# Maximum flow problem

Pramesh Kumar

IIT Delhi

March 28, 2024

# Maximum flow problem

Given a capacitated directed network $G(N, A)$, find the maximum value of flow that can be sent between two special nodes, namely source $s \in N$ and sink $t \in N$, without exceeding the capacity of any link in the network.

# Assumptions

- Network is directed.
- All capacities are non-negative integers.[1]
- One $s$ source and one sink $t$.[2]
- The network does not contain a path from $s$ to $t$ with only infinite capacity links.[3]
- If $(i, j) \in A$ then $(j, i) \notin A$.[4]

---

[1] You can convert the rational capacities to integers by multiplying them with significantly large integer.

[2] If there are multiple sources $s_1, ..., s_p$ and multiple sinks $t_1, ..., t_q$, then create a super source $s$ and a super sink $t$ and create links $(s, s_i), \forall i = 1, ..., p$ and $(t_j, t), \forall j = 1, ..., q$ with infinite capacities.

[3] Otherwise the problem will be unbounded, i.e., you can send $\infty$ flow from $s$ to $t$.

[4] If there are links from $(i, j)$ and $(j, i)$, then create another node $j'$ and remove link $(i, j)$ and create links $(i, j')$ and $(j', j)$, both with capacities $u_{ij}$.

# Flow

Definition (Flow). A flow in $G$ is a real-valued function $x : A \mapsto \mathbb{R}$ that satisfies two properties:

1. Capacity constraints

$$0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in A \tag{1}$$

2. Flow conservation

$$\sum_{j \in FS(i)} x_{ij} - \sum_{j \in BS(i)} x_{ij} = 0, \quad \forall i \in N \backslash \{s,t\} \tag{2}$$

$$\sum_{j \in FS(i)} x_{sj} = v \tag{3}$$

$$\sum_{j \in BS(i)} x_{jt} = v \tag{4}$$

For the max flow problem, we need to maximize $v$.

# Residual network corresponding to a flow

Residual capacity $r_{ij}(\mathbf{x}) = \begin{cases} u_{ij} - x_{ij} & \text{if } (i,j) \in A \\ x_{ji} & \text{if } (j,i) \in A \\ 0 & \text{otherwise} \end{cases}$

- Either one of the first two cases will occur (assumption 5).
- Residual network consists of links whose capacities represent how the flow can change on links.
- We only have links with positive residual capacities.
- For $(i,j) \in A$, even if do not have $(j,i) \in A$, we might still have $(j,i)$ in the residual network. The purpose of creating this link to decrease the flow on $(i,j) \in A$ so as to increase the overall flow from $s$ to $t$.

Definition (Augmenting path). A path with non-zero residual capacity is an augmenting path.

