Assignment 11.

Q 1.

(1.a)

- Let $v_0 \to v_1 \to \dots \to v_k$ be an arbitary cycle Z in P such that v_0 has the lowest index in the cycle.

$e(v_0, v_1)$ has to be selected wrt either v_0 or v_1 else it wouldn't be present in P.

Suppose $e(v_0, v_1)$ was selected wrt v_0.

$\to$ For v_1,

- $w(v_1, v_2)$ cannot be greater than $w(v_0, v_1)$ if that was the case, at the time of processing v_1, $e(v_0, v_1)$ would be selected instead of $e(v_1, v_2)$.

- if $w(v_0, v_1) = w(v_1, v_2)$, then $v_0 < v_2$, as our problem defines.

$\therefore \quad w(v_0, v_1) \geq w(v_1, v_2)$

Similarly,

$$w(v_1, v_2) \geq w(v_2, v_3)$$

if $w(v_1, v_2) = w(v_2, v_3)$

then $v_1 < v_3$

Now we have,

$$w(v_0, v_1) \geq w(v_1, v_2)$$

$$w(v_1, v_2) \geq w(v_2, v_3)$$

$$\vdots$$

$$w(v_i, v_{i+1}) \geq w(v_{i+1}, v_{i+2})$$

At vertex v_k,

$$w(v_{k-1}, v_k) \geq w(v_k, v_0)$$

From transitivity of the inequality,

$$w(v_0, v_1) \geq w(v_k, v_0)$$

However, at the time of selection for v_0, if $w(v_k, v_0)$ was lesser than $w(v_0, v_1)$, then $e(v_k, v_0)$ would have selected by our algorithm instead of $e(v_0, v_1)$. Without $e(v_0, v_1)$ the cycle wouldn't exist in first place. Therefore we get a contradiction. -1

For the case of equality in the inequalities, by the transitivity of the conditions,

$$w(v_0, v_1) \geq w(v_1, v_2) \_\_\_ \geq w(v_k, v_0)$$

for equality of,

$$w(v_0, v_1) = w(v_k, v_0)$$

$$w(v_0, v_1) = w(v_1, v_2) \ldots$$

which would imply, as discussed,

$v_0 < v_2$

$v_1 < v_2$

$$v_{k-1} < v_0$$

However, v_0 is the vertex with the smallest index. We again recieve a contradiction. - 2

$\therefore$ From Points 1 and 2, such a cycle cannot exist in P.

Suppose $e(v_0, v_1)$ was selected wrt v_1,

$$w(v_0, v_1) \leq w(v_1, v_2)$$

$$w(v_1, v_2) \leq w(v_2, v_3)$$

$$w(v_2, v_3) \leq w(v_3, v_4)$$

$$\vdots$$

$$w(v_{k-2}, v_{k-1}) \leq w(v_{k-1}, v_k)$$

From our computations, (v_0, v_1) is selected for v_1, .. (v_{k-1}, v_k) is selected for v_k.

$\therefore \quad w(v_k, v_0) \geq w(v_k, v_{k-1})$

Putting the inequalities together,

$$w(v_k, v_0) \geq w(v_k, v_{k-1}) \geq \_\_ \; w(v_1, v_0)$$

$$w(v_k, v_0) \geq w(v_1, v_0) \quad - \alpha$$

For v_0,
$(v_0, v_1) \in P$, selected wrt v_1
$(v_0, v_k) \in P$

(v_{k-1}, v_k) selected wrt v_k.

$\therefore \quad w(v_k, v_0)$ selected wrt to v_0.

$$w(v_k, v_0) \leq w(v_0, v_1) \quad - \beta$$

From α, β, we get a contradiction.

$\therefore$ Such a cycle cannot exist.

$\rightarrow$ In case of equality, we get a contradiction from the strict inequality of the indices as seen previously.

The ... as seen previously.

Q1.

(b) $\rightarrow$

Let $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \dots v_k$ be a path p in P.

Let $e(v_k, v_0)$ be the edge added to cause a cycle.

$e(v_0, v_1)$ will have to be selected either wrt v_0 or v_1; else it wouldn't be in P.

