Angel Avelar-Bonilla

Collaborators: Ryan, Karen, Paige

Data Structures

3/29/2021

PSET4

1)

Create a variable that can hold multiple connected components (Maybe a set of sets?)

For each vertex “v”:

If “v” is not marked as visited:

DFS(v)

Begin filling out next component (Ex: increment i in set[i][j])

DFS(vertex v):

Mark vertex as visited

Add vertex to current component (Ex: add vertex to set[i][j] and increment j)

Call DFS(u) for all unvisited vertices next to vertex v

Essentially, fill out the components by doing a DFS recursively on each unvisited vertex, adding and marking vertices as we go. This algorithm satisfies the O(V+E) requirement since we are just doing a DFS on a graph.

2a)

Variable to count visited vertices, initially 1 since we start at a vertex

Add all adjacent/visitable vertices of “v” into our queue for which vertices to explore

While our queue is not empty:

Grab vertex “u” from queue

Increment visited vertex count

If visited vertex count > V:

We can return stating that there is a cycle

Add all adjacent/visitable vertices of “u” into our queue

Return that there is no cycle that “v” is a part of

Essentially, we traverse each vertex using BFS keeping count of how many vertices visited, and if we traverse more vertices then we have vertices we know that “v” is a part of a cycle. Since we are using BFS we satisfy the O(V + E) requirement

2b) i. If we visit a node twice during a BFS of an undirected graph we can be sure there is a cycle since there are two paths going to the same node that can be traversed in either direction.

ii. In a directed graph, just because we visited a node twice during BFS does not mean there is a cycle. There could just be two separate paths to the node U that go from V -> U but not from U -> V in which because there is no cycle.

2c)

```
For each vertex “v” in G:
    If DFS_Cycle(v) == true:
        Return true
Return false

DFS_Cycle( vertex v):
    Mark v visited
    If an adjacent, reachable vertex has been visited:
        Return true;
    For each unvisited reachable vertex “u” adjacent to v:
        DFS_Cycle(u)
    Return false;
```

Essentially we are using DFS to traverse thru our graph, marking vertexes as we visit them. If there is an adjacent vertex that we can reach that has been marked visited, then we know there is a cycle.