CPTS 515 HW2

Yang Zhang 11529139

Review of Tarjan's algorithm

① using DFS to label each vertex in G
in such format $\frac{n}{m}$ → 1st visit time / 2nd visit time
with the following rule:

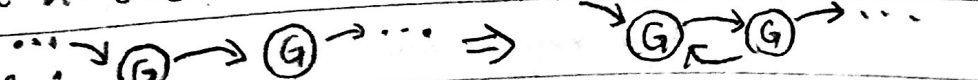
```
for each unvisit node:
    mark the time as 1st visit time, and visited the node as
    for each neighbor n
        if n is unvisit;
            recursive_call
    mark the time as 2nd visit time
```

② sort the vertices based on 2nd visit time (decreasing)

③ reverse every edge in the Graph G
and start with the 1st element in the sorted vertices
to find scc and store each scc in scc_list

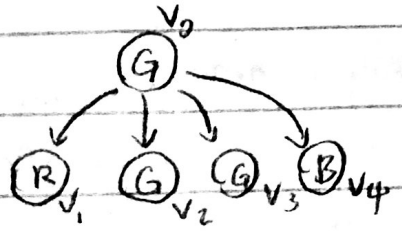
Before applying tarjan's algorithm, lets make the following changes:

① pre-process the graph: for each pair of adjacent green vertices (or adjacent red vertices) add neccessary edge to make them compose a scc

For example: 

② modify the get-neighbors method, now only the neighors with the same color as the parent are neighbors.

For example:



neighbors (V_0) returns

$[V_2, V_3]$

After the above 2 changes are made run tarjan algorithm on the modified graph twice:

1st run: find all the green SCC

2nd run: find all the red SCC

if there exist $size(SCC_i^{Red}) >= size(SCC_j^{Green})$

return true

else return false

2. The problem is to find the max network flow subject to the constraint $C_1 + C_2 < K$, in other words, to make sure finding the max flow, $C_1 + C_2 = K-1$

My idea is to apply greedy hill climbing algorithm.

based of C_1, the range of C_1 is $[0, K-1]$

① randomly select value for C_1 from $[0, K-1]$

then $C_2 = K-1-C_1$

② plug current C_1, C_2 into the max-flow algorithm to get current max M_{mid}.

③ repeat step 1 and 2 with (C_1-1) and (C_1+1) to get M_{left} and M_{right} accordingly

④ if $M_{mid} >= max(M_{left}, M_{right})$

M_{mid} is the local maximum

else $M_{mid} = \max(M_{left}, M_{right})$, $C_1 = \max(C_1^{M_{left}}, C_1^{M_{ri}}$
and repeat from step 2

Restart the whole process for a few time with different ran
start C_1 value.

3 ① using BFS to find all the yellow nodes
② detach every yellow nodes from the original graph
③ run tarjan's algorithm on updated graph
for each SCC that has size >1:
if scc contains both red nodes and gree
return true;
endfor
return false

/***
Dear Prof. Dang,
On the homework set, you state that α is a inifinte
Long path. Shouldn't α be a walk instead? I d
the question with the assumption that α is a wal

***/

4 (1) path_count = 0

```
DFS (v, v')
    for each neighbor Vn of v:
        if Vn == v', path_count ++;
        else DFS(Vn, v');
    end for
end DFS
```

(2) good-path-counter = 0

```
DFS (v, v', green_count = 0, yellow_counter = 0)
    if v.color == green, green_count ++
    elif v.color == yellow, yellow_count ++
    for each neighbor Vn of v:
        if (Vn == v')
            if v'.color == green, green_counter ++
            elif v'.color == yellow, yellow_counter ++
            if green_counter > yellow_counter
                good-path-counter ++
    end for
end DFS
```