School of Computer Science and En
University of New South Wales

8. MAXIMUM FLOW

Flow Networks

- A **flow network** $G = (V, E)$ is a directed graph in which each edge $e = (u, v) \in E$ has a positive integer capacity $c(u, v) > 0$.

  There are two distinguished vertices: a **source** s and a **sink** t; no edge leaves the sink and no edge enters the source.

- A **flow** in G is a function $f : E \to \mathbb{R}$ that satisfies:
  1. **Capacity constraint:** for all edges $f(u, v) \leq c(u, v)$.
  2. **Flow conservation:** For all $v \in V - \{s, t\}$ we require

$$\sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w).$$

Flow Networks

- A **flow network** $G = (V, E)$ is a directed graph in which each edge $e = (u, v) \in E$ has a positive integer capacity $c(u, v) > 0$.

  There are two distinguished vertices: a **source** s and a **sink** t; no edge leaves the sink and no edge enters the source.

- A **flow** in G is a function $f : E \to \mathbb{R}^+$, $f($
  1. **Capacity constraint:** for all edges $f(u, v) \leq c(u, v)$.
  2. **Flow conservation:** For all $v \in V - \{s, t\}$ we require

$$\sum_{(u,v) \in E} f(u, v) = \sum_{(v,w) \in E} f(v, w).$$

Flow Networks

- The **value of the flow** is defined as $|f| = \sum_{v:(s,v)\in E} f(s,v)$.

- Exa

- The first number on an edge: the flow through that edge;
- The second number: the capacity of the edge.

Flow Networks

- The **value of the flow** is defined as $|f| = \sum_{v:(s,v)\in E} f(s,v)$.
- Clearly, also $|f| = \sum_{v:(v,t)\in E} f(v,t)$.
- Exa
- The first number on an edge: the flow through that edge;
- The second number: the capacity of the edge.

Flow Networks

- The **value of the flow** is defined as $|f| = \sum_{v:(s,v)\in E} f(s,v)$.
- Clearly, also $|f| = \sum_{v:(v,t)\in E} f(v,t)$.
- Exa

- The first number on an edge: the flow through that edge;
- The second number: the capacity of the edge.

Flow Networks

- The **value of the flow** is defined as $|f| = \sum_{v:(s,v)\in E} f(s,v)$.
- Clearly, also $|f| = \sum_{v:(v,t)\in E} f(v,t)$.
- Exa
- The first number on an edge: the flow through that edge;
- The second number: the capacity of the edge.

Flow Networks

- The **value of the flow** is defined as $|f| = \sum_{v:(s,v)\in E} f(s,v)$.
- Clearly, also $|f| = \sum_{v:(v,t)\in E} f(v,t)$.
- Exa

- The first number on an edge: the flow through that edge;
- The second number: the capacity of the edge.

Flow Networks

- Examples of flow networks (possibly with several sources and many sinks): transportation networks, gas pipelines, computer networks...
- The **residual flow network** for a flow network with some flow in it: the network with the leftover capacities.
- Each edge of the original network has a leftover capacity: the capacity of the edge minus the flow through the edge.
- If the flow through an edge is equal to the capacity of the edge, the edge disappears in the residual network.
- New "virtual" edges appear in opposite direction of an original edge with some flow in it (unless there were already an edge in the opposite direction).
- They represent the possibility to reduce the flow through the original edge; thus their capacity is equal to the flow through the original edge (or, if there were already an edge in the opposite direction, the capacity of such an edge is increased for the amount of that flow; see vertices v_1 and v_2).

Flow Networks

- Examples of flow networks (possibly with several sources and many sinks): transportation networks, gas pipelines, computer networks...
- The **residual flow network** for a flow network with some flow in it: the network with the leftover capacities
  - Each edge of the original network has a leftover capacity: the capacity of the edge minus the flow through the edge.
  - If the flow through an edge is equal to the capacity of the edge, it disappears in the residual network.
  - New "virtual" edges appear in opposite direction of an original edge with some flow in it (unless there were already an edge in the opposite direction).
  - They represent the possibility to reduce the flow through the original edge; thus their capacity is equal to the flow through the original edge (or, if there were already an edge in the opposite direction, the capacity of such an edge is increased for the amount of that flow; see vertices v_1 and v_2).

Flow Networks

- Examples of flow networks (possibly with several sources and many sinks): transportation networks, gas pipelines, computer networks...
- The **residual flow network** for a flow network with some flow in it: the network with the leftover capacities
- Each edge of the original network has a leftover capacit
  the capacity of the edge minus the flow through the edge
- If the flow through an edge is equal to the capacity of the edge, the edge disappears in the residual network.
- New "virtual" edges appear in opposite direction of an original edge with some flow in it (unless there were already an edge in the opposite direction).
- They represent the possibility to reduce the flow through the original edge; thus their capacity is equal to the flow through the original edge (or, if there were already an edge in the opposite direction, the capacity of such an edge is increased for the amount of that flow; see vertices v_1 and v_2).

Flow Networks

- Examples of flow networks (possibly with several sources and many sinks): transportation networks, gas pipelines, computer networks...
- The **residual flow network** for a flow network with some flow in it: the network with the leftover capacities
- Each edge of the original network has a leftover capacit
  the capacity of the edge minus the flow through the edge
- If the flow through an edge is equal to the capacity of the edg
  disappears in the residual network.
- New "virtual" edges appear in opposite direction of an original edge with some flow in it (unless there were already an edge in the opposite direction).
- They represent the possibility to reduce the flow through the original edge; thus their capacity is equal to the flow through the original edge (or, if there were already an edge in the opposite direction, the capacity of such an edge is increased for the amount of that flow; see vertices v_1 and v_2).

Flow Networks

- Examples of flow networks (possibly with several sources and many sinks): transportation networks, gas pipelines, computer networks...
- The **residual flow network** for a flow network with some flow in it: the network with the leftover capacities

- Each edge of the original network has a leftover capacit
  the capacity of the edge minus the flow through the edge
- If the flow through an edge is equal to the capacity of the edg
  disappears in the residual network.
- New "virtual" edges appear in opposite direction of an original edge with some flow in it (unless there were already an edge in the opposite direction).
- They represent the possibility to reduce the flow through the original edge; thus their capacity is equal to the flow through the original edge (or, if there were already an edge in the opposite direction, the capacity of such an edge is increased for the amount of that flow; see vertices v_1 and v_2).

Flow Networks

- Examples of flow networks (possibly with several sources and many sinks): transportation networks, gas pipelines, computer networks...
- The **residual flow network** for a flow network with some flow in it: the network with the leftover capacities

- Each edge of the original network has a leftover capacit
  the capacity of the edge minus the flow through the edge
- If the flow through an edge is equal to the capacity of the edg
  disappears in the residual network.
- New "virtual" edges appear in opposite direction of an original edge with some flow in it (unless there were already an edge in the opposite direction).
- They represent the possibility to reduce the flow through the original edge; thus their capacity is equal to the flow through the original edge (or, if there were already an edge in the opposite direction, the capacity of such an edge is increased for the amount of that flow; see vertices v_1 and v_2).

Flow Networks

- Residual flow networks can be used to increase the total flow through the network by adding an **augmenting path**

- The s
  i.e., t
- We can now recalculate the flow through all edges alon
  by adding the additional flow through the path if the flo
  augmenting path is in the same direction as the original
  if in opposite direction

Flow Networks

- Residual flow networks can be used to increase the total flow through the network by adding an **augmenting path**

- The c
  i.e., t
- We can now recalculate the flow through all edges alon
  by adding the additional flow through the path if the flo
  augmenting path is in the same direction as the original
  if in opposite direction

Flow Networks

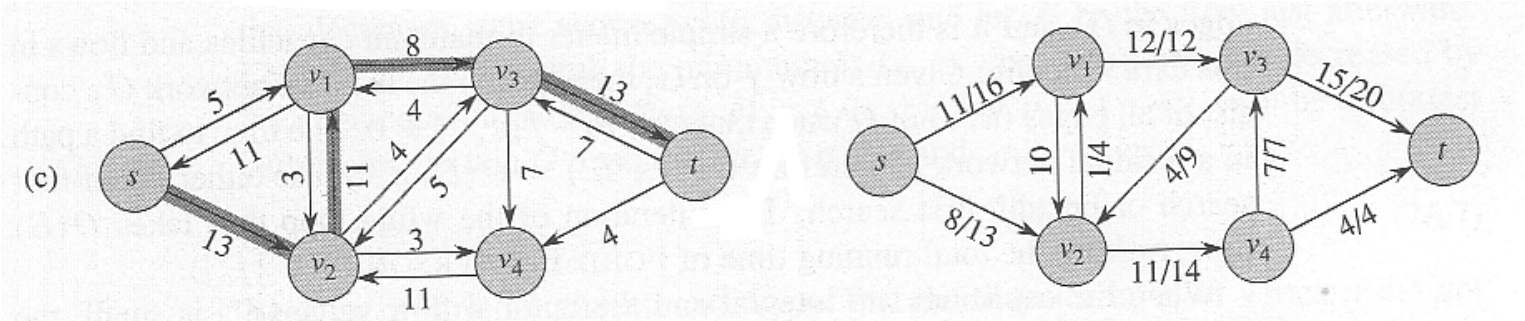
- Residual flow networks can be used to increase the total flow through the network by adding an **augmenting path**

- The c
  i.e., t
- We can now recalculate the flow through all edges alon
  by adding the additional flow through the path if the flo
  augmenting path is in the same direction as the original
  if in opposite direction.

