CALIFORNIA STATE UNIVERSITY
FULLERTON

CPSC 131

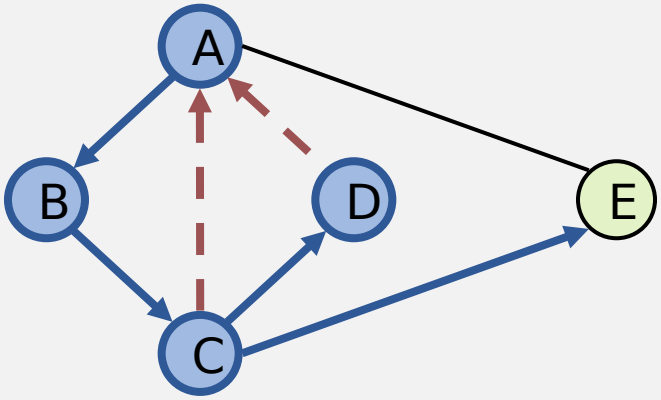
Data Structures

Graph Traversals
Depth-First Search

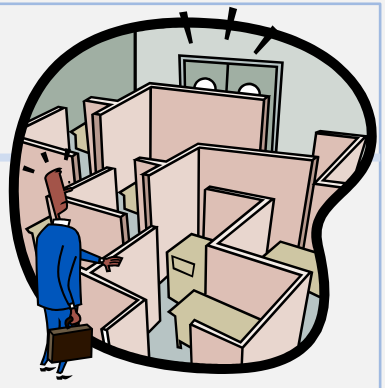
Graph Traversals

- A systematic procedure for exploring a graph by examining all of its vertices and edges
- Traversal algorithms
  - Depth-First Search (DFS)
    - Visits the child vertices before visiting the sibling vertices
    - A stack is used when implementing DFS
  - Breadth-First Search (BFS)
    - Visits the neighbor vertices before visiting the child vertices
    - A queue is used in the search process

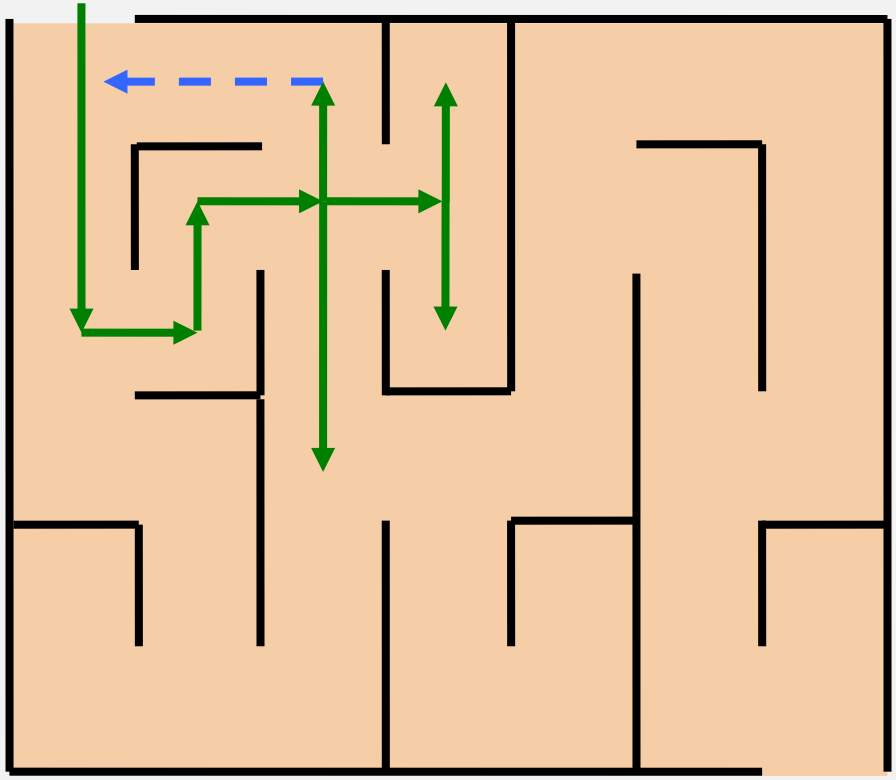
Depth-First Search (DFS)



