Elementary Graph Algorithms

NIKITA UIKEY 181210033

Presentation Outline

MAIN TOPICS

- What is a Graph?
- Real-life Applications
- Running Times
- Representations of Graphs
- Graph traversals

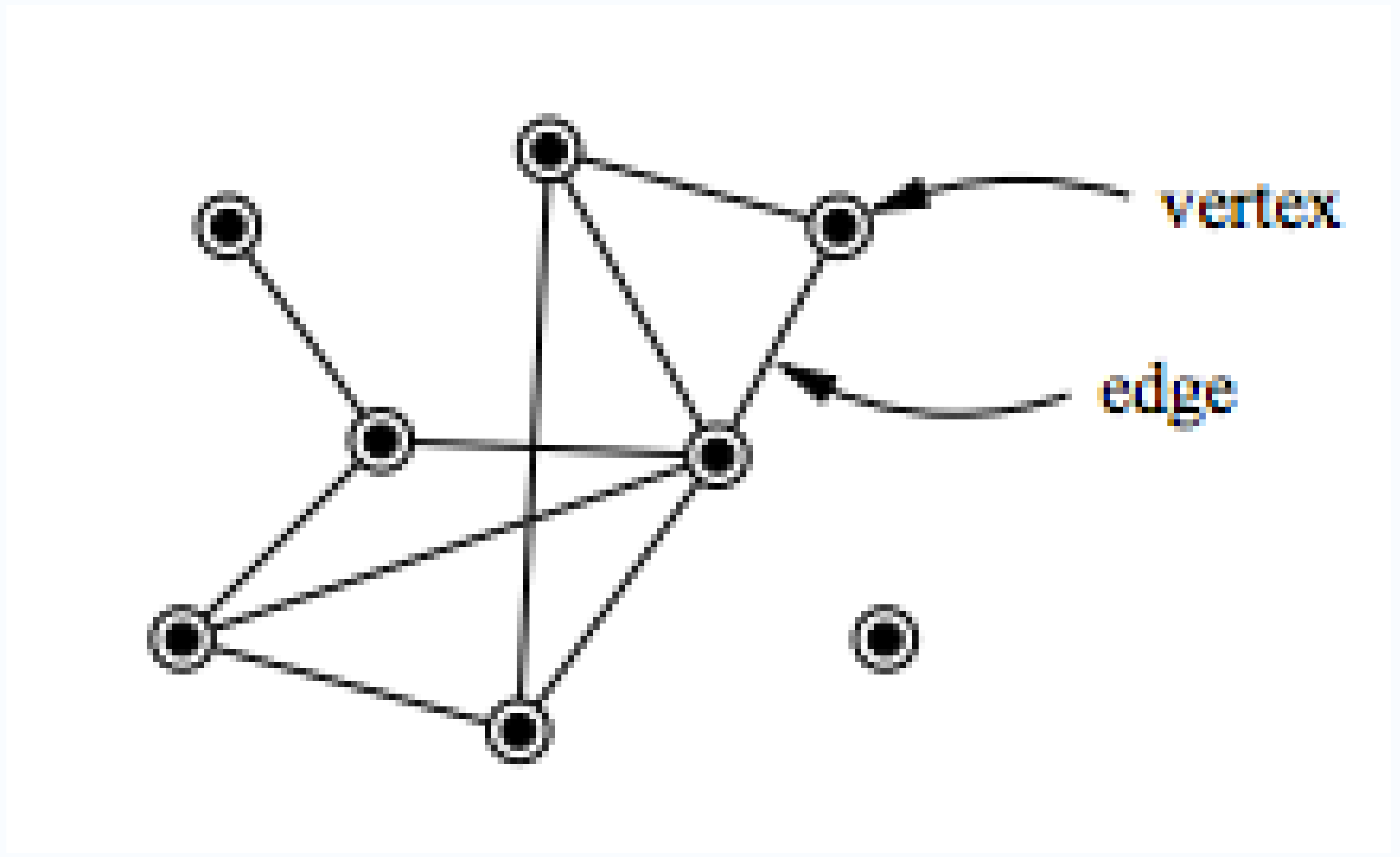
1. BREADTH FIRST SEARCH (BFS)
2. DEPTH FIRST SEARCH (DFS)

What is a Graph?

