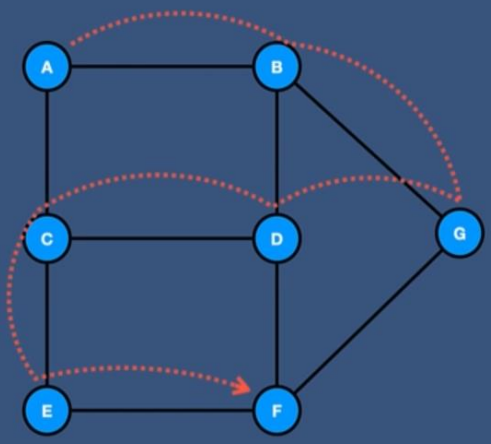
Graph Traversal

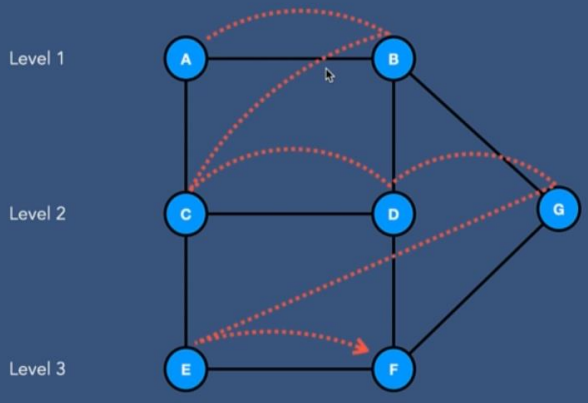
It is a process of visiting all vertices in a given Graph



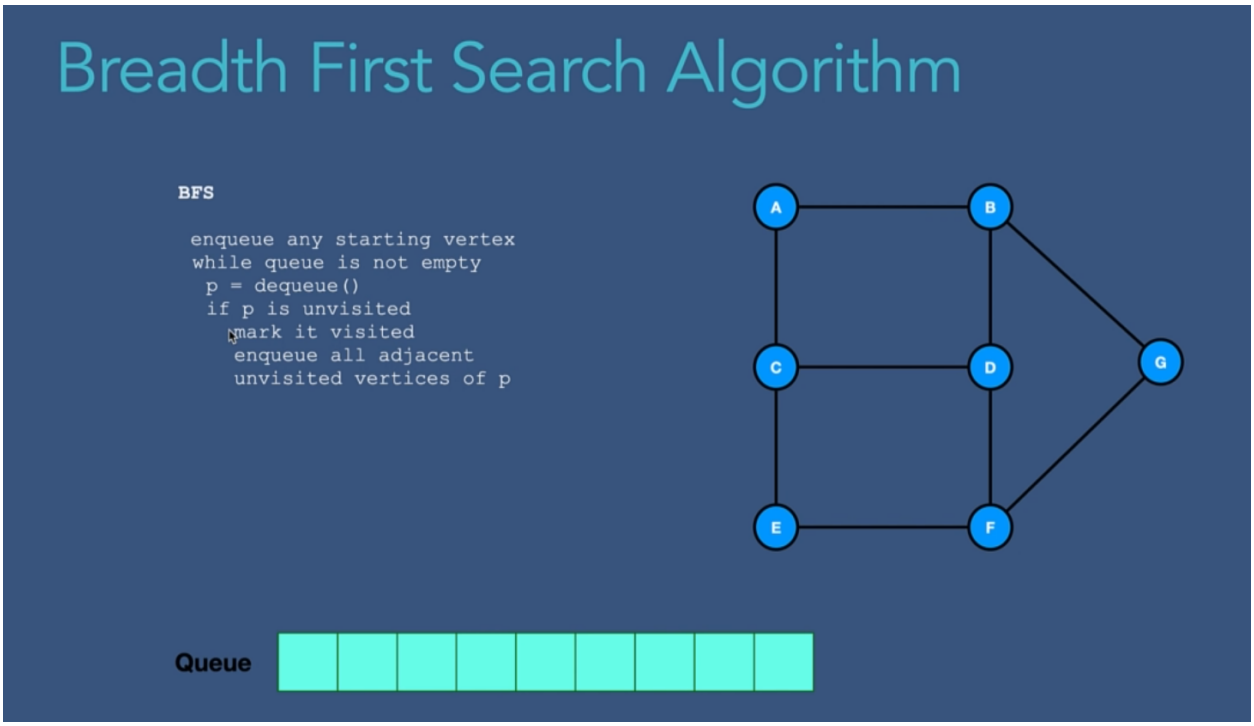
- Breadth First Search
- Depth First Search

Breadth First Search (BFS)

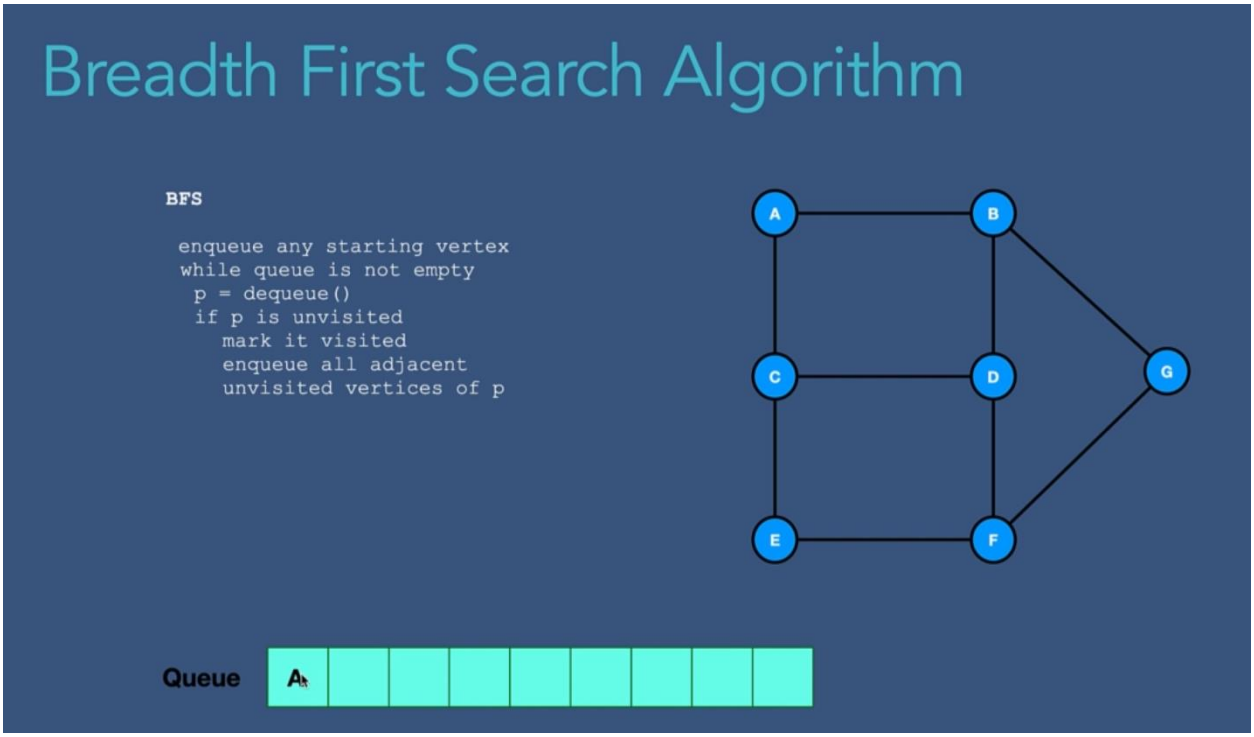
BFS is an algorithm for traversing Graph data structure. It starts at some arbitrary node of a graph and explores the neighbor nodes (which are at current level) first, before moving to the next level neighbors.



