



# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat `edu_assist_pro`  
School of Computer Science and Engineering  
University of New South Wales

## 8. MAXIMUM FLOW

- 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.

<https://eduassistpro.github.io>

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

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

- 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.

<https://eduassistpro.github.io>

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

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

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

# Assignment Project Exam Help

- Exa

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

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

- 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

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

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

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

Assignment Project Exam Help

Clearly, also  $|f| = \sum_{v:(v,t) \in E} f(v, t)$ .

- Exa

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

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

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

Assignment Project Exam Help

- Clearly, also  $|f| = \sum_{v:(v,t) \in E} f(v, t)$ .

- Exa

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

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

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

Assignment Project Exam Help

- Clearly, also  $|f| = \sum_{v:(v,t) \in E} f(v, t)$ .
- Exa

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

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

- 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.

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- Each edge of the original network has a leftover capacity equal to the capacity of the edge minus the flow through the edge.
- If there is no flow through an edge, it can be removed. If 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  $v_2$ ).

Add WeChat `edu_assist_pro`

## 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.

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- Each edge of the original network has a leftover capacity equal to the capacity of the edge minus the flow through the edge.
- If there is no flow through an edge, it ends up being removed if the original 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  $v_2$ ).

- 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.

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- Each edge of the original network has a leftover capacity: the capacity of the edge minus the flow through the edge.
- If there is enough free capacity on the edge, if 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  $v_2$ ).

Add WeChat `edu_assist_pro`

- 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.

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- 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  $v_2$ ).

- 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.

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- 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  $v_2$ ).

- 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.

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- 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  $v_2$ ).

Add WeChat `edu_assist_pro`

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

# Assignment Project Exam Help

- The flow is increased by adding flow along an augmenting path, i.e., a path from s to t that does not contain any edges with positive residual capacity.
- We can now recalculate the flow through all edges along the augmenting path by adding the additional flow through the path if the flow along the path is less than the maximum available residual flow if in pp. 10. The direction of the flow is reversed.

Add WeChat edu\_assist\_pro

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

# Assignment Project Exam Help

- The  $s-t$  path is the one that starts at  $s$  and ends at  $t$ , i.e., the path from  $s$  to  $t$ .
- We can now recalculate the flow through all edges along the augmenting path by adding the additional flow through the path if the flow augmentation is in the same direction as the original flow.

Add WeChat edu\_assist\_pro

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

# Assignment Project Exam Help

- The  $s \rightarrow t$  path is called an **augmenting path**, i.e., there is a path from  $s$  to  $t$ .
- We can now recalculate the flow through all edges along the augmenting path by adding the additional flow through the path if the flow on the edges in the path is zero. If the augmenting path is in the same direction as the original path, we add the flow; if in opposite direction, we subtract it.

Add WeChat edu\_assist\_pro

# Ford Fulkerson method for finding maximal flow

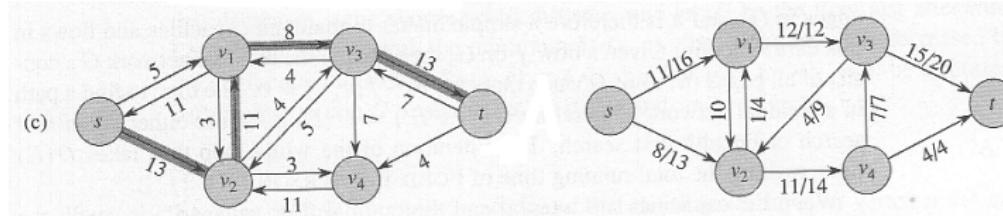
**Ford - Fulkerson algorithm for finding maximal flow in a flow network:**

- 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.

# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro



# Ford Fulkerson method for finding maximal flow

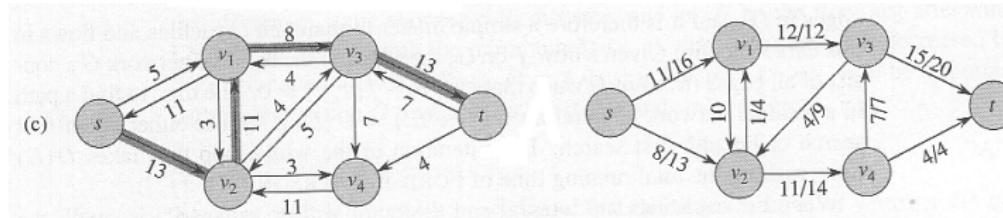
**Ford - Fulkerson algorithm for finding maximal flow in a flow network:**

- 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.

# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro



# Ford Fulkerson method for finding maximal flow

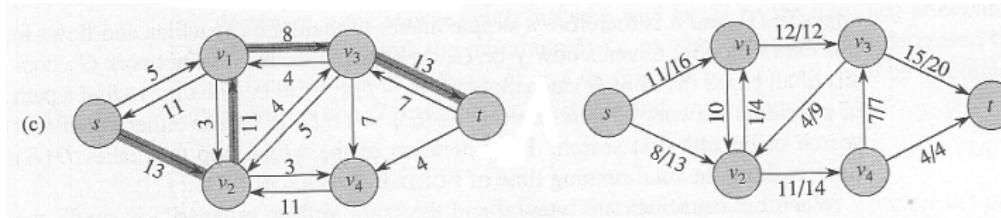
**Ford - Fulkerson algorithm for finding maximal flow in a flow network:**

- 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

# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro



# Ford Fulkerson method for finding maximal flow

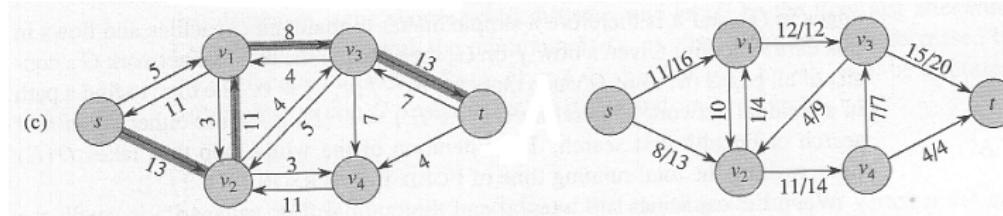
**Ford - Fulkerson algorithm for finding maximal flow in a flow network:**

- 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

# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro



# Assignment Project Exam Help

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

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.

