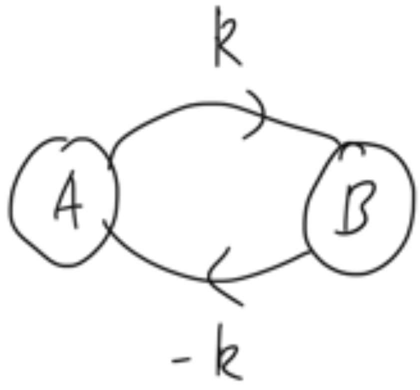
FA 12

Q1.

1.a.

Base Case



$n := 2$ edges

The zero weight cycle for 2 vertices is a union of zero weight cycles i.e. itself.

Inductive hypothesis:

Suppose that for zero weight cycles of edges less than or equal to n; we can make breakdown into a union of simple cycles for a non simple cycle.

Inductive Step:

Consider the process of creating a zero weight cycle of $(n+1)$ from a zero weight cycle of n edges.

cycle ... edges

Let C be a zero weight cycle of edges n. For a zero weight cycle of C' of edges $(n+1)$.

Let $C' \equiv V_1 - V_2 - - - \quad - V_1$ of size $(n+1)$.

If there is cycle Z_j of length $k \leq n$ i.e $V_j - V_{j+1} — V_j$, then, by our inductive assumption, $V_j - - V_j$ can be broken down as a union of simple cycles. $\{S_j\}$ let $Z_1, Z_2 - Z_k$ be such cycles.

→ Now, we cannot have a combination of the form $Z_j \; x_1 — x_k \; Z_i$ such that $x_1 \neq x_k$.

Since we want to go back to the source through a connected path, we need to start and end at the same node for our traversal such that our path corresponds to a zero weight cycle. Otherwise, the traversal sum won't be zero.

In the constructive case,
Consider a zero weight cycle of n edges to

be $a_1 - a_n - a_1$. for some path $a_i - - - a_j$
We can add an add an edge of weight of magnitude $(-r)$ such that we get a non simple cycle with $(n+1)$ edges. i.e.

$$- \{a_i \text{—} \quad a_i\} + \{a_i - a_j, e(r)\}$$

$e(r)$ goes from j to i with weight $(-r)$

$$\text{Union breakdown} \equiv \underbrace{S_n}_{\substack{\text{from} \\ \text{inductive} \\ \text{assumption}}} \cup \underbrace{\{a_i - - - a_j - a_i\}}_{\text{Simple cycle}}$$

For the general case,

$$S_{n+1} = \underbrace{\bigcup_{i=1}^{k} S_i \, (i<n)}_{\substack{\text{from inductive} \\ \text{assumption}}} + \{\text{outer simple cycle if it exists}\}$$

composite union obtained

Q1. b.

For $a \in V$, $b \in V$,

a, b are related if they lie on some cycle of zero weight or $a = b$.

1. Reflexivity :

a is related to a. Since $a = a$.
Equality of vertices u, v is a sufficient condition for the relation R to hold on (u, v) as given in the question.

2. Symmetry :

Consider $a \overset{0}{=} b$, a is related to b.

That would mean a and b lie on some zero weight cycle.
Let that cycle be C. $\{ c_1 - c_2 - a \cdots - b - c_1 \}$

Now, b and a also also lie on C, since the cycle under consideration can be the same.

$\therefore$ If $a \overset{0}{=} b$ then $b \overset{0}{=} a$

$\therefore$ If $a \overset{0}{\equiv} b$, then $b \overset{0}{\equiv} a$

Similarly,

$$\text{If } b \overset{0}{\equiv} a, \text{ then } a \overset{0}{\equiv} b$$

The case of symmetry holds in R.