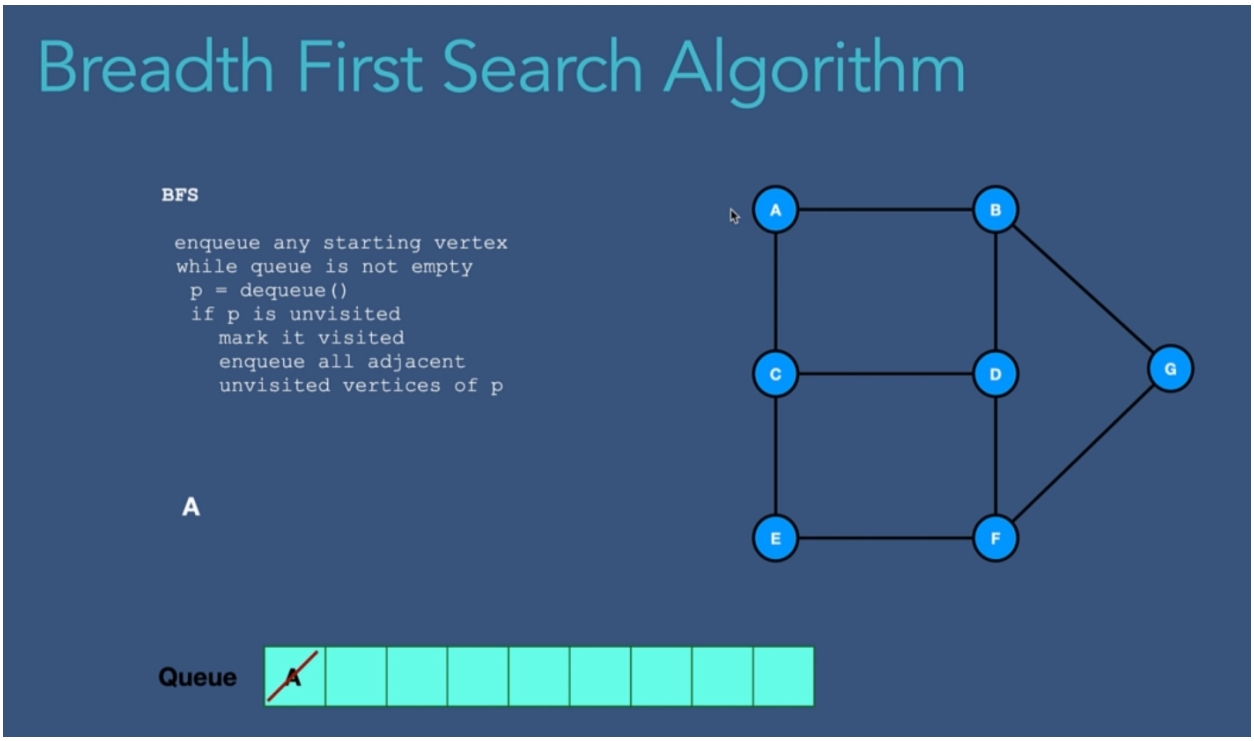
This is similar to Level Order Traversal



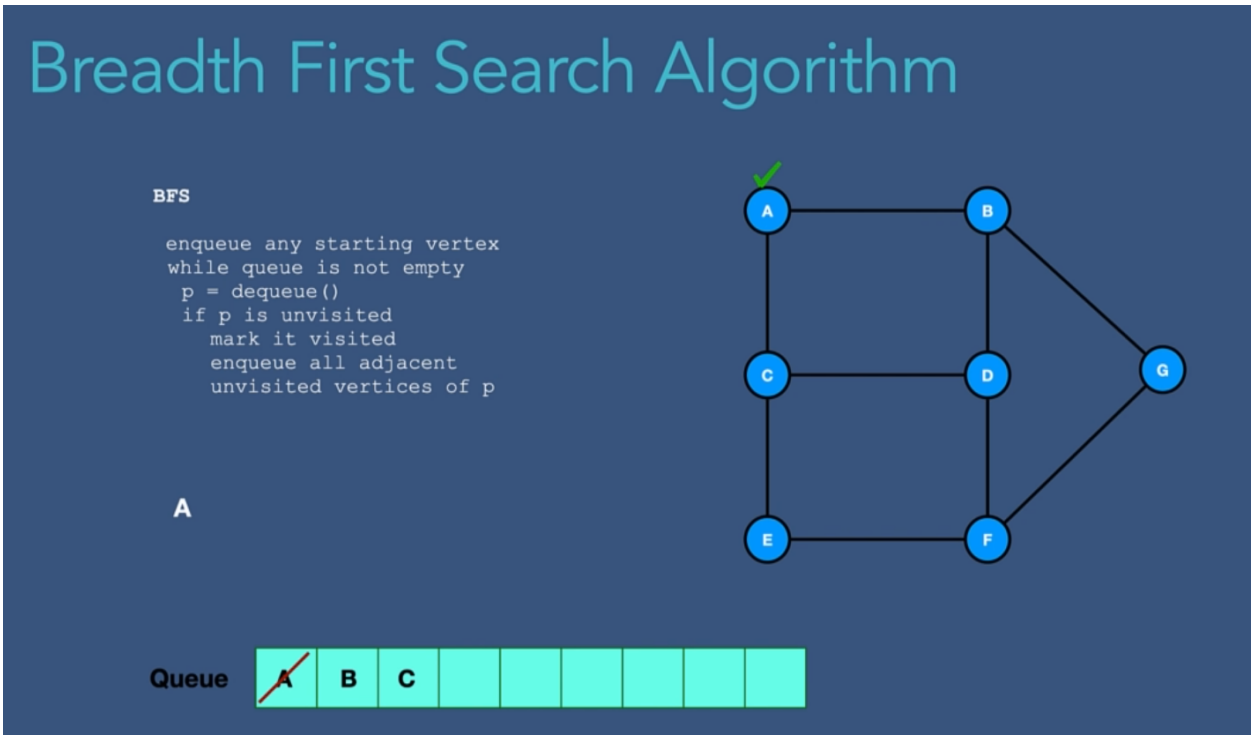
For Breadth First Search we use Queue Data Structure



