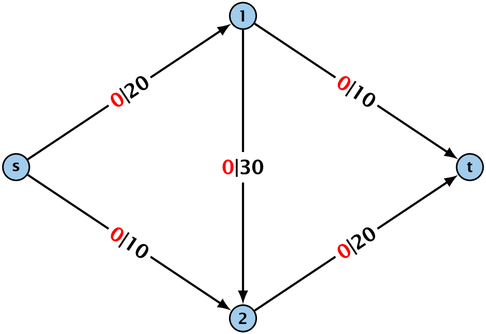
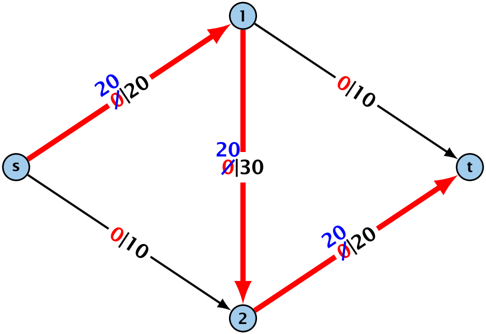
**Greedy-algorithm:**

- start with $f(e) = 0$ everywhere
- find an $s$-$t$ path with $f(e) < c(e)$ on every edge
- augment flow along the path
- repeat as long as possible



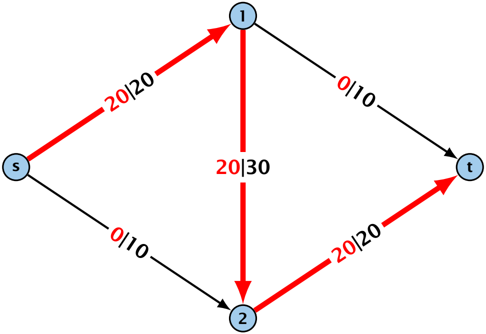
**Greedy-algorithm:**

- start with $f(e) = 0$ everywhere
- find an $s$-$t$ path with $f(e) < c(e)$ on every edge
- augment flow along the path
- repeat as long as possible



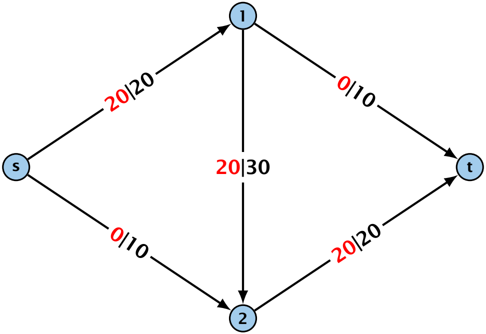
**Greedy-algorithm:**

- start with $f(e) = 0$ everywhere
- find an $s$-$t$ path with $f(e) < c(e)$ on every edge
- augment flow along the path
- repeat as long as possible



**Greedy-algorithm:**

- start with $f(e) = 0$ everywhere
- find an $s$-$t$ path with $f(e) < c(e)$ on every edge
- augment flow along the path
- repeat as long as possible



# The Residual Graph

From the graph $G = (V, E, c)$ and the current flow $f$ we construct an auxiliary graph $G_f = (V, E_f, c_f)$ (the residual graph):

# The Residual Graph

From the graph $G = (V, E, c)$ and the current flow $f$ we construct an auxiliary graph $G_f = (V, E_f, c_f)$ (the residual graph):

- Suppose the original graph has edges $e_1 = (u, v)$, and $e_2 = (v, u)$ between $u$ and $v$.

# The Residual Graph

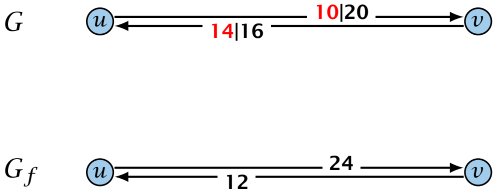
From the graph $G = (V, E, c)$ and the current flow $f$ we construct an auxiliary graph $G_f = (V, E_f, c_f)$ (the residual graph):

- Suppose the original graph has edges $e_1 = (u, v)$, and $e_2 = (v, u)$ between $u$ and $v$.
- $G_f$ has edge $e_1'$ with capacity $\max\{0, c(e_1) - f(e_1) + f(e_2)\}$ and $e_2'$ with with capacity $\max\{0, c(e_2) - f(e_2) + f(e_1)\}$.

# The Residual Graph

From the graph $G = (V, E, c)$ and the current flow $f$ we construct an auxiliary graph $G_f = (V, E_f, c_f)$ (the residual graph):

- Suppose the original graph has edges $e_1 = (u, v)$, and $e_2 = (v, u)$ between $u$ and $v$.
- $G_f$ has edge $e_1'$ with capacity $\max\{0, c(e_1) - f(e_1) + f(e_2)\}$ and $e_2'$ with with capacity $\max\{0, c(e_2) - f(e_2) + f(e_1)\}$.



# Augmenting Path Algorithm

**Definition 1**

An augmenting path with respect to flow $f$, is a path from $s$ to $t$ in the auxiliary graph $G_f$ that contains only edges with non-zero capacity.

**Algorithm 1** FordFulkerson($G = (V, E, c)$)

```
Initialize f(e) ← 0 for all edges.
while ∃ augmenting path p in G_f do
    augment as much flow along p as possible.
```

# Augmenting Path Algorithm

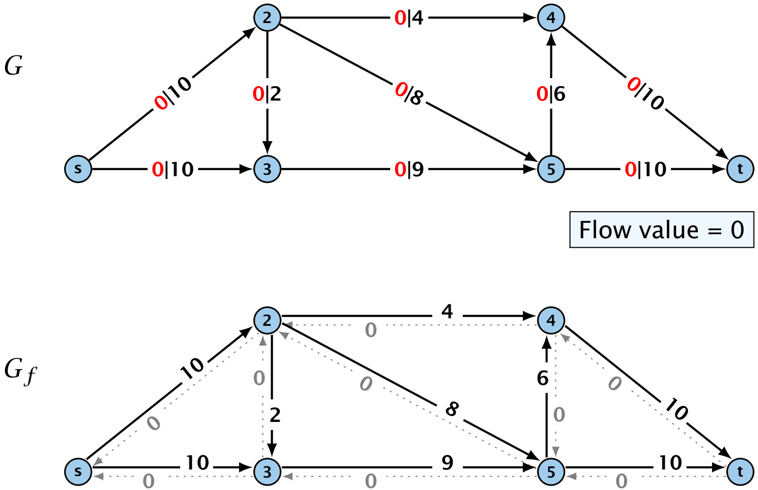
**Definition 1**

An augmenting path with respect to flow $f$, is a path from $s$ to $t$ in the auxiliary graph $G_f$ that contains only edges with non-zero capacity.

