DATA STRUCTURES (ITPC-203)

Graphs – Part II

Dr. Sanga Chaki

Department of Information Technology

Dr. B. R. Ambedkar National Institute of Technology, Jalandhar

Contents

- Recap of graphs
- Graph Traversal :
  - Depth First Search,
  - Breadth First Search

Graphs - Recap

1. Graphs are non-linear data structures consisting of vertices and edges
2. Edges connect the vertices.
3. Types of graphs:
   - Finite and infinite
   - Directed and undirected
4. Adjacent vertices in Directed and undirected graphs
5. Incident edges in Directed and undirected graphs
6. Graph representations:
   - Adjacency matrices
   - Adjacency lists

Graph Traversals

1. Methods to explore a graph
2. Given a vertex in a directed or undirected graph we may wish to visit all vertices in the graph that are reachable from this vertex.
3. This can be done in two ways:
   a) Depth First Search algorithm
   b) Breadth First Search algorithm.

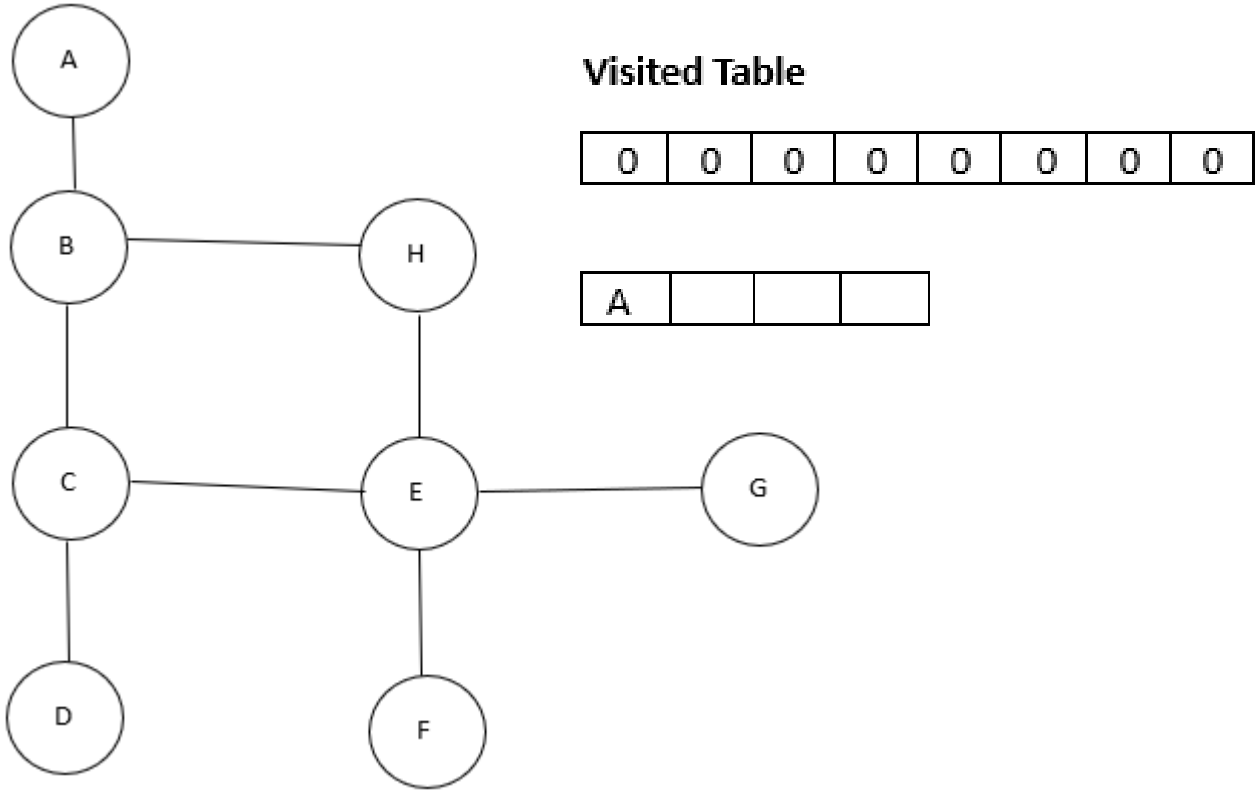
Breadth First Search (BFS) Algorithm

1. First explore all the nodes one step away, then all nodes two step away etc.
2. Idea:
   i. Start at a node which is at level 0
   ii. Visit all vertices at level 1 – vertices whose distance is 1 from the start vertex
   iii. Visit all vertices at level 2
   iv. Terminate when all levels are visited
3. Use a queue internally
4. To avoid processing a node more than once, we divide the vertices into two categories:
   - Visited,
   - Not visited.

BFS – Pseudocode and Example

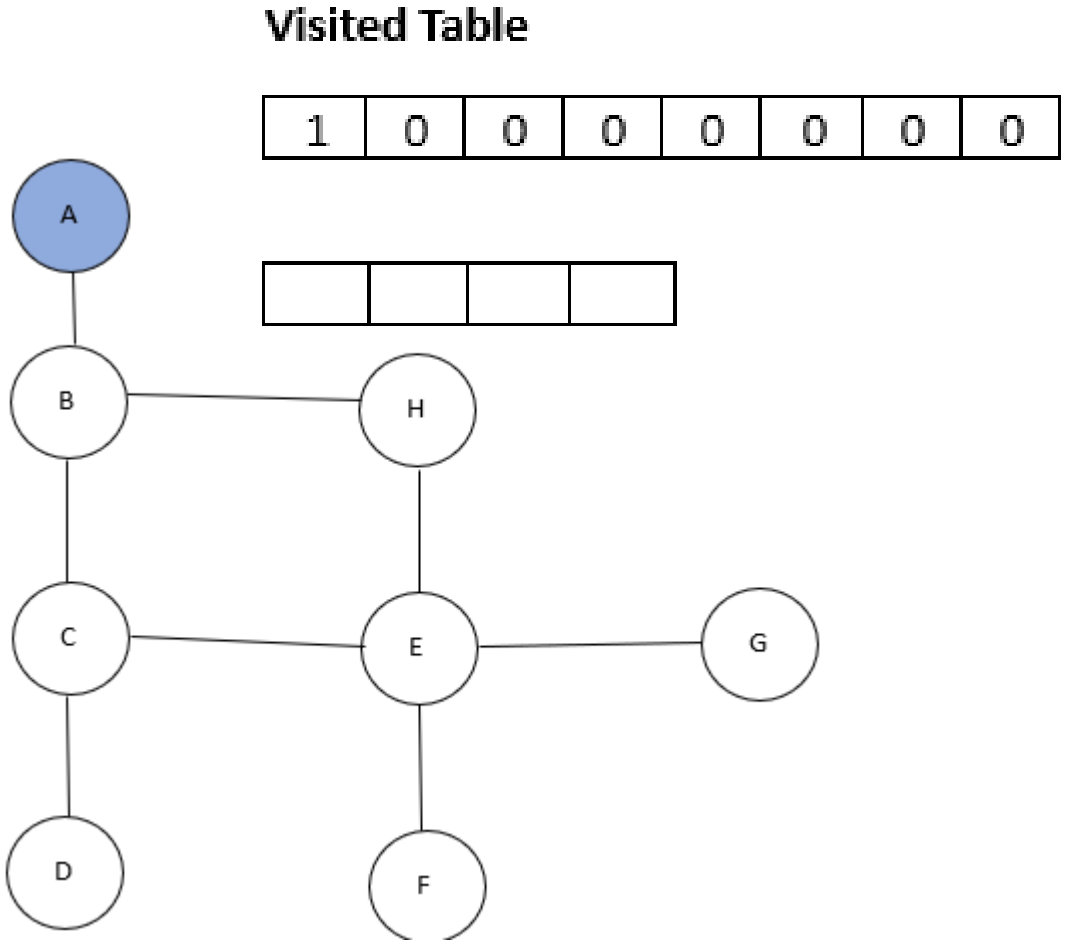
1. Start from A (level 0)
2. Insert A in queue.
3. No other nodes in this level

