Algorithms

Lecture 4
Graph Algorithms (1)

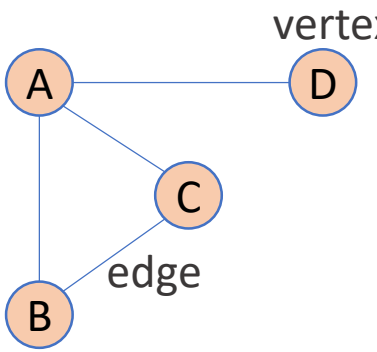
A. S. M. Sanwar Hosen

Email: sanwar@wsu.ac.kr

Date: 30 March, 2023

Graph Data Structure (1)

- **Definition:** A graph is a collection of nodes have data and are connected to other nodes. More precisely, a graph G is a pair, G = (V, E), where V is a finite nonempty set, called the set of vertices of G, and $E \subseteq V \times V$, E is called the set of edges.



A Graph

Vertices (V)= {A, B, C, D}
Edges (E) = {(A, B), (A, D), (A, C), (B, C)}
Graph (G) = {V,E}
Path P of (B to D) = {B, A, D} or {B, C, A, D}

- **Terminologies of a Graph**

✓ **Vertex or node (v):** A node in a graph that contains data and references to other nodes.

✓ **Edge (e):** A connection between two vertices.

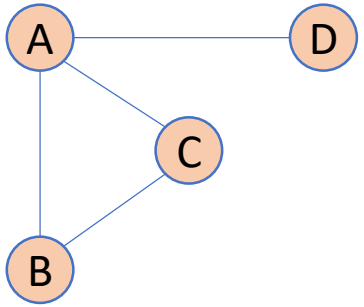
✓ **Path (P):** A sequence of edges that allow to traverse one vertex to another vertex.

✓ **Weight (w):** Indicates the strength of an edge in a graph.

Graph Data Structure (2)

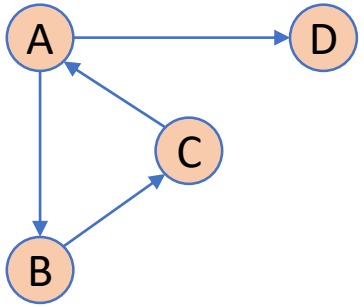
❑ Terminologies of a Graph (2)

✓ **Undirected Graph:** If the elements of edges of a graph *E*(G) are disordered pairs, the graph G is called an undirected graph. The edges indicate two way-relationship, in that each edge can be traversed in both directions. An example of an undirected graph as follows:



An Undirected Graph

✓ **Directed Graph:** If the elements of edges of graph G, *E*(G) are ordered pairs, G is called a directed or digraph. The edges indicate one way-relationship, in that each edge can only be traversed in a single direction. An example of a directed graph as follows:



A Directed Graph

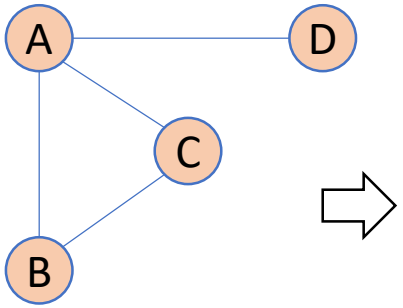
Graph Data Structure (3)

- **Graph Representation:** Graphs are commonly represented in two ways, as follows:

✓ **Adjacency Matrix:** Let G be a graph with n vertices, where $n > 0$. Let $V(G) = \{v_1, v_2, v_3, \dots, v_n\}$. The adjacency matrix A_G is a two-dimensional $n \times n$ matrix such that the (i, j)th entry of A_G is 1 if there is an edge from v_i to v_j ; otherwise, the (i, j)th entry is zero. That is,

$$A_G = \begin{cases} 1, if\ (v_i, v_j) \in E(G) \\ 0, otherwise \end{cases}$$

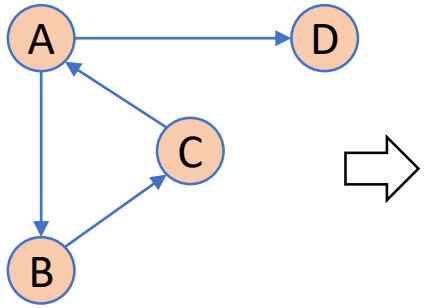
Example: Consider the following undirected and directed graphs, the adjacency matrices of the graphs are as follows:



