1 Homework 4

Name: Kangdong Yuan

1.1 problem1

a).I did not work in a group.
b).I did not consult without anyone my group members
c).I did not consult any non-class materials.

1.2 problem2

a).
graph1: A,B,D,E,G,F
graph2: C,H,I
b).

$node$	$pre-number$	$post-number$
A	1	12
B	2	11
D	3	6
E	4	5
G	8	9
F	7	10
C	13	18
H	14	17
I	15	16

c).edges and labels
$\{A,B\}\ Tree,\ \{B,D\}\ Tree,\ \{D,E\}\ Tree,\ \{E,D\}\ Back,\ \{A,E\}\ Forward,$
$\{B,G\}\ Tree,\ \{G,F\}\ Tree,\ \{G,D\}\ Cross,\ \{C,H\}\ Tree,\ \{H,I\}\ Tree,$
$\{C,I\}\ Forward$

1.3 problem3

a). The v is the ancestor of u, if we explore the v before u, which means $pre(v) < pre(u)$.
Given $post(u) < post(v)$, there are two cases.
first case is $pre(u) < post(u) < pre(v) < post(v)$, but this case is not possible. Because, the dfs need to visit all the neighbors of a vertex before mark it as visited and return the post number.
The second case is $pre(v) < pre(u) < post(u) < post(v)$, this is the only one possible arrangement for given condition $post(u) < post(v)$. And in this

case, the $pre(v) < pre(u)$, so the v is the ancestor of the u.
Finally we can conclude that give $post(u) < post(v)$, v is the ancestor of u.
b).First, traverse the graph by dfs order and record the pre and post number of each vertex. The time complexity of this algorithm is $O(|V| + |E|)$. Then we check the pre-number and post-number of u and v, if $pre(u) < post(u) < pre(v) < post(v)$, we can say that u is the ancestor of v. We define this comparison take constant O(C) time. And, this algorithm can be done in linear time $O(|V| + |E| + C)$

1.4 problem4

First, If we want to find which vertex can reach the vertex i, we can do it inversely. So, for graph G, we convert it to reversed graph G^R, the time complexity of reverse graph algorithm is $O(|V| + |E|)$. Then we use this reversed graph to find $m(i)$, because we have reversed the graph, the vertex can be reached is the vertex that reached from previously.
To find $m(i)$, use the dfs to traverse all the vertex that i can reach, then return the smallest one.

```
program find m(i):
Input: graph: G and integer i
Output: integer j
find − mi(G, i)
j=i
stack.push(i)
while stack not empty
        u pop from stack
        if u is not visited
                mark u is visited
                for every neighbor vertex w of vertex u
                        stack.push(w)
                        if w < j
                                j=w
return j
```

This algorithm go through all the vertex that i can reach by dfs order, then it compare the vertices to find the smallest, finally it return the smallest vertex as a integer. It time complexity is $O(|V| + |E|)$, because it is the same time complexity as dfs algorithm.
The total time complexity is the running time of reverse graph and $find -$

$m(i)$, which is the $2(|V|+|E|) = O(|V|+|E|)$. So, all m(i) can be computed in $O(|V|+|E|)$