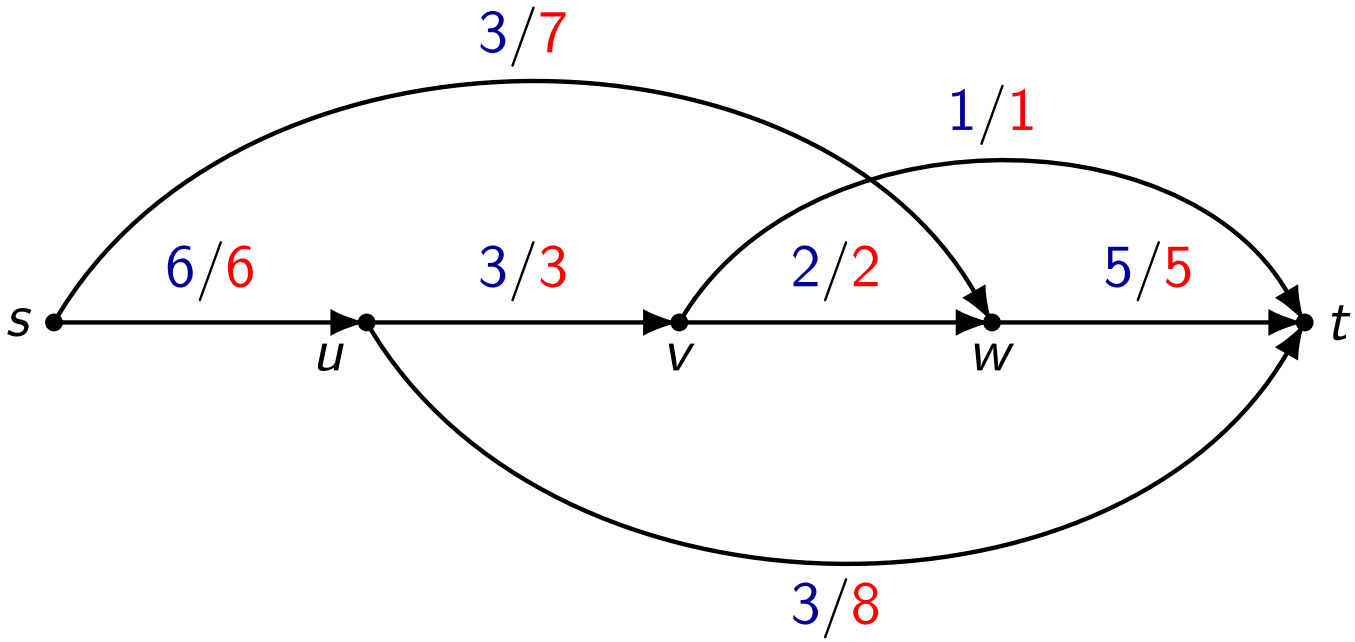
Delft University of Technology

October 2, 2023

- WebLab outage? Deadline extended to tonight.
- WebLab groups. If you accidentally create new groups, please let the TA's know.
- Things allowed in exam: non-graphical calculator, straightedge.
- Things not allowed in exam: Notes, smartwatches, phones, etc.
- Vraaguur 3rd October, from 14:00 - 15:45?

Max Flow

- Given is a network with distinct vertices s and t.
- Each arc has a certain **capacity** $b_{uv} \geq 0$.
- We want to find a largest possible **flow** from s to t.
- Such that the **capacities** of the arcs are not exceeded
- and there is **conservation of flow** at each vertex except s and t.



Network with **flow** and **capacities**.

Mathematical Formulation

Problem

MAX FLOW

Given: directed graph $D = (V, A)$, vertices $s, t \in V$ and a capacity $b_{uv} \geq 0$, for each arc $(u, v) \in A$.

Find: a **value** $f_{uv} \geq 0$, for each arc $(u, v) \in A$ such that:

(i) there is **conservation of flow** in each vertex except s and t:

$$\sum_{(u,v)\in A} f_{uv} = \sum_{(v,w)\in A} f_{vw}, \quad \forall v \in V \setminus \{s, t\}$$

(ii) the flow does not exceed the **capacities** of the arcs:

$$f_{uv} \leq b_{uv}, \qquad \forall (u, v) \in A$$

(iii) the **net outflow** at s is **maximized**:

$$\sum_{(s,v)\in A} f_{sv} - \sum_{(u,s)\in A} f_{us} \qquad \text{is maximized.}$$

Observation

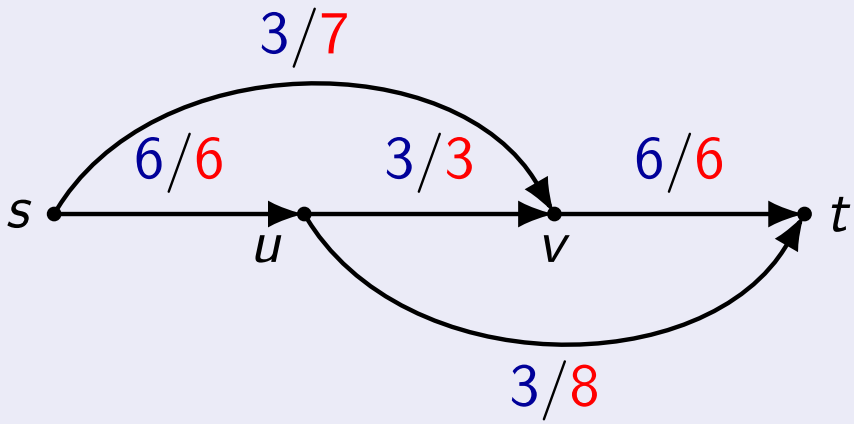
The **net outflow at s** is equal to the **net inflow at t**:

$$\sum_{(s,v)\in A} f_{sv} - \sum_{(u,s)\in A} f_{us} = \sum_{(u,t)\in A} f_{ut} - \sum_{(t,v)\in A} f_{tv}$$

This is the **value** of the flow.

Question (1)

What is the value of the flow indicated in blue in the network below?



Given is a directed graph $D = (V, a)$. Let M be the node-arc incidence matrix, and let b_{uv} be the capacity of arc $(u, v) \in A$.

Decision variables:

- f_{uv}: the flow on arc (u, v)
- w: the value of the flow from s to t

$$
\begin{array}{lrcl}
\max & & & w \\
\text{s.t} & Mf + \begin{pmatrix} -1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} w & = & 0 \\
& f & \leq & b \\
& f,\ w \in \mathbb{Z}_{\geq 0} & &
\end{array}
$$

We can view w as adding an extra arc on which the flow goes back from t to s.

The constraint matrix

Notice, the constraint matrix for the max flow problem is as follows:

$$\begin{pmatrix} M' \\ I \end{pmatrix},$$

where M' is the matrix M with the extra column

$$\begin{pmatrix} -1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix},$$

which is still a node-arc incidence matrix, and hence TUM.

