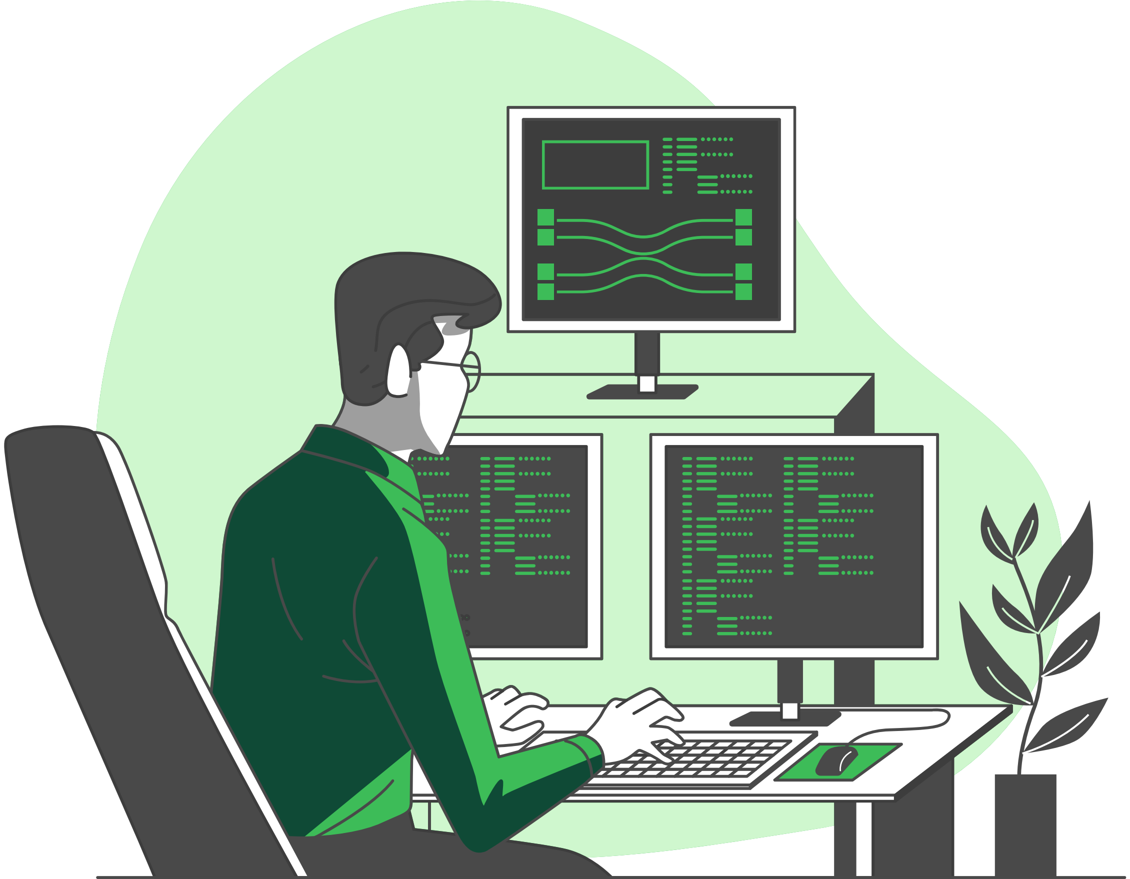
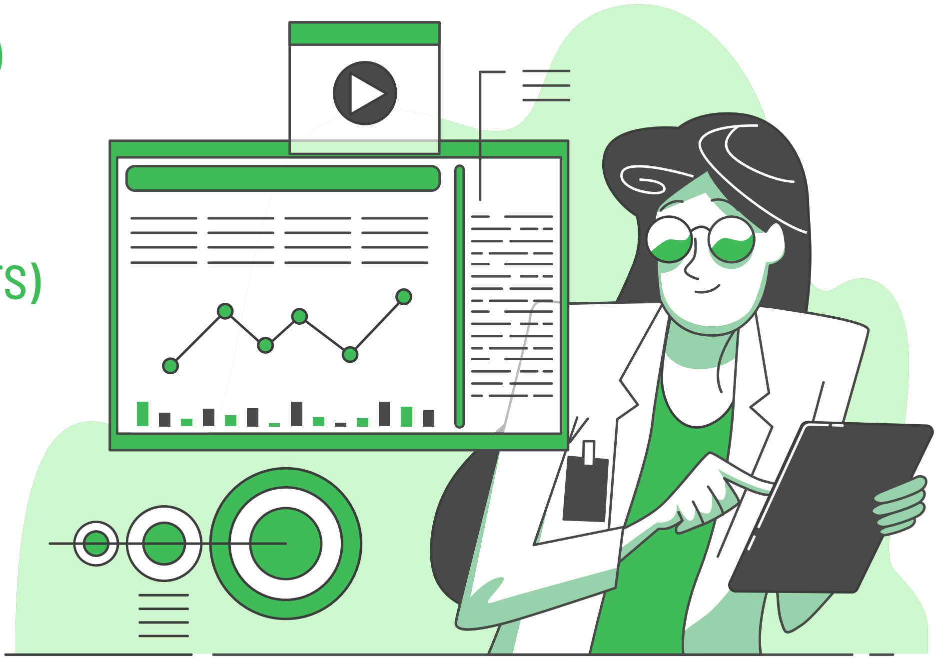


Graph Traversals

CS 251 - Data Structures and Algorithms

Note:
Slides complement the discussion in class

Table of Contents



01

Subgraphs & Trees

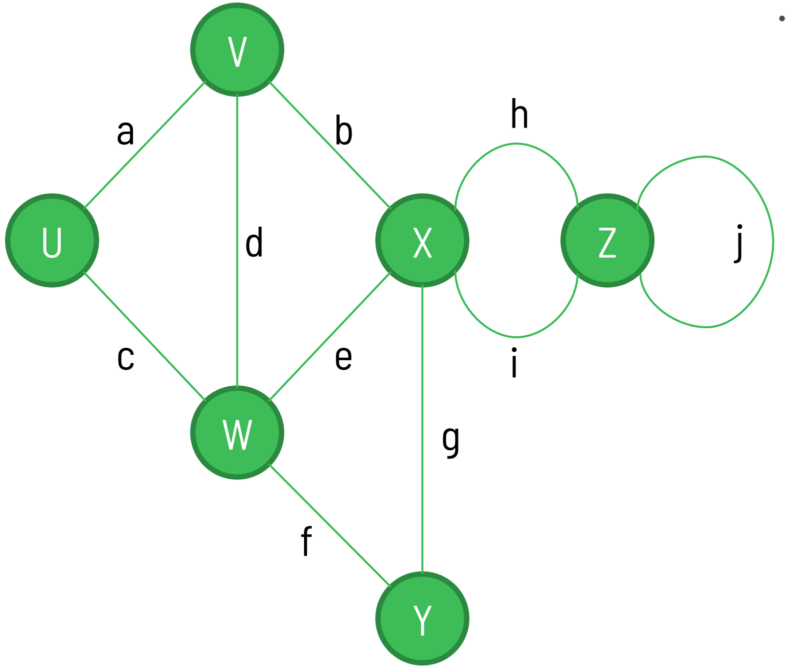
They are also graphs

Revisit Subgraphs

Subgraph (of a graph): a graph made up of a subset of the vertices of G and a subset of the edges of G.

E.g.,

$$S = \{\{\mathrm{U, V, W, X, Y}\}, \{\mathrm{a, b, c, d, e, f, g}\}\}$$

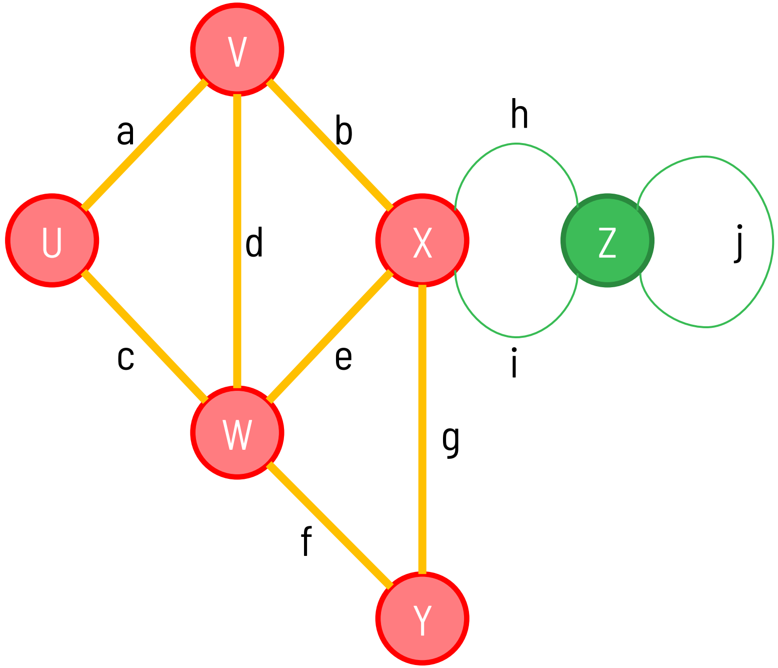


Revisit Subgraphs

Subgraph (of a graph): a graph made up of a subset of the vertices of G and a subset of the edges of G.

