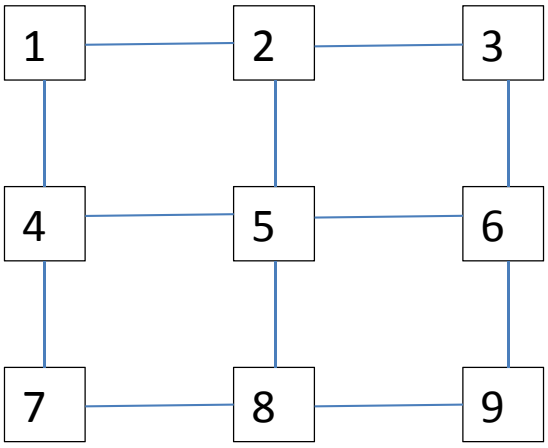
DFS

Depth-First Search in Graphs:

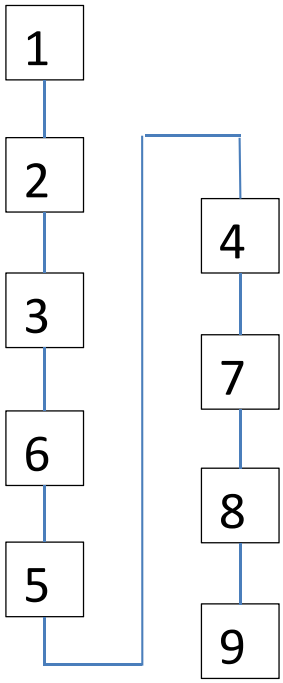
DFS is most similar to pre-order traversal of a tree, and can also be similar to post-order traversal

```
boolean seen[1…n];

DFS (Graph G) {
    for (k = 1; k<=n; k++)
        seen[k] = false;
    choose start vertex;
    DFS (G, start);
}

DFS (G, x) {
    seen[x] = true;
    preVisit (x);          //  visits nodes in DFS order or pre-order
    for each vertex y such that (x,y) is an edge
        if (! seen[y])  {
            //  optionally add edge (x,y) to the DFS tree;
            DFS (G, y);
        }
    postVisit (x);         //  optional, visits nodes in post-order
}
```

Example: Undirected Graph

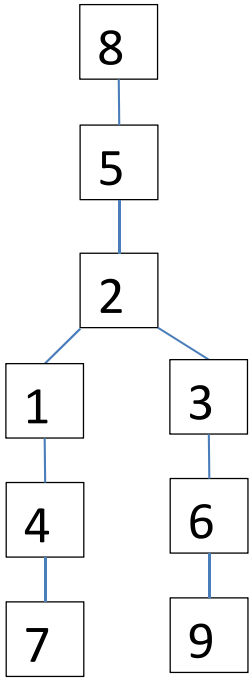


DFS starting at vertex 1:



DFS order: 1,2,3,6,5,4,7,8,9

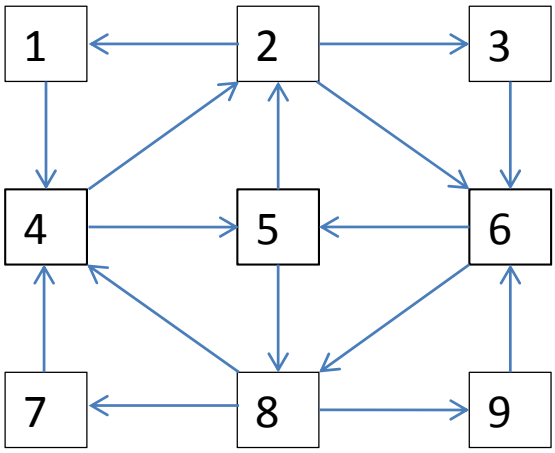
DFS starting at vertex 8:



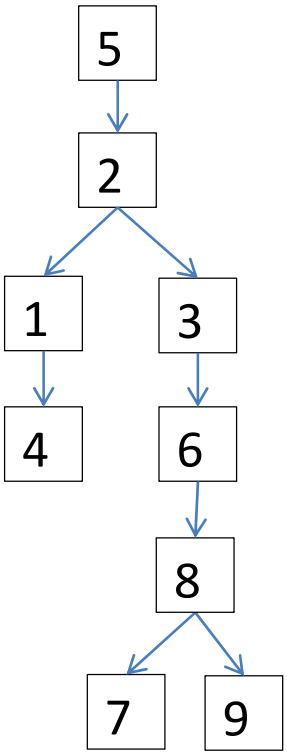
DFS order: 8,5,2,1,4,7,3,6,9

(DFS order is the same as pre-order of the DFS tree)