3. Transitivity:

Given,

$a \overset{0}{\equiv} b$, let the cycle be C_1

$b \overset{0}{\equiv} c$, let the cycle be C_2

We construct a cycle C' as follows:

Start with b.
Traverse C_1 and back to b. $\Sigma W_1 = 0$
Traverse C_2 and back to b. $\Sigma W_2 = 0$

In the general case, start with some node in C_1; go to b. Now, traverse the cycle C_2, back to b, go through the remaining C_1 back to the original node. In the composite path, each weight is added once.

$$C' \equiv \sum w_1 + \sum w_2 = 0$$

$\therefore C' \equiv C_{1\{b\}} \cup C_{2\{b\}}$ is also a zero weight cycle

Cycle C' contains both a and c.

(a, c) are Related.

$\therefore$ Transitivity holds true for our relation.

Q1. c.

Flloyd - Warshall (W)

{

n = W. rows

$D^{(0)} = W$

for $i = 1$ to n .

$$D^{(0)}(i,i) = \infty$$

for $k = 1$ to n :

let $D^k = (d_{ij}^k)$ be a new $n \times n$ matrix

for $i = 1$ to n :

for $j = 1$ to n :

$$d_{ij}^{k} = \min \{ d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1} \}$$

for $i := 1$ to n :

if $d_{ii}^{n} = 0$:

print(i)

7

→ For the flloyd Warshalls algorithm, we assign the diagonal d values to infinity, we want to find $d_{ii}^{k} = 0$ values such that there is a non trivial zero weight cycle.

→ The flloyd Warshalls algorithm will have three loops over n.

→ We know that the flloyd Warshall's algorithm is correct. When we find $d_{ii}^{k} = 0$, we know that there is a cycle from i to i.

→ In the last loop, if d_{ii}^{n} is zero, we print i. If for some i, d_{ii}^{k} is zero then for all increasing number of nodes under consideration, we will have d_{ii}^{k} values to be zero as negative weight cycles are not allowed and that is a min a cycle can be.

→ The three loops of flloyd Warshalls will dominate,

$$T(n) = \Theta(n^3)$$

Q1.d.

- We have $D^{(n)}$ from Floyd Warshall's.
- Our Vertices are $V_1, V_2 \ldots V_j \ldots V_N$.

- for vertex j, if for some vertex i, if $d_{ij} = -d_{ji}$, then, i and j are part of some zero weight cycle.
  $\because$ $d_{ij} + d_{ji} = 0$ ($i \to j$ and $j \to i$) is a zero weight cycle.

- If i and j are part of some zero weight cycle, $d_{ij} + d_{ji} = 0$.

- Let us say $d_{ij} > 0$. $\therefore$ We know d_{ji} must be less than zero, we are looking at a shortest distances and i, j are part of a zero weight cycle. d_{ji} through the path is negative. must be $-d_{ij}$ for zero weight. If d_{ji} less than $-d_{ij}$, there will be a negative weight cycle which can't be present.
- If i, j is a part of a cycle, then j, i

will also be a part of the cycle, from symmetry.

- From transitivity, if d_{ij} is part of a cycle, and, d_{jk} is part of a cycle, then d_{ik} will be part of some cycle.
- We can further find components in the cycles of zero weights.

```
Find_equivalance_class ( Dn )
{
    for j from 1 to n:
        for i from 1 to n:
            if (dij = - dji and i!=j)
                # i,j are present in the
                  same zero weight cycle.
```

$M[i,j] = 1$

DFS(M)

}

→ Wrt to matrix M, apply DFS.

→ Start with some vertex. Apply DFS. Now, at the termination of the recursion of that DFS, all vertices covered will be a part of one equivalence class.

→ For each of these vertices, find the min value and assign it to the equivalance class variable.

→ Repeat the DFS procedure for other unvisited vertices.

$$T(V, \mathcal{E}) = \Theta(V^2)$$

→ We are parsing over the matrix of size V^2. DFS takes $\Theta(V+\mathcal{E})$ time.

Q1.e.

