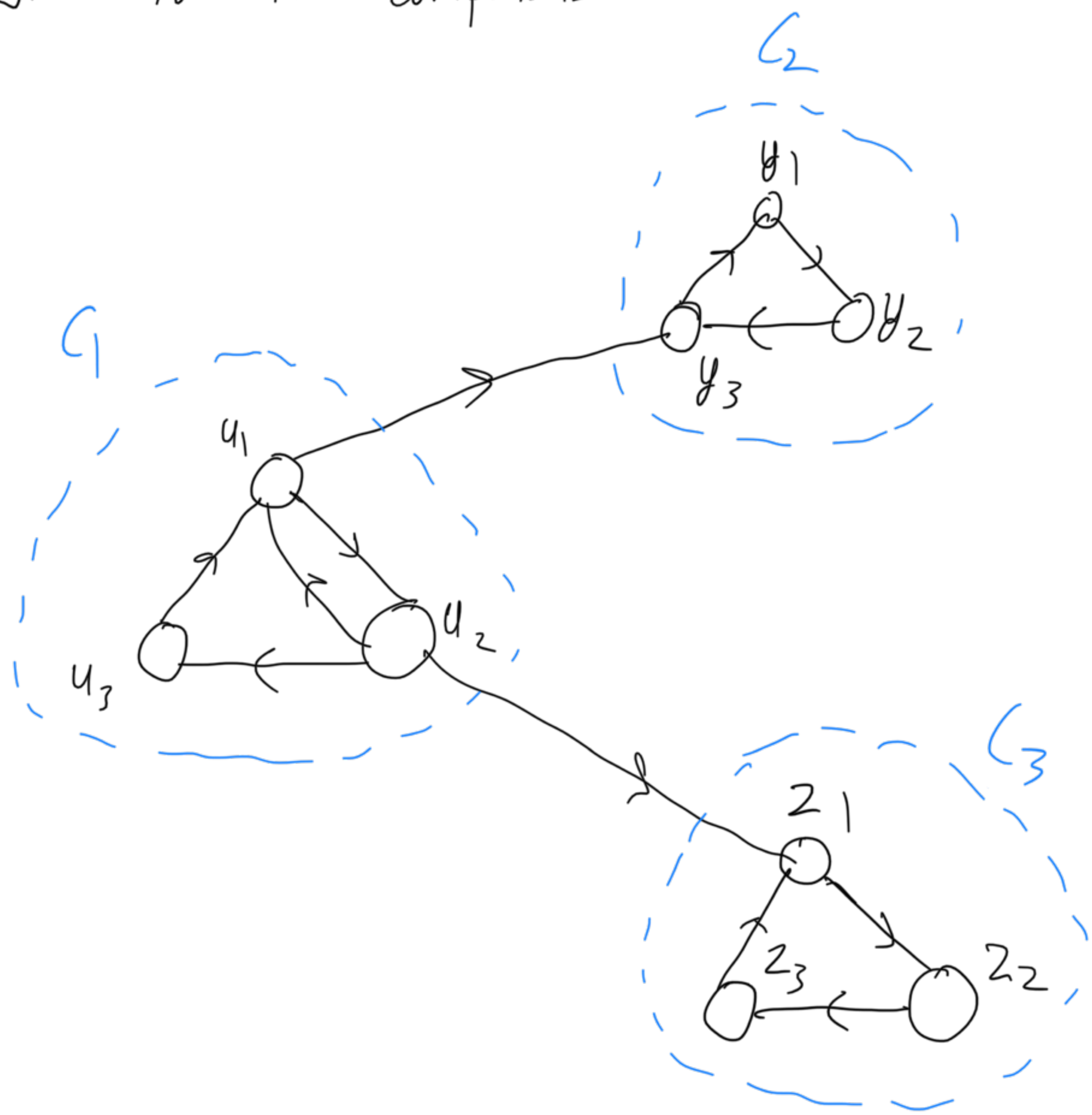
FA Assignment 11

Q2.