E.g.,

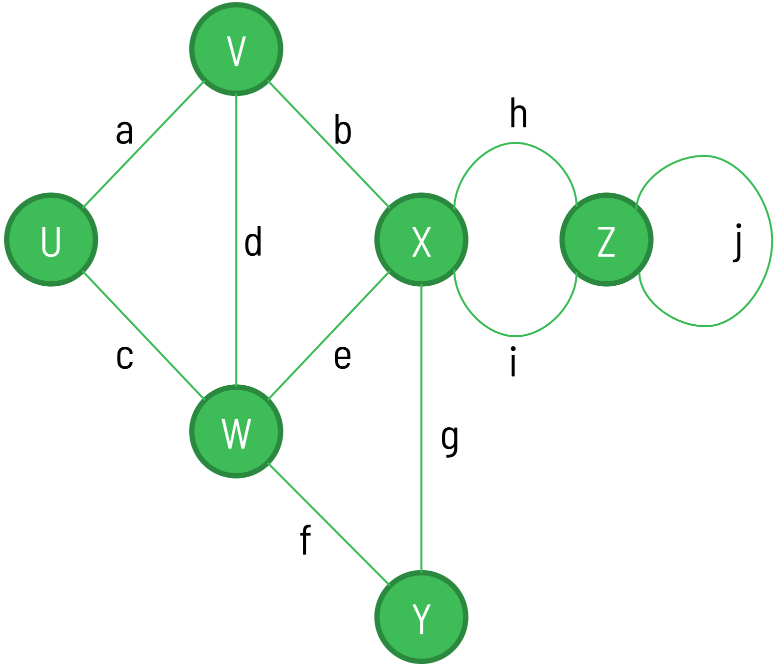
$S = \{\{\mathrm{U, V, W, X, Y}\}, \{\mathrm{a, b, c, d, e, f, g}\}\}$



Revisit Subgraphs

Spanning Subgraph (of a graph): a subgraph that includes all vertices of G.

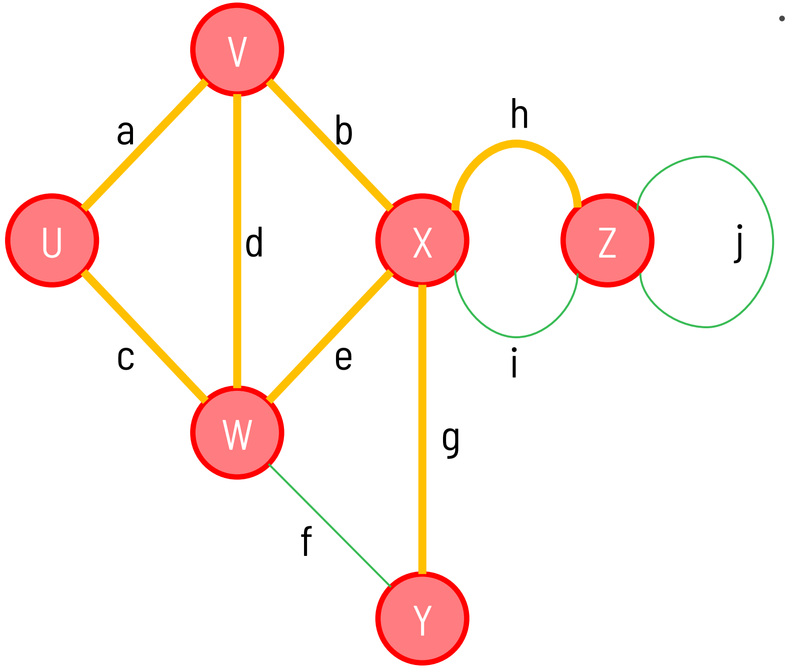
E.g.,
$S = \{\{\mathrm{U, V, W, X, Y, Z}\}, \{\mathrm{a, b, c, d, e, g, h}\}\}$



Revisit Subgraphs

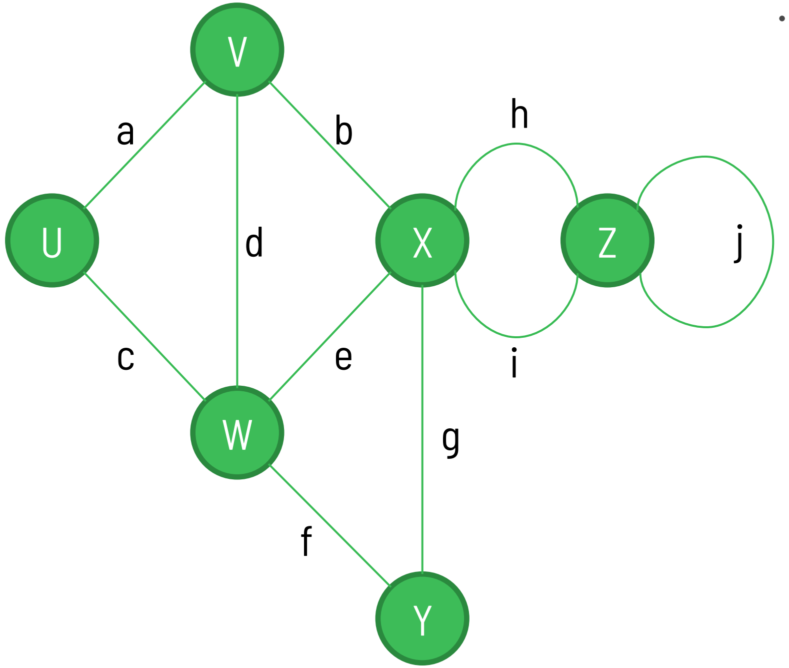
Spanning Subgraph (of a graph): a subgraph that includes all vertices of G.

E.g.,
$S = \{\{\mathrm{U, V, W, X, Y, Z}\}, \{\mathrm{a, b, c, d, e, g, h}\}\}$



Connected Graphs

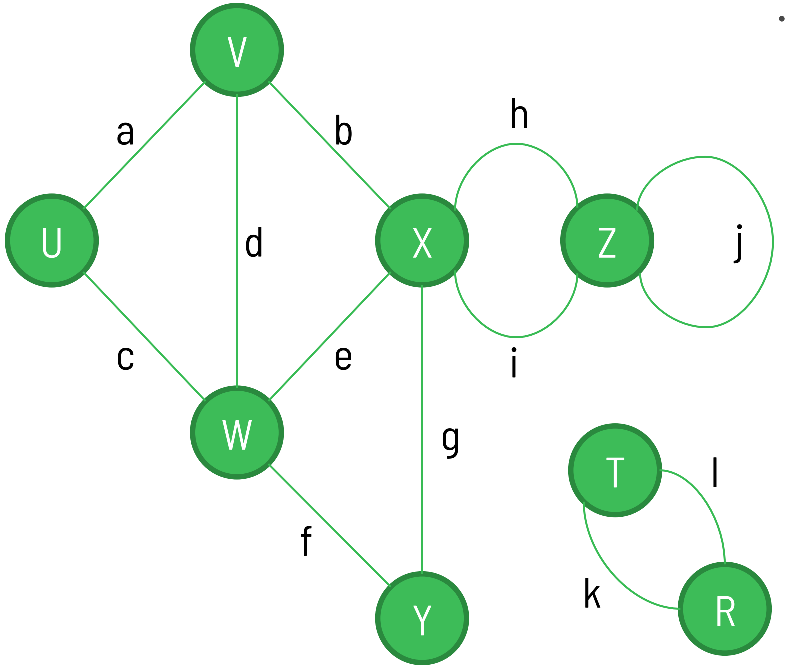
A **connected graph** is a graph where there is a path from every vertex to every other vertex.



Connected Graphs

A **connected graph** is a graph where there is a path from every vertex to every other vertex.

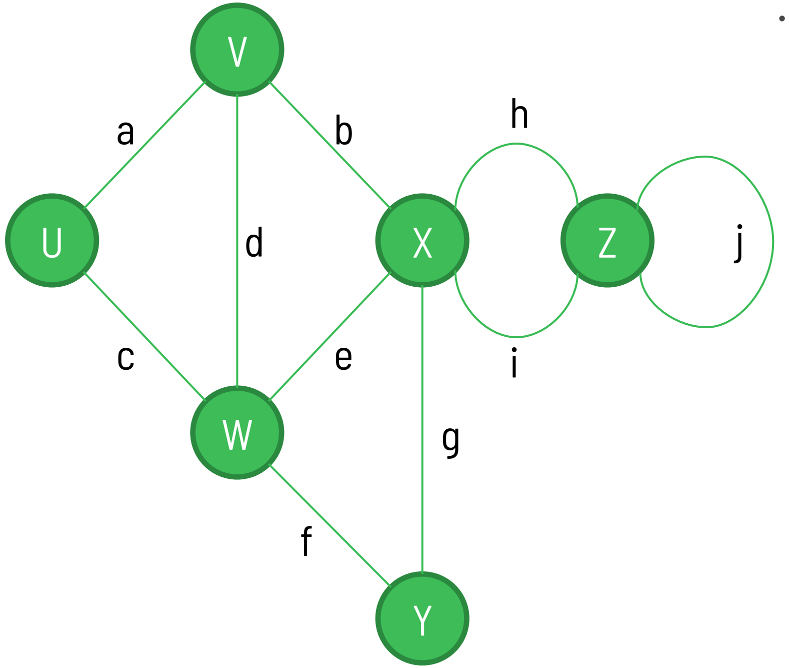
A **connected component** of a graph G is a maximal connected subgraph of G.