Ford Fulkerson method for finding maximal flow

Ford - Fulkerson algorithm for finding maximal flow in a flow network:

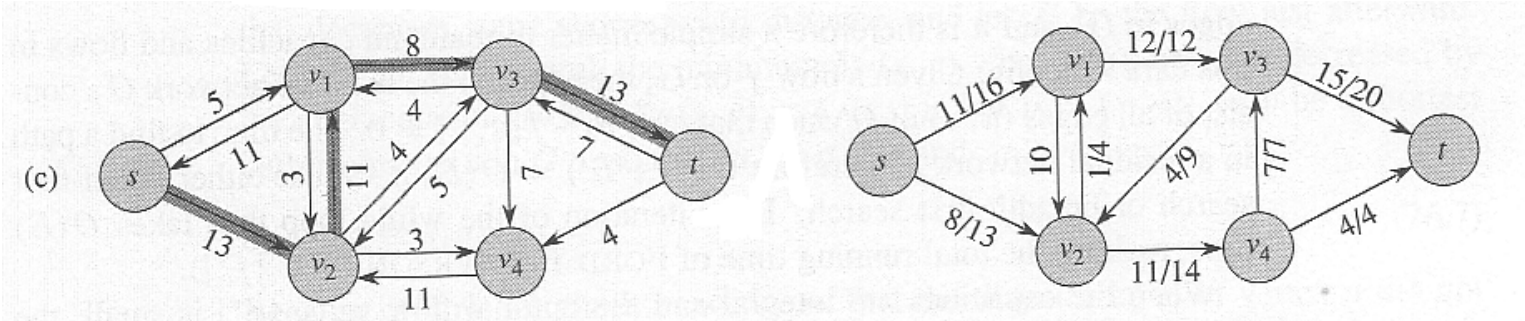
- Keep adding flow through new augmenting paths for as long as it is possible.
- When there are no more augmenting paths, you have achieved the largest possible flow in the network.



Ford Fulkerson method for finding maximal flow

Ford - Fulkerson algorithm for finding maximal flow in a flow network:

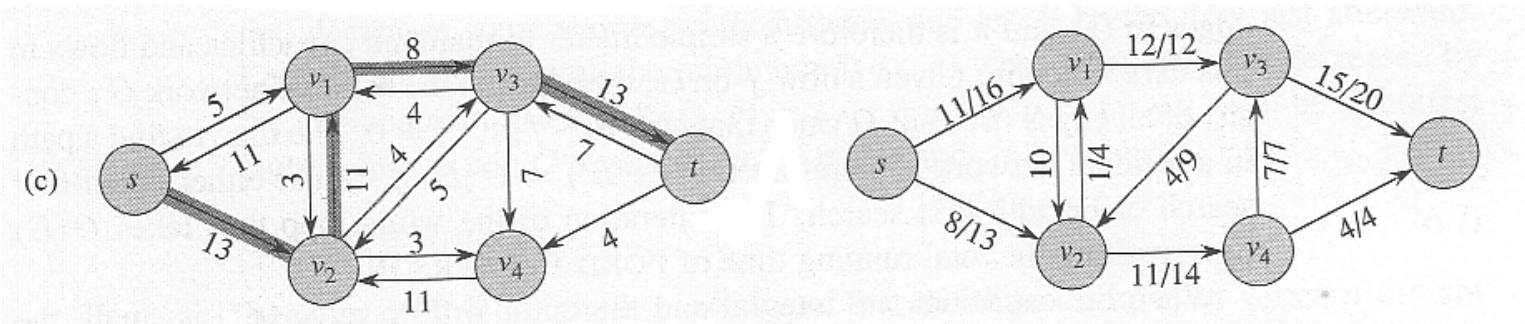
- Keep adding flow through new augmenting paths for as long as it is possible.
- When there are no more augmenting paths, you have achieved the largest possible flow in the network.



Ford Fulkerson method for finding maximal flow

Ford - Fulkerson algorithm for finding maximal flow in a flow network:

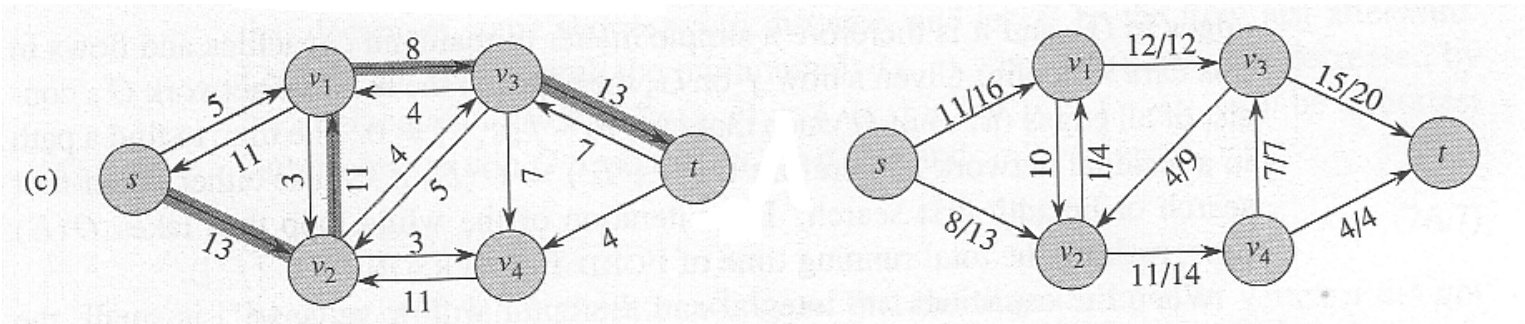
- Keep adding flow through new augmenting paths for as long as it is possible.
- When there are no more augmenting paths, you have achieved the largest possible flow in the network.



Ford Fulkerson method for finding maximal flow

Ford - Fulkerson algorithm for finding maximal flow in a flow network:

- Keep adding flow through new augmenting paths for as long as it is possible.
- When there are no more augmenting paths, you have achieved the largest possible flow in the network.



Ford - Fulkerson method for finding maximal flow

Ford Fulkerson algorithm for finding maximal flow in a flow network:

- Keep adding flow through new augmenting paths for as long as it is possible;
- When there are no more augmenting paths, you have achieved the largest possible flow in the network.

- Why can't we get stuck in a loop, which keeps adding augmenting paths forever?
- If all the capacities are integers, then each augmenting path through the network for at least 1 unit;
- the total flow cannot be larger than the sum of all capacities of all edges leaving the source, so eventually the process must terminate.

Ford - Fulkerson method for finding maximal flow

Ford Fulkerson algorithm for finding maximal flow in a flow network:

- Keep adding flow through new augmenting paths for as long as it is possible;
- When there are no more augmenting paths, you have achieved the largest possible flow in the network.

- Wh

- Why can't we get stuck in a loop, which keeps adding augmenting paths forever?
- If all the capacities are integers, then each augmentin through the network for at least 1 unit;
- the total flow cannot be larger than the sum of all capacities of all edges leaving the source, so eventually the process must terminate.

Ford - Fulkerson method for finding maximal flow

Ford Fulkerson algorithm for finding maximal flow in a flow network:

- Keep adding flow through new augmenting paths for as long as it is possible;
- When there are no more augmenting paths, you have achieved the largest possible flow in the network.

- Wh
- Why can't we get stuck in a loop, which keeps adding augmenting paths forever?
- If all the capacities are integers, then each augmentin through the network for at least 1 unit;
- the total flow cannot be larger than the sum of all capacities of all edges leaving the source, so eventually the process must terminate.

Ford - Fulkerson method for finding maximal flow

Ford Fulkerson algorithm for finding maximal flow in a flow network:

- Keep adding flow through new augmenting paths for as long as it is possible;
- When there are no more augmenting paths, you have achieved the largest possible flow in the network.

- Wh
- Why can't we get stuck in a loop, which keeps adding augmenting paths forever?
- If all the capacities are integers, then each augmentin through the network for at least 1 unit;
- the total flow cannot be larger than the sum of all capacities of all edges leaving the source, so eventually the process must terminate.

Ford - Fulkerson method for finding maximal flow

Ford Fulkerson algorithm for finding maximal flow in a flow network:

- Keep adding flow through new augmenting paths for as long as it is possible;
- When there are no more augmenting paths, you have achieved the largest possible flow in the network.

- Wh
- Why can't we get stuck in a loop, which keeps adding augmenting paths forever?
- If all the capacities are integers, then each augmentin through the network for at least 1 unit;
- the total flow cannot be larger than the sum of all capacities of all edges leaving the source, so eventually the process must terminate.

Ford - Fulkerson method for finding maximal flow

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?
- Maybe we have created bottlenecks by choosing bad augmenting paths; maybe better choices of augmenting paths could produce a larger total flow through the n
- This i
  mat
- The p
- A **cut** in a flow network is any partition of the vertices o
  into two subsets S and T such that
  1. $S \cup T = V$
  2. $S \cap T = \emptyset$
  3. $s \in S$ and $t \in T$.

Ford - Fulkerson method for finding maximal flow

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?
- Maybe we have created bottlenecks by choosing bad augmenting paths; maybe better choices of augmenting paths could produce a larger total flow through the n
- This i
  mat
- The p
- A **cut** in a flow network is any partition of the vertices o
  into two subsets S and T such that
  1. $S \cup T = V$
  2. $S \cap T = \emptyset$
  3. $s \in S$ and $t \in T$.

Ford - Fulkerson method for finding maximal flow

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?
- Maybe we have created bottlenecks by choosing bad augmenting paths; maybe better choices of augmenting paths could produce a larger total flow through the n
- This i
mat
- The p
- A **cut** in a flow network is any partition of the vertices o
into two subsets S and T such that:
  1. $S \cup T = V$
  2. $S \cap T = \emptyset$
  3. $s \in S$ and $t \in T$.

Ford - Fulkerson method for finding maximal flow

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?
- Maybe we have created bottlenecks by choosing bad augmenting paths; maybe better choices of augmenting paths could produce a larger total flow through the n
- This i
mat
- The p

- A **cut** in a flow network is any partition of the vertices o
into two subsets S and T such that
  1. $S \cup T = V$
  2. $S \cap T = \emptyset$
  3. $s \in S$ and $t \in T$.