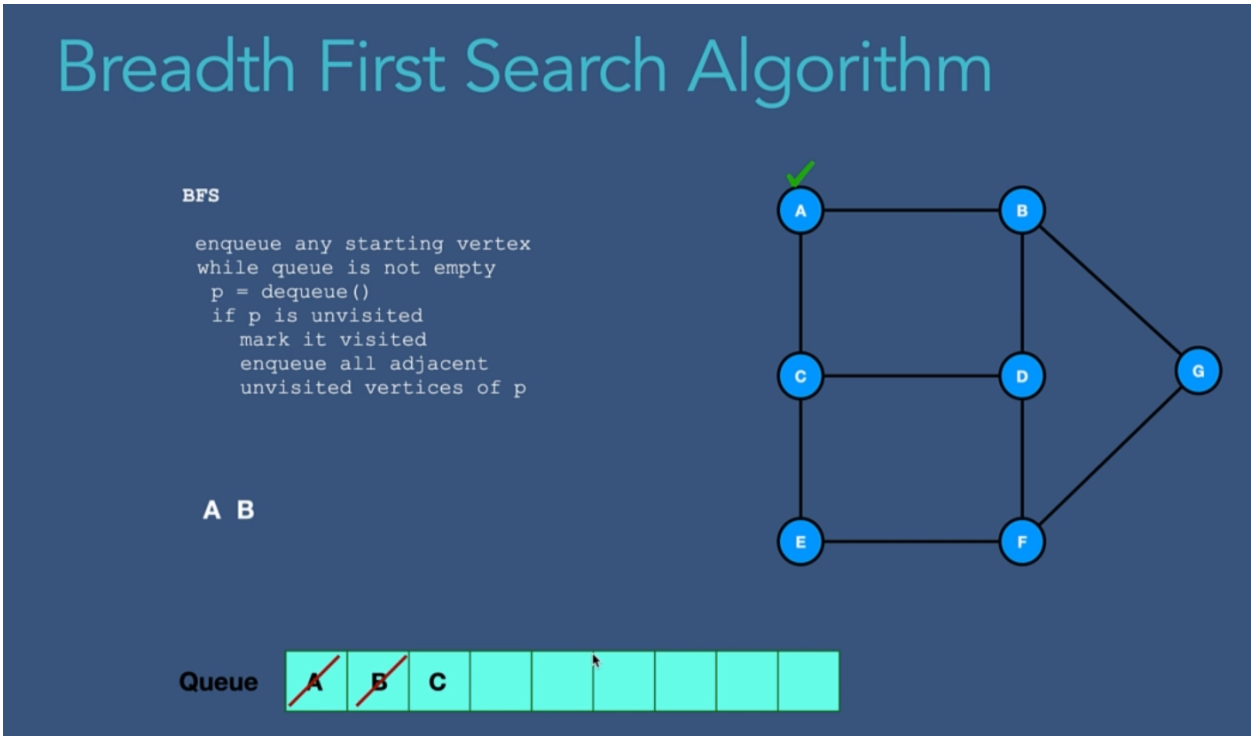
Enqueue A



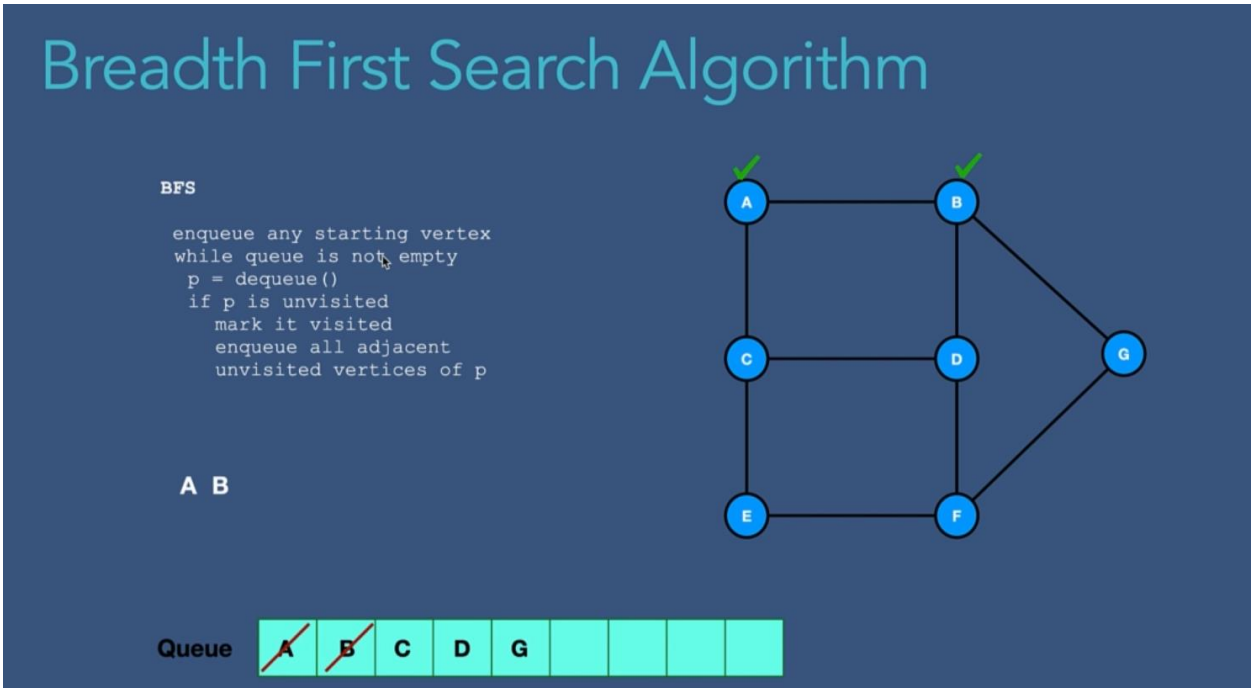
Then, Dequeue A



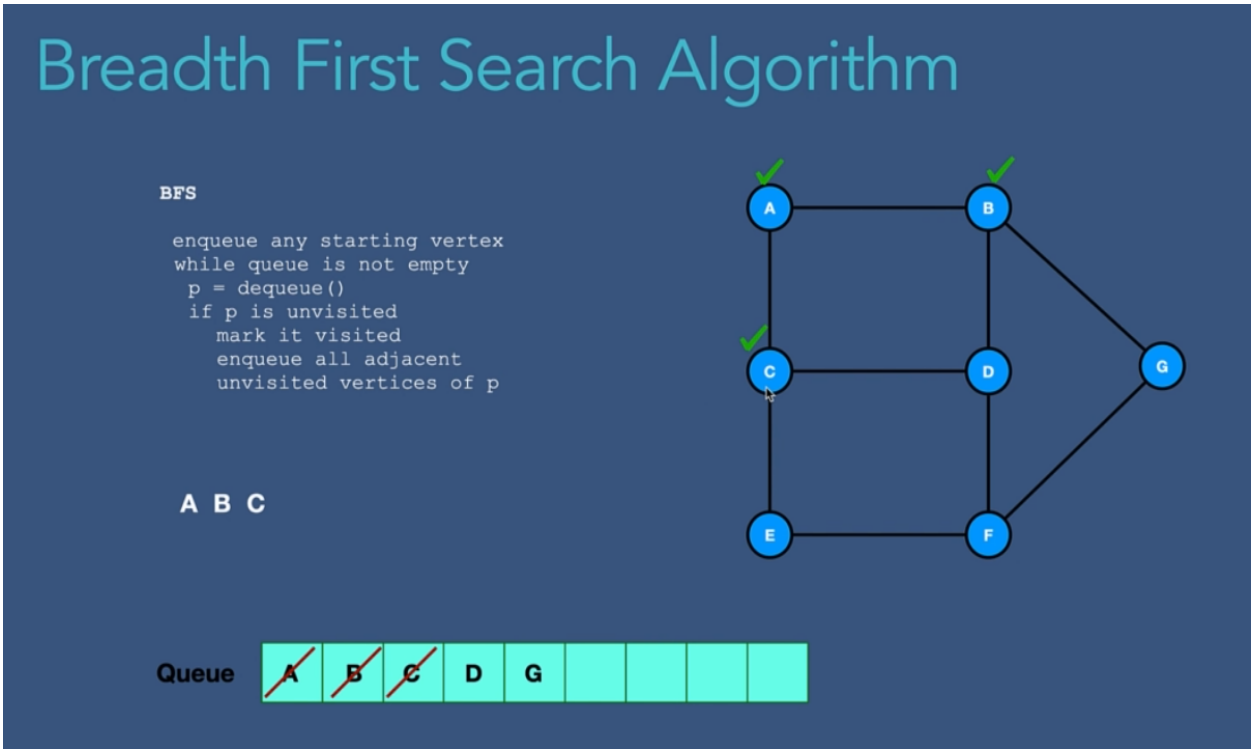
Find the adjacent nodes of A which is B and C.



