1 Directed Graphs

Before we were only considering graphs with symmetric relations in the edges. Now we're considering **directed graphs** or **digraphs**, where the edges have a defined directionality. The vertex where an edge starts is the **head** and the vertex that is pointed to is the **tail**. These together are the **endpoints**. We also term the head as the **predecessor** of the tail and the tail as the **successor** of the head.

Like with our undirected graphs. We can consider digraphs as **simple digraphs** if they don't have repeated edges or loops. Note that a simple digraph can have two edges between the same two vertices as long as they point in opposite directions.

We have similar definitions in directed graphs for **walks**, **paths**, **trails**, and **cycles**. Likewise, we have the same concepts of **subgraphs**, **isomorphism**. The **adjacency** matrix is created in a similar row-wise fashion, except now it is now longer symmetric.

Instead of just degree, digraphs consider both **out degree** ($d^+(v)$) or **in degree** ($d^-(v)$). We also have the out neighborhood ($N^+(v)$) or successor set and the in neighborhood ($N^-(v)$) or predecessor set. Likewise minimum and maximum out and in degrees:

$$\delta^-(v), \delta^+(v), \Delta^+(v), \Delta^-(v)$$

And our degree sum formula:

$$\sum_{v \in V(G)} d^+(v) = |E(G)| = \sum_{v \in V(G)} d^-(v)$$

2 Directed Connectivity

For digraphs, we have the concepts of **strong connectivity** and **weak connectivity**. The definition of strong connectivity is similar to connectivity in undirected graphs: for any u, v is a strongly connected component, there exists a directed path from u to v. Weak connectivity of a directed graph is equivalent to connectivity of its underlying graph, where the **underlying graph** of a digraph is the undirected representation created by removing directionality from the directed edges.

2.1 Connectivity Algorithms

The optimal (serial algorithm) for detecting the maximal strongly connected components in a graph is Tarjan's Algorithm. The algorithm completes one DFS of the graph, labeling each vertex with two labels. The first label is the *index*, or the DFS order. The second label, *lowlink* is the *index* of the lowest index vertex that a given vertex can reach following its out edges. The algorithm is given below (note the similarity of the outer loop to our connectivity algorithms):

```
for all $v \in G$ do
    index($v$)$\leftarrow -1$          ▷ Assume all arrays and variables are globally accessible
    lowlink($v$)$\leftarrow -1$
    onstack($v$)$\leftarrow$false
$curIndex \leftarrow 1$                ▷ DFS order counter
$Stack \leftarrow \emptyset$                   ▷ DFS stack
$SCC \leftarrow \emptyset$                     ▷ Sets of SCCs
for all $v \in G$ do
    if index($v$)$< 0$ then
        tarjan($v$)
return $SCC$
```

```
procedure TARJAN($v$)
    index($v$)$\leftarrow curIndex$
    lowlink($v$)$\leftarrow curIndex$
    onstack($v$)$\leftarrow$true
    $curIndex \leftarrow curIndex + 1$
    $Stack$.push($v$)
    for all $u \in N^+(v)$ do
        if index($u$)$= -1$ then
            tarjan($u$)
            lowlink($v$)=min(lowlink($u$), lowlink($v$))
        else if onstack($u$)=true then
            lowlink($v$)=min(index($u$), lowlink($v$))
    if lowlink($v$)=index($v$) then        ▷ $v$ is root of new SCC
        while ($w = Stack$.pop(v))$\neq v$ do
            onstack($w$)$\leftarrow$false
            $SCC$(index(v))$\leftarrow w$
    return
```

There is also Kosaraju's Algorithm, which is also optimal in terms of work complexity. However, this algorithm requires two traversals of all edges (or one of both out edges and in edges) so it isn't as efficient in practice. For weak connectivity, undirected connectivity algorithms can be used but with the edges mirrored (out edge becomes out and in edge between same vertices) to create the underlying graph.

3 Eulerian Digraphs

Since our definition for walks, paths, etc. can correspond to directed graphs, we have the concept of an **Eulerian Circuit** on digraphs. We can use a very similar methodology we used with on undirected graphs to prove the properties a directed graph must satisfy to be Eulerian.

Prove: A directed graph has a cycle if $\delta^+(v) \geq 1$.

Prove: A directed graph is Eulerian iff $\forall v \in V(G) : d^+(v) = d^-(v)$ and there is a single nontrivial component.