Ford - Fulkerson method for finding maximal flow

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?
- Maybe we have created bottlenecks by choosing bad augmenting paths; maybe better choices of augmenting paths could produce a larger total flow through the n
- This i
mat
- The p

- A **cut** in a flow network is any partition of the vertices o
into two subsets S and T such that:
  1. $S \cup T = V$
  2. $S \cap T = \emptyset$
  3. $s \in S$ and $t \in T$.

Ford - Fulkerson method for finding maximal flow

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?
- Maybe we have created bottlenecks by choosing bad augmenting paths; maybe better choices of augmenting paths could produce a larger total flow through the n
- This i
  mat
- The p

- A **cut** in a flow network is any partition of the vertices o
  into two subsets S and T such that:
  1. $S \cup T = V$
  2. $S \cap T = \emptyset$
  3. $s \in S$ and $t \in T$.

Ford - Fulkerson method for finding maximal flow

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?
- Maybe we have created bottlenecks by choosing bad augmenting paths; maybe better choices of augmenting paths could produce a larger total flow through the n
- This i
mat
- The p

- A **cut** in a flow network is any partition of the vertices o into two subsets S and T such that:
  1. $S \cup T = V$
  2. $S \cap T = \emptyset$
  3. $s \in S$ and $t \in T$.

Ford - Fulkerson method for finding maximal flow

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?
- Maybe we have created bottlenecks by choosing bad augmenting paths; maybe better choices of augmenting paths could produce a larger total flow through the n
- This i
mat
- The p

- A **cut** in a flow network is any partition of the vertices o into two subsets S and T such that:
  1. $S \cup T = V$
  2. $S \cap T = \emptyset$
  3. $s \in S$ and $t \in T$.

Cuts in flow networks

- The **capacity** $c(S,T)$ **of a cut** (S,T) is the sum of capacities of all edges leaving S and entering T, i.e.

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

- Not S d T to

- The **flow through a cut** $f(S,T)$ is the total fl S to T minus the total flow through edges from T

$$f(S,T) = \sum_{(u,v)\in E} \{f(u,v) : u \in S \& v \in T \qquad v \in S\}$$

- Clearly, $f(S,T) \leq c(S,T)$ because for every edge $(u,v) \in E$ we assumed $f(u,v) \leq c(u,v)$, and $f(u,v) \geq 0$.

Cuts in flow networks

- The **capacity** $c(S,T)$ **of a cut** (S,T) is the sum of capacities of all edges leaving S and entering T, i.e.

$$c(S,T) = \sum_{(u,v)\in E} \{c(u,v) : u \in S \;\&\; v \in T\}$$

- Not
  S d
  T to
- The **flow through a cut** $f(S,T)$ is the total fl
  S to T
  minus the total flow through edges from T

$$f(S,T) = \sum_{(u,v)\in E} \{f(u,v) : u \in S \;\&\; v \in T$$
$$\sum_{(u,v)\in E} \{f(u,v) : u \in S \;\&\; v \in S\}$$

- Clearly, $f(S,T) \leq c(S,T)$ because for every edge $(u,v) \in E$ we assumed $f(u,v) \leq c(u,v)$, and $f(u,v) \geq 0$.

Cuts in flow networks

- The **capacity** $c(S,T)$ **of a cut** (S,T) is the sum of capacities of all edges leaving S and entering T, i.e.

$$c(S,T) = \sum_{(u,v)\in E} \{c(u,v) : u \in S \;\&\; v \in T\}$$

- Not
S d T to

- The **flow through a cut** $f(S,T)$ is the total fl S to T minus the total flow through edges from T

$$f(S,T) = \sum_{(u,v)\in E} \{f(u,v) : u \in S \;\&\; v \in T \qquad v \in S\}$$

- Clearly, $f(S,T) \leq c(S,T)$ because for every edge $(u,v) \in E$ we assumed $f(u,v) \leq c(u,v)$, and $f(u,v) \geq 0$.

Cuts in flow networks

- The **capacity** $c(S,T)$ **of a cut** (S,T) is the sum of capacities of all edges leaving S and entering T, i.e.

$$c(S,T) = \sum_{(u,v)\in E} \{c(u,v) : u \in S \;\&\; v \in T\}$$

- Not
S d T to

- The **flow through a cut** $f(S,T)$ is the total fl S to T minus the total flow through edges from T

$$f(S,T) = \sum_{(u,v)\in E} \{f(u,v) : u \in S \;\&\; v \in T \qquad\qquad (u,v)\in E \qquad v \in S\}$$

- Clearly, $f(S,T) \leq c(S,T)$ because for every edge $(u,v) \in E$ we assumed $f(u,v) \leq c(u,v)$, and $f(u,v) \geq 0$.

Cuts in flow networks

- The **capacity** $c(S,T)$ **of a cut** (S,T) is the sum of capacities of all edges leaving S and entering T, i.e.

$$c(S,T) = \sum_{(u,v)\in E} \{c(u,v) : u \in S \;\&\; v \in T\}$$

- Not
S d
T to

- The **flow through a cut** $f(S,T)$ is the total fl
S to T
minus the total flow through edges from T

$$f(S,T) = \sum_{(u,v)\in E} \{f(u,v) : u \in S \;\&\; v \in T$$
$$(u,v)\in E \qquad v \in S\}$$

- Clearly, $f(S,T) \leq c(S,T)$ because for every edge $(u,v) \in E$ we assumed $f(u,v) \leq c(u,v)$, and $f(u,v) \geq 0$.

Cuts in flow networks

- Example:

- In the

$$f(S,T) = f(v_1, v_3) + f(v_2, v_4) -$$

- Note that the flow in the opposite direction (from T to S) is s
- The capacity of the cut $c(S,T)$ is given by

$$c(S,T) = c(v_1, v_3) + c(v_2, v_4) = 12 + 14 = 26$$

- As we have mentioned, we add only the capacities of vertices from S to T and not of vertices in the opposite direction.

Cuts in flow networks

- Example:

- In the

$$f(S,T) = f(v_1,v_3) + f(v_2,v_4) -$$

- Note that the flow in the opposite direction (from T to S) is s
- The capacity of the cut $c(S,T)$ is given by

$$c(S,T) = c(v_1,v_3) + c(v_2,v_4) = 12 + 14 = 26$$

- As we have mentioned, we add only the capacities of vertices from S to T and not of vertices in the opposite direction.

Cuts in flow networks

- Example:

- In the

$$f(S,T) = f(v_1,v_3) + f(v_2,v_4) -$$

- Note that the flow in the opposite direction (from T to S) is s
- The capacity of the cut $c(S,T)$ is given by

$$c(S,T) = c(v_1,v_3) + c(v_2,v_4) = 12 + 14 = 26$$

- As we have mentioned, we add only the capacities of vertices from S to T and not of vertices in the opposite direction.

Cuts in flow networks

- Example:

- In the

$$f(S,T) = f(v_1,v_3) + f(v_2,v_4) -$$

- Note that the flow in the opposite direction (from T to S) is s
- The capacity of the cut $c(S,T)$ is given by

$$c(S,T) = c(v_1,v_3) + c(v_2,v_4) = 12 + 14 = 26$$

- As we have mentioned, we add only the capacities of vertices from S to T and not of vertices in the opposite direction.

Cuts in flow networks

- Example:

- In the

$$f(S,T) = f(v_1,v_3) + f(v_2,v_4) -$$

- Note that the flow in the opposite direction (from T to S) is s
- The capacity of the cut $c(S,T)$ is given by

$$c(S,T) = c(v_1,v_3) + c(v_2,v_4) = 12 + 14 = 26$$

- As we have mentioned, we add only the capacities of vertices from S to T and not of vertices in the opposite direction.

Cuts in flow networks

- **Theorem:** The maximal amount of flow in a flow network is equal to the capacity of the cut of minimal capacity.
- Since any flow has to cross every cut, any flow must be smaller than the capa
- Thu $S, T)$, then such
- We now show that when the Ford - Fulkerson algorithm terminates, it produces a flow equal to the capacity of an appropriately defined cut.

Cuts in flow networks

- **Theorem:** The maximal amount of flow in a flow network is equal to the capacity of the cut of minimal capacity.
- Since any flow has to cross every cut, any flow must be smaller than the capa
- Thu $S, T)$, then such
- We now show that when the Ford - Fulkerson algorithm terminates, it produces a flow equal to the capacity of an appropriately defined cut.

Cuts in flow networks

- **Theorem:** The maximal amount of flow in a flow network is equal to the capacity of the cut of minimal capacity.
- Since any flow has to cross every cut, any flow must be smaller than the capa
- Thu $S, T)$, then such

- We now show that when the Ford - Fulkerson algorithm terminates, it produces a flow equal to the capacity of an appropriately defined cut.

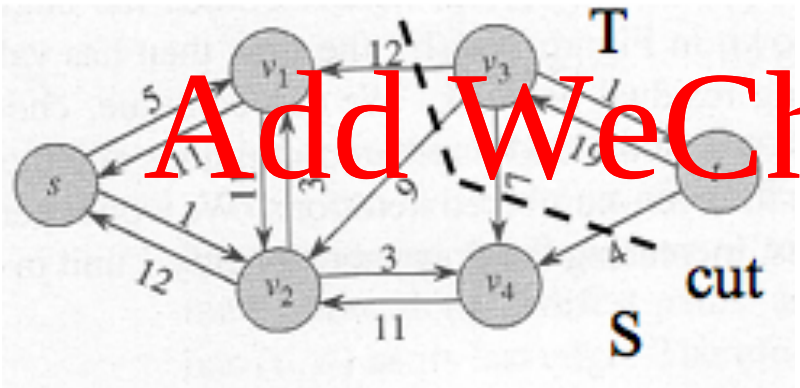
Cuts in flow networks

- **Theorem:** The maximal amount of flow in a flow network is equal to the capacity of the cut of minimal capacity.
- Since any flow has to cross every cut, any flow must be smaller than the capa
- Thu $S, T)$, then such

- We now show that when the Ford - Fulkerson algorithm terminates, it produces a flow equal to the capacity of an appropriately defined cut.

