



DATA STRUCTURES AND ITS APPLICATIONS

Graph Traversal Methods

Sandesh B. J

Department of Computer Science & Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Graph Traversals Methods

Sandesh B. J

Department of Computer Science & Engineering

- Why we need to traverse the graph?
- Traversing the graph using different techniques
 - ✓ Depth First Search and Breadth First Search Traversal
- Algorithms for Depth First and Breadth First Search Traversal

- **Graph traversal**(also known as graph search) refers to the process of visiting/investigating (checking or updating) each vertex of the graph in some systematic order -Wiki
- Traversal can start in any arbitrary vertex
- Traversals are classified by the order in which the vertices are visited

Methods to Traverse the Graph

- Depth First Search
- Breadth First Search

DATA STRUCTURES AND ITS APPLICATIONS

Depth first search (DFS)

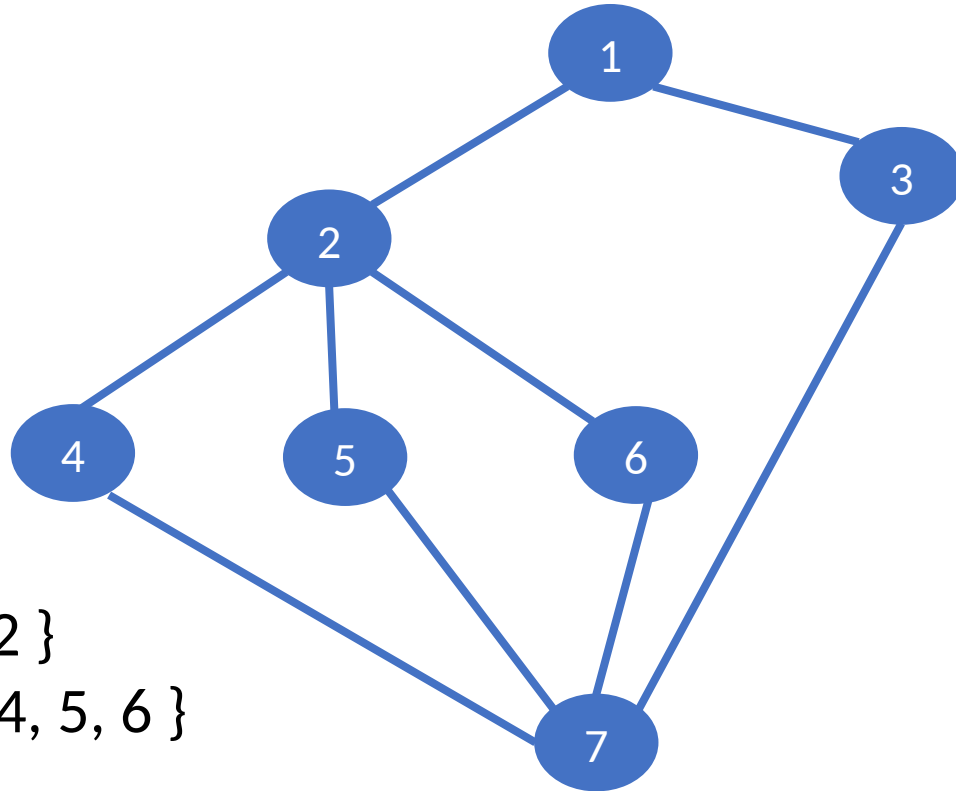


- Visits all the nodes related to one neighbour before visiting the other neighbours and its related nodes. Once it reaches dead end, it backtracks along previously visited node and continue traversing from there.
- Analogues to pre-order traversal of an ordered tree
- Uses stack behaviour, hence implemented using recursive algorithm

DATA STRUCTURES AND ITS APPLICATIONS

Depth First Search Traversal

- DFS traverse the depth of the graph, until it cannot go any further at which point it backtracks and continues



