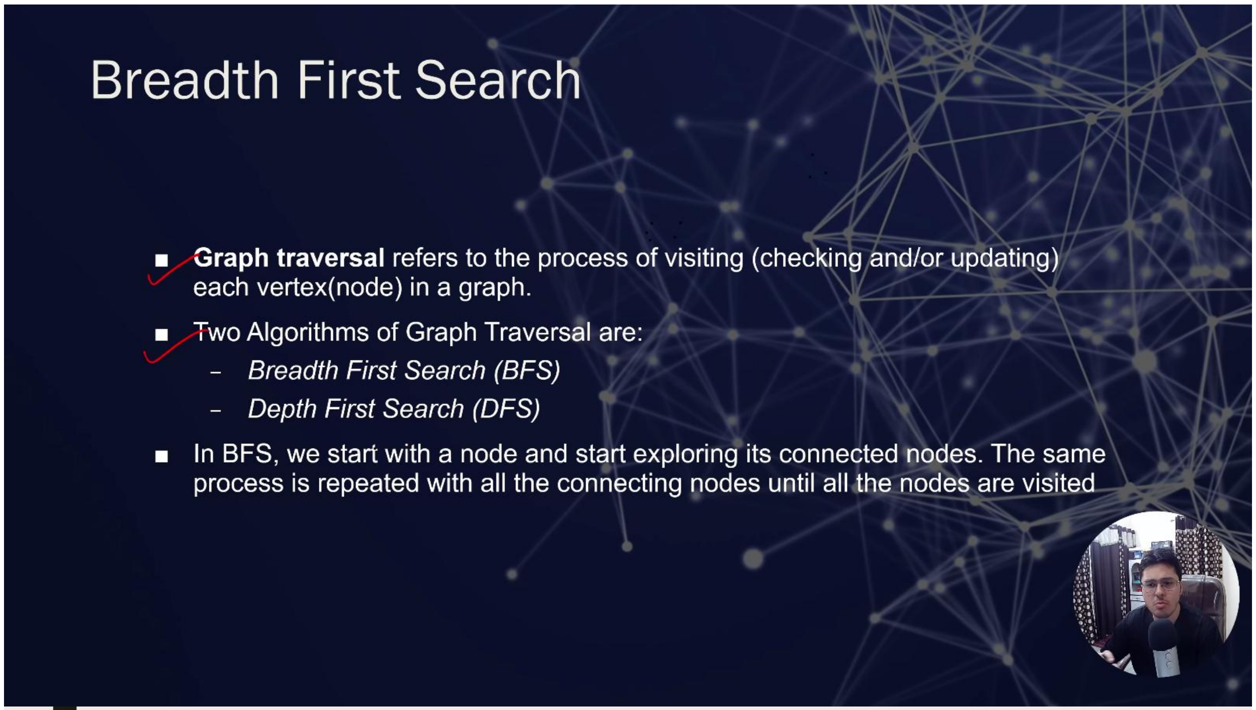
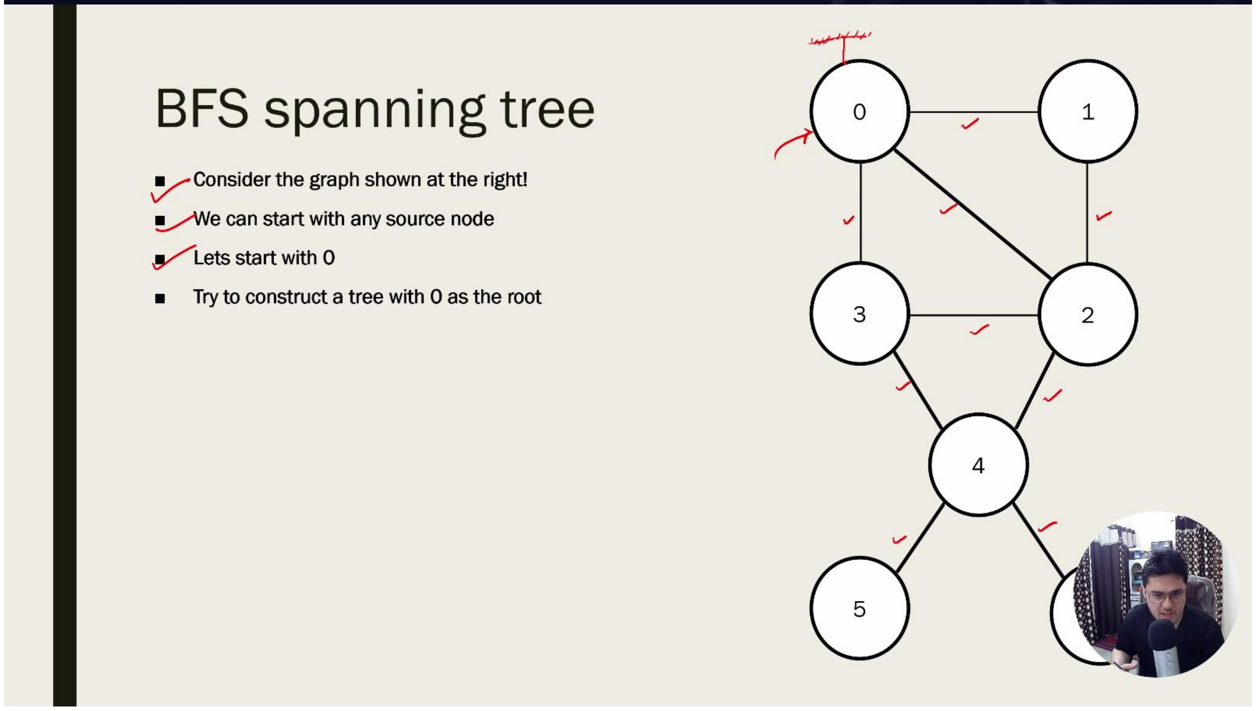