An Undirected Graph

	A	B	C	D
A	0	1	1	1
B	1	0	1	0
C	1	1	0	0
D	1	0	0	0

The Adjacency Matrix



A Directed Graph

	A	B	C	D
A	0	1	0	1
B	0	0	1	0
C	1	0	0	0
D	0	0	0	0

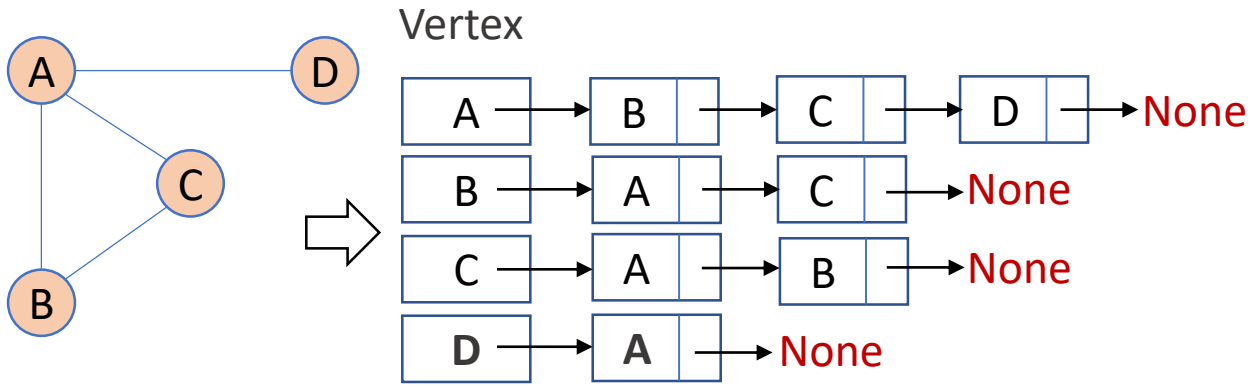
The Adjacency Matrix

Graph Data Structure (4)

- **Graph Representation:** Graphs are commonly represented in two ways, as follows:

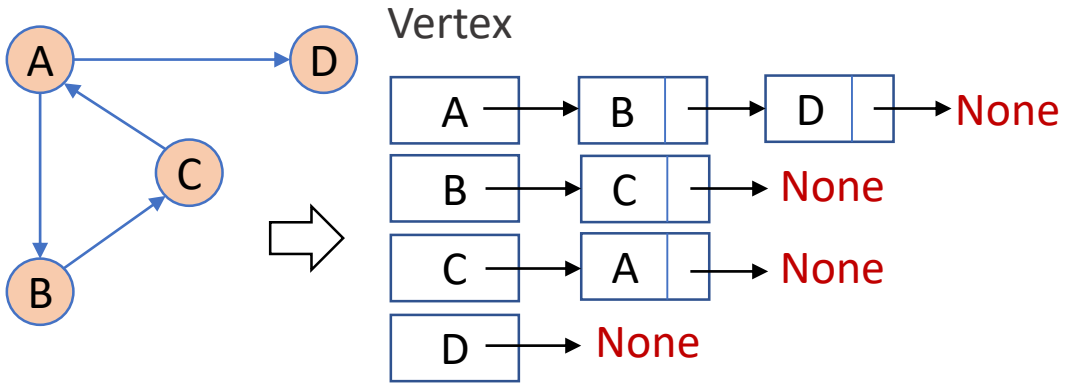
✓ **Adjacency List:** Let G be a graph with n vertices, where $n > 0$. Let $V(G) = \{v_1, v_2, v_3, \dots, v_n\}$. In the adjacency list, corresponding to each vertex, v, there is a linked list such that each node of the linked list contain the vertex u, such that $(v, u) \in E(G)$. Because, there are n nodes, we use an array of size A, of size n, such that $A[i]$ is a reference variable pointing to the first node of the linked list containing the vertices to which v_i is adjacent. Clearly, each node has to components, say **vertex**, and **link**, the component vertex contains the index of the vertex adjacent to i.

Example: Consider the following undirected and directed graph, the adjacency matrices of the graphs are as follows:



An Undirected Graph

The Adjacency List



A Directed Graph

The Adjacency List

Functions on Graphs Data Structure

- **Operations on Graphs:** The operations commonly performed on a graph are as follows:

- ✓ Create a graph
- ✓ Clear the graph
- ✓ Determine weather a graph is empty
- ✓ Traverse/search the graph
- ✓ Print the graph

Graph Traversal Algorithm (1)

- **Graph Traversal:** The process of visiting (search) each vertex in a graph. The most common graph traversal algorithms are:
  - ✓ Depth-First Search (DFS)
  - ✓ Breadth-Frist Search (BFS)

Graph Traversal Algorithm (2)

- ✓ **Depth-First Search (DFS):** The DFS is similar to the preorder traversal of a binary tree.

  The steps of DFS of a undirected graph as follows:

  Step 1: Start by putting any one of the graph's vertices on top of a stack.

  Step 2: Take the top item of the stack and add it to the visited list.

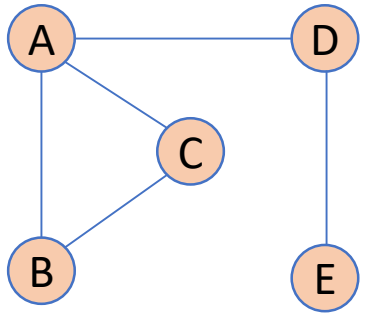
  Step 3: Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.

  Step 4: Keep repeating steps 2 and 3 until the stack is empty.

Graph Traversal Algorithm (3)

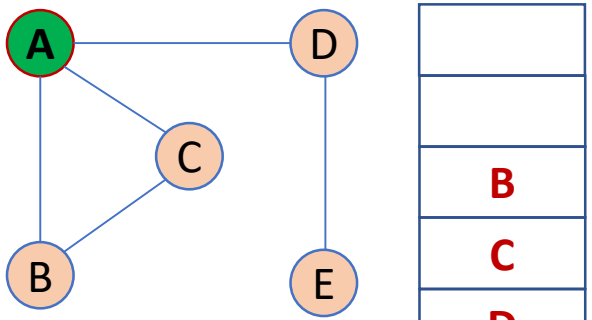
✓ **Depth-First Search (DFS)**

Example: Consider the following undirected graph to perform the DFS.