a) Without loss of generality, let the strongly connected components be $C_1, C_2 \ldots C_k$

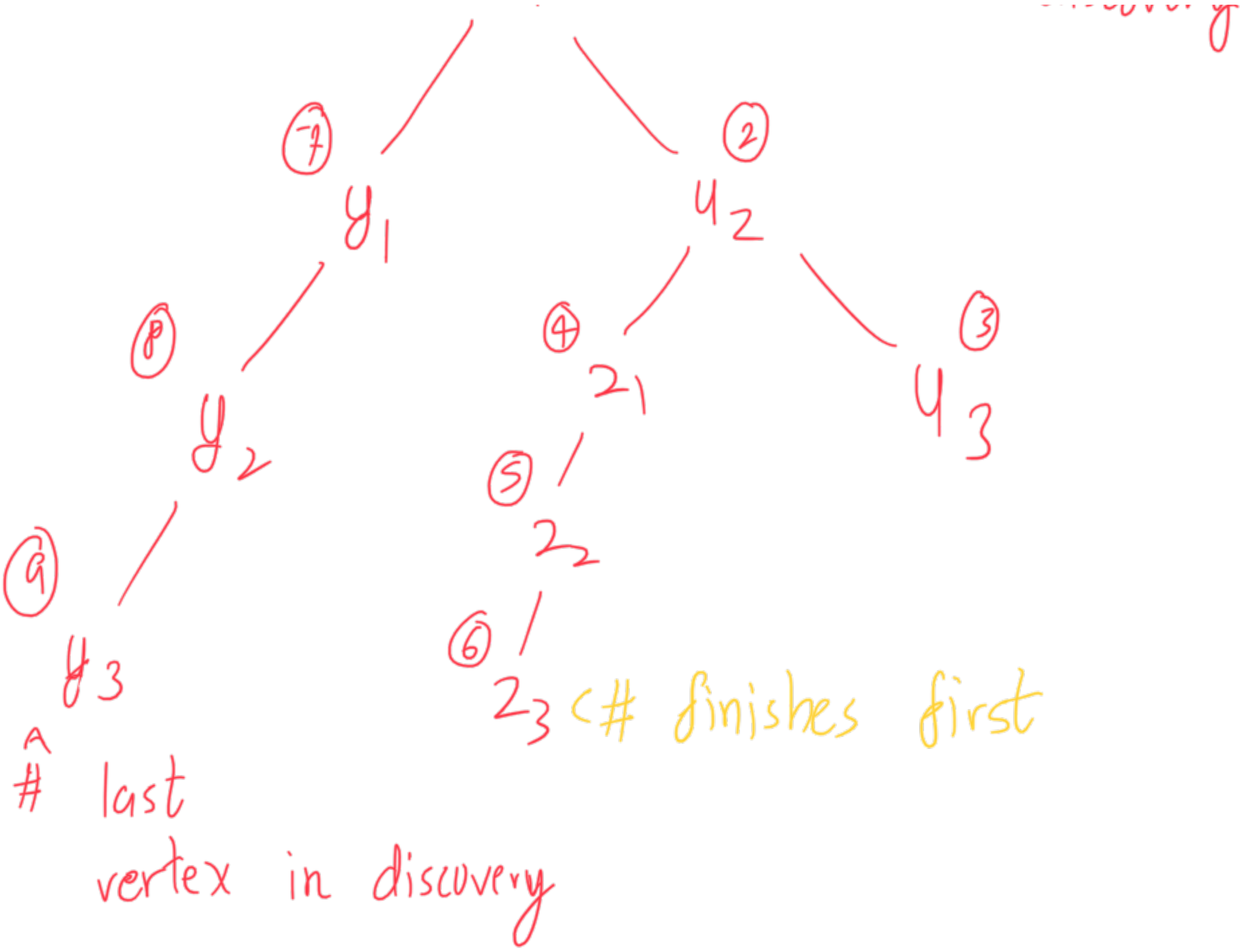
We know that we can draw a DAG wrt to the components.



