Network Flow Problems

Dr. Rajiv Misra
Associate Professor
Dept. of Computer Science & Engg.
Indian Institute of Technology Patna
rajivm@iitp.ac.in

Preface

Recap of Previous Lecture:

In previous lecture, we have discussed the k-connected graphs, k-edge-connected graphs, Menger's theorem and Line graph.

Content of this Lecture:

In this lecture, we will discuss the Network Flow Problems *i.e.* Maximum Network Flow, f-augmenting path, Ford-Fulkerson labeling algorithm, Max-flow Min-cut Theorem and the Proof of Menger's Theorem using max-flow min-cut theorem.

Network and Flows

- Consider a network of pipes where valves allow flow in only one direction.
- Each pipe has a capacity per unit time. We can model this with a vertex for each junction and a (directed) edge for each pipe, weighted by the capacity. We also assume that flow cannot accumulate at a junction.
- Given two locations *s*, *t* in the network, we may ask **"what is the maximum flow (per unit time) from *s* to *t*?"**
- This questions arises in many contexts. The network may represent roads with traffic capacities, or links in a computer network with data transmission capacities, or currents in an electrical network. There are applications in industrial settings and to combinatorial min-max theorems.

Network 4.3

- A ***network*** is :
  - A digraph with a nonegative capacity $c(e)$ on each edge e and
  - A distinguished **source vertex *s*** and **sink vertex *t***.
  - Vertices are also called **node s**.

Network Flow: Definitions and constraints 4.3

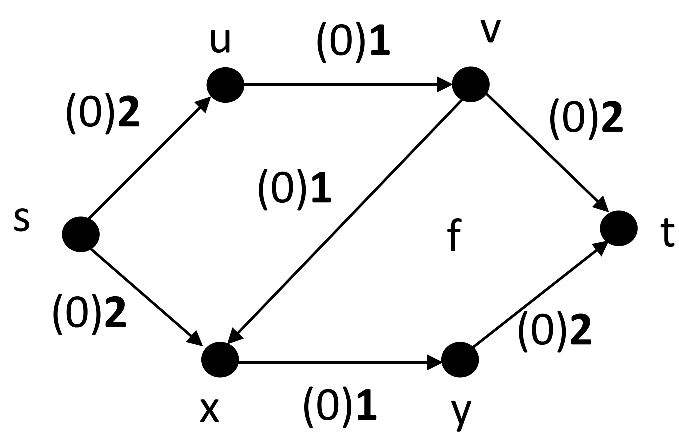
- A flow f assigns a value $f(e)$ to each edge e.
- Let:
  - $f^+(v)$: the total flow on edges leaving v and
  - $f^-(v)$: the total flow on edges entering v
- A flow is feasible if it satisfies
  - The capacity constraints $0 \leq f(e) \leq c(e)$ for each edge and
  - The conservation constraints $f^+(v) = f^-(v)$ for each node $v \notin \{s, t\}$.

Maximum Network Flow

- The ***value*** val(f) of a flow f is the net flow $f^{-}(t)$ - $f^{+}(t)$ into the sink.
- A ***maximum flow*** is a feasible flow of maximum value.

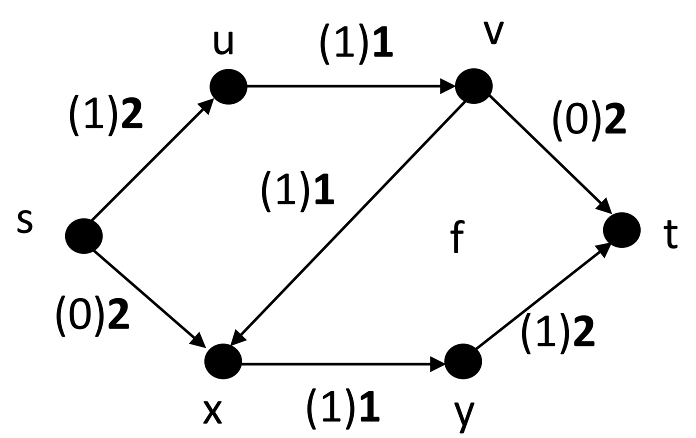
Example of Max Flow

- The ***zero flow*** assigns flow 0 to each edge
- It is feasible.



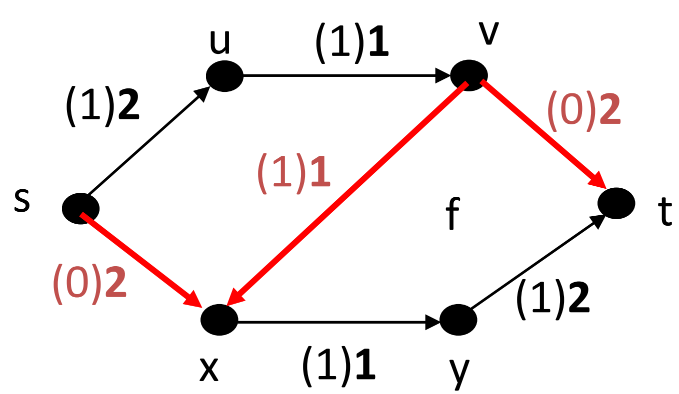
Example of Max Flow

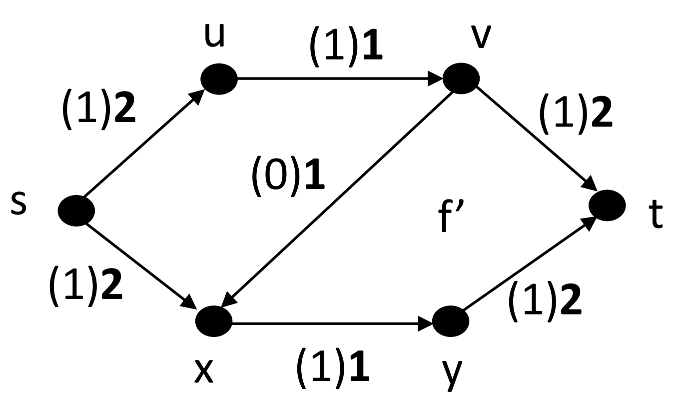
- In the network below we illustrate a non-zero feasible flow.
  - Capacities are shown in bold, flow values in parentheses.
  - Our flow f assigns $f(sx) = f(vt) = 0$, and $f(e) = 1$ for every other edge e. This is a feasible flow of value 1.



Example of Max Flow

- A path from the source to the sink with excess capacity would allow us to increase flow.
  - In this example, no path remains with excess capacity, but the flow f' with $f'(vx) = 0$ and $f'(e) = 1$ for $e \neq vx$ has value 2.





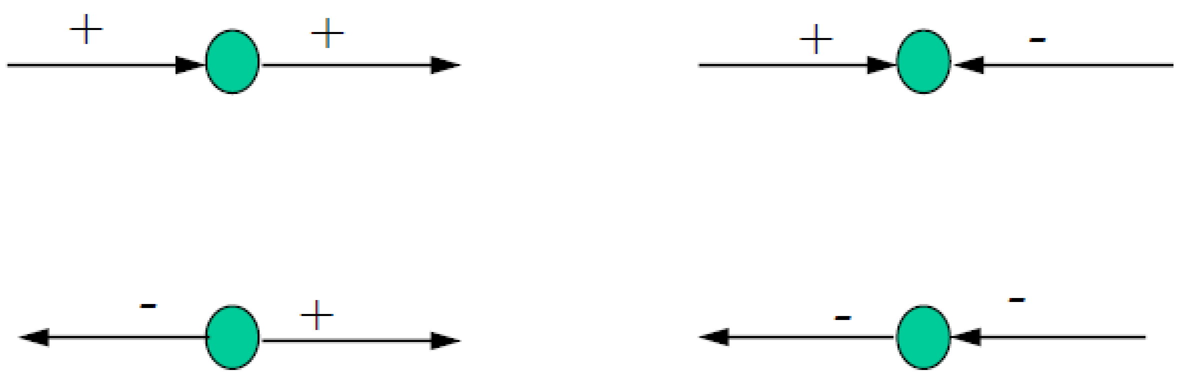
f-Augmenting Path 4.3.4

- When f is a feasible flow in a network N, an ***f-augmenting path*** is a source-to-sink path P in the underlying graph G such that for each $e \in E(P)$,
  - a) if P follows e in the forward direction, then $f(e) < c(e)$.
  - b) if P follows e in the backward direction, then $f(e)>0$.
- Let $\varepsilon(e)=c(e) - f(e)$ when e is forward on P, and let $\varepsilon(e)=f(e)$ when e is backward on P.
- The ***tolerance*** of P is $\min_{e\in E(P)}\varepsilon(e)$.