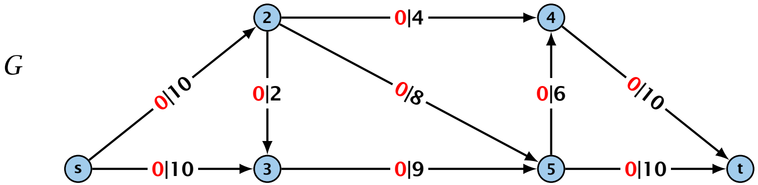
**Algorithm 1** FordFulkerson($G = (V, E, c)$)

1: Initialize $f(e) \leftarrow 0$ for all edges.
2: **while** $\exists$ augmenting path $p$ in $G_f$ **do**
3:     augment as much flow along $p$ as possible.

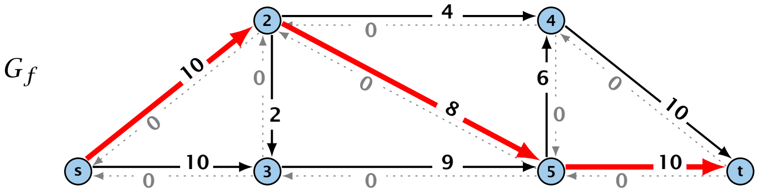
# Augmenting Path Algorithm



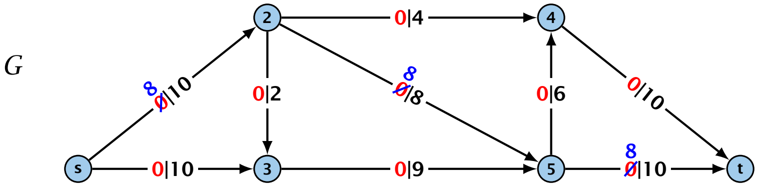
# Augmenting Path Algorithm



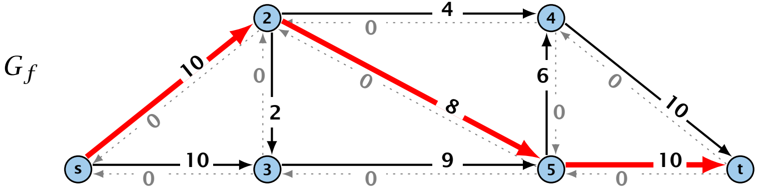

Flow value = 0



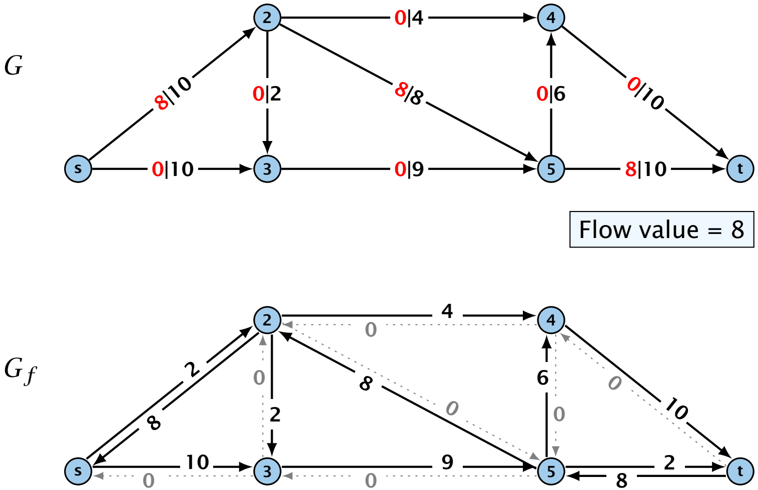
# Augmenting Path Algorithm



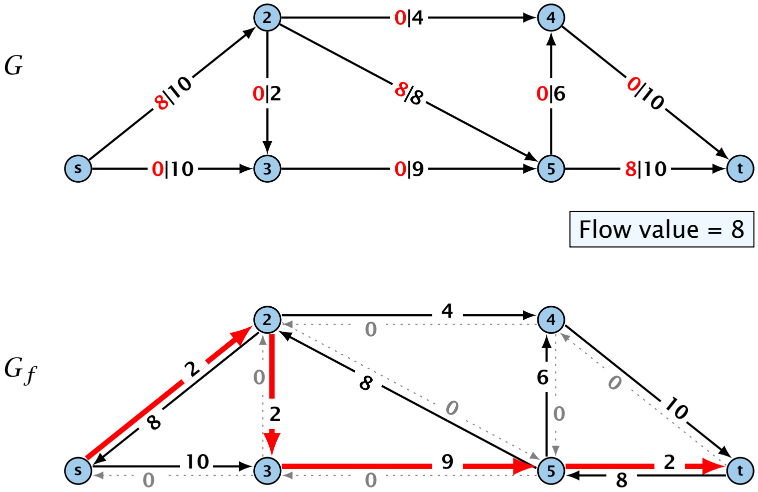

Flow value = 8



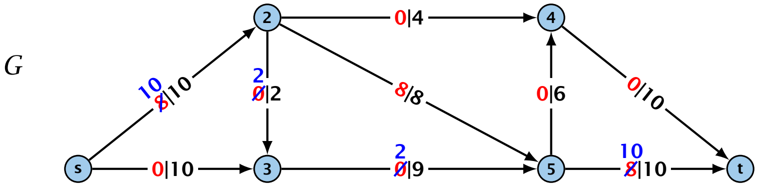
# Augmenting Path Algorithm



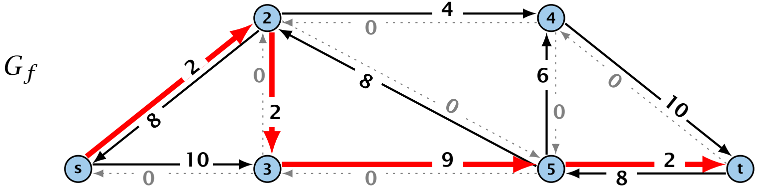
# Augmenting Path Algorithm



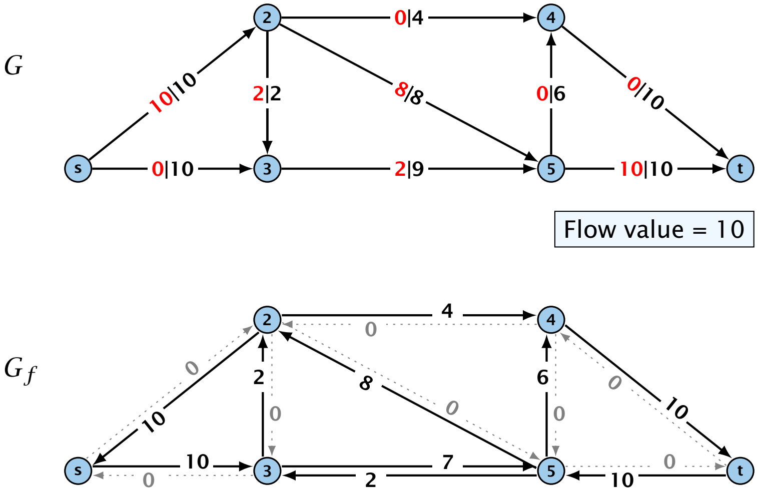