Suppose $e(v_0, v_1)$ was selected wrt v_1.

Then,

$$w(v_1, v_2) \geq w(v_0, v_1)$$

else $e(v_1, v_2)$ would have been selected wrt v_2.

Similarly

$$w(v_2, v_3) \geq w(v_1, v_2)$$

→ else $e(v_2, v_3)$ would have been selected wrt v_2 and $e(v_1, v_2)$ wouldn't be in P as we have selected $e(v_0, v_1)$ wrt v_1.

$$\vdots$$

$$w(v_{k-1}, v_k) \geq w(v_{k-2}, v_{k-1})$$

→ $e(v_{k-1}, v_k)$ is selected wrt v_k.

→ We can follow that $w(v_{k-1}, v_{k-2})$ is selected wrt to v_{k-1}.

→ If there was some other edge selected wrt to v_k, $e(v_{k-1}, v_k)$ would not have been selected to be present in P.

let e' be the edge selected wrt to

to v.

$\rightarrow$ $e' \leq w(v_0, v_1)$ else $e(v_0, v_1)$ would be selected for v_0.

Now, addition of $e(v_k, v_0)$ causes a cycle.

$\rightarrow$ $w(v_k, v_0) \geq w(v_k, v_{k-1})$

else,
$e(v_k, v_0)$ would have been selected for v_k.

$\rightarrow$ We have,

$$w(v_k, v_0) \geq w(v_k, v_{k-1}) \text{ --- }$$
$$\geq w(v_1, v_0) \geq e'$$

$\therefore$ The largest edge is selected to complete the size. //

Now, if $e' \equiv e(v_0, v_1)$ i.e. $e(v_0, v_1)$ was selected wrt to v_0,

$$w(v_0, v_1) \geq w(v_1, v_2)$$

else $w(v_0, v_1)$ would have again been selected wrt to v_1. In case of equality,
$v_0 < v_2$

We get,
$$w(v_1, v_2) \geq w(v_2, v_3)$$

$$\vdots$$

$$w(v_{k-2}, v_{k-1}) \geq w(v_k, v_{k-1})$$

Now, addition of edge (v_k, v_0) causes a cycle.

$\rightarrow \quad w(v_k, v_0) \geq w(v_0, v_1)$

else, $e(v_k, v_0)$ would have been selected for v_0.

We have,

$w(v_k, v_0) \geq w(v_0, v_1) \geq w(v_1, v_2) \geq w(v_2, v_3)$

$\ldots \geq w(v_k, v_{k-1})$

$\therefore$ The largest edge is selected to form the cycle. //

Q1.

c.

$C[1, \_\_\_, n]$ # Global Array.

variable repeat stores the repeat edge if any.

repeat = Nil

Compute P(G)

{

for all $e(u,v)$ such that $e \in E$:

if $C[u] == -1$:

$C[u] = v$

```
if C[v] == -1:
        C[v] = u


if w(u,v) > w(u, C[u]):
    C[u] = v
    if C[v] == u:
        repeat = (u,v)

if w(u,v) > w(v, C[v]):
    C[v] = u
    if C[u] == v:
        repeat = (u,v)

}

Find P()
{
```

```
{
    repeat_counted = False
    for i from 1 to n:
        if (i, C[i]) == repeat and
                    repeat_counted == false:
            repeat_counted = True
            P.append (i, C[i])

        P.append (i, C[i])
    return P
}
```

As we iterate through the edges, for every new edge, we update the C values.

If there exists an edge (u,v) such that C[u] = v and C[v] = u, we make sure to mark it as on the repeat parameter.

When we have filled in the C values, We do one parse over the vertices, counting if repeat edge once, and obtain P.

We fill in C in the first algorithm. The first algorithm is one parse over the edges, The second algorithm to obtain P from establishing the array C is one parse over C which corresponds to a pass over the number of vertices.

$$T(V, E) = \Theta(V + E)$$

Q1. d.

Lemma:

- Swap (i, T, C) maintains an MST such that the edge $(i, C[i])$ is included in the MST.