Consider the following DFS call

① - first vertex in discovery

u_1



Order of finishing times of vertices, (decreasing) -

$u_1 \; y_1 \; y_2 \; y_3 \quad u_2 \; u_3 \; z_1 \; z_2 \; z_3$

→ We know our components are (u_1, u_2, u_3), (y_1, y_2, y_3), (z_1, z_2, z_3)

→ Therefore, for our DFS run, we do not have a partition for a cool ordering.

→ A cool ordering is $u_1\ u_2\ u_3\ y_1\ y_2\ y_3\ z_1\ z_2\ z_3$

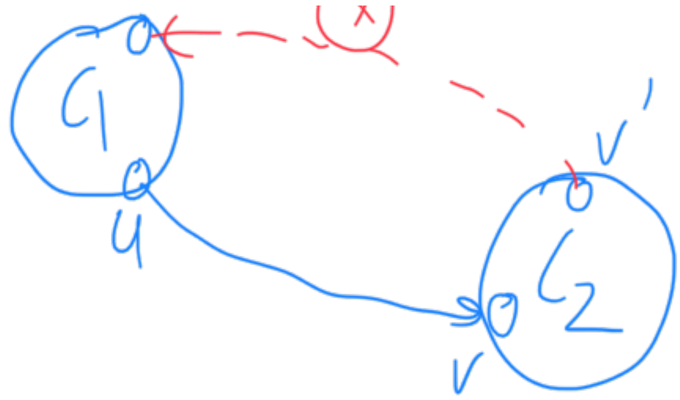
→ Sorting by decreasing order of finishing times for our DFS run is not necessarily cool.

Q2.
(b)

→ Without loss of generality, let $C_1, C_2, \ldots C_k$ be the connected components in Graph G in topological order. In a cool ordering, the vertices of a component will be together and the components will be in topological order.

Lemma 1 :
→ For some C_1 and C_2, if $u \in C_1$ and $v \in C_2$, $(u,v) \in E$, there cannot exist u' and v' such that (v', u') belong to E.

u'



More generally, if there exists a path p from u to v, there cannot exist a path p' from v' to u'.

Proof:

- → Every vertex u' in C_1 is connected to every other vertex in C_1.
- → There is a path from u to v.
- → Every vertex v' is connected to every other vertex in C_2.
- → for some arbitrary vertices u_α, v_α, there will always exist a path from $u_\alpha \to u$, $u \to v$, $v \to v_\alpha$; $v_\alpha \to v'$, $v' \to u'$, $u' \to u_\alpha$.

∴ If the proposition in lemma 1 was correct, $C_1 \cup C_2$ itself would be a SCC, but we know that C_1, C_2 are distinct SCC's.

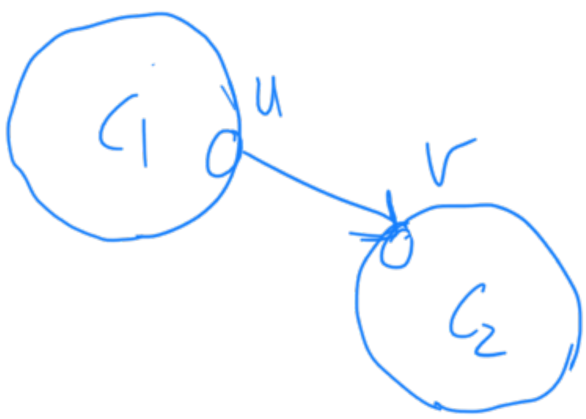
∴ Lemma 1 which negates the proposition

∵ Lemma 1 which negates the proposition, holds.

Lemma 2 :
Let C_1 and C_2 be two SCC's, let there be some edge (u,v) such that $u \in C_1$, $v \in C_2$. then,

$$f(C_1) > f(C_2)$$

where f is the max finishing time for elements in that respective cluster.



$$f(C_1) < f(C_2)$$

→ Suppose some element u' in C_1 was discovered first. The DFS call recursion must proceed to u as u, u' will have a path between them.