<https://eduassistpro.github.io>

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

Add WeChat `edu_assist_pro`

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, how much flow can be sent through the network for at least 1 minute?
- the total flow cannot be larger than the sum of all capacities of all edges leaving the source, so eventually the process must terminate.

Add WeChat `edu_assist_pro`

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 path will add at least one unit of flow through the network for at least one edge,
- the total flow cannot be larger than the sum of all capacities of all edges leaving the source, so eventually the process must terminate.

Add WeChat `edu_assist_pro`

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 augmentation 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.

Add WeChat [edu\\_assist\\_pro](https://eduassistpro.github.io)

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 augmentation 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.

Add WeChat [edu\\_assist\\_pro](https://eduassistpro.github.io)

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?

## Assignment Project Exam Help

- May we have created bottlenecks by choosing bad augmenting paths, maybe better choices of augmenting paths could produce a larger total flow through the network.

- This is a matter of choice.
- The problem is that there may be many choices.

<https://eduassistpro.github.io>

- A cut in a graph is any partition of the vertices of the graph into two sets,  $S$  and  $T$ , such that

$$\textcircled{1} \quad S \cup T = V$$

$$\textcircled{2} \quad S \cap T = \emptyset$$

$$\textcircled{3} \quad s \in S \text{ and } t \in T.$$

Add WeChat `edu_assist_pro`

- 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 network.
- This is a good question.
- The problem is that we don't know how to choose the best path.

<https://eduassistpro.github.io>

- A cut in a graph is any partition of the vertices of the graph into two sets,  $S$  and  $T$ , such that

$$\textcircled{1} \quad S \cup T = V$$

$$\textcircled{2} \quad S \cap T = \emptyset$$

$$\textcircled{3} \quad s \in S \text{ and } t \in T.$$

Add WeChat `edu_assist_pro`

- 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 network.
- This is a matter of project management.
- The problem is that the Ford-Fulkerson algorithm is not guaranteed to find the maximum flow.

<https://eduassistpro.github.io>

- A cut in a graph is any partition of the vertices of the graph into two sets,  $S$  and  $T$ , such that

$$\textcircled{1} \quad S \cup T = V$$

$$\textcircled{2} \quad S \cap T = \emptyset$$

$$\textcircled{3} \quad s \in S \text{ and } t \in T.$$

Add WeChat `edu_assist_pro`

- 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 network.
- This is a matter of project management.
- The problem is that the Ford-Fulkerson algorithm is not guaranteed to find the maximum flow.

<https://eduassistpro.github.io>

- A cut in a graph is any partition of the vertices of the graph into two sets,  $S$  and  $T$ , such that

$$\textcircled{1} \quad S \cup T = V$$

$$\textcircled{2} \quad S \cap T = \emptyset$$

$$\textcircled{3} \quad s \in S \text{ and } t \in T.$$

Add WeChat `edu_assist_pro`

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?

# Assignment Project Exam Help

- 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 is a matter of choice.
- The problem is that

- A **cut** in a flow network is any partition of the vertices of the graph into two subsets  $S$  and  $T$  such that:

$$① S \cup T = V$$

$$② S \cap T = \emptyset$$

$$③ s \in S \text{ and } t \in T.$$

Add WeChat `edu_assist_pro`

- Even if the procedure does terminate, why does it produce a flow of the largest possible value?

# Assignment Project Exam Help

- 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 is a matter of choice.
- The problem is that

- A **cut** in a flow network is any partition of the vertices of the graph into two subsets  $S$  and  $T$  such that:

$$① S \cup T = V$$

$$② S \cap T = \emptyset$$

$$③ s \in S \text{ and } t \in T.$$

Add WeChat `edu_assist_pro`

- 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 is a matter of project choice.
- The problem is:

<https://eduassistpro.github.io>

- A **cut** in a flow network is any partition of the vertices of the graph into two subsets  $S$  and  $T$  such that:

$$\textcircled{1} \quad S \cup T = V$$

$$\textcircled{2} \quad S \cap T = \emptyset$$

$$\textcircled{3} \quad s \in S \text{ and } t \in T.$$

Add WeChat `edu_assist_pro`

- 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 network.
- This is a matter of project management.
- The problem is that the Ford-Fulkerson algorithm is not guaranteed to find the maximum flow.
- A **cut** in a flow network is any partition of the vertices of the graph into two subsets  $S$  and  $T$  such that:
  - ①  $S \cup T = V$
  - ②  $S \cap T = \emptyset$
  - ③  $s \in S$  and  $t \in T$ .

Add WeChat `edu_assist_project`

# 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.

Assignment Project Exam Help

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

- Not  $S$  d  $\frac{T \text{ to}}{\text{from } S}$
- The flow through a cut  $f(S, T)$  is the total flow from  $S$  to  $T$  minus the total flow through edges from  $T$

Add WeChat edu\_assist\_pro

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

- 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.

Assignment Project Exam Help

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

- Not  $S$  d

<https://eduassistpro.github.io>

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

Add WeChat edu\_assist\_p

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

- 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.

Assignment Project Exam Help

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

- Not  $S$  d <https://eduassistpro.github.io>  $T$  to

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

Add WeChat edu\_assist\_pro

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

- 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.

Assignment Project Exam Help

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

- Not  $S$  d <https://eduassistpro.github.io>  $T$  to  $T$
- The **flow through a cut**  $f(S, T)$  is the total fl minus the total flow through edges from  $T$  to  $S$

Add WeChat edu\_assist\_pro

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

- 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.

Assignment Project Exam Help

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

- Not  $S$  d <https://eduassistpro.github.io>  $T$  to  $T$
- The **flow through a cut**  $f(S, T)$  is the total fl minus the total flow through edges from  $T$  to  $S$

Add WeChat edu\_assist\_pro

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

- 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  $f(u, v) \geq 0$ .

- Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- In the

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

Add WeChat `edu_assist_pro`

- Note that the flow in the opposite direction (from T to S) is 0
- 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.

- Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- In the

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

Add WeChat edu\_assist\_pro

- Note that the flow in the opposite direction (from T to S) is 0
- 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.

- Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- In the

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

- Add WeChat [edu\\_assist\\_pro](https://eduassistpro.github.io)
- Note that the flow in the opposite direction (from T to S) is 8
  - 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.

- Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- In the

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

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

- Note that the flow in the opposite direction (from T to S) is 8
- 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.