We know that,

h, i lie on some cycle,

$\delta(h,i) = -\delta(i,h)$

j, k lie on some cycle,

$\delta(k,j) = -\delta(j,k)$

There is a path $h - - i - - j - - k$

$\therefore \delta(h,k) \leq \delta(h,i) + \delta(i,j) + \delta(j,k)$

→ If $\delta(h,k)$ was greater than the right hand side, $\delta(h,k)$ would have assigned the value

If $\delta(h,k) = \delta(h,i) + \delta(i,j) + \delta(j,k)$ our proof is done.

Suppose, $\delta(h,k) < \delta(h,i) + \delta(i,j) + \delta(j,k)$

$$\delta(h,k) < -\delta(i,h) + \delta(i,j) - \delta(k,j))$$

$$\delta(i,h) + \delta(h,k) + \delta(k,j) < \delta(i,j))$$

Now, $\delta(i,h) + \delta(h,k) + \delta(k,j)$ is a path from i to j. It cannot be less than $\delta(i,j)$. $\therefore$ We have a contradiction,

$\therefore$ $\delta(h,k) \not< \delta(h,i) + \delta(i,j) + \delta(i,k)$ and our assumption is wrong.

$\therefore$ $\delta(h,k) = \delta(h,i) + \delta(i,j) + \delta(i,k)$

Q3.

3.a.

1 – $\Theta(1)$

2 –

2.a – $\Theta(V)$ # one pass over vertices

2.b - $\theta(V)$ # for every vertex V

2.c - $\theta(V+\mathcal{E})$ # DFS-Visit invoked

3. $\theta(V)$ # For every vertex V

4. $\theta(\mathcal{E})$ - # for a run of Bellman ford.

$\therefore$ Asymptotic running time $= \theta(V+\mathcal{E})$

(2.a)

- For the right answer, we will have $v.\pi \neq nil$.

(4). For any edge of G, if a relaxation is possible, we will have that our answer is not correct; since for the correct answer, relaxation is not possible, our algorithm will detect that.

(3) • For the right answer, $v.d = v.\pi.d + w(u,v)$

Our algorithm will check for that.
This condition must be satisfied since the equality forms the basis of assignment.

• 2.b is a check for reachability. For every v which has been assigned a parent, we make a graph G', we ensure using a DFS-Visit call that every vertex v such that $v.d < \infty$ is reachable from s.

• 2.c is for ensuring that if $\delta(s,v)$ is ∞ and $v.d < \infty$, we are wrong.
Or, if $\delta(s,v) < \infty$ and $v.d$ is ∞, we are wrong, since v is actually reachable.

3.b

1. $\delta(s,v) = \infty$ but $v.d < \infty$

$\to$ 2.c will catch; if we reached v.d and v is not reachable.

- DFS-Visit will reach v.d if $v.d < \infty$.
- We shouldn't be able to reach v.d through the DFS-Visit and hence our 2.c condition is violated.

2. $\delta(s,v) < \infty$ but $v.d = \infty$

$\to$ 2-c will take care. Given that $v.d = \infty$

$\to$ Since $\delta(s,v) < \infty$, we will be able to reach v from s. Therefore, this will violate the condition of 2.c.

There will be a path from s to v since $\delta(s,v)$.

3. $\delta(s,v) < v.d < \infty$

$\to$ Let us say that u is the parent of v.

$\to$ We get that $\delta(s,v) = \delta(s,u) + w(u,v)$

Suppose $\delta(s,u)$ is correctly assigned.

→ Then, during one pass of Bellman ford, there will be relaxation. If there is relaxation, we will know that our algorithm is wrong.

$v.d > \delta(s,u) + w(u,v)$ at the time of relaxation of u. ∴ The if statement will be true and the relaxation will proceed.

- Step 4 will detect.

- If u.d is wrong, we can extend the same argument back to the source and the error in the source will be detected by 1.