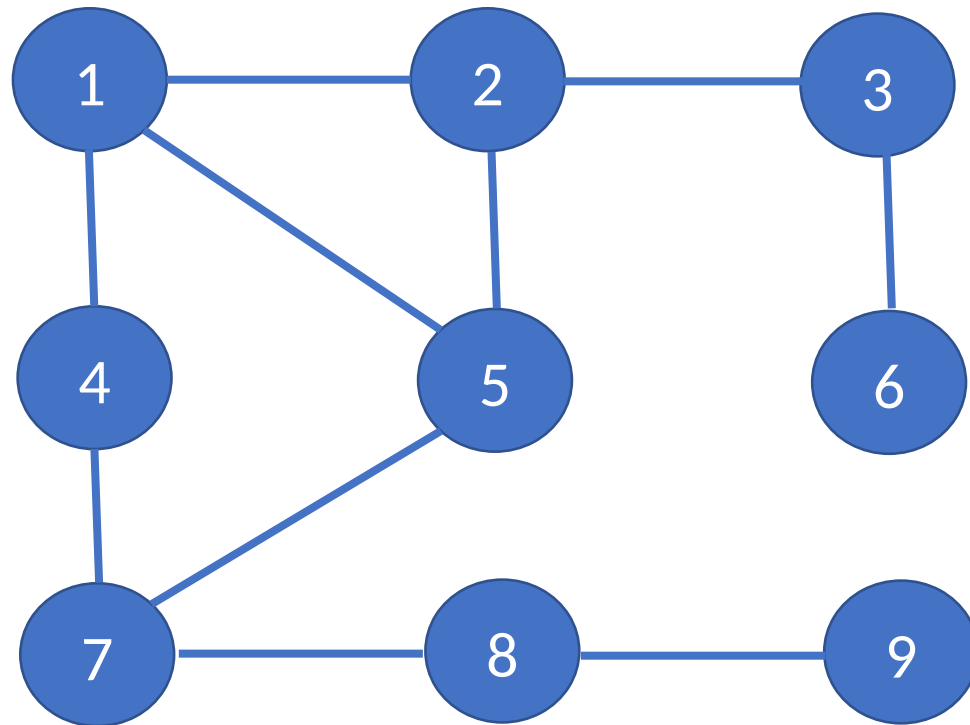
$V = \{ 2 \}$

$W = \{ 4, 5, 6 \}$

Traversal order : 1 , 2 , 4 , 7 , 3 , 5 , 6

DATA STRUCTURES AND ITS APPLICATIONS

Depth First Search Traversal



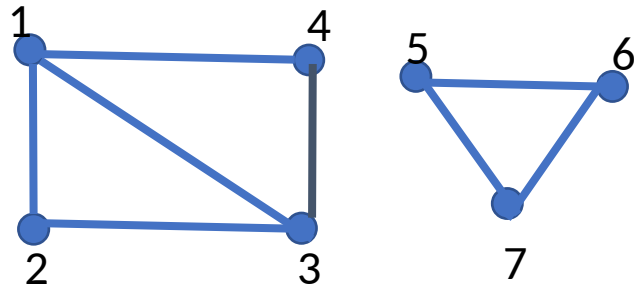
1 2 3 6 5 7 4 8 9

- **Graph may contain cycles**

- Traversal algorithm may reach the same vertex second time.
- To prevent the infinite recursion , we introduce Boolean valued array **visited**
- Set the **visited[v]** to **true** once node v is visited
- Check the **visited[w]** before processing any node w