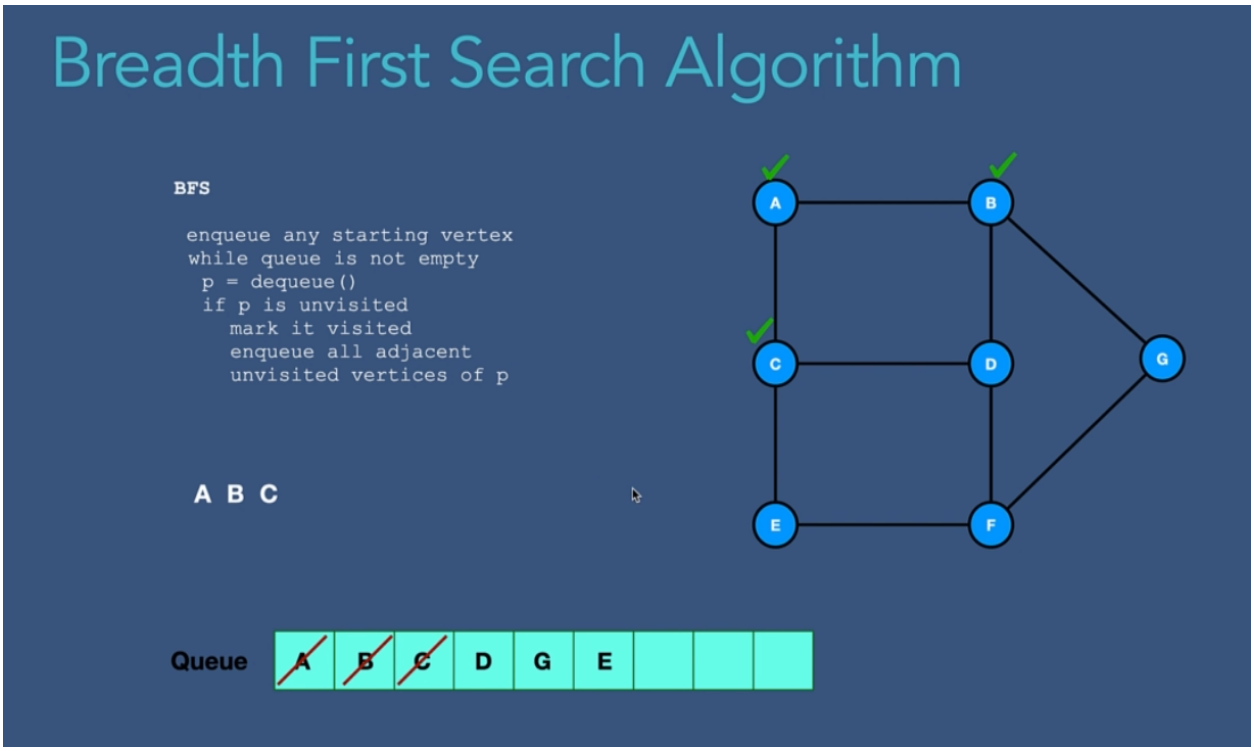
Dequeue B



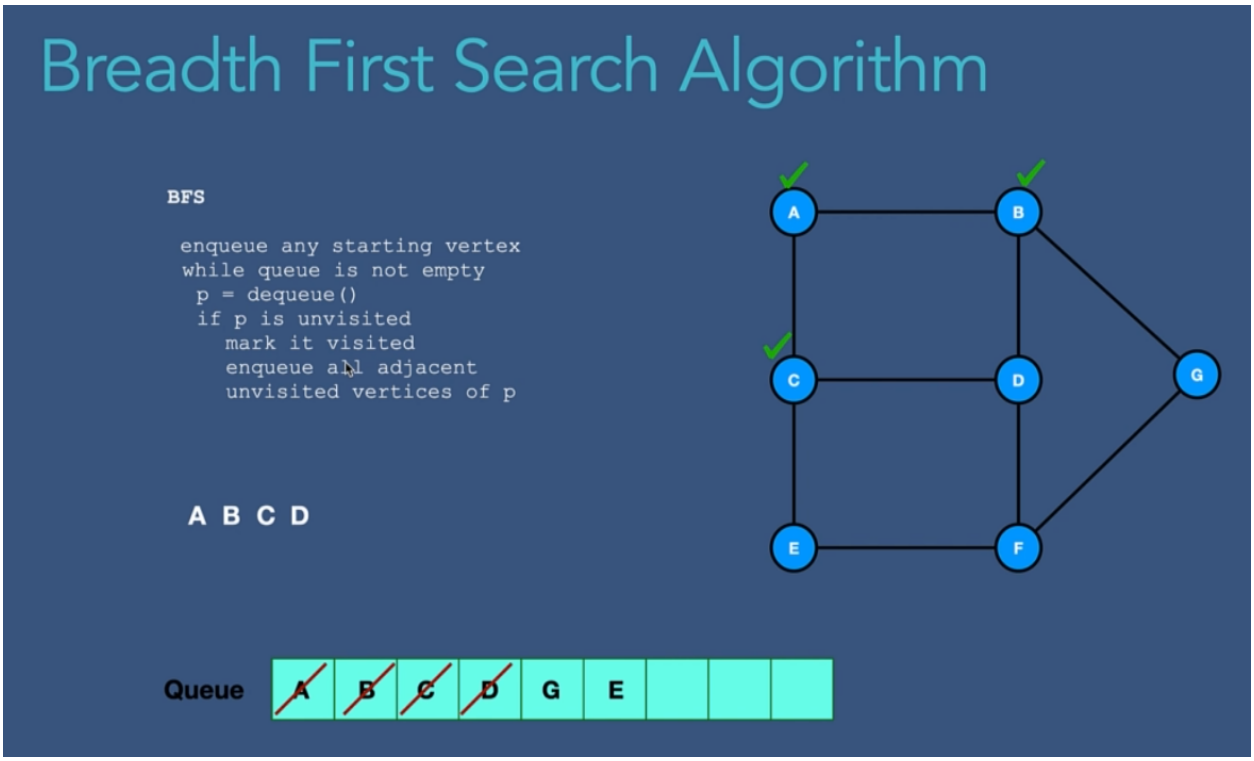
Then, the adjacent vertices of B are D and G so they are enqueued.