- Example:

# Assignment Project Exam Help

<https://eduassistpro.github.io>

- In the

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

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

- Note that the flow in the opposite direction (from T to S) is 8
- 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.

- **Theorem:** The maximal amount of flow in a flow network is equal to the capacity of the cut of minimal capacity.

## Assignment Project Exam Help

- Since any flow has to cross every cut, any flow must be smaller than the capacity of the cut of minimal capacity.
- Thus, if there is no flow from  $S$  to  $T$ , then there is no such flow.

<https://eduassistpro.github.io>

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

- **Theorem:** The maximal amount of flow in a flow network is equal to the capacity of the cut of minimal capacity.

## Assignment Project Exam Help

- Since any flow has to cross every cut, any flow must be smaller than the capacity of the cut of minimal capacity.
- Thus, if we can find such a cut  $(S, T)$ , then

<https://eduassistpro.github.io>



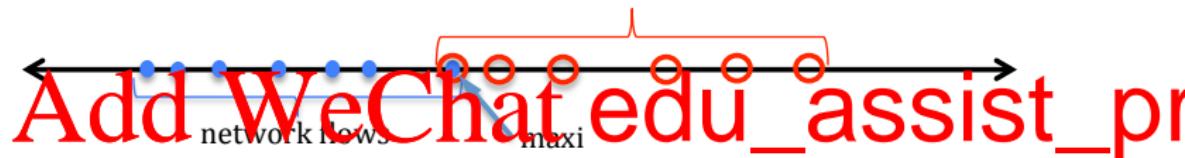
- We now show that when the Ford - Fulkerson algorithm terminates, it produces a flow equal to the capacity of an appropriately defined cut.

- **Theorem:** The maximal amount of flow in a flow network is equal to the capacity of the cut of minimal capacity.

# Assignment Project Exam Help

- Since any flow has to cross every cut, any flow must be smaller than the capacity of the cut of minimal capacity.
- Thus, if we can find a cut  $(S, T)$ , then such

<https://eduassistpro.github.io>



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

- **Theorem:** The maximal amount of flow in a flow network is equal to the capacity of the cut of minimal capacity.

# Assignment Project Exam Help

- Since any flow has to cross every cut, any flow must be smaller than the capacity of the cut of minimal capacity.
- Thus, if we can find a cut  $(S, T)$ , then such a flow must be at most the capacity of this cut.

<https://eduassistpro.github.io>

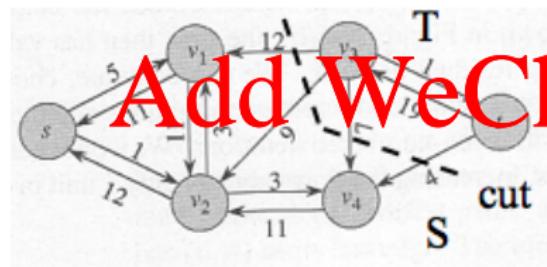
- We now show that when the Ford - Fulkerson algorithm terminates, it produces a flow equal to the capacity of an appropriately defined cut.

- 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.

# Assignment Project Exam Help

- Define  $S$  to be the source  $s$  and all vertices  $u$  such that there is a path in the resi
- Defi
- Sinc  
belo

<https://eduassistpro.github.io>



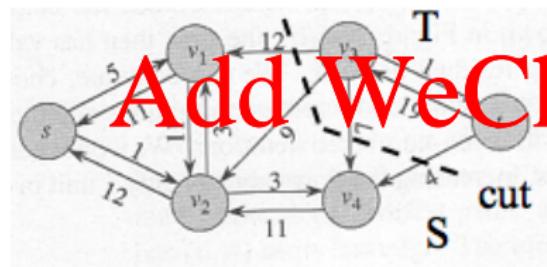
Add WeChat edu\_assist\_pro

- 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.

# Assignment Project Exam Help

- Define  $S$  to be the source  $s$  and all vertices  $u$  such that there is a path in the resi
- Def
- Since belo

<https://eduassistpro.github.io>



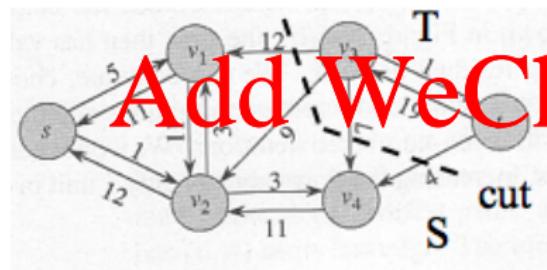
Add WeChat edu\_assist\_pro

- 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.

# Assignment Project Exam Help

- Define  $S$  to be the source  $s$  and all vertices  $u$  such that there is a path in the resi
- Defi
- Since belo

<https://eduassistpro.github.io>

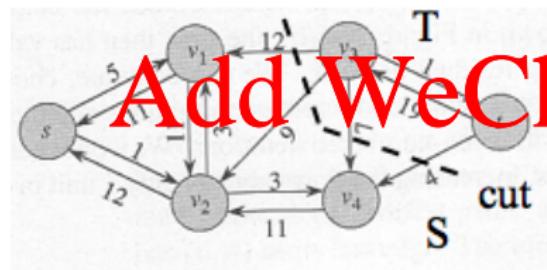


- Assume that the Ford - Fulkerson algorithm has terminated and that there no more augmenting paths from the source  $s$  to the sink  $t$  in the last residual network flow.

# Assignment Project Exam Help

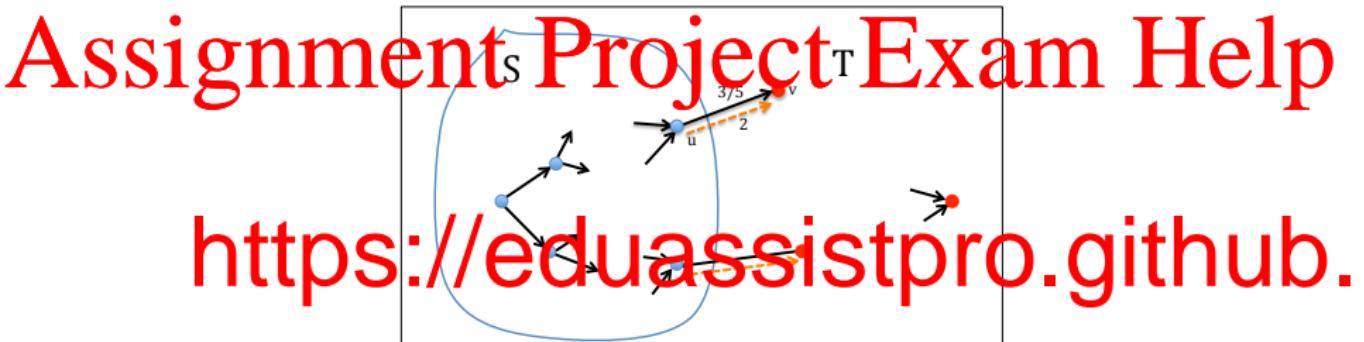
- Define  $S$  to be the source  $s$  and all vertices  $u$  such that there is a path in the resi
- Defi
- Sinc  
belo