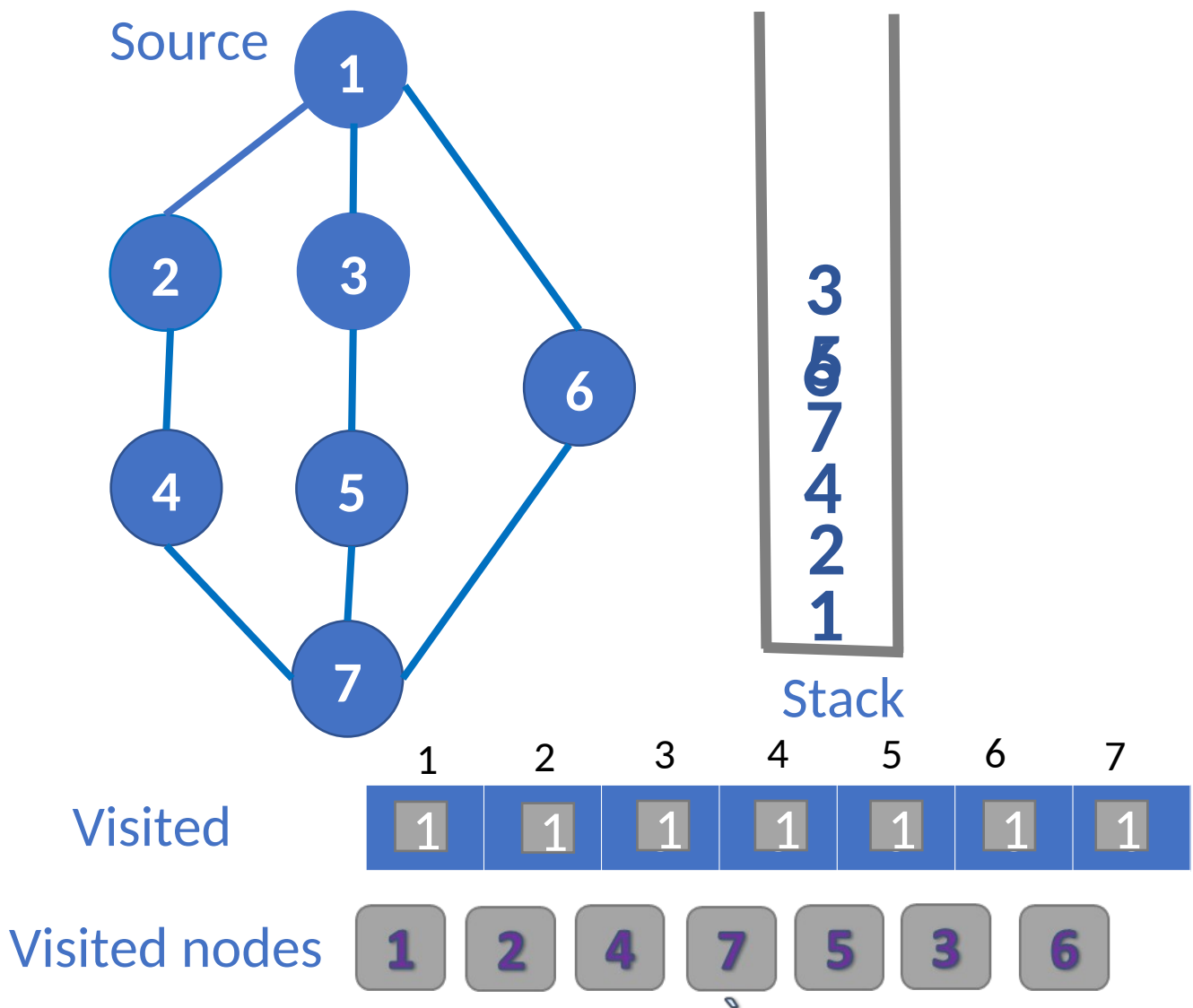
- **Graph may not be connected:**

- Traversal algorithm may fail to reach all the nodes from a single starting point