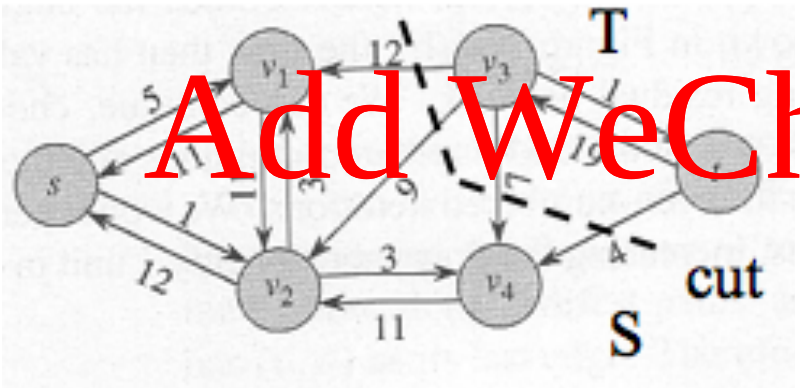
Cuts in flow networks

- Assume that the Ford - Fulkerson algorithm has terminated and that there no more augmenting paths from the source s to the sink t in the last residual network flow.
- Define S to be the source s and all vertices u such that there is a path in the resi
- Defi
- Sinc belo



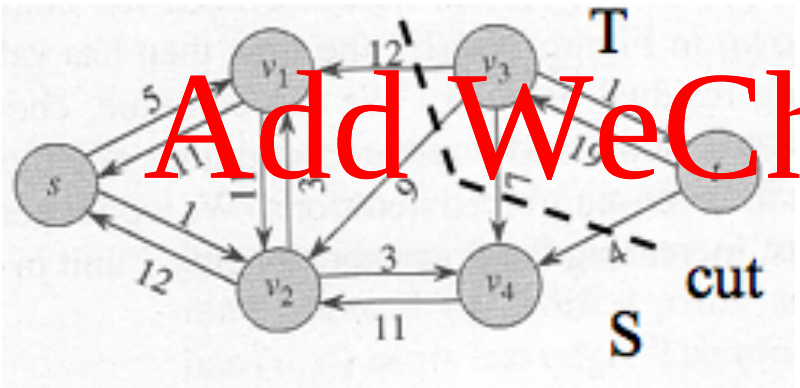
Cuts in flow networks

- Assume that the Ford - Fulkerson algorithm has terminated and that there no more augmenting paths from the source s to the sink t in the last residual network flow.
- Define S to be the source s and all vertices u such that there is a path in the resi
- Defi
- Sinc belo



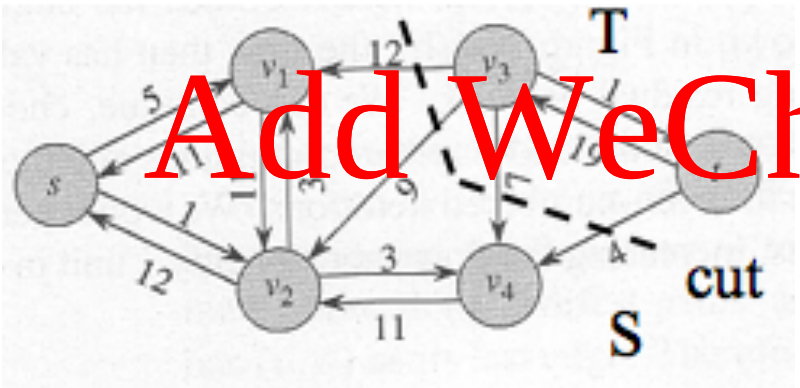
Cuts in flow networks

- Assume that the Ford - Fulkerson algorithm has terminated and that there no more augmenting paths from the source s to the sink t in the last residual network flow.
- Define S to be the source s and all vertices u such that there is a path in the resi
- Defi
- Sinc belo



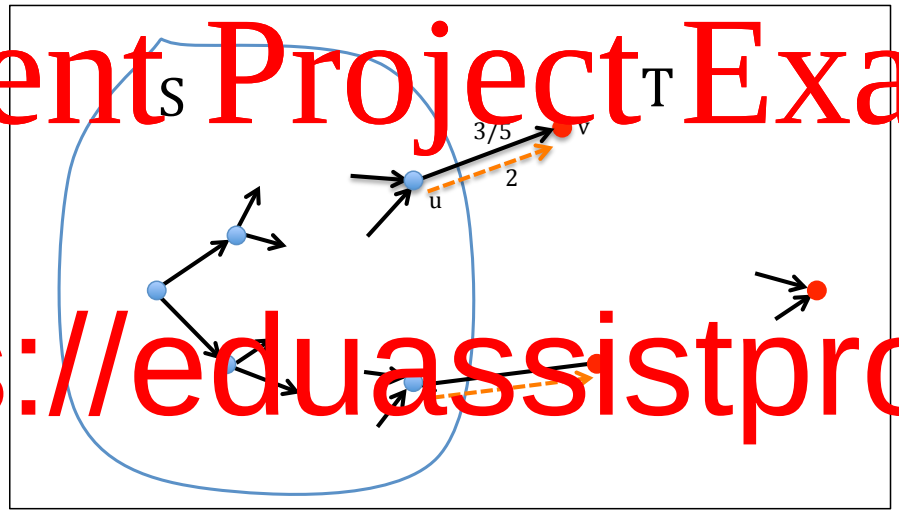
Cuts in flow networks

- Assume that the Ford - Fulkerson algorithm has terminated and that there no more augmenting paths from the source s to the sink t in the last residual network flow.
- Define S to be the source s and all vertices u such that there is a path in the resi
- Defi
- Sinc belo



Cuts in flow networks

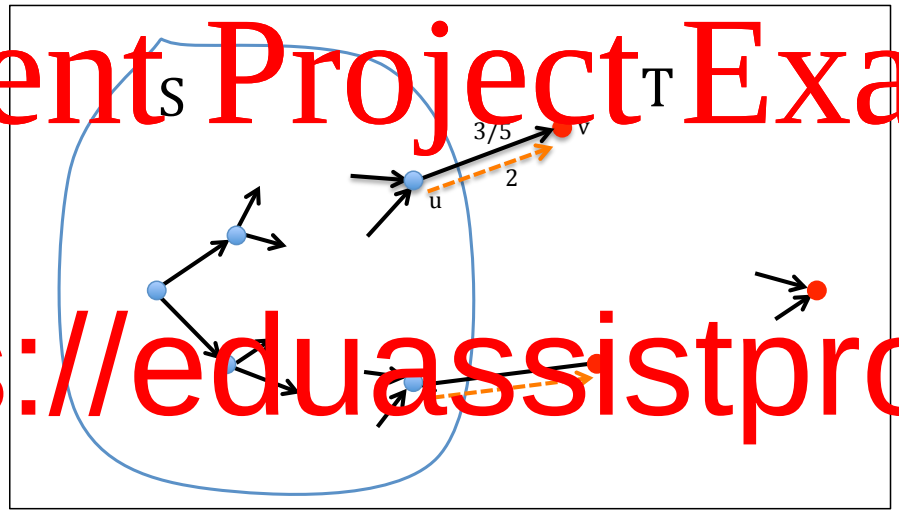
- **Claim:** all the edges from S to T are fully occupied with flow, and all the edges from T to S are empty.



- **Proof:**
  - If an edge (u, v) from S to T had so
    in the residual flow network the path from
    to a path from s to v which contra
    $v \in T$.
  - If and edge (y, x) from T to S had any flow in it, then in the residual flow network the path from s to x could be extended to a path from x to y, which is again a contradiction with our assumption that $y \in T$.

Cuts in flow networks

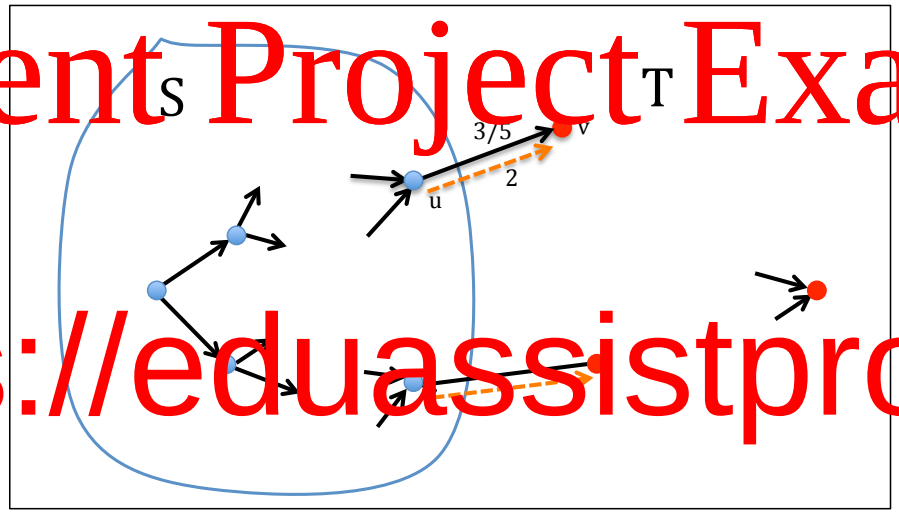
- **Claim:** all the edges from S to T are fully occupied with flow, and all the edges from T to S are empty.



- **Proof:**
  - If an edge (u, v) from S to T had so[illegible] in the residual flow network the path from [illegible] to a path from s to v which contra[illegible] $v \in T$.
  - If and edge (y, x) from T to S had any flow in it, then in the residual flow network the path from s to x could be extended to a path from x to y, which is again a contradiction with our assumption that $y \in T$.

Cuts in flow networks

- **Claim:** all the edges from S to T are fully occupied with flow, and all the edges from T to S are empty.