# Example

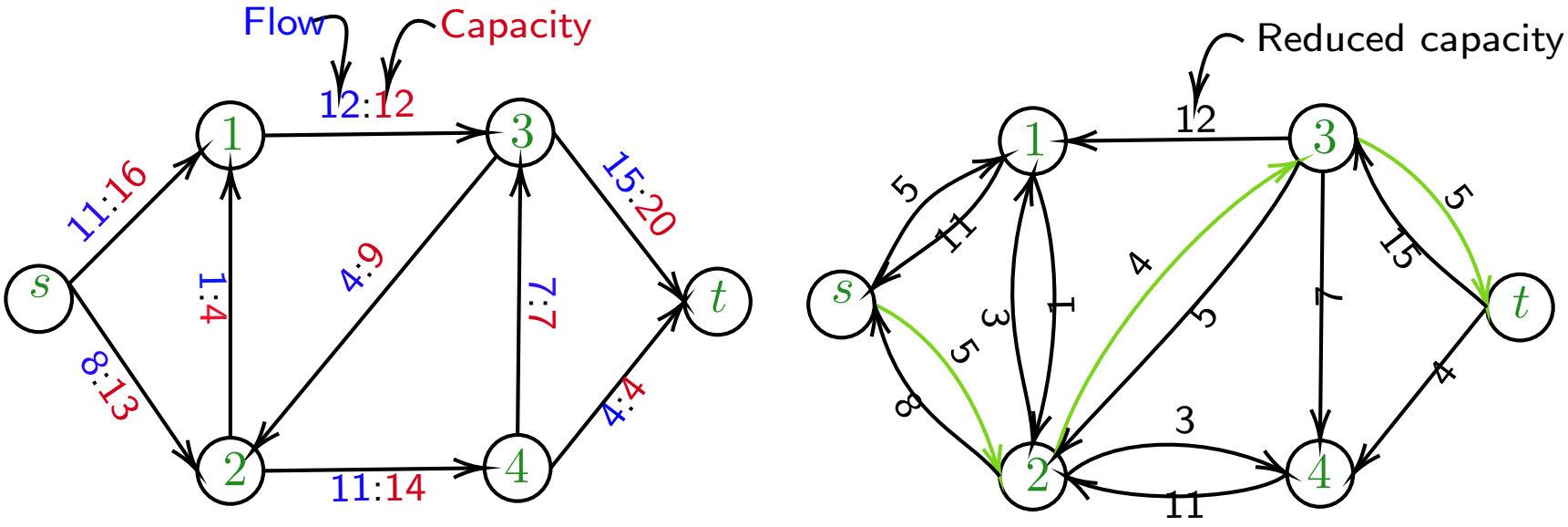



Figure: The first figure show the flow and capacities in the network. The second figure shows the residual network for the given flow. It further shows an augmenting path in green color.

# Types of algorithms

1. **Augmenting path algorithms**:
   - finds augmenting paths to push flow from $s$ to $t$ until there are none.
   - maintains flow balance at all nodes except $s$ and $t$.
2. **Preflow-push algorithms**:
   - initially, flood the network with flow.
   - may result in the excess at some nodes.
   - pushes flow forward towards the sink or backwards towards the source.

# Ford-Fulkerson method

1: **procedure** AUGMENTINGPATH($G, c, u, s, t, x$)
2: Initialize flow $\mathbf{x} = 0$
3: **while** $\exists$ an augmenting path $P$ in the residual network $G(\mathbf{x})$ **do**
4: Augment the flow along $P$.
5: **end while**
6: **return** $\mathbf{x}$
7: **end procedure**

- The residual capacity of an augmenting path $P$ is given by $\delta = \min_{(i,j)\in P} r_{ij}$.
- Let $\mathbf{x}$ be the flow in $G$ and $\mathbf{x}'$ be the flow on an $s-t$ path $P$ in the corresponding residual network $G(\mathbf{x})$. We define augmentation of the flow $\mathbf{x}$ along path $P$ using flow $\mathbf{x}'$ as:

$$\begin{cases} x_{ij} + x'_{ij} - x'_{ji}, & \text{if } (i,j) \in A \\ 0 & \text{otherwise} \end{cases}$$