Step 1:

- Select a vertex **A** as starting point (visit)
- Put the visited vertex **A** in the visited list
- Push all the adjacent vertices of **A** on the stack

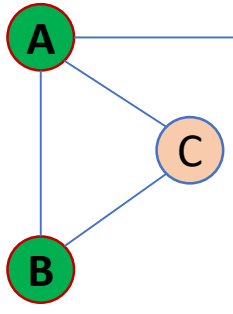


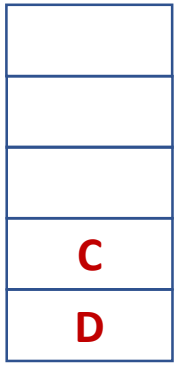
Stack

Visited List

Step 2:

- Visit the adjacent vertex of **A** on the top of the stack
- Put the visited vertex **B** in the visited list
- Remove the **B** from the stack





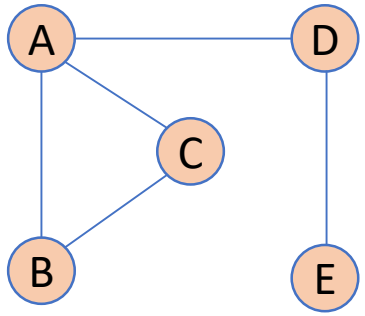
Stack

Visited List

Graph Traversal Algorithm (4)

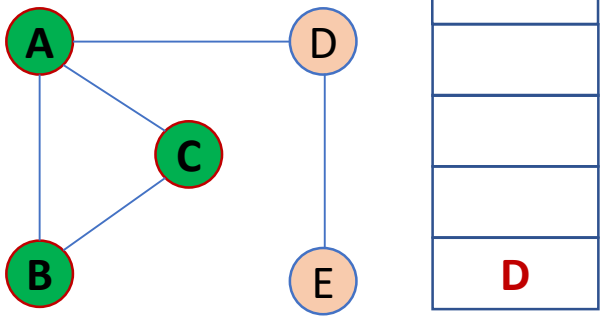
✓ Depth-First Search (DFS)

Example: Consider the following undirected graph to perform the DFS.



Step 3:

- Visit the adjacent vertex **C** of **A** on the top of the stack
- Put the **C** in the visited list
- Remove the vertex **C** from the stack

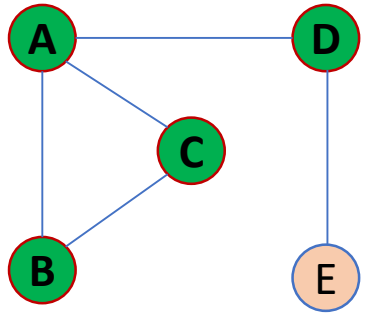


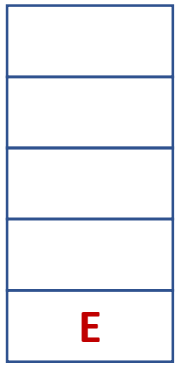
Stack

Visited List

Step 4:

- Visit the adjacent vertex **D** of **A** on the top of the stack
- Put the visited vertex **D** in the visited list
- Remove the vertex **D** from the stack
- Push the adjacent vertices of **D** on the stack





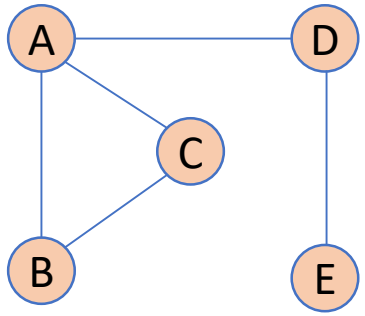
Stack

Visited List

Graph Traversal Algorithm (5)

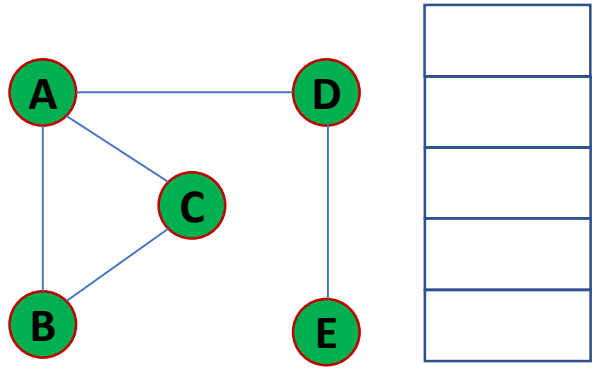
✓ **Depth-First Search (DFS)**

Example: Consider the following undirected graph to perform the DFS.



Step 5:

- Visit the adjacent vertex **E** of **D** on the top of the stack
- Put the visited vertex **E** in the visited list
- Remove the vertex **E** from the stack
- The stack is empty, the DFS is completed



Stack

A	B	C	D	E

Visited List

Graph Traversal Algorithm (6)

- ✓ **Depth-First Search (DFS)**

  The steps of DFS of a directed graph as follows:

  Step 1: Start by putting any one of the graph's vertices on top of a stack.