<https://eduassistpro.github.io>



## 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.

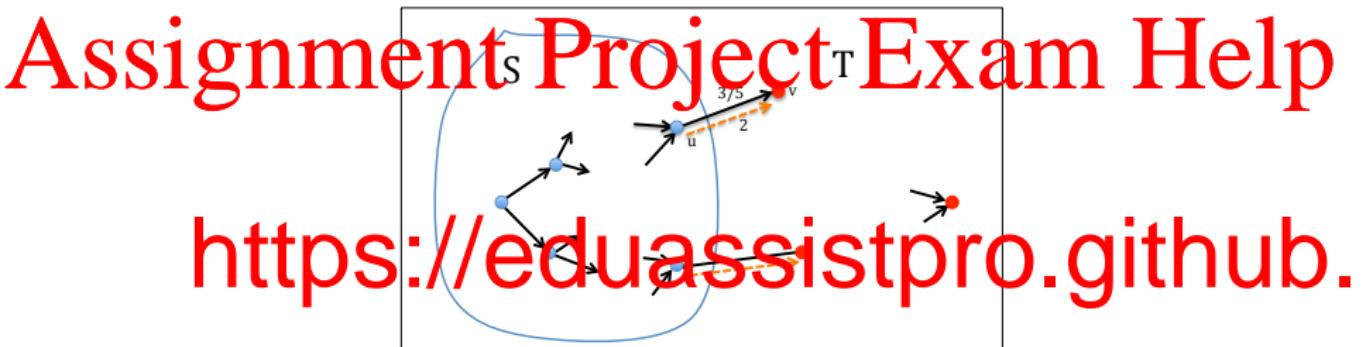


- Proof:

- If an edge  $(u, v)$  from  $S$  to  $T$  had some residual flow, then there would be a path from  $s$  to  $v$  which contradicts our assumption that  $v \in T$ .
- If an 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.

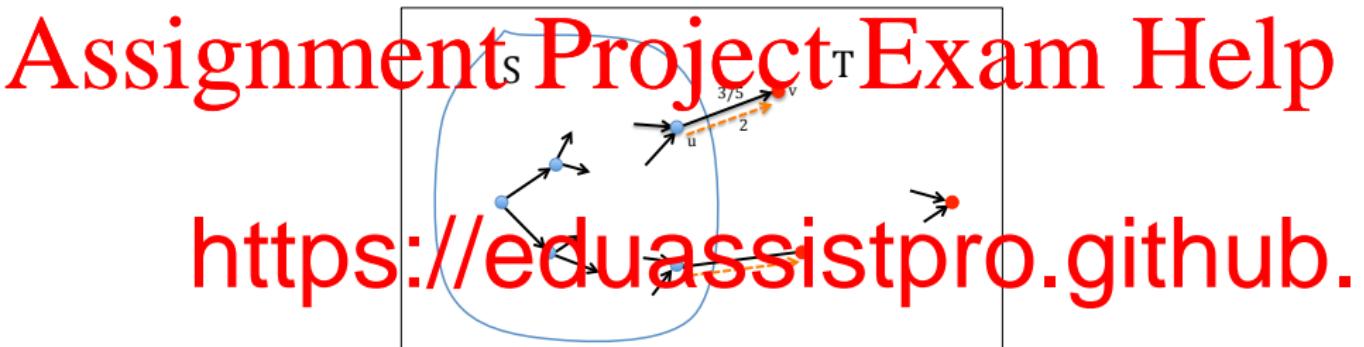


- **Proof:**

- If an edge  $(u, v)$  from  $S$  to  $T$  had some flow in the residual flow network, the path from  $s$  to  $v$  could be extended to a path from  $s$  to  $v$  which contradicts our assumption that  $v \in T$ .
- If an 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 some flow in the residual flow network, the path from  $s$  to  $v$  could be extended to a path from  $s$  to  $v$  which contradicts our assumption that  $v \in T$ .
- If an 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

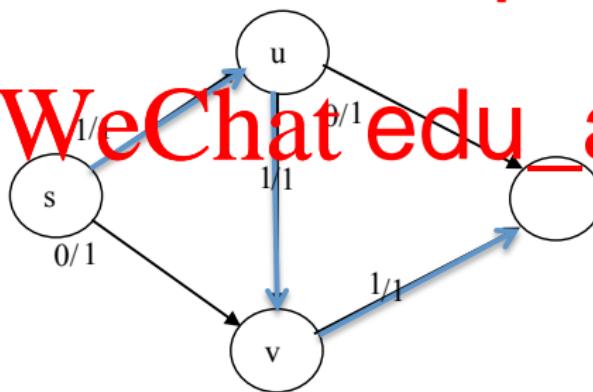
- Since all edges from  $S$  to  $T$  are occupied with flows to their full capacity and since there is no flow from  $T$  to  $S$ , the flow across the cut  $(S, T)$  is precisely equal to the capacity of this cut.

Assignment Project Exam Help  
of the particular way how the augmenting paths were chosen.

- Tryi  
mak  
that

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro



## Cuts in flow networks

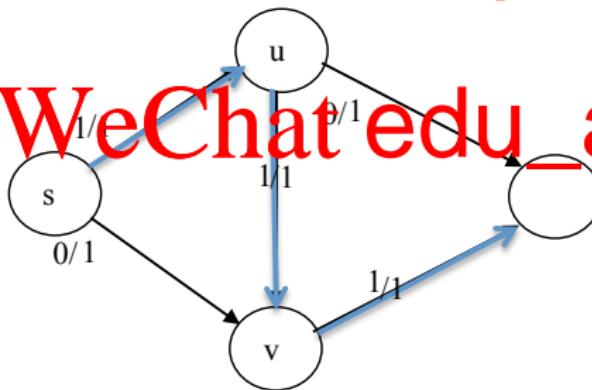
- Since all edges from  $S$  to  $T$  are occupied with flows to their full capacity and since there is no flow from  $T$  to  $S$ , the flow across the cut  $(S, T)$  is precisely equal to the capacity of this cut.