4. $v.d < \delta(s,v) < \infty$

Let u' be the real parent of v.
let u' be the one assigning v.

$$v.d = u'.d + w(u',v)$$

$$\rightarrow \delta(s,v) = \delta(s,u') + w(u',v)$$

$$\rightarrow v.d < \delta(s,v)$$

$\rightarrow \quad \therefore \; u'.d + w(u', v) < \delta(s, v)$

$$\therefore \; u'.d < \delta(s, v) - w(u', v)$$

$$u'.d < \delta(s, u')$$

$\rightarrow$ Extending the argument back to the source,

$$s.d < \delta(s, s)$$

But s.d is checked at 1 and we get a wrong answer.

for some other u'' assigning v.

$$v.d = u''.d + w(u'', v)$$

If $u''.d$ is correct,

$$v.d > \delta(s, v)$$

Since u'' is the wrong parents.

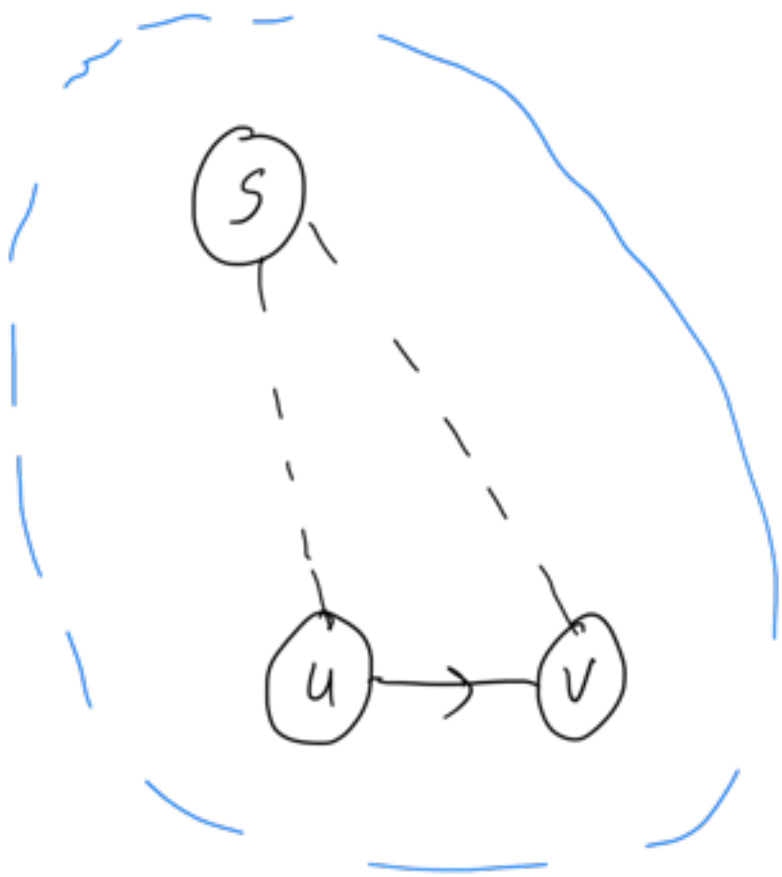
This violates our assumption.

If $u''.d$ is incorrect, $u''.d$ must be less than $\delta(s, u'')$ for $v.d$ to be less than $\delta(s,v)$ and we can extend the argument backwards to the source as before.

Q3. c.

→ Now, we have $v.d$ as $\delta(s,v)$ correctly assigned.
→ We know the correct paths from s. let u, v be reachable from s.

