

25.3 - 8)

Suppose $w(u,v) \geq 0$ for all edges $(u,v) \in E$. What is the relationship between the weight functions w and \hat{w} ?

All weights are non-negative:

$$w(u,v) = \hat{w}(u,v)$$

↳ Values computed as shortest distances on running

BELLMAN-FORD ALGORITHM will be ZERO



Computing G' of JOHNSON'S ALGORITHM

Involves placing edge weight of ZERO from s to all vertices



As there are no negative edges. Every path within the graph has no negative edges, its cost cannot be negative and does not beat the trivial path from s to any other vertex

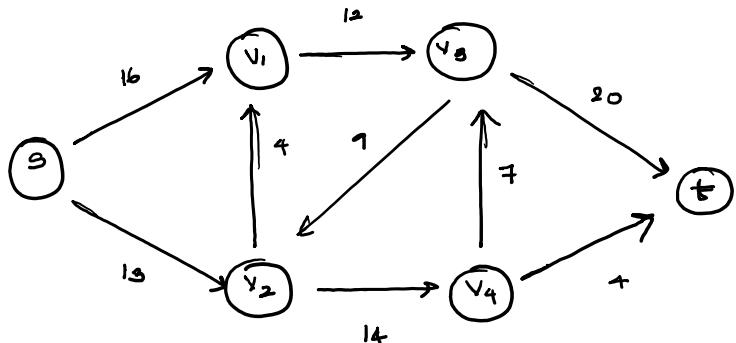


As we have $b(u) = b(v)$, reweighting does not change weights of the graph. i.e adds or subtracts [ZERO]

26-2-10) Show how to find a maximum flow in a network $G = (V, E)$ by a sequence of almost $|E|$ augmenting paths

[Hint: Determine the paths after finding the maximum flow]

Consider the graph :



Residual network for the graph with final flows

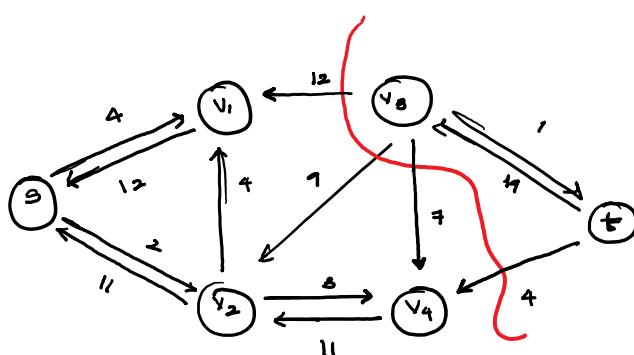
Cut of the graph

where edges from s to t

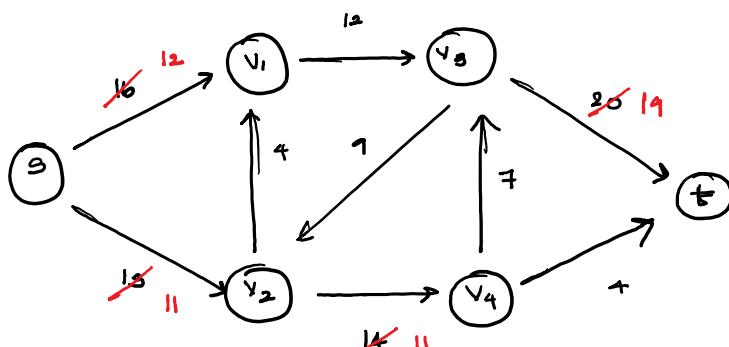
are exhausted

[NO FORWARD EDGE]

in residual network.



