CMPSC 465 – Spring 2021 — Solutions to Homework 5

Josiah Kim, SID 948500821, `juk483`

February 25, 2021

1. Getting started

(a) I did not work in a group.

(b) I did not consult with any of my group members.

(c) I did not consult any non-class materials.

2. DFS Basics

(a) A, B, D, E, G, F, C, H, I

(b) A(1, 12), B(2, 11), D(3 ,6), E(4, 5), G(7, 10), F(8, 9), C(13, 18), H(14, 17), I(15, 16)

(c) $A \rightarrow B =$ Tree

$B \rightarrow D =$ Tree

$D \rightarrow E =$ Tree

$E \rightarrow D =$ Back

$A \rightarrow E =$ Forward

$B \rightarrow G =$ Tree

$G \rightarrow F =$ Tree

$G \rightarrow D =$ Cross

$B \rightarrow D =$ Tree

$C \rightarrow H =$ Tree

$H \rightarrow I =$ Tree

$C \rightarrow I =$ Forward

3.Pre and Post Processing

(a) CLAIM:
if u, v is an edge in an undirected graph, and during depthfirst search $post(u) < post(v)$, then v is an ancestor of u in the DFS tree.

PROOF:
There are only two cases in which $post(u) < post(v)$. Using nested interval notation we can depict the relationship between vertices.
Case 1: $[pre(u), post(u)][pre(v), post(v)]$
Case 2: $[pre(v), [pre(u), post(u)], post(v)]$
Both casese represent a descendant-ancestor relationship. The first case is possible only if every other vertex has been visited and there is no edge between u and v. However, we know that there is an edge between those two vertices. Therefore, only the second case is possible which means that v is an ancestor of u.

(b) We know that for u to be an ancestor of v, v would have to be the descendant of u. This is the case where u is discovered prior to v which means $[pre(u), [pre(v), post(v)], post(u)]$. More importantly, $post(u) > post(v)$.

Therefore, in T, we would get the pre and post processing numbers of node v and compare it with node u's pre and post processing numbers to see if $pre(u) < pre(v) < post(v) < post(u)$ holds true. If so, then u is an ancestor of v. Else, it is not.

This preprocessing would take constant time because the algorithm would always rely on the comparison of two nodes no matter the number of nodes in a path.

4.Application of DFS

Let G^R be the reverse graph of G.

```
function smallest_j(G):
        while there are vertices unvisited in G:
                run DFS on G^R starting with the unvisited vertex
                        with the smallest i
                for every vertex visited:
                        m(j) = i
```

The values $m(1), ..., m(n)$ can be computed in $O(|V| + |E|)$ time. We know this because we know that the time complexity for DFS is $O(|V| + |E|)$ for any graph. We are essentially just running a DFS on the reverse graph.