BFS AND DFS

WHAT IS GRAPH TRAVERSALS?

- Methodology for locating the vertex position in the graph.
- It is an advanced search algorithm that can analyze the graph with speed and precision
- It traverse along with marking the sequence of the visited vertices.
- quickly visit each node in a graph without being locked in an infinite loop.

BREADTH FIRST SEARCH (BFS)

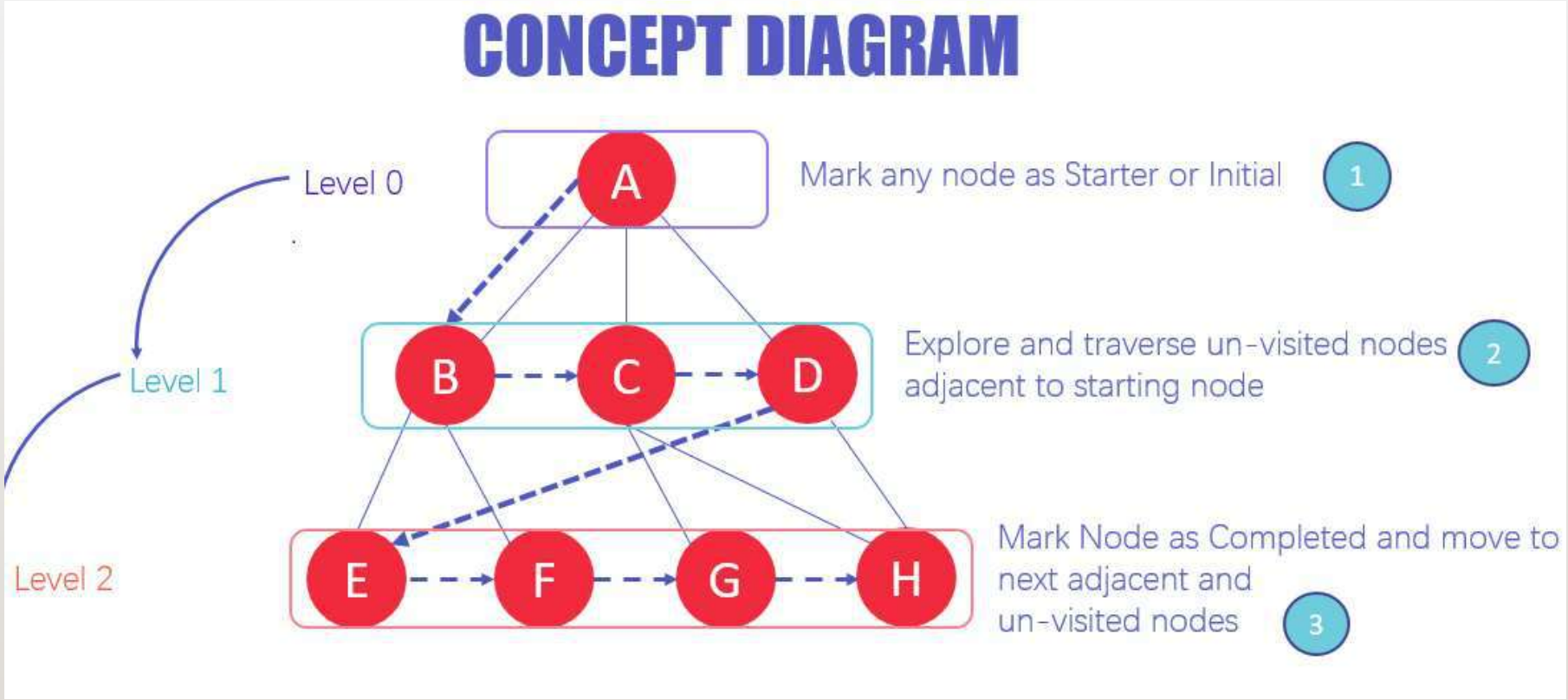
BFS Stands for ***Breadth First Search***. It is also known as **level order traversal**.

- The Queue data structure is used for the Breadth First Search traversal.
- When we use the BFS algorithm for the traversal in a graph, we can consider any node as a root node.