.



1. Dequeue A, display A, mark A visited

.



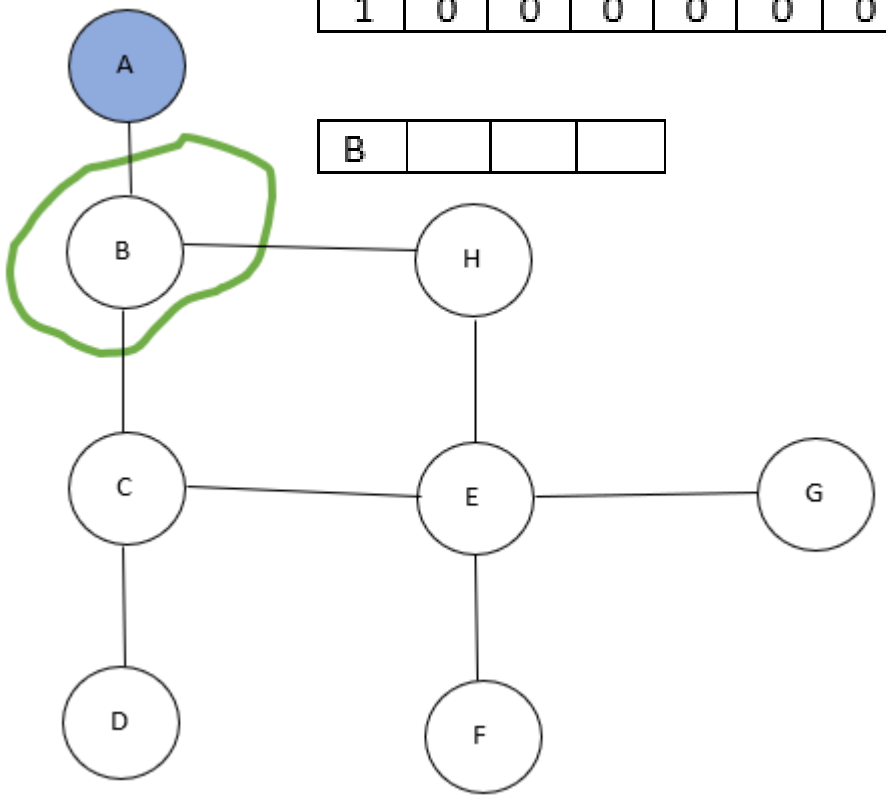
BFS – Pseudocode and Example

1. Find unvisited adjacent node of A = B.
2. Level of B = 1. Enqueue B.
3. No other nodes in this level

Visited Table

1	0	0	0	0	0	0	0

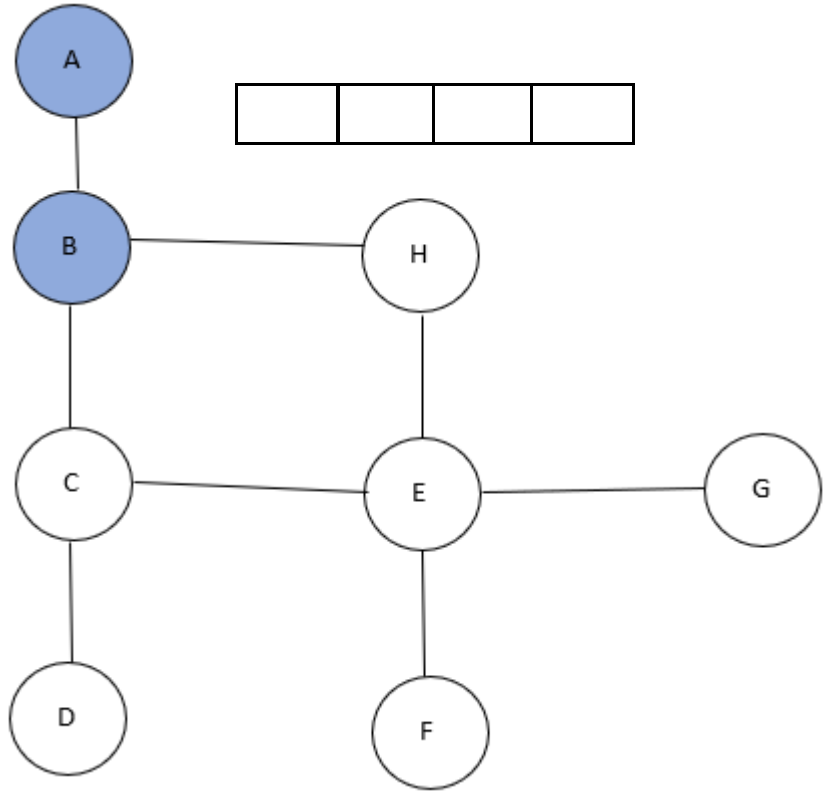
B			



1. Dequeue B, display B, mark B visited

Visited Table

1	1	0	0	0	0	0	0



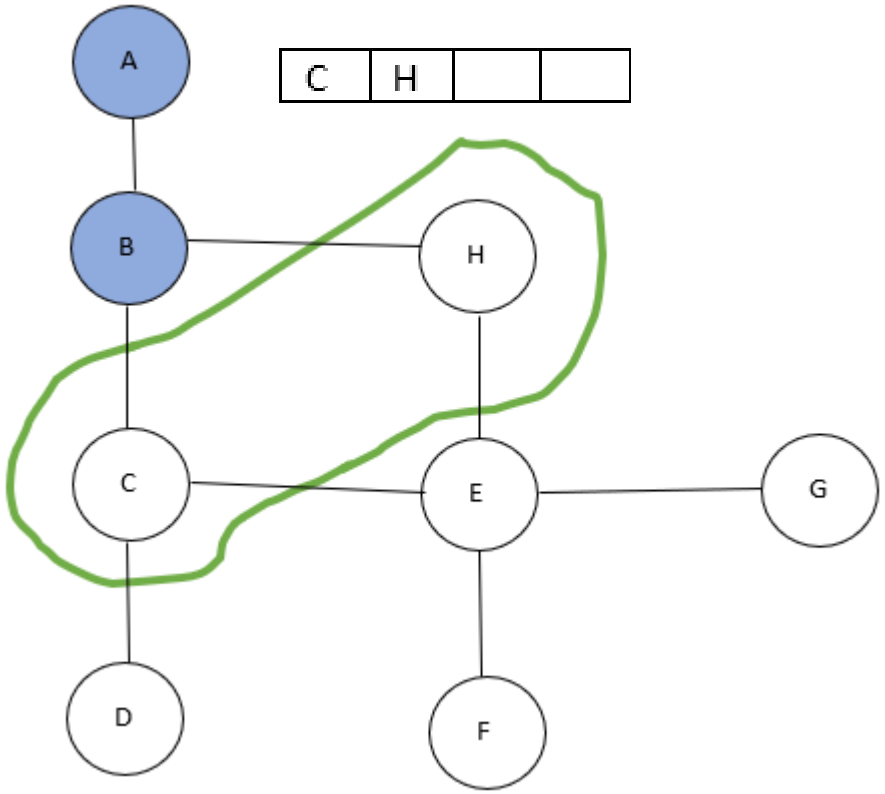
BFS – Pseudocode and Example

1. Look for unvisited adjacent node to B = C and H (Level 2)
2. Enqueue C and H

Visited Table

1	1	0	0	0	0	0	0

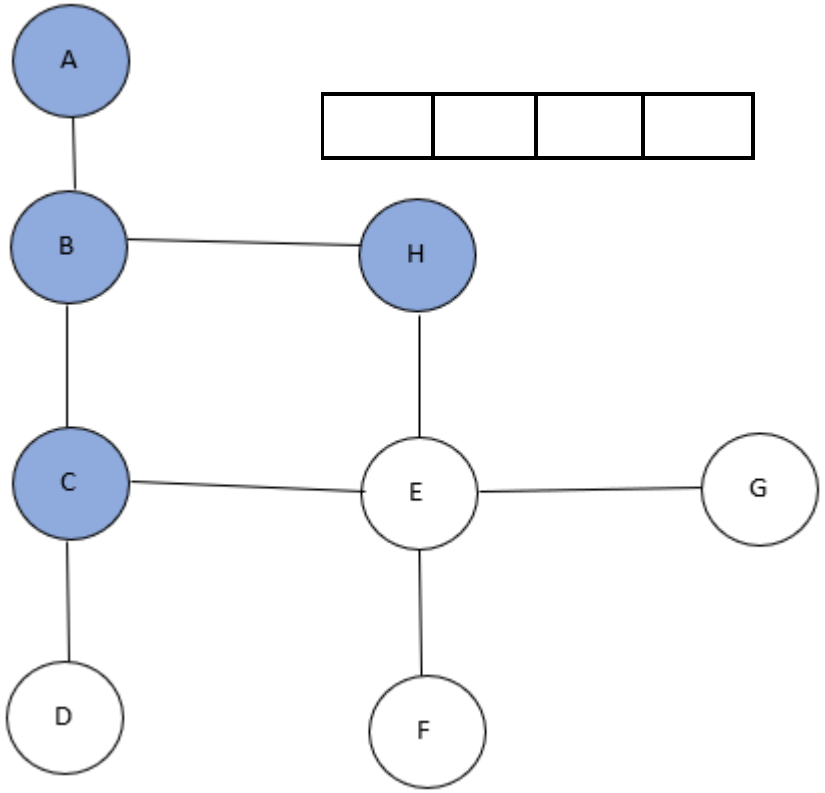
C	H		



1. Dequeue C, display C, mark it visited
2. Dequeue H, display H, mark it visited

Visited Table

1	1	1	0	0	0	0	1