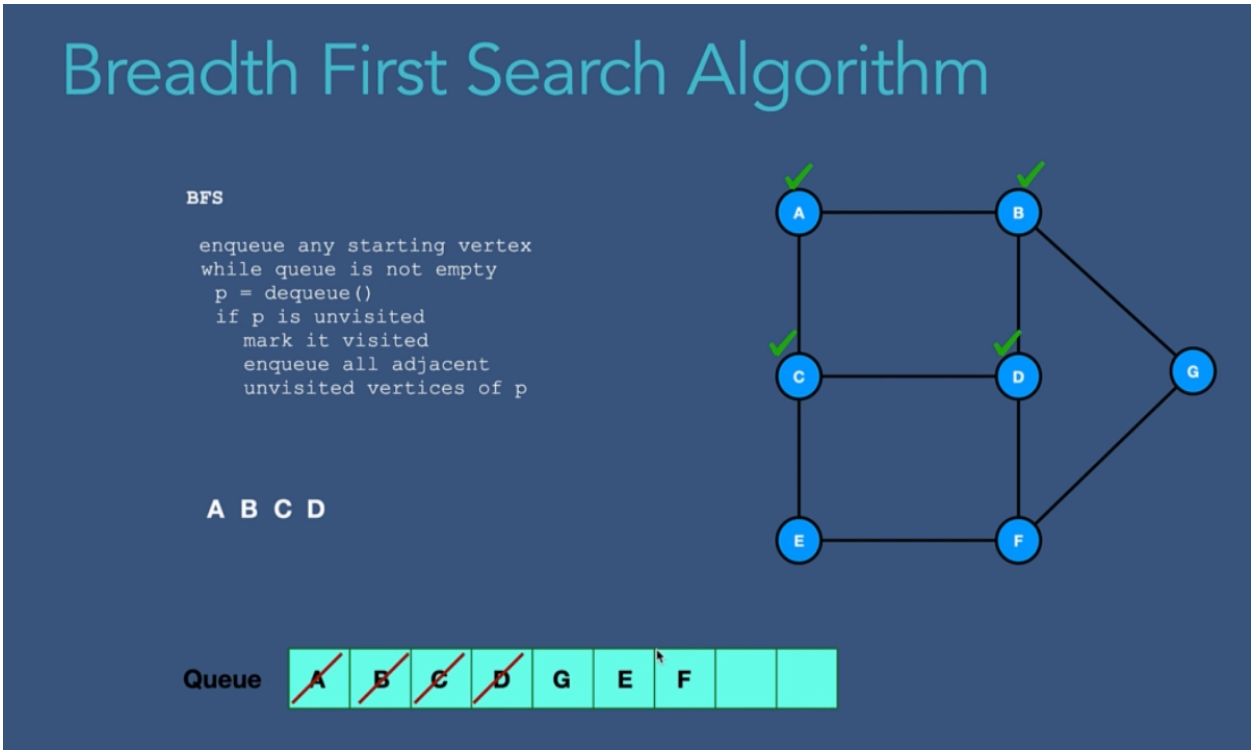
C is dequeued.



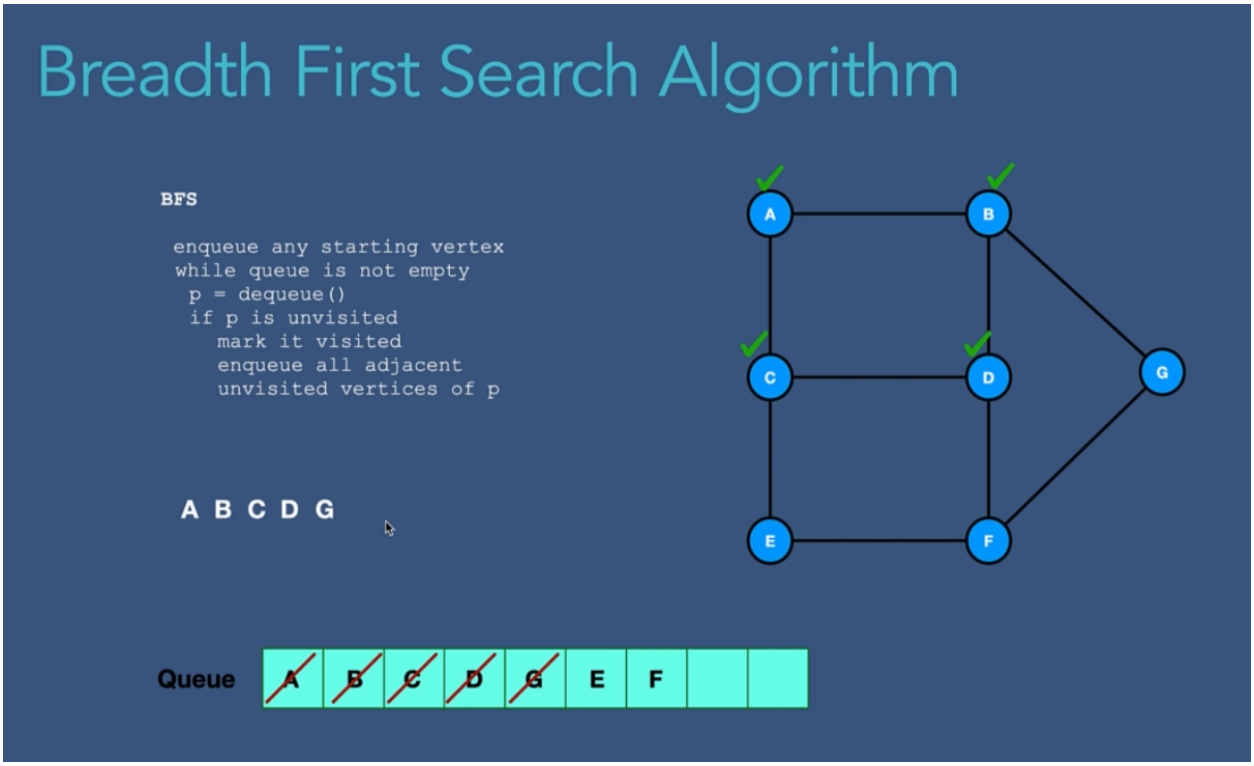
The adjacent verticesof C are D and E. D is already present in the Queue so we only enqueue E.