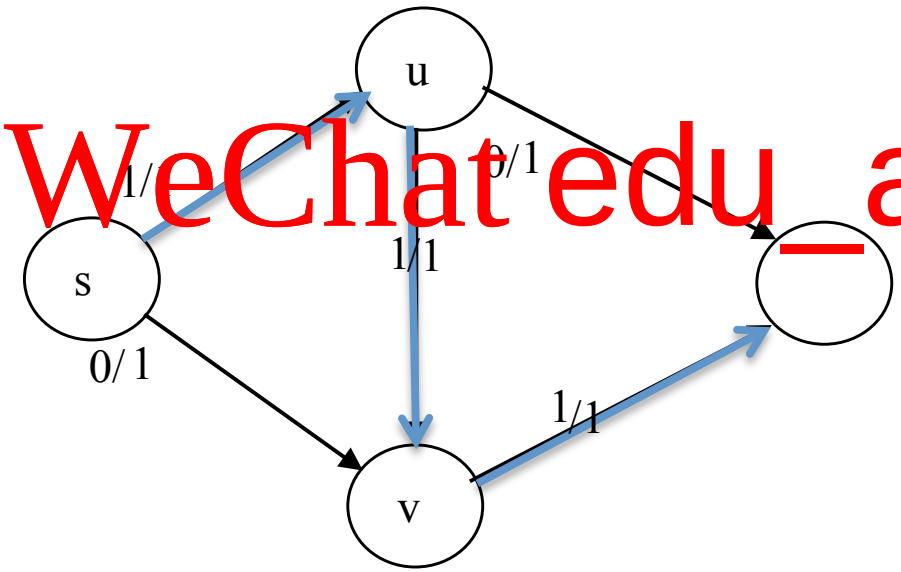


- **Proof:**
  - If an edge (u, v) from S to T had so[illegible] in the residual flow network the path from [illegible] to a path from s to v which contra[illegible] $v \in T$.

  - If and edge (y, x) from T to S had any flow in it, then in the residual flow network the path from s to x could be extended to a path from x to y, which is again a contradiction with our assumption that $y \in T$.

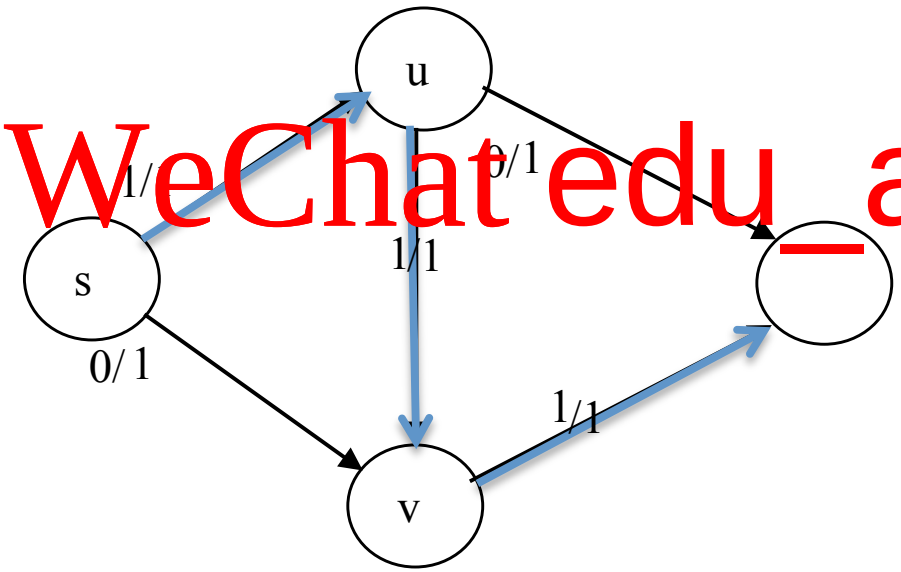
Cuts in flow networks

- Since all edges from S to T are occupied with flows to their full capacity and since there is no flow from T to S, the flow across the cut (S, T) is precisely equal to the capacity of this cut.
- Thus, such a flow is maximal and the corresponding cut is a minimal cut, regardless of the particular way how the augmenting paths were chosen.
- Tryi
mak
that



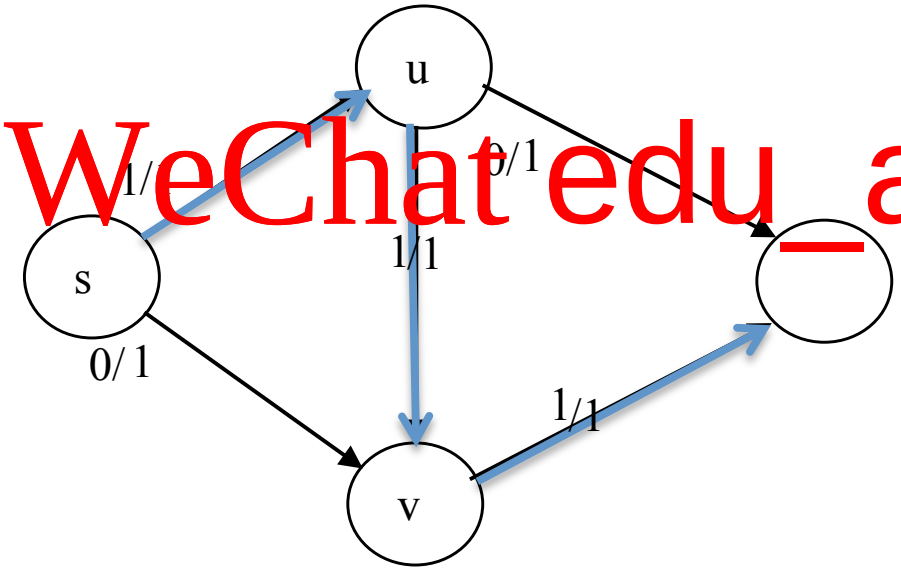
Cuts in flow networks

- Since all edges from S to T are occupied with flows to their full capacity and since there is no flow from T to S, the flow across the cut (S, T) is precisely equal to the capacity of this cut.
- Thus, such a flow is maximal and the corresponding cut is a minimal cut, regardless of the particular way how the augmenting paths were chosen.
- Tryi
mak
that



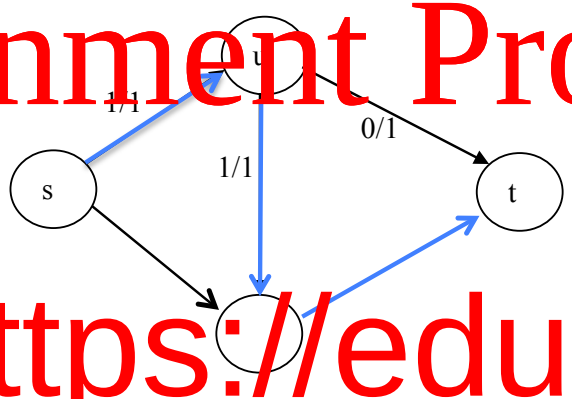
Cuts in flow networks

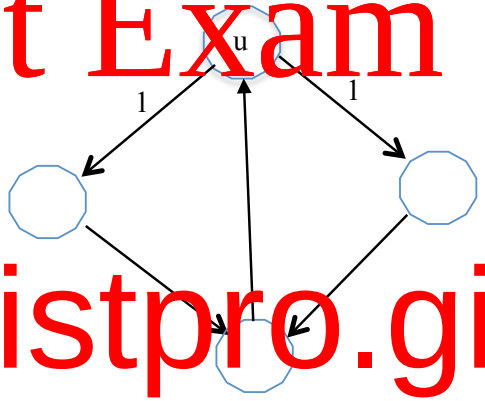
- Since all edges from S to T are occupied with flows to their full capacity and since there is no flow from T to S, the flow across the cut (S, T) is precisely equal to the capacity of this cut.
- Thus, such a flow is maximal and the corresponding cut is a minimal cut, regardless of the particular way how the augmenting paths were chosen.
- Tryi
mak
that



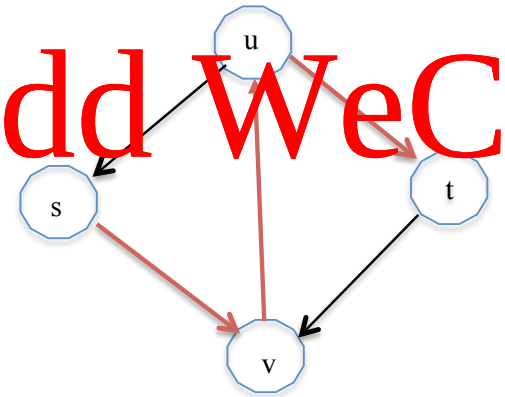
Cuts in flow networks

- But such a path is obvious in the residual network flow:

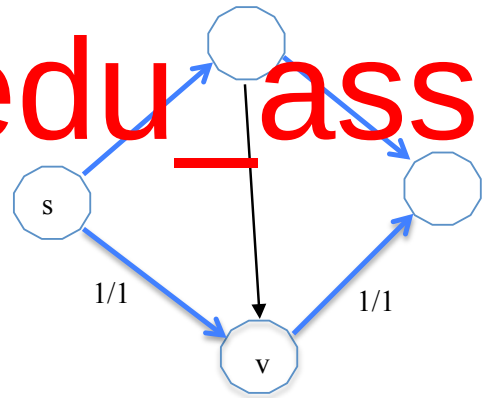




Residual network



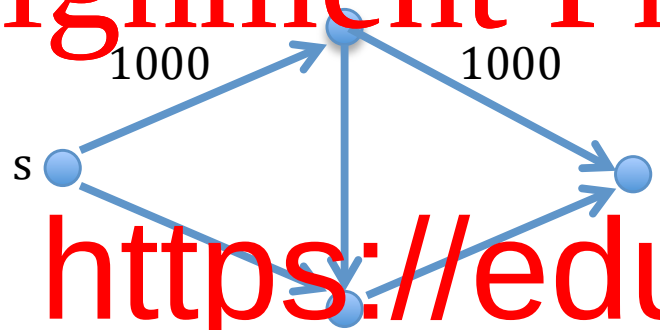
Augmenting path in the residual network

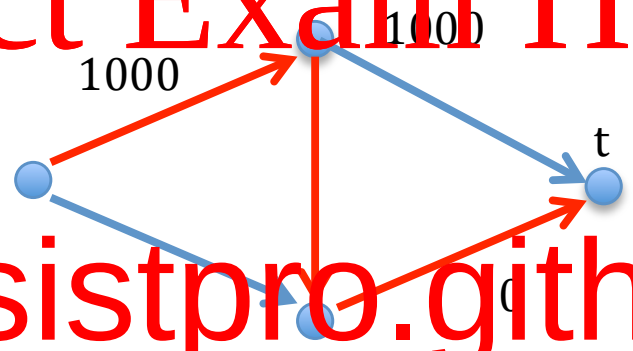


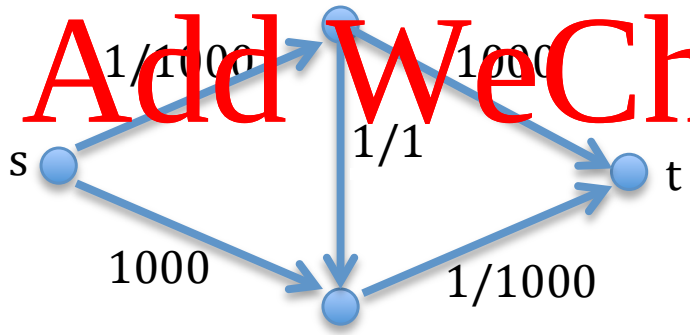
Final flow

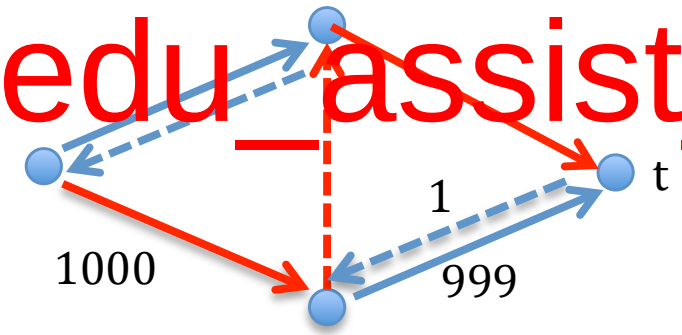
Cuts in flow networks

- How efficient is the Ford-Fulkerson algorithm?

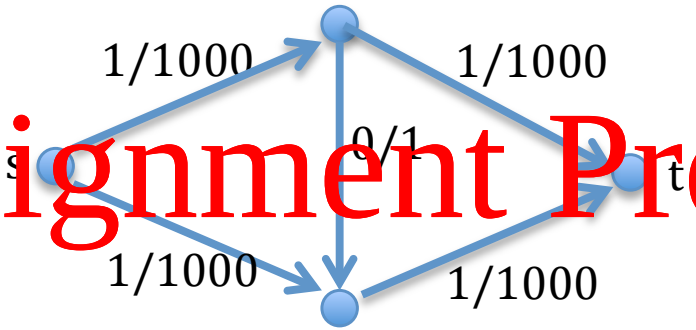


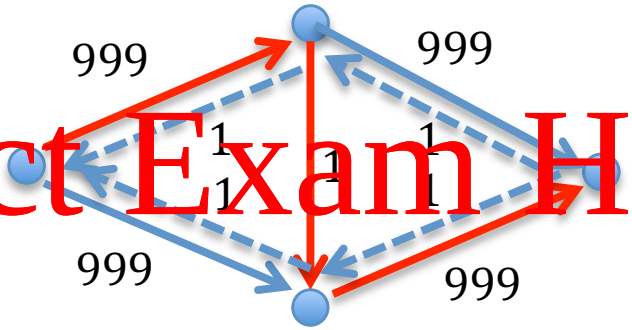


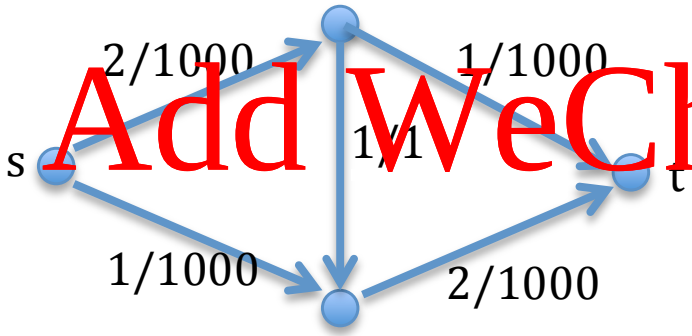




Cuts in flow networks







- The Ford-Fulkerson algorithm can potentially run in time proportional to the value of max flow, which can be exponential in the size of the input.

Edmonds-Karp Max Flow Algorithm

- The Edmonds-Karp algorithm improves the Ford Fulkerson algorithm in a simple way: always choose the shortest path from the source s to the sink t, where the "shortest path" means the fewest number of edges, regardless of their capacities (i.e., each edge has the same unit weight).
- Not with alon
- Why does such a choice speed up the Ford - Fulkerson alg one needs a tricky mathematical proof, see the textboo such algorithm runs in time $O(|V| \cdot |E|^2)$.
- The fastest max flow algorithm to date, an extension of the Preflow-Push algorithm runs in time $|V|^3$.

Edmonds-Karp Max Flow Algorithm

- The Edmonds-Karp algorithm improves the Ford Fulkerson algorithm in a simple way: always choose the shortest path from the source s to the sink t, where the "shortest path" means the fewest number of edges, regardless of their capacities (i.e., each edge has the same unit weight).

- Not with s alon

- Why does such a choice speed up the Ford - Fulkerson alg one needs a tricky mathematical proof, see the textboo such algorithm runs in time $O(|V| \cdot |E|^2)$.

- The fastest max flow algorithm to date, an extension of the Preflow-Push algorithm runs in time $|V|^3$.

Edmonds-Karp Max Flow Algorithm

- The Edmonds-Karp algorithm improves the Ford Fulkerson algorithm in a simple way: always choose the shortest path from the source s to the sink t, where the "shortest path" means the fewest number of edges, regardless of their capacities (i.e., each edge has the same unit weight).

- Not
with s
alon

- Why does such a choice speed up the Ford - Fulkerson alg
one needs a tricky mathematical proof, see the textboo
such algorithm runs in time $O(|V||E|^2)$.

- The fastest max flow algorithm to date, an extension of the PREFLOW-PUSH algorithm runs in time $|V|^3$.

Edmonds-Karp Max Flow Algorithm

- The Edmonds-Karp algorithm improves the Ford Fulkerson algorithm in a simple way: always choose the shortest path from the source s to the sink t, where the "shortest path" means the fewest number of edges, regardless of their capacities (i.e., each edge has the same unit weight).

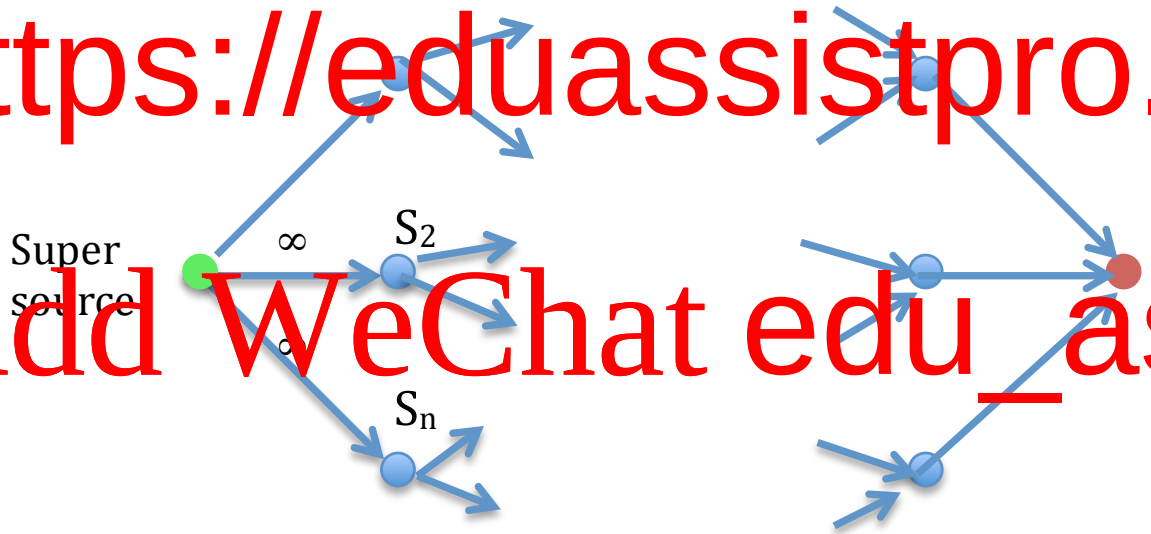
- Not
with s
alon

- Why does such a choice speed up the Ford - Fulkerson alg
one needs a tricky mathematical proof, see the textboo
such algorithm runs in time $O(|V||E|^2)$.

- The fastest max flow algorithm to date, an extension of the PREFLOW-PUSH algorithm runs in time $|V|^3$.