  Step 2: Push all the outgoing vertices of the visited vertex on the stack.

  Step 3: Take the top item of the stack and add it to the visited list.

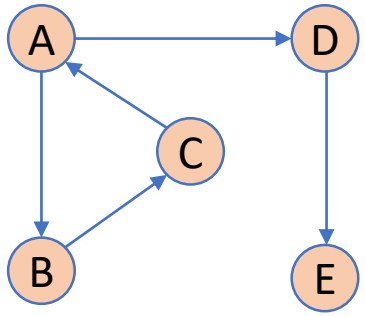
  Step 4: Create a list of that vertex's adjacent vertices. Add the ones which aren't in the visited list to the top of the stack.

  Step 5: Keep repeating steps 3 and 4 until the stack is empty.

Graph Traversal Algorithm (7)

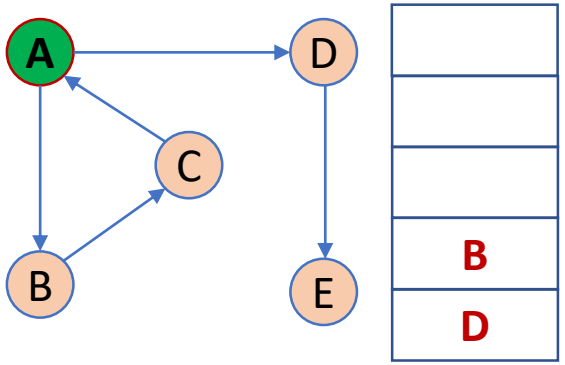
✓ Depth-First Search (DFS)

Example: Consider the following directed graph to perform the DFS.



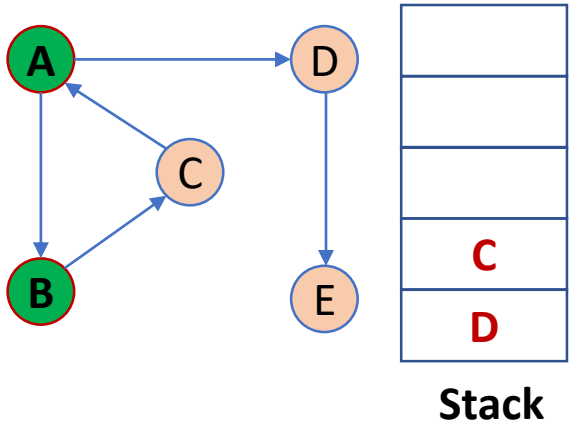
Step 1:

- Select a vertex **A** as starting point (visit)
- Put the visited vertex **A** in the visited list
- Push all the outgoing adjacent vertices of **A** on the stack



Step 2:

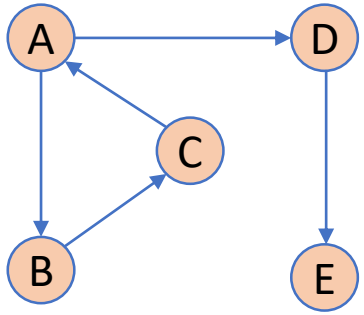
- Visit the outgoing adjacent vertex of **A** on the top of the stack
- Put the visited vertex **B** in the visited list
- Remove the **B** from the stack
- Put the outgoing adjacent vertices (**C**) of **B** on the stack



Graph Traversal Algorithm (8)

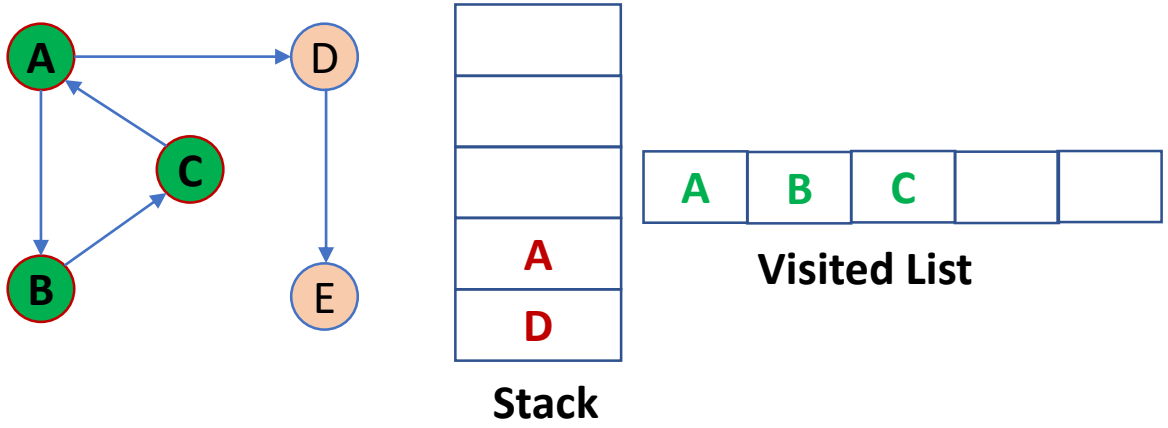
✓ Depth-First Search (DFS)

Example: Consider the following directed graph to perform the DFS.