BREADTH FIRST SEARCH (BFS) ALGORITHM

- Used to graph data or searching tree or traversing structures.
- The algorithm efficiently visits and marks all the key nodes in a graph in an accurate **breadthwise fashion**.
- This algorithm selects a single node (initial or source point) in a graph and then visits all the nodes adjacent to the selected node.
- Remember, BFS accesses these nodes one by one.
- Once the algorithm visits and marks the starting node, then it moves towards the nearest unvisited nodes and analyses them.
- Once visited, all nodes are marked.
- These iterations continue until all the nodes of the graph have been successfully visited and marked.

THE ARCHITECTURE OF BFS ALGORITHM



THE ARCHITECTURE OF BFS ALGORITHM

- Various levels of the data
- mark any node as the starting or initial node to begin traversing.
- The BFS will visit the node and mark it as visited and places it in the queue.
- Now the BFS will visit the nearest and un-visited nodes and marks them.
- These values are also added to the queue.
- The queue works on the FIFO model.
- In a similar manner, the remaining nearest and un-visited nodes on the graph are analyzed marked and added to the queue.
- These items are deleted from the queue as receive and printed as the result.

WHY DO WE NEED BFS ALGORITHM?

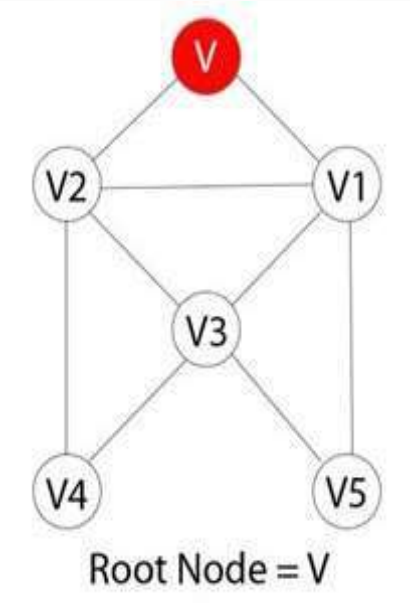
There are numerous reasons to utilize the BFS Algorithm to use as searching for your dataset.

- BFS is useful for analyzing the nodes in a graph and constructing the shortest path of traversing through these.
- BFS can traverse through a graph in the smallest number of iterations.
- The architecture of the BFS algorithm is simple and robust.
- The result of the BFS algorithm holds a high level of accuracy in comparison to other algorithms.
- BFS iterations are seamless, and there is no possibility of this algorithm getting caught up in an infinite loop problem.

HOW DOES BFS ALGORITHM WORK?

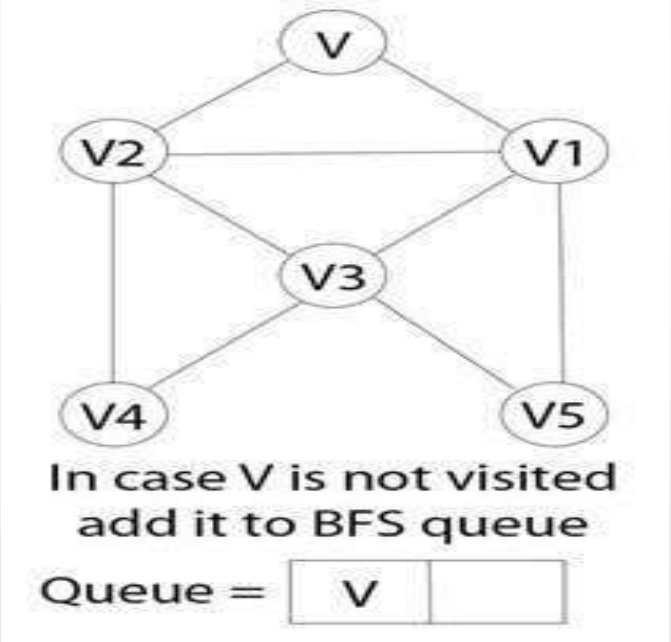
- Graph traversal requires the algorithm to visit, check, and/or update every single un-visited node in a tree-like structure.
- Graph traversals are categorized by the order in which they visit the nodes on the graph.
- BFS algorithm starts the operation from the first or starting node in a graph and traverses it thoroughly.
- Once it successfully traverses the initial node, then the next non-traversed vertex in the graph is visited and marked.
- Hence, you can say that all the nodes adjacent to the current vertex are visited and traversed in the first iteration.