What is a Graph? What is a Graph?

A GRAPH G = (V, E) IS A NON-LINEAR DATA STRUCTURE CONSISTING OF :

- V(G): A finite set of vertices also called as nodes.
- E(G): A finite set of ordered pair of the form (u, v) called as edge



Real-life Applications Real-life Applications

Real-life Applications

- Graphs are used to represent networks.

  The networks may include paths in a city or telephone network or circuit network.
- Graphs are also used in social networks like linkedIn, Facebook. For example, in Facebook, each person is represented with a vertex(or node).

Running Times

- The running time of graph algorithms is in terms of both vertices |V| and edges |E|.
- We can remove cardinality when in asymptotic notation.
- Example :

  O(VE) , which basically means

  O (|V||E|)

Representations of Graphs

FOLLOWING TWO ARE THE MOST COMMONLY USED REPRESENTATIONS OF A GRAPH:

1. ADJACENCY MATRIX REPRESENTATION
2. ADJACENCY LISTS REPRESENTATION

Types of Graphs

1. DIRECTED
2. UNDIRECTED

Representations of Graphs

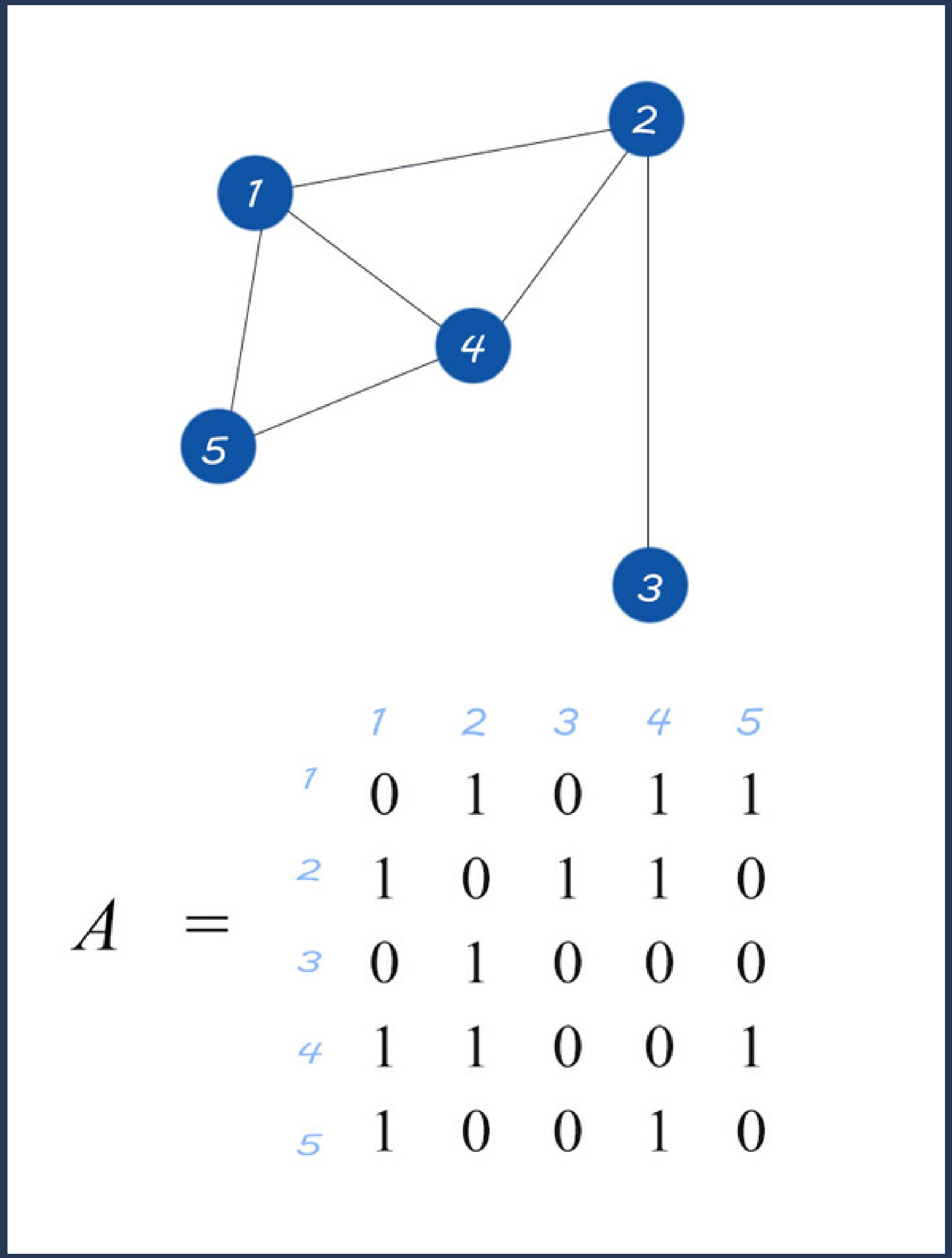
Adjacency matrix

- Adjacency Matrix is a 2D array of size |V| x |V|matrix A= (aij), such that

  aij = 1 , if (i,j) ∈ E,

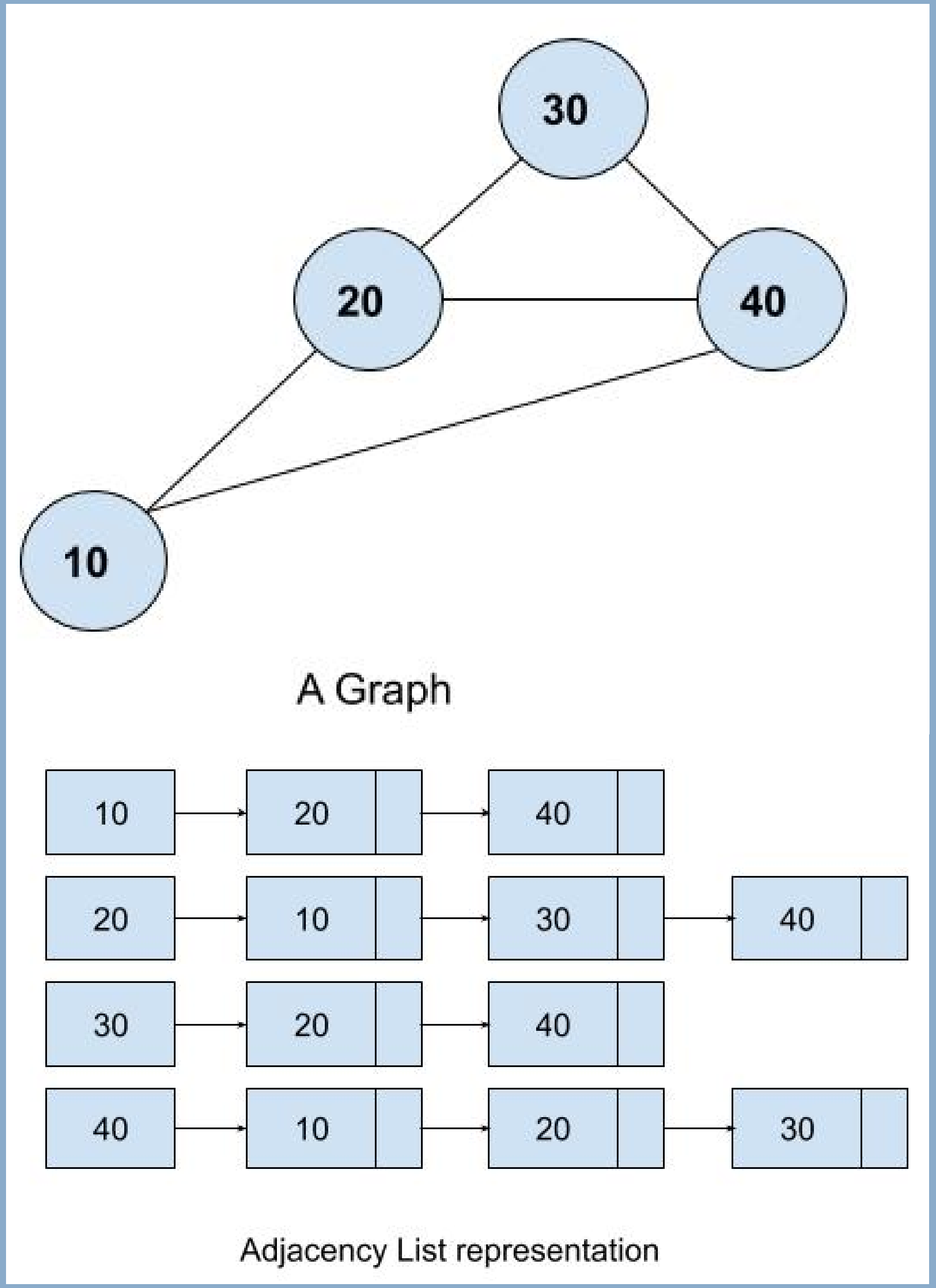
  0 , otherwise

  where V is the number of vertices in a graph.
- Adjacency Matrix is also used to represent weighted graphs.



$$A = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 & 1 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 4 & 1 & 1 & 0 & 0 & 1 \\ 5 & 1 & 0 & 0 & 1 & 0 \end{array}$$

Representations of Graphs



Adjacency lists

- An array of lists is used. Size of the array is equal to the number of vertices.
- The weights of edges can be represented as lists of pairs.

GRAPH TRAVERSALS

- Graph traversal means visiting every vertex and edge exactly once in a well-defined order. While using certain graph algorithms, you must ensure that each vertex of the graph is visited exactly once.
