Graph Traversals

→ a traversal is a systematic procedure for exploring a graph by examining all its vertices and edges

→ reachability: graph traversals are key to answering many fundamental questions about reachability in graphs

Depth-First Search (DFS)

→ can be useful for testing properties of graphs, such as whether there is a path between two vertices

→ how it works:

- begin at specific starting vertex s in G -- now our 'curr' vertex u
- we then traverse G by considering arbitrary edge (u,v) leads to unvisited vertex v
  - if edge leads us to vertex v that is already visited, ignore
  - if unvisited, then visit v and make it curr
- eventually we get to a dead end -- current vertex v such that all edges incident to v lead back to vertices already listed
- to get out of this, backtrack along edge which brought us to v, going back to previously visited u
- make u the curr vertex then repeat computation for any edges incident to u which haven't been considered
- if all of the edges lead to visited vertices, backtrack to vertex we came from to get u, and repeat
- this process terminates when backtracking leads us back to start vertex u

```
DFS(G,u):
  for each outgoing edge e=(u,v) of u do
    if vertex v has not been visited
      mark vertex as visited (via edge e)
      recursively call DFS(G,v)
```

Classifying Graph Edges using DFS

→ whenever an edge e=(u,v) is used to discover a new vertex v using DFS, it is called a discovery/tree edge

- all other edges considered during execution are nontree edges
  - back edges: connect a vertex to an ancestor in DFS tree
  - forward edges: connect a vertex to a descendant in DFS tree
  - cross edges: connect vertex to a vertex which is neither ancestor/descendant