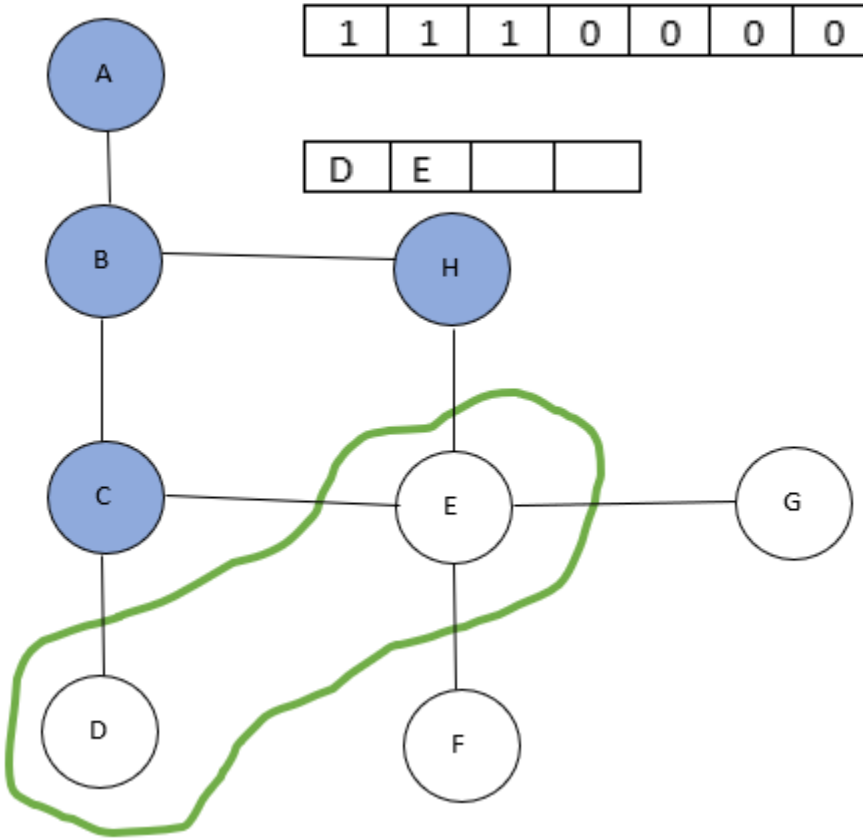
BFS – Pseudocode and Example

1. Look for unvisited adjacent node to C & H
2. Enqueue D and E for C. No unvisited adjacent node for H.

Visited Table

1	1	1	0	0	0	0	1

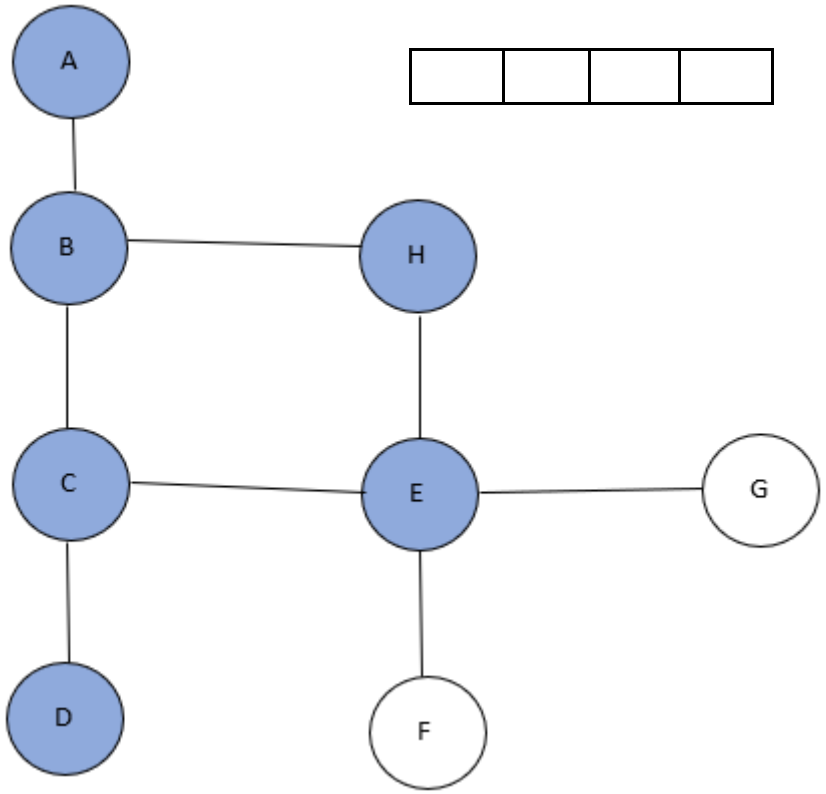
D	E		



1. Dequeue & display D, mark it visited
2. Dequeue & display E, mark it visited,

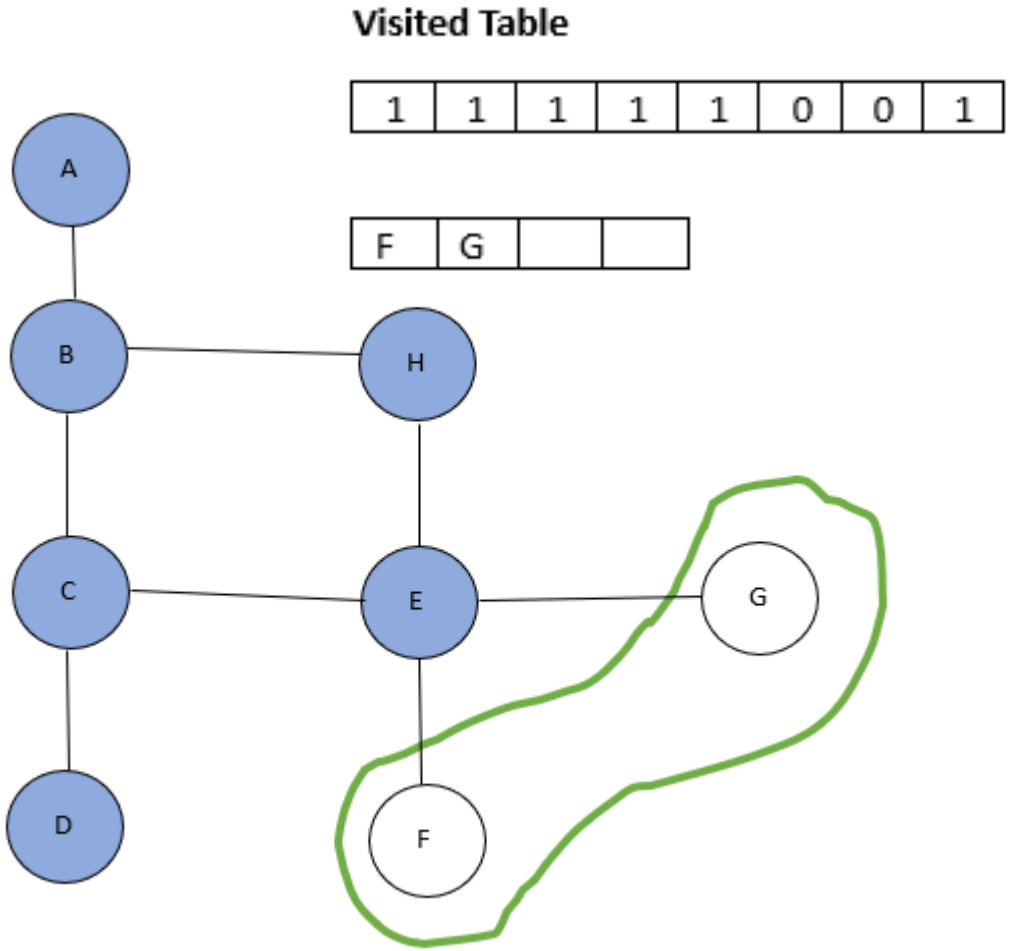
Visited Table

1	1	1	1	1	0	0	1

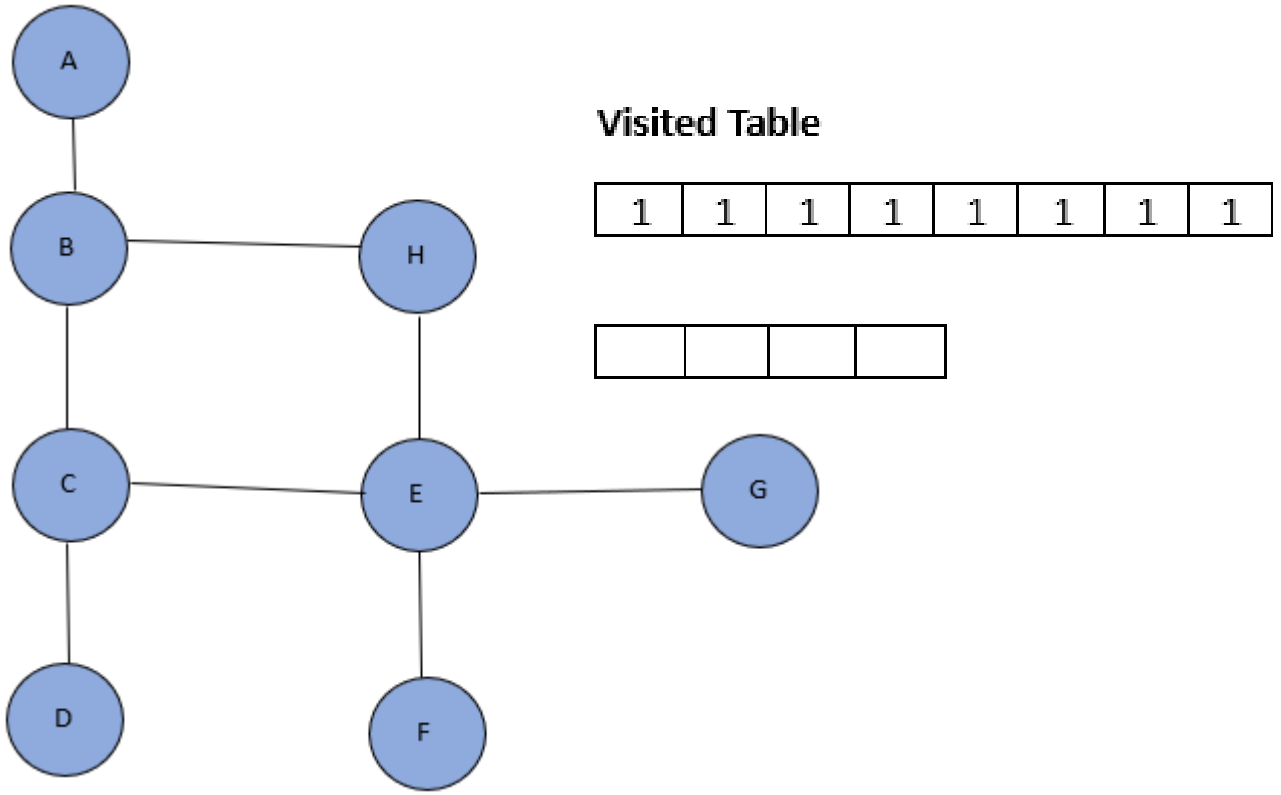


BFS – Pseudocode and Example

1. Look for unvisited adjacent node to D & E
2. Enqueue F and G for E. No unvisited adjacent node for D.



1. Dequeue & display F, mark it visited
2. Dequeue & display G, mark it visited
3. No unvisited vertices left – terminate