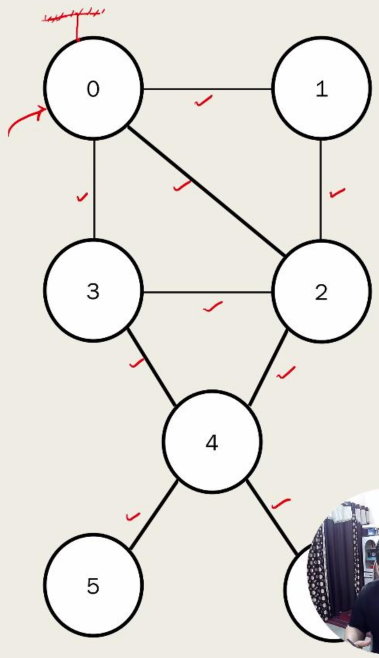
## Breadth First Search

- **Graph traversal** refers to the process of visiting (checking and/or updating) each vertex(node) in a graph.
- Two Algorithms of Graph Traversal are:
  - *Breadth First Search (BFS)*
  - *Depth First Search (DFS)*
- In BFS, we start with a node and start exploring its connected nodes. The same process is repeated with all the connecting nodes until all the nodes are visited



## BFS spanning tree

- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0
- Try to construct a tree with 0 as the root



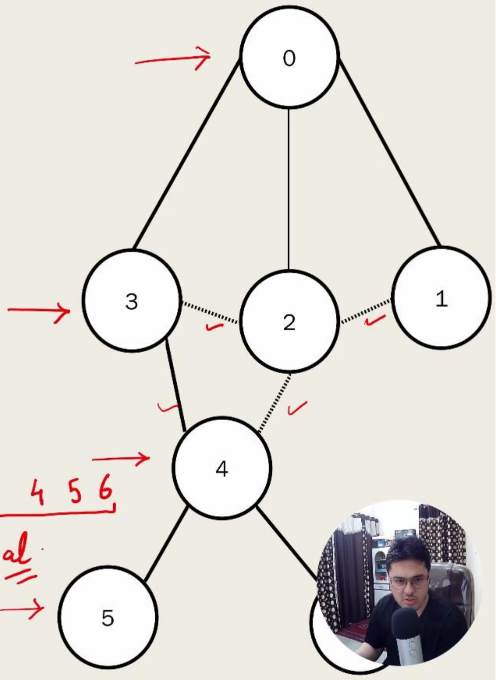
Method 1 :

## BFS spanning tree

- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0
- Try to construct a tree with 0 as the root
- Mark all the sideways or duplicate edges (above a node) as dashed
- This constructed tree is called as BFS Spanning Tree
- Level order traversal of a BFS spanning tree is a valid BFS traversal of a graph!