Networks with multiple sources and sinks

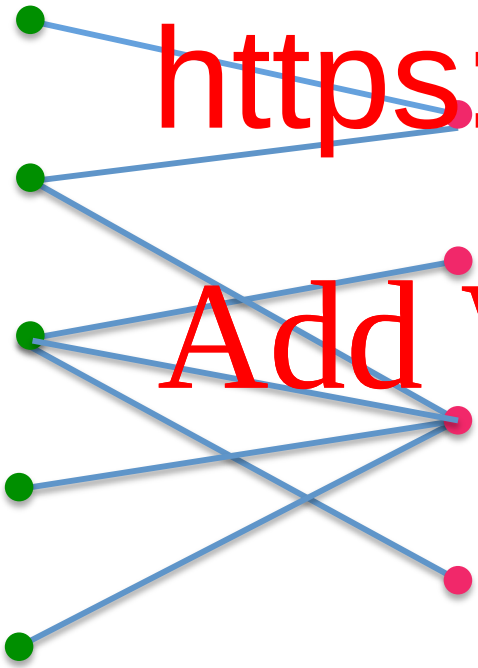
- Flow networks with multiple sources and sinks are reducible to networks with a single source and single sink by adding a "super-sink" and "super-source" and connecting them to all sources and sinks, respectively, by edges of infinite capacities.



Maximum matching in bipartite graphs

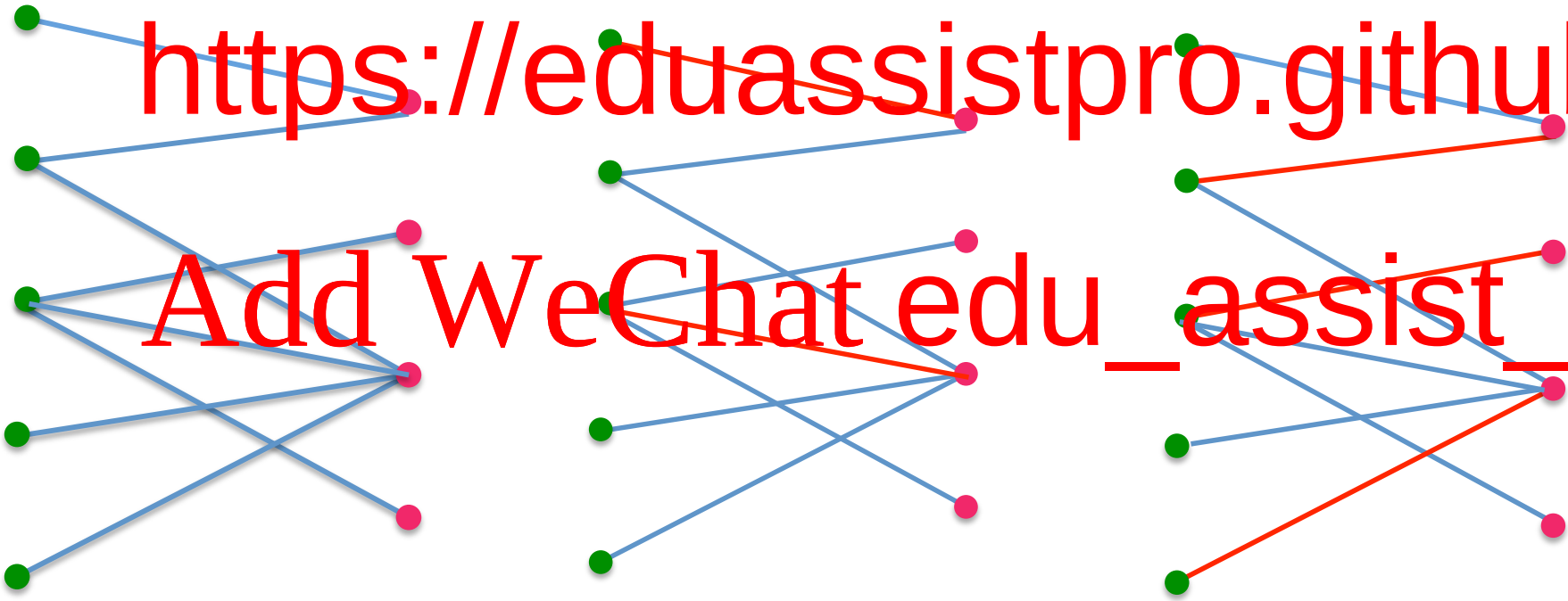
- We will consider bipartite graphs; i.e., graphs whose vertices can be split into two subsets, L and R such that every edge $e \in E$ has one end in the set L and the other in the set R.

- A matching in a graph G is a subset M of all edges E such that each vertex of the graph belongs to at most one of the edges in the matching M.



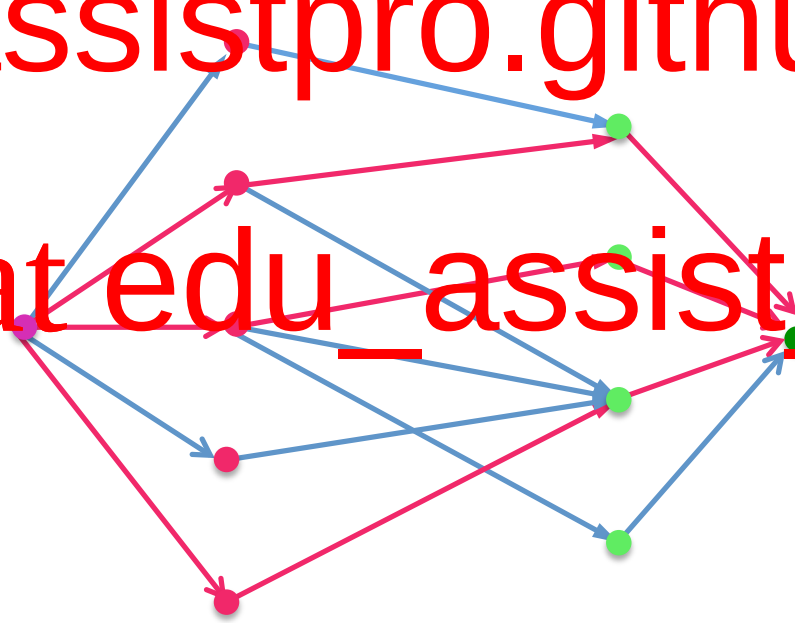
Maximum matching in bipartite graphs

- We will consider bipartite graphs; i.e., graphs whose vertices can be split into two subsets, L and R such that every edge $e \in E$ has one end in the set L and the other in the set R.
- A matching in a graph G is a subset M of all edges E such that each vertex of the graph belongs to at most one of the edges in the matching M.



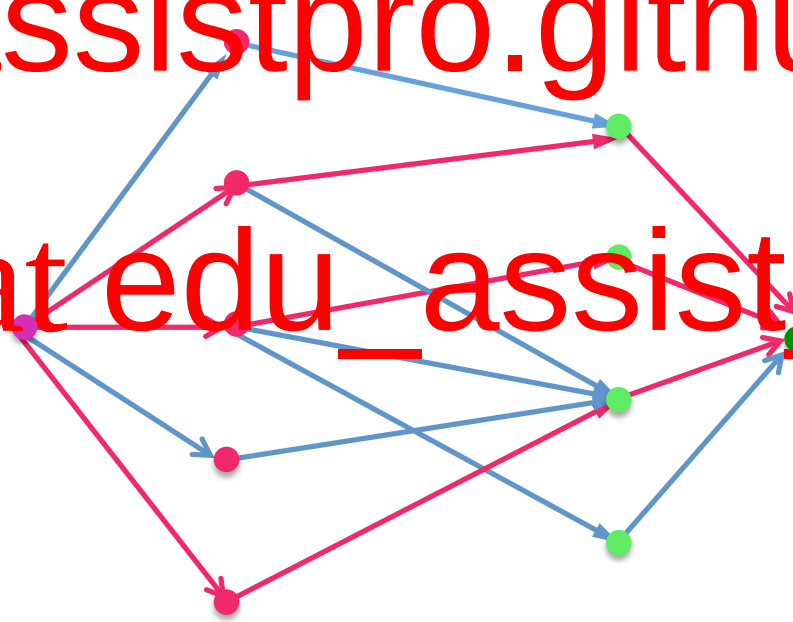
Maximum matching in bipartite graphs

- A maximum matching in a bipartite graph G is a matching containing the largest possible number of edges.

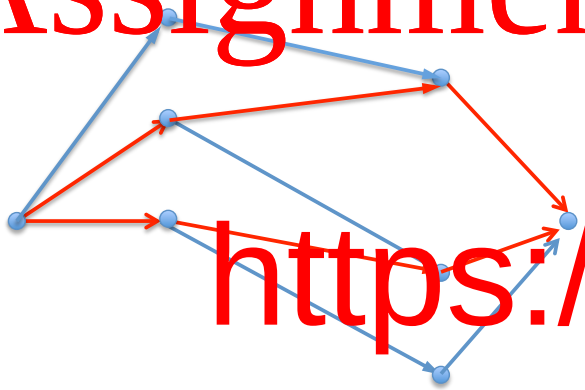


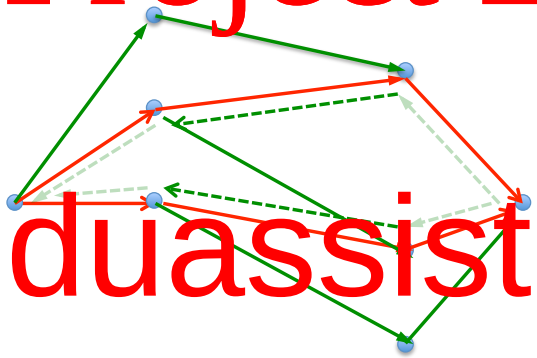
Maximum matching in bipartite graphs

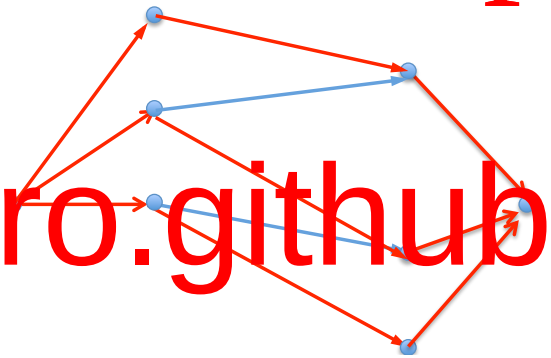
- A maximum matching in a bipartite graph G is a matching containing the largest possible number of edges.
- We turn a Maximum Matching problem into a Max Flow problem by adding a super source and a super sink, and by giving all edges a capacity of 1