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.

- Try  
making  
that

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro



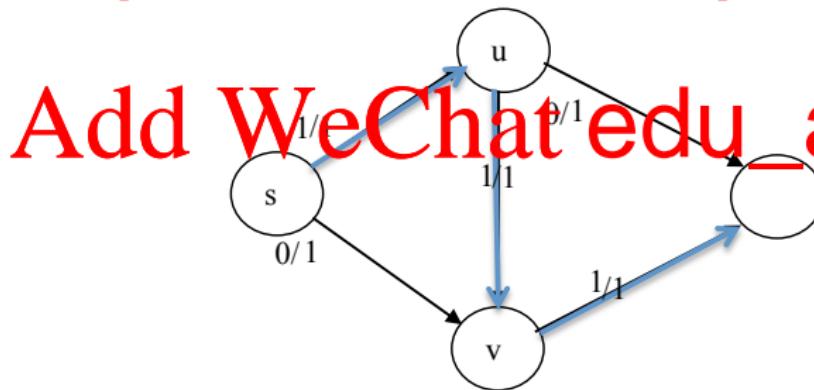
## Cuts in flow networks

- 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

<https://eduassistpro.github.io>

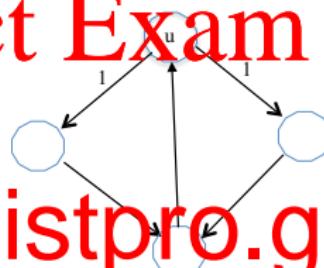
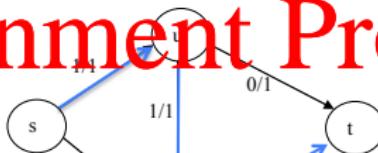


Add WeChat edu\_assist\_pro

## Cuts in flow networks

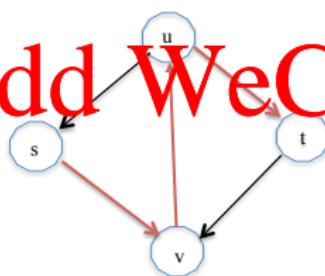
- But such a path is obvious in the residual network flow:

Assignment Project Exam Help

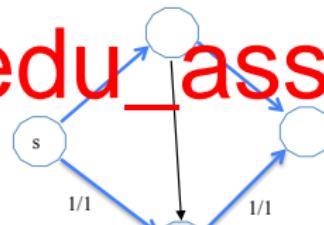


Residual network

Add WeChat edu\_assist\_pr



Augmenting path in the residual network

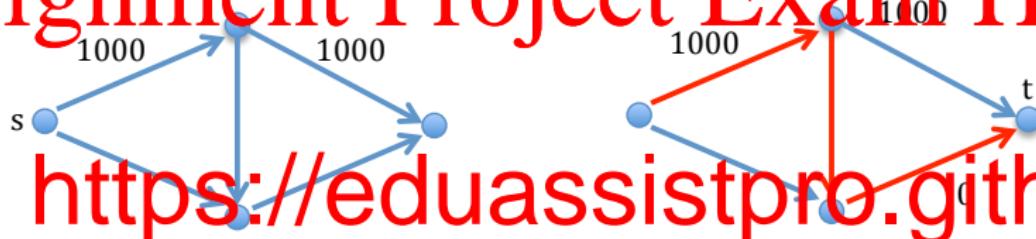


Final flow

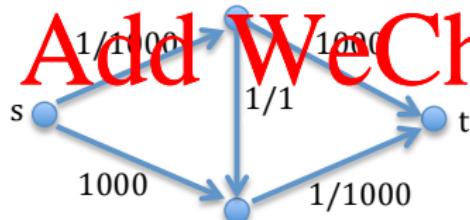
## Cuts in flow networks

- How efficient is the Ford-Fulkerson algorithm?

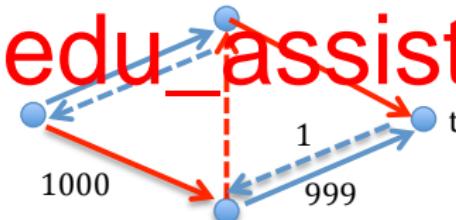
Assignment Project Exam Help



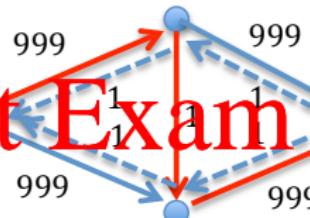
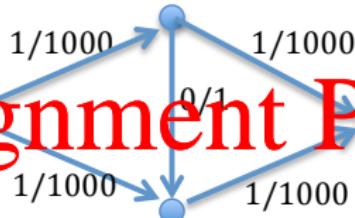
<https://eduassistpro.github.io>



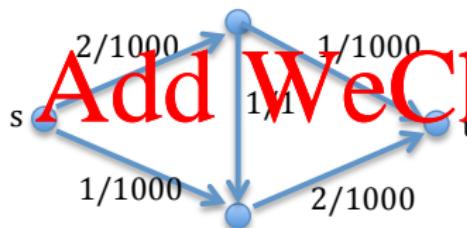
Add WeChat edu\_assist\_pr



## Cuts in flow networks



<https://eduassistpro.github.io>



- 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.

- The Edmonds-Karp algorithm improves the Ford-Fulkerson algorithm in a simple way: always choose the shortest path from the source node to the sink node, 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

<https://eduassistpro.github.io>

- Why does such a choice speed up the Ford - Fulkerson algorithm? One needs a tricky mathematical proof (see the following slide). Such a proof runs in time  $O(|V|^3)$ .
- The fastest max flow algorithm to date, an extension of the PREFLOW-PUSH algorithm runs in time  $|V|^3$ .

Add WeChat `edu_assist_pro`

- The Edmonds-Karp algorithm improves the Ford-Fulkerson algorithm in a simple way: always choose the shortest path from the source node to the sink node, 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

<https://eduassistpro.github.io>

- Why does such a choice speed up the Ford - Fulkerson algorithm? One needs a tricky mathematical proof (see the [FPTAS](#) slide). Such a proof runs in time  $O(|V|^3)$ .
- The fastest max flow algorithm to date, an extension of the PREFLOW-PUSH algorithm runs in time  $|V|^3$ .