LOT= $\underline{0\ 3\ 2\ 1\ 4\ 5\ 6}$

↳ BFS Traversal



Method 2

## BFS Traversal

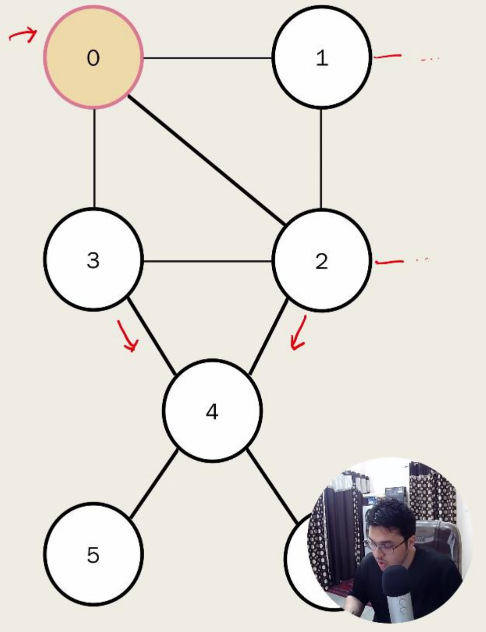
- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration

→ V : 0 1 2 3

→ EQ : $\cancel{0}$ 1 2 3

Visited: 0, 1, 2, 3

Exploration Queue: 0, 1, 3, 2



Method 2

## BFS Traversal

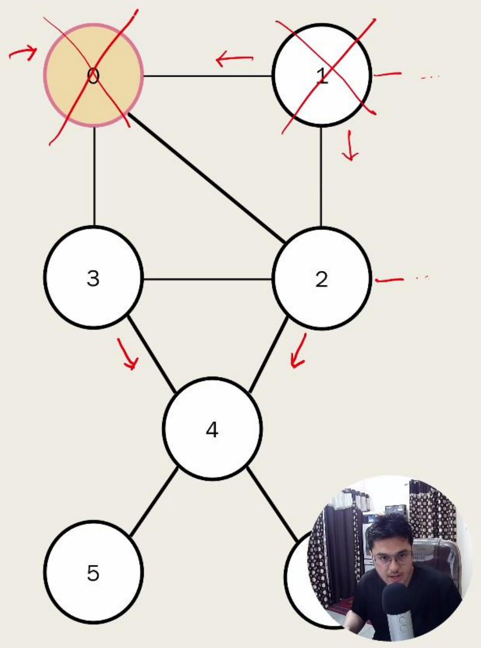
- ✓ Consider the graph shown at the right!
- ✓ We can start with any source node
- ✓ Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration

→ V : 0 1 2 3 4

→ EQ : ~~0~~ ~~1~~ ~~2~~ 3 4

Visited: 0, 1, 2, 3

Exploration Queue: 0, 1, 3, 2



Method 2

## BFS Traversal

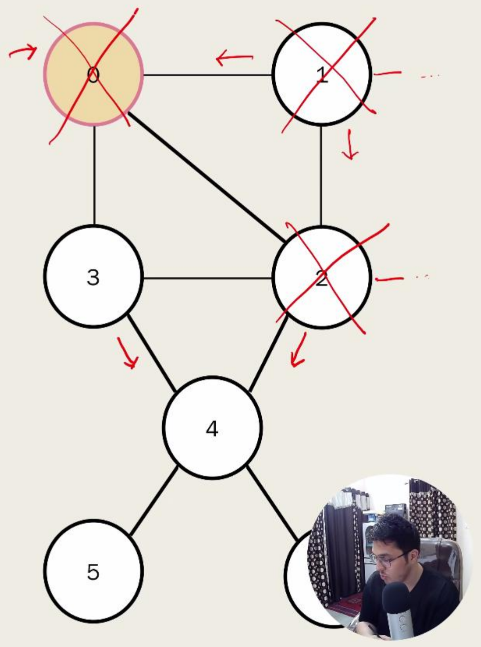
- ✓ Consider the graph shown at the right!
- ✓ We can start with any source node
- ✓ Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration

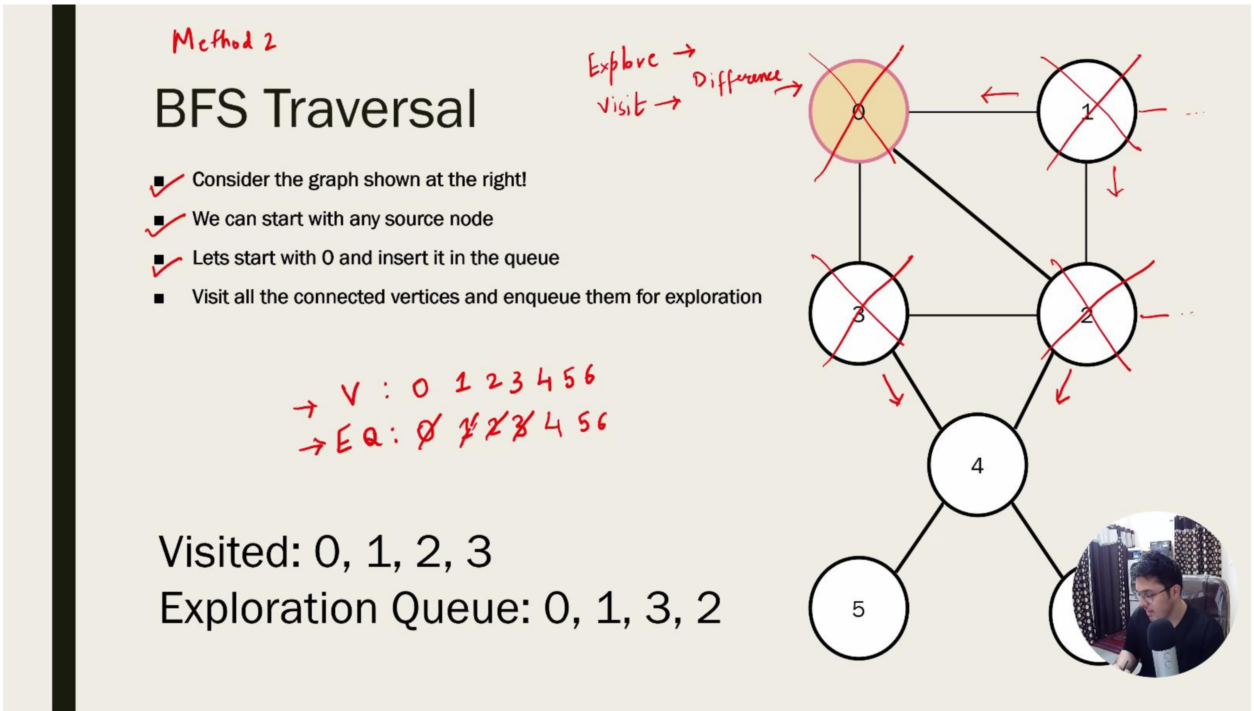
→ V : 0 1 2 3 4

→ EQ : ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

Visited: 0, 1, 2, 3

Exploration Queue: 0, 1, 3, 2





Method 2

## BFS Traversal

- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration

Explore →

Visit →

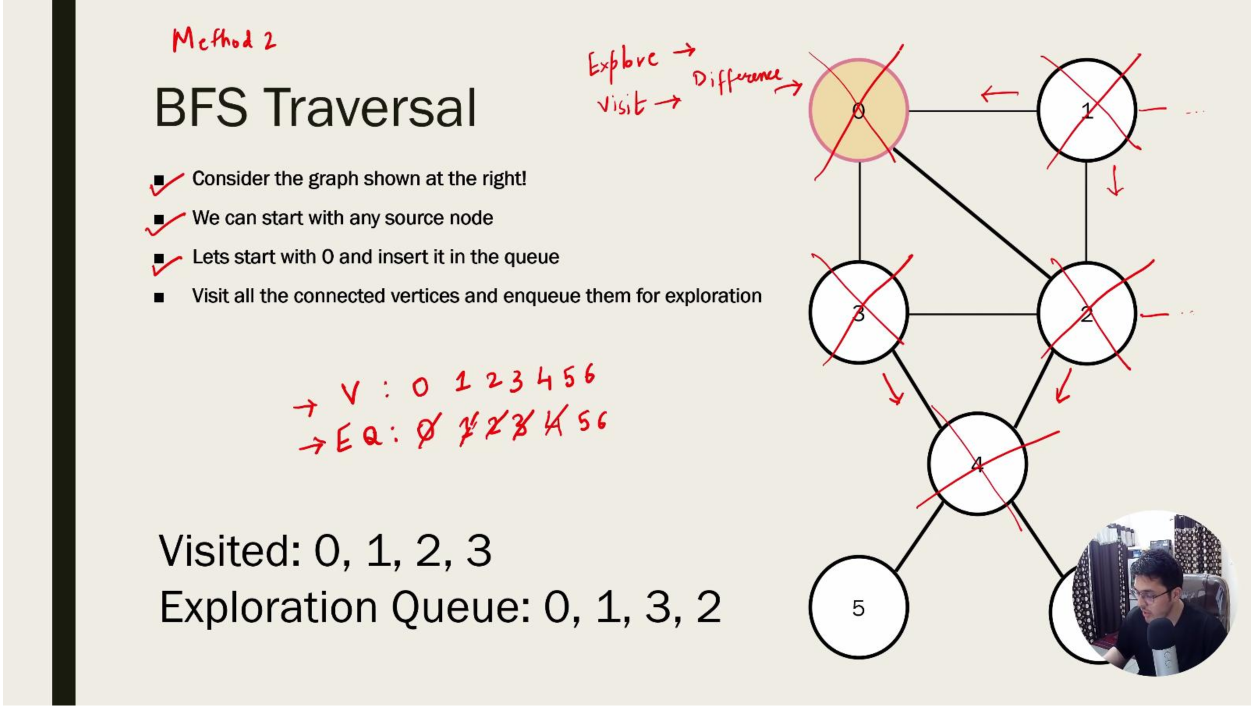