# Augmenting Path Algorithm



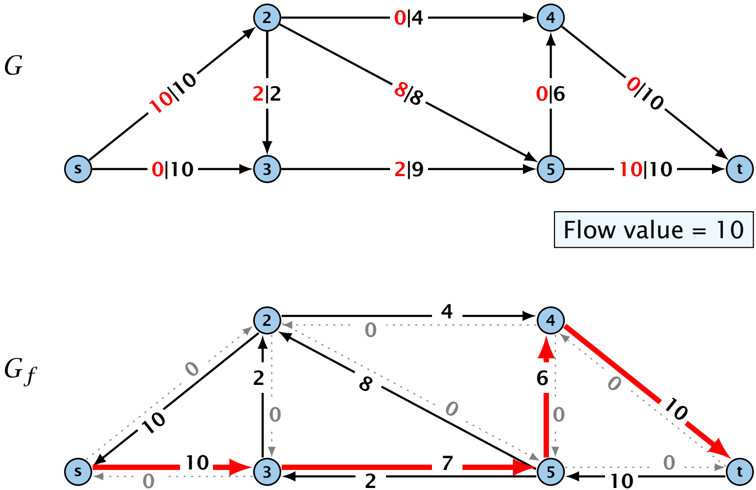

Flow value = 10



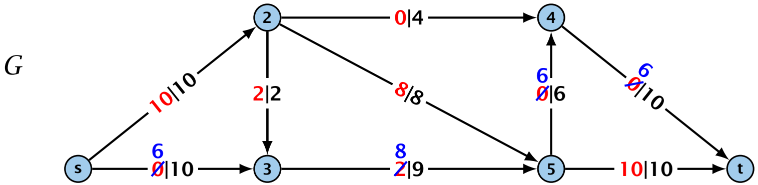
# Augmenting Path Algorithm



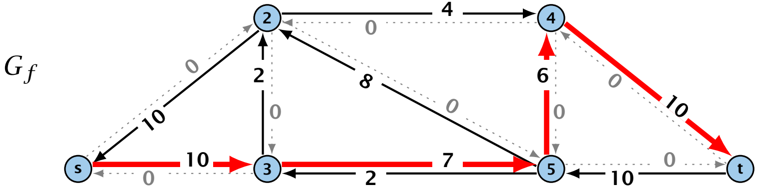
# Augmenting Path Algorithm



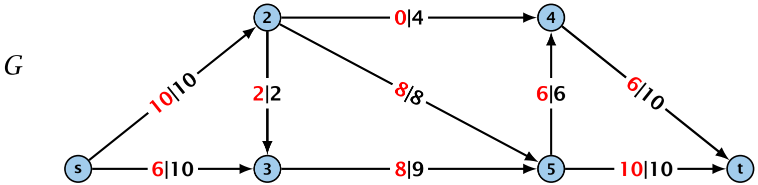
# Augmenting Path Algorithm



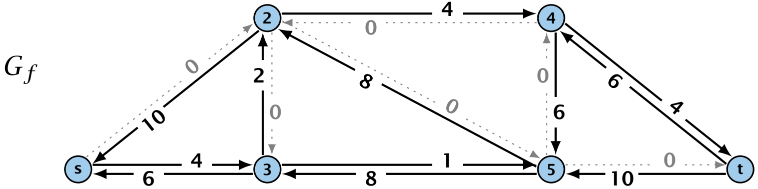

Flow value = 16



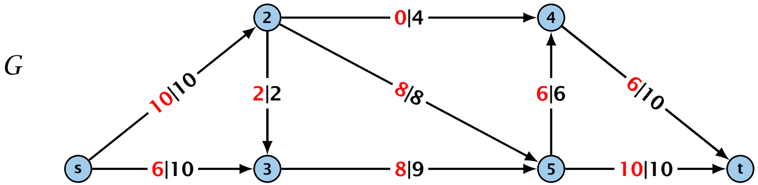
# Augmenting Path Algorithm



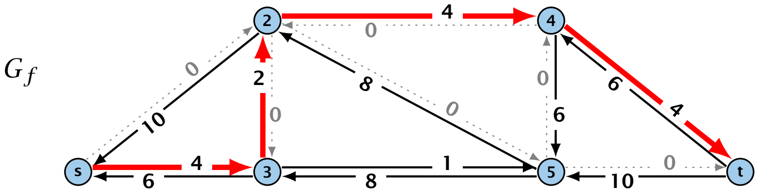

Flow value = 16



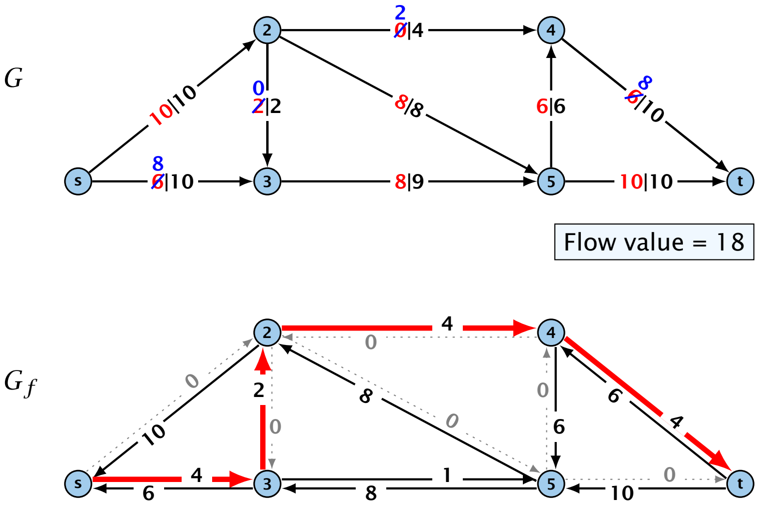
# Augmenting Path Algorithm



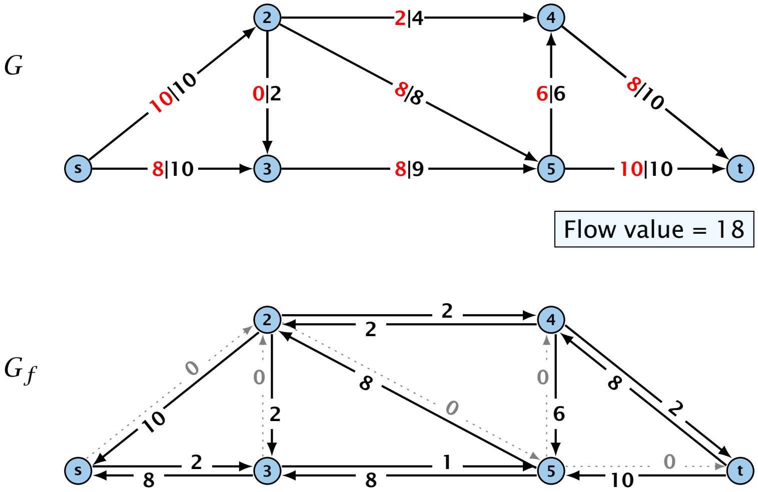

Flow value = 16



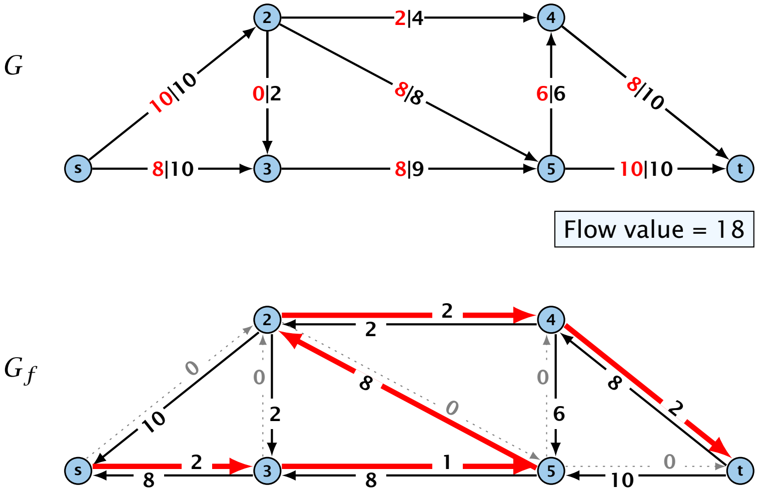
# Augmenting Path Algorithm



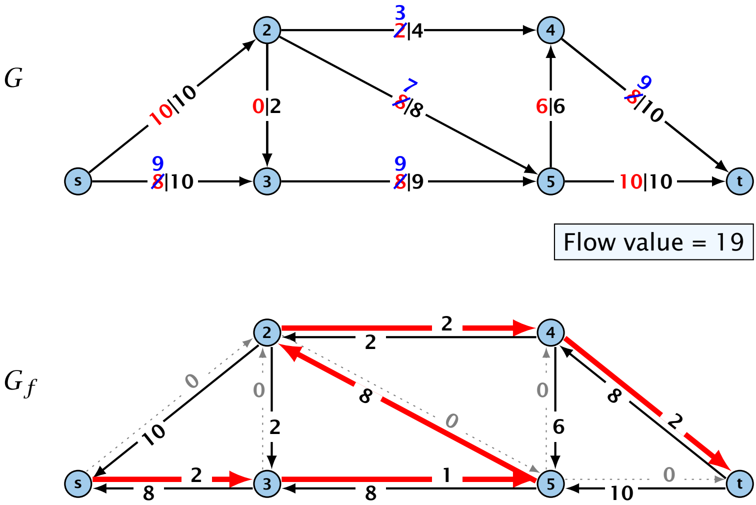