→



Let the reweighting be as follows,

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

Now,
We say,

$$h(u) = \delta(s, u)$$

we have $\delta(s,u)$, $\delta(s,v)$ values with us.

$$h(v) = \delta(s, v)$$

We have,

$$\delta(s,u) + w(u,v) \geq \delta(s,v)$$

else $\delta(s,v)$ will be assigned the value of $\delta(s,u) + w(u,v)$ and won't be our original $\delta(s,v)$. We know that we have the correct values of $\delta(s,v)$

$$\therefore \delta(s,u) + w(u,v) - \delta(s,v) \geq 0$$

$$\therefore \hat{w}(u,v) \geq 0$$

$$w_e'(u,v) = w_e(u,v) + h(u) - h(v)$$

Q3.d.

let $P \equiv u_1 - u_2 - \cdots - u_n$

$$\hat{w}(P) = w(u_1, u_2) + h(u_1) - h(u_2)$$

$$+ \; w(u_2, u_3) + h(u_2) - h(u_3)$$

$$\vdots$$

$$+ \; w(u_{n-1}, u_n) + h(u_{n-1}) - h(u_n)$$

Adding them all together, we get,

$$\hat{w}(P) = w(P) + h(u_1) - h(u_n)$$

$\therefore$ The min cost in path P is not affected by the reweighting as $h(u_1)$ and $h(u_n)$ do not depend on the path.
So, if a path has a different sum of weights in some path in G, it will also have a different sum of weights in some path in G' after reweighting.

3.(e)

→ Consider the case when s' is connected from s.

→ Now, if k is connected from s', it is also connected from s.

→ For vertices connected through s, we have $\delta(s,v)$ defined.

→ Now, if we run Dijkstra's from s', we will have all weights $w'(u,v)$ in G' as positive, Dijkstra's will give us the correct result for $w'(P)$.

We know,

$$w'(P) = w(P) + h(u_1) - h(u_n)$$

$$\therefore w(P) = w'(P) - h(u_1) + h(u_n)$$

Having $w'(P)$, we can get to $w(P)$ using the previous equation.

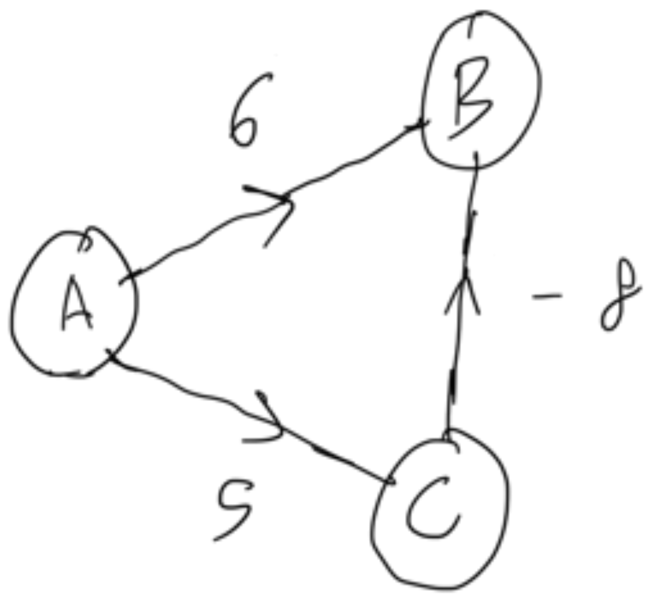
Hence we get $w(p)$.

$$T(V,E) \equiv T(\text{Dijkstra's})$$

$$\equiv \theta(V \log V + E)$$

Q2.

a.



$S = \phi$

$Q = \{A, B, C\}$
$\quad\;\; 0 \;\; \infty \;\; \infty$

$S = A$

$Q = \{B, C\}$

$S = \{A, B\}$

$Q = \{C\}$

$S = \{A, B, C\}$

$Q = \{\ \}$

$A - C : 6 \leftarrow c.d$

But, $A - B - C : -2 \leftarrow \delta(a, c)$

$\therefore$ We have a contradiction in the standard Dijkstra's.

Q2.

b.

Consider the number of edges in some path from s to v and the corresponding predecessor sub graph.

In the base case, $E = 0$ or $E = 1$

for $E = 0$, $\delta(s, s) = 0$ and won't be updated later.

for $E = 1$, $\delta(s, u) = w(s, u)$

As s is the predecessor, then, $\delta(s, u) = w(s, u)$

u.d after being assigned as $w(s, u)$, all other potential assignments will be greater than $\delta(s, u)$ so there won't be any assignment.

