Solution For Homework 6

Mohan Zhang, 1002748716, morgan.zhang@mail.utoronto.ca

December 4, 2016

1 Question 1.a

We prove by induction:
Claim: For two arbitrary node u and v in G such that v $\in$ G.Adj[u], when executing DFS-VISIT(G,u), either (u,v) is a tree edge, or v is an ancestor of u in the DFS tree.
Base case: when executing DFS-VISIT(G,u) for the first two nodes u and v (they are initially white), we know that DFS-VISIT(G,u) makes u be v's parent when executing line 6.
Induction hypothesis: For any arbitrary node u in G when starting execute DFS-VISIT(G,u)'s line 4, for all previous executed DFS-VISIT(G,u') for u' $\in$ G, for any v' $\in$ G.Adj[u'], either (u',v') is a tree edge, or v' is an ancestor of u' in the DFS tree.
Next step, let v be an arbitrary node in G.Adj[u].
If v.color==white, we know that v has not connected to any previous node. And in line 6, we make u the parent of v, which means that u is the parent of v and (u,v) is a tree edge.
Else if v.color==black. Since we know that v $\in$ G.Adj[u], we must have reached u previously from executing DFS-VISIT(G,u)'s (DFS-VISIT(G,u) must finish to make v.color black) line 4. So by induction hypothesis, either (v,u) is a tree edge, or u is an ancestor of v in the DFS tree.
Else if v.color==gray, means DFS-VISIT(G,v) has not finished yet. During DFS-VISIT(G,v), since u $\in$ G.Adj[v], there is a execution of { for u in G.Adj[v], if u.color == WHITE.....}. From the induction hypothesis, either (v,u) is a tree edge, or u is an ancestor of v in the DFS tree.
In all tree cases, the induction hypothesis satisfied, So the induction hypothesis is true for any call to DFS-Visit(G,u). Then, we have: in any call to DFS-Visit(G,u) executed during the run of the algorithm, for any v $\in$ G.Adj[u], either (u,v) is a tree edge, or v is an ancestor of u in the DFS tree. We are done.

2 Question 1.b

edit DFS-VISIT(G,u) to be:
time = time + 1

```
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == white
        v.π==u
        return DFS-VISIT(G,v)
    else
        return True // it does cotain a cycle
u.color = black
time = time + 1
u.f=time
return False
```

The above algorithm runs in O(|V|) times since the number of DFS-VISIT call is exactly number of branches(|V|) times. And, the algorithm is correct since it returns True iff it enters else statement, and from (a) we know that, In any call to DFS-Visit(G,u) executed during the run of the algorithm, for any v $\in$ G.Adj[u], either (u,v) is a tree edge, or v is an ancestor of u in the DFS tree, ,which is the same as: in any call to DFS-Visit(G,u) executed during the run of the algorithm, for any v $\in$ G.Adj[u], if (u,v) is not a tree edge, then v is an ancestor of u in the DFS tree, (which means (u,v) is a back edge from the tutorial). And we know that a back edge creates a cycle (we can create a cycle from the DFS tree by following step: find the path from v to u, and with the back edge (u,v), those nodes create a cycle.). So, the algorithm returns True iff G contains a cycle. So the algorithm is true.