DATA STRUCTURES AND ITS APPLICATIONS

DFS Traversal – Using Stack

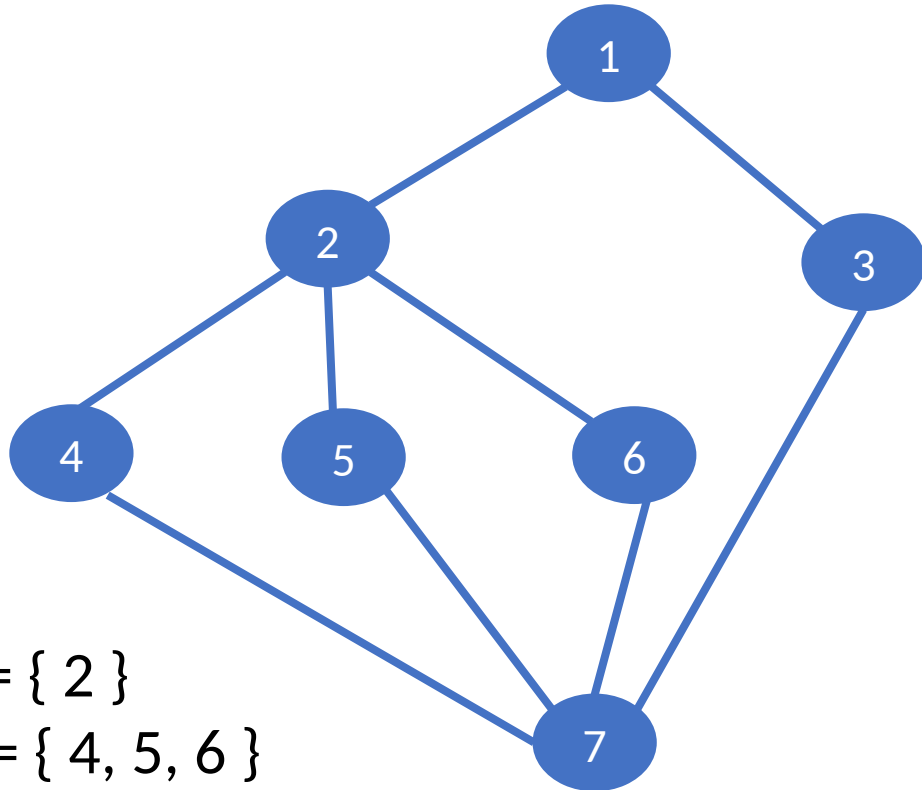


DATA STRUCTURES AND ITS APPLICATIONS

Breadth first search (BFS)

- Explores all the neighbour nodes in first level from an arbitrary node, before moving to next level of neighbouring nodes.
- Uses Queue behaviour