→ At the sub tree at u, the DFS must proceed to v, the vertices at C_2 will thus be covered, the DFS call must later terminate at v without going to u, as from lemma 1, there is no path from C_2 to C_1. The DFS control later goes into u and subsequent vertices in C_1.

→ Since there is no back edge, such a formulation is valid even when some vertex v' in C_2 was discovered first.

$$\therefore \ f(C_1) > f(C_2).$$

→ $C_1, C_2, \_\_, C_k$ will form a DAG, with an edge between C_1 and C_2 if some edge exists in G between a vertex in C_1 and a vertex in C_2.

→ If it wasn't a DAG, if there was a path from C_2 back to C_1, then, from lemma 1, we know that such a back path isn't possible.

Let $T(G)$ be the corresponding topological sort of G.

To Prove: If DFS is run on G^T w.r.t

To Prove : If DFS is start on G^T, w.r.t to some cool ordering, the resulting DFS trees are SCC's.

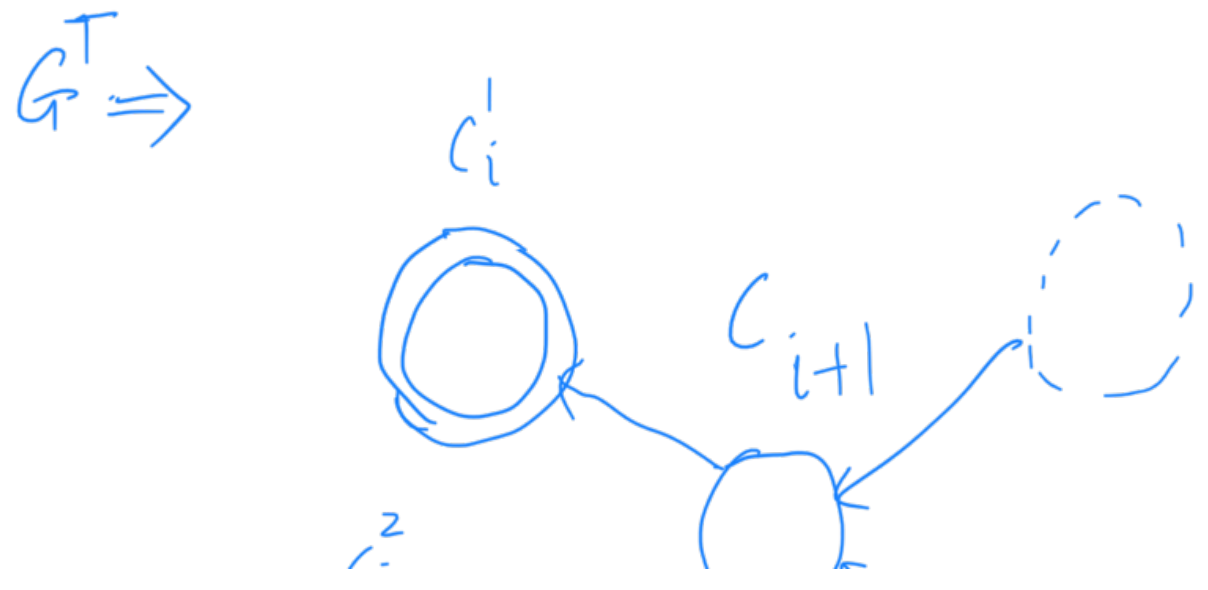
Inductive Hypothesis :