The constraint matrix

Theorem (12.4)

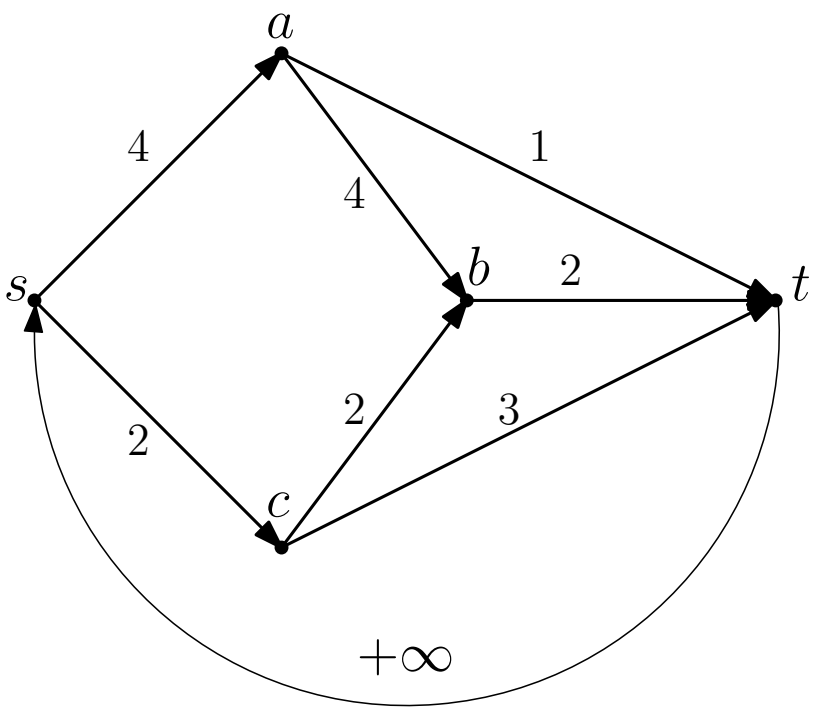
If A is TUM, then the matrix

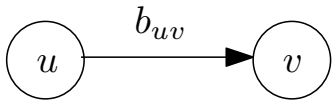
$$\begin{pmatrix} A \\ I \end{pmatrix}$$

is TUM.

So, the constraint matrix for Max flow is TUM and if the capacities are integral, then we can drop the integrality restrictions, and solve Max flow as an LP!

Example





$$
\begin{array}{lrcrcrcrcrcrcrcrcl}
\max & & & & & & & & & & & & & w & & \\
\text{s.t.} & f_{sa} & + & f_{sc} & & & & & & & & & - & w & = & 0 \\
& -f_{sa} & & & + & f_{ab} & & & f_{at} & & & & & & = & 0 \\
& & & & - & f_{ab} & - & f_{cb} & & + & f_{bt} & & & & = & 0 \\
& & - & f_{sc} & & & + & f_{cb} & + & & & + & f_{ct} & & = & 0 \\
& & & & & & & & - & f_{at} & - & f_{bt} & - & f_{ct} + & w & = & 0 \\
& f_{sa} & & & & & & & & & & & & & \leq & 4 \\
& & & f_{sc} & & & & & & & & & & & \leq & 2 \\
& & & & & f_{ab} & & & & & & & & & \leq & 4 \\
& & & & & & & f_{cb} & & & & & & & \leq & 2 \\
& & & & & & & & & f_{at} & & & & & \leq & 1 \\
& & & & & & & & & & & f_{bt} & & & \leq & 2 \\
& & & & & & & & & & & & & f_{st} & \leq & 3
\end{array}
$$

$$
f_{uv} \geq 0, \text{ for all } (u, v) \in A, w \geq 0
$$

Primal:

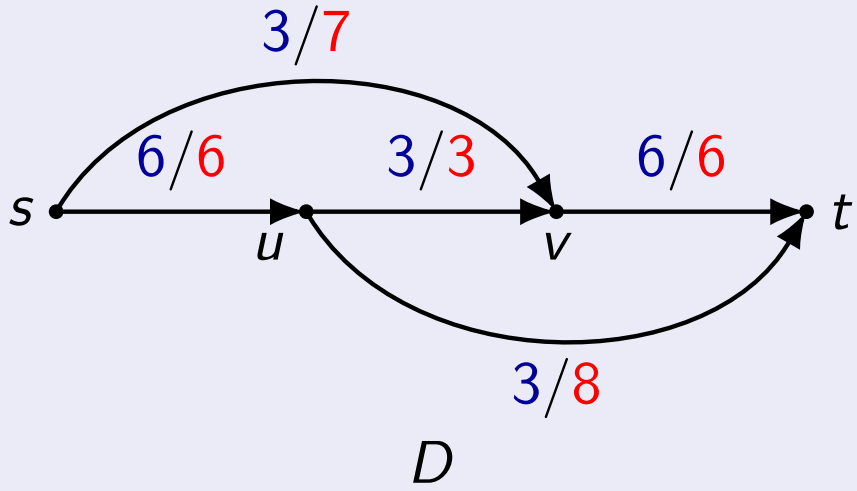
$$
\begin{aligned}
\max \quad & w \\
\text{s.t.} \quad & Mf + \begin{pmatrix} -1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} w = 0 \\
& f \leq b \\
& f, w \geq 0
\end{aligned}
$$

Dual:

$$
\begin{aligned}
\min \quad & \sum_{(u,v) \in A} b_{uv} \gamma_{uv} & \\
\text{s.t.} \quad & \pi_u - \pi_v + \gamma_{uv} \geq 0 & \forall (u,v) \in A \\
& -\pi_s + \pi_t \geq 1 & \\
& \pi_v \in \mathbb{R} & \forall v \in V \\
& \gamma_{uv} \geq 0 & \forall (u,v) \in A
\end{aligned}
$$

Question (2)

Prove that there does **not** exist a flow with **value 15** or more in the network below.

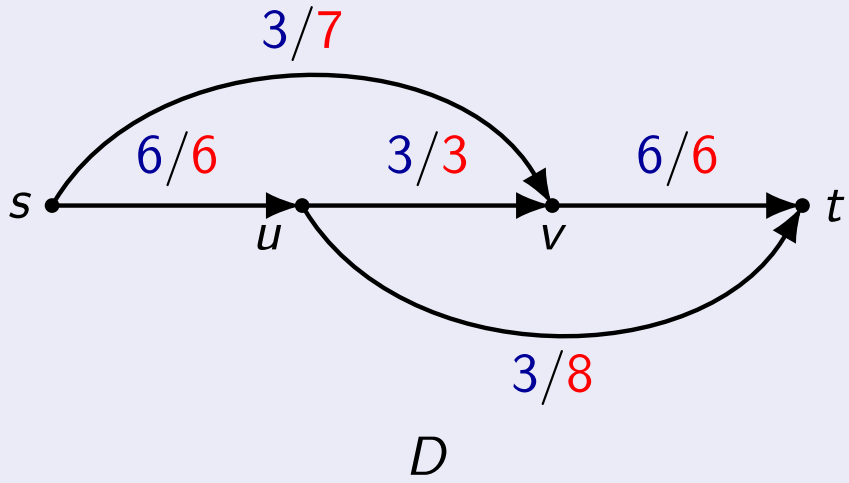


Answer

The total capacity on the incoming arcs to t is 6+8=14.

Question (3)

Prove that there does **not** exist a flow with **value 14** or more in the network below.



D

Answer

The total capacity on the arcs leaving s is 7+6=13.

Both cases are examples of a so-called "s-t cut" in the graph, i.e., a partition of the vertex set such that s belongs to one set and t to the other set.

Definition

- An s-t **cut** in a directed graph $D = (V, A)$ is a partition $(W, V \setminus W)$ of the vertices such that $s \in W$ and $t \in V \setminus W = \overline{W}$.
- The **capacity** of the cut $(W, \overline{W})$ is

$$\text{capacity}(W, \overline{W}) = \sum_{(u,v) \in A \mid u \in W,\, v \in \overline{W}} b_{uv}$$

Notice: The capacity of a cut is the sum of the capacity on the "forward" arcs of the cut, i.e., arcs that have a tail in W and a head in $\overline{W}$.

Lemma

An s-t cut $(W, \overline{W})$ determines a feasible solution to the dual with objective function value equal to capacity$(W, \overline{W})$.

