San Skulrattanakulchai
Gustavus Adolphus College

Depth-first Search

CLRS: Ch 22.3

Throughout this handout $G = (V, E)$ can be directed or undirected.

Depth-first search (DFS) is a method of *graph searching* or *graph exploration* that is suited for computing graph connectivity properties, testing whether a graph is planar, solving many other graph problems, and serving as a subroutine to other algorithms.

A graph search "visits" all the vertices and "scans" all the edges.

DFS idea: Repeatedly scan an edge that is incident to the most recently discovered vertex with unscanned edges.

Recursive Implementation of this idea.

```
DFS(v) {
    D:
    for each edge vw do {
        S:
        if vertex w has not been discovered then
            dfs(w)
    }
    F:
}
```

Label D is where vertex v is discovered. Label S is where each edge incident to v is being scanned. Label F is where the visit of vertex v is about to finish. All depth-first search algorithms have this same structure. They differ in their codes for label D, S and F.

The main procedure (method) starts the search by calling DFS(s) from some starting vertex s.

Two ways to conceptualize DFS:

1. Path view of DFS

A *DFS path* is the path of edges the search traverses to discover a vertex v.

At any given point in time, the sequence of vertices in the DFS path corresponds to the vertices on the recursion stack of DFS.

2. Tree view of DFS

All the DFS paths together can be represented by a forest (tree if G is connected).
The DFS forest consists of all the edges that lead to the discovery of some new vertex.
A DFS tree is the recursion tree of DFS. Node v is the parent of node w if DFS(w) is called from DFS(v).

Notes

1. In general, to ensure that the search explores the entire graph, the main procedure is written

```
for each vertex v do
    if v has not been discovered then
        dfs(v)
```

2. DFS can be implemented in time $O(m + n)$.

3. We can test whether a graph is connected in time $O(m + n)$ by DFS. For an unconnected graph, we can find the connected components in this same time.

4. We can test if all vertices of a digraph are reachable from a source vertex s in time $O(m + n)$ by DFS.

Exercises

1. Write a DFS algorithm to find the connected components of an undirected graph.

2. Write a DFS algorithm to solve the Reachability Problem.

3. Write a DFS algorithm to test whether a given undirected graph is bipartite.