New Flow after Augmenting

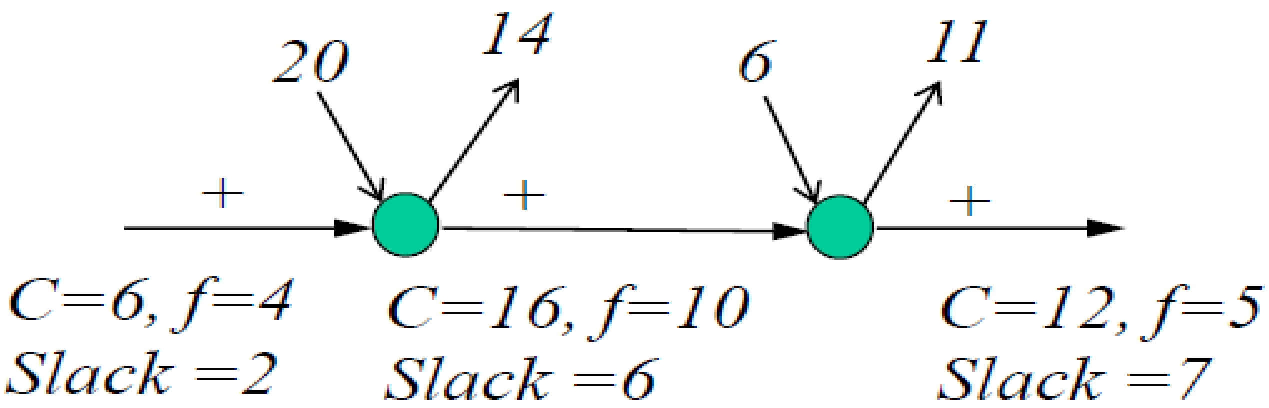
- The edges of P incident to an internal vertex v of P occur in one of the four ways shown below.
- In each case, the change to the flow out of v is the same as the change to the flow into v, so