Set:
$\pi_v = 0$ for all $v \in W$,
$\pi_v = 1$ for all $v \in \overline{W}$,
$\gamma_{uv} = 1$ if $u \in W,\ v \in \overline{W}$,
$\gamma_{uv} = 0$ for all other $(u, v) \in A$
Find a minimum $s - t$ cut.

$$
\begin{aligned}
\min \quad & \sum_{(u,v)\in A} b_{uv}\gamma_{uv} & \\
\text{s.t.} \quad & \pi_u - \pi_v + \gamma_{uv} \geq 0 & \forall (u, v) \in A \\
& -\pi_s + \pi_t \geq 1 & \\
& \pi_v \in \mathbb{R} & \forall v \in V
\end{aligned}
$$

Theorem (Max Flow Min Cut)

(i) For each feasible s-t flow f and for each s-t cut $(W, \overline{W})$ holds that value$(f) \leq$ capacity$(W, \overline{W})$. (**Weak duality!**)

(ii) The **maximum** value of an s-t **flow** is equal to the **minimum** capacity of an s-t **cut**. (**Strong duality!**)

Theorem (Complementary Slackness for Max Flow)

A flow f and a cut $(W, \overline{W})$ are both optimal if and only if:

- $f_{uv} = b_{uv}$ for all $(u, v) \in A$ with $u \in W$ and $v \in \overline{W}$ ("forward arcs");
- $f_{uv} = 0$ for all $(u, v) \in A$ with $u \in \overline{W}$ and $v \in W$ ("backward arcs").

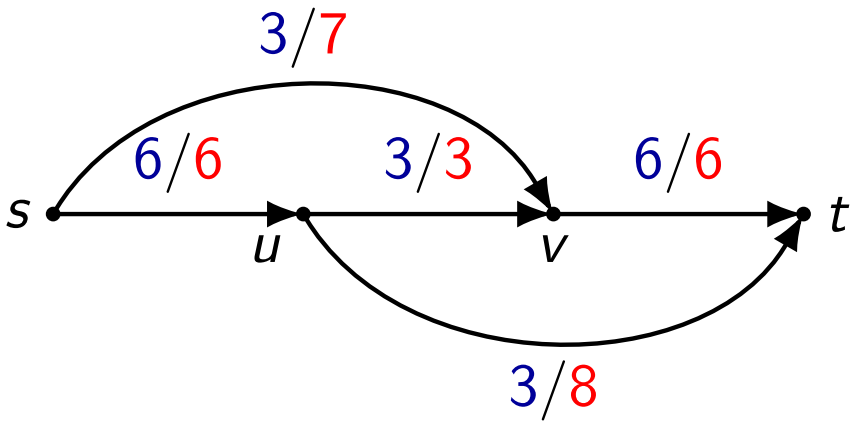
Algorithm of Ford-Fulkerson (1962)

- Algorithm for the MAX FLOW problem
- The idea is as follows:
  - **Start** with **0** flow on each arc.
  - **Augment** the flow over a **path** from s to t.
  - If the path goes **"against the flow"** on some arcs, then **decrease** the flow on those arcs.
  - Repeat until there is no **augmenting path** anymore.
  - To find the augmenting paths, we use an **auxiliary directed graph**.

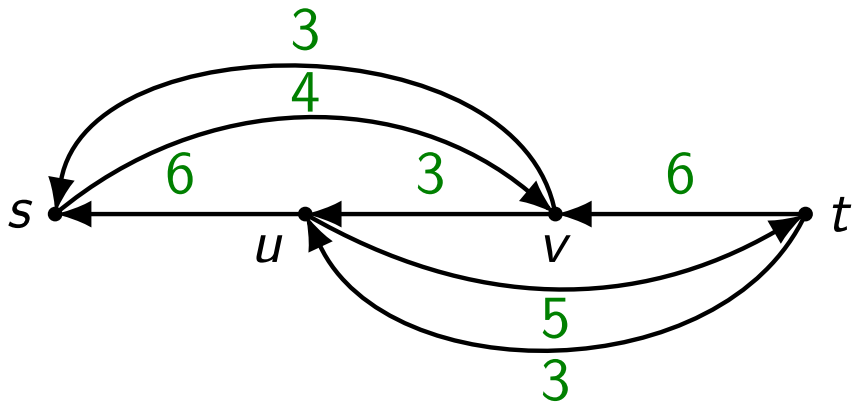
Auxiliary directed graph

For a given flow f, the auxiliary directed graph D_f has the same vertices as D and the following arcs:

- for each arc (u, v) of D with $f_{uv} < b_{uv}$
  - D_f gets an arc (u, v) with "capacity" label $b_{uv} - f_{uv}$
  - because we can increase the flow by at most $b_{uv} - f_{uv}$
- for each arc (u, v) of D with $f_{uv} > 0$
  - D_f gets an arc (v, u) with "capacity" label f_{uv}
  - because we can decrease the flow by at most f_{uv}.



D



D_f

Ford-Fulkerson in pseudo code

1. $f_{uv} := 0$ for all $(u, v) \in A$
2. Construct D_f with the same vertices as D and
   for each arc (u, v) of D:
   if $f_{uv} < b_{uv}$ then D_f gets an arc (u, v) with label $\bar{b}_{uv} = b_{uv} - f_{uv}$
   if $f_{uv} > 0$ then D_f gets an arc (v, u) with label $\bar{b}_{uv} = f_{uv}$
3. **Case 1:** there is a directed **path** P **from** s **to** t **in** D_f.

   $$\alpha := \min\{\bar{b}_{uv} \mid (u, v) \text{ lies on } P\}$$

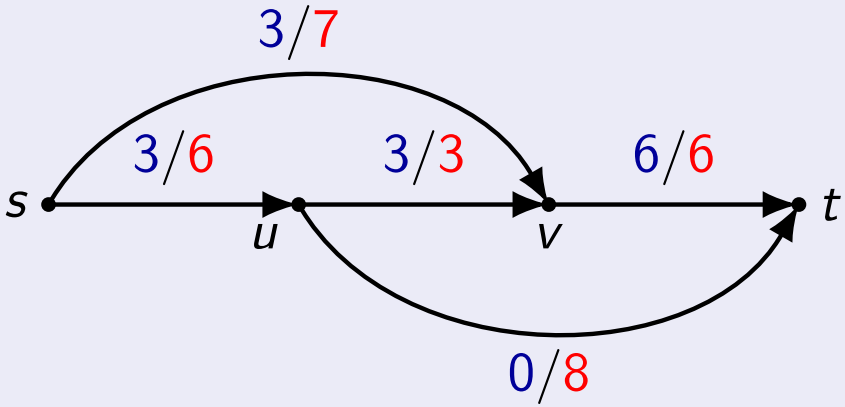
   Augment the flow f **by** α **along** P

   Go to (2).
4. **Case 2:** there is **no path** from s to t in D_f. Define:

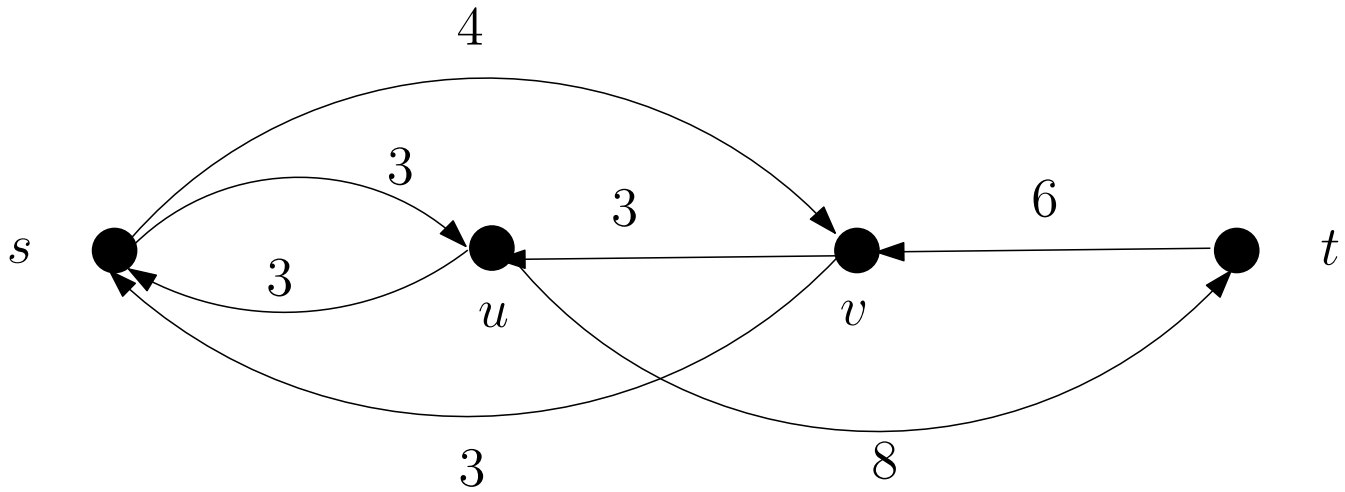
   $W := \{u \in V \mid$ there is a path from s to u in $D_f\}$
   $(W, \bar{W})$ is an s-t **cut** with **capacity** equal to
   the **value** of f.

