Zachary Seymour
CS 575
Theory Assignment 3
October 17, 2013

1 Given an undirected graph G, modify the code of DFS to check if G contains cycles. If it does, print the first one discovered and return "yes", otherwise return "no".

Our original pseudocode for depth-first search consists of two functions:

```
function DFS(G, color, d, f, parent)
    for all vertex u do
        color[u]=white
        parent[u]=-1
        time=0
    end for
    for all vertex u do
        if color[u]==white then
            DFS-Visit(u)
        end if
    end for
end function
```

and

```
function DFS-VISIT(u)
    color[u]=gray
    time=time+1
    d[u]=time
    for all v ∈ adj[u] do
        if color[v]==white then
            parent[v]=u
            DFS-Visit(v)
        end if
    end for
    color[u]=black
    time=time+1
    f[u]=time
end function
```

We modify the `DFS-Visit` function to check if any each v adjacent to vertex u have been marked gray, which means this is the second time we are visiting it while checking the same node, so a cycle exists. Otherwise, we recurse as normal. So, we now have:

```
function HASCYCLE?(G, color, d, f, parent)
    for all vertex u do
```

```
        color[u]=white
        parent[u]=-1
        time=0
    end for
    for all vertex u do
        if color[u]==white then
            if DFS-Visit(u) then
                print cycle
                return "yes"
            end if
        end if
    end for
    return "no"
end function
```

and

```
function DFS-Visit(u)
    color[u]=gray
    time=time+1
    d[u]=time
    for all v ∈ adj[u] do
        if color[v]==gray then
            return true
        else if color[v]==white then
            parent[v]=u
            if DFS-Visit(v) then
                return true
            end if
        end if
    end for
    color[u]=black
    time=time+1
    f[u]=time
    return false
end function
```

2 For each node u in an undirected graph, let `sDegree(u)` be the sum of the degrees of the neighbors of u. Give pseudocode for an $O(E+V)$ algorithm that outputs for each node u its `sDegree`.

```
function sDegree(u)
    sDegree=0
    for all v ∈ adj[u] do
        for all t ∈ adj[v], s.t. t≠u do
            sDegree=sDegree+1
```

end for
end for
end function

3 The reverse of a directed graph $G = (V, E)$ is another directed graph $G^R = (V, E^R)$ on the same vertex set, but with all edges reversed. So $E^R = \{(v, u) | (u, v) \in E\}$. Give an $\mathrm{O}(E + V)$ algorithm for computing the reverse of a graph in adjacency list format.

To reverse the adjacency list, rather than transposing the adjacency matrix, we cycle through all nodes u and for each edge to v, add u to the beginning of the adjacency list for v.

for all u ∈ V **do**
  for all v ∈ adj[u] **do**
    adj[v]←u
  end for
end for

4 Assume the graph of the Web where a link y stored in page x is represented by a directed edge from x to y. Write pseudocode for printing the address of all the pages at distance d from p. (The address can be derived from `x.address`).

I'm going to use a modification of `DFS-Visit`, keeping track of how far away from x we've traveled, printing the address once we have gone d steps down. I'll also assume that each node is at distance zero from itself.

function PAGESATDISTANCE(d, x)
  if d==0 **then**
    print x.address
  else
    for all y ∈ adj[x] **do**
      PagesAtDistance(d-1,y)
    end for
  end if
end function