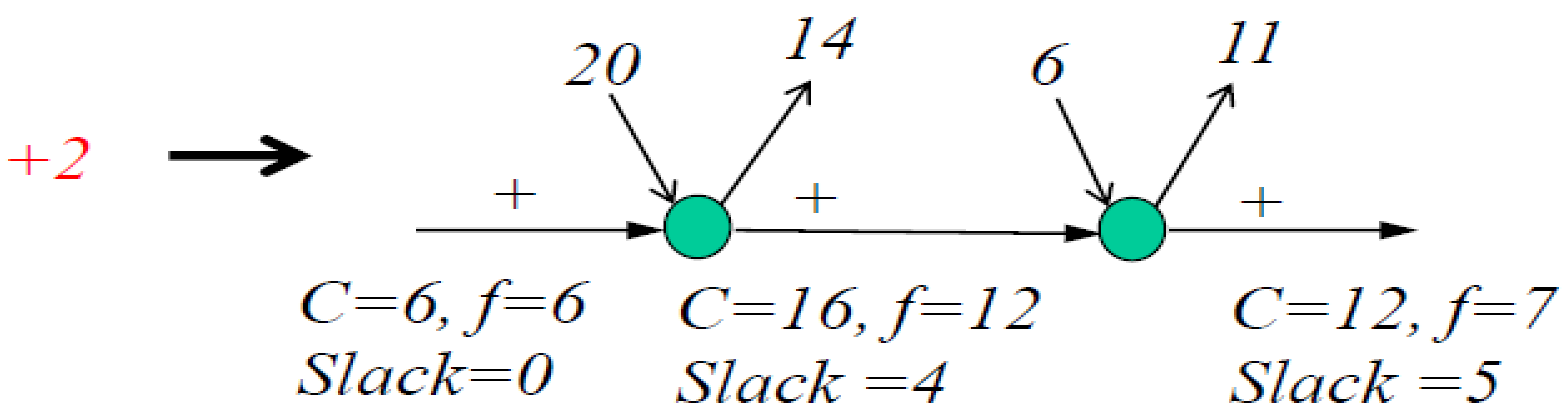
$$f^{-}(v) = f^{+}(v).$$



New Flow after Augmenting

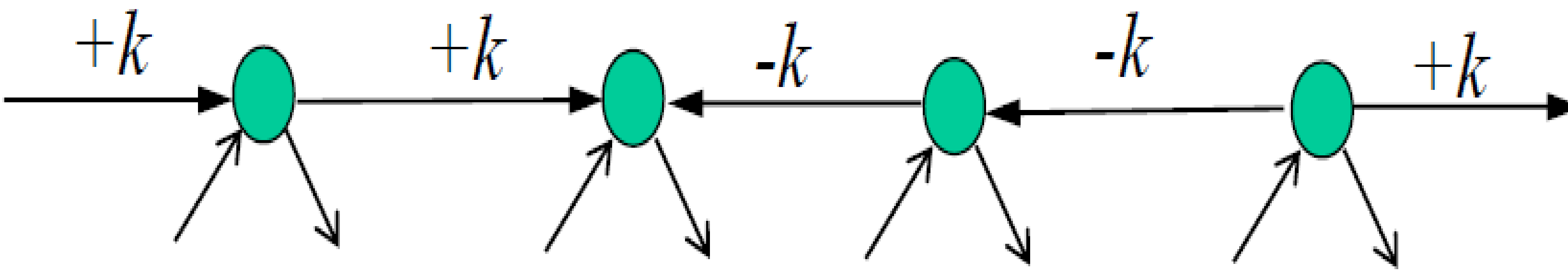
- Examples





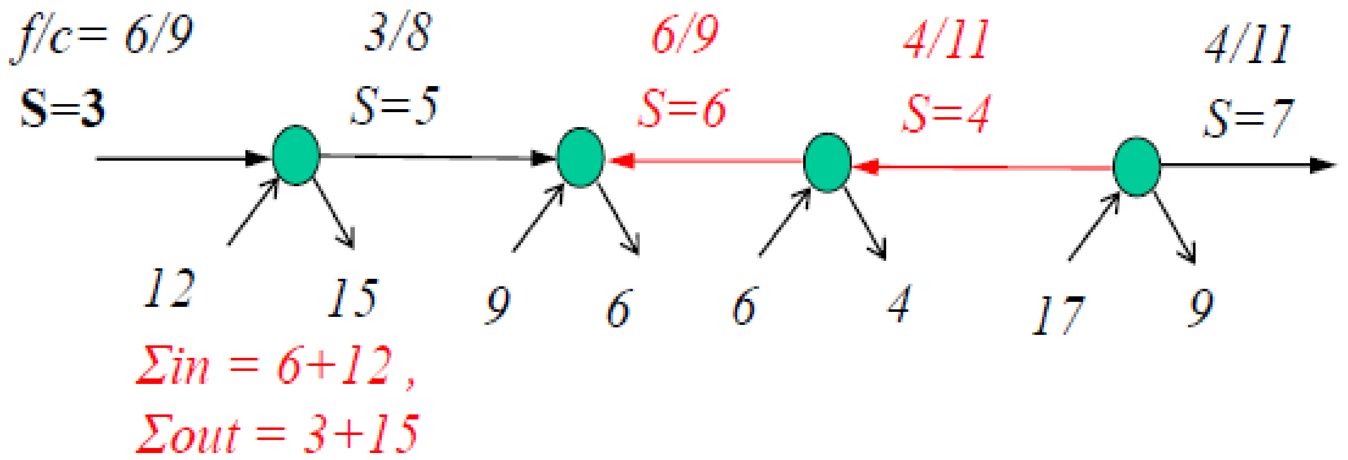
New Flow after Augmenting

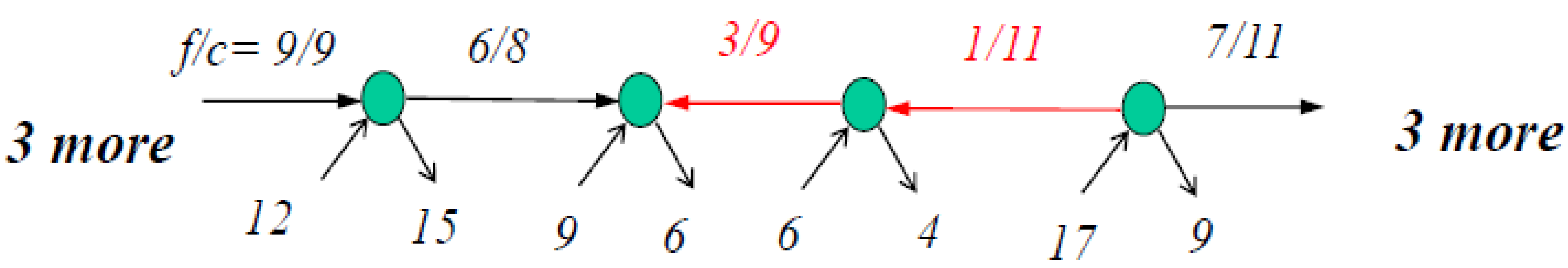
- Examples



New Flow after Augmenting

- Examples

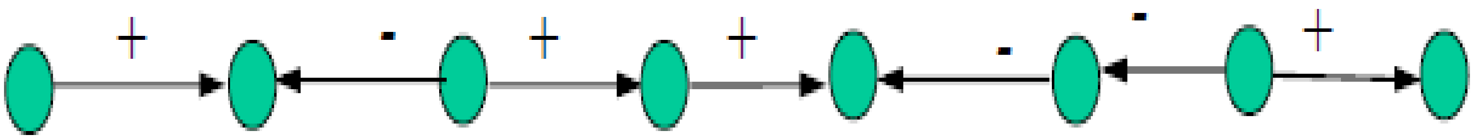




Lemma. If P is an f-augmenting path with tolerance z, then changing flow by $+z$ on edges followed forward by P and by $-z$ on edges followed backward by P produces a feasible flow f' with $\mathrm{val}(f') = \mathrm{val}(f)+z$.

Proof:

- The definition of tolerance ensures that $0 \le f'(e) \le c(e)$ for every edge e, so the capacity constraints hold.
  - We need only check vertices of P, since flow elsewhere has not changed.
- For every vertex v, $f^+(v) = f^-(v)$
- Finally, the net flow into the sink t increases by z.



Source/sink cut

- In a network, a ***source/sink cut*** $[S,T]$ consists of the edges from a source set S to a sink set T, where S and T partition the set of nodes, with $s \in S$ and $t \in T$.

- The ***capacity*** of the cut $[S, T]$,written cap(S, T), is the total of the capacities on the edges of $[S, T]$.

- Keep in mind that in a digraph $[S, T]$ denotes the set of edges with tail in S and head in T. Thus the capacity of a cut $[S, T]$ is completely unaffected by edges from T to S.

Ford-Fulkerson Labeling Alg. For Max- flow 1 4.3.9

- **Input:** A feasible flow f in a network.
- **Output:** An f-augmenting path or a cut with capacity val(f).
- **Idea:** Find the nodes reachable from s by paths with positive tolerance. Reaching t completes an f-augmenting path. During the search, R is the set of nodes labeled ***Reached***, and S is the subset of R labeled ***Searched***.

Ford-Fulkerson Labeling Alg. For Max- flow 2

Initialization: $R = \{s\}$, $S = \phi$.

Iteration: Choose $v \in R\text{-}S$.

- For each exiting edge vw with $f(vw) < c(vw)$ and $w \notin R$, add w to R.
- For each entering edge uv with $f(uv>0)$ and $u \notin R$, add u to R.
- Label each vertex added to R as "reached", and record v as the vertex reaching it. After exploring all edges at v, add v to S.
- If the sink t has been reached (put in R), then trace the path reaching t to report an f-augmenting path and terminate. If $R = S$, then return the cut $[S, \hat{S}]$ and terminate. Otherwise, iterate.

Theorem: Max-flow Min-cut Theorem- Ford and Fulkerson [1956] 4.3.11

- In every network, the maximum value of a feasible flow equals the minimum capacity of a source/sink cut.

Proof:

- In the max-flow problem, the zero flow (*f*(e)=0 for all e) is always a feasible flow and gives us a place to start. Given a feasible flow, we apply the labeling algorithm. It iteratively adds vertices to S (each vertex at most once) and terminates with t ϵ R ("breakthrough") or with S=R.
- In the breakthrough case, we have an *f*-augmenting path and increase the flow value. We then repeat the labeling algorithm, When the capacities are rational, each augmentation increases the flow by a multiple of 1/a, where a is the least common multiple of the denominators, so after finitely many augmentations the capacity of some cut is reached. The labeling algorithm then terminates with S=R.

Theorem: Max-flow Min-cut-Ford and Fulkerson [1956] contd...

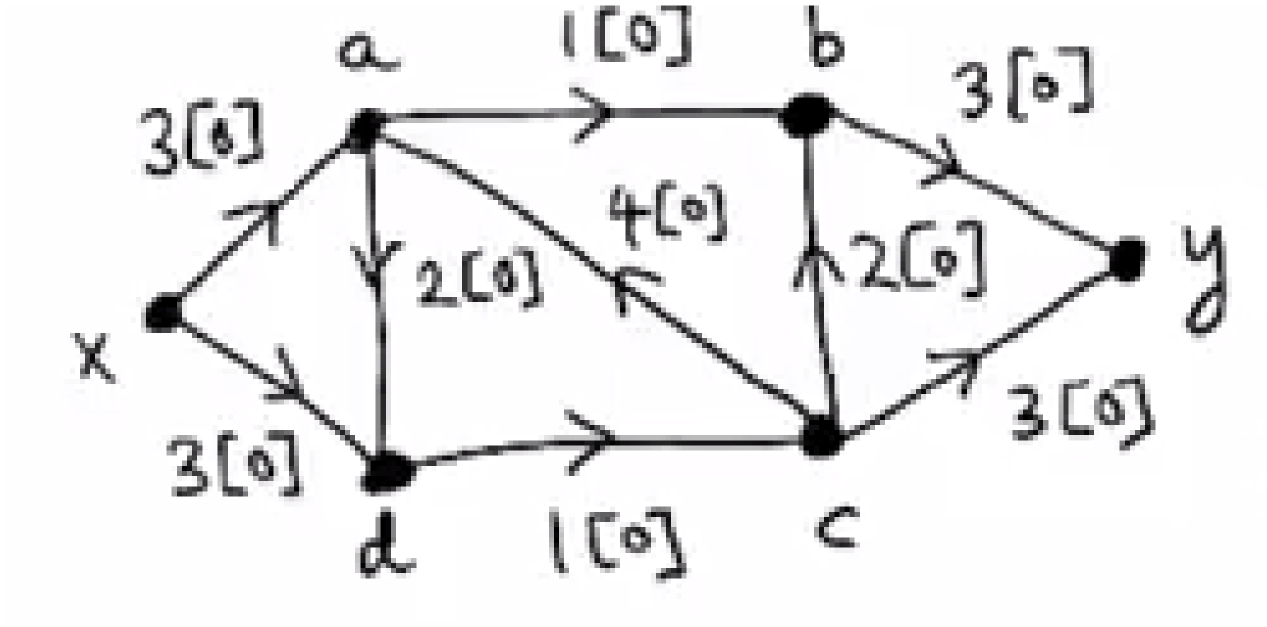
- When terminating this way, we claim that $[S,T]$ is a source/sink cut with capacity val(f), where $T=\bar{S}$ and f is the present flow, It is a cut because $s \in S$ and $t \notin R = S$.
- Since applying the labeling algorithm to the flow f introduces no node of T into R, no edge from S to T has excess capacity, and no edge from T to S has nonzero flow in f. Hence $f^{+}(S)$= cap(S,T) and $f^{-}(S)=0$.
- Since the net flow out of any set containing the source but not the sink is val(f), we have proved

$$\text{Val}(f) = f^{+}(S) - f^{-}(S) = f^{+}(S) = \text{cap}(S,T).$$

Examples:
Ford Fulkerson Algorithm

Example (1)

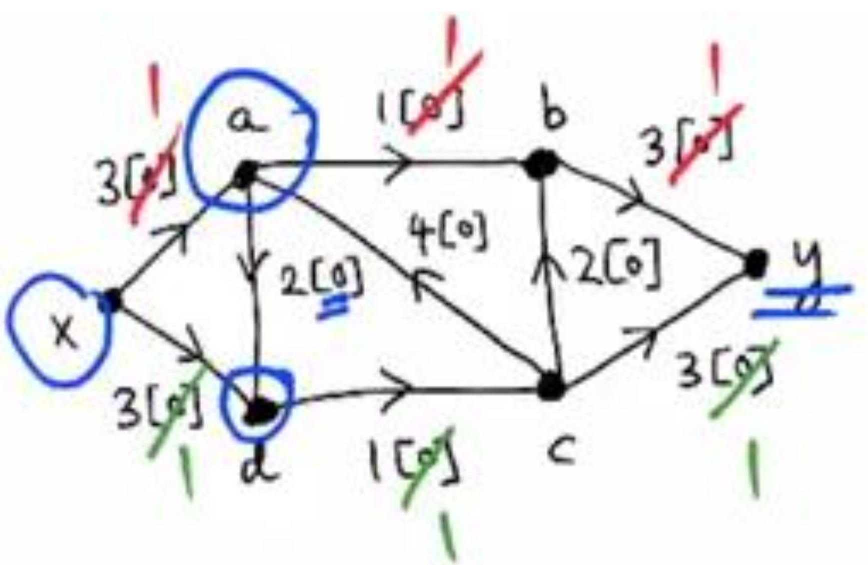
Q. 1 Apply the Ford Fulkerson Algorithm to determine the value of maximum flow from the source x to the sink y.



