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

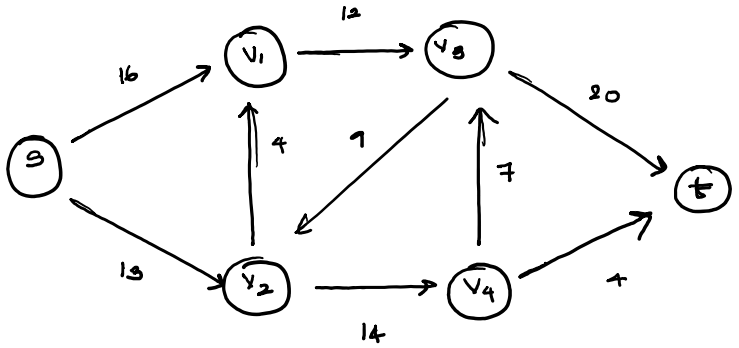
⇓

As we have $h(u) = h(v)$, re weighting does not change weights of the graph. i.e adds & subtracts [ZERO]

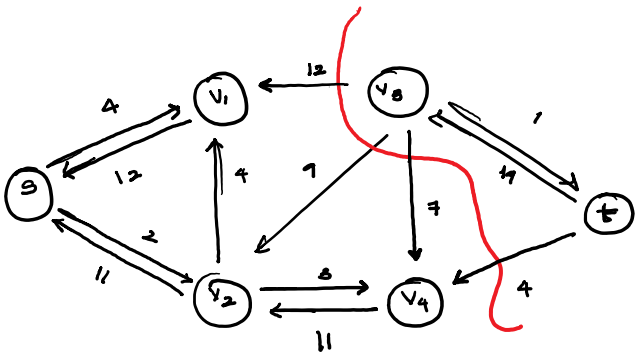
26.2-10) Show how to find a maximum flow in a network $G = (V, E)$ by a sequence of almost $|E|$ augmenting paths

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

Consider the graph :



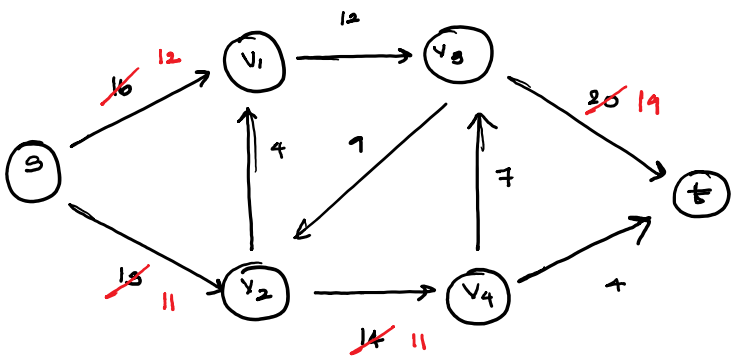
Residual network for the graph with final flows