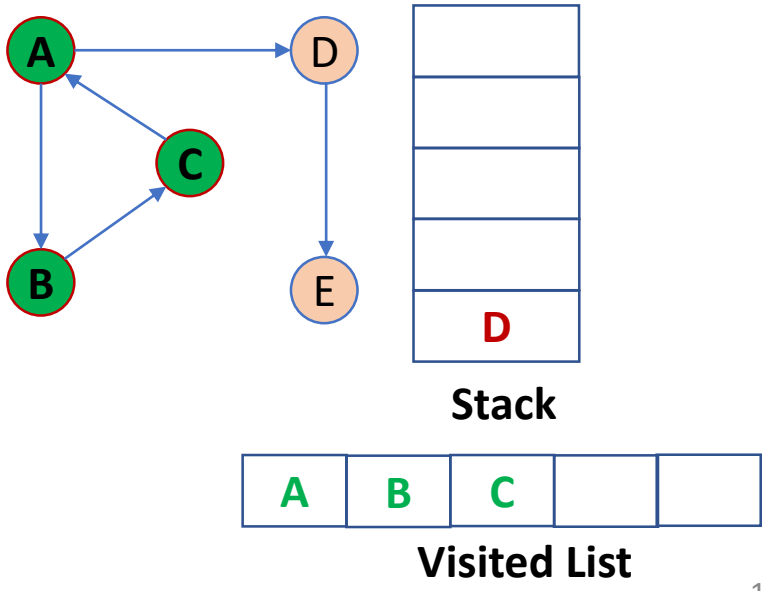
Step 3:

- Visit the adjacent vertex **C** of **B** on the top of the stack
- Put the **C** in the visited list
- Remove the vertex **C** from the stack
- push the outgoing adjacent vertices (**A**) of **C** on the stack



Step 4:

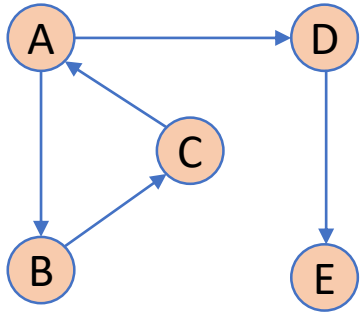
- Visit the adjacent vertex **A** on the top of the stack, **A** is already visited
- Remove the vertex **A** from the stack
- Push the outgoing adjacent vertices of **A** on the stack



Graph Traversal Algorithm (9)

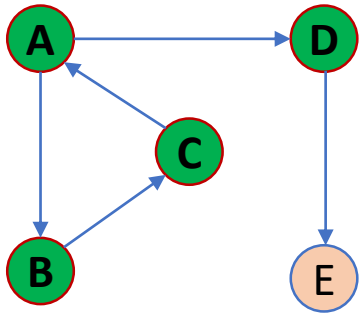
✓ Depth-First Search (DFS)

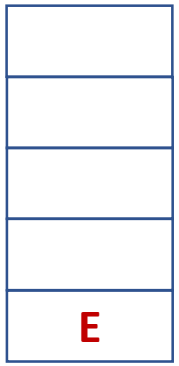
Example: Consider the following directed graph to perform the DFS.



Step 5:

- Visit the adjacent vertex **D** of **A** on the top of the stack, which is not visited
- Put the **D** in the visited list
- Remove the vertex **D** from the stack
- push the outgoing adjacent vertices (**E**) of **D** on the stack



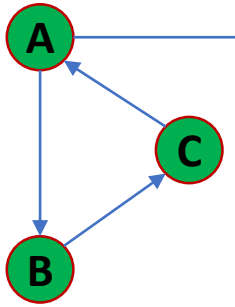


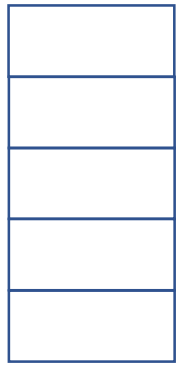
Stack

Visited List

Step 6:

- Visit the outgoing adjacent vertex **E** of **D** on the top of the stack
- Remove the vertex **E** from the stack
- The stack is empty, the DFS is completed





Stack

Visited List

Graph Traversal Algorithm (10)

- ✓ **Breadth-First Search (BFS):** The BFS in graph is similar to the BFS traversal of a binary tree.

The steps of BFS of a undirected graph as follows:

Step 1: Start by putting any one of the graph's vertices at the back of a queue.

Step 2: Take the front item of the queue and add it to the visited list.

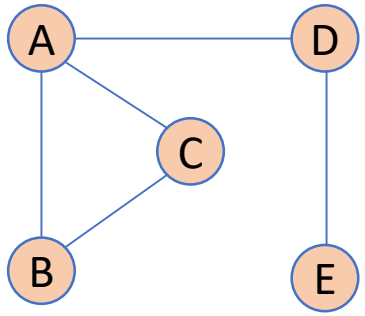
Step 3: Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.

Step 4: Keep repeating steps 2 and 3 until the queue is empty.