Let's Revisit Trees

A **tree** is an undirected graph T such that:

- T is connected
- T has no cycles (acyclic)



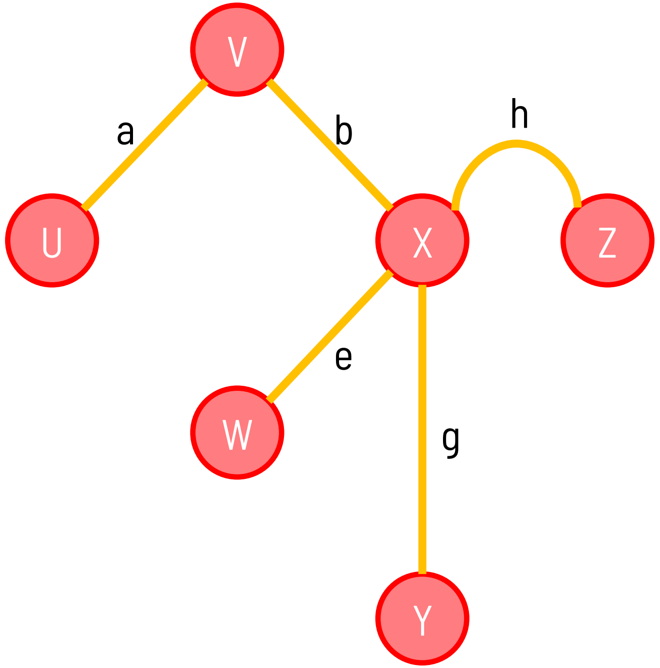
Let's Revisit Trees

A **tree** is an undirected graph T such that:

- T is connected
- T has no cycles (acyclic)

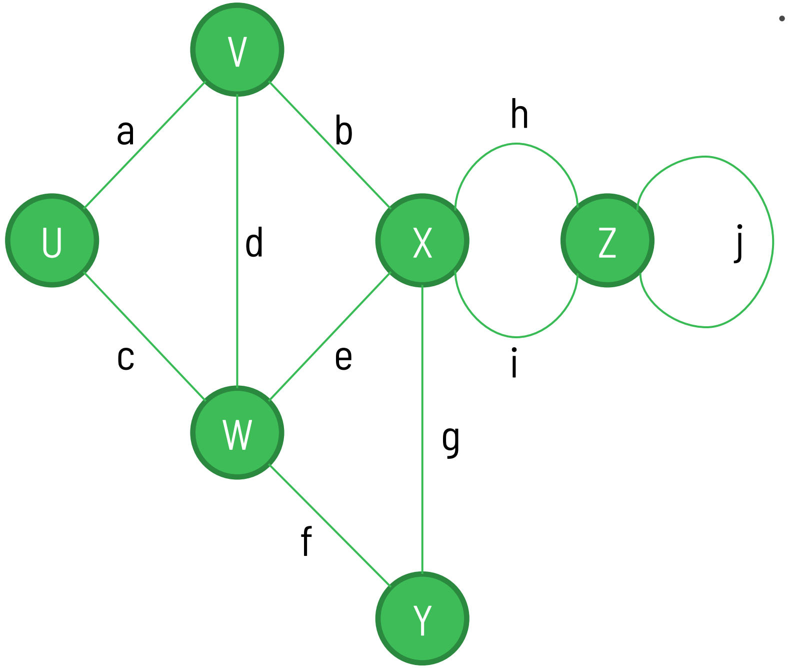
A forest is an undirected graph without cycles (i.e., one or more trees).

The connected components of a forest are trees.



Spanning Tree

A **spanning tree** of a connected graph G is a subgraph of G that is a tree.

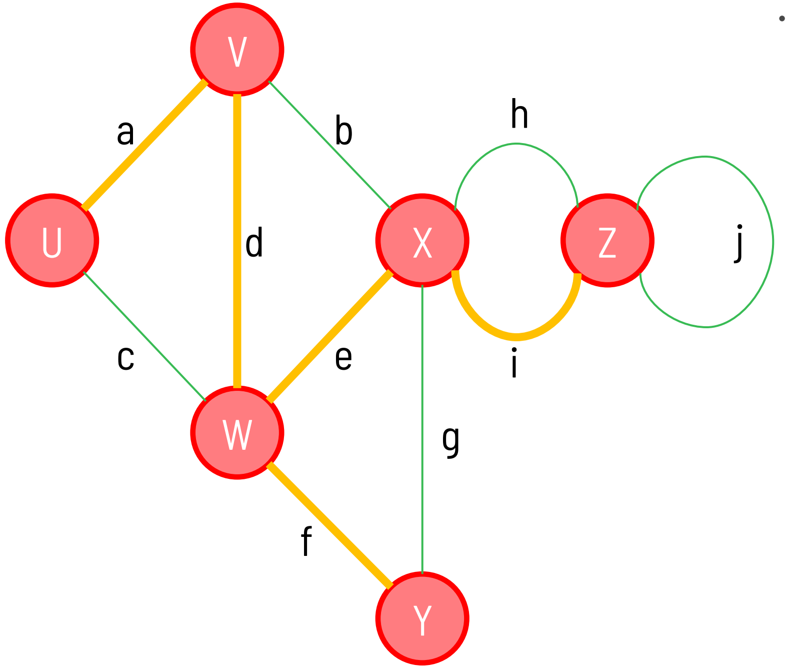


Spanning Tree

A **spanning tree** of a connected graph G is a subgraph of G that is a tree.

Q: Given a graph G, is it possible to have more than one spanning tree?
A: Yes, unless G is already a tree.

A **spanning forest** is a subgraph that is a forest.



02

Depth-First Search (DFS)

Traverse by branches

Depth-First Search (DFS)

- A general technique for traversing a graph.
- Visits all the vertices and edges of a graph.
- Determines whether a graph is connected or not.
- Computes the connected components of a graph.
- Computes a spanning forest of a graph (spanning tree if the graph is connected).
- Useful as basis for solving other problems (e.g., finding a path between two vertices).

Recursive DFS

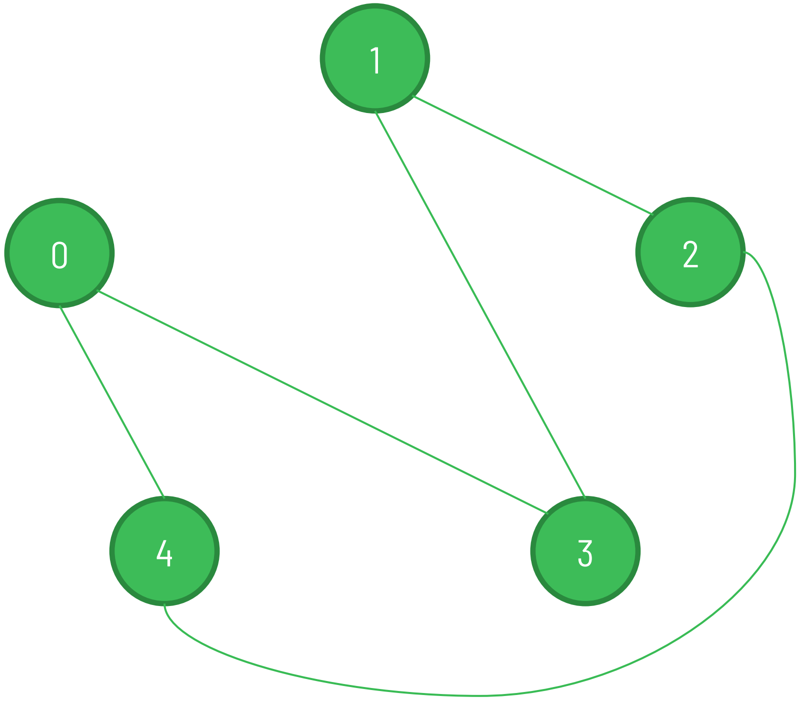
```
algorithm DFS(G(V,E), s ∈ V, T)

   mark s as visited

   for each w ∈ V adjacent to s do
      if w is not visited then
         insert (s,w) to T
         DFS(G, w)
      end if
   end for

end algorithm
```

Iterative DFS

```
algorithm DFS(G(V,E), s ∈ V)
   let S be an empty stack
   S.push(s)

   while S is not empty do
      u ← S.pop()
      if u is not visited then
         mark u as visited
         for each vertex w adjacent to u do
            S.push(w)
         end for
      end if
   end while

end algorithm
```