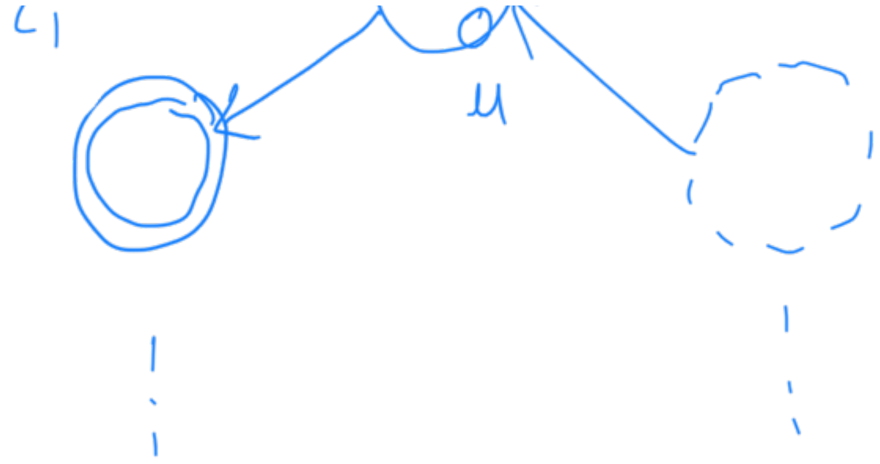
→ The first k trees produced, as per this procedure, running DFS on G^T, are SCC's.

Base Case :
$k = 0$, trivial

Let the assumption be valid for some i,
i.e. we have found $C_1, C_2, \_\_, C_i$.

→ For C_{i+1}, let u be the starting vertex on which DFS is called.
i.e. let $u \in C_{i+1}$.





We will show that this graph works and explain it as well.

As per our inductive assumption,

$C_i^1, C_i^2, \ldots, C_i^m$ have already been established as SCC's.

For any strongly connected component j which has yet to be visited, the recursion starting at u has to end at u.

$\therefore\ f(C_j) < f(C_{i+1})$

$\therefore\ e(C_j \to C_{i+1})$ cannot belong to E. Such a condition will be in direct violation of lemma 2.

let there be i^* such that $e(C_{i^*} \to C_{i+1})$

belongs to E^T. i therefore cannot belong to the set of C_j components yet to be explored. i^* must therefore belong to component set $C_1, \ldots, C_i$ which have been explored.

$\rightarrow$ $e(C_{i+1} \rightarrow C_{i^*})$ belongs to E^T.

Similarly,

Let the set C_i denote SCC's such that, $X \in \{0, 1, 2, \ldots, i\}$, for some $x \in X$,

$\rightarrow$ Since, C_i^x has been explored,

$$\delta(C_{i+1}) < \delta(C_i^X)$$

$\therefore$ $e(C_{i+1} \rightarrow C_i^X)$ cannot be in E. Violation of lemma 2.

For some C_y,

$$e(C_{i+1} \rightarrow C_y) \in E$$

$\therefore$ y cannot belong to the set of explored SCC's $\{1, \_, i\}$. C_y must belong to the set j of undiscovered SCC's

$\rightarrow$ $e(C_y \rightarrow C_{i+1}) \in E^T$ and y is an undiscovered SCC.

All vertices in C_{i+1} will be found in that call, and the recursion will then terminate as there are no outgoing edges from C_{i+1} left to be explored. There is no white path to any of the other C_j's left to be explored. The C_is which correspond to outgoing edges have already been visited.

Thus, we find C_{i+1}.
This completes our proof by induction.

Q3.

a)

DFS-VISIT(G, u)

```
{
  u.min = u.val
  for each vertex v ∈ G.Adj[u] :
      DFS-VISIT (G, v)
      u.min = min (u.min , v.min)
}
```

Base case:

u is a leaf node.

u.min is u.val.

let $v \equiv$ node at height k of the tree.

Induction hypothesis:

For each u such that height of $u \leq k-1$, the DFS-Visit call enters the correct min value in u.min.

Induction step:

→ For node v of height k, $u' \in child(v)$ height $(u') \leq k-1$, if not the case, height of v would be greater than k.

→ let $u_1, —, u_i$ be children of v.
let m be the min value for v.

Case 1: $m \equiv v.val$, covered in call to v.

Case 2: $m \equiv u_p.val$ for some p; following from inductive assumption.
Any min value previously assigned to $v.min$ from u's other than p will be greater than m, at the iteration of p, the value of m will be assigned to $v.min$, other subsequent values in comparision being greater than m.