Example

Suppose that, after a few iterations, we have found the following flow. Find a maximum flow by continuing the Ford-Fulkerson algorithm.

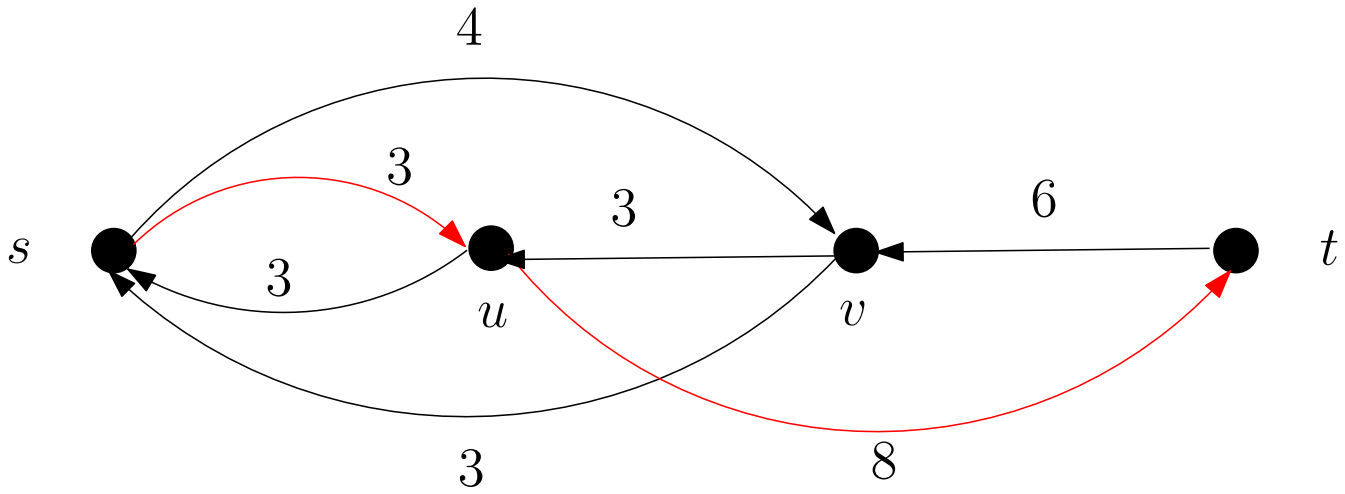


The current flow from s to t is 6.