4. **BFS Sequence: A, B, C, H, D, E, F, G**



BFS - Properties

1. Time complexity using adjacency lists = O(V+E)
2. Time complexity using adjacency matrices = $O(V^2)$
3. Advantages: A BFS will find the shortest path between the starting node and any other reachable node.

Depth First Search

1. Go as deep as possible down one path in a graph, before backing up and trying a new path
2. Uses stacks internally
3. Requires a visited/unvisited array to keep track of visited/unvisited nodes of graph
4. Initially, all vertices are marked unvisited (false)

Depth First Search - Pseudocode

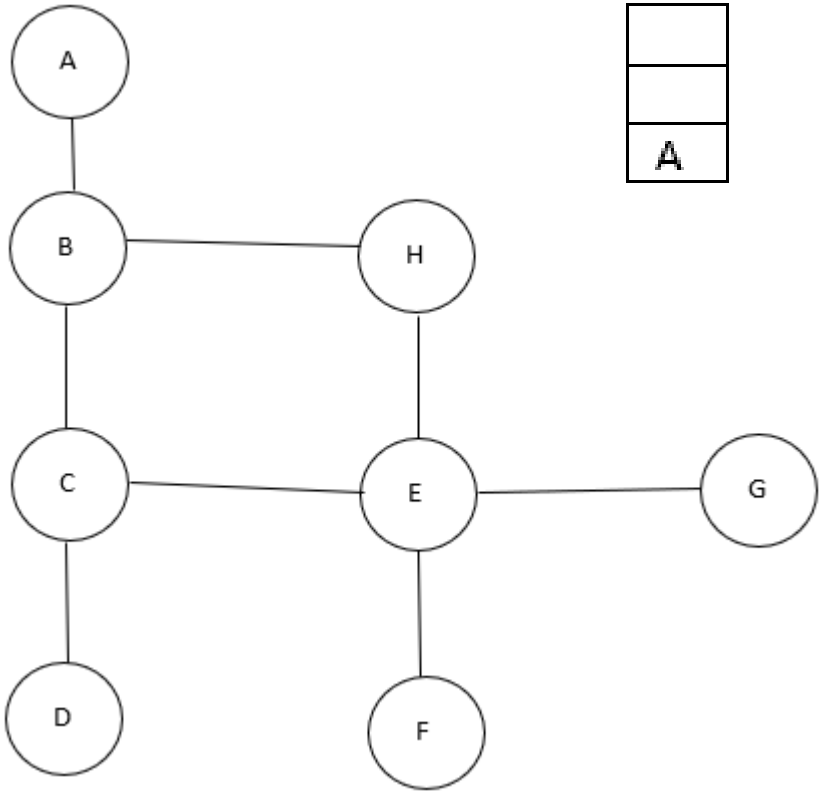
1. Initially, all vertices are marked unvisited (false)
2. DFS starts at vertex u
3. Consider the edges from u to other vertices
   - If edge leads to visited vertex, backtrack to u
   - If edge leads to unvisited vertex, go to new vertex and start processing from that vertex. So, new vertex becomes the current vertex
4. Follow this procedure until you reach a dead end. Start backtracking
5. End procedure when backtracking leads back to the start vertex

Depth First Search – Example and Pseudocode

1. Start at A – update visited table, display it, push it into a stack.
2. Visit adjacent unvisited vertex, mark it visited, display it, push it into a stack.
3. If no adjacent vertex found, pop a vertex from the stack – it will pop all vertices which do not have adjacent vertices
4. Repeat until stack is empty