- At the time of execution of Swap (i, T, C), Swap $(i-1, T, C)$ has been executed previous and thus based on our inductive assumption, edge $(i-1, C[i-1])$ and previous are included in the MST until now.

Let p be the path from i to $C[i]$.
Let (i,j) be the first edge in that path.

$$MST(T) \equiv$$

$$w(T) = w(p) + w(p')$$

$$w(T) = w(i,j) + w(p\_rest) + w(p')$$

Now, for i,

$$w(i, C[i]) \leq w(i,j)$$

based on how we select in P.

If $w(i,j) < w(i, C[i])$,

i would have selected j as it's $C[i]$ value hence violating our assumption. $\therefore$ It cannot be that $w(i,j) < w(i, C[i])$

If we swap (i,j) with $(i, C[i])$ we will still have a tree.
We know that originally T is a tree, suppose introduction of $(i, C[i])$ produces a cycle, since $e(i,j)$ is removed, the cycle cannot be along path p. It must be through p'. That would mean, in T, there is a path from i to $C[i]$ through p'. But, there is a path from i to $C[i]$ which is p. This would make a cycle. Hence, contradiction. T' is a valid tree, it contains edge $(i, C[i])$ different from T. $C[i]$ is fully connected to all other edges in the graph through the edges in T.

$$w(T') = w(i, C[i]) + w(p\_rest) + w(p')$$

Since T is an MST, if $w(i,j)$ was greater than $w(i, C[i])$, then,

$$w(T) = w(i,j) + w(p\_rest) + w(p')$$

$$w(T') = w(i, C[i]) + w(p\_rest) + w(p')$$

$\therefore \omega(T) > \omega(T')$

But T is on MST. Hence contradiction.

Consider $\omega(i, c[i]) = \omega(i, j)$

$$\omega(T') = \omega(i, c[i]) + \omega(p\text{-rest}) + \omega(p')$$

$$\omega(T) = \omega(i, j) + \omega(p\text{-rest}) + \omega(p')$$

$\therefore \omega(T') = \omega(T)$

We saw that T' is a valid tree. T is an MST. Hence, T' is a valid MST.

If, $j > c[i]$

if $j > i$, we do not have to worry about $e(i, i) = (i, c[i])$ as that will be taken care of in higher indexed inductive calls.

if $j < i$, if $C[j] \neq i$, our hypothesis as we framed it, is valid.

$\therefore$ for $w(i,j) \geq w(i, C[i])$ our MST definition is valid and for $w(i,j) < w(i, C[i])$, we cannot have that as then $j = C[i]$ which is not the case.

if $C[j] = i$,

let the path through i to $C[i]$ be

$$i - j - x_1 - x_2 - - x_k - C[i]$$

then,

$$w(x_1, j) \geq w(i, j)$$

we won't have $e(x_1, j)$ in our MST T.

In case of equality, we get the strict inequality from indices and extend the previous argument. $\therefore$ $C[j]$ can't be i for the original tree to exist.

Now swap for i only replaces (i,j) with $(i, C[i])$
$\therefore$ for some k, $(C[k], k)$ is not replaced. $(k<i)$

- As we cover all i values, every edge in P of the form $(i, C[i])$ is included in the MST.

Q1.e.

- We know that P is present in entirety in some MST.

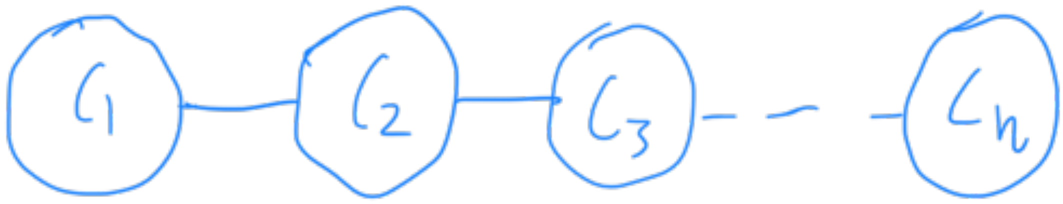
- let T be that MST.

- In order to create equivalance classes, we will remove edges from T in order to obtain P.