Solution:

x → a → d

x→ d



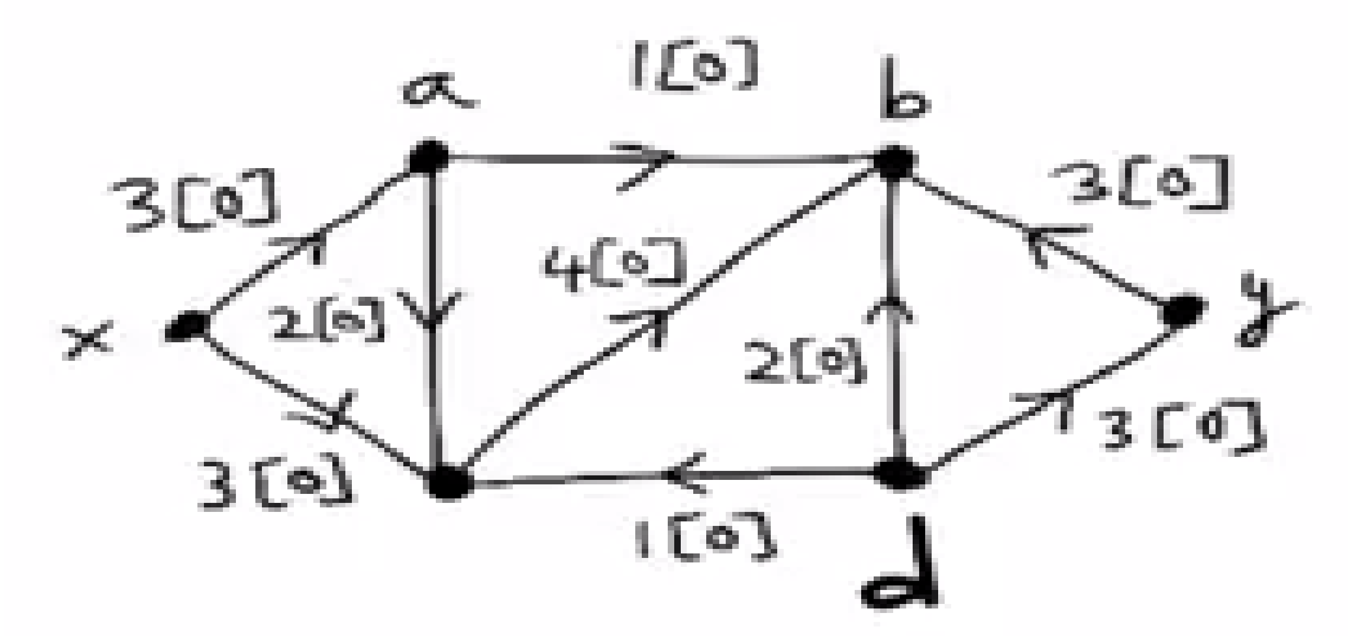
Q	Residual Capacity (forward)	Flow (reverse)	∑ (Q)
xaby	3,1,3	-	1
xdcy	3,1,3	-	1

Max flow = val(f)

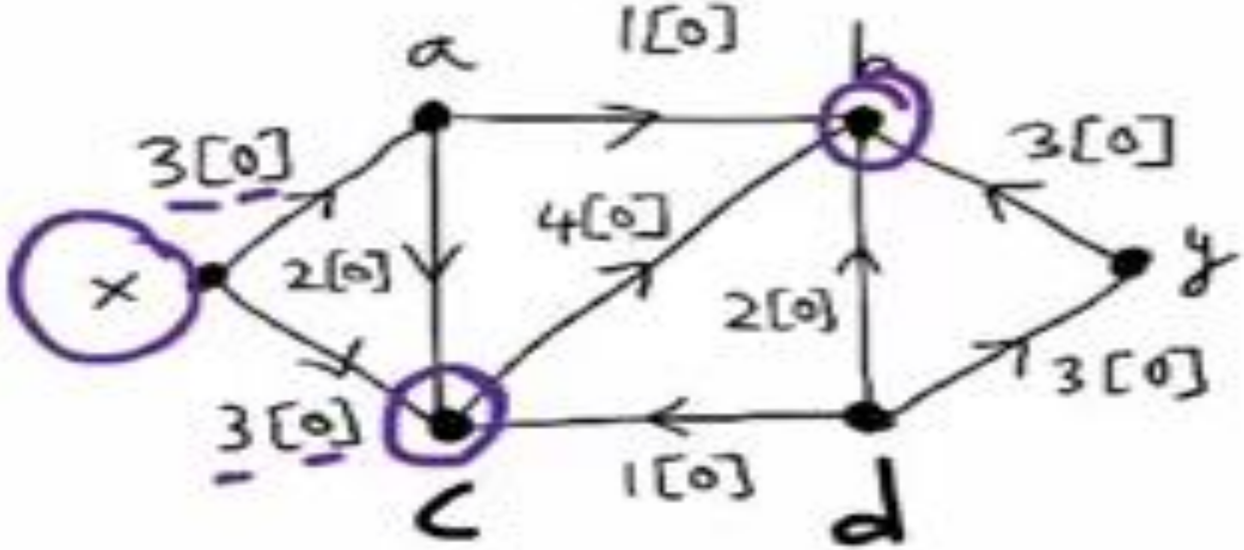
$= f^{+}(x) - f^{-}(x) = 1 + 1 - 0 = 2$

Example (2)

Q. 2 Apply the Ford Fulkerson Algorithm to determine the value of maximum flow from the source x to the sink y.



Solution:



$x \rightarrow a \rightarrow b^*$

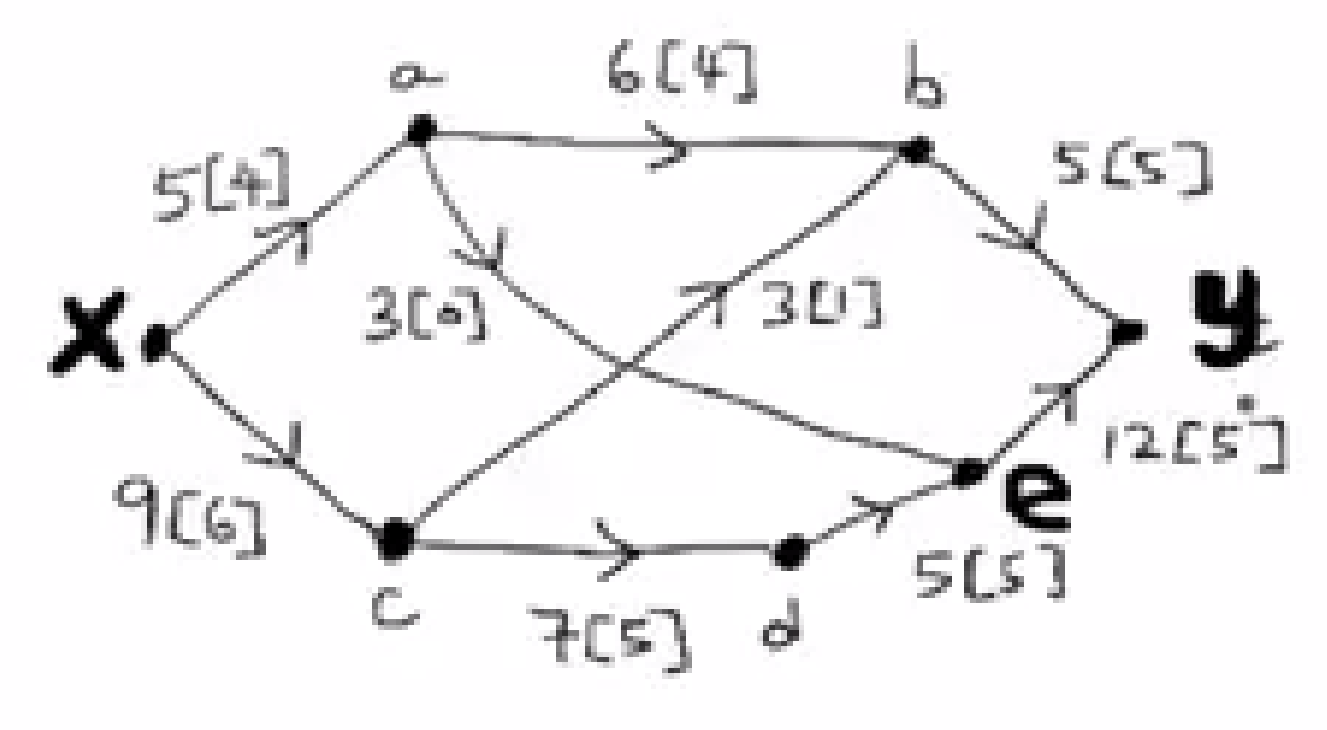
$x \rightarrow a \rightarrow c \rightarrow b^*$

