CSE 6331 – Algorithms – Spring, 2015 – Prof. Supowit

Homework 3 – Due: Wednesday, February 4

1. Here is the DFS paradigm modified by adding a global variable *time*, and recording, for each vertex *v*, when *v* was colored gray and when it was colored black:

```
procedure DFS ( graph G = (V,E) )
  /* G may or may not be directed. */
    begin
        for each v ∈ V do v.color ← WHITE ;

        time ← 0 ;

        for each v ∈ V do

                if v.color = WHITE then Visit( v )
    end .

  procedure Visit( vertex v ) ;
    begin
        time ← time + 1 ;
        v.start ← time ;
        v.color ← GRAY ;

        for each w s.t. (v, w) ∈ E do
            if w.color = WHITE then Visit( w );

        time ← time + 1 ;
        v.finish ← time ;
        v.color ← BLACK ;
    end .
```

For each vertex *v*, define $active(v) = \{t : v.start \le t \le v.finish\}$.

For each of the following sentences regarding these *active* intervals after the DFS has terminated, answer "true" or "false;" prove your answers.

(a)
$(\forall\ v, w \in V)[active(v) \subset active(w)$ or $active(w) \subset active(v)$ or $active(w) \cap active(v) = \varnothing]$

(b)
$(\forall\ v, w \in V)[(active(w) \cap active(v) = \varnothing) \Rightarrow (v$ is neither a descendant nor an ancestor of w in the DFS forest$)]$

2. Design and analyze an efficient algorithm, that given a digraph $G = (V, E)$ and a weight function $weight : V \to \{1, 2, \ldots\}$, outputs, for each $v \in V$, the number

$$M(v) = \max_{u \in V} \{ weight(u) : \text{there is a path in } G \text{ from } u \text{ to } v \}.$$

In other words, we need to know the weight of the heaviest vertex from which v is reachable.

NOTE 1: since v is reachable from itself, we know that $(\forall v \in V)[M(v) \geq weight(v)]$.

NOTE 2: $(\forall v, w \in V)[\exists$ a cycle containing both v and $w \implies M(v) = M(w)]$.

Describe your algorithm in words, and then in pseudo-code.

3. Design and analyze an efficient algorithm, that given an acyclic digraph $G = (V, E)$ and two vertices s and t, outputs the number of simple paths from s to t.

Describe your algorithm in words, and then in pseudo-code.