- A simple queue methodology is utilized to implement the working of a BFS algorithm, and it consists of the following steps:
- **Step 1)**



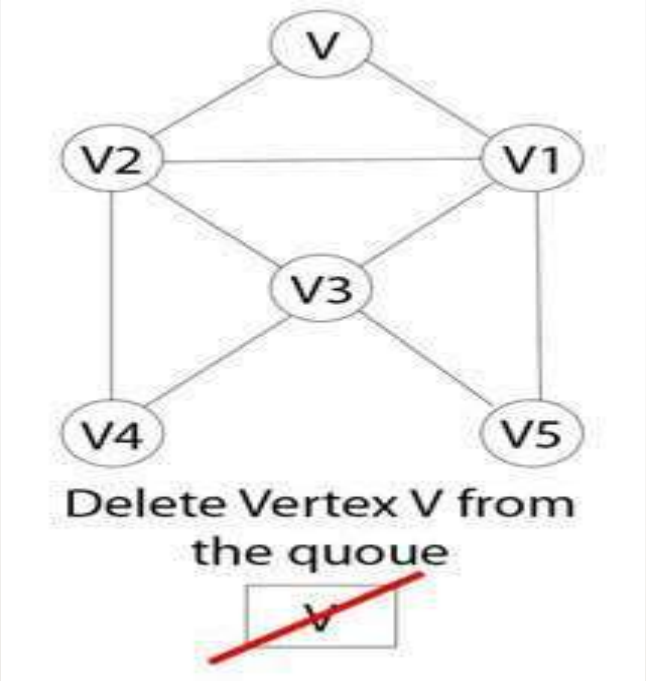
Each vertex or node in the graph is known. For instance, you can mark the node as V.

- **Step 2)**



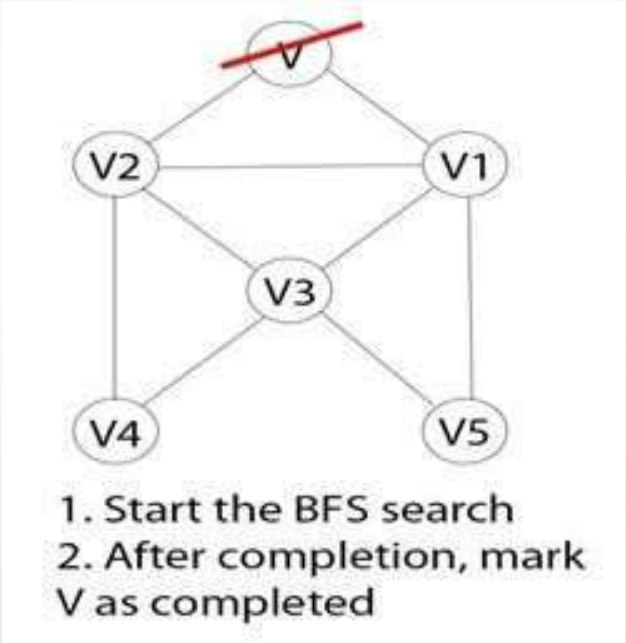
In case the vertex V is not accessed then add the vertex V into the BFS Queue

- **Step 3)**



Start the BFS search, and after completion, Mark vertex V as visited.

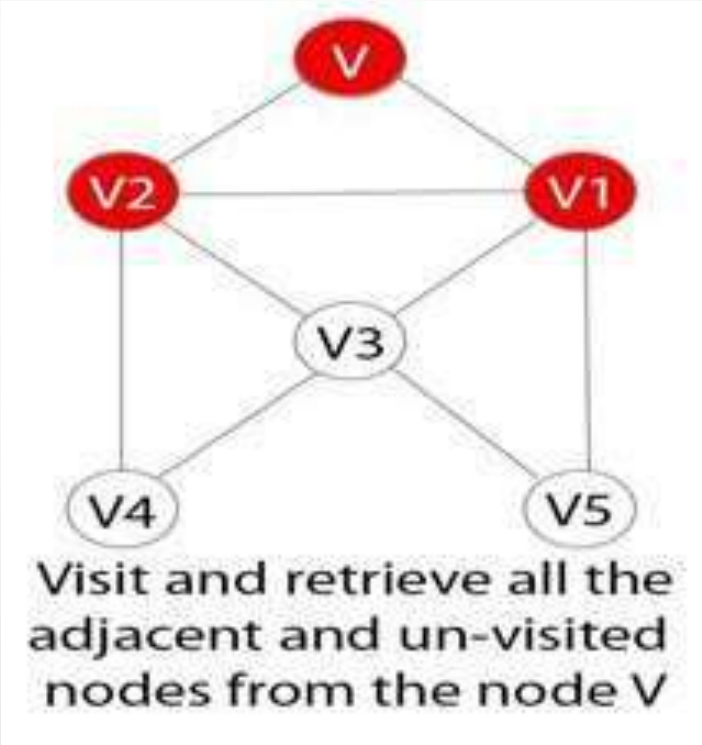
- **Step 4)**



The BFS queue is still not empty, hence remove the vertex V of the graph from the queue.