Graph Traversal Algorithm (11)

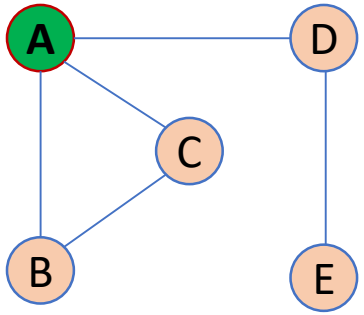
✓ **Breadth-First Search (BFS)**

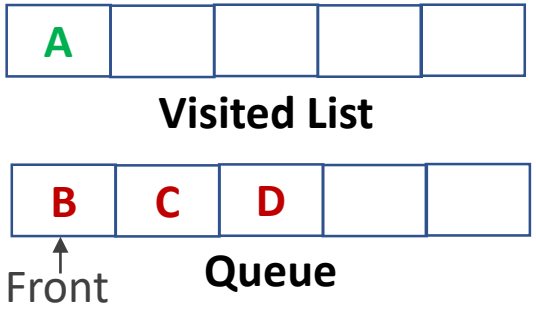
Example: Consider the following undirected graph to perform the BFS.



Step 1:

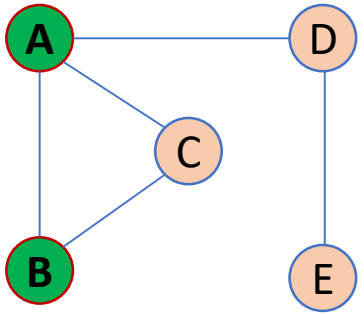
- Select a vertex **A** as starting point (visit)
- Put the visited vertex **A** in the visited list
- Put all the adjacent vertices of **A** in the queue

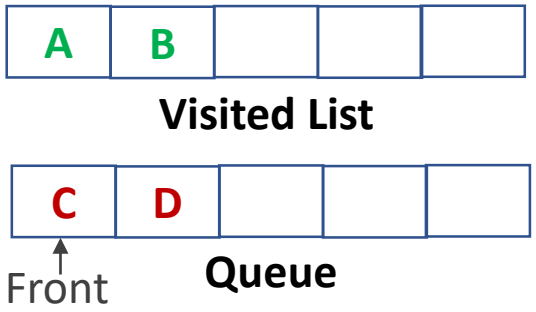




Step 2:

- Visit the adjacent vertex **B** of **A** in the queue
- Put the visited vertex **B** in the visited list
- Remove the vertex **B** from the queue

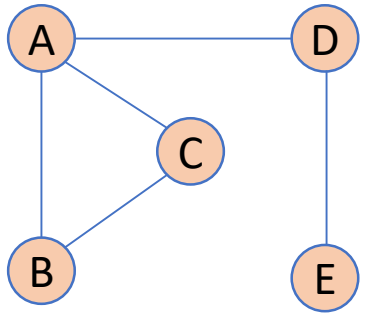




Graph Traversal Algorithm (12)

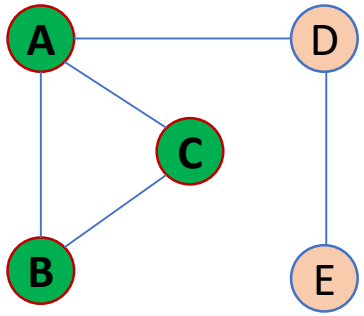
✓ **Breadth-First Search (BFS)**

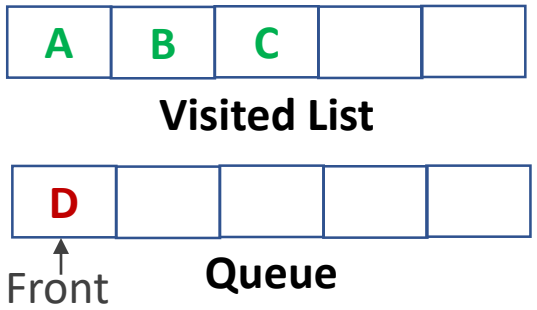
Example: Consider the following undirected graph to perform the BFS.



Step 3:

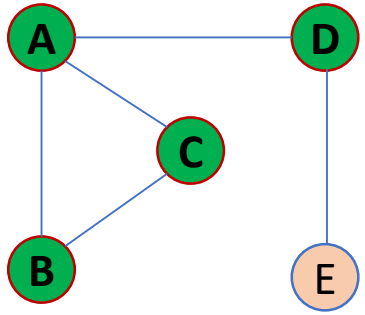
- Visit the adjacent vertex **C** of **B** in the queue
- Put the visited vertex **C** in the visited list
- Remove the vertex **C** from the queue

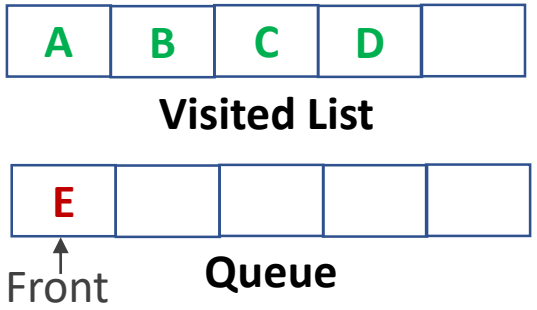




Step 4:

- Visit the adjacent vertex **D** of **A** in the queue
- Put the visited vertex **D** in the visited list
- Remove the vertex **D** from the queue
- Put the adjacent vertices (**E**) of **D** in the queue