# Augmenting Path Algorithm



# Augmenting Path Algorithm

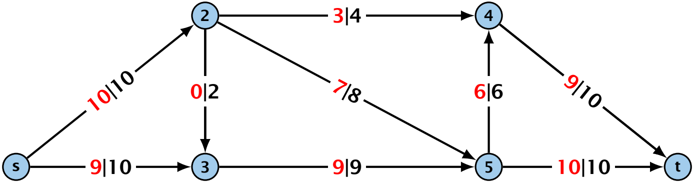


# Augmenting Path Algorithm



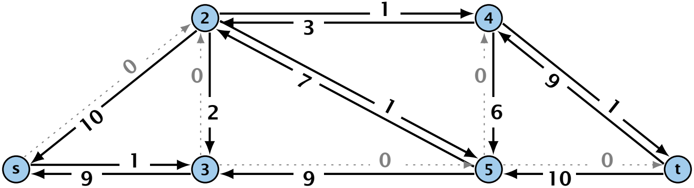
# Augmenting Path Algorithm

$G$

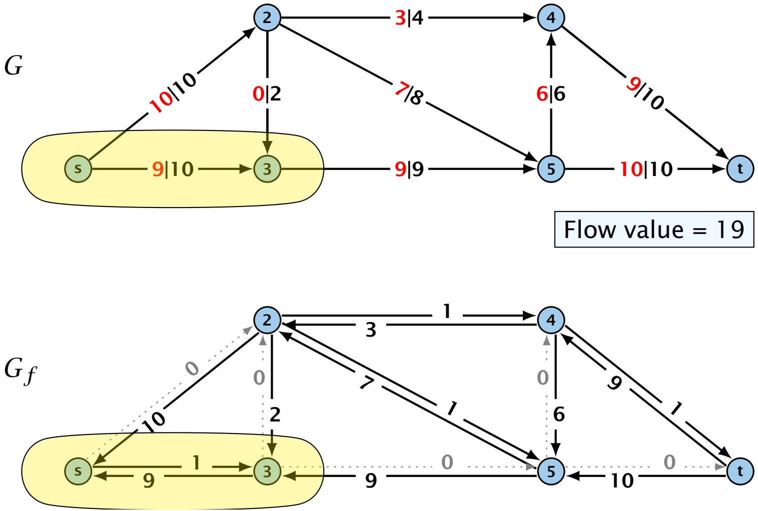



Flow value = 19

$G_f$



# Augmenting Path Algorithm



# Augmenting Path Algorithm

**Theorem 2**
*A flow $f$ is a maximum flow **iff** there are no augmenting paths.*

**Theorem 3**
*The value of a maximum flow is equal to the value of a minimum cut.*

**Proof.**
Let $f$ be a flow. The following are equivalent:

1. There exists a cut $A$ such that $\mathrm{val}(f) = \mathrm{cap}(A, V \setminus A)$.
2. Flow $f$ is a maximum flow.
3. There is no augmenting path w.r.t. $f$.

