COMP 182: Algorithmic Thinking
11 February 2014

An algorithm that explores graphs in a different fashion than **BFS** is the *depth-first search*, or **DFS**, algorithm. To explore a graph g with **DFS**, the algorithm is called as **DFS**(g, p) with p_i initialized to $null$ for every node $i \in V$. The algorithm modifies the p values for every node in the graph.

Algorithm 1: DFS

```
Input: Graph $g = (V, E)$, $V = \{0, 1, \ldots, n-1\}$, and $p_i, \forall i \in V$.
Output: None.
Modifies: $p$.

foreach $i \in V$ do
    if $p_i = null$ then
        $p_i \leftarrow -1$;          // We designate the parent of the initial node to be '-1'
        Visit($g, i, p$);
```

Algorithm 2: Visit

```
Input: Graph $g = (V, E)$, node $i \in V$, and $p_j, \forall j \in V$.
Output: None.
Modifies: $p$.

foreach neighbor $h$ of $i$ do
    if $p_h = null$ then
        $p_h \leftarrow i$;
        Visit($g, h, p$);
```

1. Consider graph $g = (V, E)$, $V = \{0, 1, 2, 3, 4, 5\}$ and $E = \{(0, 1), (0, 3), (1, 4), (2, 4), (2, 5), (3, 1), (4, 3)\}$. Run **DFS** on g and report the p values for all nodes.

2. For a graph $g = (V, E)$ given by its adjacency list, what is worst-case running time of **DFS**, as a function of $m = |E|$ and $n = |V|$?

3. A directed, acyclic graph (DAG) is a directed graph that has no cycles. A *topological sort* of a DAG $g = (V, E)$ is a linear ordering of all its nodes such that if g contains an edge (u, v), then u appears before v in the ordering. Give the pseudo-code of an $O(m + n)$ algorithm for topologically sorting a DAG.