Cut of the graph where edges from s to T are exhausted

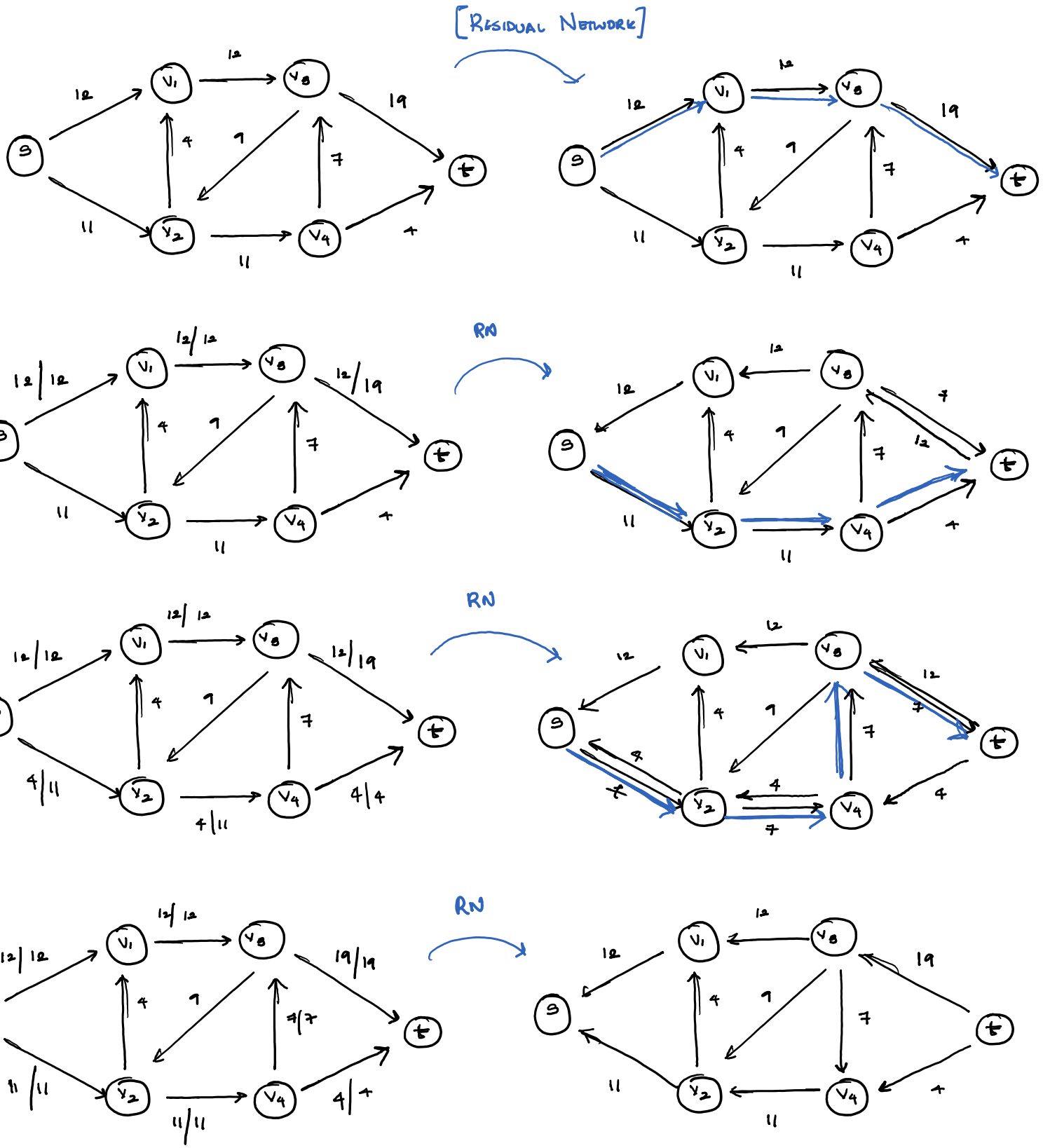
[NO FORWARD EDGE] in residual network.

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



=> With this modification there shall be no reverse edge in the final residual network.

[RUN FORD FULKERSON on the above graph & store augmenting paths]



Max |E| paths

For each iteration → one edge is exhausted & no reverse edges cancelling flow

|E| Augmenting Paths → Paths obtained here after each iterations

① As we modified capacity of each edge in original network to equal the value of its flow during max flow.

② Each augmenting path produces atleast one edge where flow equals capacity

Thus at most $|E|$ augmenting paths exist

MODIFICATION performed above prevents us from stopping the progress

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 witness smallest increase in the capacity of the cut.

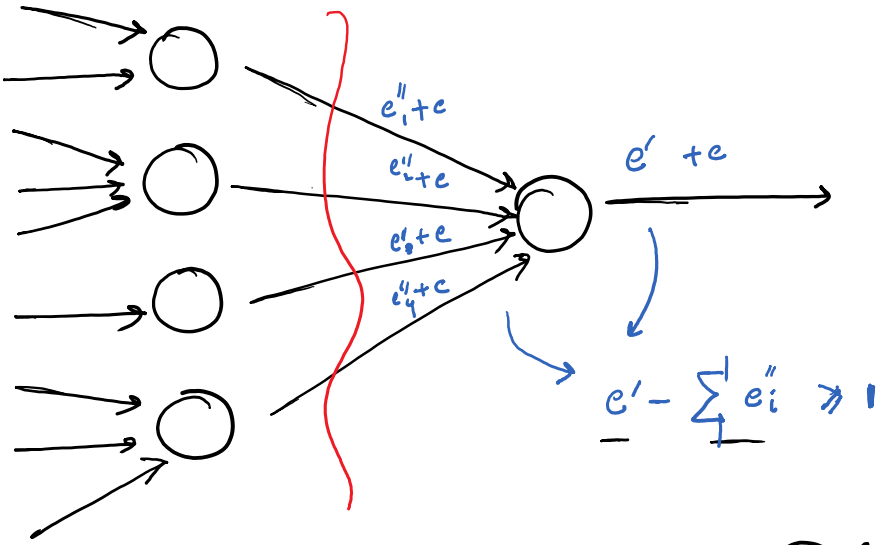
Let edges in k min cuts be

$p_1 \quad p_2 \quad \cdots \quad p_k \qquad (p_1 < p_2 < p_3 \cdots p_k)$

So, new capacities of the cuts are → $\boxed{C_{min} + p_1 e}$, $C_{min} + p_2 e \cdots C_{min} + p_k e$