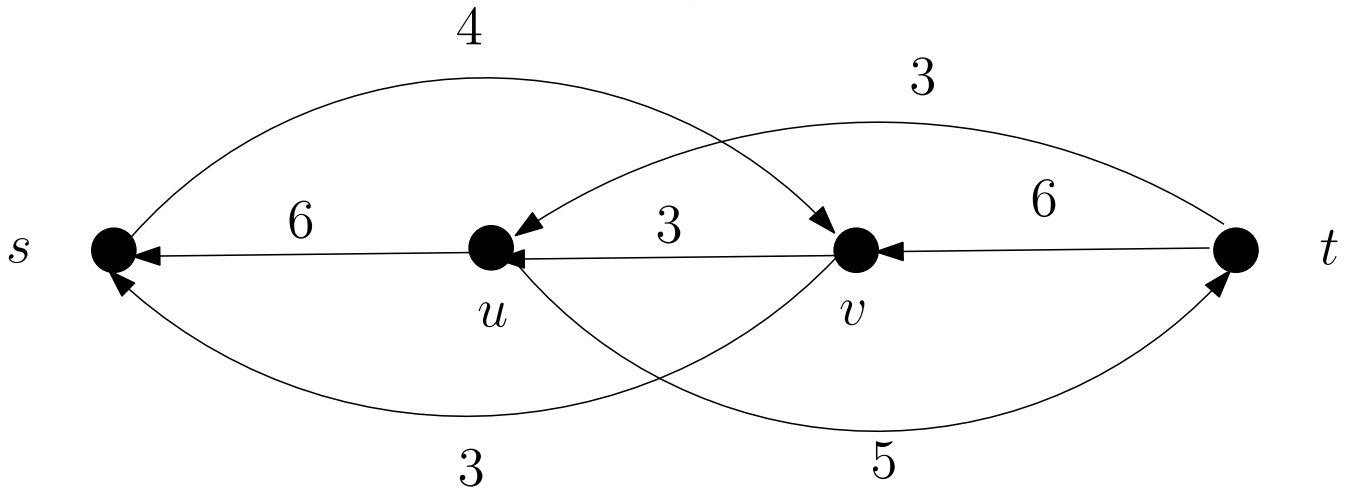


Auxiliary graph

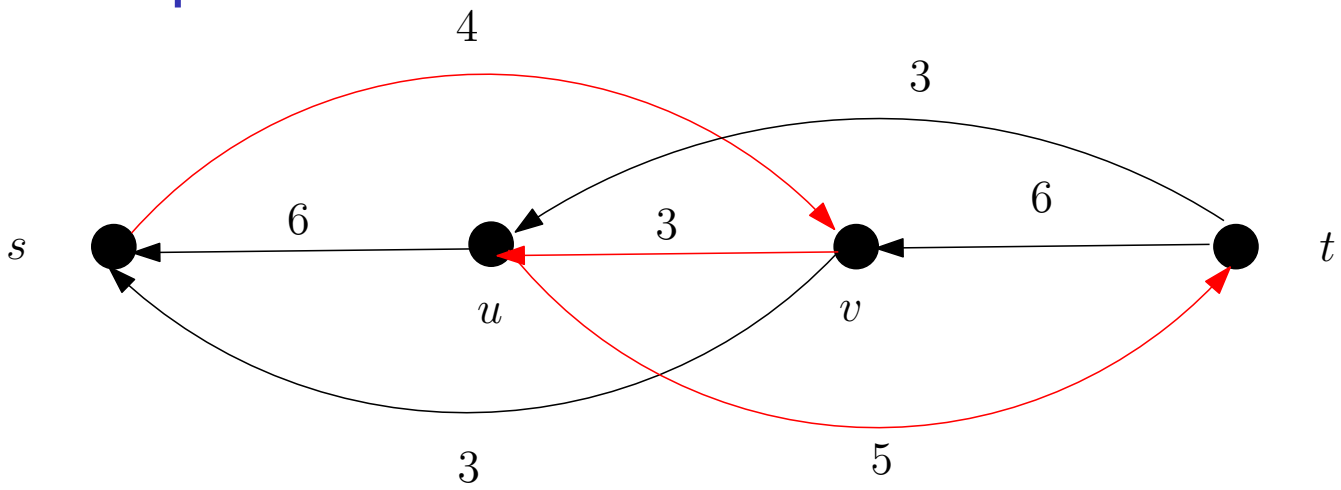
Example



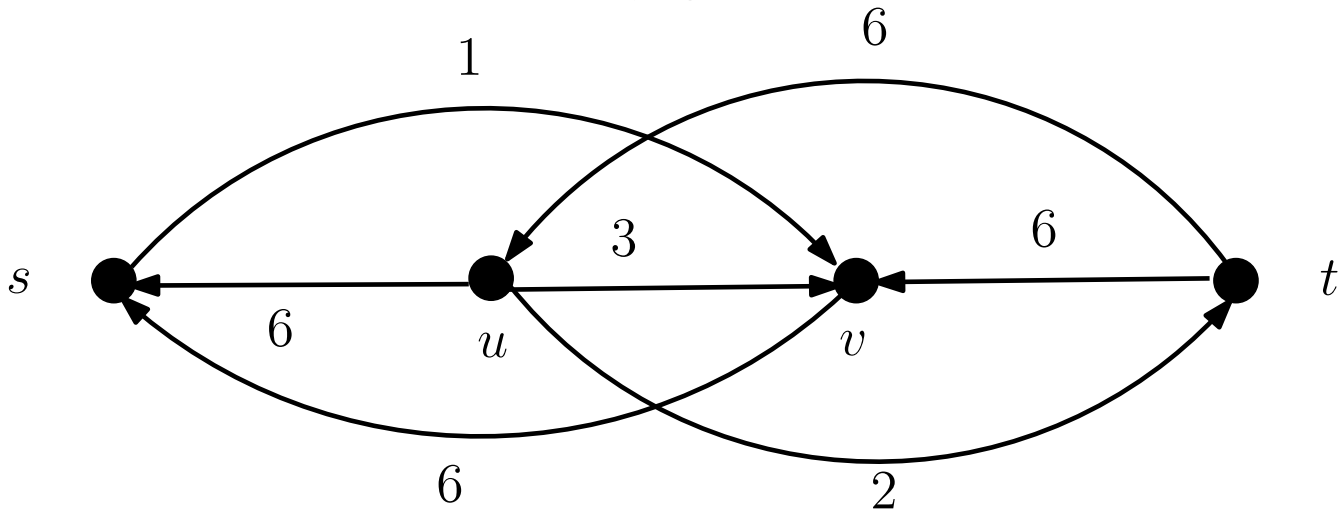
There is an augmenting path s–u–t with capacity $\min\{3, 8\} = 3$.
Augment the flow by 3 along that path.
The current flow from s to t is 6+3=9.
We obtain a new auxiliary graph:



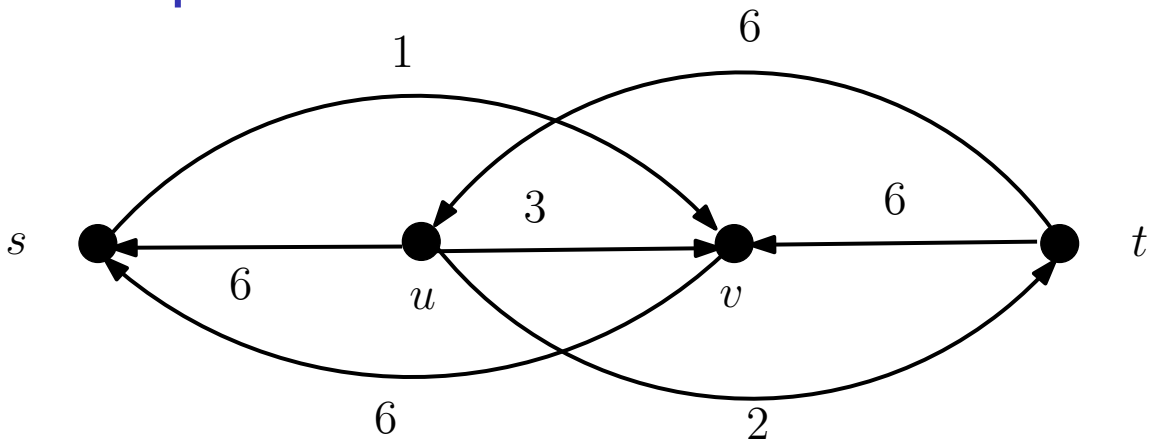
Example



There is an augmenting path s–v–u–t with capacity $\min\{4, 3, 5\} = 3$.
Augment the flow by 3 along that path.
The current flow from s to t is $9+3 = 12$.
We obtain a new auxiliary graph:

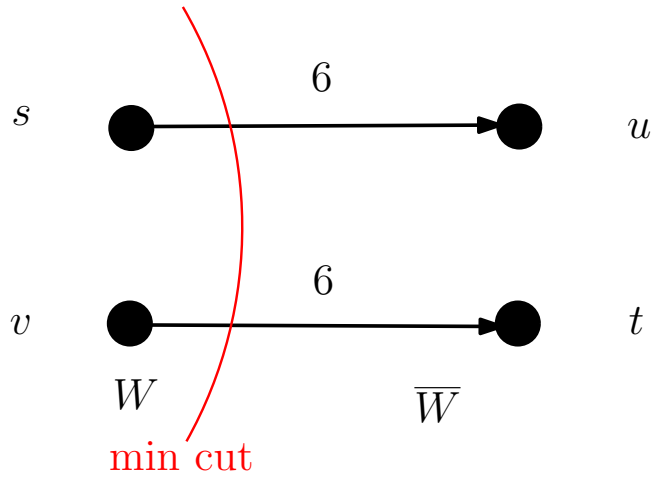


Example

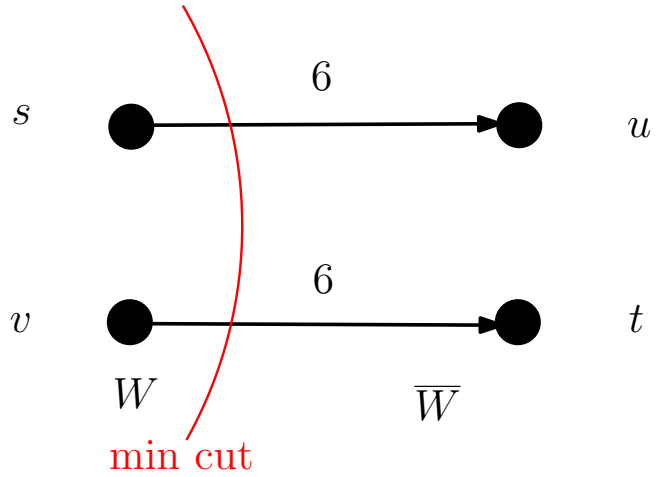


We can only reach vertex v from s. So $W = \{s, v\}$ and $\bar{W} = \{u, t\}$. Look in the **original graph** and calculate the capacity of all arcs that go in the forward direction from W to $\bar{W}$.

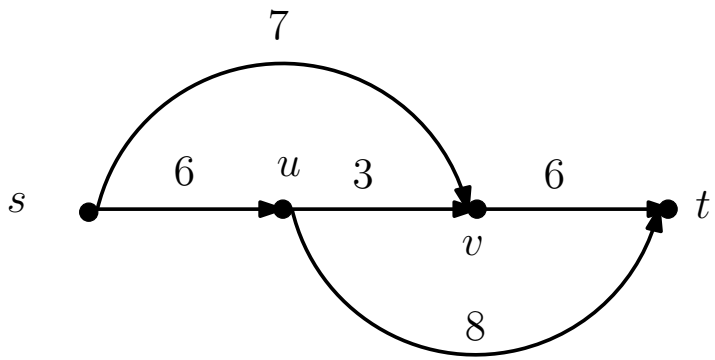
To make it easier to see, I have redrawn the graph such that W, $\bar{W}$ are easier to see.