Maximum matching in bipartite graphs







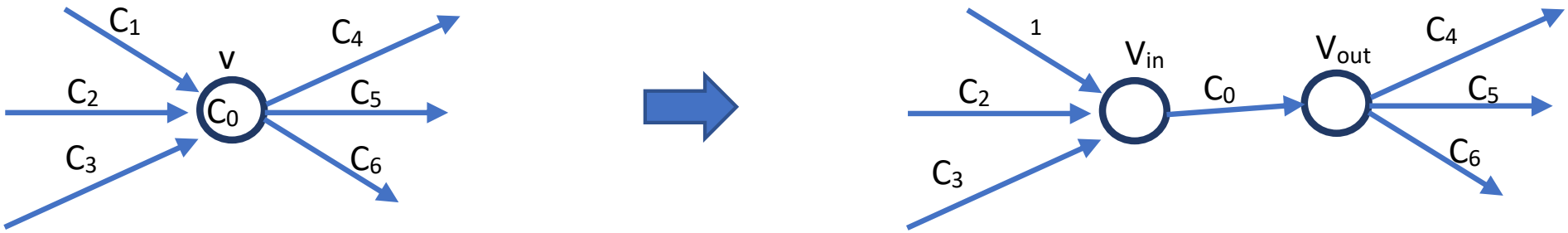
- Note how the residual flow network allows reroute increase the total throughput.

Max Flow with vertex capacities

- Sometimes not only the edges but also the vertices v_i of the flow graph might have capacities $C(v_i)$, which limit the total throughput of the flow coming to the vertex (and, consequently, also leaving the vertex):

$$\sum_{e(u,v) \in E} f(u,v) = \sum_{e(v,w) \in E} f(v,w) \leq C(v)$$

- Suc
each
all edges coming into v go into v_{in}, all edges leaving v now leave v_{out} and by
connecting the new vertices v_{in} and v_{out} ith
capacity equal to the capacity of the original vertex

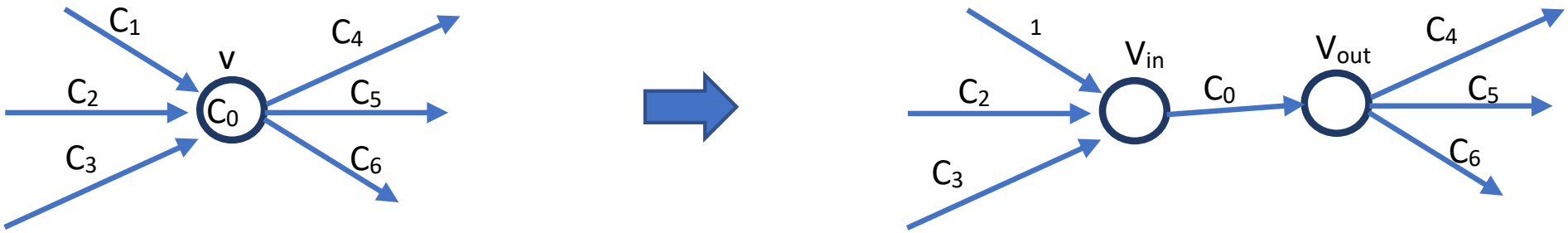


Max Flow with vertex capacities

- Sometimes not only the edges but also the vertices v_i of the flow graph might have capacities $C(v_i)$, which limit the total throughput of the flow coming to the vertex (and, consequently, also leaving the vertex):

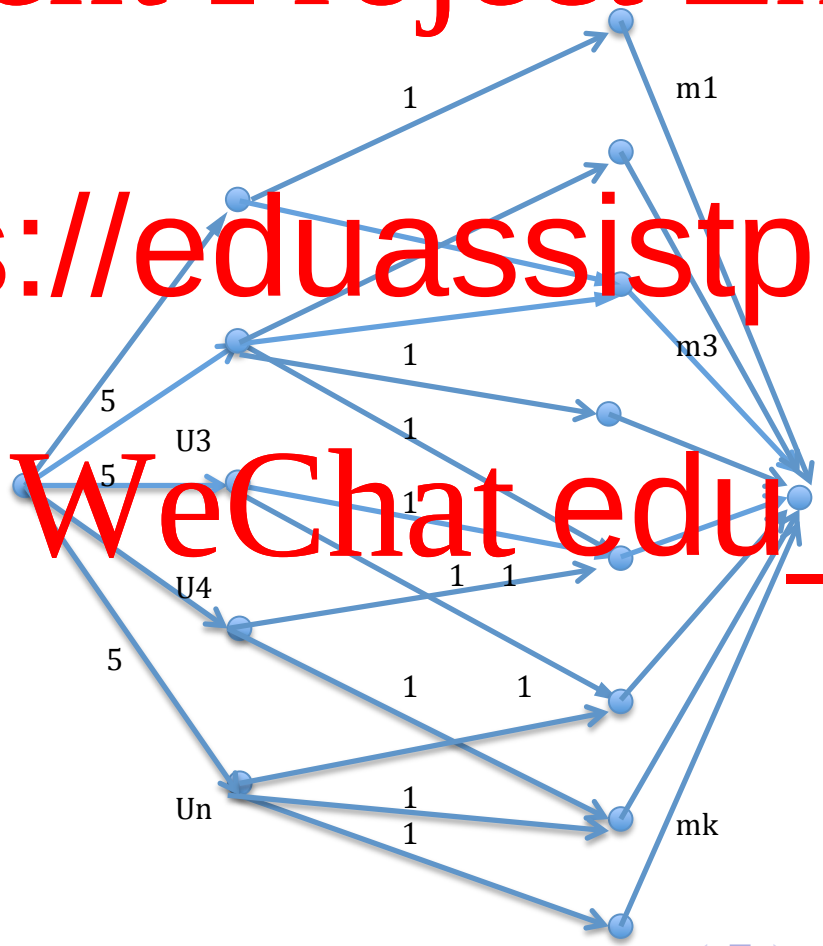
$$\sum_{e(u,v) \in E} f(u,v) = \sum_{e(v,w) \in E} f(v,w) \leq C(v)$$

- Suc
each d v_{out} so that all edges coming into v go into v_{in}, all edges leaving v now leave v_{out} and by connecting the new vertices v_{in} and v_{out} ith capacity equal to the capacity of the original vertex



Applications of Max Flow algorithm

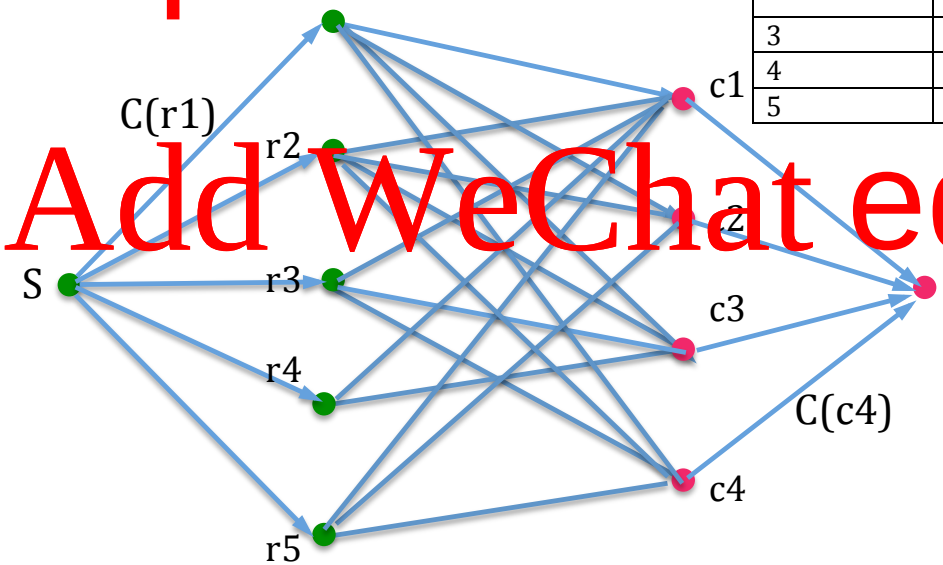
- Assume you have a movie rental agency. At the moment you have k movies in stock, with m_i copies of movie i. Each of n customers can rent out at most 5 movies at a time. The customers have sent you their preferences which are lists of movies they would like to see. Your goal is to dispatch the largest possible number of movies.



Applications of Max Flow algorithm

- The storage space of a ship is in the form of a rectangular grid of cells with n rows and m columns. Some of the cells are taken by support pillars and cannot be used for storage, so they have 0 capacity. You are given the capacity of every cell; cell in row r_i and column c_j has capacity $C(i, j)$. To ensure the stability of the ship, the total weight in each row r_i must not exceed $C(r_i)$ and the total weight in each column c_j must not exceed $C(c_j)$. Find how to allocate the cargo weight to each cell to maximise to total load without exce

3	C(3,1)	C(3,2)	0	C(3,4)
4	C(4,1)	0	C(4,3)	0
5				



r3	C(r3)
r4	C(r4)
r5	C(r5)

column	capacity
c1	C(c1)
c2	C(c2)
c3	C(c3)
c4	C(c4)

Applications of Max Flow algorithm

- You are given a connected, directed graph G with N vertices. Out of these N vertices k are painted red, m are painted blue, and the remaining $N - k - m > 0$ of the vertices are black. Red vertices have only outgoing edges and blue vertices have only incoming edges. Your task is to determine the largest possible number of disjoint (i.e., non-intersecting) paths in this graph, each of which starts at a red vertex and fi

PUZZLE!!

You are taking two kinds of medicines, A and B; pills of A are completely indistinguishable from pills of B. You take one of each every day

both bott

floor but y

tell which ones (if any) are of type A an w

can you solve the problem of continuing to take 1 of eac

without throwing away any pills?