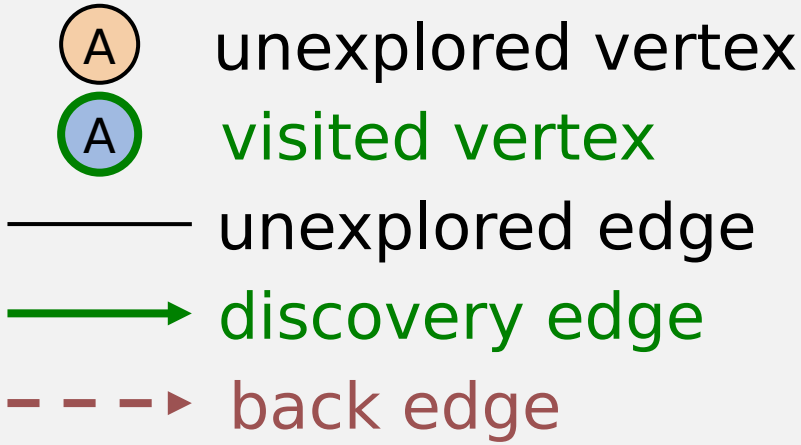
Depth-First Search

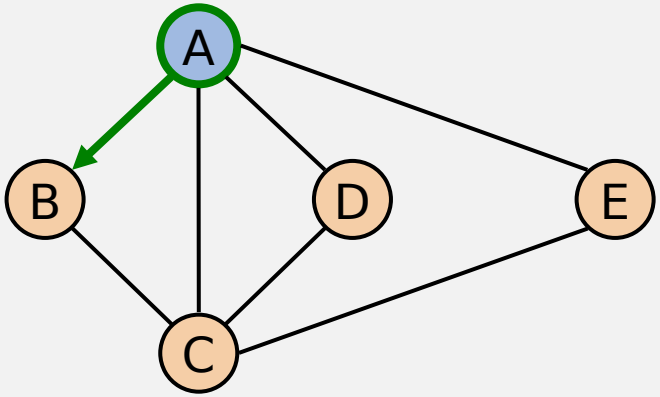


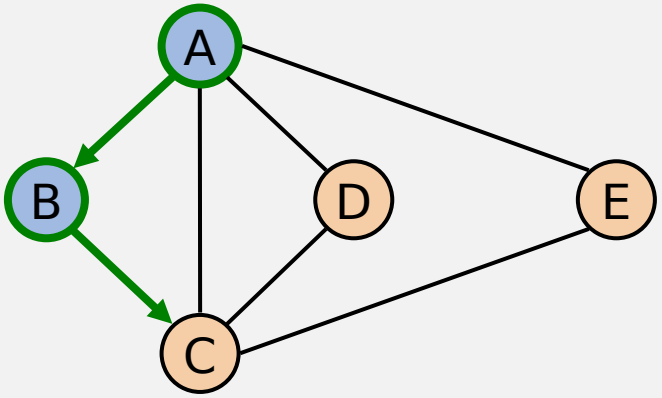
- The DFS algorithm is similar to a classic strategy for exploring a maze
  - Mark each intersection, corner and dead end (vertex) visited
  - Mark each corridor (edge) traversed
  - Keep track of the path back to the entrance (start vertex) by means of a rope (recursion stack)