$x \rightarrow c \rightarrow b^*$

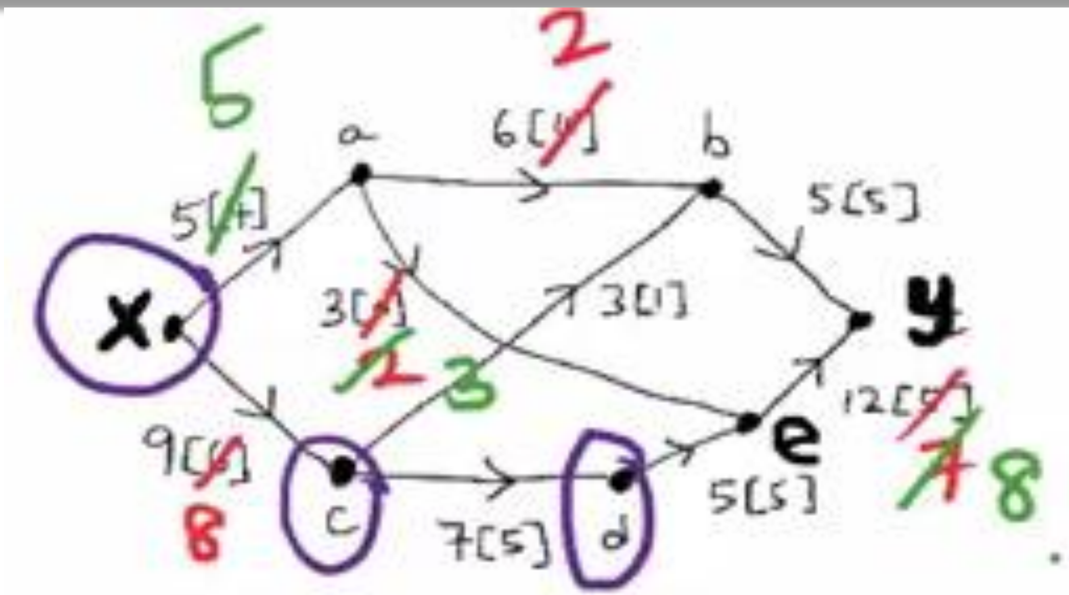
max flow = 0

Example (3)

Q. 3 Apply the Ford Fulkerson Algorithm to determine the value of maximum flow from the source x to the sink y.



Solution:



x → c → b → a*

d*

Max flow

$= f^{+}(x) - f^{-}(x)$

= 5 - 0

= 5

Q	Residual Capacity (forward)	Flow (reverse)	∑ (Q)
xcbaey	3,2,3,7	4	2
xaey	1,1,5	-	1

Menger's Theorem Proof

Remark: Menger from Max-flow Min-cut 4.3.13

- When x, y are vertices in a digraph D, we can view D as a network with source x and sink y and capacity 1 on every edge. Capacity 1 ensures that units of flow from x to y correspond to pairwise edge-disjoint x, y-paths in D. Thus a flow of value k yields a set of k such paths.
- Similarly, every source/sink partition S, T defines a set of edges whose deletion makes y unreachable from x: the set [S,T]. Since every capacity is 1, the size of this set is cap(S,T).
- The paths and the edge cut we have obtained might not be optimal, but by the Max-flow Min-cut Theorem we have

$$\lambda'_D(x,y) \geq \max \mathrm{val}(f) = \min \mathrm{cap}(S,T) \geq \kappa'_D(x,y)$$

- Since always $\kappa'(x,y) \geq \lambda'(x,y)$, equality now holds.

Menger's Theorem Proof

- **Goal:** Deduce Menger's Theorem (vertex version) from max-flow min-cut

Theorem (Menger's Theorem-vertex version)

Let x,y be two distinct vertices in a connected graph $G=(V,E)$, such that $xy \notin E$. Then $\kappa(x,y) = \lambda(x,y)$

Proof:

We will show that

(1) $\lambda(x,y) \leq \kappa(x,y)$ (use definitions)

(2) $\kappa(x,y) \geq \lambda(x,y)$ (use max-flow min-cut)

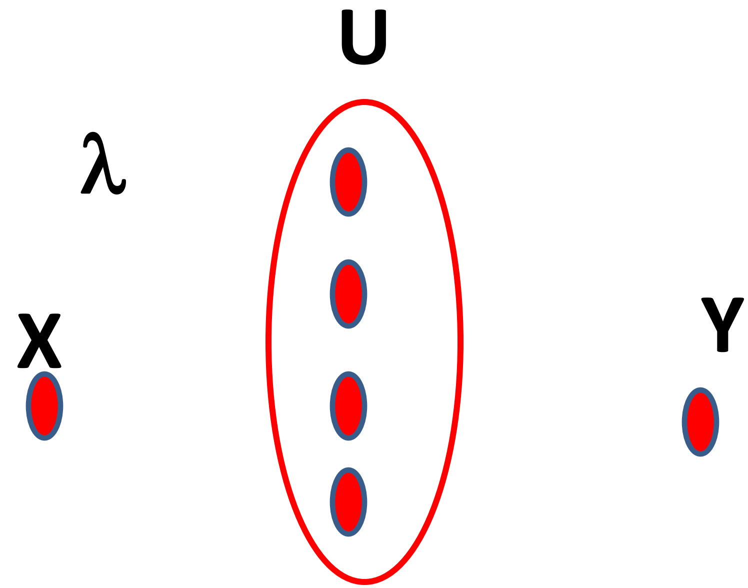
$\kappa(x,y)$ is the minimum size of an x,y cut

$\lambda(x,y)$ is the maximum number of pairwise vertex disjoint/internally disjoint paths

(1) To show $\lambda(x,y) \le \kappa(x,y)$: Proof

- Take a minimum (x, y)-cut U.
- Every xy-path must go through U.
- Number of (pairwise) internally disjoint paths is at most |U|

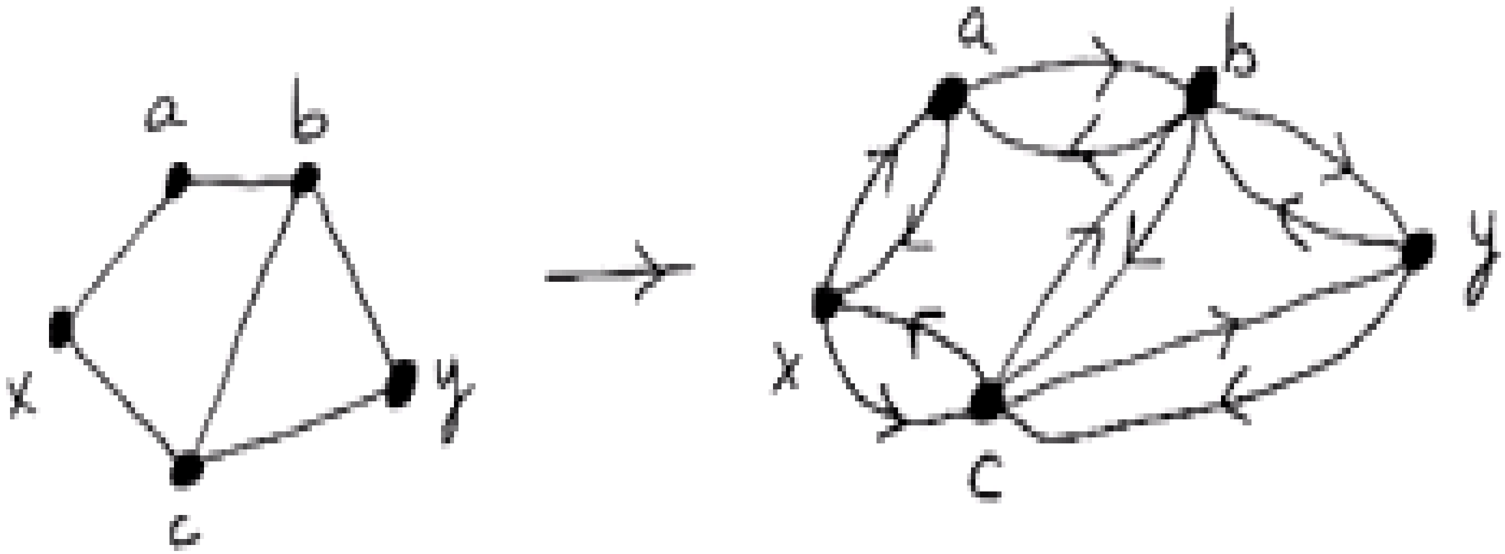
$$\Longrightarrow \quad \lambda(x, y) \le |U| = \kappa(x, y)$$