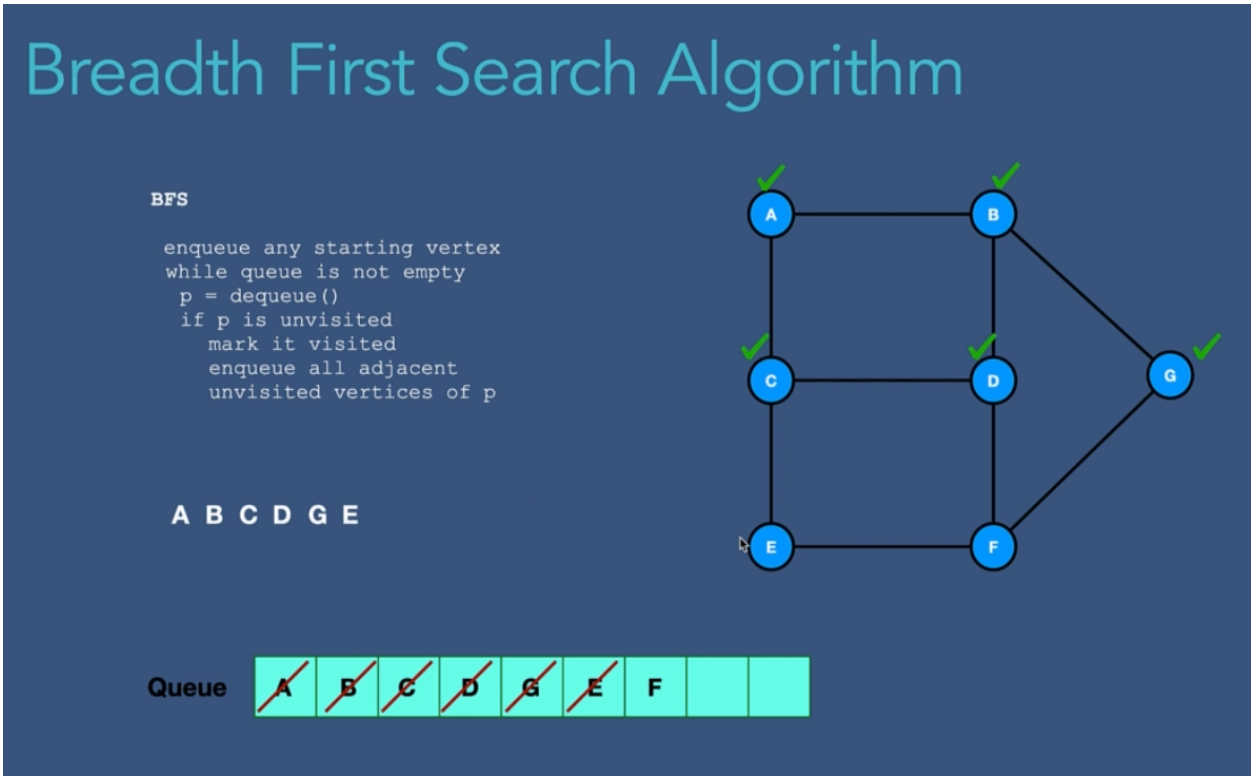
Vertice D is dequeued.



The adjacent of D are B and F.B is already in the Queue. So, we only enqueue F to the Queue.



We then dequeue the elements one by one to get the result.



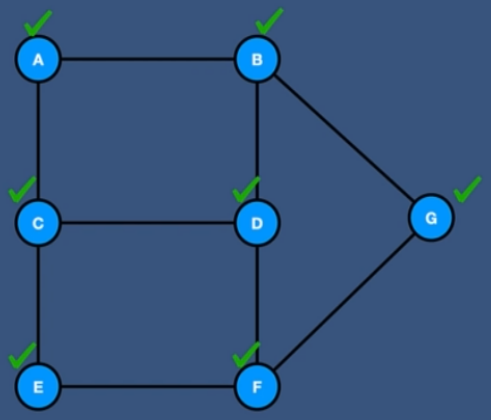
Dequeued E.

Breadth First Search Algorithm

BFS

```
enqueue any starting vertex
while queue is not empty
 p = dequeue()
 if p is unvisited
   mark it visited
   enqueue all adjacent
   unvisited vertices of p
```

A B C D G E F



Queue ~~A~~ ~~B~~ ~~C~~ ~~D~~ ~~G~~ ~~E~~ ~~F~~

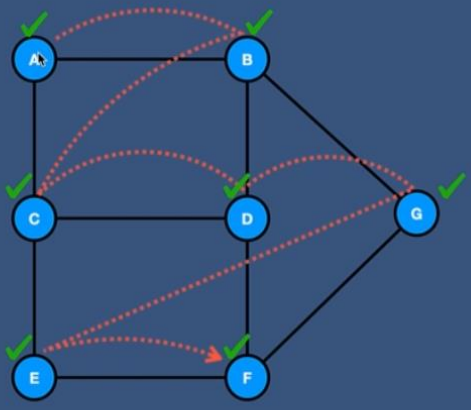
Dequeued F.

Breadth First Search Algorithm

BFS

```
enqueue any starting vertex
while queue is not empty
 p = dequeue()
 if p is unvisited
   mark it visited
   enqueue all adjacent
   unvisited vertices of p
```

A B C D G E F



Queue ~~A~~ ~~B~~ ~~C~~ ~~D~~ ~~G~~ ~~E~~ ~~F~~

Time and Space Complexity of BFS

```
BFS

while all vertices are not explored ..... O(V)
  enqueue any starting vertex ..... O(1)
  while queue is not empty ..... O(V)
   p = dequeue() ..... O(1)
   if p is unvisited ..... O(1)
     mark it visited ..... O(1)
     enqueue unvisited adjacent vertices of p ..... O(Adj)
```

O(Adj) + O(1) + O(1) + O(1) = O(Adj) + O(3) = O(Adj)

O(Adj) + O(V) = O(E)

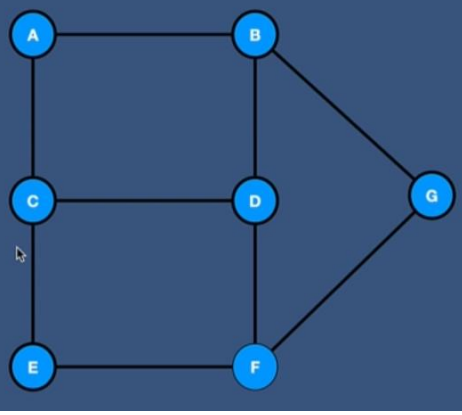
O(E) + O(1) + O(V) = O(E) + O(V) = O(V+E)

Time Complexity : O(V+E)

Space Complexity : O(V+E)

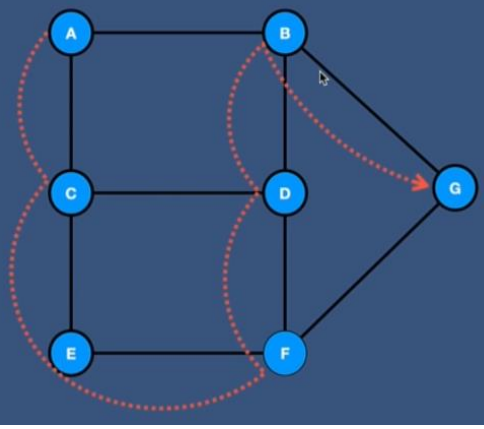
Depth First Search (DFS)

DFS is an algorithm for traversing a graph data structure which starts selecting some arbitrary node and explores as far as possible along each edge before backtracking



Depth First Search (DFS)

DFS is an algorithm for traversing a graph data structure which starts selecting some arbitrary node and explores as far as possible along each edge before backtracking