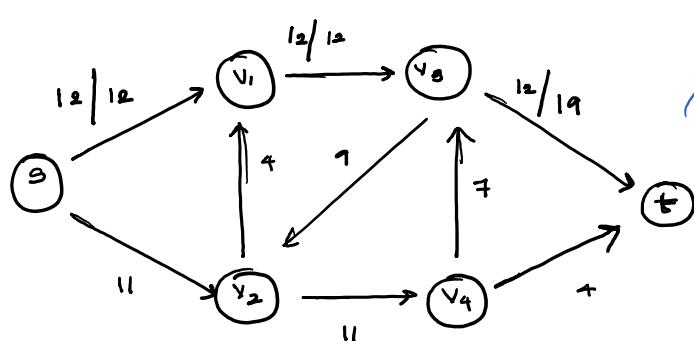
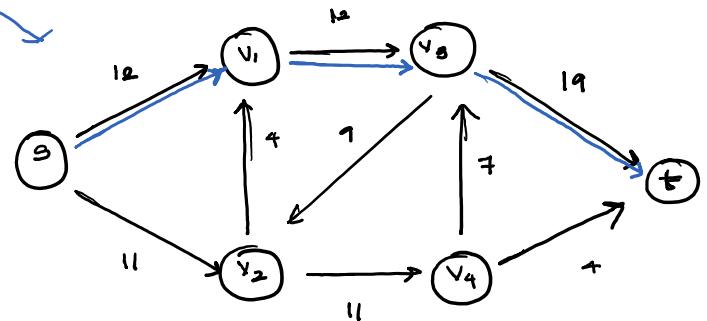
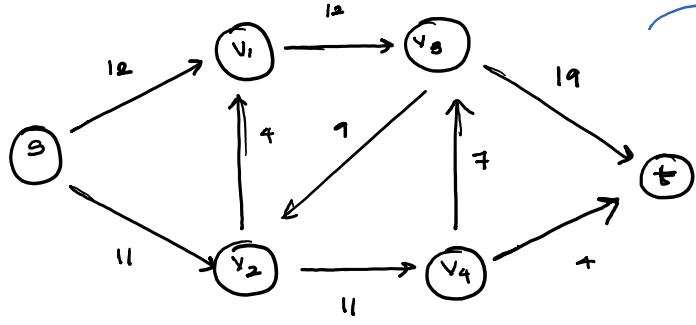
Now consider modifying capacity in the original graph with the maximum flow values. Then resulting graph shall be as follows



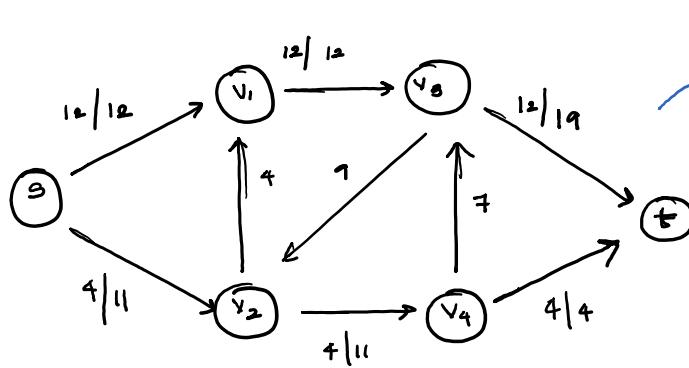
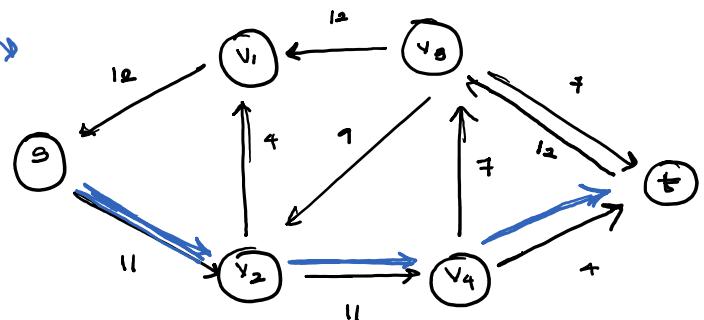
→ With this modification there shall be no reverse edges in the final residual network.