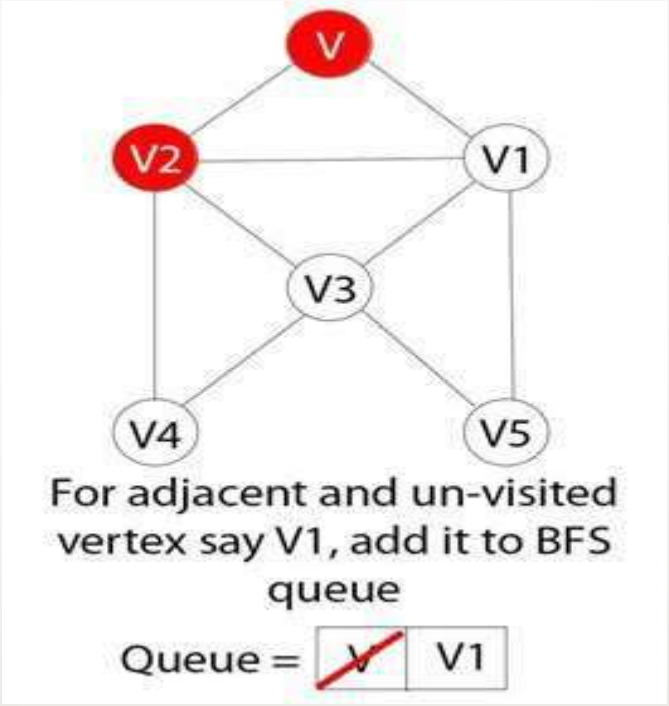
- Retrieve all the remaining vertices on the graph that are adjacent to the vertex V

step.5



Retrieve all the remaining vertices on the graph that are adjacent to the vertex V

- **Step 6)**



For each adjacent vertex let's say V1, in case it is not visited yet then add V1 to the BFS queue

RULES OF BFS ALGORITHM

Here, are important rules for using BFS algorithm:

- A queue (FIFO-First in First Out) data structure is used by BFS.
- You mark any node in the graph as root and start traversing the data from it.
- BFS traverses all the nodes in the graph and keeps dropping them as completed.
- BFS visits an adjacent unvisited node, marks it as done, and inserts it into a queue.
- Removes the previous vertex from the queue in case no adjacent vertex is found.
- BFS algorithm iterates until all the vertices in the graph are successfully traversed and marked as completed.
- There are no loops caused by BFS during the traversing of data from any node.

BFS (PSEUDO CODE)

```
BFS(input: graph G) {
   Queue Q;    Integer x, z, y;
   while (G has an unvisited node x) {
         visit(x); Enqueue(x,Q);
         while (Q is not empty){
               z := Dequeue(Q);
               for all (unvisited neighbor y of z){
               visit(y); Enqueue(y,Q);
               }
         }
   }
}
```

APPLICATIONS OF BFS ALGORITHM

Real-life applications where a BFS algorithm implementation can be highly effective.

- **Un-weighted Graphs:** BFS algorithm can easily create the shortest path and a minimum spanning tree to visit all the vertices of the graph in the shortest time possible with high accuracy.
- **P2P Networks:** BFS can be implemented to locate all the nearest or neighboring nodes in a peer-to-peer network. This will find the required data faster.
- **Web Crawlers:** Search engines or web crawlers can easily build multiple levels of indexes by employing BFS. BFS implementation starts from the source, which is the web page, and then it visits all the links from that source.
- **Navigation Systems:** BFS can help find all the neighboring locations from the main or source location.
- **Network Broadcasting:** A broadcasted packet is guided by the BFS algorithm to find and reach all the nodes it has the address for.