- The order in which the vertices are visited are important and may depend upon the algorithm or question that you are solving.
- During a traversal, it is important that you track which vertices have been visited. The most common way of tracking vertices is to mark them.

BREADTH FIRST SEARCH (BFS)

BREADTH-FIRST SEARCH IS A GRAPH TRAVERSAL ALGORITHM WHICH TRAVERSE A GRAPH OR TREE LEVEL BY LEVEL

- An algorithm for searching all the vertices of a graph, works on both directed and undirected graphs.
- The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.
- Input : graph G=(V,E) and a distinguished source vertex ‘s’ ∈ V.
- Output :

i) d[v] ,distance (smallest no. of edges) from s to every reachable vertex v ∈ V.

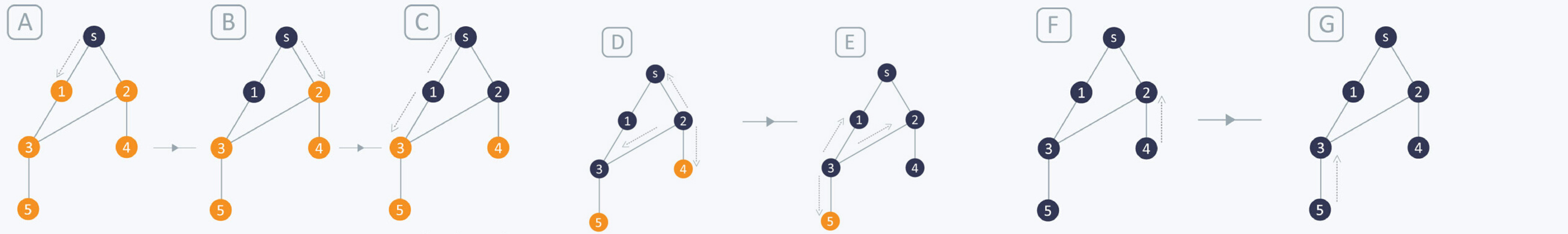
ii) π[v] = ‘u’ such that (u,v) is last edge on shortest path s->v.

- ‘u’ is the predecessor or parent of v in BFS tree.