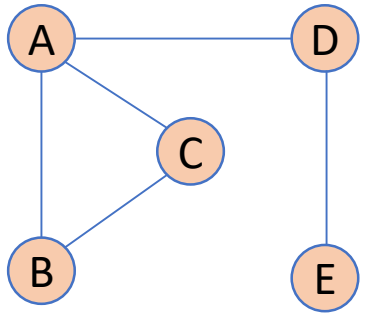




Graph Traversal Algorithm (13)

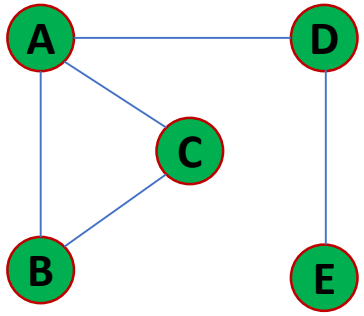
✓ **Breadth-First Search (BFS)**

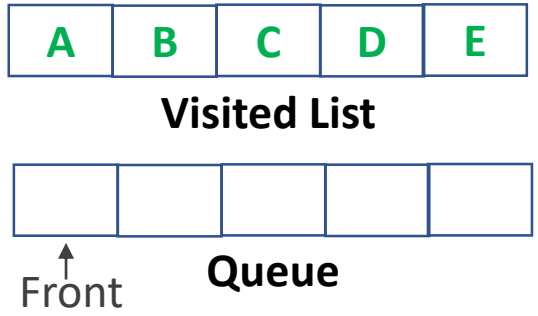
Example: Consider the following undirected graph to perform the BFS.



Step 5:

- Visit the adjacent vertex **E** of **D** in the queue
- Put the visited vertex **E** in the visited list
- Remove the vertex **E** from the queue
- The queue is empty, the traversal is completed





Graph Traversal Algorithm (14)

✓ **Breadth-First Search (BFS)**

The steps of BFS of a directed graph as follows:

Step 1: Start by putting any one of the graph's vertices at the back of a queue.

Step 2: Push all the outgoing vertices of the visited vertex on the queue.

Step 3: Take the front item of the queue and add it to the visited list.

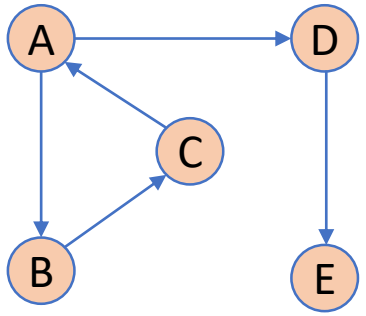
Step 4: Create a list of that vertex's adjacent vertices. Add the ones which aren't in the visited list to the back of the queue.

Step 5: Keep repeating steps 3 and 4 until the queue is empty.

Graph Traversal Algorithm (15)

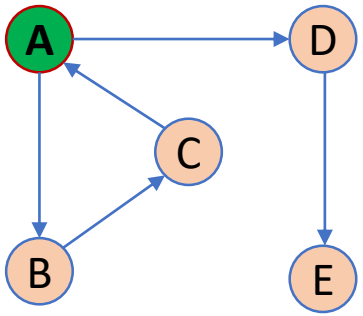
✓ Breadth-First Search (BFS)

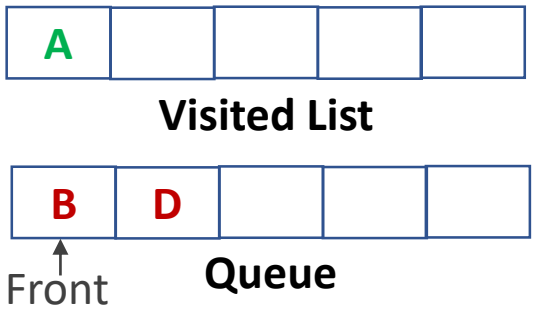
Example: Consider the following directed graph to perform the BFS.



Step 1:

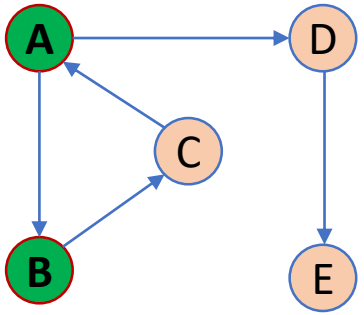
- Select a vertex **A** as starting point (visit)
- Put the visited vertex **A** in the visited list
- Put all the outgoing adjacent vertices of **A** in the queue

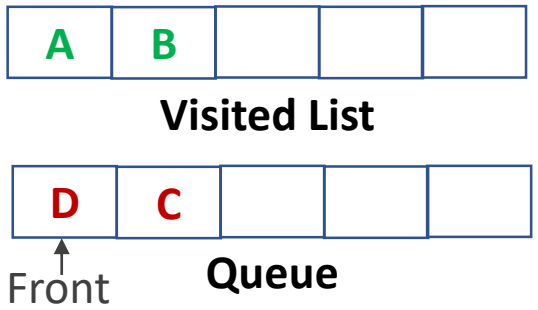




Step 2:

- Visit the outgoing adjacent vertex **B** of **A** in the queue
- Put the visited vertex **B** in the visited list
- Remove the vertex **B** from the queue
- Put the outgoing vertices (**C**) of **B** in the queue