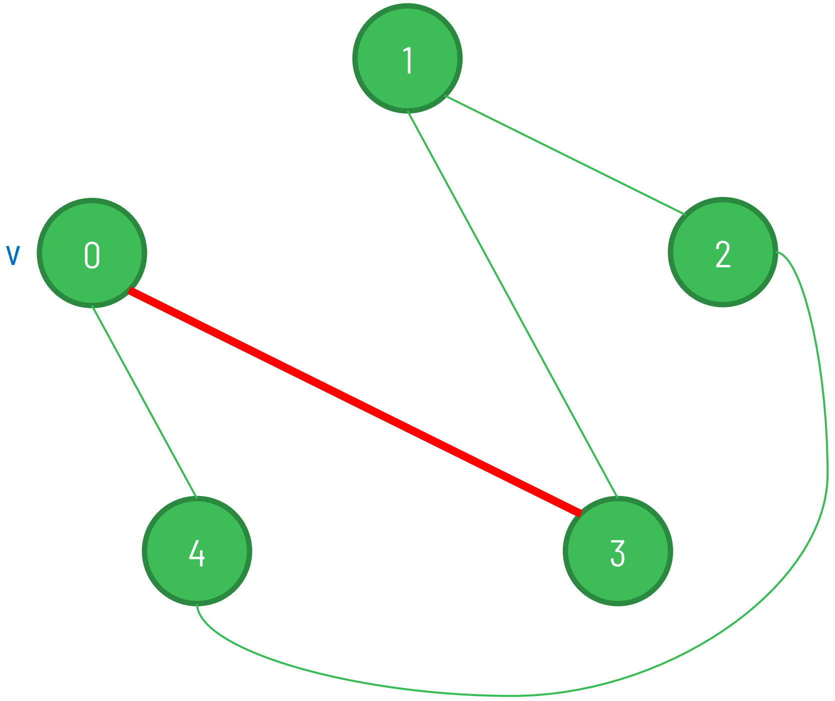
Show the DFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}



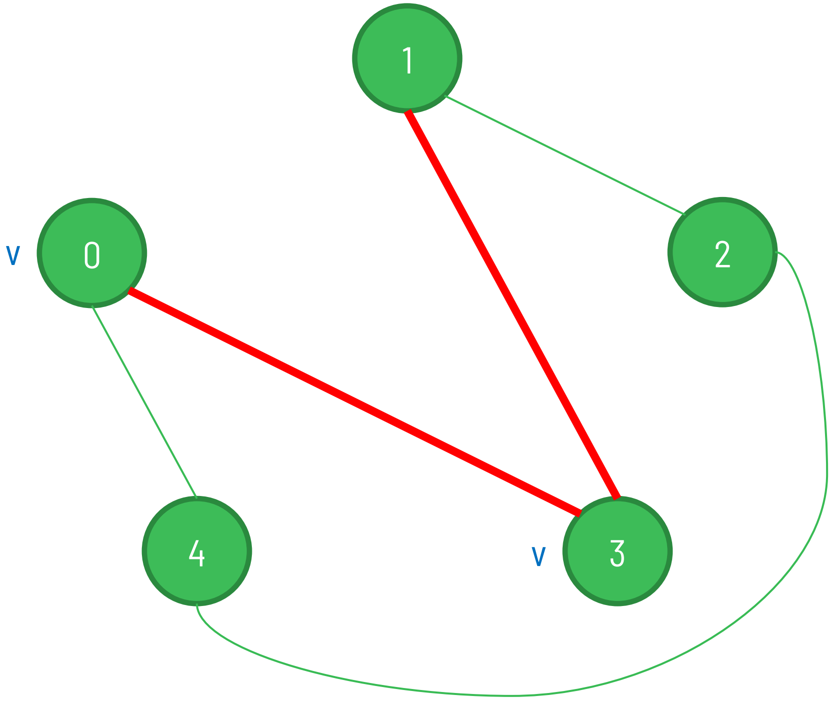
Show the DFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}



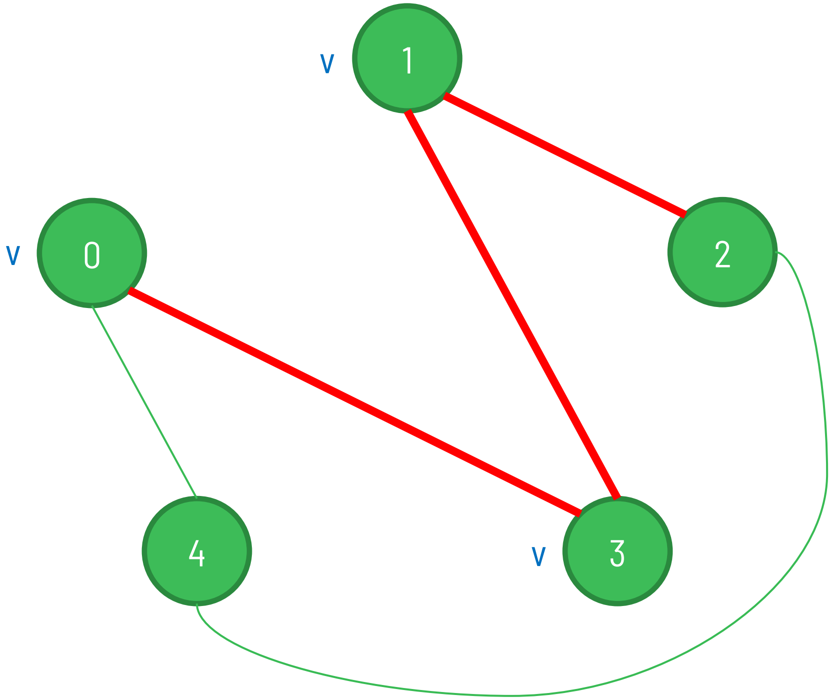
Show the DFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}



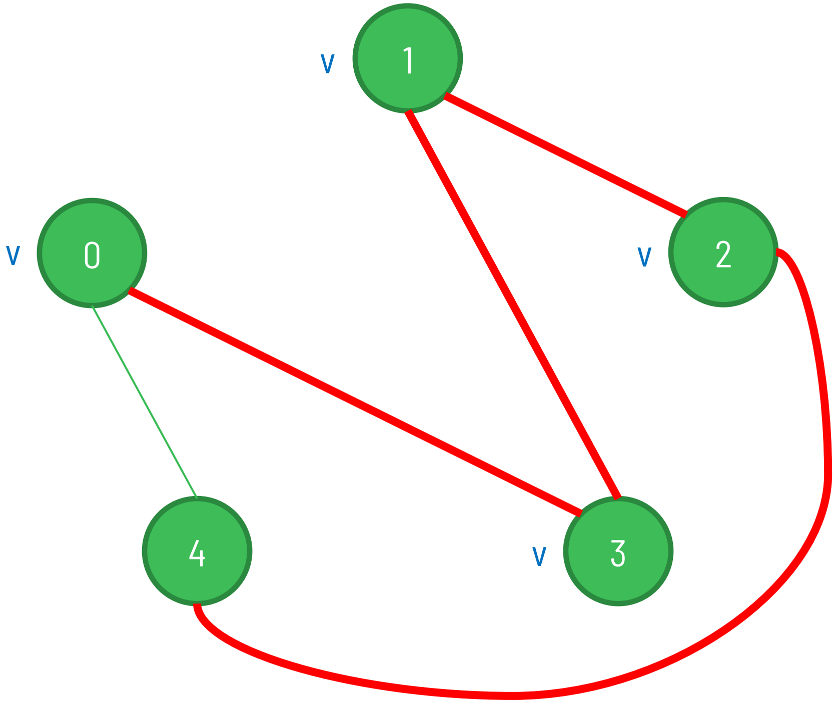
Show the DFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}



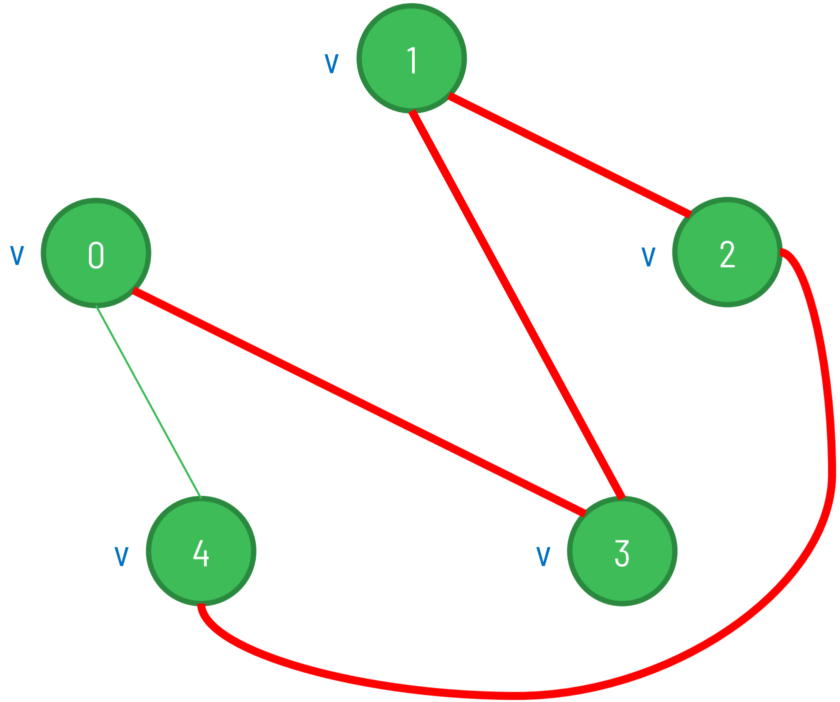
Show the DFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}