# Example: Augmenting the flow

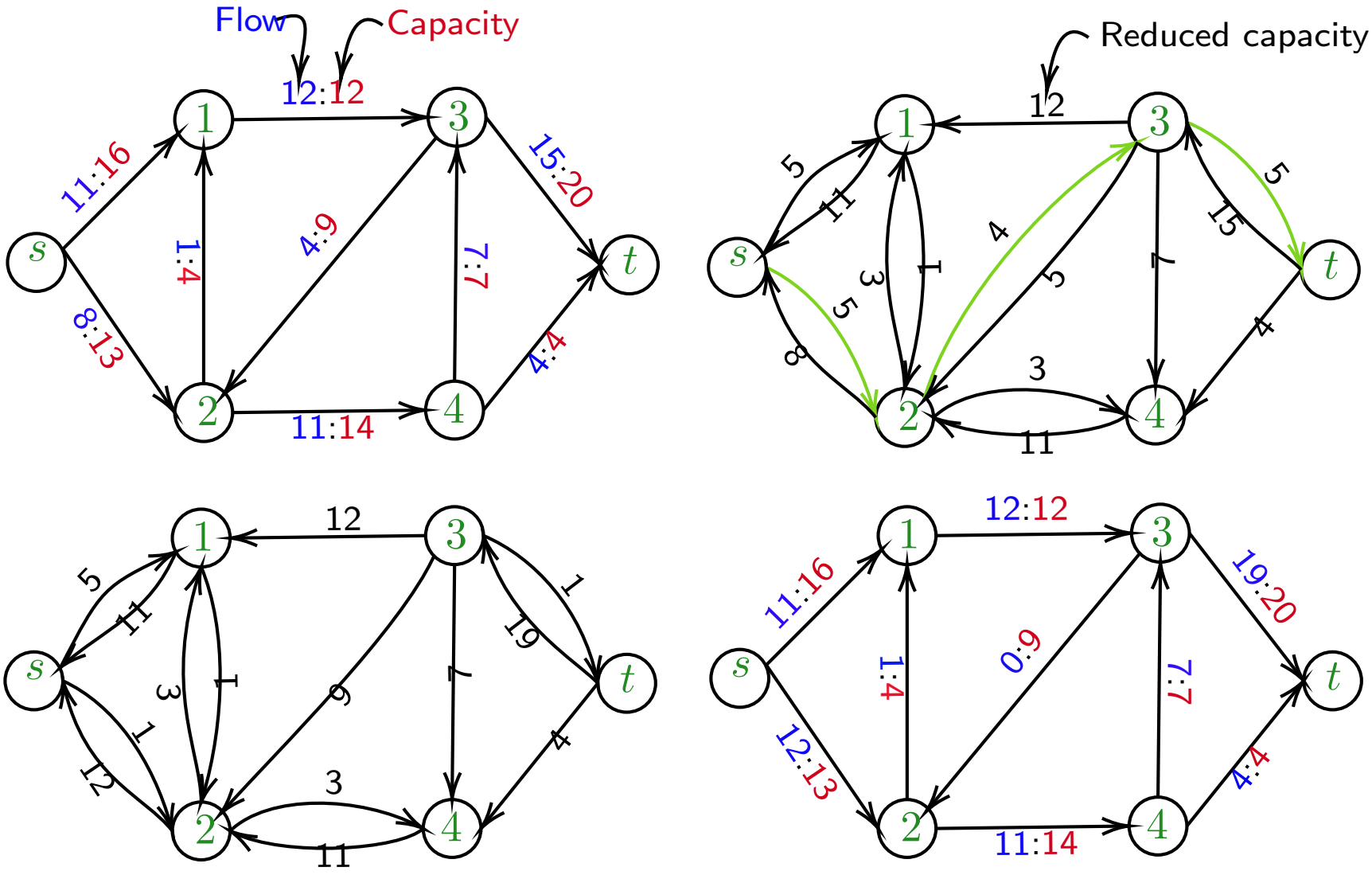



Figure: Augmenting the flow along path $s - 2 - 3 - t$

## Lemma

*After flow augmentation, the new flows will be feasible.*

## Proof.

For any link $(i, j) \in A$, the updated flow is $x_{ij} + x'_{ij} - x'_{ji}$.

1. It is non-negative.

$$\begin{aligned} x_{ij} + x'_{ij} - x'_{ji} &\geq x_{ij} + x'_{ij} - x_{ij} && (5)\\ &= x'_{ij} && (6)\\ &\geq 0 && (7) \end{aligned}$$

2. It is less than the capacity $u_{ij}$.

$$\begin{aligned} x_{ij} + x'_{ij} - x'_{ji} &\leq x_{ij} + x'_{ij} && (8)\\ &\leq x_{ij} + r_{ij} && (9)\\ &= x_{ij} + u_{ij} - x_{ij} && (10)\\ &= u_{ij} && (11) \end{aligned}$$