[Run FORD FULKERSON on the above graph by storing augmenting paths]

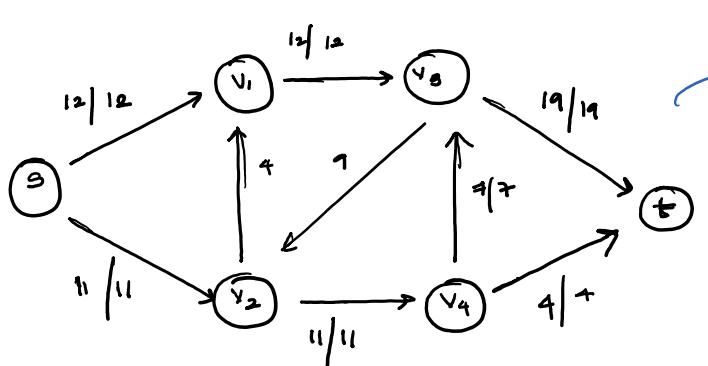
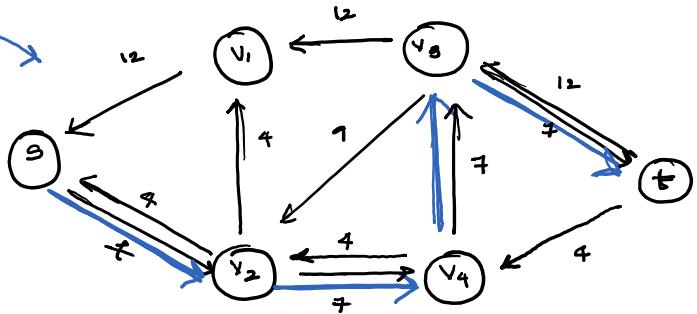
[Residual Network]



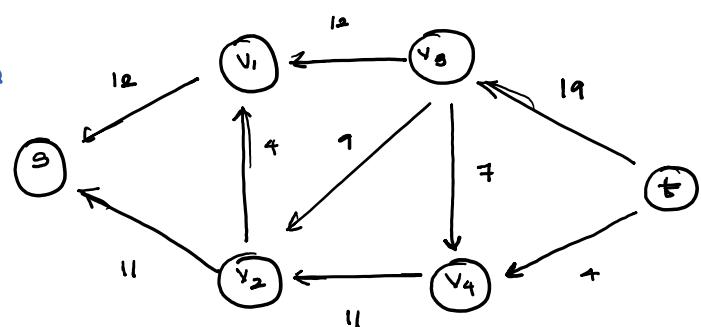
RN



RN



RN



Max |EI| paths

[No Augmenting Path]

For each iteration \rightarrow one edge is exhausted by no reverse edges cancelling flow

|EI| AUGMENTING PATHS \rightarrow Paths obtained here after each iterations

26-2-13) Suppose that you wish to find among all minimum cuts in a flow network G with integral capacities of G one that contains the smallest number of edges. Show how to modify the capacities of G to create a new flow network G' in which any minimum cut in G' is a minimum cut with the smallest number of edges in G .

Observations :

Let there be k min cuts with varied number of edges in each cut.

So, adding an extra capacity (e) to each edge changes the capacities of k min cuts by different amounts.

Now, as per max-flow min-cut theorem flow shall be equal to MIN CUT

Here as we increase capacity of each edge by e

Min cut in G with smallest no. of edges across the cut \leftarrow smallest increase in the capacity of the cut.

