EECS 340: Assignment 7

Shaochen (Henry) ZHONG, sxz517

Due and submitted on 04/27/2020
EECS 340, Dr. Koyutürk

Problem 1

(a)

Set the each team as a node (vertex) in graph G and make every node fully connected (representing the fact that all teams have played against each other). If one team A beats another team B, we will have a directed edge of $A \to B$ betwen the nodes representing A and B. A team's z is set to be ∞ if such team has not won against any other team.

Algorithm 1 DFS-Game-Visit(G, u)

```
procedure DFS-GAME-VISIT(G, u)
    z = NULL                         ▷ Potential z value for this node, if not end of tree.
    u.color = GRAY
    for each v ∈ Adj(u) do:
        if v.color == WHITE then
            z = v.rank               ▷ If have decendent, set decedent's rank as current's z.
            z = min(z, DFS-GAME-VISIT(G, v))   ▷ Find the lowest z from all decendents.
        else
            z = min(z, v.z)          ▷ Check if explored reachable nodes have lower z.
    u.color = BLACK
    if z == NULL then                ▷ If this node is end of tree.
        u.z = ∞                      ▷ Set z to ∞ to indicate beats no one.
        z = u.rank                   ▷ For ancestor nodes to read its rank during recursion.
    else
        if u.z == NULL then
            u.z = z
        else
            u.z = min(u.z, z)        ▷ If node have been explored, check if the tree's z is better.
    return z
```

Algorithm 2 DFS(G)

```
procedure DFS(G)
    for each u ∈ G.V do:
        u.color = WHITE
        u.z = NULL
    for each u ∈ G.V do:
        if u.color == WHITE then
            DFS-GAME-VISIT(G, u)
    for each u ∈ G.V do:
        print u.team, u.z                                  ▷ output all z_i.
```

(b)

Justification for runtime The algorithm is $O(m+n)$ as DFS will first traverse every node, which means it will at least be $O(n)$. In addition, since the graph G is implemented using adjacency list, for each node we will have to traverse through all its adjacent edges – where such number can be at most m (total number of edges in G) for a single node. Thus, the total time complexity is $O(m+n)$.

Justification for correctness The algorithm basically performs a DFS travesal on the graph G and set node `u` with the smallest `rank` value as `.z` to all of node `u`'s ancestor nodes. Due to the nature of DFS, node `u`'s ancestor nodes can legally have `u.rank` as their domination factor, as being `u`'s ancestors imply the fact that these teams have won against team `u`. Since the graph is implemented as A beats B being $A \to B$, after the DFS tavesal of a root, all nodes within this DFS tree is guaranteed to have the correct `.z` value, as all teams have been beaten by these nodes have been checked[1].
Also in DFS-GAME-VISIT we will not set a node's `.z` value unless such node is marked as `BLACK` (which means all of its decedents have been explored, and all of its reachable node – regardless marked as `BLACK` or not, are taken into account). Combine these two observations, each node has explored all of its reachable nodes, and the node's reachable node `u` with the smallest `rank` value will be assigned as the `.z` value of all nodes which have reached node `u`. The algorithm is a perfect mimic of the game logic of domination factor and guaranteed to be correct.

To speak in a simple way, if we have a node u where $u.rank$ should be the deminate factor of node x, we should either have u being a direct decedents of x and therefore promote $u.rank$ as $x.z$ during the DFS travesal from node x; or u is explored (marked as `BLACK`) when x is being traversed, in such case either u or u's ancestor v (marked as `BLACK` as well) will have the $.z$ value of $u.rank$, and when the travesal of x visits this u or v, it will take $u.z$ or $v.z$ directly if it is smaller than its current z, and therefore not skiping any beaten team with the lowest `rank` value. As the algorithm works correctly in both scenarios, the algorithem is guaranteed to be correct.

Problem 2

Note the tuple below each node represents their `(start-time, finish-time)`.