- If we remove i edges, we create $(i+1)$

equivalance classes. Can be shown inductively.

-



The maximum removal of edges will take place if each equivalance class C_i contains two vertices i, i' such that $C[i] = i'$ and $C[i'] = i$.

- For each vertex i, there has to be some value of $C[i]$. If the C value of $C[i]$ is x, $i, C[i], x$ (atleast three) will fall under an equivalance class,

An equivalance class will consist of the following:

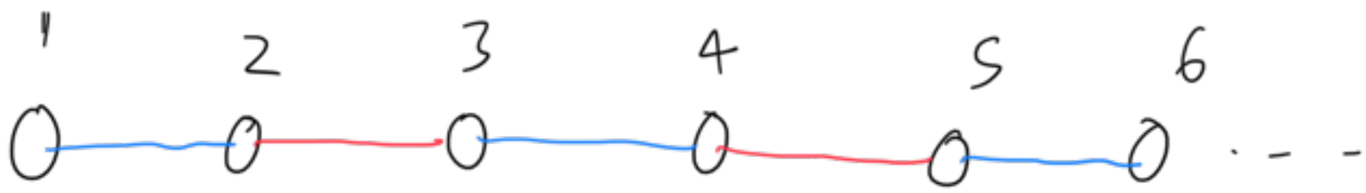
let $C[i] = i'$, $C[i'] = i''$ ...

$\rightarrow (i, i'), (i', i''), (i'', i''')$ ...

Since the total number of vertices is the same, we can minimize the number of expected number of vertices in each equivalance class in order to create the maximum number of equivalence classes.

if $C[i] = i'$ and $C[i'] = i$, we can break the linkage wrt to this equivalance class and can remove other edges connecting i, i' if they are not a part of any C value. Thus if we were to keep such an equivalance class pattern, we could create the maximum number of equivalance classes by repeating the pattern.

$P_{max} \equiv$



— $\in P$

— $\notin P$

$$\therefore |V'|_{max} = \left\lfloor \frac{|V|}{2} \right\rfloor$$

for other arrangements, $|V'| \leq |V'|_{max}$

$\therefore \quad |V'| \leq |V|/2$

Q1-f.

V' - set of equivalence classes

edge $([u],[v]) \in E'$ such that

$$w([u],[v]) \equiv \min \{ w(u',v') : u' \in [u], v' \in [v], (u',v') \in E \}$$

else $([u],[v]) \notin E'$ if empty set.

Find_MST_Cost ( )

{

sum = 0

while (V(G) != 1) :

while (v(G) ... 1)

```
        P = Compute P ( G )
        G' = Collapse ( G, P )
        for each equivalance class v* in V':
            sum = sum + Cost ( v* )
                        # sum of edges in v*
        # v* is part of P and MST.

        G = transform ( G' )

    return sum

}

transform ( G' )
{
    for each equivalence class v* in V,
    represent one vertex x*.
```

$\{v^1, v^2, \_, v^k\} \equiv \{x^1, x^2, \_\, x^k\}$

for each edge $e(v^i, v^j)$ in $\mathcal{E}'$:

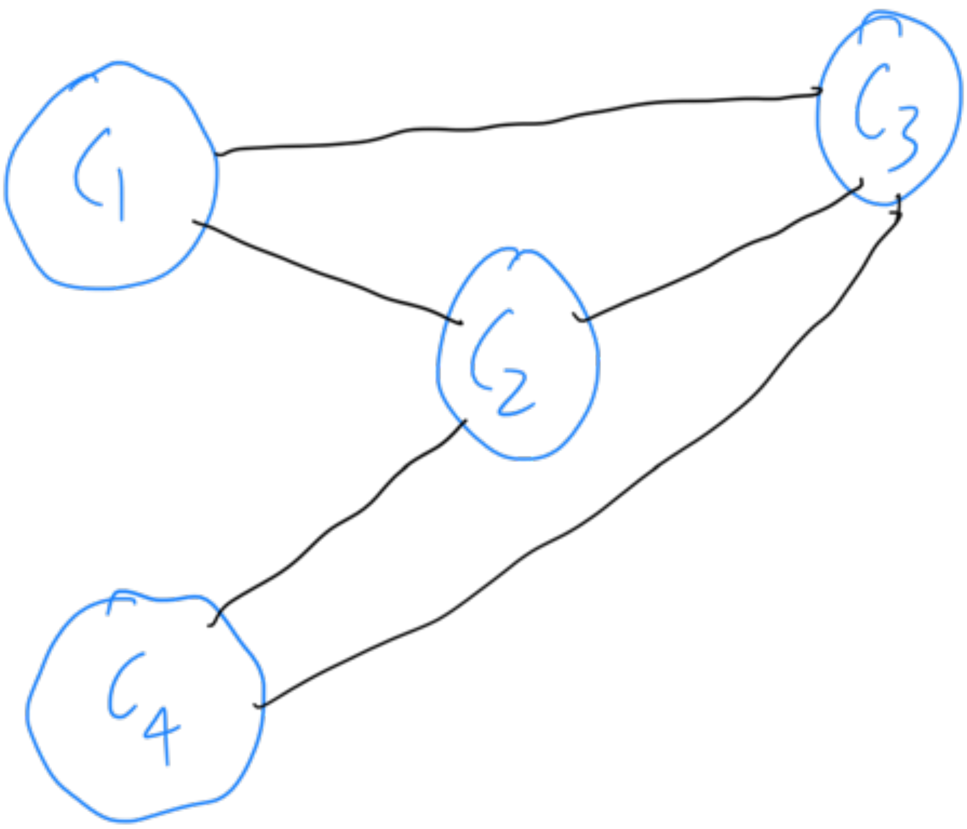
$e''(x^i, x^j) \leftarrow e(v^i, v^j)$

return $G\_new \equiv (X, \mathcal{E}'')$

}

We have P, we have the collapsed Graph G'.

eg:



→ For all [illegible] inside an equivalence class

→ for all edges inside an equivalence class, they are present in P and in the MST, we take them in our calculation of the MST.

→ for the edges in G and not in P, we have a new equivalent graph where in we need to find the MST for this graph and work with those edges, since these edges are part of the MST in

→ We can repeat this process until the transformed graph contains a single vertex corresponding to one equivalance class in which case we add all the corresponding edges.

(g)

$$T(\text{Compute } P(G)) \equiv \Theta(V+E)$$

$$T(\text{Collapse}(G, P)) \equiv \Theta(V+E)$$

→ In the transform procedure, from (c),

$$V\_new \leq V/2$$

$$E\_new \leq E \qquad (\# E')$$

the edges outside the equivalance classes will be maximum when we have maximum number of distinct equivalance classes corresponding to V_{max}. For other configurations, edges outside P potentially in MST T and in F will be lesser,

$$\therefore T(V, \mathcal{E}) \propto k(V+\mathcal{E}) + k\left(\frac{V}{2}+\mathcal{E}\right)$$

$$+ \; k\left(\frac{V}{4}+\mathcal{E}\right) \; \ldots \; \log V \text{ times}$$

$$T(V, \mathcal{E}) \doteq \Theta\left(V \log V + \mathcal{E} \log V\right)$$

Q 11.1.h

From the Compute P procedure we get P.

V[1, ___, n] be the array of vertices

Start with some vertex, compute the DFS from that vertex for graph corresponding to P.

In that DFS traversal, assign the same

value of v. equivalence for all vertices reachable from the start.

Repeat the DFS procedure until all vertices have been assigned an equivalence class.

This will take time analogous to DFS.
$\theta(V+E)$

Now, to fill,

$e(u^*, v^*)$, initialized to ∞.

u^*, v^* are equivalence classes.

for each edge (u,v) in G:

if u.equivalence != v.equivalance:

$$w(u.eq, v.eq) = \min \{ w(u.eq, v.eq), w(u,v) \}$$

Take all ω(u.eq, v.eq) values not equal to inf. This will give us E:

$$\Theta(E)$$

$\therefore$ Total time

$$T(V, E) = \Theta(V+E)$$