Show the DFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}



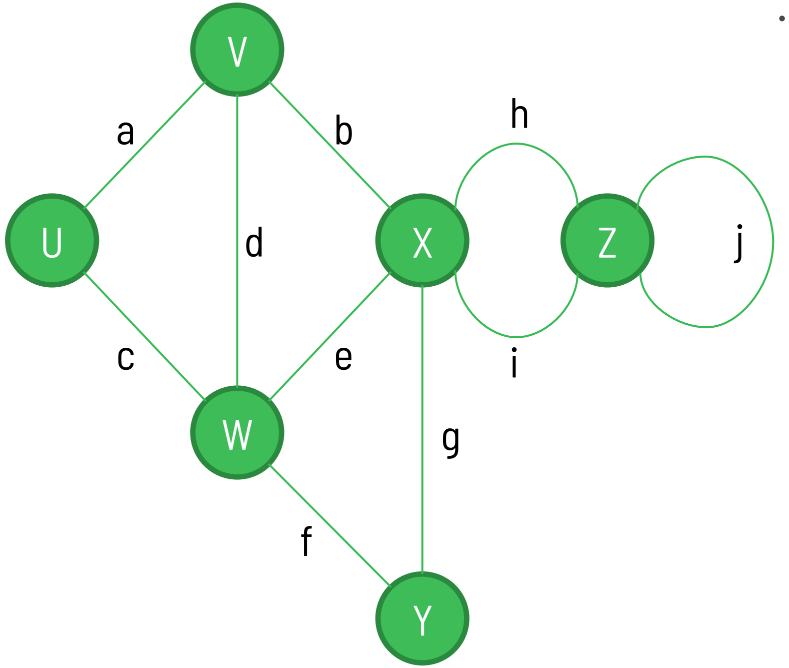
DFS Analysis

- By the end of the algorithm, we visited all the nodes $(|V|)$.
- We traverse every edge twice $(2|E|)$.
- Total runtime with adjacency list representation: $O(|V| + |E|)$.
- What happens if DFS ends, and we got at least one non-visited node?

Application: Finding a Path

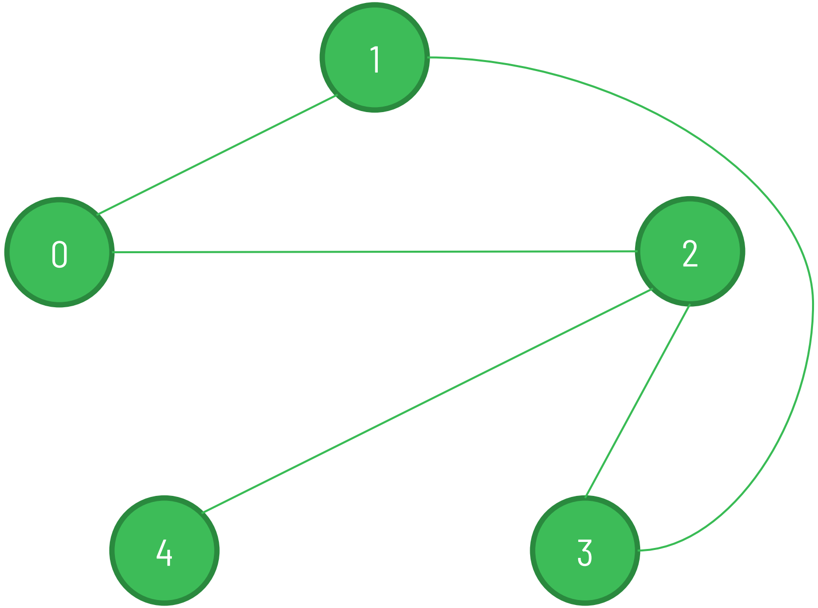
Given a graph $G(V, E)$, a source vertex $v \in V$, and a destination vertex $w \in V$, determine if there is a path from v to w, and if there is, find the path.

Solution: Use DFS(G, v) and a stack S to keep track of the current path.



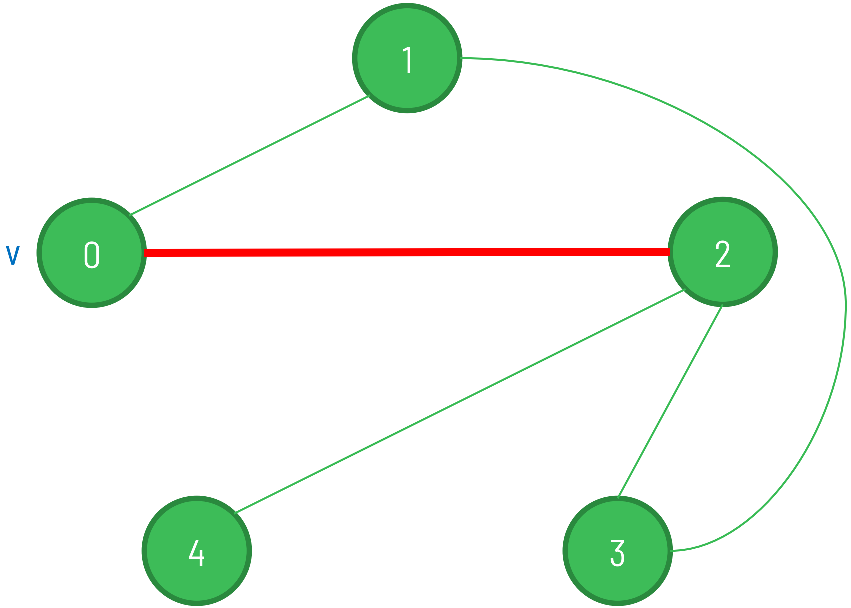
Use DFS to find a path from vertex 0 to vertex 4 in the undirected graph below:

0:	{2, 1}
1:	{0, 3}
2:	{0, 3, 4}
3:	{1, 2}
4:	{2}



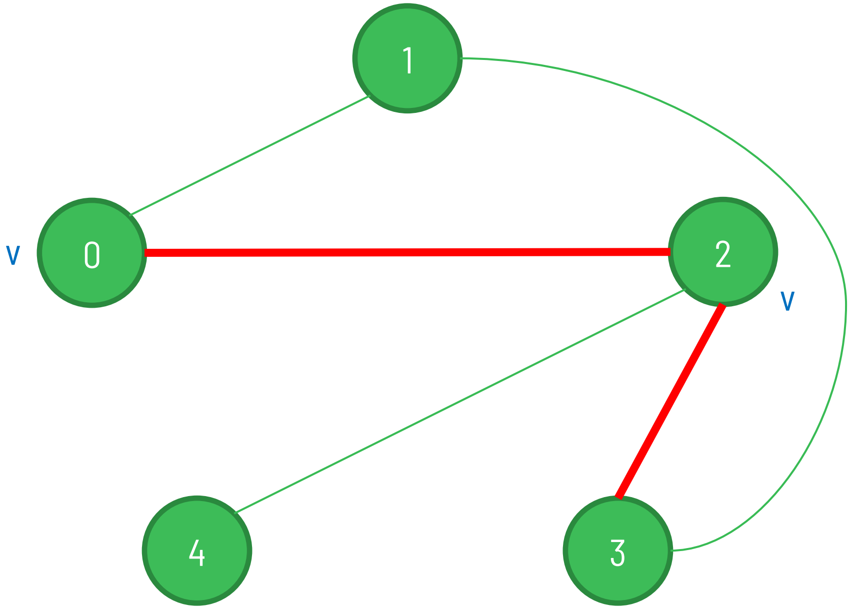
Use DFS to find a path from vertex 0 to vertex 4 in the undirected graph below:

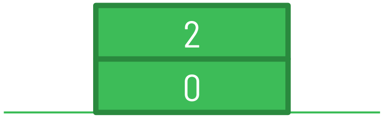
0:	{2, 1}
1:	{0, 3}
2:	{0, 3, 4}
3:	{1, 2}
4:	{2}



Use DFS to find a path from vertex 0 to vertex 4 in the undirected graph below:

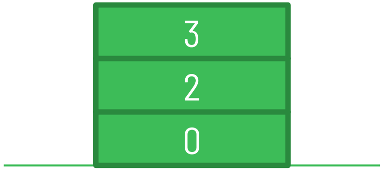
0:	{2, 1}
1:	{0, 3}
2:	{0, 3, 4}
3:	{1, 2}
4:	{2}