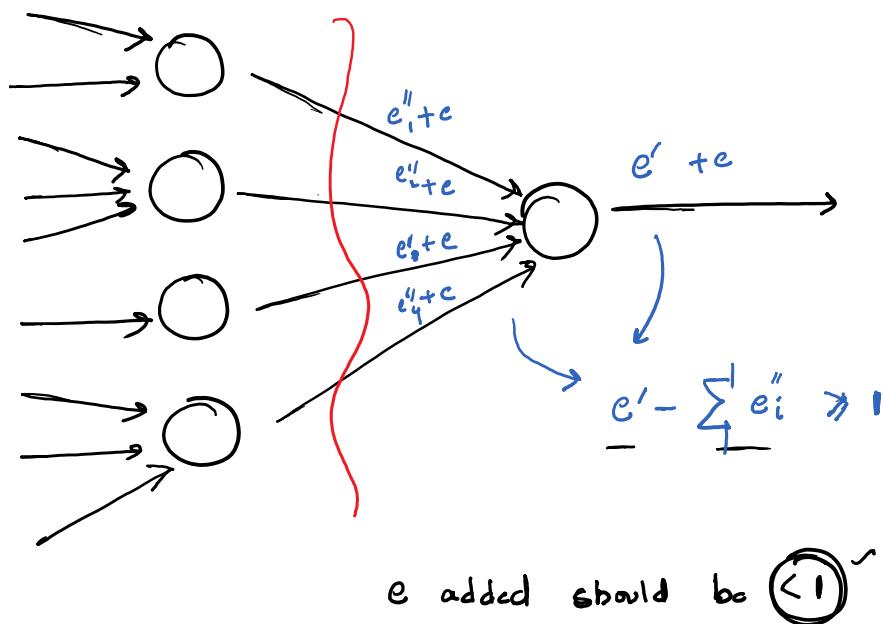
Let edges in k min cuts be

$$p_1 \ p_2 \ \dots \ p_k \quad (p_1 < p_2 < p_3 \dots < p_k)$$

So, new capacities of the cuts are $\rightarrow [C_{min} + p_1 e], C_{min} + p_2 e \dots C_{min} + p_k e$

\downarrow [MINIMUM CUT]

The new graph should also respect the flow network i.e.
in a scenario like



Choice of $c = \frac{1}{|E|+1}$ is a perfect one

New min cut shall be atmost $\left(\frac{|E|}{|E|+1} \right)$

So, values of min cuts in original graph, shall be

$$C_{\min} + \frac{P_1}{|E|+1}, C_{\min} + \frac{P_2}{|E|+1}, \dots, C_{\min} + \frac{P_k}{|E|+1}$$

$\leq \quad \leq$

Also preserve the min cuts i.e. in the graph G' with modified capacities formed by adding $\frac{1}{|E|+1}$ min cut corresponds to min cut in the original graph with minimum number of edges.

34.5-2) Given an integer $m \times n$ matrix A and an integer m vector b , the 0-1 integer programming problem asks whether there exists an integer n -vector x with elements in the set $\{0, 1\}$ such that $Ax \leq b$. Prove that 0-1 integer programming is NP-complete (HINT: Reduce from 3-CNF-SAT)

To prove OILP as NPC:

- ⇒ Prove OILP \in NP
- ⇒ Prove OILP is NP-Hard. (Use 3-CNF-SAT)
- ⇒ To prove OILP \in NP

CERTIFICATE: An n -vector x is the certificate
it can be verified in polynomial time by checking $Ax \leq b$

 \Rightarrow OILP \in NP

- 2) To prove OILP is NP-hard.

REDUCTION: 3-CNF-SAT \leq_p OILP.

Consider an instance of 3-CNF-SAT, we show by reducing this instance in polynomial time to an instance of OILP.