Hence proved for correctness of assignment at v.

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

For a particular DFS visit call from our algorithm at vertex i, consider the following equation, then generalize to all vertices.

$$T(V, \mathcal{E}) = \sum_i \left(c_1 \{\text{for vertex } i\} + c_2 \cdot e(i) \right\}$$

outgoing edges for vertex i.

$\sum_i e(i) \equiv E$

$\therefore \ T(V, \mathcal{E}) = \Theta(V + \mathcal{E})$

for a tree, $E = V - 1$

$T(V, \mathcal{E}) \equiv \Theta(\mathcal{E})$, complexity analysis is valid.

Invocation call :
DFS-Visit (G, s)

Q3.

(b)

```
DFS (G)
{
  for each u ∈ G.V:
        u.color = white

  for each u ∈ G.V:
        if u.color = white:
              DFS-Visit (G, u)
}

DFS-Visit (G, u)
{
      u.min = u.val
      u.color = Gray

      for each v ∈ G.Adj[u]:
            if v.color == white:
                  DFS-Visit (G, v)
```

$u.min = min(u.min, v.min)$

$u.color = black$

}

Invocation call,

DFS- Visit (G, s)

→ The asymptotic running time of the algorithm will be same as that of the standard DFS.

→ For each adjecent vertex u of v, after the completion of the DFS- VISIT call at u, the correct value of $u.min$ will be placed at u. This will be compared with the value assigned to $v.min$. Initially, we assign $v.min$ as $v.val$ as $v.val$ is also under consideration.

Base case $\equiv$ leaf $\equiv$ $leaf.min = leaf.val$

Induction hypothesis :

For each u such that height of $u \leq k-1$, the DFS- Visit call enters the correct min value in $u.min$

Induction step:

→ For node v of height k, $u' \in child(v)$
height $(u') \leq k-1$, if not the case, height of v would be greater than k.

→ let $u_1, —, u_i$ be children of v.
let m be the min value for v.

Case 1: $m \equiv v.val$, covered in call to v.

Case 2: $m \equiv u_p.val$ for some p; following from inductive assumption.
Any min value previously assigned to $v.min$ from u's other than p will be greater than m, at the iteration of p, the value of m will be assigned to $v.min$, other subsequent values in comparision being greater than m.

Hence proved for correctness of assignment at v.

Our analysis based on the DFS loops will hold.

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

$$\# \quad T(V, E) = \sum_{v_i} \left(c_1 + c_2 \left(e(v_i) \right) \right)$$

$$\# \quad \sum_{v_i} e(v_i) \equiv \text{sum of outgoing edges}$$
$$\equiv E$$

Q3.

(c)

→ For an arbitrary directed graph, we find the SCC's of the graph.

→ We know that the SCC's form a DAG, once we have a DAG for SCC's, for each component, we assign a temporary minimum corresponding to that component.

since all vertices within a component are reachable from each other. For a vertex, we have to find the min value of all reachable paths from that vertex, this is a valid intermediate assignment.

→ The true min value will be less than or equal to this min value. This is for each vertex in that component.

→ Once we have a DAG, the problem esentially reduces to the previous problem.

DFS-prev(G) – algorithm to find min values for a DAG as in previous question.

Also use Strongly-Connected-Components(G)
$\equiv$ SCC(G)

DFS-new(G)

{

$G' = SCC(G)$

for each C in G':

$$C.min = \min(v.val \text{ such that } v \in C)$$

DFS-prev(G')

for each C in G':
    for each v in C:
        $v.min = C.min$

}

- SCC(G) takes $\Theta(V+E)$ time
- The second step where we do the initial assignment takes time,

$$T \propto \sum_k V(C_k) = \Theta(V)$$

time

- The running of DFS on a produced DAG takes $\Theta(V+E)$ time

- The last step is an iteration through the

The last step is an iteration through the vertices of a component for all components, takes $\Theta(V)$ time.

$\therefore$ Overall time complexity

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