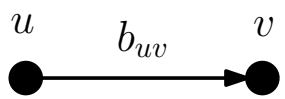


Optimal solution

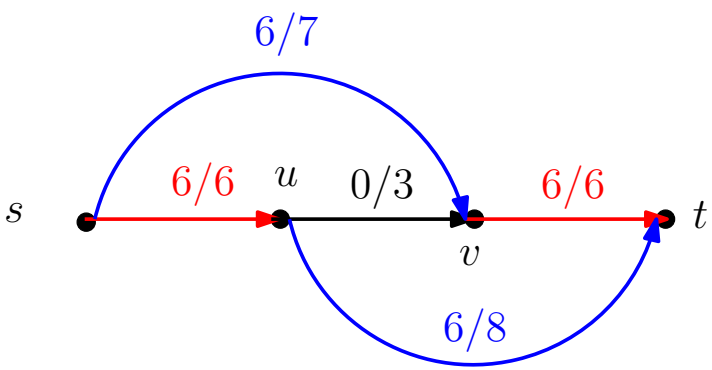


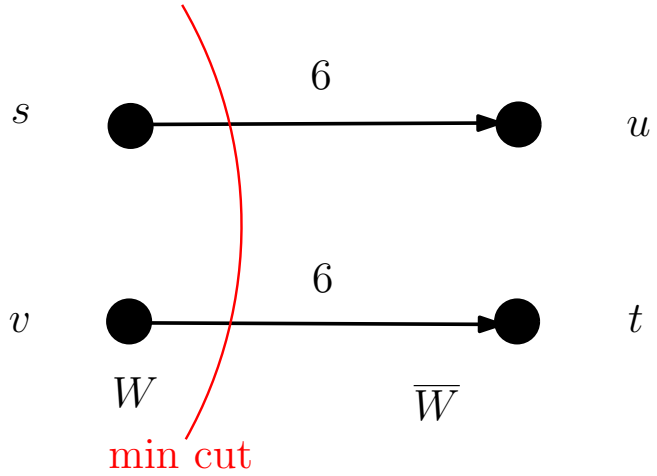
Since the value of the s-t cut is 12, and we have an s-t flow of value 12, the flow has to be maximum, and the s-t cut minimum!





Optimal solution



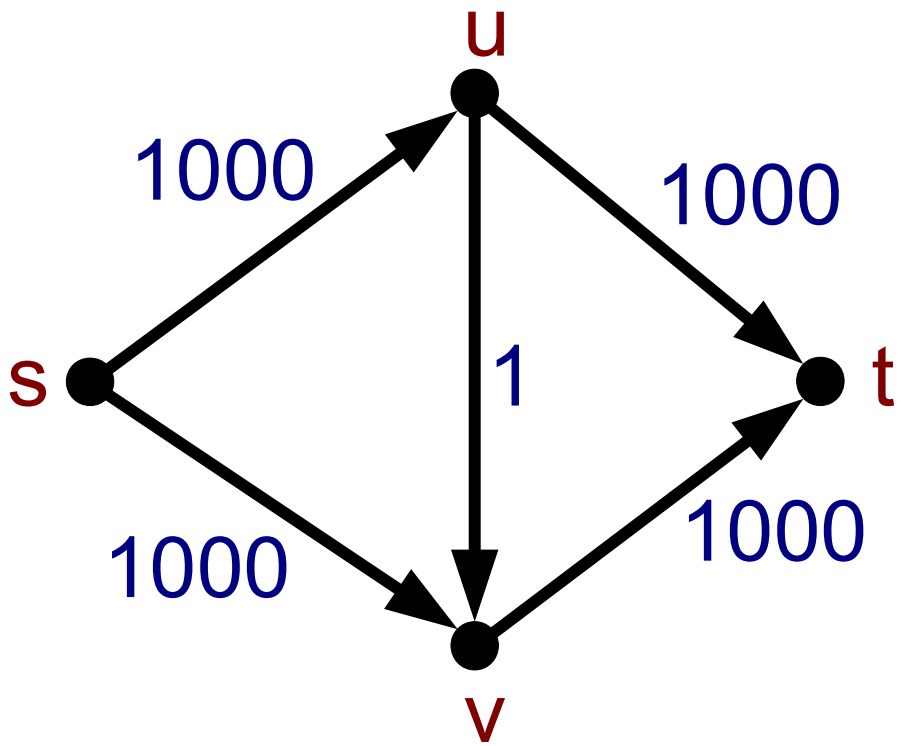


We can now verify the complementary slackness conditions:

- $f_{uv} = b_{uv}$ for all $(u, v) \in A$ with $u \in W$ and $v \in \overline{W}$ ("forward arcs");
- $f_{uv} = 0$ for all $(u, v) \in A$ with $u \in \overline{W}$ and $v \in W$ ("backward arcs"). Here (u, v) is a backward arc, which has zero flow.

Example

The number of iterations of the **Ford-Fulkerson** algorithm can be exponential in the input size.

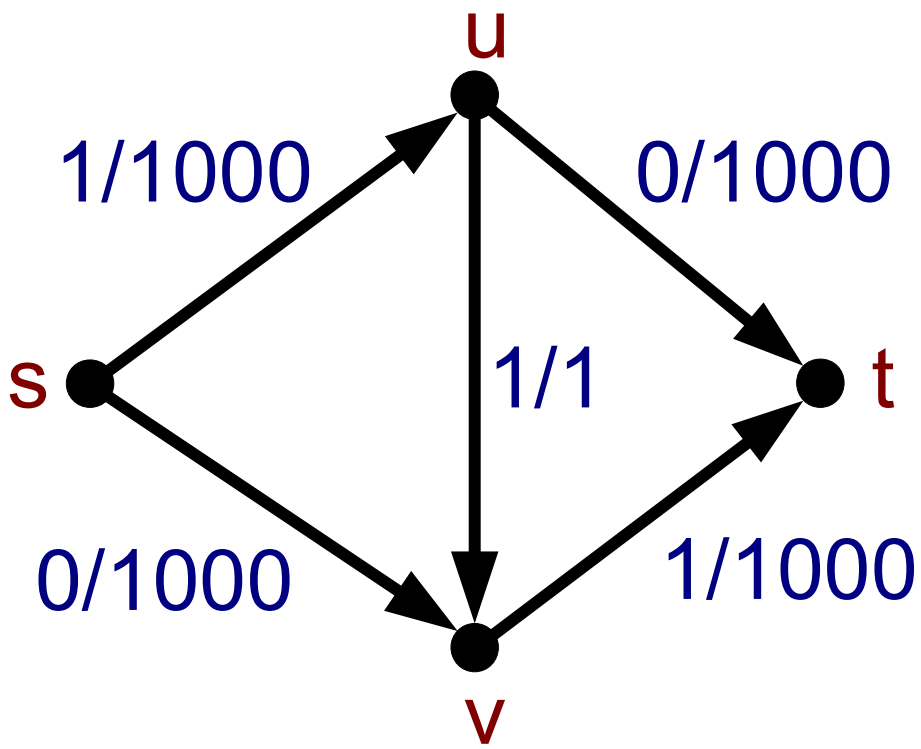


First iteration: augment the flow over the path (s, u, v, t).

Example

The number of iterations of the **Ford-Fulkerson** algorithm can be exponential in the input size.

This gives a flow with value 1:

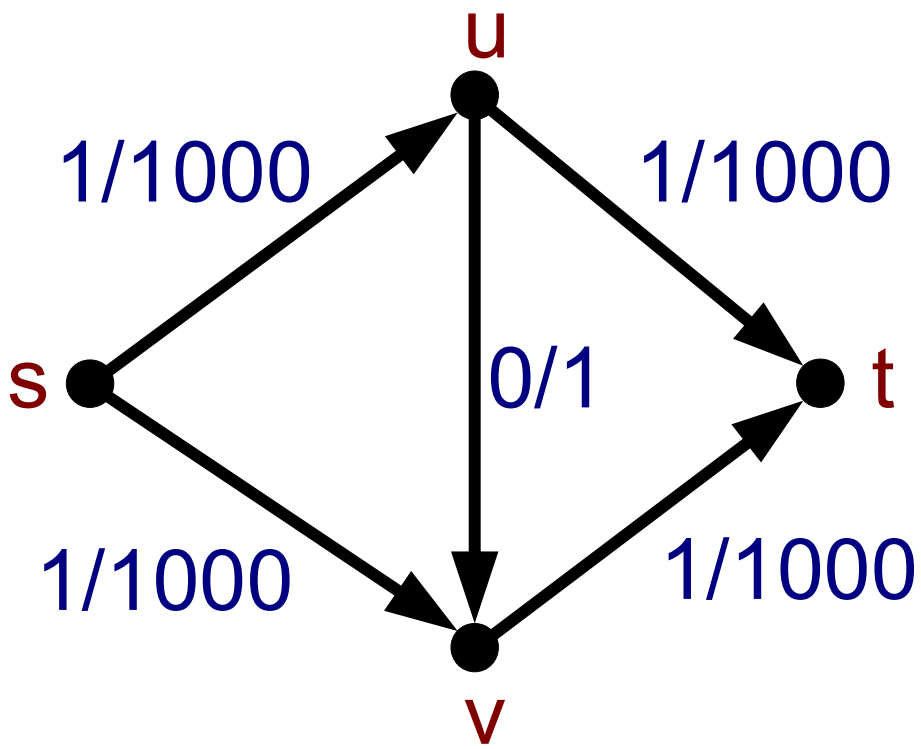


Second iteration: augment the flow over the path (s, v, u, t).