[1] Including `BLACK` nodes, as shown in DFS-GAME-VISIT's `Line 8-9`: if a node is marked as `BLACK` it means this node's `.z` value is correct, so that nodes which are reachable to this `BLACK` node can simply take its `.z` value without go into the tree of the `BLACK` node any further

(a)

Proof with cases:

There are two possible cases where we can have a $v.f > u.f$:

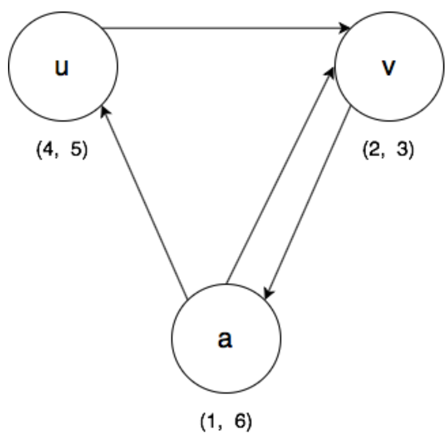
- Case 1: $v.d < u.d < u.f < v.f$
  In this case u is a decent of v, which means there must be a path of $v \to u$; and known that we have $uv \in E$ (from the question instruction), this implies there is a direct connection of $u \to v$. Thus, now we have a path of $v \to u$ and a path of $u \to v$, uv is guaranteed to be on a circle.

- Case 2: $u.d < u.f < v.d < v.f$
  In this case we have u completely discovered before v. This suggests there shoule be no relationship between u and v, as v should not be a decendent nor ancestor of u. However, it is known from the question that we have $uv \in E$, which implies v must be a decendent of u. Thus, this case will not be valid under the question restriction and therefore disregarded.

As all legal case(s) shows uv is guaranteed to be on a circle, we may prove the statement to be true.

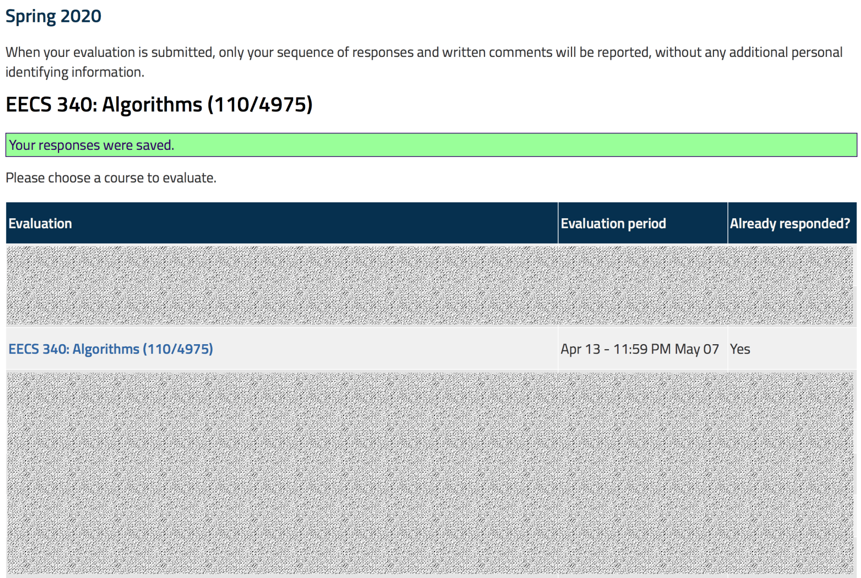
(b)

Disprove with conterexample:



It is shown that there is a path from v to u as $v \to a \to u$, however uv is still a cross edge as $v.d < v.f < u.d < u.f$ $(2 < 3 < 4 < 5)$. Thus, this conterexample disproves the statement.

Problem 4



Spring 2020

When your evaluation is submitted, only your sequence of responses and written comments will be reported, without any additional personal identifying information.

EECS 340: Algorithms (110/4975)

Your responses were saved.

Please choose a course to evaluate.

Evaluation	Evaluation period	Already responded?
EECS 340: Algorithms (110/4975)	Apr 13 - 11:59 PM May 07	Yes

ALL DONE  THANK YOU !!!!!!!!!!!!!!!!!!!