(2) To show $\kappa(x,y) \leq \lambda(x,y)$: Proof

Construct a network N with source x and sink y as follows:

- Replace each edge uv with two directed arcs: u → v and v → u

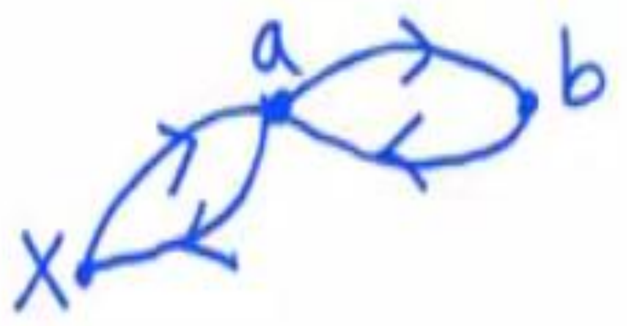


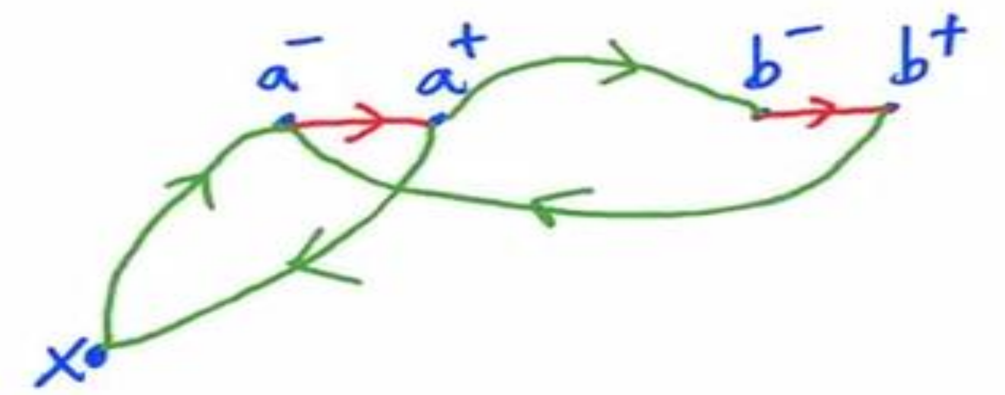
- Split each vertex $w \notin \{x,y\}$ into two vertices w^- and w^+, together with a (new) directed edge from w^- to w^+.

- Such an arc $w^- \rightarrow w^+$ is called an **internal arc** of the network.

- Other arcs $u \rightarrow v$ will be replaced by
  - $u^+ \rightarrow v^-$ if $u, v \notin \{x, y\}$;
  - $u \rightarrow v^-$ if $u = x, y$;
  - $u^+ \rightarrow v$ if $v = x, y$;

$\overset{+}{u} \longrightarrow \overset{-}{v}$





After Step 1, we have the source X directed to a and another arc in the opposite direction and we have an arc from a to b and another arc in the opposite direction

According to Step 2, we are going to split the vertex which are not the source or sink.

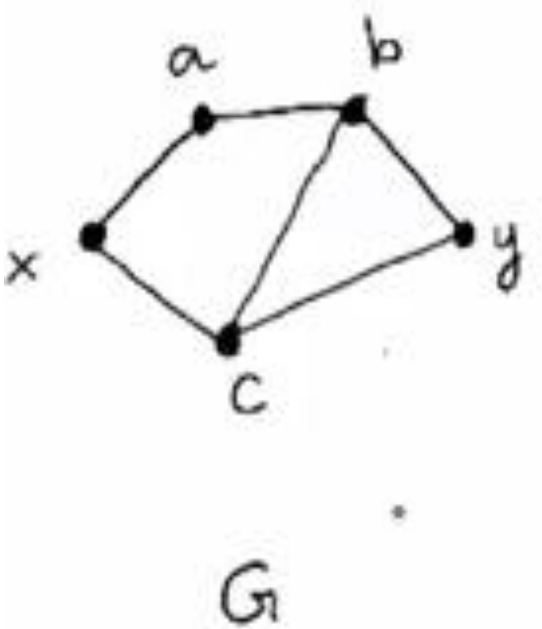
Vertex a is splitted to – vertex and + vertex. Similarly vertex b is splitted and source remain the same

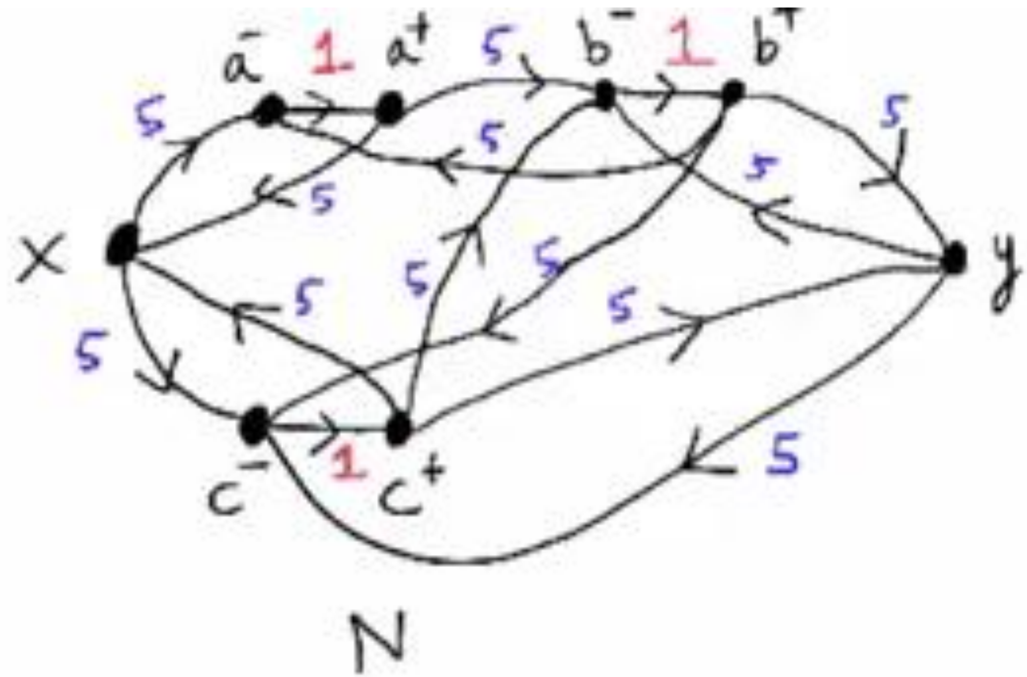
As mentioned in step 2- there will be internal arcs directed from a^- to a^+, and b^- to b^+ highlighted by red. Also there is an arc from a to b then we are going to direct from the + vertex to – vertex. i.e. replace by a^+ to b^- , b^+ to a^- highlighted by green.

By rule draw arc from source to (-) vertex then draw arc from (+) vertex to the source vertex. i.e. x to a^- and a^+ to x.

Rule is form the + vertex to the minus vertex
$u^+ \rightarrow v^-$

- Every internal arc will have a capacity of 1, and other arcs will have a capacity of *n* (which is the number of vertices).





- In Graph G, there are 5 vertices all together. Therefore for every arc which is not an internal arc in network N, we have a capacity of 5 where as all internal arcs assigned the capacity 1.