□

## Proof (contd.)

3. The updated flows satisfy the flow conservation. For any node $i \in N \backslash \{s, t\}$

$$\sum_{j \in FS(i)} \left( x_{ij} + x'_{ij} - x'_{ji} \right) - \sum_{j \in BS(i)} \left( x_{ji} + x'_{ji} - x'_{ij} \right) \tag{12}$$

$$= \left( \sum_{j \in FS(i)} x_{ij} - \sum_{j \in BS(i)} x_{ji} \right) + \left( \sum_{j \in FS(i)} x'_{ij} - \sum_{j \in BS(i)} x'_{ji} \right) \tag{13}$$

$$+ \left( \sum_{j \in BS(i)} x'_{ij} - \sum_{j \in FS(i)} x'_{ji} \right) = 0$$

Similarly for nodes $s, t$. □

## Corollary

*The augmented flow will be strictly greater than the previous flows.*

# Cut

Definition ($s-t$ cut). An $s-t$ cut is a partition of nodes into two subsets $S$ and $T = N \backslash S$ such that $s \in S$ and $t \in T$.

Definition (Flow across cut). The flow across an $s-t$ cut $(S,T)$ is given as:

$$x(S,T) = \sum_{i \in S} \sum_{j \in T} x_{ij} - \sum_{i \in S} \sum_{j \in T} x_{ji} \tag{14}$$

Definition (Capacity of cut). The capacity of an $s-t$ cut $(S,T)$ is given as:

$$u(S,T) = \sum_{i \in S} \sum_{j \in T} u_{ij} \tag{15}$$

# Example

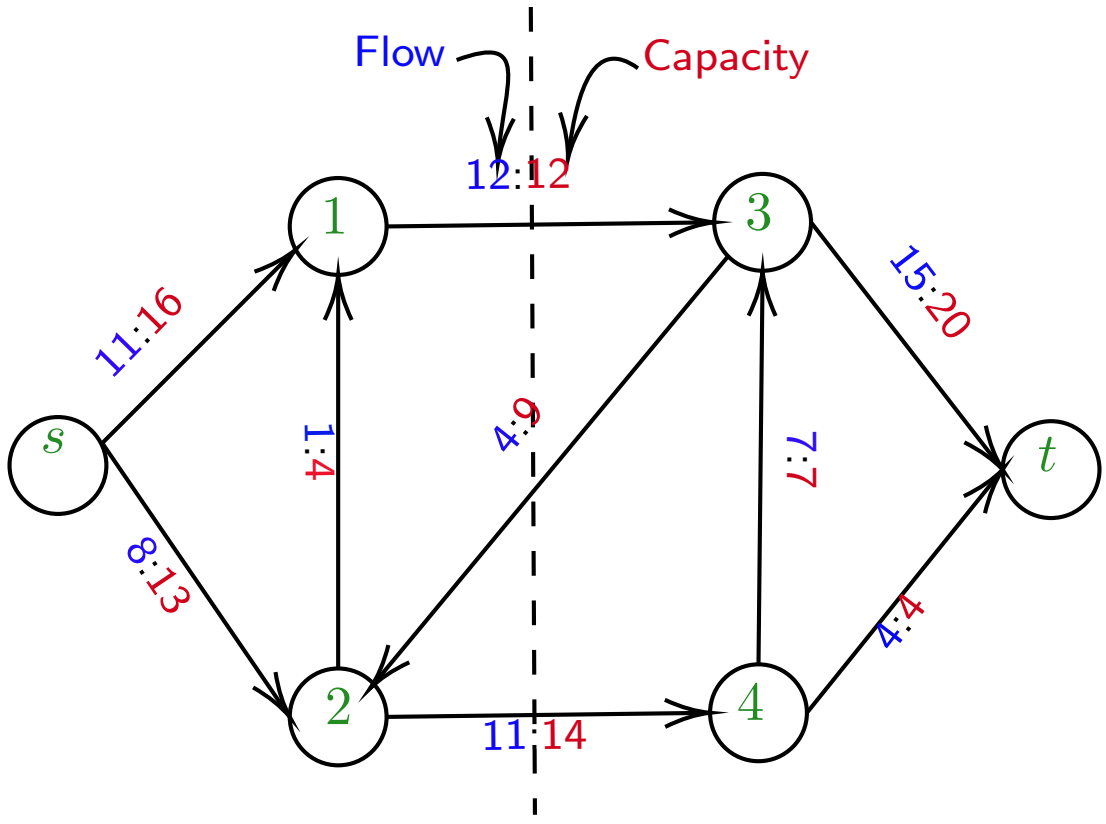



Figure: An $s-t$ cut

$S = \{s, 1, 2\}$ and $T = \{3, 4, t\}$.
$x(S,T) = x_{13} + x_{24} - x_{32} = 12 + 11 - 4 = 19$ and
$u(S,T) = u_{13} + u_{24} = 12 + 14 = 26$.