Difference →

→ V : 0 1 2 3 4 5 6

→ EQ : ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4 5 6

Visited: 0, 1, 2, 3

Exploration Queue: 0, 1, 3, 2



Method 2

## BFS Traversal

- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration

Explore →

Visit →

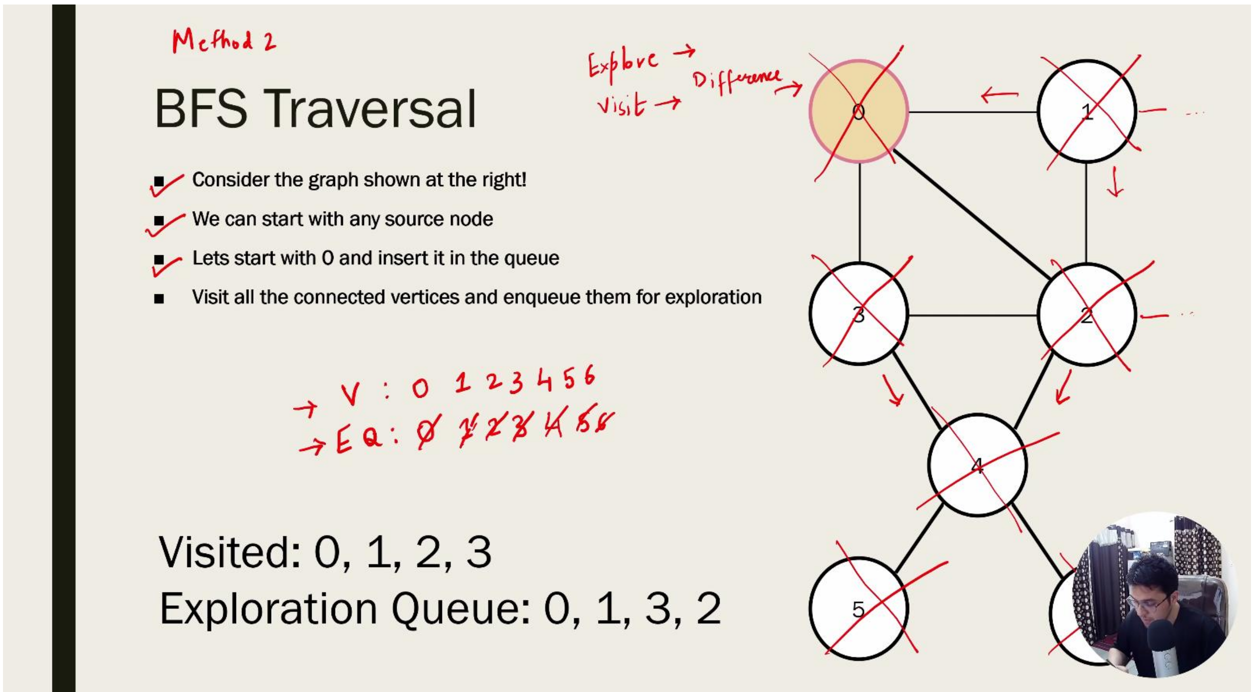