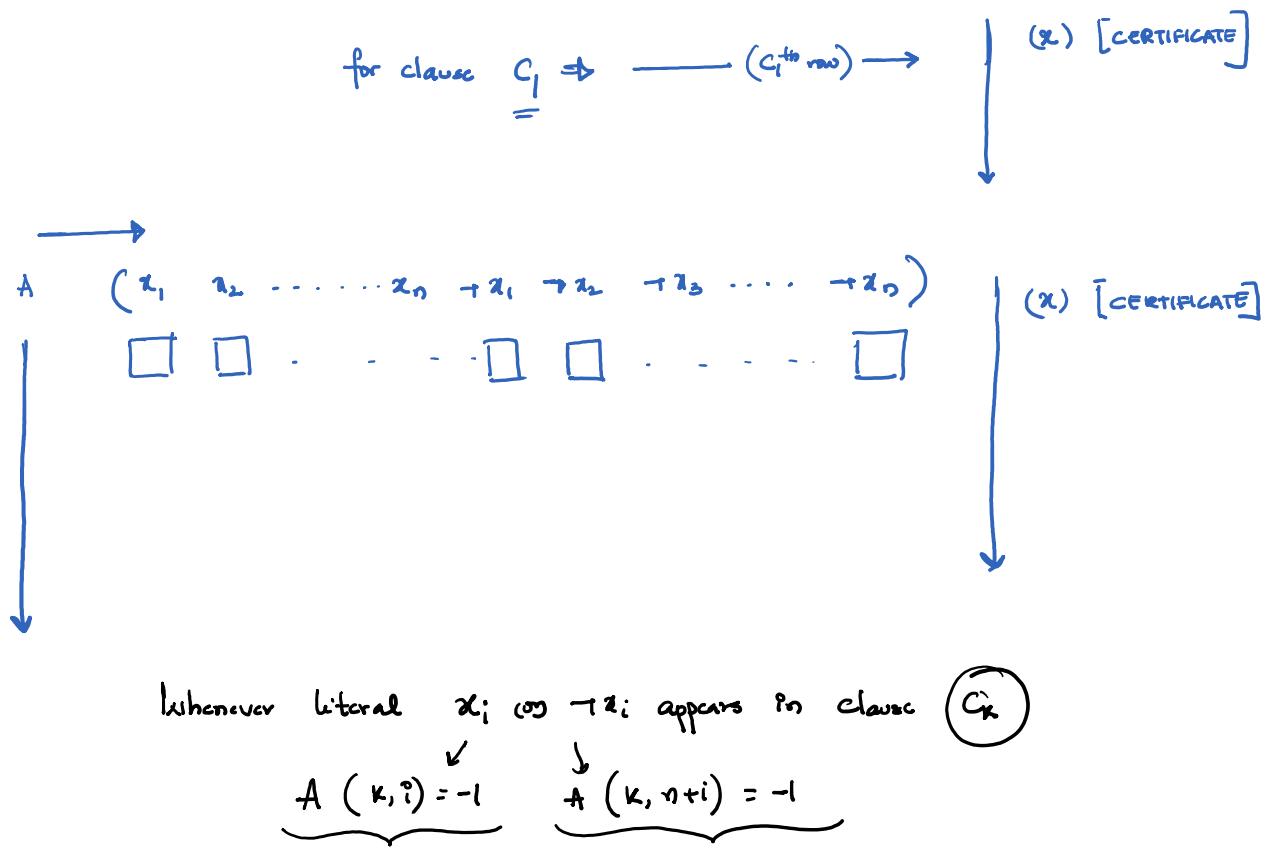
Let $\phi = C_1 \wedge C_2 \wedge C_3 \wedge \dots \wedge C_K$ where each C_i consists of 3 literals.
& with n input variables

OBSERVATIONS

① k -clauses with n -variables \rightarrow x vector can be $(2n \times 1)$ matrix.

where values of x hold $(0, 1)$ for each of $x_i^0, \neg x_i$

② Apart from positions of x_i^0 's in clause C_k , $A(k, j) = 0$ for $i \neq j$



③ We need to ensure that only one of $x_i^0, \neg x_i$ is true at any point of time. In order to impose this restriction

$$x_i^0(p_i') + \neg x_i(p_i'') = 1 \quad \rightarrow \quad [x_i(p_i') + \neg x_i(p_i'') \leq 1] \text{ (or)} \quad [x_i(-p_i') + \neg x_i(-p_i'') \leq -1]$$

These equations can also be made part of A

So, now our matrices shall be

$$\begin{array}{c|c}
 \text{A} & \\
 \hline
 \left(\begin{array}{cccccc|cccccc}
 \text{if } x_i \text{ is present} & & & & & & \text{if } \neg x_i \text{ is present} & & & & & \\
 \boxed{-1} & \text{otherwise } = 0 & & & & & \boxed{-1} & \text{otherwise } = 0 & & & & \\
 \hline
 1 & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & & & \\
 0 & 1 & 0 & \dots & 0 & 0 & 1 & \dots & 0 & & & \\
 \vdots & & & & & & & & & & & \\
 0 & 0 & 0 & \dots & 1 & 0 & 0 & \dots & 1 & & & \\
 \hline
 1 & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & & & \\
 0 & 1 & 0 & \dots & 0 & 0 & 1 & \dots & 0 & & & \\
 \vdots & & & & & & & & & & & \\
 0 & 0 & 0 & \dots & 1 & 0 & 0 & \dots & 1 & & &
 \end{array} \right)_{(k+2n \times 2n)}
 \end{array}$$