□

# Augmenting Path Algorithm

**Theorem 2**

*A flow $f$ is a maximum flow **iff** there are no augmenting paths.*

**Theorem 3**

*The value of a maximum flow is equal to the value of a minimum cut.*

**Proof.**

Let $f$ be a flow. The following are equivalent:

1. There exists a cut $A$ such that $\text{val}(f) = \text{cap}(A, V \setminus A)$.
2. Flow $f$ is a maximum flow.
3. There is no augmenting path w.r.t. $f$.

□

# Augmenting Path Algorithm

**Theorem 2**

*A flow $f$ is a maximum flow **iff** there are no augmenting paths.*

**Theorem 3**

*The value of a maximum flow is equal to the value of a minimum cut.*

**Proof.**

Let $f$ be a flow. The following are equivalent:

1. There exists a cut $A$ such that $\text{val}(f) = \text{cap}(A, V \setminus A)$.
2. Flow $f$ is a maximum flow.
3. There is no augmenting path w.r.t. $f$.

□

# Augmenting Path Algorithm

**Theorem 2**

*A flow $f$ is a maximum flow* ***iff*** *there are no augmenting paths.*

**Theorem 3**

*The value of a maximum flow is equal to the value of a minimum cut.*

**Proof.**

Let $f$ be a flow. The following are equivalent:

1. There exists a cut $A$ such that $\text{val}(f) = \text{cap}(A, V \setminus A)$.
2. Flow $f$ is a maximum flow.
3. There is no augmenting path w.r.t. $f$.

□

# Augmenting Path Algorithm

**Theorem 2**

*A flow $f$ is a maximum flow **iff** there are no augmenting paths.*

**Theorem 3**

*The value of a maximum flow is equal to the value of a minimum cut.*

**Proof.**

Let $f$ be a flow. The following are equivalent:

1. There exists a cut $A$ such that $\mathrm{val}(f) = \mathrm{cap}(A, V \setminus A)$.
2. Flow $f$ is a maximum flow.
3. There is no augmenting path w.r.t. $f$.

□

# Augmenting Path Algorithm

**Theorem 2**

*A flow $f$ is a maximum flow **iff** there are no augmenting paths.*

**Theorem 3**

*The value of a maximum flow is equal to the value of a minimum cut.*

**Proof.**

Let $f$ be a flow. The following are equivalent:

1. There exists a cut $A$ such that $\mathrm{val}(f) = \mathrm{cap}(A, V \setminus A)$.
2. Flow $f$ is a maximum flow.
3. There is no augmenting path w.r.t. $f$.

□

# Augmenting Path Algorithm

$1. \Longrightarrow 2.$

This we already showed.

$2. \Longrightarrow 3.$

If there were an augmenting path, we could improve the flow. Contradiction.

$3. \Longrightarrow 1.$

- Let $f$ be a flow with no augmenting paths.
- Let $A$ be the set of vertices reachable from $s$ in the residual graph along non-zero capacity edges.
- Since there is no augmenting path we have $s \in A$ and $t \notin A$.

# Augmenting Path Algorithm

1. $\Longrightarrow$ 2.

This we already showed.

2. $\Longrightarrow$ 3.

If there were an augmenting path, we could improve the flow. Contradiction.

3. $\Longrightarrow$ 1.

- Let $f$ be a flow with no augmenting paths.
- Let $A$ be the set of vertices reachable from $s$ in the residual graph along non-zero capacity edges.
- Since there is no augmenting path we have $s \in A$ and $t \notin A$.

# Augmenting Path Algorithm

1\. $\Longrightarrow$ 2.

This we already showed.

2\. $\Longrightarrow$ 3.

If there were an augmenting path, we could improve the flow. Contradiction.

3\. $\Longrightarrow$ 1.

- Let $f$ be a flow with no augmenting paths.
- Let $A$ be the set of vertices [illegible] graph along non-zero capacity edges.
- Since there is no augmenting path [illegible]

# Augmenting Path Algorithm

1. $\Longrightarrow$ 2.

This we already showed.

2. $\Longrightarrow$ 3.

If there were an augmenting path, we could improve the flow. Contradiction.

3. $\Longrightarrow$ 1.

- Let $f$ be a flow with no augmenting paths.
- Let $A$ be the set of vertices reachable from $s$ in the residual graph along non-zero capacity edges.
- Since there is no augmenting path we have $s \in A$ and $t \notin A$.

# Augmenting Path Algorithm

1. $\Longrightarrow$ 2.

This we already showed.

2. $\Longrightarrow$ 3.

If there were an augmenting path, we could improve the flow. Contradiction.

3. $\Longrightarrow$ 1.

- Let $f$ be a flow with no augmenting paths.
- Let $A$ be the set of vertices reachable from $s$ in the residual graph along non-zero capacity edges.
- Since there is no augmenting path we have $s \in A$ and $t \notin A$.

# Augmenting Path Algorithm

1. $\Rightarrow$ 2.

This we already showed.

2. $\Rightarrow$ 3.

If there were an augmenting path, we could improve the flow. Contradiction.

3. $\Rightarrow$ 1.

- Let $f$ be a flow with no augmenting paths.
- Let $A$ be the set of vertices reachable from $s$ in the residual graph along non-zero capacity edges.
- Since there is no augmenting path we have $s \in A$ and $t \notin A$.

# Augmenting Path Algorithm

$$\text{val}(f)$$

# Augmenting Path Algorithm

$$\text{val}(f) = \sum_{e \in \text{out}(A)} f(e) - \sum_{e \in \text{into}(A)} f(e)$$

# Augmenting Path Algorithm

$$\begin{aligned}\mathrm{val}(f) &= \sum_{e\in\mathrm{out}(A)} f(e) - \sum_{e\in\mathrm{into}(A)} f(e)\\ &= \sum_{e\in\mathrm{out}(A)} c(e)\end{aligned}$$

# Augmenting Path Algorithm

$$
\begin{aligned}
\mathrm{val}(f) &= \sum_{e\in\mathrm{out}(A)} f(e) - \sum_{e\in\mathrm{into}(A)} f(e) \\
&= \sum_{e\in\mathrm{out}(A)} c(e) \\
&= \mathrm{cap}(A, V\setminus A)
\end{aligned}
$$

# Augmenting Path Algorithm

$$\begin{aligned}\mathrm{val}(f) &= \sum_{e\in\mathrm{out}(A)} f(e) - \sum_{e\in\mathrm{into}(A)} f(e)\\ &= \sum_{e\in\mathrm{out}(A)} c(e)\\ &= \mathrm{cap}(A, V\setminus A)\end{aligned}$$

This finishes the proof.