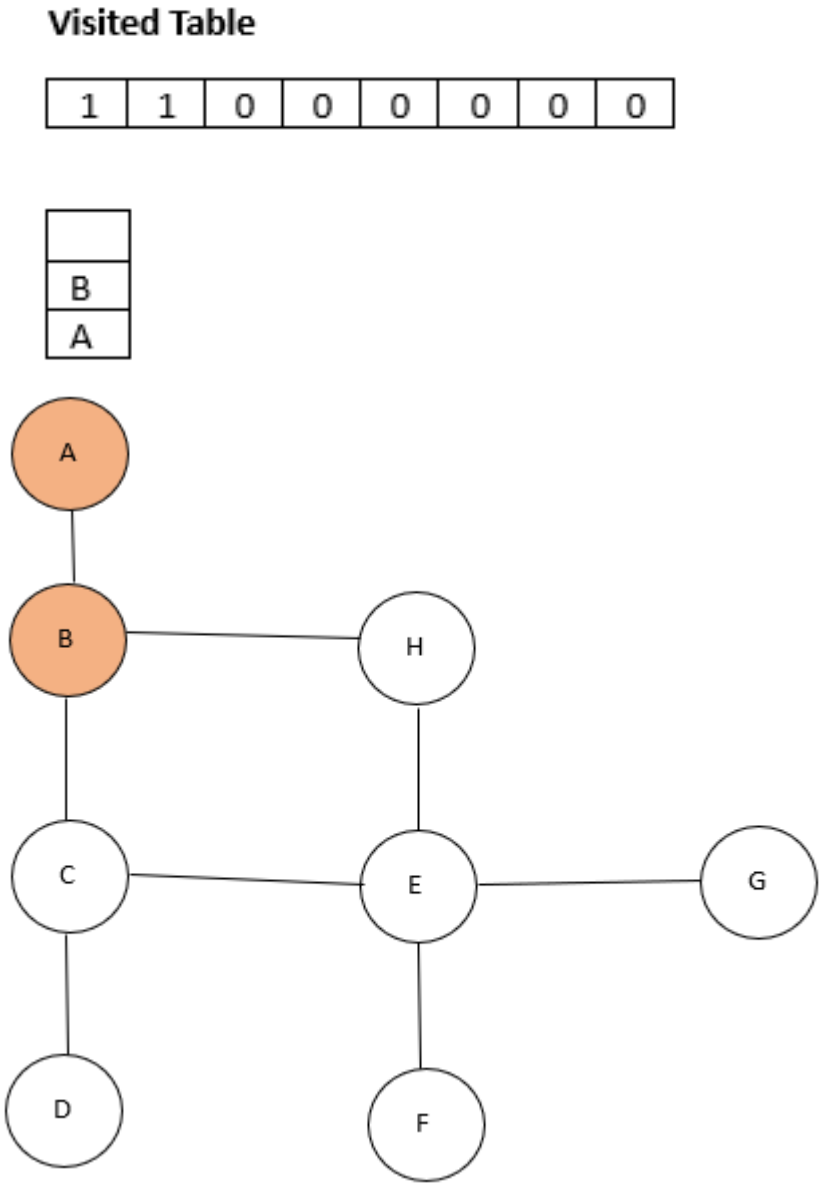
Visited Table

1	0	0	0	0	0	0	0

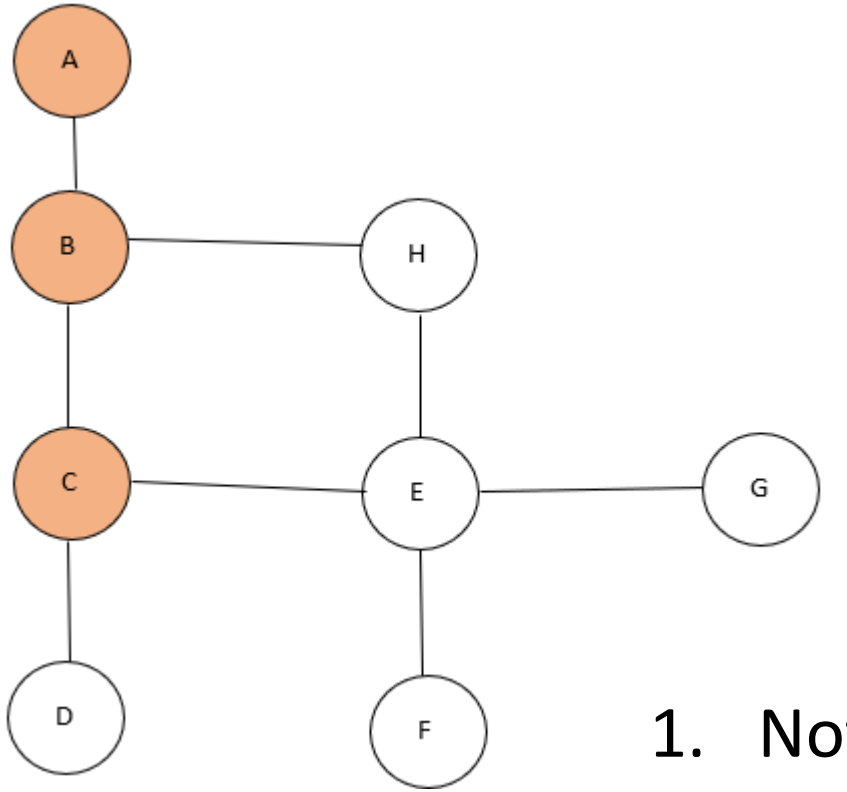


Depth First Search – Example and Pseudocode

1. Start at A – update visited table, display it, push it into a stack.
2. Visit adjacent unvisited vertex, mark it visited, display it, push it into a stack.
3. If no adjacent vertex found, pop a vertex from the stack – it will pop all vertices which do not have adjacent vertices
4. Repeat until stack is empty