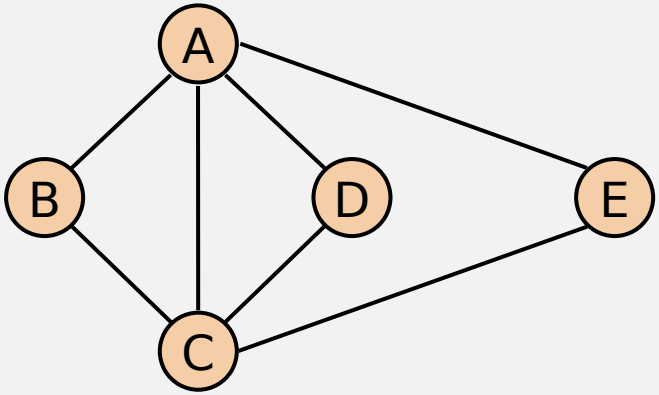


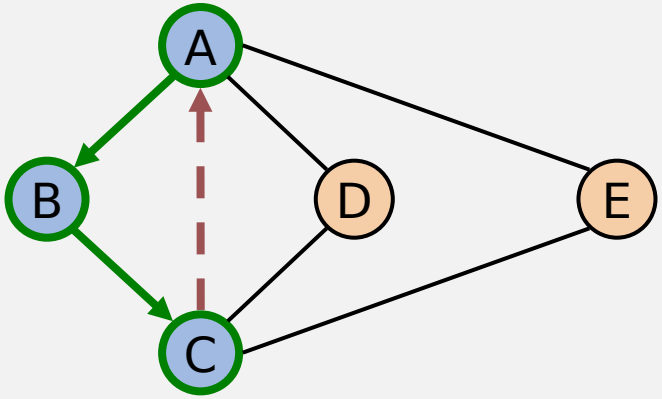
DFS Traversal
Terminologies & Sketches











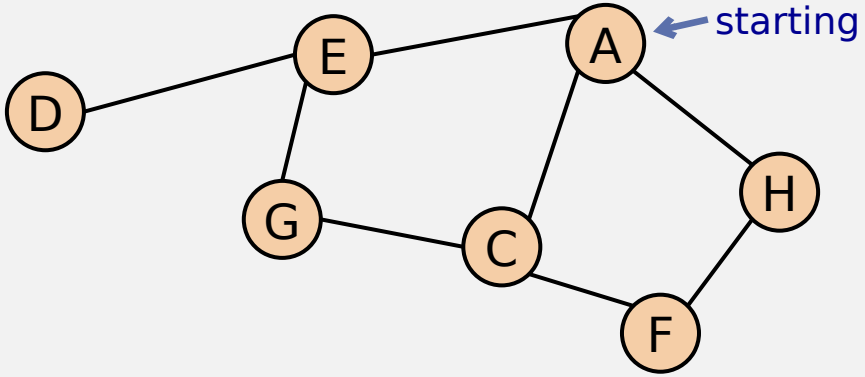
DFS Algorithm Pseudo Code

```
procedure DFS-recursive(v):
    if v is not visited then
                    label v as visited
      for all nodes w adjacent to v do
        call DFS-recursive(w)
```

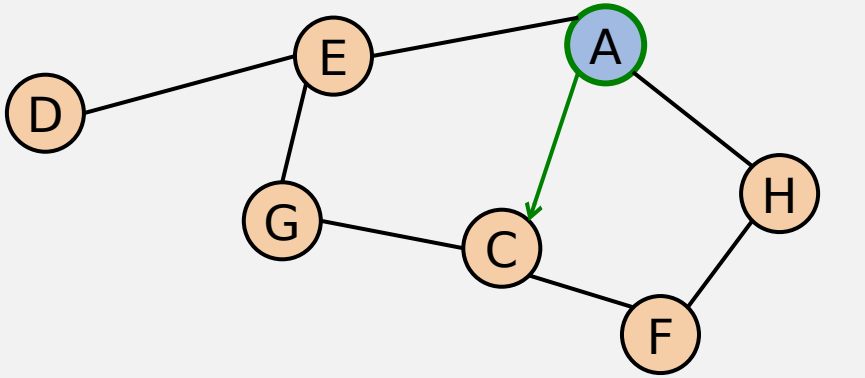
The algorithm sets and gets labels of vertices as “visited” or “not visited”