↓ [MINIMUM CUT]

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



ϵ added should be < 1

Choice of $\epsilon = \frac{1}{|E|+1}$ 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

$C_{min} + \frac{p_1}{|E|+1}$, $C_{MIN} + \frac{p_2}{|E|+1}$, ... $C_{MIN} + \frac{p_k}{|E|+1}$

$<$ $<$ $<$

Also preserves 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:

1) Prove OILP $\in$ NP

2) Prove OILP is NP-Hard. (Use 3-CNF-SAT)

1) 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.

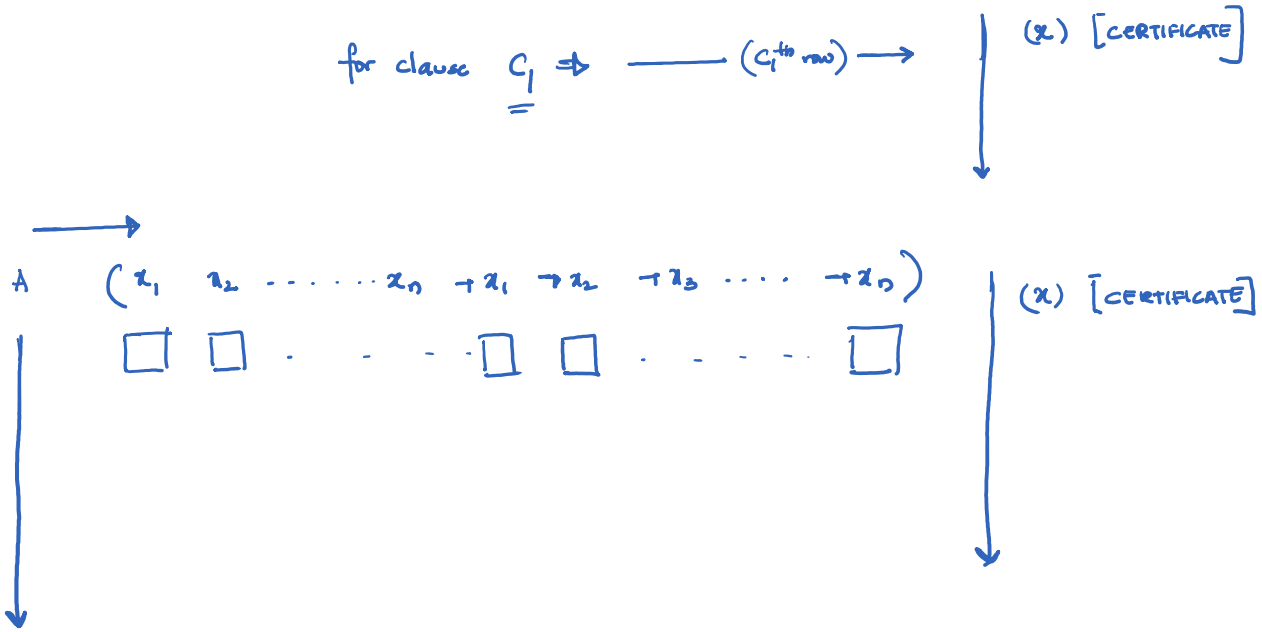
Let $\phi = C_1 \wedge C_2 \wedge C_3 \cdots C_k$ where each C_i consists of 3 literals & with n input variables