- Analogous to level-by-level traversal of ordered tree



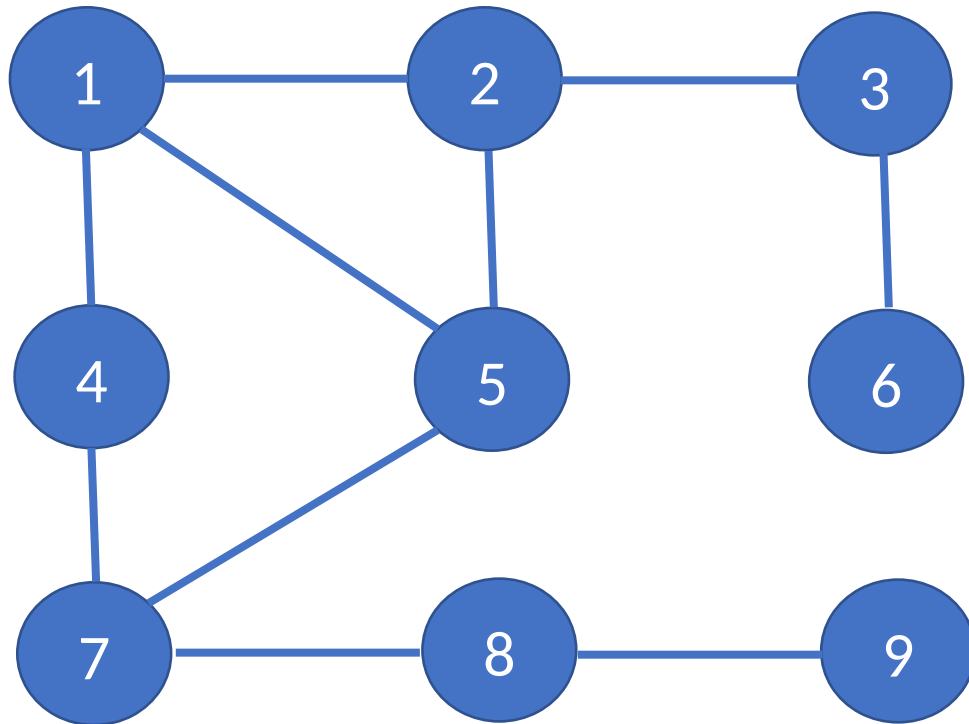
$V = \{ 2 \}$

$W = \{ 4, 5, 6 \}$

Traversal order : 1 , 2, 3, 4, 5, 6 , 7

DATA STRUCTURES AND ITS APPLICATIONS

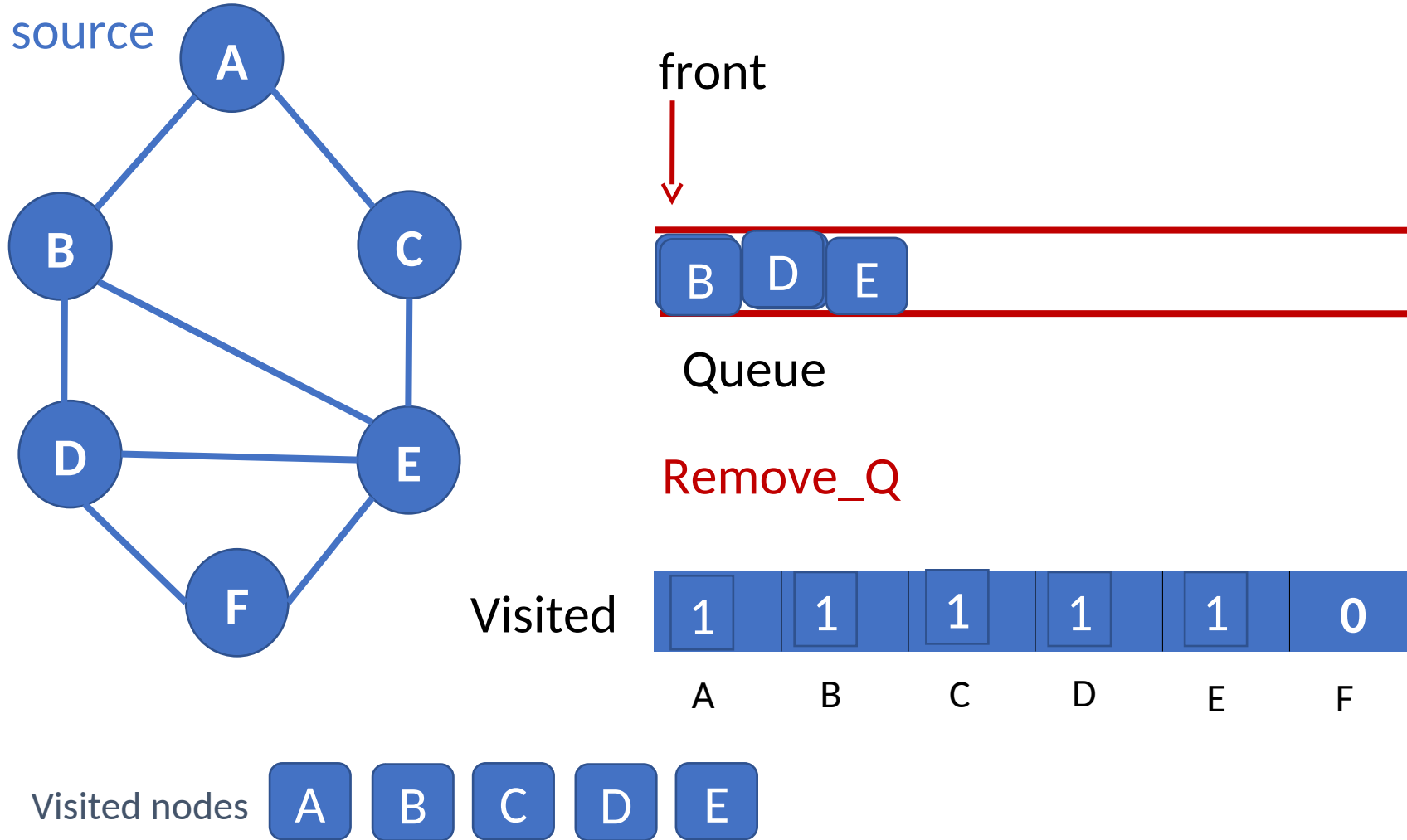
Breadth first search Traversal



1 2 4 5 3 7 6 8 9

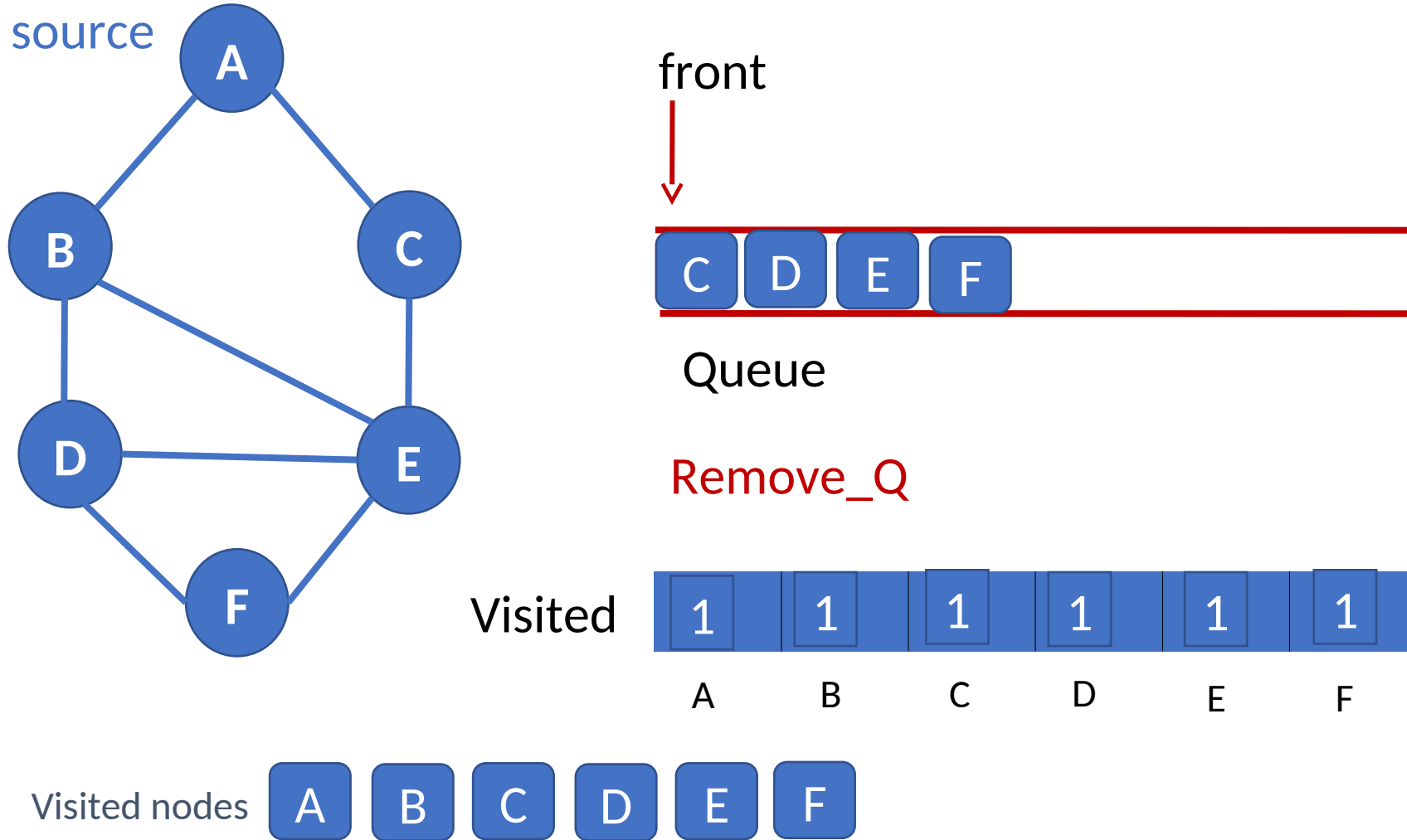
DATA STRUCTURES AND ITS APPLICATIONS

BFS Traversal - Using Queue



DATA STRUCTURES AND ITS APPLICATIONS

BFS - Traversal - Using Queue



DATA STRUCTURES AND ITS APPLICATIONS

DFS - Algorithm



```
Algorithm DFS (Graph G )  
//Implements DFS traversal for given graph  
// Input Graph G = (V, E)  
//Output Graph G with vertices marked as visited  
mark each vertex in V with 0 as a mark of being “unvisited”  
  for each vertex v in V do  
    if v is marked with 0  
      dfs(v)
```

```
dfs(v)  
// visits recursively all the unvisited vertices connected to  
  vertex v by a path  
For each vertex w in V adjacent to v do  
  if w is marked with 0  
    mark w as visited  
    dfs(w)
```

Courtesy: “Introduction to Design and Analysis of Algorithms” By Anany Levitin

```
Algorithm BFS (Graph G )  
//Implements BFS traversal for given graph  
// Input Graph G = (V, E)  
//Output Graph G with vertices marked as visited  
mark each vertex in V with 0 as a mark of being “unvisited”  
  for each vertex v in V do  
    if v is marked with 0  
      bfs(v)
```



```
bfs(v)
// visits recursively all the unvisited vertices connected to
// vertex v by a path
While the queue is not empty
  for each vertex w in V adjacent to front vertex do
    if w is marked with 0
      mark w as visited
      add w to queue
  remove the front vertex from the queue
```



THANK YOU

Sandesh B. J

Department of Computer Science & Engineering

sandesh_bj@pes.edu

+91 80 6618 6623