Add WeChat `edu_assist_pro`

- 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

<https://eduassistpro.github.io>

- Why does such a choice speed up the Ford - Fulkerson alg  
one needs a tricky mathematical proof, see the textbook  
such algorithm runs in time  $\tilde{O}(|V||E|^2)$
- The fastest max flow algorithm to date, an extension of the PREFLOW-PUSH algorithm runs in time  $|V|^3$ .

Add WeChat [edu\\_assist\\_pro](https://edu_assist_pro)

- 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).

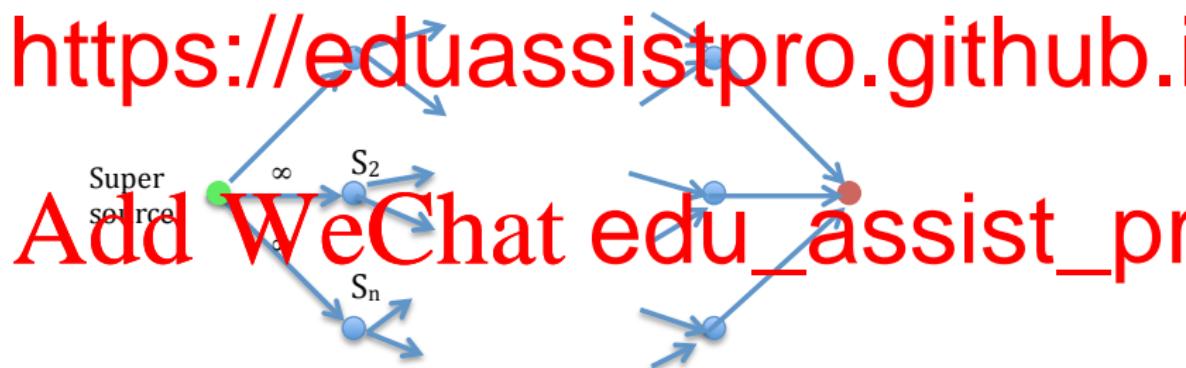
- Not  
with s  
alon

<https://eduassistpro.github.io>

- Why does such a choice speed up the Ford - Fulkerson alg  
one needs a tricky mathematical proof, see the textbook  
such algorithm runs in time  $\tilde{O}(|V||E|^2)$
- The fastest max flow algorithm to date, an extension of the PREFLOW-PUSH algorithm runs in time  $|V|^3$ .

Add WeChat `edu_assist_pro`

- 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.

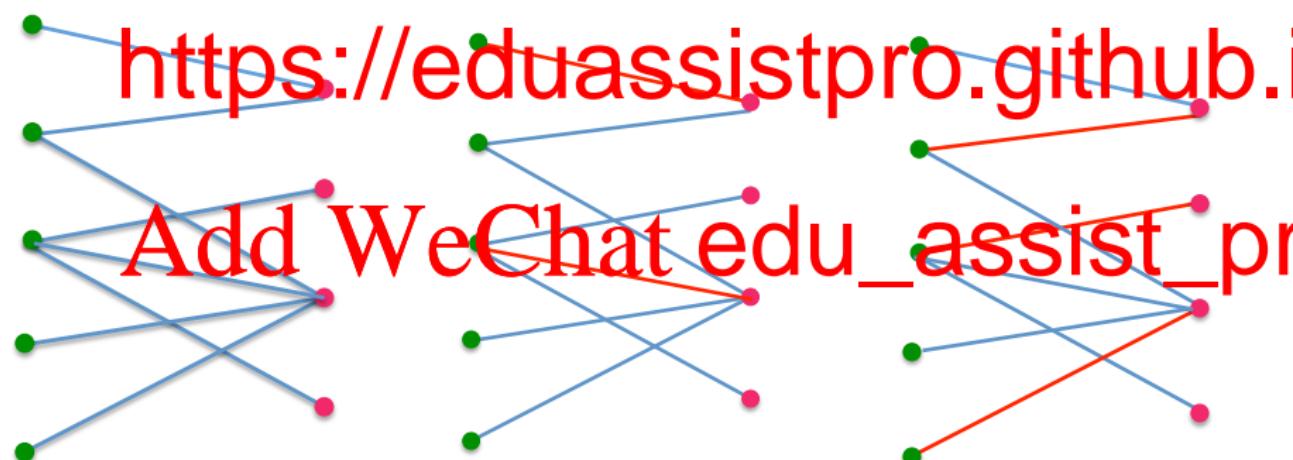


# Maximum matching in bipartite graphs

- 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  $R$ .

## Assignment Project Exam Help

graph belongs to at most one of the edges in the matching  $M$ .

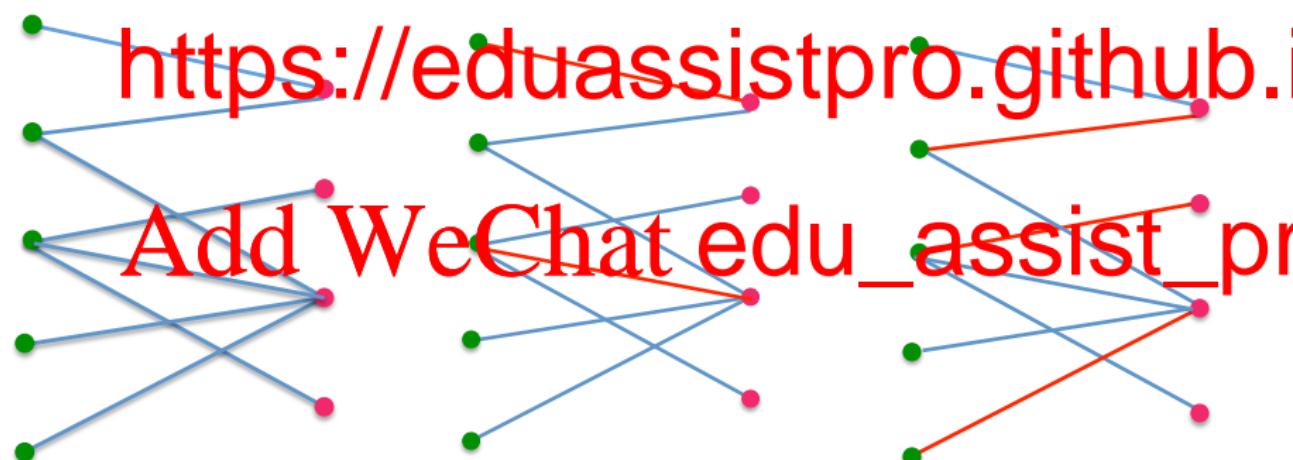


# Maximum matching in bipartite graphs

- 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  $R$ .