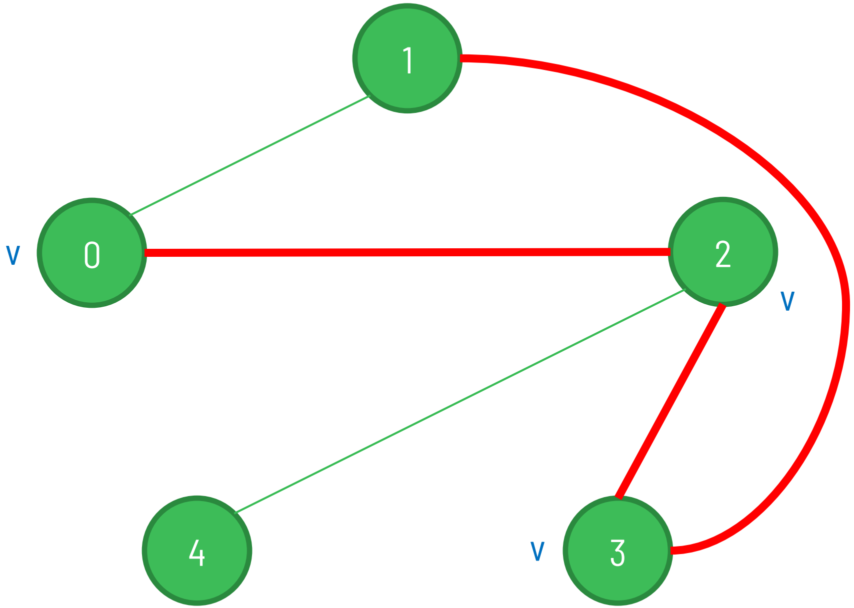




Use DFS to find a path from vertex 0 to vertex 4 in the undirected graph below:

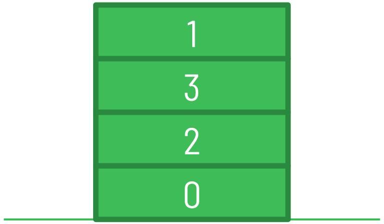
0:	{2, 1}
1:	{0, 3}
2:	{0, 3, 4}
3:	{1, 2}
4:	{2}

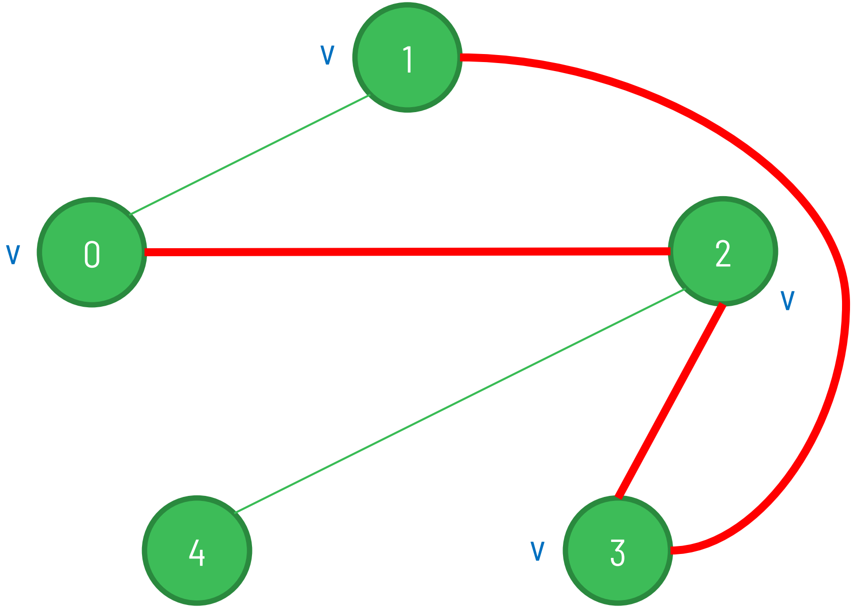




Use DFS to find a path from vertex 0 to vertex 4 in the undirected graph below:

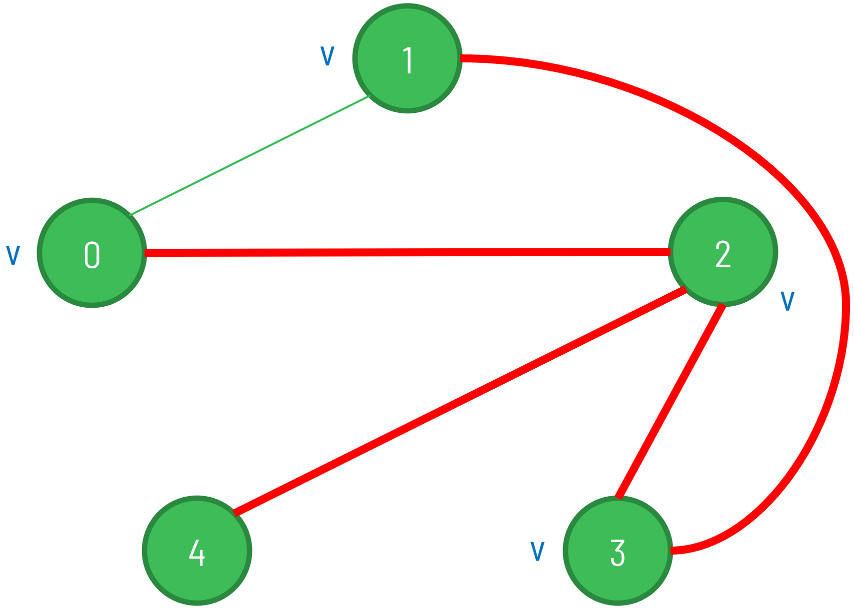
0:	{2, 1}
1:	{0, 3}
2:	{0, 3, 4}
3:	{1, 2}
4:	{2}

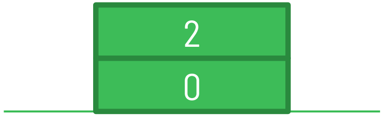




Use DFS to find a path from vertex 0 to vertex 4 in the undirected graph below:

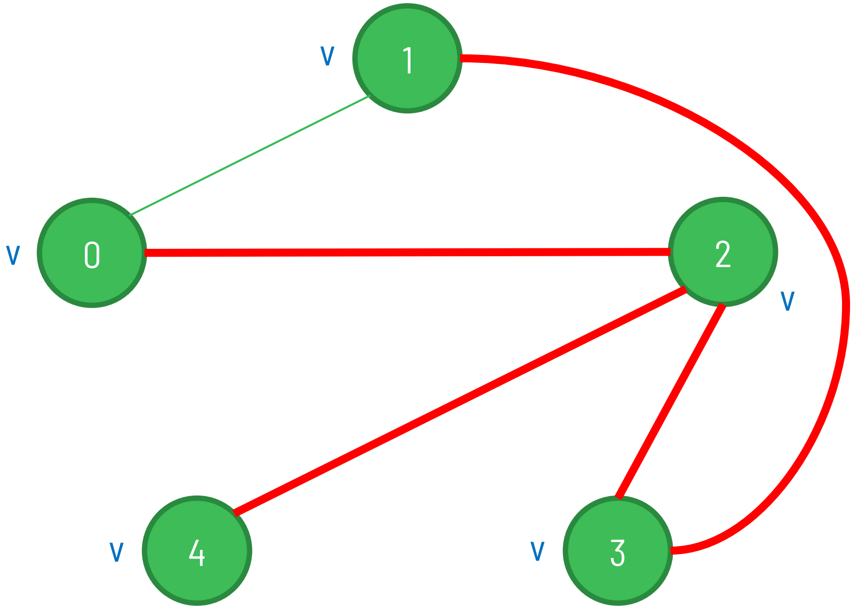
0:	{2, 1}
1:	{0, 3}
2:	{0, 3, 4}
3:	{1, 2}
4:	{2}

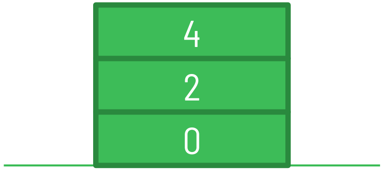




Use DFS to find a path from vertex 0 to vertex 4 in the undirected graph below:

0:	{2, 1}
1:	{0, 3}
2:	{0, 3, 4}
3:	{1, 2}
4:	{2}

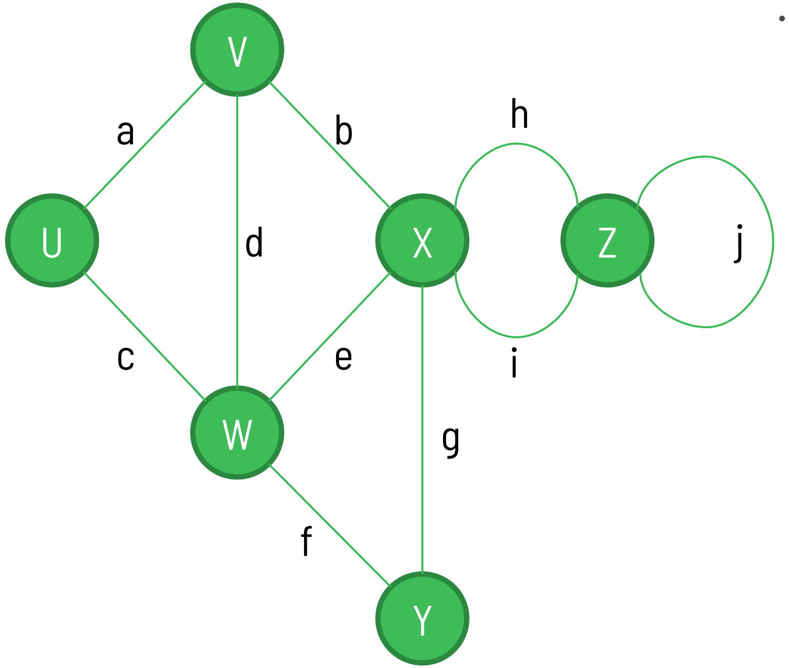




Application: Finding a Simple Cycle

Given a graph $G(V, E)$, find a simple cycle.