Difference →

→ V : 0 1 2 3 4 5 6

→ EQ : ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5 6

Visited: 0, 1, 2, 3

Exploration Queue: 0, 1, 3, 2



Method 2

## BFS Traversal

Explore →
Visit →
Difference →

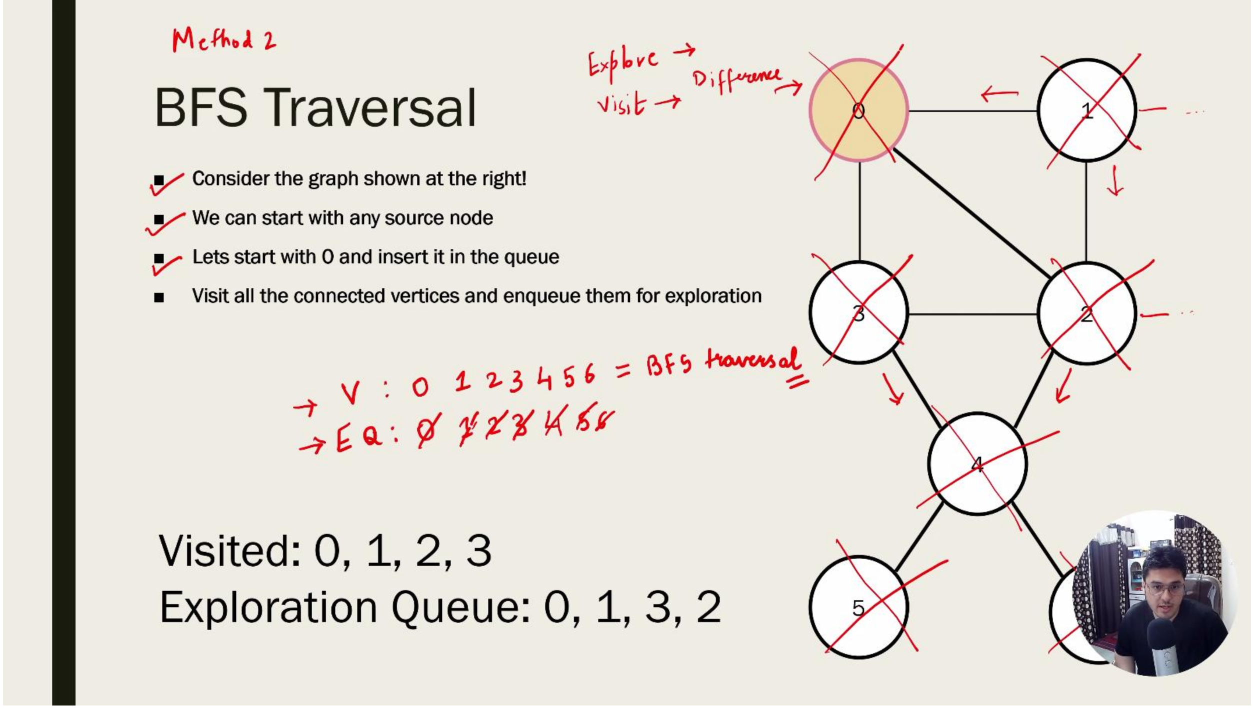
- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration

