Depth First Search

The input is a graph. This algorithm traverses each component of the graph, numbering vertices as they are encountered (n[u]) and saving for each vertex u the neighbor (if any) from which it was first discovered (p[u]). During the traversal a vertex can be one of three colors, white as long as it is undiscovered, gray after it is discovered and is being explored, and finally black when its exploration is finished. A rooted tree is obtained for each component. The roots are saved in the set R and each vertex in the component except the root has p[u] as its parent in the tree. The vertices also are numbered in the order in which their exploration terminates (f[u]). This numbering is referred to as the post-dfs-numbering while the n[u]'s are the pre-dfs-numbering.

```
begin_DFS_main
    S ← ∅ ; R ← ∅ ; count1 ← 0 ; count2 ← 0
    for each vertex u ∈ V(G)
        color[u] ← white
        Append(u, S)
    endfor
    while (S ≠ ∅)
        w ← Front(S)
        p[w] ← nil
        Append(w, R)
        DFS(w)
    endwhile
end_DFS_main


begin_DFS(u)
    color[u] ← gray ; Delete(u, S)
    count1 ← count1 + 1 ; n[u] ← count1
    for each vertex v adjacent to u
        If color[v] = white
            then  p[v] ← u
                  DFS(v)
    endfor
    color[u] ← black
    count2 ← count2 + 1 ; f[u] ← count2
    return
end_DFS()
```

Observations:

- $\texttt{DFS(}u\texttt{)}$ is called once on each vertex u of G.
- The vertices discovered within the call to $\texttt{DFS(}u\texttt{)}$ are all the proper descendants of u in a tree.
- If v is a proper descendant of u in one of the depth-first-search trees, then $\texttt{n[}v\texttt{]} > \texttt{n[}u\texttt{]}$ and $\texttt{f[}v\texttt{]} < \texttt{f[}u\texttt{]}$.
- At any time during the execution of `DFS_main` the gray vertices correspond exactly to those vertices whose `DFS()`'s have begun but are not yet finished, and these vertices form the unique path from the current vertex to the root of its tree.

Using the same definitions of *tree*, *back* and *cross* edges as in the breadth first search handout, we obtain almost the opposite result about the types of edges present.
Lemma If T is a rooted depth-first-search tree of a connected graph G, then each edge of G is either a tree edge or a back edge with respect to T.

Proof: Consider an edge $e = (u, v)$. If e is a loop, then $u = v$ and e is a back edge. So assume $u \neq v$ and without loss of generality assume that $\texttt{n[}u\texttt{]}$ is less than $\texttt{n[}v\texttt{]}$ (that is, u is discovered before v). Since vertices are numbered as their `DFS()` calls are made, $\texttt{DFS(}u\texttt{)}$ must have been made before $\texttt{DFS(}v\texttt{)}$. There are two cases depending on whether v is *white* or not, when the edge e is examined in the `for-loop` that cycles through all of u's neighbors.

Case 1: v is *white* when e is examined.
In this case, u will become v's parent and e will be a tree edge.

Case 2: v is not *white* when e is examined.
By assumption $\texttt{DFS(}u\texttt{)}$ occurs before $\texttt{DFS(}v\texttt{)}$ which means v was *white* when $\texttt{DFS(}u\texttt{)}$ began. Since v changes its *white* color only when its call ($\texttt{DFS(}v\texttt{)}$) is made, it must have been discovered as the result of $\texttt{DFS(}z\texttt{)}$ for some other edge of u which is examined ahead of e in the `for-loop`. (Note that $z = v$ is possible.) Before the edge $e = (u, v)$ gets its turn, $\texttt{DFS(}z\texttt{)}$ must finish and return. Since $\texttt{DFS(}v\texttt{)}$ is a nested call within $\texttt{DFS(}z\texttt{)}$ it must also finish before $\texttt{DFS(}z\texttt{)}$ returns. In this case v will be *black* and it will be a descendant of z and hence of u (z will have u as a parent since its `DFS()` call is made in u's `for-loop`). The edge (u, v) is a back edge.

Cases 1 and 2 exhaust all possibilities, so an edge will either be a tree edge (Case 1) or a back edge (Case 2). □

Note that in the `for-loop` it is possible for some of u's neighbors to be *gray*. Such a neighbor is an ancestor of u in the tree since its `DFS()` call was begun prior to u's and has not yet finished. In particular, if u is not the root, then its parent will be examined and found to be *gray*.