Solution: Use DFS(G, v) and a stack S to keep track of the current path. Repeat until the DFS algorithm is complete, or until you encounter a back edge. Then, return the stack from that edge back to the first occurrence of the repeated vertex.



Fun Application: Mazes

03

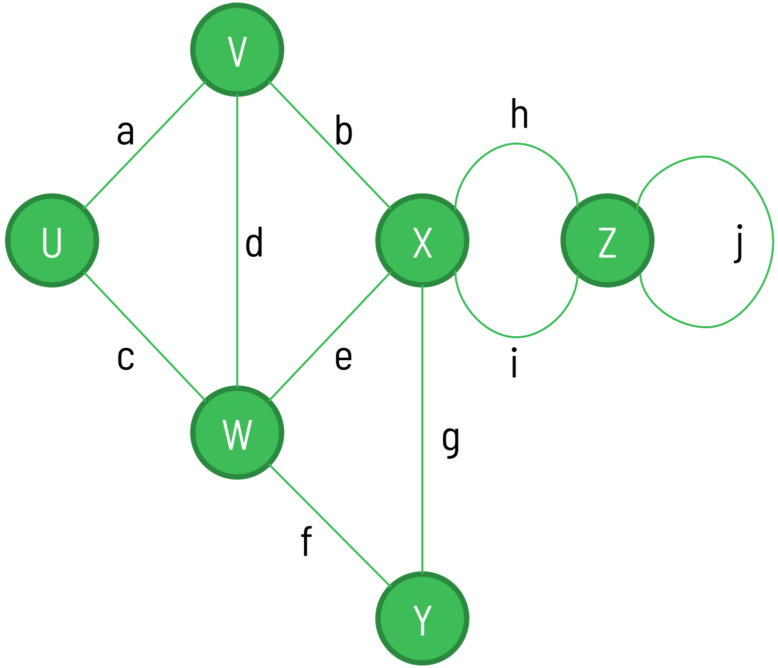
Breadth-First Search (BFS)

Traverse by levels

Single-Source Shortest Path

Given a graph $G(V, E)$ and a source vertex $v \in V$, find the shortest path (if a path exists) to a target vertex $w \in V$.

Can we do this with DFS?



Breadth-First Search (BFS)

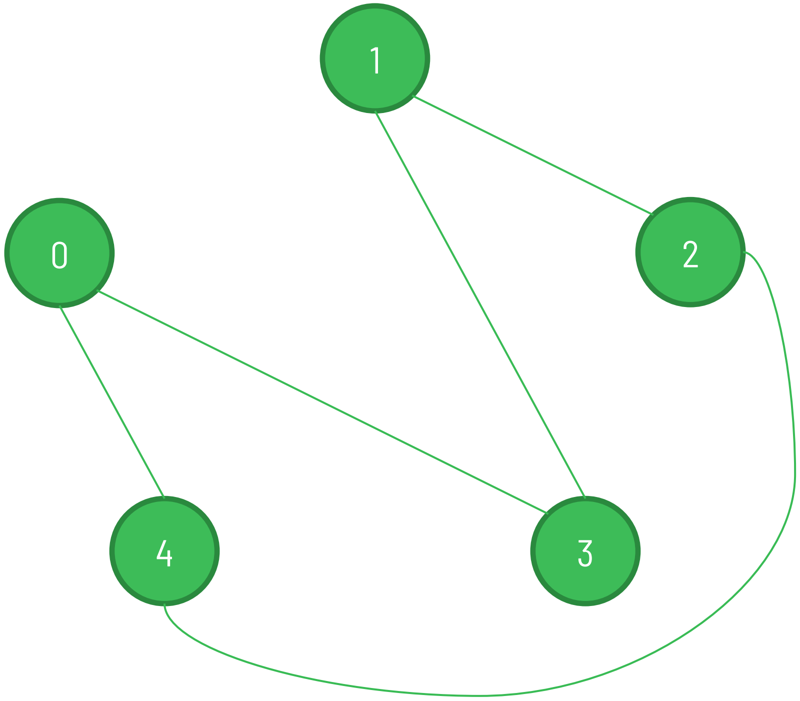
- A general technique for traversing a graph.
- Visits all the vertices and edges of a graph.
- It computes the distance (smallest number of edges) from a single vertex to each reachable vertex.
- For any vertex v reachable from u, the simple path in the BFS tree from u to v corresponds to a "shortest path" from u to v in the graph.

Breadth-First Search (BFS)

```
algorithm BFS(G(V,E), s ∈ V)

    let Q be an empty queue
    let edgeTo be an array of size |V|
    mark s as visited
    Q.enqueue(s)

    while Q is not empty do
        u ← Q.dequeue()
        for all w adjacent to u do
            if w is not visited then
                mark w as visited
                Q.enqueue(w)
                edgeTo[w] ← u
            end if
        end for
    end while

    return edgeTo
end algorithm
```