Lexicograhic order

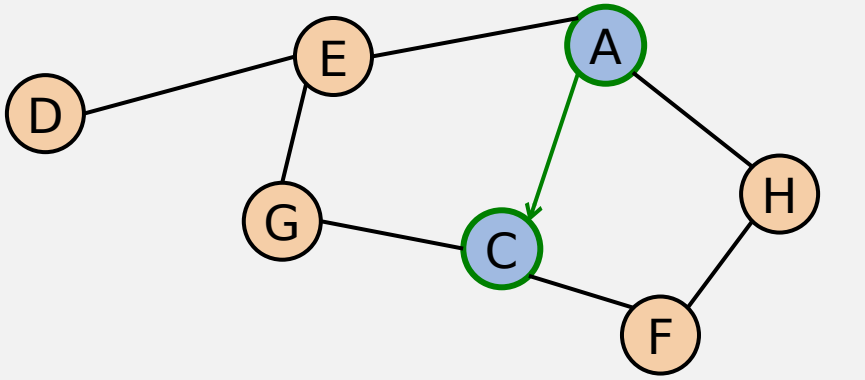
- If there are multiple vertices to choose from, which do we choose first?
- Any order will give a correct traversal
- For consistency in this class, we will select vertices in alphabetical order
  - Also called lexicographic order