## Assignment Project Exam Help

- 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  $M$ .

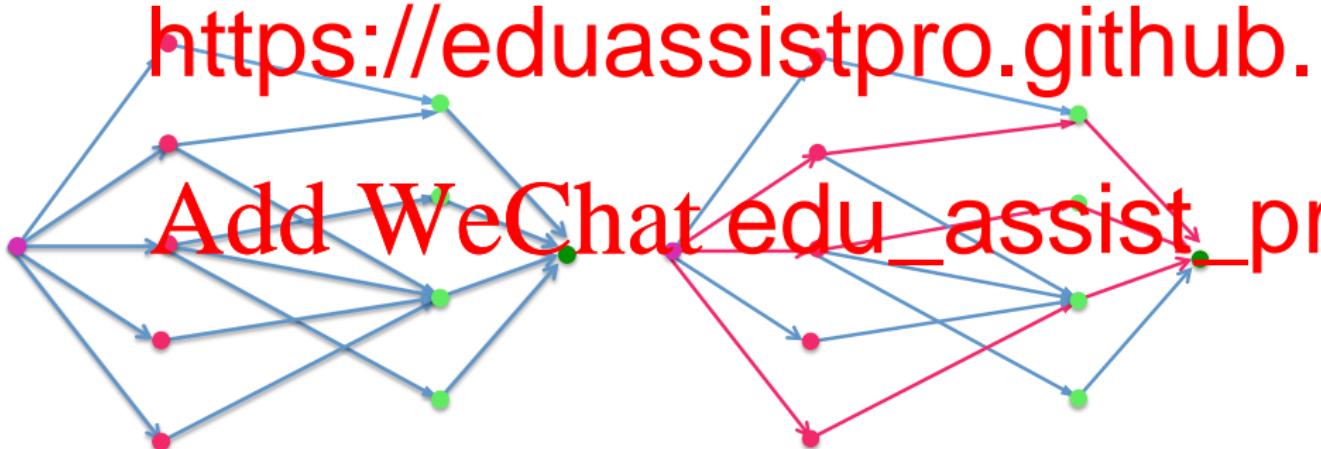


# Maximum matching in bipartite graphs

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

# Assignment Project Exam Help

<https://eduassistpro.github.io>

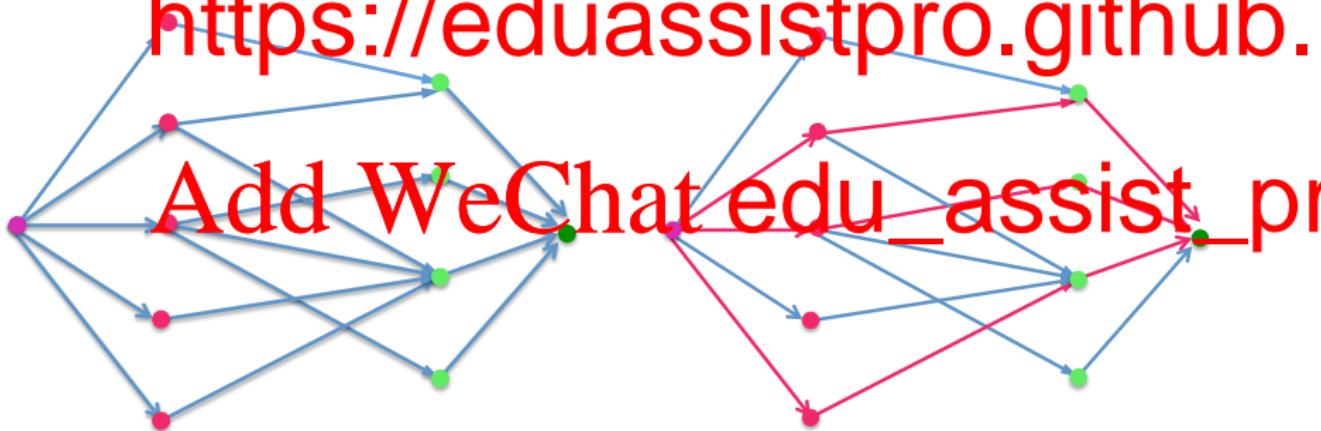


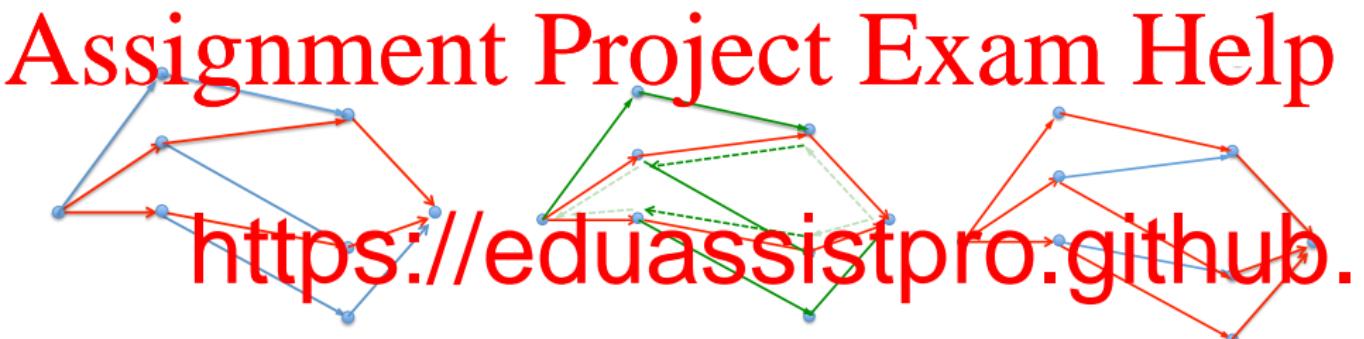
# Maximum matching in bipartite graphs

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

We turn the 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

<https://eduassistpro.github.io>





- Add WeChat `edu_assist_pr`

# Max Flow with vertex capacities

- Sometimes not only the edges  $v_i$  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):