For v, let the number of edges be E.

We induct on the longest path from s to v.

Now, based on the inductive hypothesis, let there be finite updates for $\delta(s, u)$ such that number of edges in s to u is less than E.

for the parent x of v, if the number of edges to v is E, number of edges to parents will be less than equal to E.
So our inductive hypothesis can hold.

Now,

v has finite parents. Let them be the set X. Since each of it's parents are updated a finite number of times, v itself will be updated a finite number of times based on the permutations of the parent values.

$\therefore$ The algorithm will terminate for v.

Q2.c.

We show that during the execution of Dijkstra's, $\delta(s,v) \leq v.d$ and at termination $\delta(s,v) = v.d$.

For the Base Case,

$$\delta(s,s) \leq s.d = 0$$

Assume that for u which is an ancestor of v, $\delta(s,u) \leq u.d$ and after termination $\delta(s,u) = u.d$.

Now, while updating $v.d$, if v is discovered freshly,

$$v.d = u.d + \omega(u,v)$$

Triangle inequality,

$$\delta(s,v) \leq \delta(s,u) + \omega(u,v)$$

$$\delta(s,v) \leq u.d + \omega(u,v)$$

$$\delta(s,v) \leq v.d$$

If u is the correct parent, $\delta(s,v) = v.d$ and there will not be further updates as we have the shortest path.

Now, if v has been discovered earlier, v is in S. If we find a path to v such that $v.d > u.d + \omega(u,v)$, we we will take v into Q, update the path for v.d and use the new path for propogation in further steps.

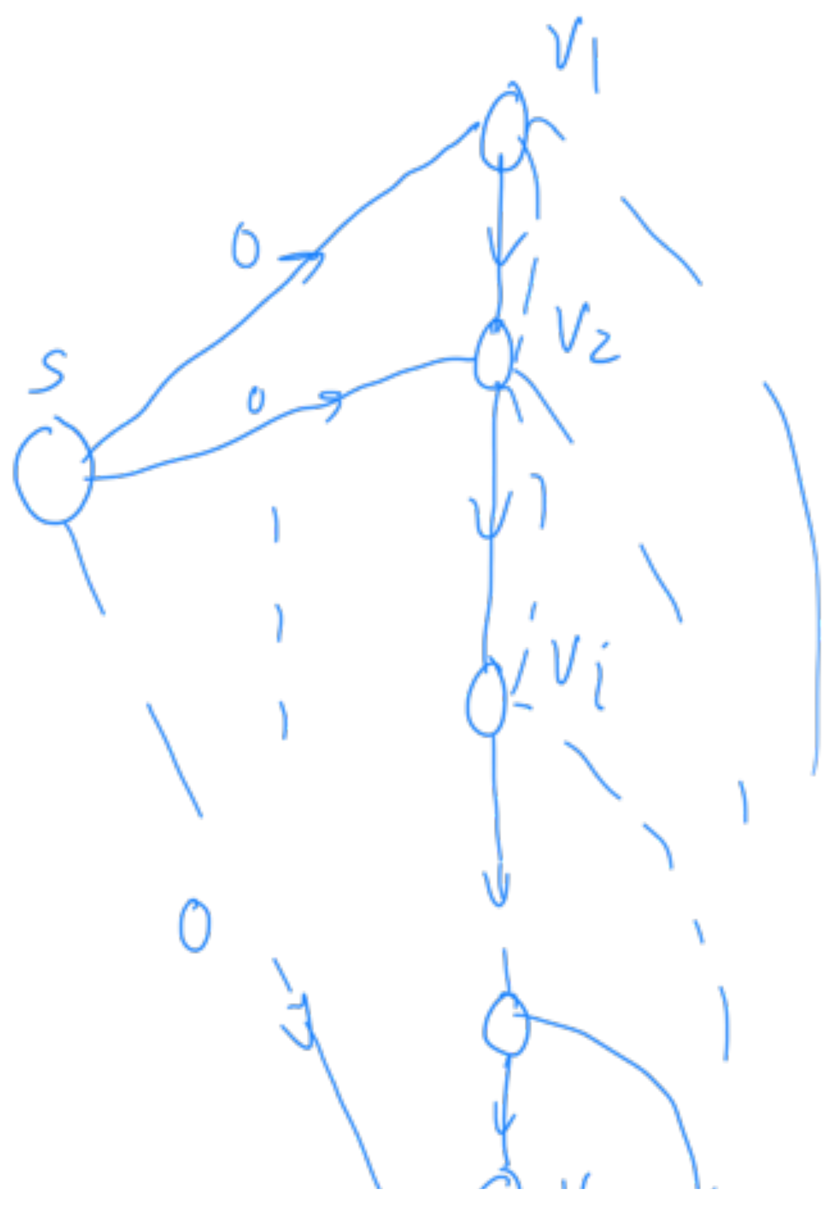
→ This is where 12 will be used in our proof.