Depth First Search – Example and Pseudocode



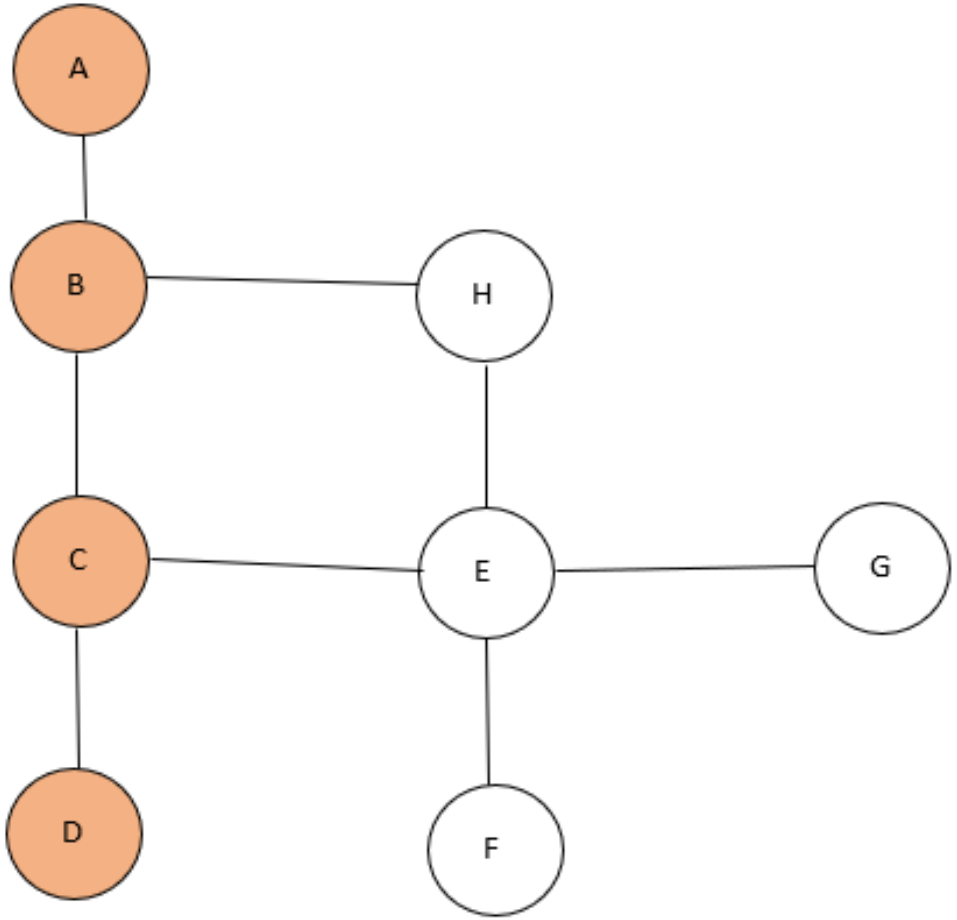
Visited Table

1	1	1	0	0	0	0	0

C
B
A

1. Note: We could have chosen H in place of C also.
2. Exercise: How would the traversal change in that case?

Depth First Search – Example and Pseudocode



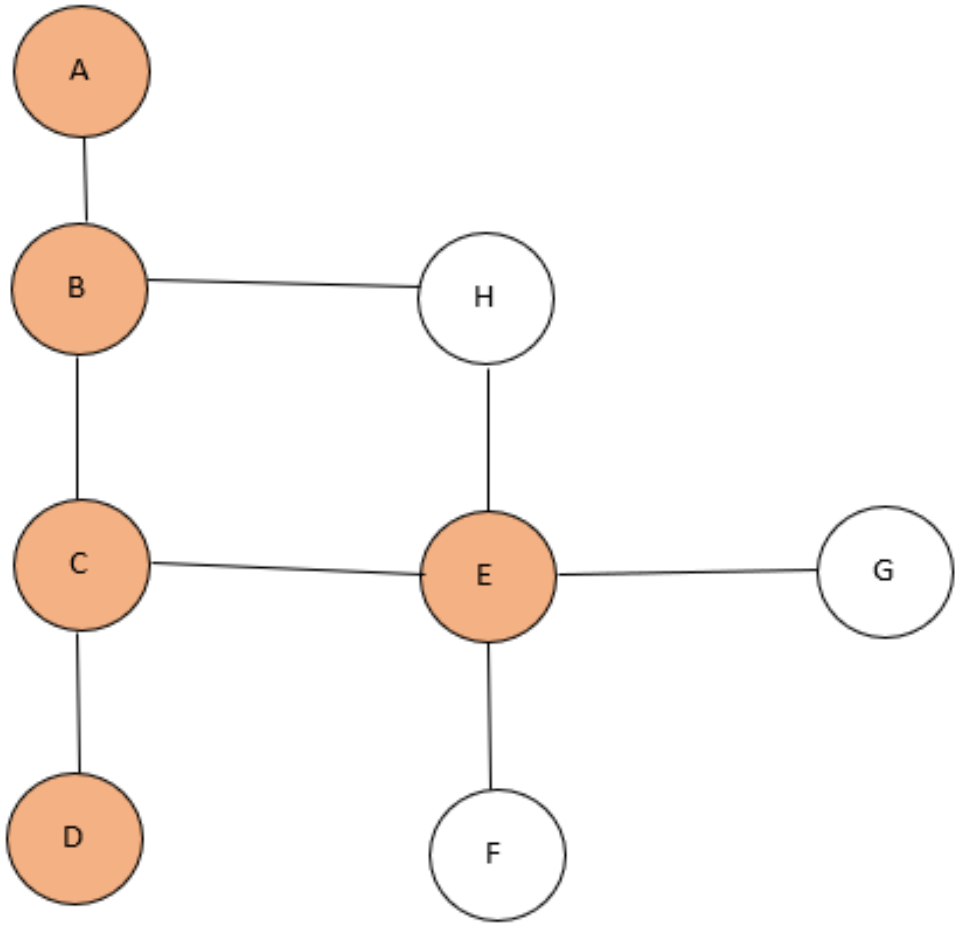
Visited Table

1	1	1	1	0	0	0	0

D
C
B
A

1. Now, D does not have any unvisited adjacent vertex, so pop D from stack.
2. C is the new stack top, check C for unvisited adjacent vertex
3. E is unvisited
4. Visit E