# Minimum cut problem

Among all the $s-t$ cuts in the network, find one with minimum capacity.

# LP formulation

Max flow problem

$$\max_{\mathbf{x},v} \quad v$$

$$\text{s.t.} \sum_{j \in FS(i)} x_{ij} - \sum_{j \in BS(i)} x_{ji} = \begin{cases} v & \text{if } i = s \\ -v & \text{if } i = t \\ 0 & \forall i \in N \backslash \{s\} \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij}, \forall (i,j) \in A$$

Min cut problem

$$\min_{\lambda,\mu} \sum_{(i,j) \in A} \lambda_{ij} u_{ij}$$

$$\text{s.t. } \mu_i - \mu_j + \lambda_{ij} \geq 0, \forall (i,j) \in A$$

$$-\mu_s + \mu_t = 1$$

$$\lambda_{ij} \geq 0, \forall (i,j) \in A$$

Any cut $(S,T)$ can be associated to the dual problem as $\mu_i = 0$, if $i \in S$ and $\mu_i = 1$, if $i \in T$. $\lambda_{ij} = 1$, if $i \in S, j \in T$, 0, otherwise

## Lemma (Weak duality)

$$x(S,T) \leq u(S,T)$$

## Proof.

$$x(S,T) = \sum_{i \in S} \sum_{j \in T} x_{ij} - \sum_{i \in S} \sum_{j \in T} x_{ji} \tag{16}$$

$$\leq \sum_{i \in S} \sum_{j \in T} x_{ij} \tag{17}$$

$$\leq \sum_{i \in S} \sum_{j \in T} u_{ij} = u(S,T) \tag{18}$$

We did not use result from LP duality!

# Max-flow min-cut theorem

## Theorem (Strong duality)

*If $\mathbf{x}$ be a flow in the network $G(N, A)$ with source $s$ and sink $t$, then the following conditions are equivalent.*

1. $v = \sum_{j \in FS(s)} x_{sj}$ *is a maximum flow in* $G$.
2. *The residual network* $G(\mathbf{x})$ *contains no augmenting path.*
3. $v = u(S, T)$ *for some cut* $(S, T)$ *of* $G$.

## Proof.

$1 \implies 2$ Suppose that $v$ is the maximum flow in the network bu there still exists an augmenting path $P$ in $G(\mathbf{x})$. Then, we can increase the flow along $P$ by its residual capacity, contradicting that $v$ is maximum flow.
$2 \implies 3$ Suppose that $G(\mathbf{x})$ contains no augmenting path, i.e., no path from $s$ to $t$ in $G(\mathbf{x})$. Define a set $S = \{i \in N : \exists \text{ a path from } s \text{ to } i\}$ and $T = N \backslash S$. The partition $(S, T)$ defines a cut. $s \in S$ and $t \in T$ trivially, otherwise there is an augmenting path from $s$ to $t$. Now, consider nodes $i \in S$ and $j \in T$. If $(i, j) \in A$, we must have $x_{ij} = u_{ij}$ since otherwise $j$ is reachable from $i$ and $j$ should be in $S$, a contradiction. Further, if $(j, i) \in A$, we must have $x_{ji} = 0$ since otherwise $r_{ij} = x_{ji}$ would be positive and we would have $(i, j)$ in the residual graph making $j$ reachable from $i$ and hence put $j$ in $S$, a contradiction. If $(i, j) \notin A$ and $(j, i) \notin A$, then $x_{ij} = x_{ji} = 0$.