Example

The number of iterations of the **Ford-Fulkerson** algorithm can be exponential in the input size.

This gives a flow with value 2:

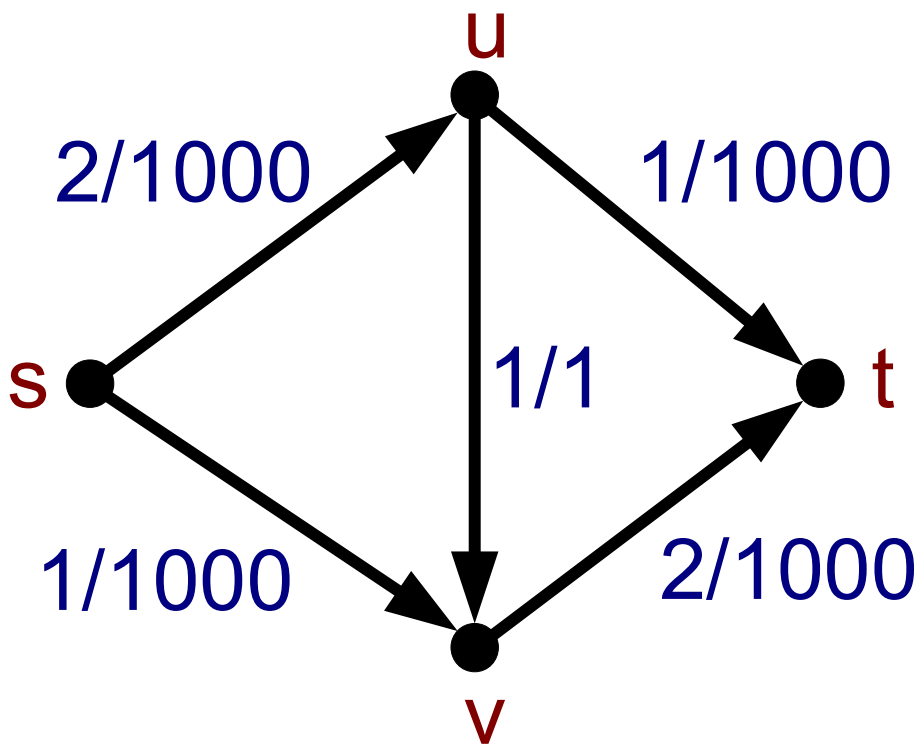


Third iteration: augment the flow over the path (s, u, v, t).

Example

The number of iterations of the **Ford-Fulkerson** algorithm can be exponential in the input size.

This gives a flow with value 3:

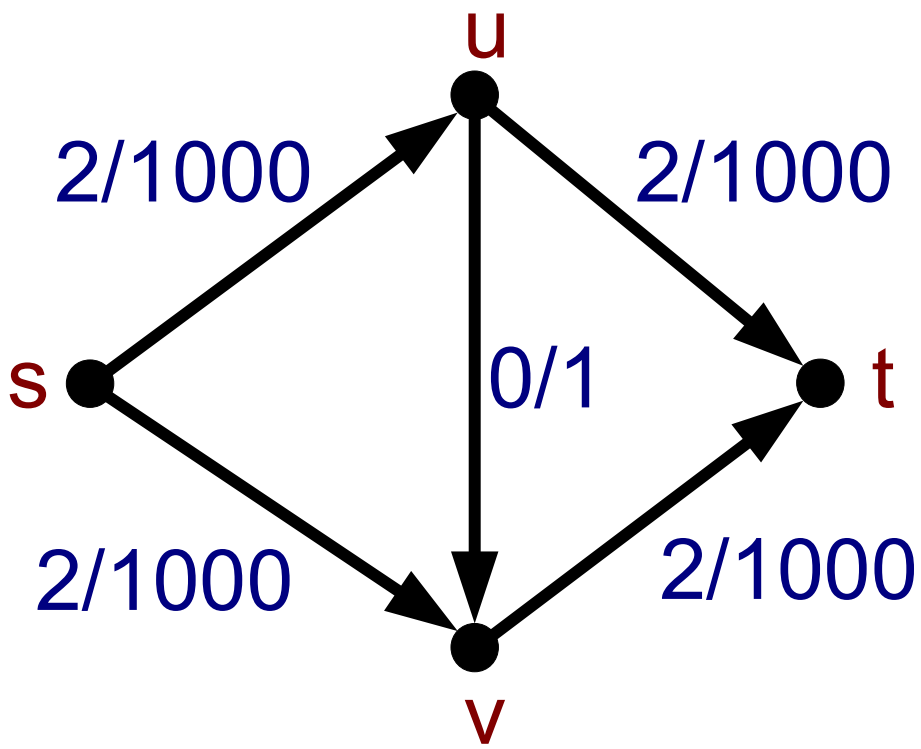


Fourth iteration: augment the flow over the path (s, v, u, t).

Example

The number of iterations of the **Ford-Fulkerson** algorithm can be exponential in the input size.

This gives a flow with value 4:

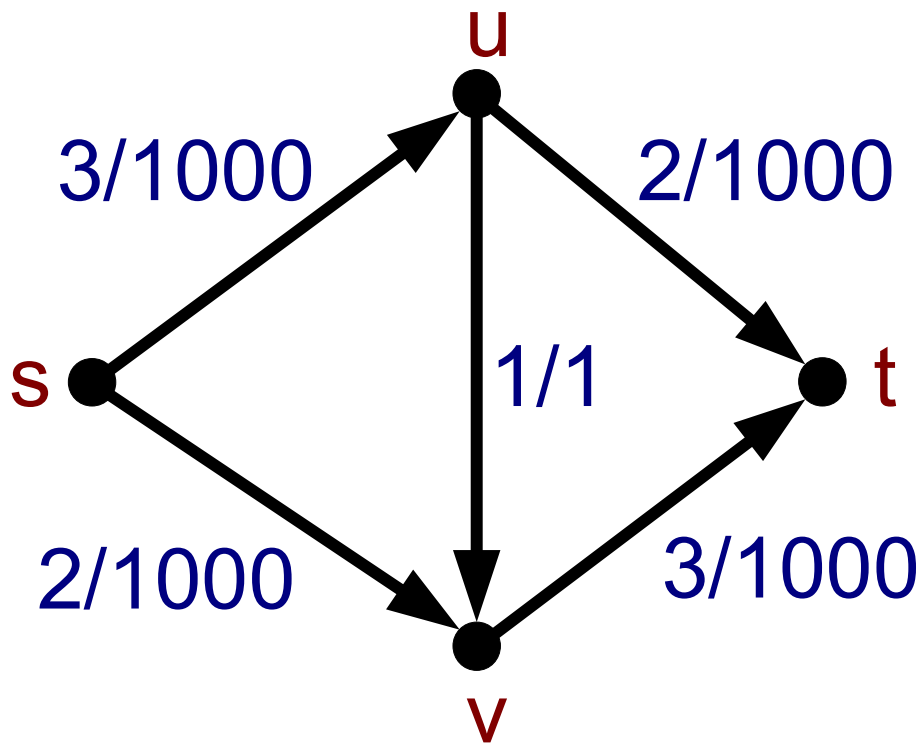


Fifth iteration: augment the flow over the path (s, u, v, t).