Observation:

(1) Every vertex of the form w^- has **exactly** one arc going out from it, which is the internal arc $w^- \rightarrow w^+$

(2) Every vertex of the form w^+ has **exactly** one arc coming into it, which is the internal arc $w^- \rightarrow w^+$

Approach: Show

(1) $\lambda(x,y) \geq$ *max flow*

(2) $\kappa(x,y) \leq$ min cut

$\boldsymbol{\kappa(x,y) \leq \lambda(x,y)}$

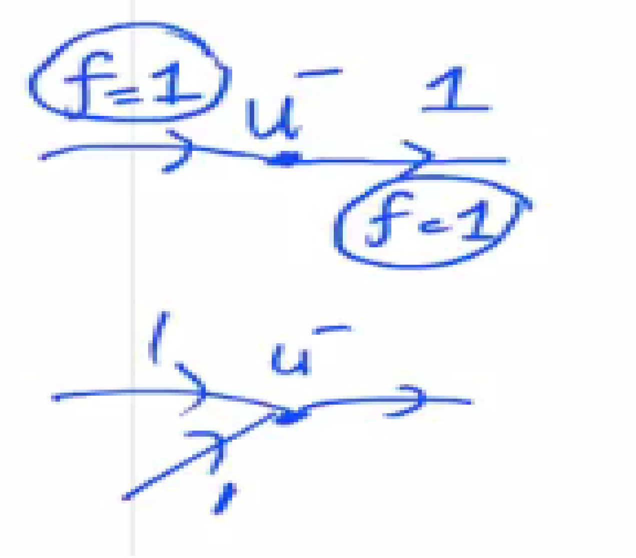
Max-flow min cut $\Longrightarrow$

$\kappa(x,y) \leq$ min cut = *max flow* $\leq \lambda(x,y)$

To show that $\lambda(x,y) \geq$ *max flow*

- Let f be a max flow, with val(f) = m.
- If there is a flow into u-, then the value must be 1 (since there is only one arc directed from u- - Observation (1))
- This one unit of flow must travel from x to y.
- m units flow transform into m internally disjoint xy-paths

$\Longrightarrow \lambda(x,y) \geq m =$ *max flow*

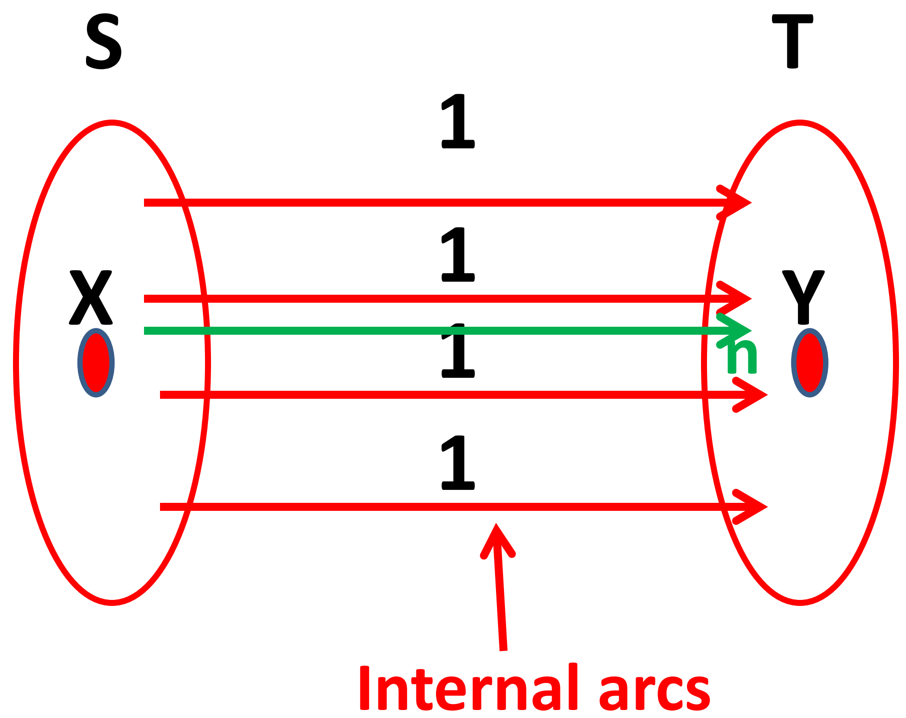


$\kappa(x,y) \leq$ min cut

1) K = [S,T] minimum (x,y)-cut of N.

Assume not.
Can find a non-internal arc directed from S to T

=> cap(K) ≥ n



To show that $\kappa(x,y) \leq$ min cut

- Consider the following x-y-cut $K^* = [S^*, T^*]$ given by

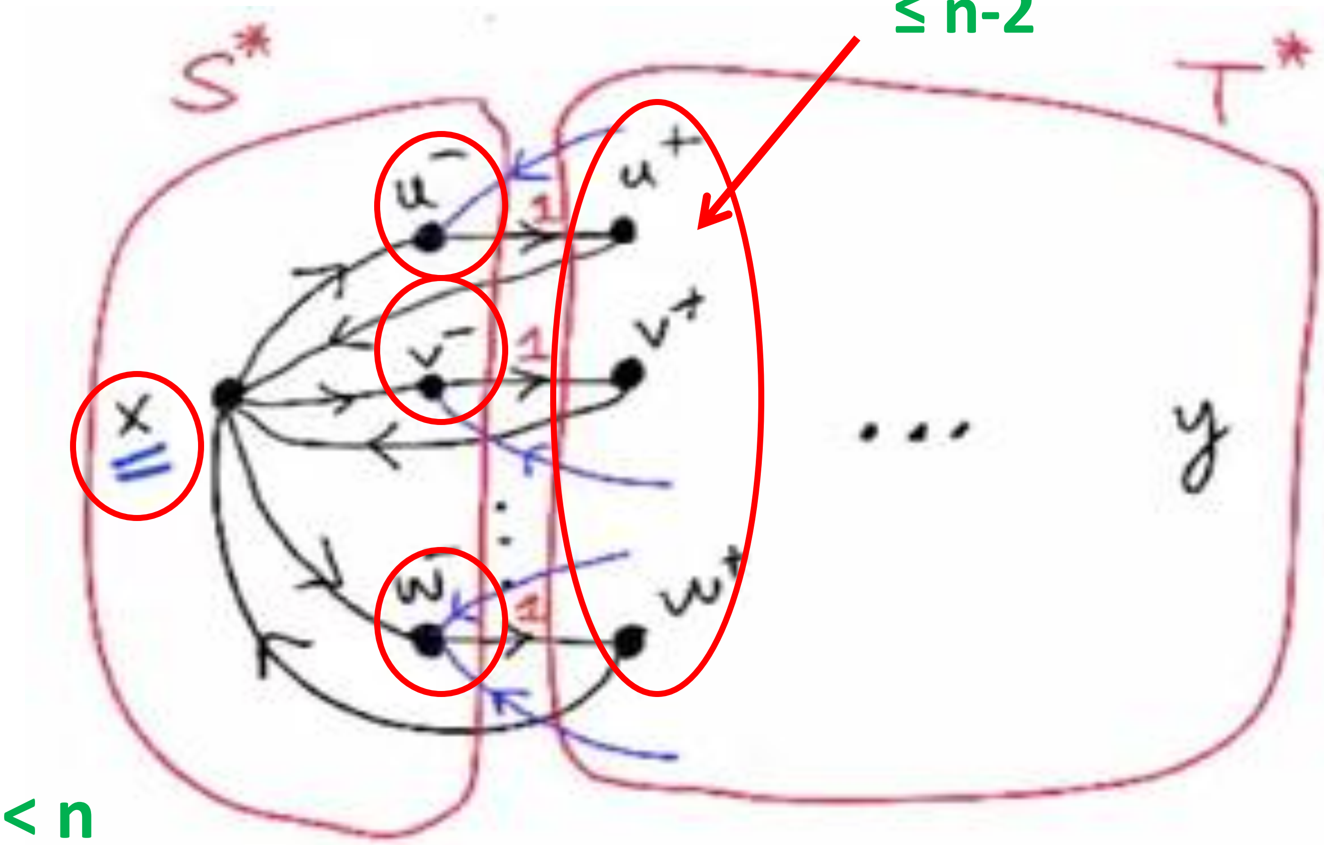
$$S^* = \{x\} \cup \{u^- : xu \in E(G)\},\ T^* = \overline{S^*}$$

- Note: $cap(K^*) \leq n-2$

- Since x has at most n-2 neighbors in G ($n = |V(G)|$)

Source X, all the vertices which are the endpoints of arc directed from X and all the vertices inside the T* including the sink Y. Inside the S* all the vertices are the (-) vertices except from the source . All the arcs are going from the S* are the internal arcs. Therefore the capacity of these arcs are 1.

