During the execution of CONNECTED-COMPONENTS on a undirected graph $G = (V, E)$ with $k$ connected components, how many time is FIND-SET called? How many times is UNION called? Express your answers in terms of $|V|$, $|E|$, and $k$.

CONNECTED-COMPONENTS(G)
1 for each vertex $v \in V[G]$
2 do MAKE-SET($v$)
3 for each edge $(u, v) \in E[G]$
4 do if FIND-SET($u$) ≠ FIND-SET($v$)
5 then UNION($u$, $v$)

FIND-SET is called 2$|E|$ times.
FIND-SET is called twice on line 4, which is executed once for each edge in $E[G]$. Therefore, FIND-SET is called 2$|E|$ times.

UNION is called $|V| - k$ times.
Lines 1 and 2 create $|V|$ disjoint sets. Each UNION operation decreases the number of disjoint sets by one. Therefore, in order to reduce the number of disjoint sets to $k$, UNION is called $|V| - k$ times. $|V| - (|V| - k) = k$ disjoint sets.

Give a sequence of $m$ MAKE-SET, UNION, and FIND-SET operations, $n$ of which are MAKE-SET operations, that takes $\Omega(m \lg n)$ time when we use union by rank only.

Note the following observations:
• MAKE-SET; constant time, $O(1)$
• UNION (by rank) calls two FIND-SET operations; at least constant time, $\Omega(1)$.
• FIND-SET for a leaf node is worst case; at least $\Omega(\lg n)$ since building a tree using the union rank heuristic will result in a tree with height at least $\lg n$.

Now consider the sequence of operations:
• $n$ MAKE-SET (as required by the problem)
• $n-1$ UNION (maximum possible)
• $n+1$ FIND-SET (for $m = 3n$ total operations)

Total running time:
$n*O(1) + (n-1)*\Omega(1) + (n+1)*\Omega(\lg n) = \Omega(n\lg n)$
Since $n = m/3$,
$\Omega(n\lg n) = \Omega(m\lg n)$
The least common ancestor of two nodes $u$ and $v$ in a rooted tree $T$ is the node $w$ that is an ancestor of both $u$ and $v$ and that has the greatest depth in $T$. In the off-line least common ancestors problem, we are given a rooted tree $T$ and an arbitrary set $P = \{ \{u, v\} \}$ of unordered pairs of nodes in $T$, and we wish to determine the least common ancestor of each pair in $P$.

To solve the off-line least common ancestors problem, the following procedure performs a tree walk of $T$ with the initial call LCA(root[T]). Each node is assumed to be colored WHITE prior to the walk.

LCA($u$)

1. MAKE-SET($u$)
2. ancestor[ FIND-SET($u$)] $\leftarrow$ $u$
3. **for** each child $v$ of $u$ in $T$ **do**
4.   LCA($v$)
5.   UNION($u$, $v$)
6.   ancestor[ FIND-SET($u$)] $\leftarrow$ $u$
7.   color[$u$] $\leftarrow$ BLACK
8. **for** each node $v$ such that $\{u, v\} \in P$ **do**
9.   if color[$v$] = BLACK **then**
10.      print “The least common ancestor of ”
11.         $u$ “and ” $v$ “is ” ancestor[FIND-SET($v$)]

(a) Argue that line 10 is executed exactly once for each pair $\{u, v\} \in P$.

For any pair $(a, b) \in P$, the body of the loop on line 8 is only entered twice, namely
1. $(1)$ $u = a$ and $v$ is identified as $b$
2. $(2)$ $u = b$ and $v$ is identified as $a$

Initially, the nodes are all colored WHITE.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>W</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

Case 1: $(1)$ is first.

- Line 7 has just colored node $u = a$ BLACK.
- Line 9 checks if $v = b$ is BLACK, false and line 10 skipped.
- Then, $(2)$ occurs.
- Line 7 colors node $u = b$ BLACK.
- Line 9 checks if $v = a$ is BLACK, true and line 10 executed.
- Thus, line 10 is executed only once.

Case 2: $(2)$ is first. Following a similar argument, line 10 is executed only once.

Thus, in either case, line 10 is executed only once.
(b) Argue that at the time of the call $\text{LCA}(u)$, the number of sets in the disjoint-set data structure is equal to the depth of $u$ in $T$.

**Base Case:**
At the time of the call $\text{LCA}(\text{root}[T])$, there are zero disjoint sets in the data structure and the depth of $\text{root}[T]$ is also zero.

**Main Observation:**
The least common ancestors problem is simply a depth first traversal of the tree.

Prior to a $\text{LCA}$ call on any children of a node $x$, a new disjoint set is created for node $x$. This $\text{MAKE-SET}$ operation increases the number of disjoint sets by one. When $\text{LCA}$ is then called on the child of node $x$, it is called on a node that has depth one greater than $x$.

Upon returning to node $x$ from $\text{LCA}$ on the child of $x$, the child of $x$ is $\text{UNION}$ed with the calling node, $x$, thus decreasing the number of disjoint sets in the data structure by one. At the same time, returning from the child decreases the current depth in the tree by one.

Thus, whether you are traversing down the tree and creating new disjoint sets, or traversing up the tree and unioning sets, the depth($u$) is equal to the number of disjoint sets when $\text{LCA}(u)$ called.

```
LCA(1) depth(1) = 0
      components = 0
      MAKE-SET(1)

LCA(2) depth(2) = 1
      components = 1
      MAKE-SET(2)

LCA(4) depth(4) = 2
      components = 2
      MAKE-SET(4), UNION(2,4)

LCA(5) depth(5) = 2
      components = 2
      MAKE-SET(5), UNION(2,5), UNION(1,2)

LCA(3) depth(3) = 1
      components = 1
      MAKE-SET(3), UNION(1,3)
```