DEPTH FIRST SEARCH (DFS)

- DFS stands for Depth First Search. In DFS traversal, the stack data structure is used, which works on the LIFO (Last In First Out) principle
- Depth first Search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph or tree data structure.
- Traversal means visiting all the nodes of a graph.
- A standard DFS implementation puts each vertex of the graph into one of two categories:

1. Visited
2. Not Visited

DEPTH FIRST SEARCH (DFS)

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

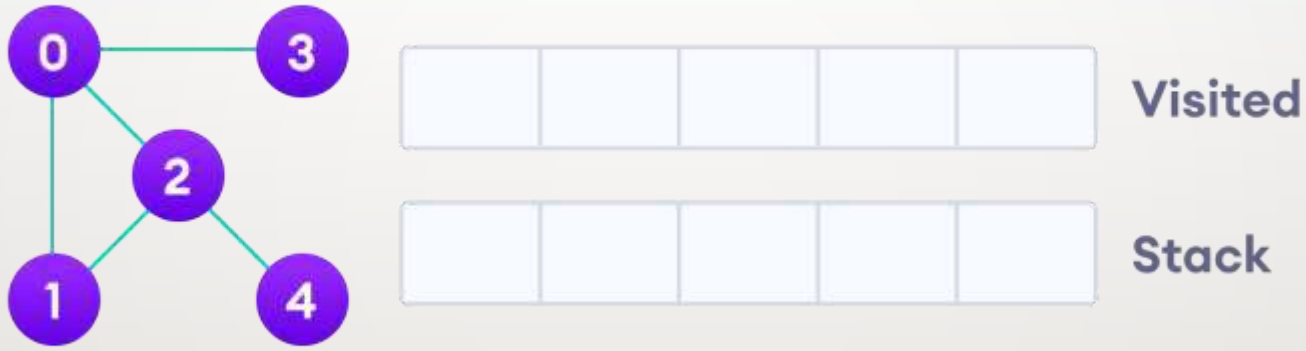
The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
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.
4. Keep repeating steps 2 and 3 until the stack is empty.

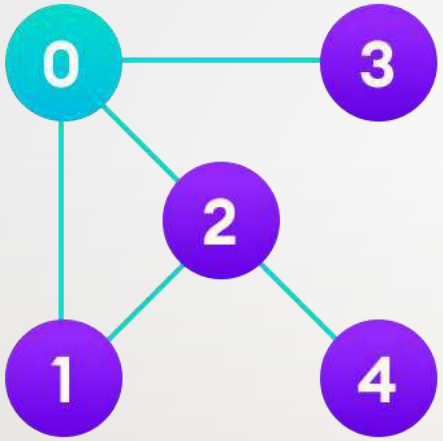
DEPTH FIRST SEARCH EXAMPLE

- Let's see how the Depth First Search algorithm works with an example.

We use an undirected graph with 5 vertices.



We start from vertex 0, the DFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.



