COMPSCI 330: Design and Analysis of Algorithms **October 12, 2017**

Lecture 11: Graph Algorithms

Lecturer: Rong Ge *Scribe: Zachary Liu*

11.1 Definitions

Graph is a way to represent relationships between pairs of objects. In a graph, the objects are abstracted to be vertices, and the relationships between objects are abstracted as edges.

Formally, a graph $G = (V, E)$ consists of a set of vertices V and a set of edges E. The edge set is a subset of pairs of vertices ($E \subset V \times V$).

As an example, we can represent locations on map as vertices, and roads as edges. A graph can be either directed or undirected. In a directed graph, edge (u, v) is different from edge (v, u); for a undirected graph the two edges are equivalent.

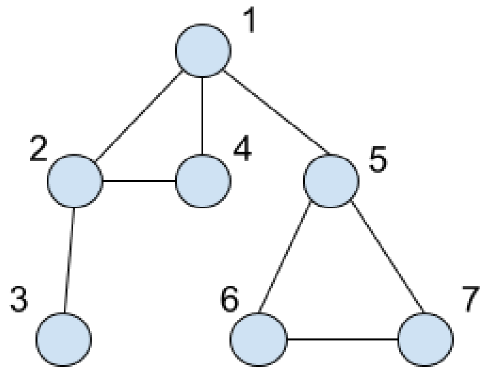
The degree of a vertex v refers to the number of edges connected to v. For a directed graph, we also use the term in-degree of a vertex to refer to the number of incoming edges to v ($|\{(u, v) \in E\}|$), and out-degree to refer to the number of out-going edges from $v(|\{(v, u) \in E\}|)$.

A path p between two vertices s, t is a sequence of edges $(e_1, e_2, ..., e_k)$, where $e_1 = (s, u_1)$, $e_k = (u_{k-1}, t)$ and $e_i = (u_{i-1}, u_i)$. The end-point of an edge e_i is the starting point of the next edge e_{i+1} in the path. We say the graph is (strongly)-connected if there is a path between every pair of vertices. A set of vertices that are pair-wise connected by paths is called a (strongly) connected component.

11.2 Graph Traversal Algorithms and Depth First Search

The first graph algorithms we will look at are graph traversal algorithms. These algorithms can be used to find paths in graphs.

Depth First Search is a method of graph traversal that is recursive in nature and visits each node in the graph.

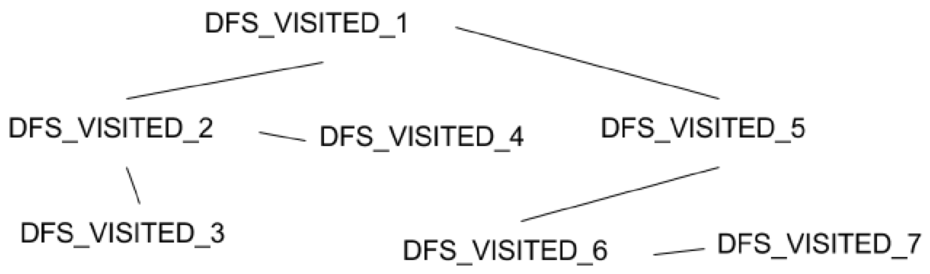


Consider the above graph and the following DFS algorithm:

```
Algorithm: DFSvisit(u)
u.visited = True;
for every outgoing edge (u,v) do
    if v.visited = False then
        DFSvisit(v)
    end
end
```

```
Algorithm: DFS
for each vertex u do
    if u.visited = False then
        DFSvisit(u)
    end
end
```

The algorithm is recursively executed as follows:



DFS Tree When executing the DFS algorithm, if $DFSvisit(u)$ called $DFSvisit(v)$, then we connect (u, v) by a tree edge, and consider v a child of u. In this way we can construct a tree (or when the graph is not strongly connected, a set of trees) called the DFS tree.

Notice that not all the edges in the original graph occur in the DFS tree.

11.2.1 Pre Order and Post Order

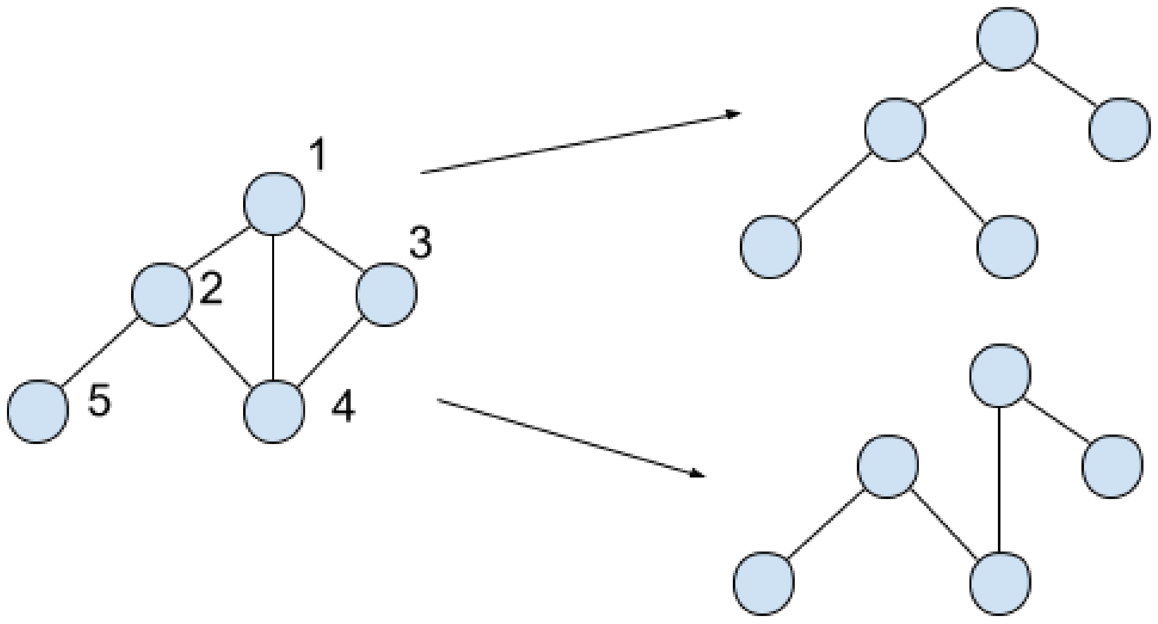
The ordering in which the nodes are visited by DFS algorithm is called the pre-ordering. In the example, the actual pre-ordering of the nodes (order in which nodes were marked visited) is 1-2-3-4-5-6-7.

Another ordering that can be generated from DFS is the post-ordering, the order in which the DFSvisit(u) finishes. For the previous example, the post-order is 3-4-2-7-6-5-1.

We can also consider DFS as running on a stack (because recursive calls are implemented using stacks). In this sense, pre-order is the ordering in which the vertices are pushed onto the stack, and post-order is the ordering in which the vertices are popped from the stack.

11.2.2 DFS tree is not necessarily unique

In the following figure we can visualize two possible DFS trees of a given graph



A possible algorithm for generating the upper tree would be the aforementioned DFS starting with node 3 as the root.

The lower tree could be generated with a different algorithm that went in order 5-2-4-1-3 or 3-1-4-2-5