Solution For Homework 6

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

December 5, 2016

1 Question 1.a

Claim: If the DFS tree contains back edge, it contains cycle.
Proof: 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.

We prove by induction:
Goal: 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

```
DFS2(G):
1 for each vertex u in G.V:
2       u.color := white
% line 1, 2 is initialization.
3 for each vertex u in G.V:
4       if u.color==white:
5       DFS–VISIT2(G,u)

DFS–VISIT2(G,u)
1 u.color = gray\\
2 for each v in G.Adj[u]:
3       if v.color == white
4               DFS–VISIT2(G,u)
5       else if v.color == gray
6               return True
                %find a circle.
7       v.color:=black
```

Brief introduction:
If we want to find out whether a given undirected graph G = (V, E) contains a cycle, we run DFS2(G) and see whether it returns True or not. DFS2(G) returns true iff G contains a cycle. And, the algorithm DFS-VISIT2(G,u) just deleted unnecessary part in DFS-VISIT(G,u) and added one line return, which runs no more than 1 time, so we can ommit that when commuting upper bound of running time.
Justification of running time:
Initialization runs in |V| times since it initialize all vertices, and no more. The worst case is finished the whole program without return (True). What's more, from part (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 ((u,v) is a back edge). And, if a DFS tree doesn't have a back edge, then every edge links one parent to one child which is one level below it, means that the execution of "for each v in G.Adj[u]" executes no more than |V| times, means |E| $\leq$ |V|. In other words, the DFS-tree is a "pure tree", its branches is no more than its nodes, which means that the graph G has edges no more than vertices.
So, in DFS2(G) and DFS-VISIT2(G,u), each vertices has been visited no more than once. So, DFS2(G) runs in O(|V|)times. Justification of correctness:
First, if G contains a cycle, DFS2(G) returns true.
Proof: 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)}. And we know that a back edge creates a cycle (from the claim in part (a)). So, if G contains a cycle, DFS2(G) returns true. Second, if DFS2(G) returns true, G contains a cycle.
Proof: DFS2(G) returns true iff v.color == gray. v.color == gray means we are currently searching v, means v has not finished yet. So, pick all gray vertices reached after u and before v at that time, call them (u_0=u, u_1, ... , u_l=v), (u_0, u_1, ... , u_l) form into a cycle in G, since (u_m,u_n) in G if ($|m - n|$=1) by definition of DFS-VISIT(G,u), and (u,v) in G.
So, the algorithm returns true iff G contains a cycle, it is correct.