ASSIGNMENT WEEK 10

Finding connected components of a graph is a fundamental algorithm used by many other algorithms. For this assignment, you will implement a Java method **`connected_components()`** which returns the connected components of a graph (sets of sets of vertices that partition the vertex set of the graph).

```
CONNECTED-COMPONENTS($G$)
for each vertex $v \in G.V$
    MAKE-SET($v$)
for each edge $(u, v) \in G.E$
    if FIND-SET($u$) $\neq$ FIND-SET($v$)
        UNION($u, v$)
```

```
SAME-COMPONENT($u, v$)
if FIND-SET($u$) == FIND-SET($v$)
    return TRUE
else return FALSE
```

0. Use the code stub provided in class. It provides a rudimentary **`Graph`** class and a **`GraphEdge`** class so that we can create graphs on integer vertices labeled 0…$|V| - 1$ and add edges with **`addEdge(i,j)`**. The class contains attributes V (a list of vertices) and E (a lists of edges which are implemented as pairs of integers).
1. Implement the pseudo-code for CONNECTED-COMPONENTS above. Write a method **`connected_components()`** for the **`Graph`** class which returns a list of lists of integers (vertices belonging to the same connected component). Use **`ArrayList`** as data structure for lists. Inside your method **`connected_components()`**, do not write methods MAKE-SET or FIND-SET, but create the equivalent inline code. They are one-liners. Start with a list of integers **`labels`** which at index v will hold the label of the component that vertex v belongs to. Hence **`labels.get(v)`** returns the value FIND-SET(V) should return. Use this list **`labels`** for your inline implementation of MAKE-SET, FIND-SET, and UNION.
2. For your UNION implementation, chose the smaller label and update both sets in **`labels`**.
3. After you have run through all edges, do not return the list **`labels`**. Instead, create a list of components where each component is a list of vertices with the same label. Loop over distinct labels using a set of the labels. An implementation of ADT set is **`HashSet`**.
4. Test your program on the graph example created in class with 6 vertices and edges 0→2, 0→3, 2→1, and 4→5. Your program should return **`[[0, 1, 2, 3], [4, 5]]`**.

Each of the steps 1 – 4 will be graded according to the following rubric for a total of 16 points.

SCORE	**4**	**3**	**2**	**1**	**0**
SKILL LEVEL	Response gives evidence of a complete understanding of the problem; is fully developed; is clearly communicated.	Response gives the evidence of a clear understanding of the problem but contains minor errors or is not fully communicated.	Response gives evidence of a reasonable approach but indicates gaps in conceptual understanding. Explanations are	Response gives some evidence of problem understanding but contains major math or reasoning errors.	No response or response is completely incorrect or irrelevant.

		incomplete, vague, or muddled.