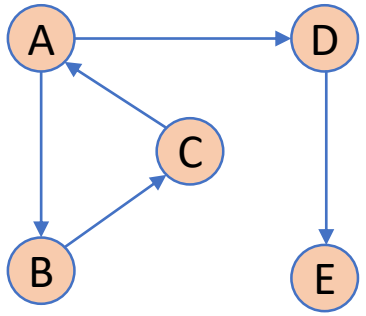




Graph Traversal Algorithm (16)

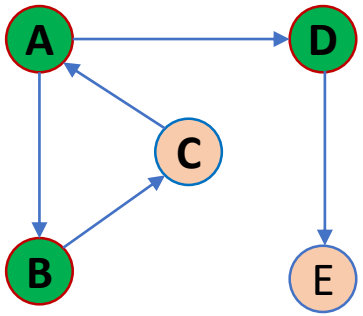
✓ Breadth-First Search (BFS)

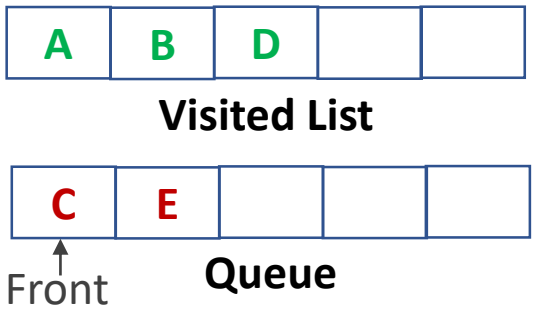
Example: Consider the following directed graph to perform the BFS.



Step 3:

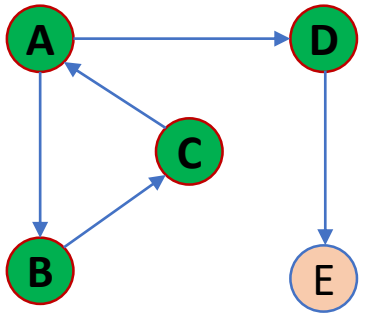
- Visit the adjacent vertex **D** of **A** in the queue
- Put the visited vertex **D** in the visited list
- Remove the vertex **D** from the queue
- Put the outgoing vertices (**E**) of **D** in the queue

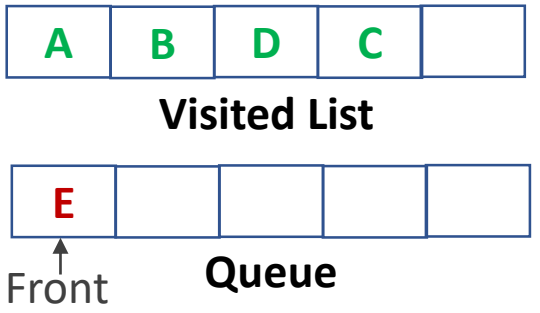




Step 4:

- Visit the adjacent vertex **C** of **B** in the queue
- Put the visited vertex **C** in the visited list
- Remove the vertex **C** from the queue
- Put the outgoing vertices (**A**) of **C** in the queue

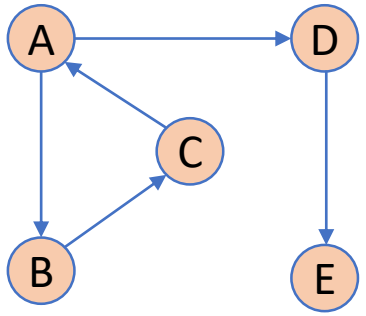




Graph Traversal Algorithm (17)

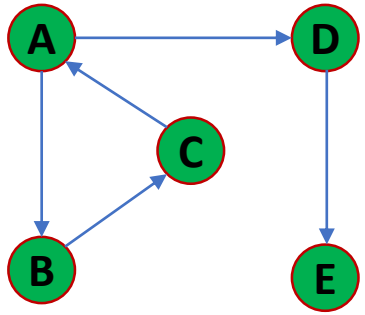
✓ Breadth-First Search (BFS)

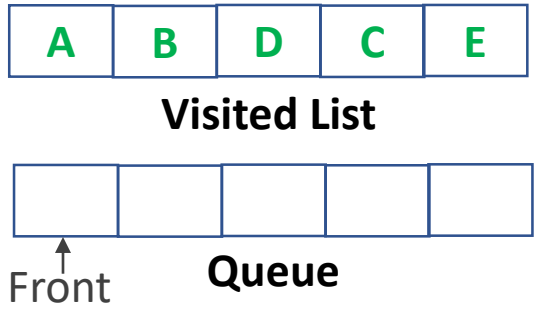
Example: Consider the following directed graph to perform the BFS.



Step 5:

- Visit the adjacent vertex **E** of **D** in the queue
- Put the visited vertex **E** in the visited list
- Remove the vertex **E** from the queue
- The queue is empty, the BSF is completed





Lecture Review

❑ This Lecture Discussed About:

✓ Graph Data Structure

- Terminologies of Graph

✓ Graph Representation

- Adjacency Matrix

- Adjacency List

✓ Graph Traversal Algorithms

- Depth-Frist Search (DFS) for Undirected and Directed graphs

- Breadth-First Search (BFS) for Undirected and Directed graphs