Here the first equality uses the flow value lemma, and the second exploits the fact that the flow along incoming edges must be $0$ as the residual graph does not have edges leaving $A$.

# Analysis

Assumption:
All capacities are integers between $1$ and $C$.

Invariant:
Every flow value $f(e)$ and every residual capacity $c_f(e)$ remains integral troughout the algorithm.

# Analysis

Assumption:
All capacities are integers between $1$ and $C$.

Invariant:
Every flow value $f(e)$ and every residual capacity $c_f(e)$ remains integral troughout the algorithm.

**Lemma 4**

*The algorithm terminates in at most* $\mathrm{val}(f^*) \le nC$ *iterations, where* $f^*$ *denotes the maximum flow. Each iteration can be implemented in time* $\mathcal{O}(m)$*. This gives a total running time of* $\mathcal{O}(nmC)$*.*

**Theorem 5**

*If all capacities are integers, then there exists a maximum flow for which every flow value* $f(e)$ *is integral.*

**Lemma 4**

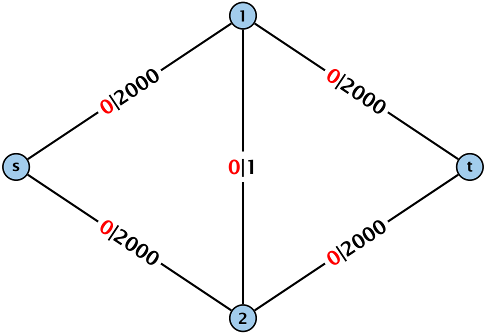
*The algorithm terminates in at most* $\mathrm{val}(f^*) \le nC$ *iterations, where* $f^*$ *denotes the maximum flow. Each iteration can be implemented in time* $\mathcal{O}(m)$*. This gives a total running time of* $\mathcal{O}(nmC)$*.*

**Theorem 5**

*If all capacities are integers, then there exists a maximum flow for which every flow value* $f(e)$ *is integral.*

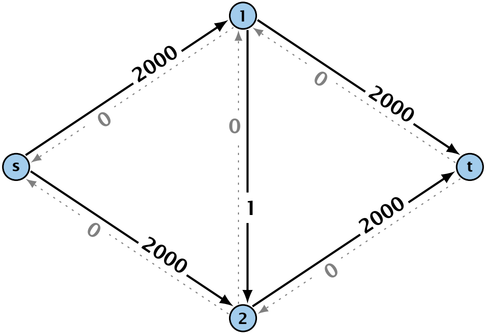
# A Bad Input

Problem: The running time may not be polynomial.



# A Bad Input

Problem: The running time may not be polynomial.

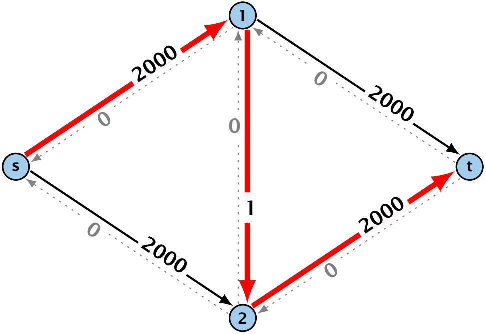



Question:

Can we tweak the algorithm so that the running time is polynomial in the input length?

# A Bad Input

Problem: The running time may not be polynomial.

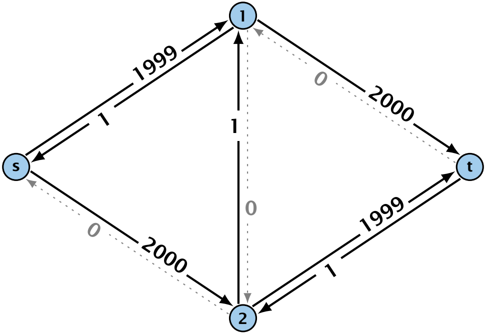



Question:

Can we tweak the algorithm so that the running time is polynomial in the input length?

# A Bad Input

Problem: The running time may not be polynomial.

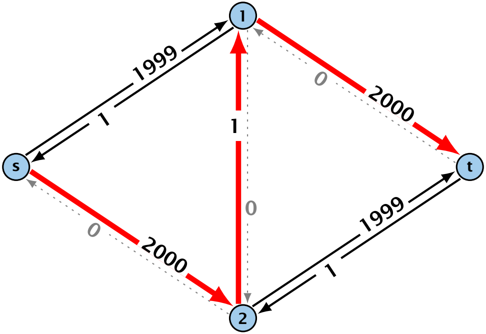



Question:

Can we tweak the algorithm so that the running time is polynomial in the input length?

# A Bad Input

Problem: The running time may not be polynomial.

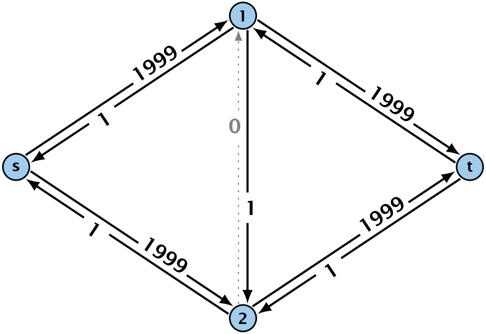



Question:

Can we tweak the algorithm so that the running time is polynomial in the input length?

# A Bad Input

Problem: The running time may not be polynomial.

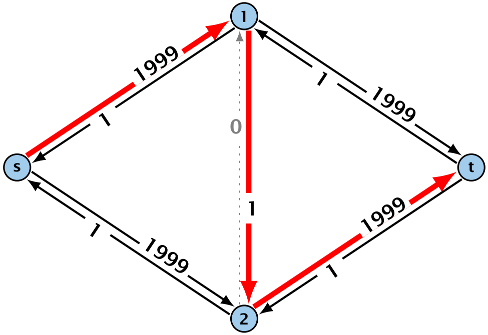



Question:

Can we tweak the algorithm so that the running time is polynomial in the input length?

# A Bad Input

Problem: The running time may not be polynomial.

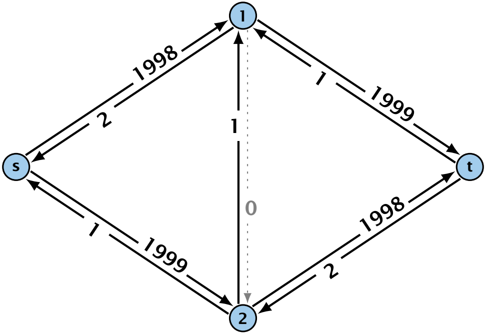



Question:

Can we tweak the algorithm so that the running time is polynomial in the input length?