Pseudocode

```
BFS (G, s)                    //Where G is the graph and s is the source node
    let Q be queue.
    Q.enqueue( s )  //Inserting s in queue until all its neighbour vertices are marked.
    mark s as visited.
    while ( Q is not empty)
        //Removing that vertex from queue,whose neighbour will be visited now
        v  =  Q.dequeue( )
       //processing all the neighbours of v
       for all neighbours w of v in Graph G
           if w is not visited
                  Q.enqueue( w )          //Stores w in Q to further visit its neighbour
                  mark w as visited.
```

Example



Here s is already marked, so it will be ignored

Here s and 3 are already marked, so they will be ignored

Here 1 & 2 are alr marked so they w ignored

Here 2 is already marked, so it will be ignored

Here 3 is already marked, so it will be ignored

Time Complexity Analysis

- The operations of enqueuing and dequeuing take O(1) time, and so the total time devoted to queue operations is O(V).
- Because the procedure scans the adjacency list of each vertex only when the vertex is dequeued, it scans each adjacency list at most once. Since the sum of the lengths of all the adjacency lists is O(E). The total time spent in scanning adjacency lists is O(E).
- The overhead for initialization is O(V) and thus the total running time of the BFS procedure is O(V+E).
- Thus, breadth-first search runs in time linear in the size of the adjacency-list representation of G.

DEPTH FIRST SEARCH)

DEPTH FIRST SEARCH (DFS)

DEPTH FIRST SEARCH

THE DFS ALGORITHM IS A RECURSIVE ALGORITHM THAT USES THE IDEA OF BACKTRACKING. IT INVOLVES EXHAUSTIVE SEARCHES OF ALL THE NODES BY GOING AHEAD, IF POSSIBLE, ELSE BY BACKTRACKING.

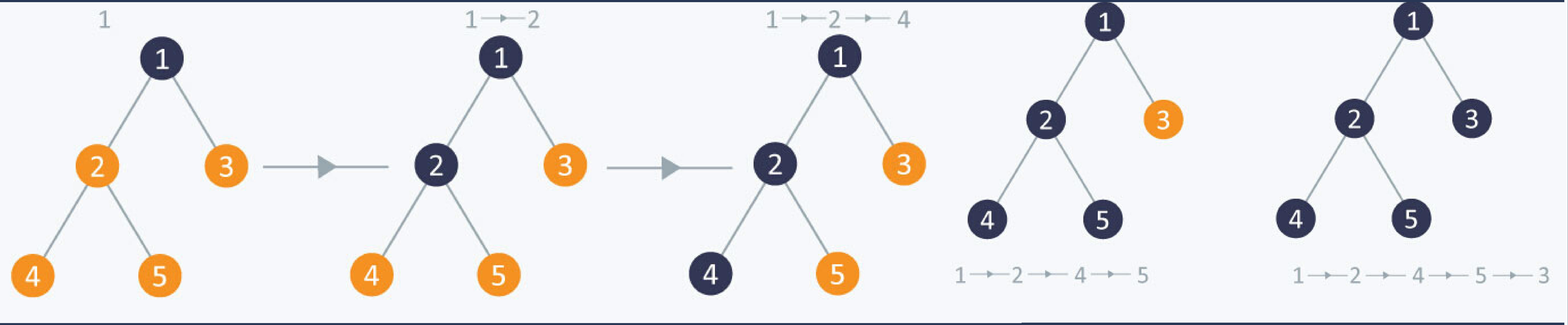
- This strategy works by searching ‘deeper’ in the graph whenever possible, works on both directed and undirected graphs.
- This recursive nature of DFS can be implemented using stacks.

Pseudocode

```
DFS-iterative (G, s):            //Where G is graph and s is source vertex
    let S be stack
    S.push( s )        //Inserting s in stack
    mark s as visited.
    while ( S is not empty):
        //Pop a vertex from stack to visit next
        v  =  S.top( )
        S.pop( )
        //Push all the neighbours of v in stack that are not visited
        for all neighbours w of v in Graph G:
            if w is not visited :
                S.push( w )
                mark w as visited


DFS-recursive(G, s):
    mark s as visited
    for all neighbours w of s in Graph G:
        if w is not visited:
            DFS-recursive(G, w)
```

Example



Time Complexity Analysis

- In DFS,you traverse each node exactly once.
  Therefore, the time complexity of DFS is at least O(V).
- Fora directed graph, the sum of the sizes of the adjacency lists of all the nodes is E (total number of edges).
- So, the complexity of DFS is O(V) + O(E) = O(V + E).