Homework 16 sample solution

Due 11/04/16

November 1, 2016

A *tree* is a connected, acyclic graph (i.e., no cycles). Describe an algorithm to determine whether a given undirected graph with n vertices and m edges is a tree in $O(n+m)$ time.

Hint: be careful when detecting cycles—it's easy to "detect" cycles that aren't actually cycles. You may wish to simulate your algorithm on a two- or three-vertex tree to make sure it is correct.

```
   Input: G = (V, E): graph to test
   Input: n, m: order and size of G
   Output: whether G is a tree
 1 Algorithm: IsTree
 2 parent = Map(V)
 3 Set parent of V[0] to be itself
 4 Set parent of all other vertices to be −1
 5 if ¬ IsAcyclicDFS(G, V[0]) then
 6 |  return false
 7 end
 8 for v ∈ V do
 9 |  if parent[v] = −1 then
10 |  |  return false
11 |  end
12 end
13 return true
```

```
1  Algorithm: IsAcyclicDFS(G, v)
2  for u ∈ N(v) do
3  |  if parent[u] = −1 then
4  |  |  parent[u] = v
5  |  |  if ¬ IsAcyclicDFS(G, u) then
6  |  |  |  return false
7  |  |  end
8  |  else if parent[u] ≠ v then
9  |  |  return false
10 |  end
11 end
12 return true
```

Solution could use BFS or DFS. Testing for cycles is a bit tricky in that an edge to your parent in the traversal tree does not constitute a cycle (at least, not in a simple undirected graph).

Alternative 1: you could replace the for loop in lines 8–12 of IsTree with a test that $m = n - 1$, as any acyclic graph with $n - 1$ edges is a tree.

Alternative 2: you could use an unmodified DFS (i.e., not testing for cycles), change the if condition in line 9 to check whether v is undiscovered, and add in a check for $m = n - 1$ at the end. Any connected graph with $n - 1$ edges must also be a tree.