Example: Directed Graph

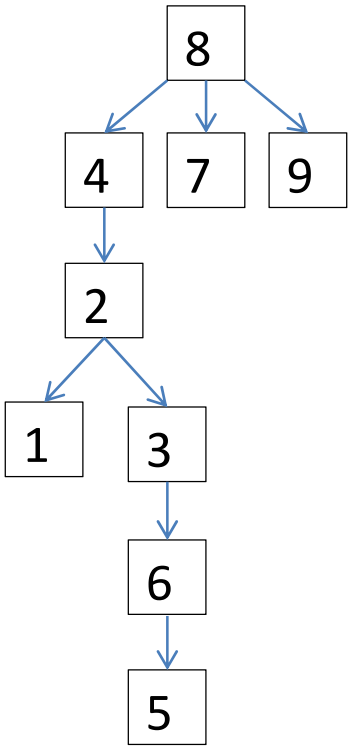


DFS starting at vertex 5:



DFS order: 5,2,1,4,3,6,8,7,9

DFS starting at vertex 8:



DFS order: 8,4,2,1,3,6,5,7,9

(DFS order is the same as pre-order of the DFS tree)

Analysis of DFS (same as for BFS):

If graph represented using adjacency matrix:

$\theta(n^2)$ time

If graph represented using adjacency lists:

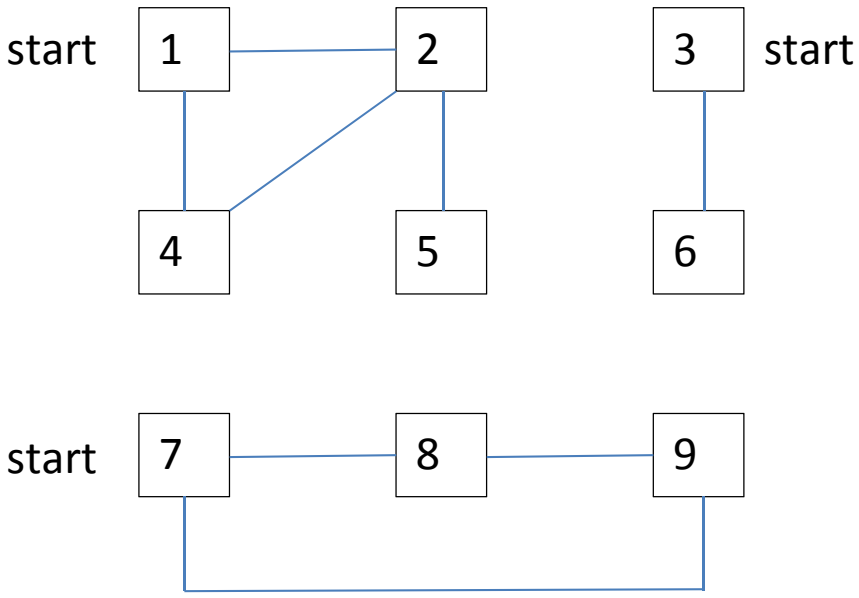
$\theta(n+m)$ time

Simplifies to $\theta(m)$ time if graph is connected, because $m \geq n-1$ for connected graphs

Note: it is more efficient to use adjacency lists, because $m \leq n^2$ for all graphs

Applications of DFS:

- Connectedness of undirected graph (same as for BFS):
  Is the graph connected?
  If not, find all the connected components



Components: {1,2,4,5} {3,6} {7,8,9}

More applications of DFS (we'll discuss these next time):

- Strong connectedness of directed graph:
  Is the graph strongly connected?
  If not, find all the strongly connected components

- Detecting a cycle (in undirected or directed graph)

- Topological sort of a Directed Acyclic Graph:
  a linear ordering of the vertices so that whenever edge $x \rightarrow y$ exists, vertex x must precede vertex y