→ V : 0 1 2 3 4 5 6

→ EQ : ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~

Visited: 0, 1, 2, 3

Exploration Queue: 0, 1, 3, 2



Method 2

## BFS Traversal

Explore →
Visit →
Difference →

- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration

→ V : 0 1 2 3 4 5 6 = BFS traversal

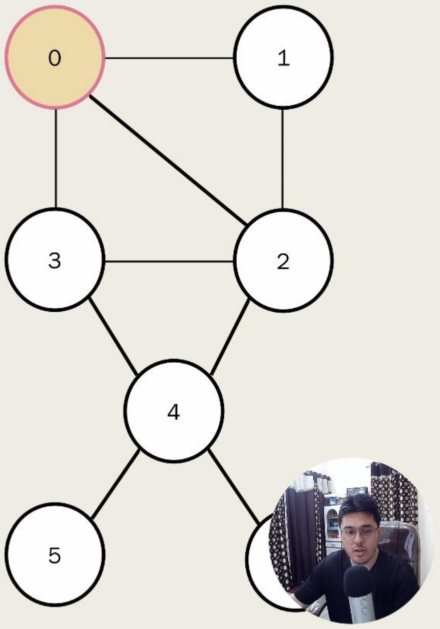
→ EQ : ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~

Visited: 0, 1, 2, 3

Exploration Queue: 0, 1, 3, 2

## BFS Traversal – Exploring 0

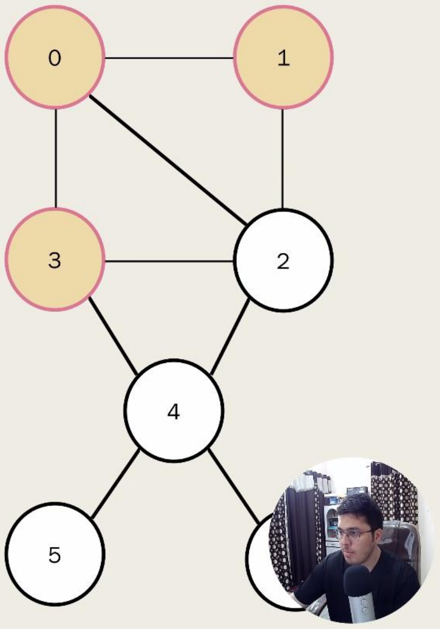
- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration (in any order)
- 0 is now explored! Let’s go to the next in queue (1)
- Repeat the same for other elements in the queue



Visited: 0, 1, 2, 3
Exploration Queue: ~~0~~, 1, 3, 2

## BFS Traversal – Exploring 3

- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration
- 0 is now explored! Let’s go to the next in queue (1)
- 1 & 3 are also explored
- Repeat the same for other elements in the queue

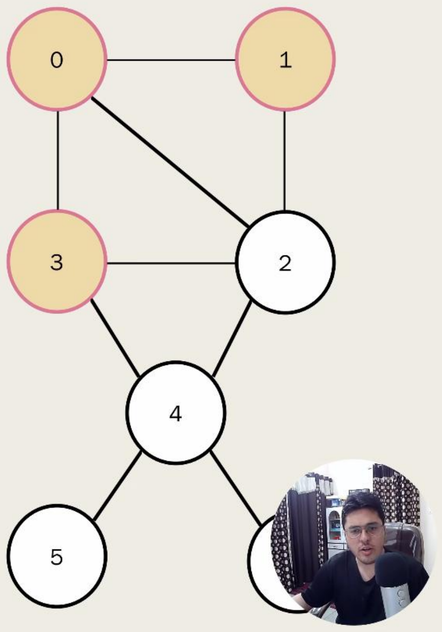


Visited: 0, 1, 2, 3
Exploration Queue: ~~0~~, ~~1~~, ~~3~~, 2

## BFS Traversal – Exploring 3

- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration
- 0 is now explored! Let's go to the next in queue (1)
- 1 & 3 are also explored
- Repeat the same for other elements in the queue