Depth First Search – Example and Pseudocode

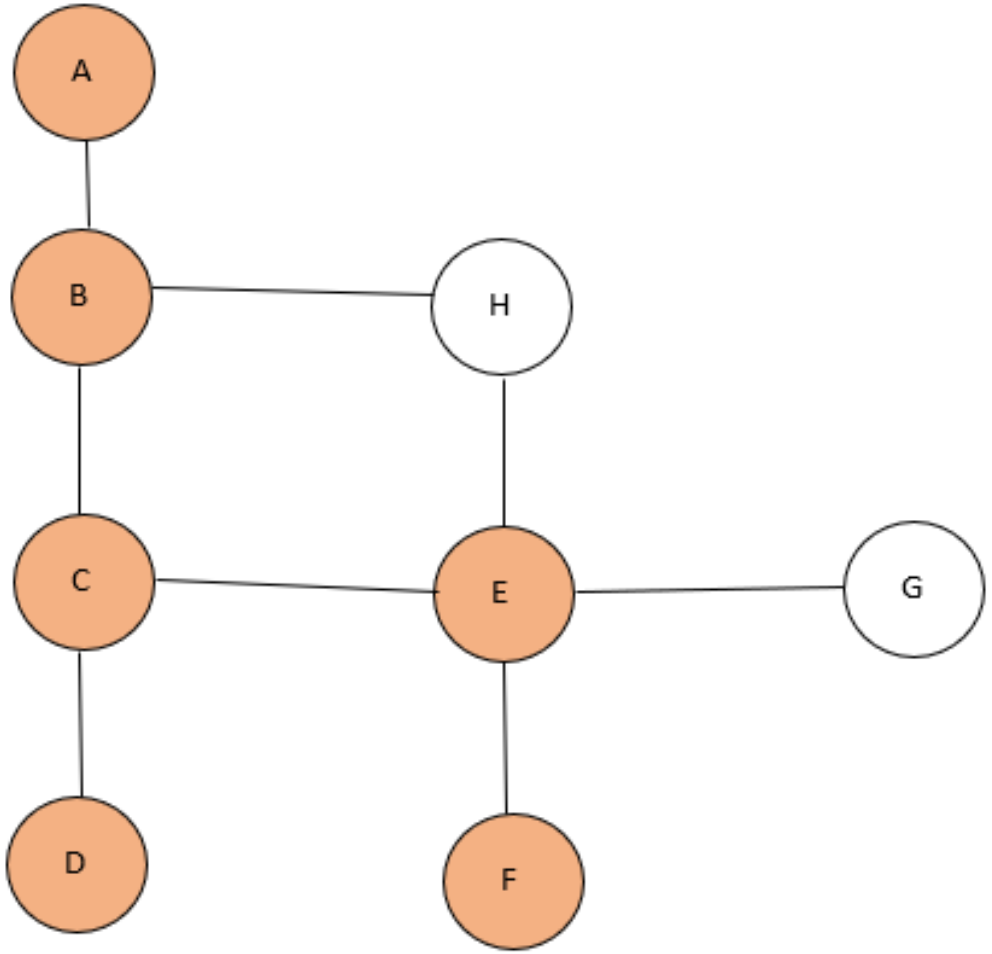


Visited Table

1	1	1	1	1	0	0	0

E
C
B
A

Depth First Search – Example and Pseudocode



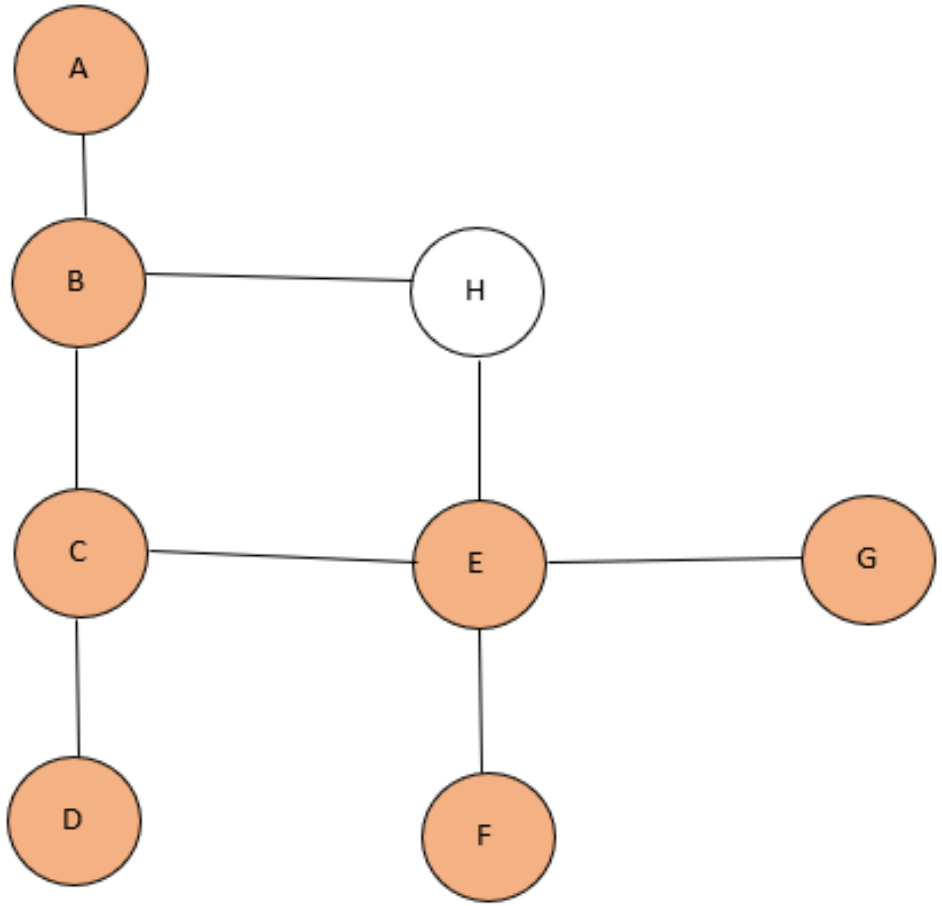
Visited Table

1	1	1	1	1	1	0	0

F
E
C
B
A

1. Again, F does not have any unvisited adjacent vertex, so pop F from stack.
2. E is the new stack top, check E for unvisited adjacent vertex
3. G is unvisited (you could have chosen H as well)
4. Visit G

Depth First Search – Example and Pseudocode



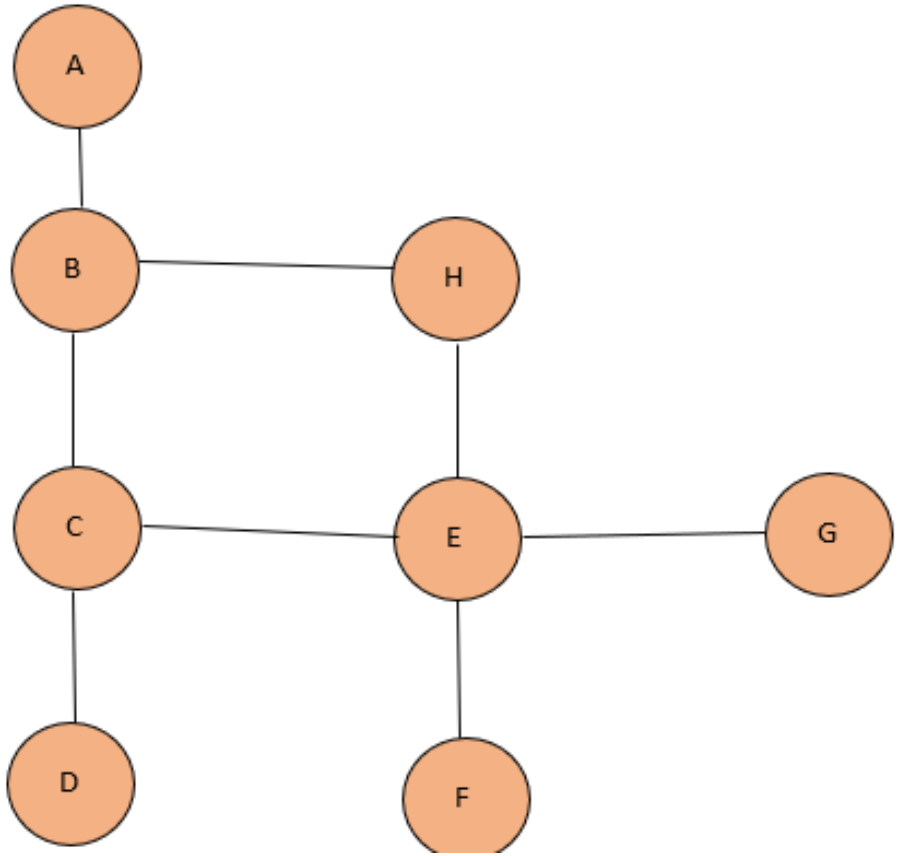
Visited Table

1	1	1	1	1	1	1	0

G
E
C
B
A

1. Again, G does not have any unvisited adjacent vertex, so pop G from stack.
2. E is the new stack top, check E for unvisited adjacent vertex
3. H is unvisited
4. Visit H

Depth First Search – Example and Pseudocode



Visited Table

1	1	1	1	1	1	1	1

H
E
C
B
A

1. Again, H does not have any unvisited adjacent vertex, so pop H from stack.
2. E is the new stack top, check E for unvisited adjacent vertex
3. Again, E does not have any unvisited adjacent vertex, so pop E from stack
4. Similarly, pop C, then B, then A
5. Stack becomes empty
6. Terminate procedure
7. **DFS Sequence: A, B, C, D, E, F, G, H**

Depth First Search - Properties

1. Advantages:
   - Easy with recursion
   - May require less memory than BFS
2. Disadvantages:
   - Does not necessarily find shorted path to a node. BFS does.
3. What is the complexity?