Show the BFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}

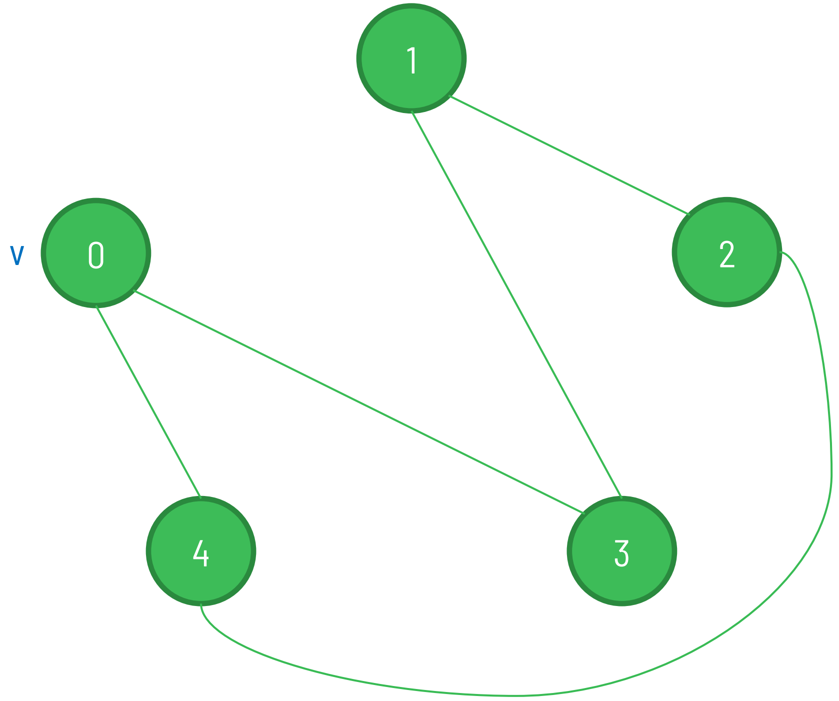


Q: {}

	0	1	2	3	4
edgeTo					

Show the BFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}

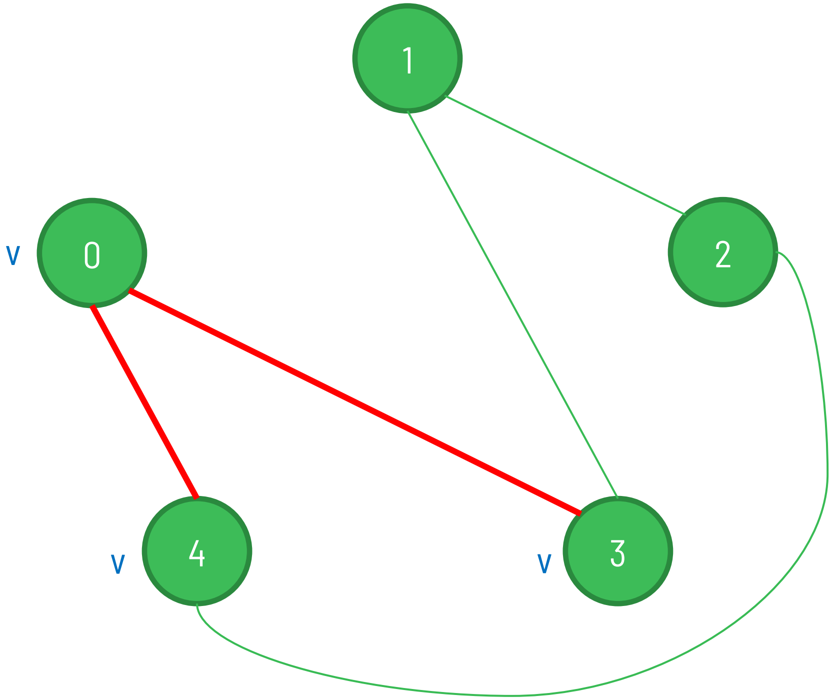


Q: {0}

	0	1	2	3	4
edgeTo					

Show the BFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}

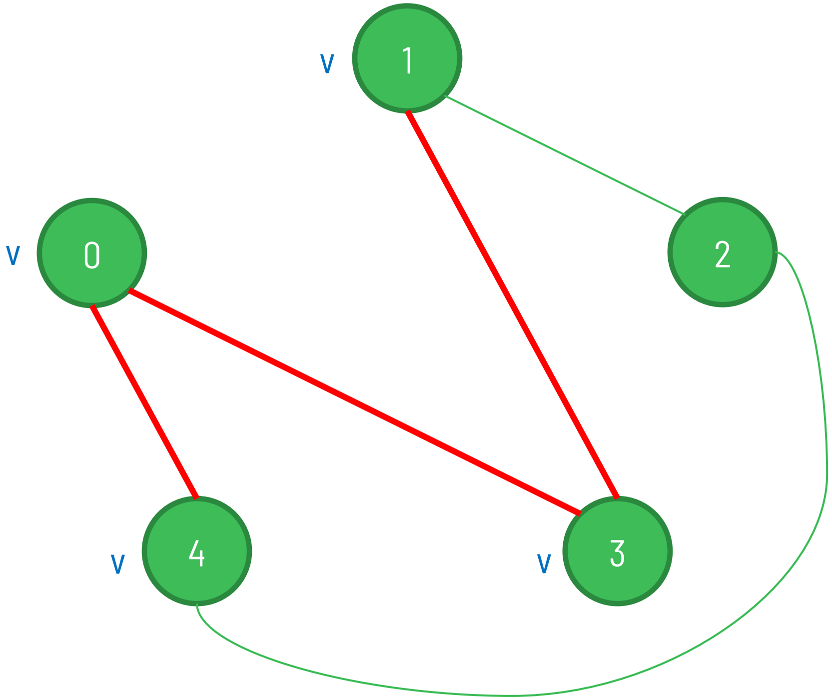


Q: {3, 4}

	0	1	2	3	4
edgeTo				0	0

Show the BFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}

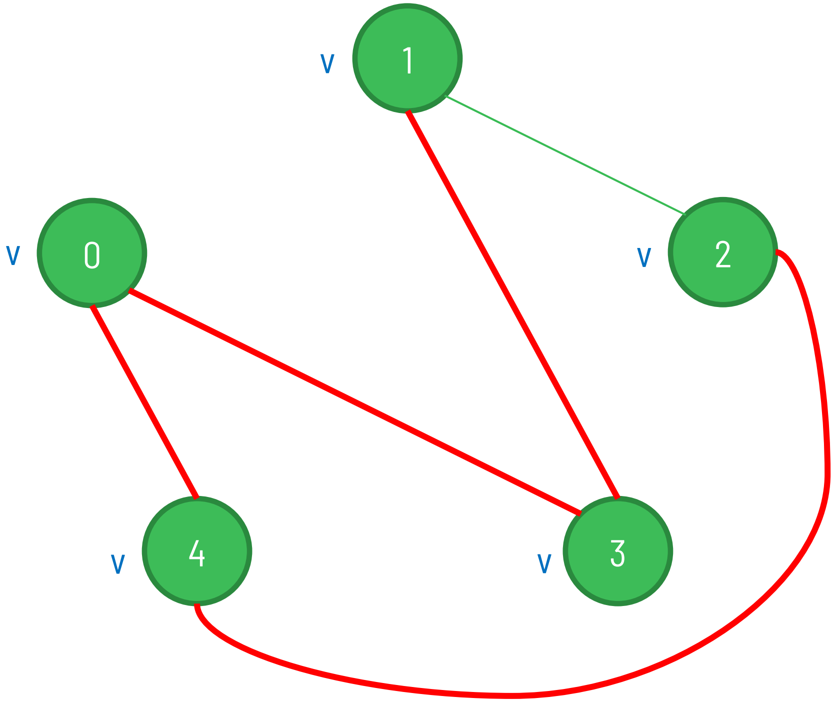


Q: {4, 1}

	0	1	2	3	4
edgeTo		3		0	0

Show the BFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}

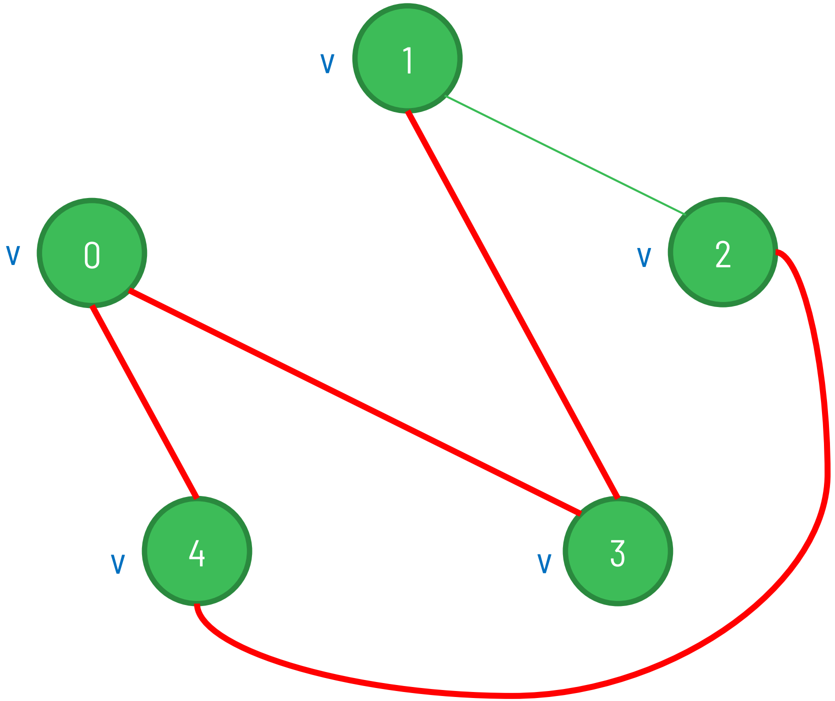


Q: {1, 2}

	0	1	2	3	4
edgeTo		3	4	0	0

Show the BFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}

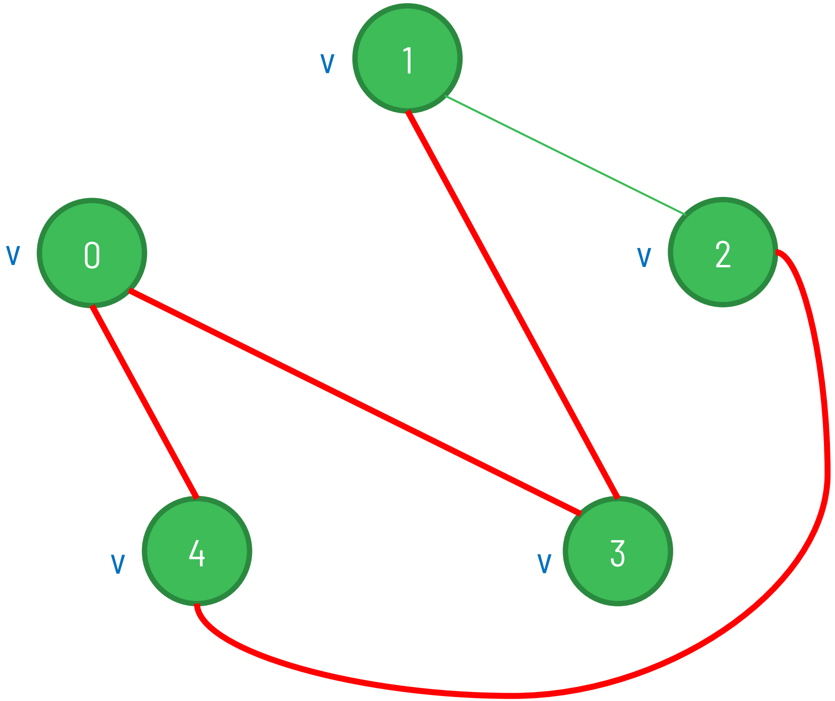


Q: {2}

	0	1	2	3	4
edgeTo		3	4	0	0

Show the BFS algorithm on the following undirected graph, starting at vertex 0.

0:	{3, 4}
1:	{2, 3}
2:	{1, 4}
3:	{0, 1}
4:	{0, 2}



Q: {}

	0	1	2	3	4
edgeTo		3	4	0	0

BFS Analysis

- By the end of the algorithm, we visited all the nodes $(|V|)$.
- We traversed every edge twice $(2|E|)$
- Total runtime with adjacency list representation: $O(|V| + |E|)$
- edgeTo array contains the traversal of the graph.

DFS -vs- BFS

DFS

LIFO style

Goes to the end of a path before coming back to an intersection

BFS

FIFO style

Tries multiple paths, one edge at a time

Next Slide = ∅

Do you have any questions?

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, infographics & images by Freepik and illustrations by Stories