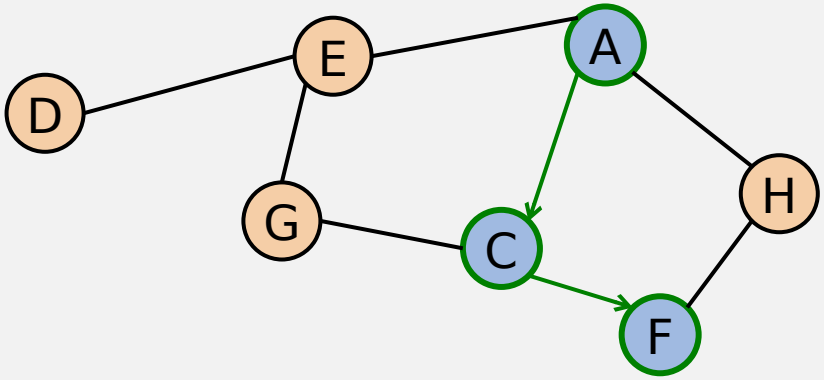


DFS Example

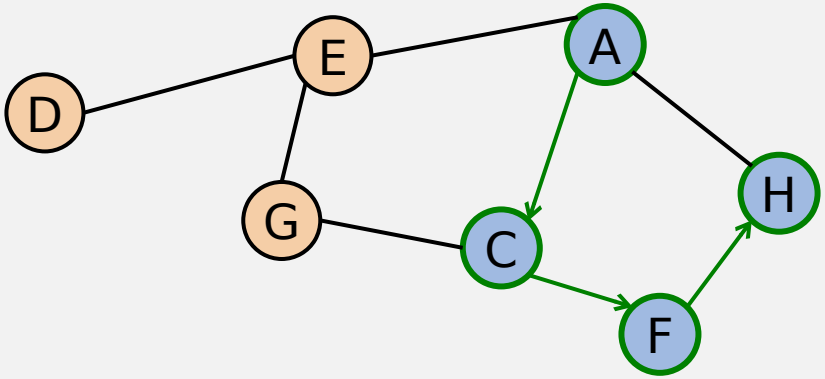


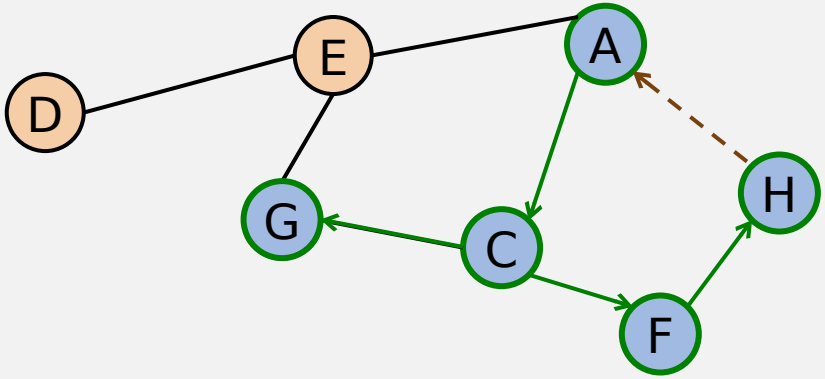
Visited: A C F

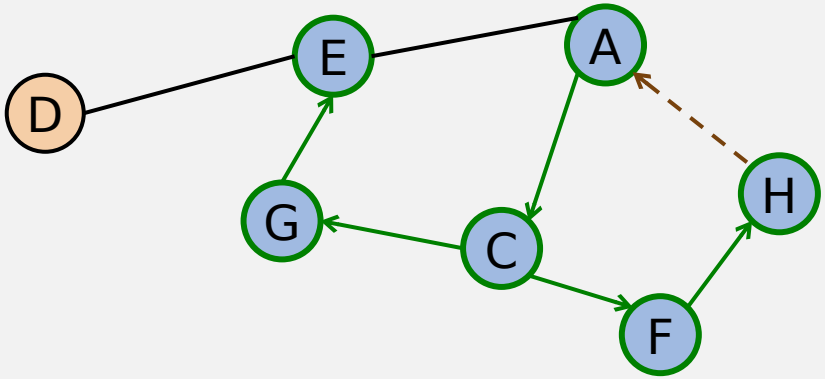




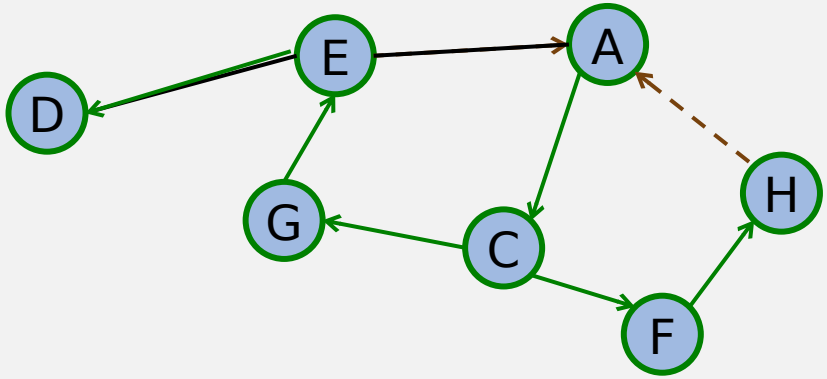
DFS Example



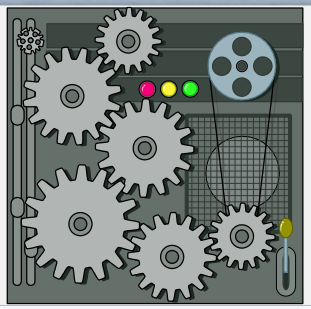




Visited: A C F H G E D



Analysis of DFS



- Setting/getting a vertex/edge label takes $O(1)$ time
- Each vertex is labeled twice
  - initially as NOT_VISITED
  - once as VISITED

  } $O(n)$
- Each edge is labeled twice
  - once as UNEXPLORED
  - once as DISCOVERY or BACK

  } $O(m)$
- Method incidentEdges is called **once for each vertex**
  - incidentEdges() returns a list of all edges touching vertex v

  } $O(m)$
  Since:
  $\Sigma_v \deg(v) = 2m$
- DFS runs in $O(n + m)$ time provided that the graph is represented by the adjacency list structure

DFS Stack-based Algorithm

```
DFS (startV) {
   Set all vertices to not_visited
   Push startV to stack



    while ( stack is not empty )
     currentV = Pop stack
     "Visit" currentV

     if ( currentV is not_visited)
        set currentV to visited
        for each vertex adjV adjacent to currentV  do
              Push adjV to stack
}
```