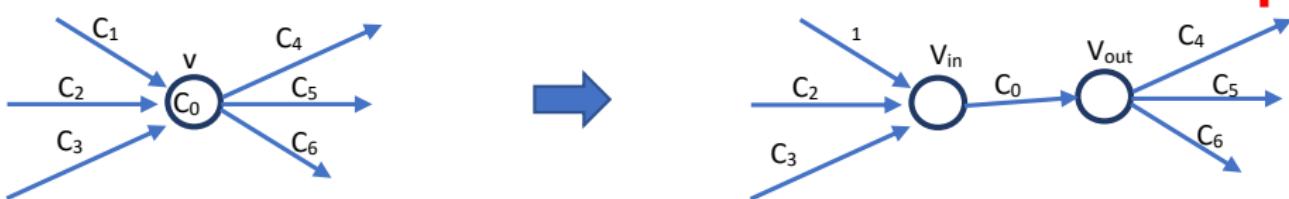
# Assignment Project Exam Help

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

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

- Such a transformation is called *vertex splitting*:  
each vertex  $v$  is split into two vertices  $v_{in}$  and  $v_{out}$  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  $v_{out}$  with an edge of capacity equal to the capacity of the original vertex  $v$ .

Add WeChat edu\_assist\_pro



# Max Flow with vertex capacities

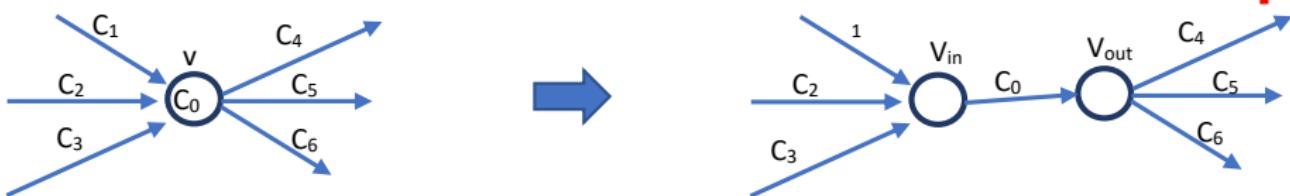
- Sometimes not only the edges  $v_i$  of the flow graph might have capacities  $C(v_i)$ , which limit the total throughput of the flow coming to the vertex (and, consequently, also leaving the vertex):

# Assignment Project Exam Help

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

- Such that each vertex  $v$  has an incoming vertex  $v_{in}$  and an outgoing vertex  $v_{out}$  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  $v_{out}$  with edges of capacity equal to the capacity of the original vertex  $v$ .

Add WeChat edu\_assist\_pro



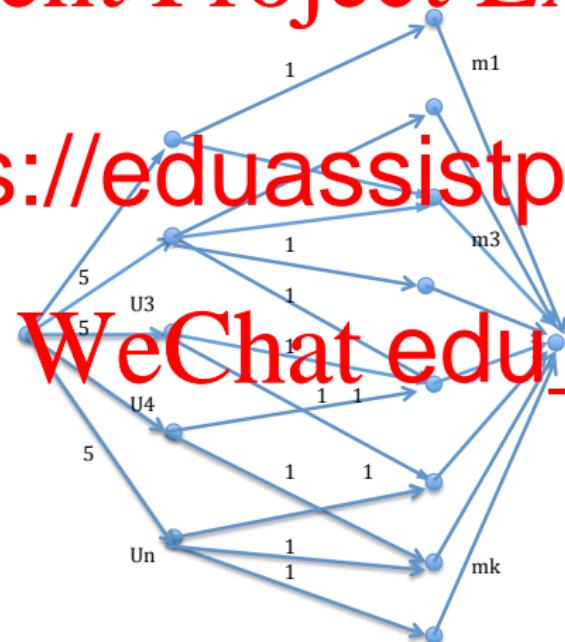
# Applications of Max Flow algorithm

- 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.

Assignment Project Exam Help

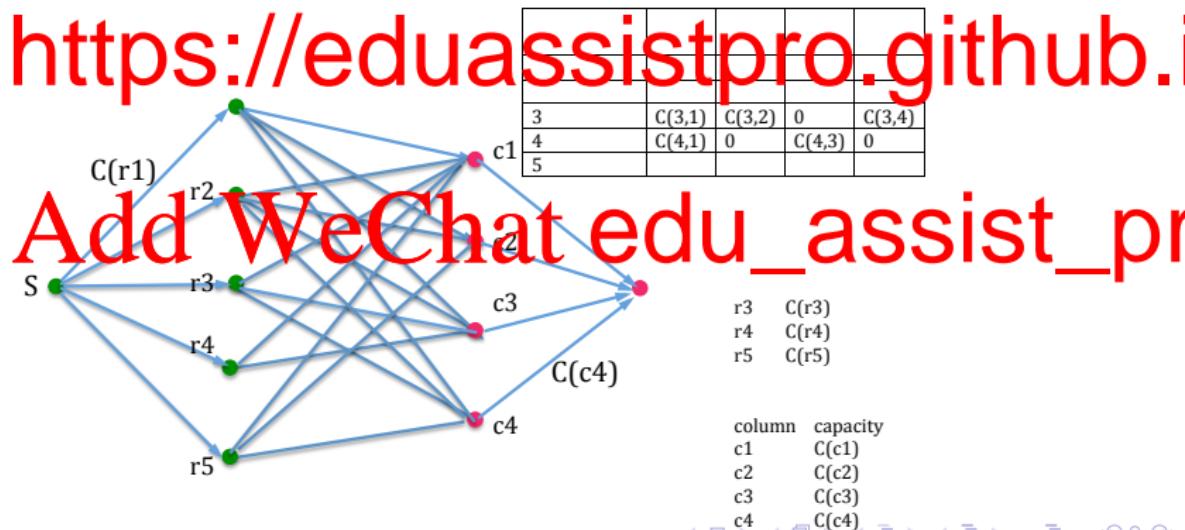
<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro



# 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 the total load without exceeding the capacities.



## 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

<https://eduassistpro.github.io>

Add WeChat edu\_assist\_pro

# PUZZLE!!

## Assignment Project Exam Help

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 bot  
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?

Add WeChat edu\_assist\_pr