$$\begin{array}{c}
 \mathbf{x} \text{ (CERTIFICATE)} \\
 \downarrow \\
 \left(\begin{array}{c|c}
 \mathbf{x} & \mathbf{b} \\
 \hline
 n & \\
 \hline
 k & \\
 \hline
 \leq n & \\
 \hline
 n & \\
 \hline
 \geq n & \\
 \hline
 -1 & \\
 \hline
 1 & \\
 \hline
 -1 &
 \end{array} \right)_{(2n \times 1)}
 \end{array}$$

Here A, b values can be constructed in polynomial time

So, it is a polynomial Time reduction

To prove:

- ① When ϕ is satisfiable, $A\mathbf{x} \leq b$ holds.
- ② When $A\mathbf{x} \leq b$ holds, ϕ is satisfiable.
- ③ When ϕ is satisfiable, \exists some assignment vector $\mathbf{x} \in (\mathbb{R}^n)$

such that ϕ shall be satisfiable when values of \mathbf{x} are assigned.

Now construct \mathbf{x}' ($2n \times 1$) from \mathbf{x} ($n \times 1$)

$$\left. \begin{array}{l}
 \text{by } \mathbf{x}'[i] = 1, \mathbf{x}'[n+i] = 0 \text{ if } \mathbf{x}[i] = 1 \\
 \mathbf{x}'[i] = 0, \mathbf{x}'[n+i] = 1 \text{ if } \mathbf{x}[i] = 0
 \end{array} \right\}$$

To show, above formed x' holds for $[Ax \leq b]$

below $2n$ equations are valid as only one of $x'[i]$, $x'[n+i]$ is 1

So, first n equations proving ≤ 1 ✓

[next n equations proving] ≤ -1

for first k equations \rightarrow each clause C_i is satisfied so one of

its literals is positive $\Rightarrow \leq (-1)(1)$

if two literals in a clause are true then it shall be $(-2) \leq -1$ ✓

Thus first k equations also hold ✓

$\Rightarrow Ax \leq b$ [Holds]

② If $Ax \leq b$ holds, then \emptyset has a satisfiable assignment

Pick a certificate holding $\circled{Ax \leq b}$

below $2n$ equations guarantee only one of $x_i, -x_i$ is positive

above k equations:

↳ Each row guarantees the clause \textcircled{i} is satisfiable

by the picked CERTIFICATE

So, $Ax \leq b \Rightarrow \emptyset$ is satisfiable

Thus proving OILP is NP-Complete

↳ Reduction ✓ ($3\text{-CNF-SAT} \leq_p \text{OILP}$)

345 → The longest simple cycle problem is the problem of determining a simple cycle (no repeated vertices) of maximum length in a graph. Formulate a related decision problem, and show that the decision problem is NP-complete.

DECISION PROBLEM

For a graph G and an integer k , does the graph have a simple cycle of length k in the graph G .

In order to prove a problem is NPC.

→ Prove $L \in NP$

→ Prove $L' \leq_p L \dashv L' \in NP$. (can also be done

by considering reductions of an NPC problem to L).

→ To prove $L \in NP$

For the above decision problem

CERTIFICATE: List of vertices on the cycle

VERIFICATION: Check if all of them are different and $|V| = k$

Check if there is an edge b/w consecutive

vertices in the list. (CERTIFICATE)

2) To prove L is NP-hard.

METHOD:

Reduce an instance of Hamilton cycle problem (NPC)
to an instance of L in polynomial time.

for an instance of Hamilton cycle, if it can be reduced to an
instance of the problem L by considering our decision problem
of checking does the graph have longest simple cycle of length $|V|$

If Hamilton is yes \rightarrow then L is yes

if L is yes then there has to be a Hamilton path of length $|V|$
 \Rightarrow yes