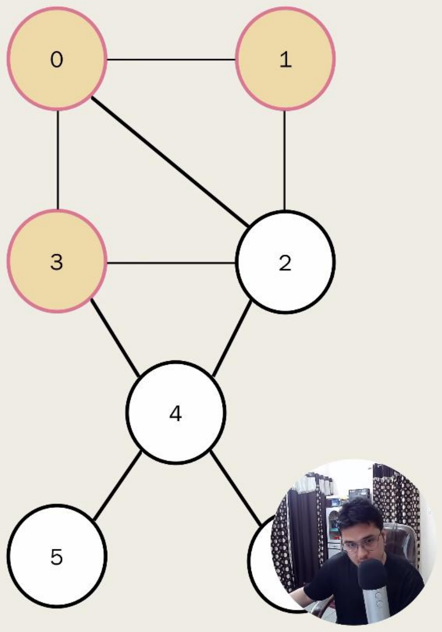
Visited: 0, 1, 2, 3, 4
Exploration Queue: ~~0~~, ~~1~~, ~~3~~, 2



## BFS Traversal – Exploring 3

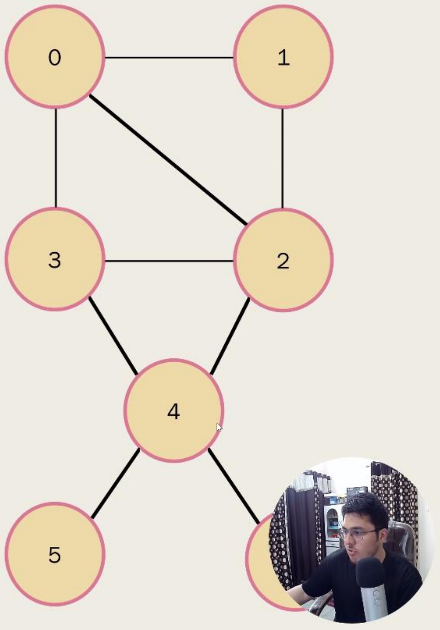
- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration
- 0 is now explored! Let's go to the next in queue (1)
- 1 & 3 are also explored
- Repeat the same for other elements in the queue

Visited: 0, 1, 2, 3, 4
Exploration Queue: ~~0~~, ~~1~~, ~~3~~, 2, 4



## BFS Traversal – Exploring 4, 5, 6

- Consider the graph shown at the right!
- We can start with any source node
- Lets start with 0 and insert it in the queue
- Visit all the connected vertices and enqueue them for exploration
- 0 is now explored! Let's go to the next in queue (1)
- Repeat the same for other elements in the queue

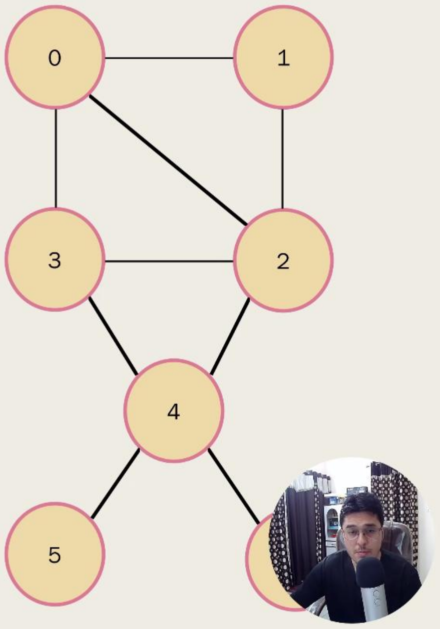


Visited: 0, 1, 2, 3, 4, 5, 6
Exploration Queue: ~~0~~, ~~1~~, ~~3~~, ~~2,~~ ~~4,~~ ~~5,~~ ~~6~~

## Algorithm: Breadth First Search

- Input: A graph G = (V,E) and source node s in V
- Algorithm:

```
Mark all nodes v in V as unvisited
Mark source node s as visited
enq(Q,s) // First-in first-out queue Q
while (Q is not empty)
  {
      u := deq(Q);
      for each unvisited neighbour v of u {
          mark v as visited;
          enq(Q,v);
      }
  }
```



## Important points

- ✓ We can start with any vertex
- ✓ There can be multiple BFS results for a given graph
- ✓ The order of visiting the vertices can be anything
- Quiz: Try to find other valid BFS for this graph (Hint: Start with nodes other than 0)