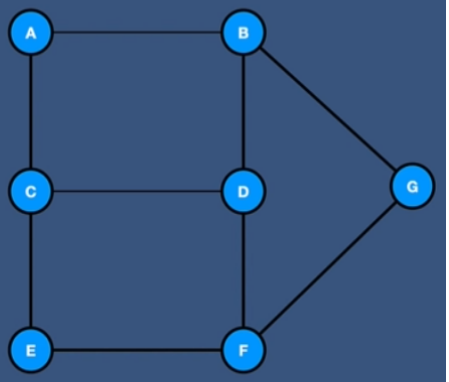


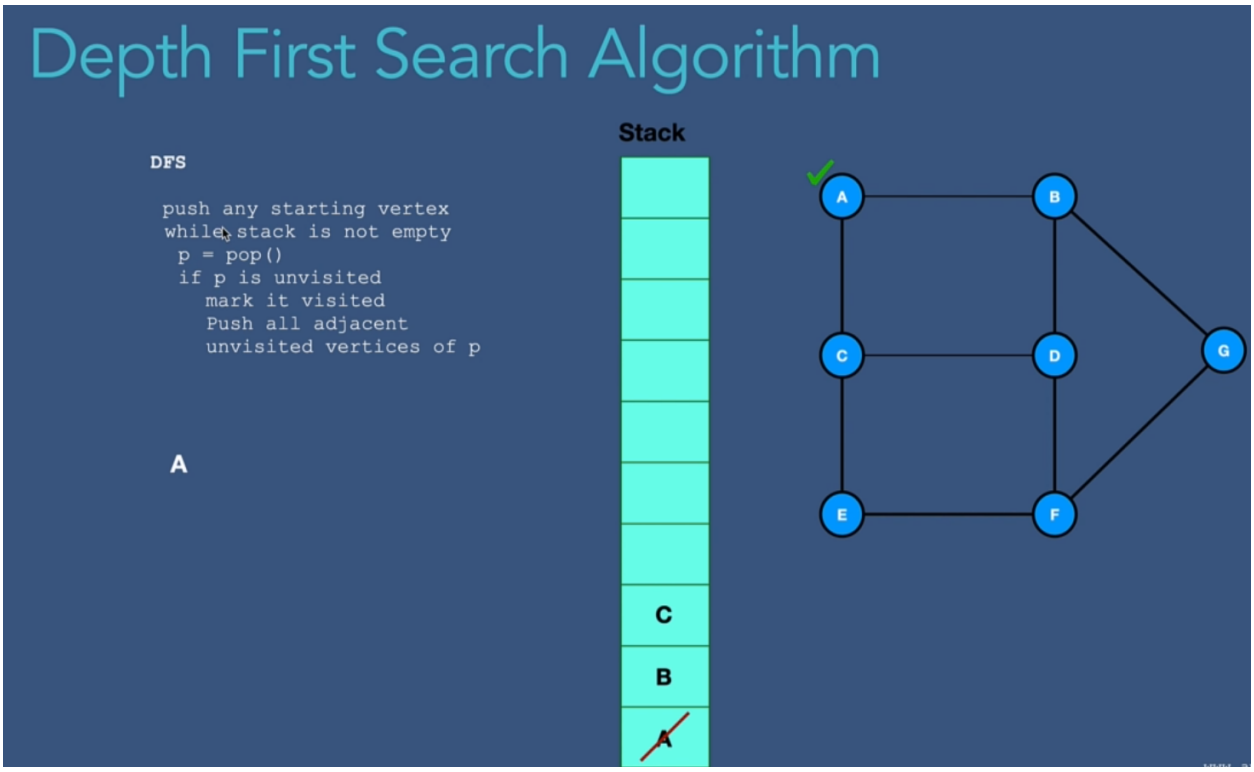
Depth First Search Algorithm

DFS

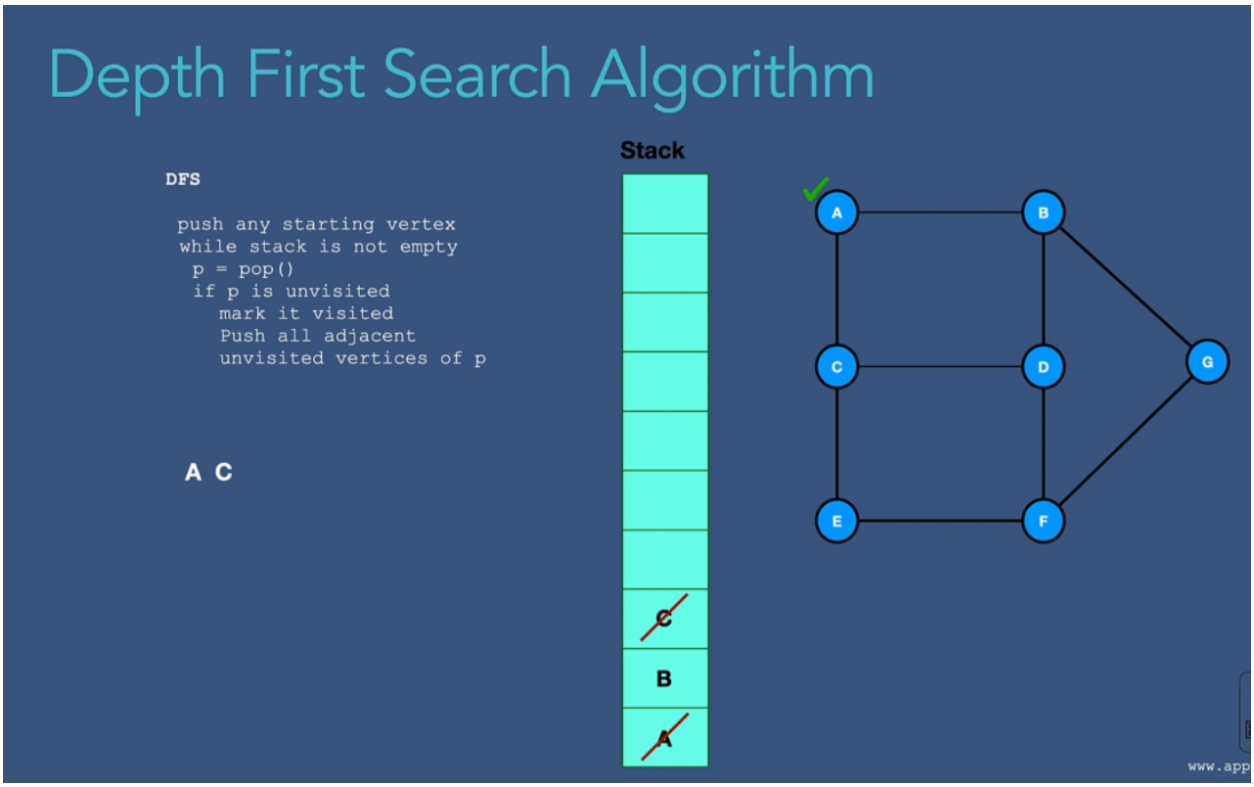
```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
   mark it visited
   Push all adjacent
   unvisited vertices of p
```

Stack





Pushing the adjacent vertices of A (the vertices B and C) to the stack.



