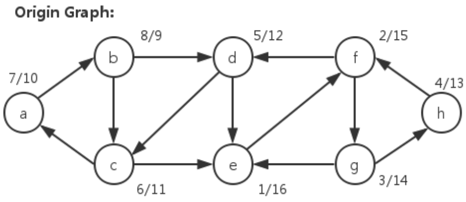
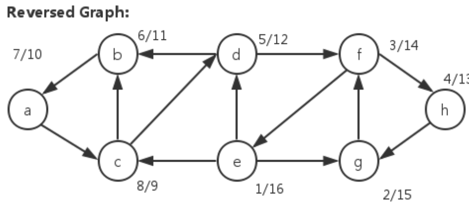
Algorithmn HW6

5140379032 JIN YI FAN

Problem 9.16





Firstly, apply DFS and update the *post* value;
Secondly, reverse graph G to G^T;
Thirdly, apply DFS to G^T starting at vertex with maximum *post* value, e this time;
A single DFS of this G^T can approach all the vertexes, so the whole graph G is strongly connected.

Problem 9.32

Require: a given graph $G = (V, E)$
Ensure: whether it is a bipartite graph

function DFS(v)
    mark v visited
    $v.mark \leftarrow (color \leftarrow !color)$ ▷ mark neighbour vertexes with different colors
    for each edge $(v, w) \in E$ **do**
        if w is marked unvisited **then**
            DFS(w)
            return $True$
        else
            return $(w.mark == v.mark?)\ False : True$
        end if
    end for
end function
$color \leftarrow 1$ ▷ main function
mark each vertex $v \in V$ unvisited
for each vertex $v \in V$ marked unvisited **do** ▷ start dfs
    if DFS(v)$==False$ **then**
        return $False$
    end if
end for
return $True$

This is an $O(V + E)$ algorithm

Reverse graph

Require: a directed graph $G = (V, E)$
Ensure: the reversed graph $G^T = (V, E^R)$

for each edge $(v, u) \in E$ **do**
    append (u, v) to E^R
end for
replace E with E^R

Find a cycle

Require: an undirected graph $G = (V, E)$, $e = (u, v) \in E$
Ensure: whether G has a cycle containing e

delete e from E
apply DFS to G with *pre*/*post* signature
if $v.pre > u.post$ or $u.pre > v.post$ **then** ▷ exist such cycle iff u and v are still in same component
    return $False$ ▷ which means there shouldn't have any cross edge between them
else
    return $True$
end if

Hamiltonian path in a DAG

Require: a DAG $G = (V, E)$
Ensure: whether it has a Hamiltonian path

topo[] $\leftarrow$ the topological sorted vertex sequence of G
for $i \leftarrow 1$ to $topo.size$ **do** ▷ whether every consecutive pairs are connected
    find the edge between $topo[i-1]$ and $topo[i]$ in E
    if cannot found **then** ▷ once unconnected, there won't exist
        return "$can\ not\ find$"
    end if
end for
return *topo* ▷ the topological sort sequence is actually the Hamiltonian Path