Example

The number of iterations of the **Ford-Fulkerson** algorithm can be exponential in the input size.

This gives a flow with value 5:

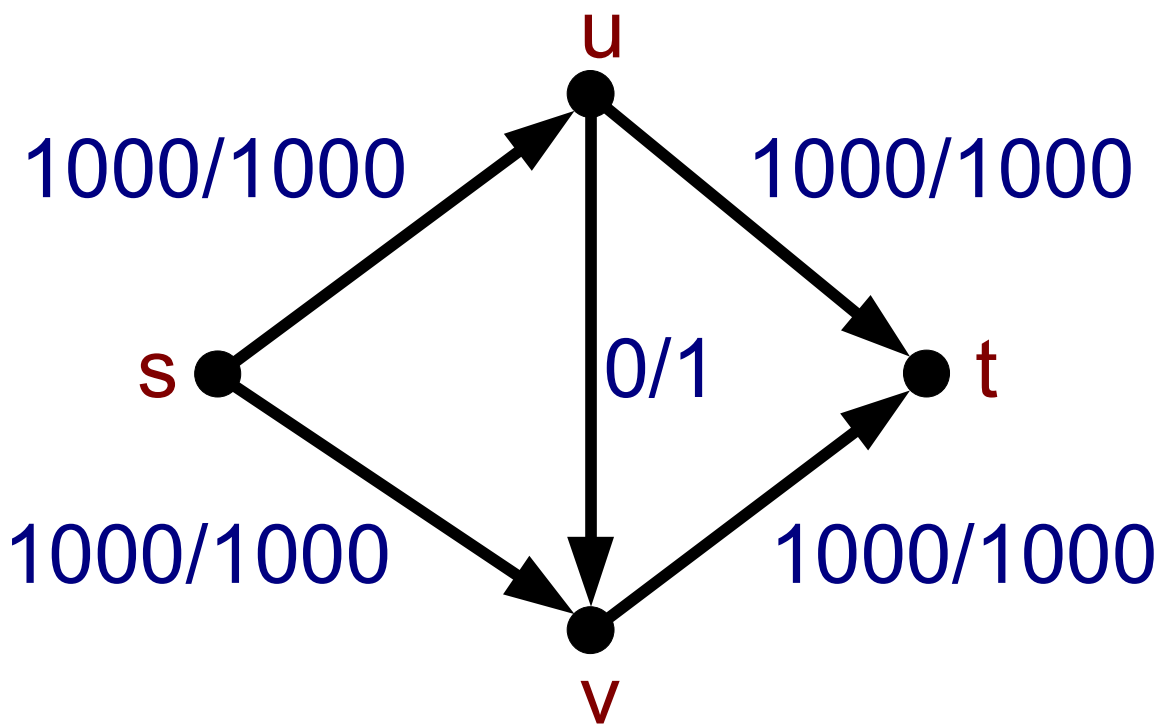


Continue like this. . .

Example

The number of iterations of the **Ford-Fulkerson** algorithm can be exponential in the input size.

After **2000 iterations** we finally find an optimal flow with value 2000:



If we replace 1000 by 2^N
then the input size is $O(\log(2^N)) = O(N)$
and we need 2^{N+1} iterations!

So the **Ford-Fulkerson** algorithm is **not polynomial**.

Question

How can we improve the algorithm?

Theorem (Dinits and Edmonds-Karp)

The Ford-Fulkerson algorithm has **polynomial running time** if we choose a **shortest** flow-augmenting path in each iteration.

Here, the length on each arc is 1, so we count "shortest" in terms of the **number of arcs** in the path.

To Do: Exercises 10.1, 9.3, 9.4

In 10.1, verify that $\pi = -\rho$ is a feasible solution of the dual, and that the value $\pi_s - \pi_t$ is equal to the length of the shortest path. Determine the shortest path using Complementary slackness.

In 9.3, also write down the dual and Complementary slackness conditions, and verify that the max flow is equal to the min cut.

Approximation Algorithms

Methods for solving LP's: **Simplex**, Ellipsoid methods, Interior point methods
Methods for solving ILP's: Branch and Bound (Cut). Can take a long time to terminate in practice, depending on breadth / depth of tree. But ILP's are NP-hard.

What can we do?

- Heuristics (TSP example: Just pick a random sequence of vertices)
- Approximation algorithms: Polynomial-time algorithms with guarantee of being within some constant times optimal solution.

What do we mean by guarantee?

Approximation Algorithms

Definition

A ρ-approximation algorithm for an optimization problem P is a polynomial-time algorithm that produces, for each instance I of P, a solution such that:

$$\text{ALG}(I) \leq \rho \cdot \text{OPT}(I) \quad \text{if } P \text{ is a minimization problem}$$

$$\text{ALG}(I) \geq \rho \cdot \text{OPT}(I) \quad \text{if } P \text{ is a maximization problem}$$

- $\text{ALG}(I)$ is the value of the solution for instance I given by the algorithm.
- $\text{OPT}(I)$ is the value of an optimal solution for instance I.
- ρ is the approximation ratio. This does not need to be a constant.

For minimization problems, we want $\rho \geq 1$ to be as small as possible.
For maximization problems, we want $\rho \leq 1$ to be as large as possible.

Travelling Salesman Problem

Problem

Traveling Salesman Problem (TSP)
Given: a graph $G = (V, E)$ and a length $\ell(e)$ for each edge $e \in E$
Find: a shortest cycle that visits each vertex of the graph exactly once.

Theorem

There does not exist a ρ-approximation algorithm for TSP with ρ a constant, unless $P = NP$.

Why?

Approximation algorithm for a particular Travelling Salesman Problem

Problem

Metric Traveling Salesman Problem (ΔTSP)

Given: a graph $G = (V, E)$ and a length $\ell(e)$ for each edge $e \in E$, such that the triangle inequality holds.

Find: a shortest cycle that visits each vertex of the graph exactly once.

Definition

The lengths satisfy the triangle inequality if

$$\ell(uv) \leq \ell(uw) + \ell(wv), \quad \forall u, v, w \in V.$$

Theorem

ΔTSP is NP-complete.

Some useful results for the approximation algorithm

Given a connected graph with weighted edges, a spanning tree is any connected subgraph that is a tree. A minimum spanning tree (MST) is a spanning tree with minimum total weight.

Kruskal's algorithm

Input: $G = (V, E)$ with edge weights $\ell : E \to \mathbb{R}$.

1. $F := \emptyset$
2. **while** $|F| < |V| - 1$,
   Choose an edge $e \in E \setminus F$ with minimum length such that $(V, F \cup \{e\})$ has no cycles. $F := F \cup \{e\}$.

Bridges of Königsberg – Eulerian cycles

Question

Find a walk through the city that crosses each of the bridges exactly once, ending at the starting point.

Definition

An Eulerian cycle is a cycle that uses each edge exactly once.

Theorem

A connected graph admits an Eulerian cycle if and only if every vertex is of even degree.

2-Approximation algorithm for ΔTSP

Problem

Metric Traveling Salesman Problem (ΔTSP)
Given: a graph $G = (V, E)$ and a length $\ell(e)$ for each edge $e \in E$, such that the triangle inequality holds.
Find: a shortest cycle that visits each vertex of the graph exactly once.

1. Find a minimum spanning tree T of G.
2. Duplicate every edge in T. Find an Euler tour, i.e., a cycle that uses every edge.
3. Retrace the steps of the cycle, skipping already-visited vertices. This gives a cycle that visits each vertex exactly once.

Theorem

The above algorithm is a 2-approximation algorithm for ΔTSP.