0				

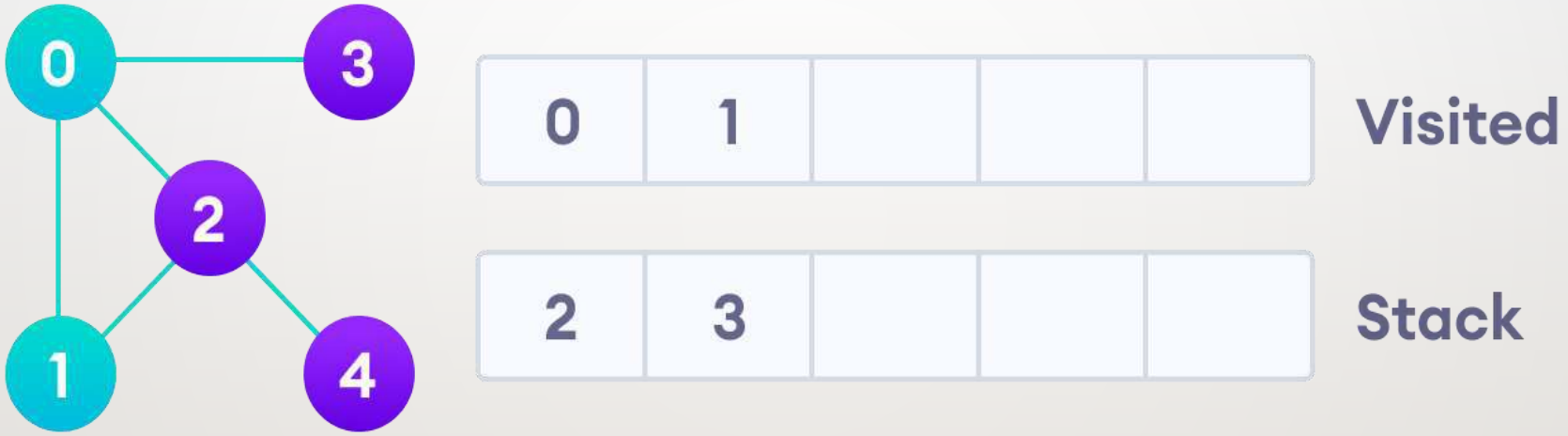
Visited

1	2	3		

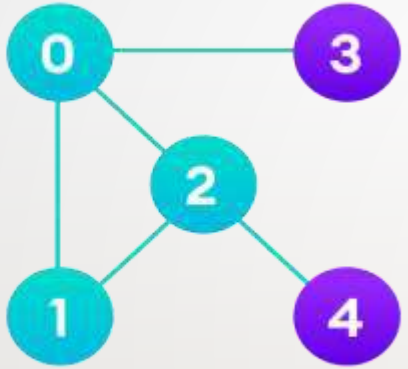
Stack

Next, we visit the element at the top of stack i.e., 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.

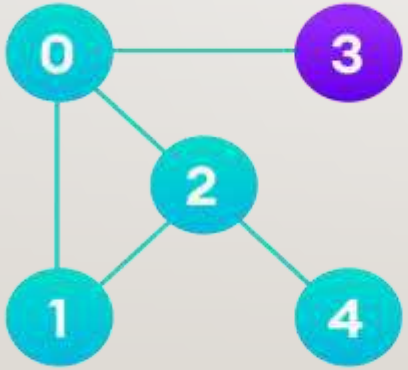
- Next, we visit the element at the top of stack i.e., 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.



- Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

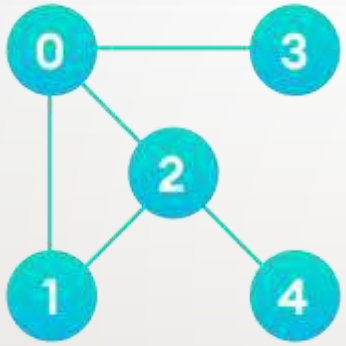


0	1	2			Visited
4	3				Stack



0	1	2	4		Visited
3					Stack

- After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.



0	1	2	4	3	Visited

					Stack

DFS (PSEUDO CODE)

```
DFS(input: Graph G) {
    Stack S; Integer x, t;
    while (G has an unvisited node x){
        visit(x); push(x,S);
        while (S is not empty){
            t := peek(S);
            if (t has an unvisited neighbor y){ visit(y); push(y,S); }
            else
                pop(S);
        }
    }
}
```

Complexity of Depth First Search

The **time complexity** of the DFS algorithm is represented in the form of O(V + E), where V is the number of nodes and E is the number of edges.
The **space complexity** of the algorithm is O(V)

Application of DFS Algorithm

1. For finding the path
2. To test if the graph is bipartite
3. For finding the strongly connected components of a graph
4. For detecting cycles in a graph

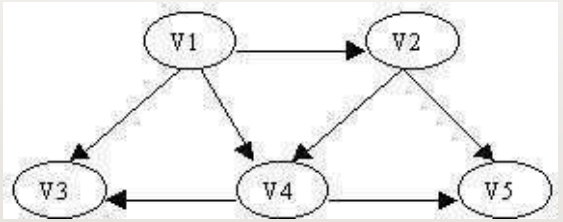
SAMPLE QUESTIONS: BFS&DFS

1)Explain: Acyclic Directed Graph, Articulation Point, Dense Graph, Breadth First Search Traversal, Depth First Search Traversal.?

2) Explain Depth First Traversal Method for Graph with algorithm with example.

3) Explain Breath First Traversal Method for Graph with algorithm with example.

4) What is DFS? Explain with example. Show the ordering of vertices produced by Topological-sort for the following graph



5) Define BFS. How it is differ from DFS.

6) Write an algorithm to topologically sort an digraph using DFS. Prove the correctness and find time efficiency