OBSERVATIONS

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

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

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



Whenever literal x_i (or) $\neg x_i$ appears in clause C_k

$A(k,i) = -1$ $\quad$ $A(k, n+i) = -1$

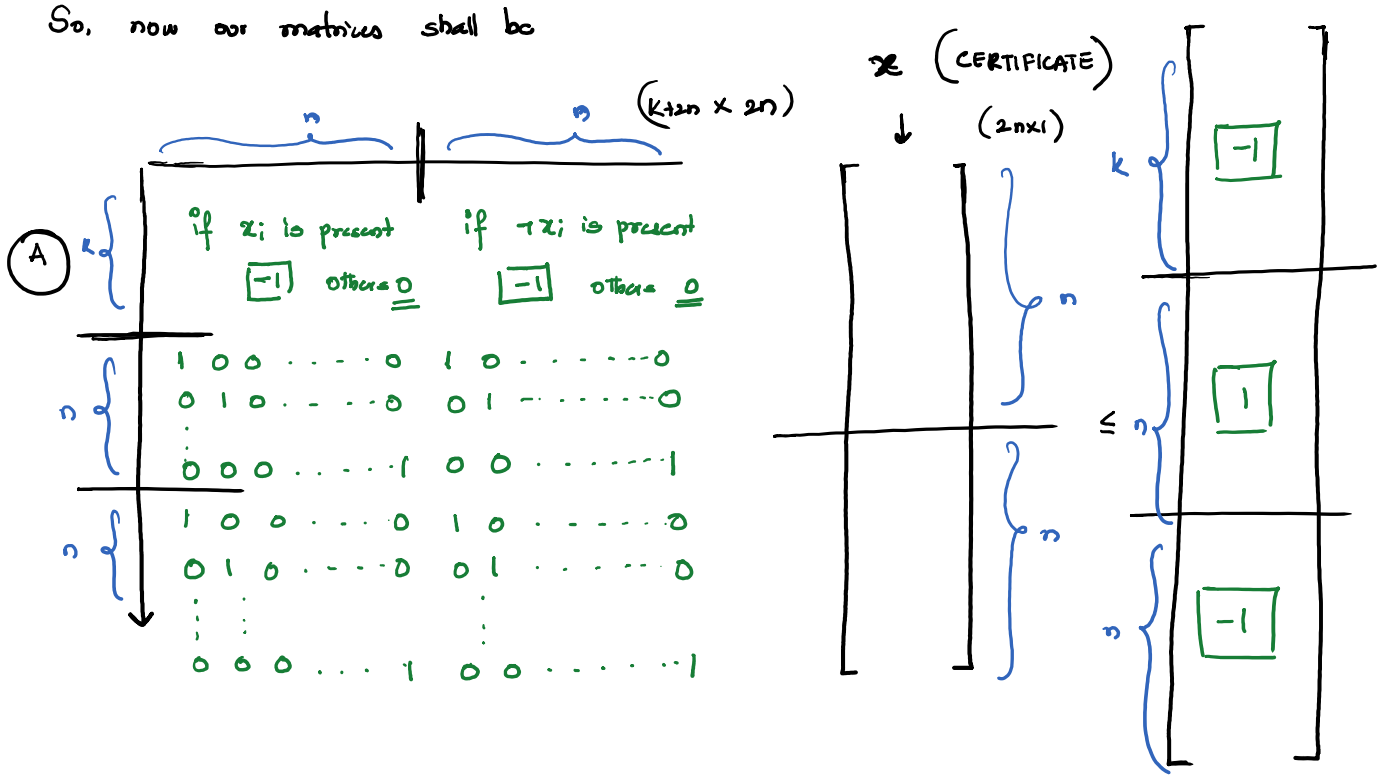
③ We need to ensure that only one of x_i, $\neg x_i$ is true at any point of time. Inorder to impose this restriction

$$x_i(p_i') + \neg x_i(p_i'') = 1$$

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

These equations can also be made part of (A)

So, now our matrices shall be



Here A, b values can be constructed in polynomial time

So, it is a polynomial time reduction

To prove :

① When ϕ is satisfiable, $Ax \leq b$ holds.

② When $Ax \leq b$ holds, ϕ is satisfiable.

① When ϕ is satisfiable. $\exists$ some assignment vector $x \rightarrow (n \times 1)$

such that ϕ shall be satisfiable when values of x are assigned.

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

by $x'[i] = 1$, $x'[n+i] = 0$ if $x[i] = 1$

$x'[i] = 0$, $x'[n+i] = 1$ if $x[i] = 0$

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 → each clause C_i is satisfied so one of its literals is positive ⇒ $\leq (-1)(1)$

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

Thus first k equations also hold ✓

⇒ $Ax \leq b$ [HOLDS]

(2) If $Ax \leq b$ holds, then ϕ has a satisfiable assignment

Pick a certificate holding $Ax \leq b$

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

above k equations

↳ Each row guarantees the clause (i) is satisfiable by the picked CERTIFICATE

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

Thus proving OILP is NP-Complete

↳ REDUCTION ✓ (3-CNF-SAT $\leq_p$ OILP)

34.5-7) 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.

Inorder to prove a problem is NPC.

1) Prove $L \in NP$

2) Prove $L' \leq_p L$ $\forall$ $L' \in NP$. (can also be done by considering reduction of an NPC problem to L).

1) 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, G 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 → then L is yes

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

=> yes