# A Bad Input

Problem: The running time may not be polynomial.

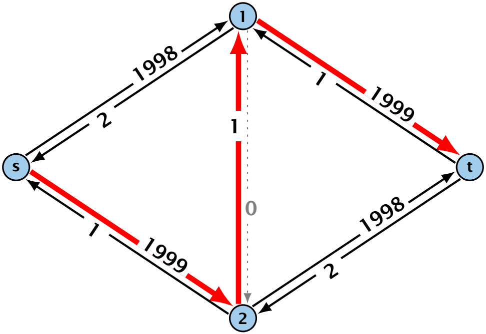



Question:

Can we tweak the algorithm so that the running time is polynomial in the input length?

# A Bad Input

Problem: The running time may not be polynomial.

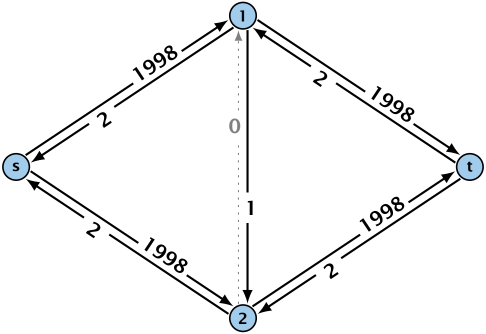



Question:

Can we tweak the algorithm so that the running time is polynomial in the input length?

# A Bad Input

Problem: The running time may not be polynomial.

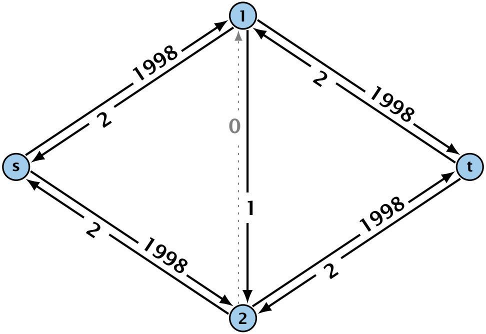



Question:

Can we tweak the algorithm so that the running time is polynomial in the input length?

# A Bad Input

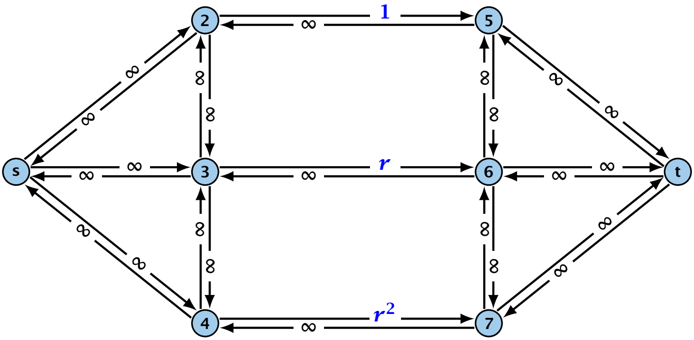
Problem: The running time may not be polynomial.




Question:
Can we tweak the algorithm so that the running time is polynomial in the input length?

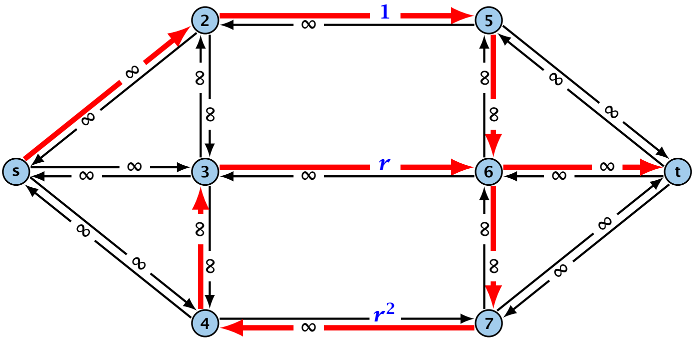
# A Pathological Input

Let $r = \frac{1}{2}(\sqrt{5} - 1)$. Then $r^{n+2} = r^n - r^{n+1}$.



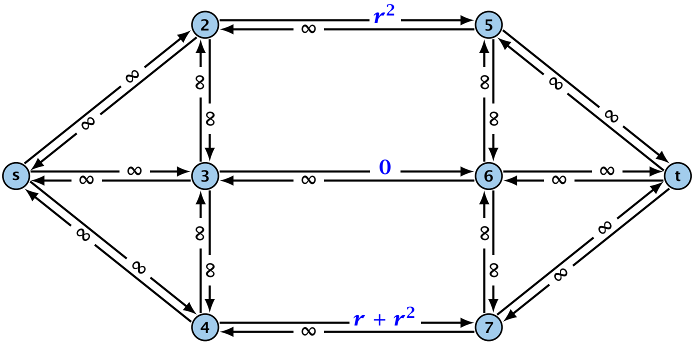
# A Pathological Input

Let $r = \frac{1}{2}(\sqrt{5} - 1)$. Then $r^{n+2} = r^n - r^{n+1}$.



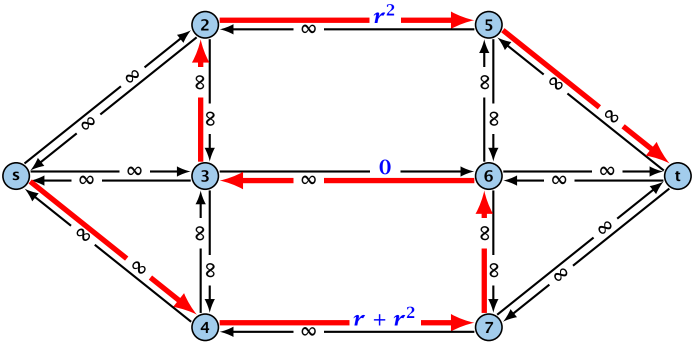
# A Pathological Input

Let $r = \frac{1}{2}(\sqrt{5} - 1)$. Then $r^{n+2} = r^n - r^{n+1}$.