## Proof (contd.)

We must have

$$v = x(S,T) = \sum_{i\in S}\sum_{j\in T} x_{ij} - \sum_{i\in S}\sum_{j\in T} x_{ji} \tag{19}$$

$$= \sum_{i\in S}\sum_{j\in T} x_{ij} \tag{20}$$

$$= \sum_{i\in S}\sum_{j\in T} u_{ij} = u(S,T) \tag{21}$$

$3 \implies 1$ From Corollary, we have $v \leq u(S,T), \forall s-t$ cuts. From previous condition,$v = u(S,T)$ implying $v$ is the maximum flow. □

## Theorem

*If all link capacities are integer, then the max flow will be integer.*

# Ford-Fulkerson labeling algorithm

**procedure** LABELING($G, c, u, s, t, x$)
    label $t$
    **while** $t$ is labeled **do**
        unlabel all the nodes
        $pred(i) = 0, \forall i \in N$
        label $s$ and $SE := \{s\}$
        **while** $SE \neq \phi$ **and** $t$ is unlabeled **do**
            Remove a node $i$ from $SE$
            **for** $j \in FS(i)$ **do**
                **if** $j$ is unlabeled **then**
                    label node $j$; $pred(j) = i$ and $SE = SE \cup \{j\}$
                **end if**
            **end for**
        **end while**
        **if** $t$ is labeled **then**
            AUGMENT($G, c, u, s, t, x, pred$)
        **end if**
    **end while**
**end procedure**

**procedure** AUGMENT($G, c, u, s, t, x, pred$)
    find a path $P$ from $s$ to $t$ using $pred$
    $\delta := \min_{(i,j) \in P}\{r_{ij}\}$
    augment $\delta$ units of flow along $P$ and update the residual capacities
**end procedure**

## Proposition

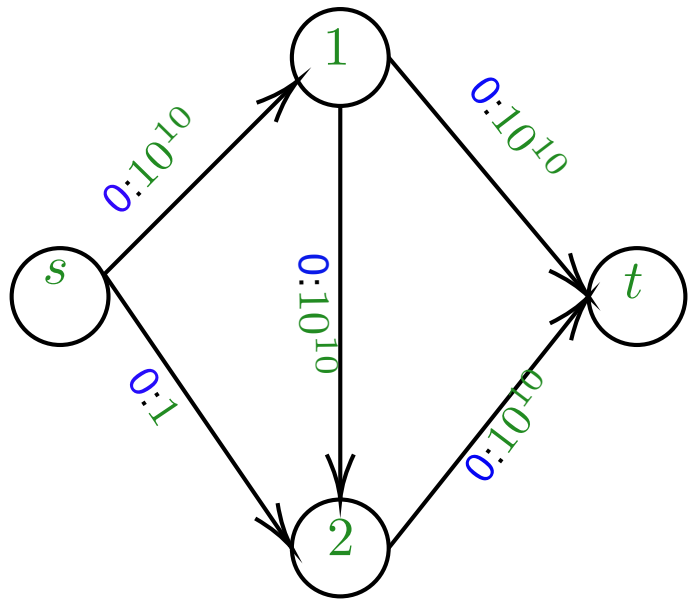
*The labeling algorithm solves the max flow problem is $O(vm)$ time, where $v$ is the value of max flow.*

## Proof.

Each augmentation takes $O(m)$ time. In the worst-case, the number of augmentations are $v$ (in case when each augmentation increases the flow by 1). Therefore, the worst-case complexity is $O(vm)$. If $U = \max_{(i,j)\in A} u_{ij}$, then capacity of any cut is at most $nU$. Then, the overall complexity is $O(mnU)$. □

# Issues with above algorithm

- If $U = 2^n$, then algorithm will run exponential number of iterations.
- If capacities are irrational, the algorithm might not terminate.
- We don't use the labeling information from one iteration to another.



# Other algorithms

- Capacity scaling algorithms
- Edmonds-Karp shortest augmenting path algorithm
- Pre-flow push algorithms

# Suggested Reading

- AMO Chapter 6 and 7

Thank you!