≤ n-2

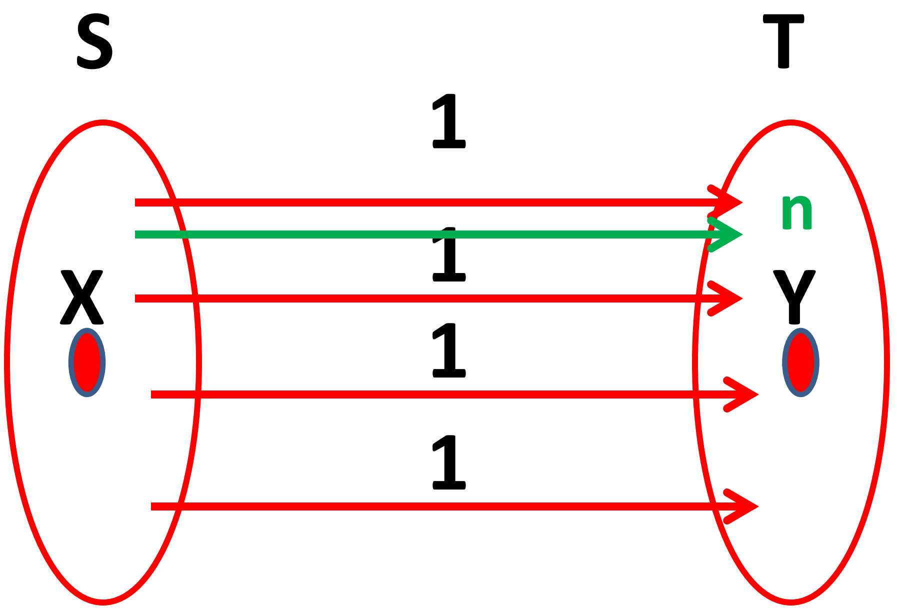


cap (K*) ≤ n-2 < n

1) K = [S,T] minimum (x,y)-cut of N.

Assume not.
Can find a non-internal arc directed from S to T

=> cap(K) ≥n



Internal arcs

Contradicts the minimality of cut K

- Let K = [S,T] be a minimum x-y cut in N.

=> all arcs directed from S to T are internal arcs.

- If not, there is a non-internal arc (whose capacity is n) directed from S to T => cap[S,T] $\geq$ n > cap [S*, T*] (contradiction)

$\kappa(x,y) \leq$ min cut

Idea:

- K = [S,T] min $\longrightarrow$ Construct (x,y)-cut U in G
  (x,y)- cut in N

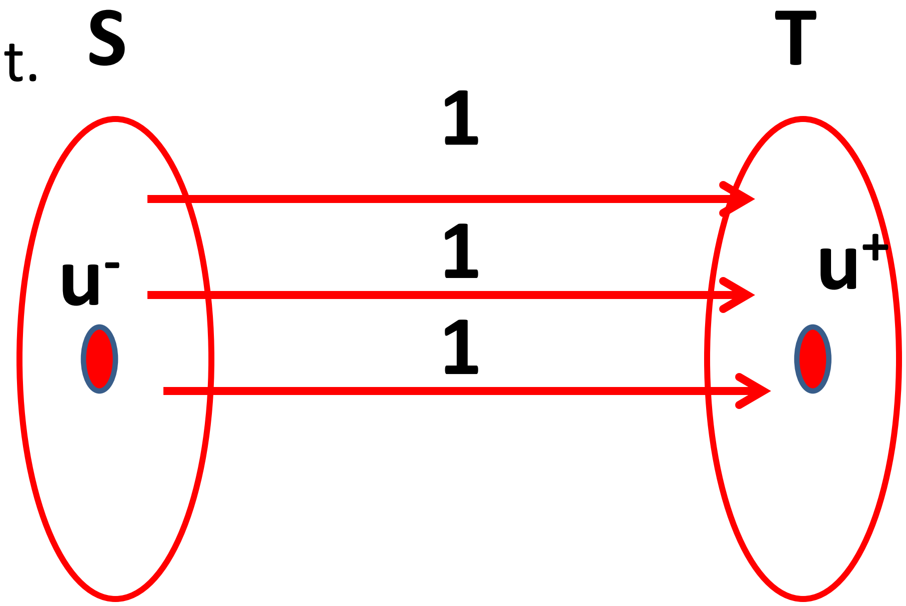
- Let

  $U = \{ u \in V(G): u^- \in S, u^+ \in T\}$

- Note that $|U| = cap(K) =$ min cut.

- Aim: U is an (x, y) –cut in G.

=> $\kappa(x, y) \leq |U| = cap(K) =$ min cut



$$P = x - u_1 - u_2 - u_3 - \cdots - u_t - y$$

Path P from source x to y. Start at source x the followed by u1,u2,u3 before it reaches to vertex y

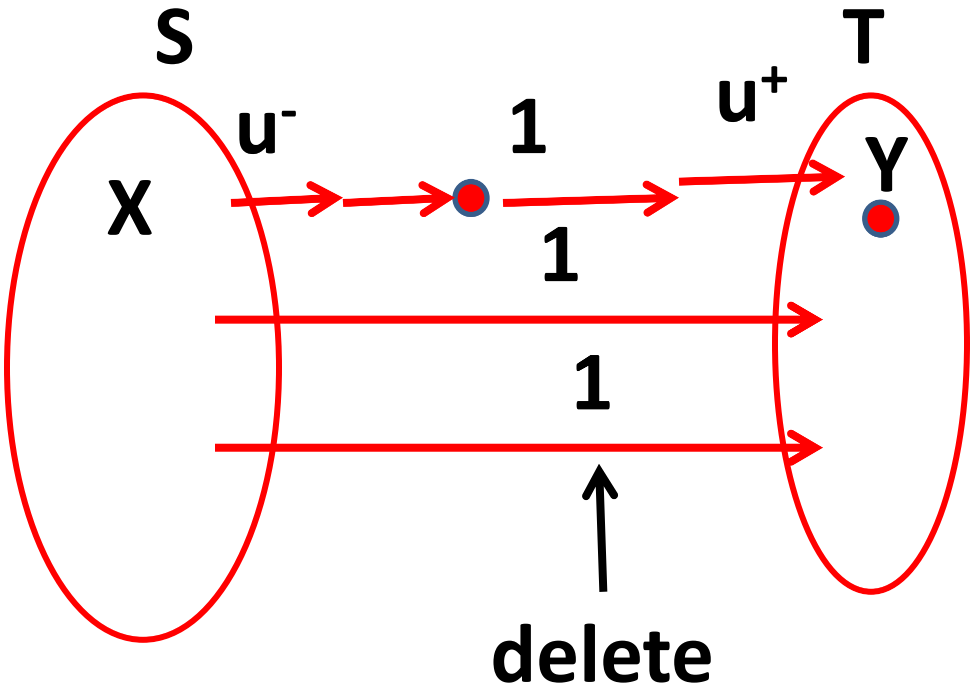
- $P = xu_1 \ldots\ldots u_t y$ be an xy-path in G.
- P can be converted into a directed xy-path $\overrightarrow{P}$ in the directed graph D underlying the network N as follows:

$$\overrightarrow{P} = x \rightarrow u_1^{+} \rightarrow u_2^{-} \rightarrow u_2^{+} \ldots\ldots \rightarrow u_t^{+} \rightarrow y.$$

$$\overrightarrow{P} = x \rightarrow u_1^{-} \rightarrow u_1^{+} \rightarrow u_2^{-} \rightarrow u_2^{+} \rightarrow \cdots \rightarrow y$$

Directed Path in network N denoted by P

$$\vec{P} = x \to u_1^- \to u_1^+ \to u_2^- \to u_2^+ \to \dots \to y$$



Let’s visualize the Path starting at X then at some point it must cross over to set T In order to reach Y. This will be an internal arc directed from the u^+ to u^- and the capacity of this arc is 1. If we delete these arcs then the source will be separated from the sink

- This directed path $\overrightarrow{P}$ must contain an internal arc directed from S to T.
    - deleting internal arcs from S to T will break the path.
    - transforming back to G, deleting vertices in U from G will separate x from y
- U is an (x,y)-cut of G.

We are done

Conclusion

- In this lecture, we have discussed the Network Flow Problems *i.e.*

- Maximum Network Flow

- *f*-augmenting path

-Ford-Fulkerson labeling algorithm

- Examples based on Ford-Fulkerson labeling algorithm

- Max-flow Min-cut Theorem and

- Proof of Menger's Theorem

- In upcoming lecture, we will discuss Graph Coloring *i.e.* Vertex Coloring and Upper Bounds.