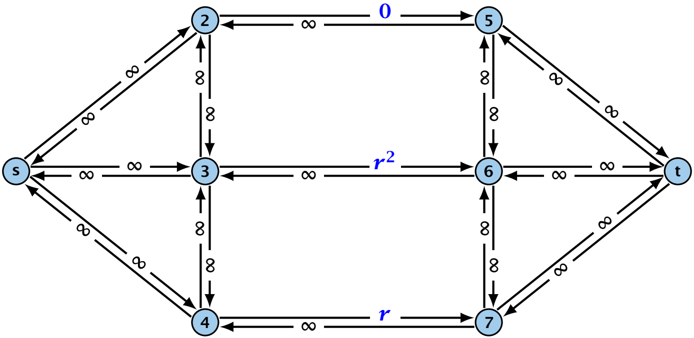
# A Pathological Input

Let $r = \frac{1}{2}(\sqrt{5} - 1)$. Then $r^{n+2} = r^n - r^{n+1}$.



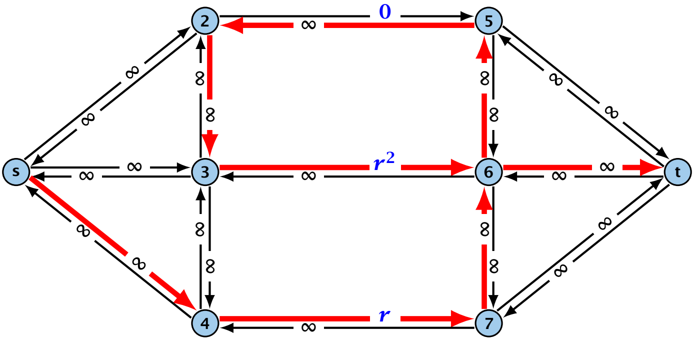
# A Pathological Input

Let $r = \frac{1}{2}(\sqrt{5} - 1)$. Then $r^{n+2} = r^n - r^{n+1}$.



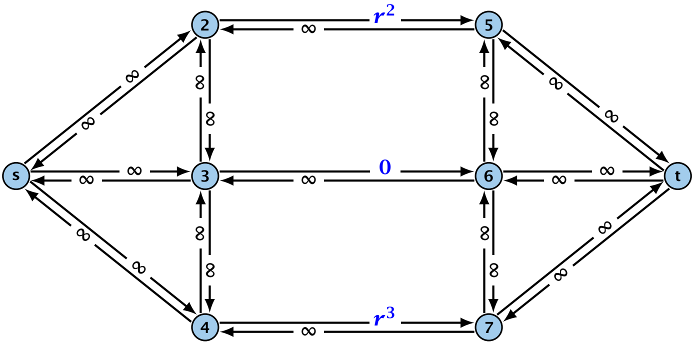
# A Pathological Input

Let $r = \frac{1}{2}(\sqrt{5} - 1)$. Then $r^{n+2} = r^n - r^{n+1}$.



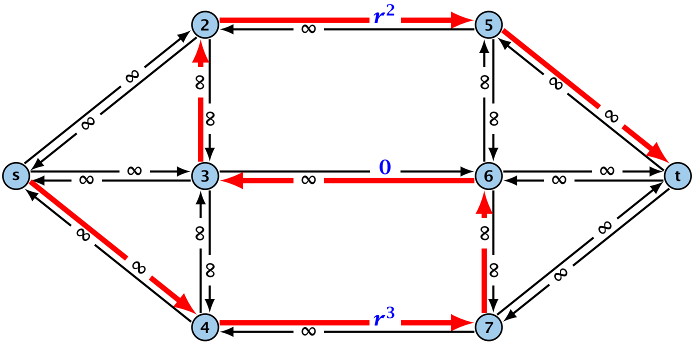
# A Pathological Input

Let $r = \frac{1}{2}(\sqrt{5} - 1)$. Then $r^{n+2} = r^n - r^{n+1}$.



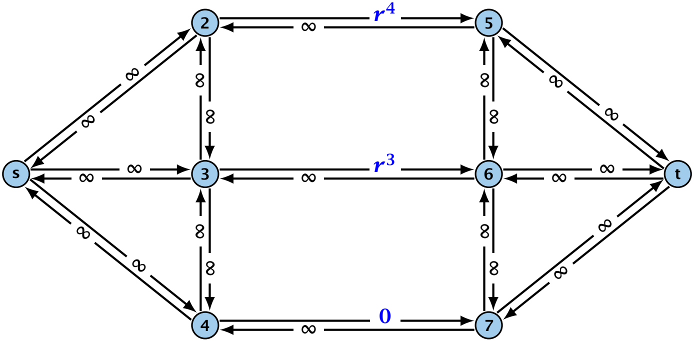
# A Pathological Input

Let $r = \frac{1}{2}(\sqrt{5} - 1)$. Then $r^{n+2} = r^n - r^{n+1}$.



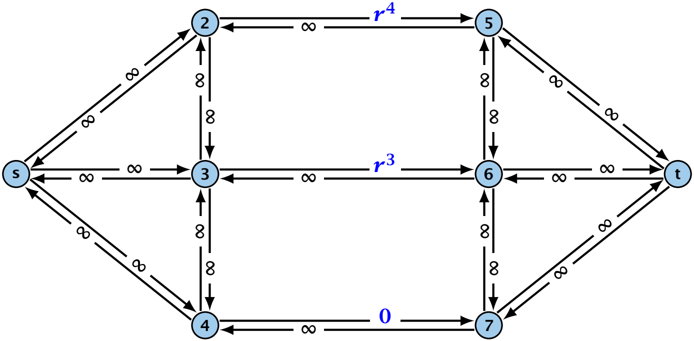
# A Pathological Input

Let $r = \frac{1}{2}(\sqrt{5} - 1)$. Then $r^{n+2} = r^n - r^{n+1}$.



# A Pathological Input

Let $r = \frac{1}{2}(\sqrt{5} - 1)$. Then $r^{n+2} = r^n - r^{n+1}$.




Running time may be infinite!!!

## How to choose augmenting paths?

## How to choose augmenting paths?

- We need to find paths efficiently.

**How to choose augmenting paths?**

- We need to find paths efficiently.
- We want to guarantee a small number of iterations.

**How to choose augmenting paths?**

- We need to find paths efficiently.
- We want to guarantee a small number of iterations.

**Several possibilities:**

**How to choose augmenting paths?**

- We need to find paths efficiently.
- We want to guarantee a small number of iterations.

**Several possibilities:**

- Choose path with maximum bottleneck capacity.

**How to choose augmenting paths?**

- We need to find paths efficiently.
- We want to guarantee a small number of iterations.

**Several possibilities:**

- Choose path with maximum bottleneck capacity.
- Choose path with sufficiently large bottleneck capacity.

**How to choose augmenting paths?**

- We need to find paths efficiently.
- We want to guarantee a small number of iterations.

**Several possibilities:**

- Choose path with maximum bottleneck capacity.
- Choose path with sufficiently large bottleneck capacity.
- Choose the shortest augmenting path.