→ From the path relaxation property, if $\delta(s,u)$ is known and we then relax (u,v), we will obtain the correct distance to $\delta(s,v)$.

→ Even if v is in S, if at the time of exploration of u, $v.d$ is to be updated, we can insert a new value of $v.d$ into Q and obtain the correct value of $v.d$.

→ We know from our inductive hypothesis that u terminates to the correct value of $\delta(s,u)$. Therefore, at the time of exploration for u, $\delta(s,v)$ is obtained as discussed.

Q2. d.



After the dequeuing of S,
$v_1, v_2, \ldots, v_n$ will be in Q.

Say we dequeue v_n. There are no outgoing edges.

→ We then put v_n in S.
We dequeue v_{n-1}, v_n will be updated and put into the queue again. v_{n-1} is now in S.

→ We dequeue v_n again.

$$-\frac{1}{2^{n-1}} < -\frac{1}{2^n}$$

→ We now dequeue v_{n-2}, both v_{n-1} and v_n will be updated and put into the Queue.
We now have to dequeue v_{n-1} and v_n again.

→ For some v_j, if we update $v_{j+1}, \ldots, v_n$, we will have to add $v_{j+1}, \ldots, v_n$ to the Queue. We will have to then dequeue these and check for every vertex adjecent to each of these vertices. As we dequeue a vertex, the subsequent checking be equivalent to the time for p.

Consider the number of updations of v_n.
Consider v_j being processed.

$$(t_n)_j = 1 + \sum_{i=j+1}^{n} (t_n)_i$$

$$(t_n)_1 = 1 + \sum_{i=2}^{n} (t_n)_i$$

For j, we update v_n wrt all vertices after j i.e. $j+1$ to n, and then one another update wrt j.

The time complexity of the algorithm will be bounded by the number of updations of t_n; Similar summations can be written for v_{n-1}, v_{n-2} etc

$$T(1)_n = 1 + \sum_{i=2}^{n} T(i)_n$$

$T(1)_n$ is the number of updations of v_n

$T(1)$ is the number of updations to v_n wrt to v_1 in the worst case.

$$\text{Time (Algorithm)} \geq T(1)_n$$

Let $T(1)_n$ be written as $S(n)$

$$S(n) = 1 + \sum_{i=2}^{n-1} S(i)$$

Assume, for $i<n$, $S(i) = \theta(2^i)$

Guess than verify
Inductive hypothesis

Base Case:

$S(1) = 1$ (V_n updated wrt V_n once)

$$S(n) \equiv 1 + \sum_{i=2}^{n-1} S(i)$$

$$\equiv \sum_{i=1}^{n-1} 2^i$$

$S(n) = 2^n$

$S(n)$ we know lower bounds $T(n)$.
time complexity of algorithm.

$\therefore T(n) = \Omega(2^n)$