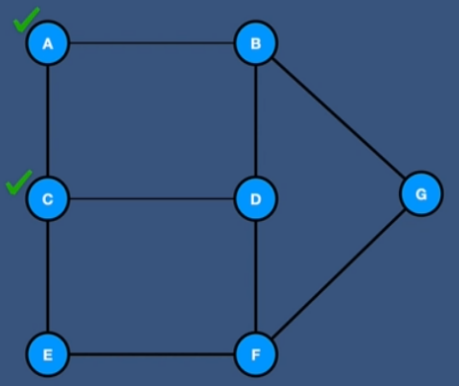
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
    mark it visited
    Push all adjacent
    unvisited vertices of p
```

A C

Stack



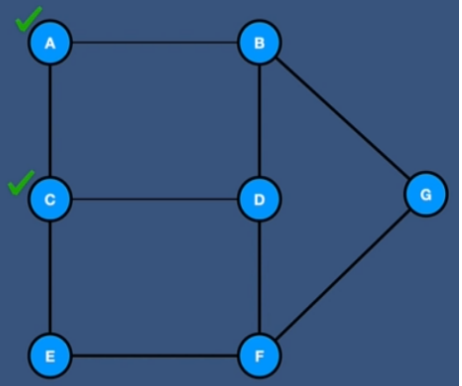
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
    mark it visited
    Push all adjacent
    unvisited vertices of p
```

A C E

Stack



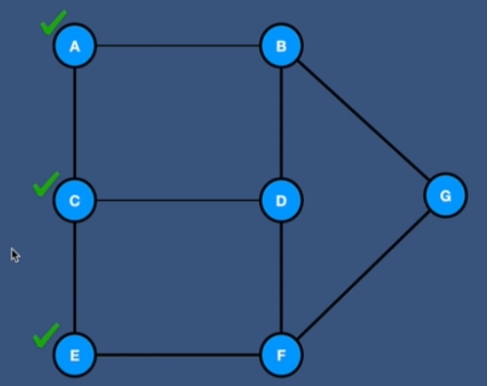
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
   mark it visited
   Push all adjacent
   unvisited vertices of p
```

A C E

Stack



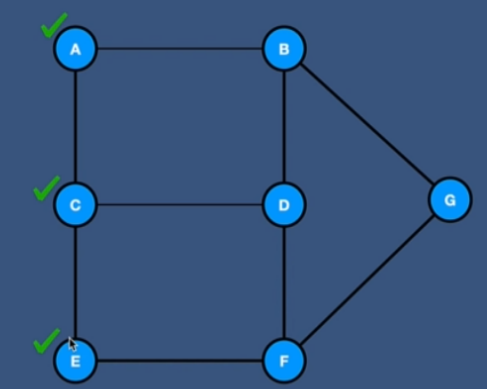
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
   mark it visited
   Push all adjacent
   unvisited vertices of p
```

A C E F

Stack



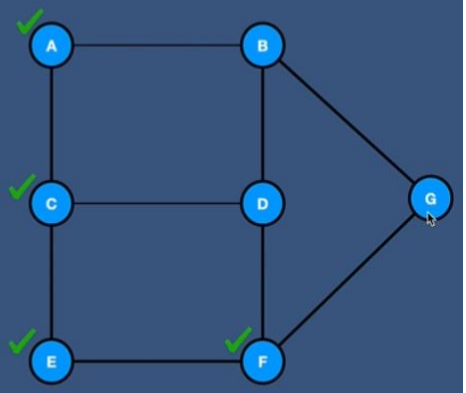
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
   mark it visited
   Push all adjacent
   unvisited vertices of p
```

A C E F

Stack



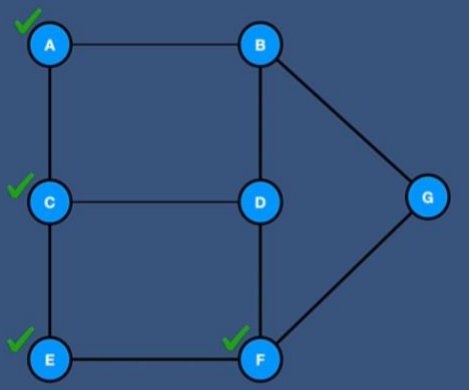
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
   mark it visited
   Push all adjacent
   unvisited vertices of p
```

A C E F D

Stack



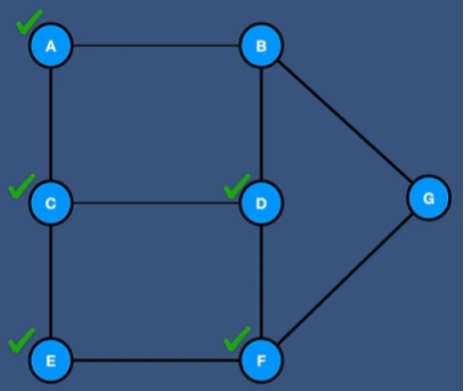
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
   mark it visited
   Push all adjacent
   unvisited vertices of p
```

A C E F D

Stack



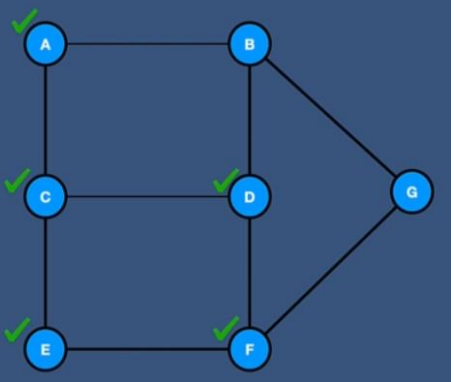
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
   mark it visited
   Push all adjacent
   unvisited vertices of p
```

A C E F D B

Stack



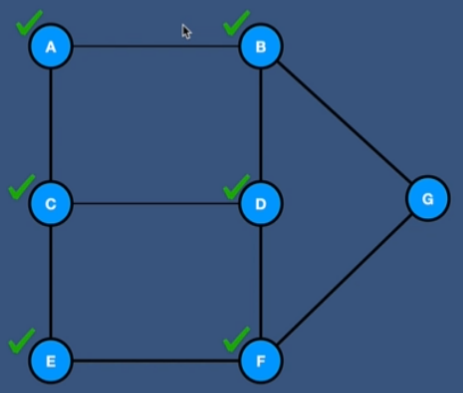
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
   mark it visited
   Push all adjacent
   unvisited vertices of p
```

A C E F D B

Stack



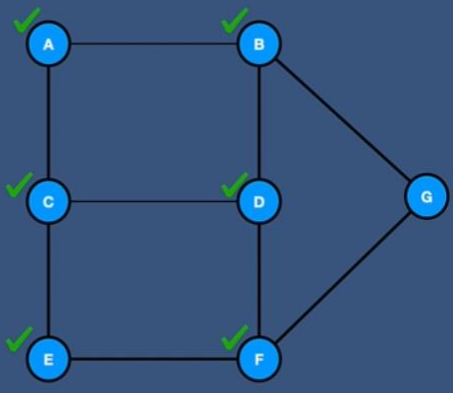
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
 p = pop()
 if p is unvisited
   mark it visited
   Push all adjacent
   unvisited vertices of p
```

A C E F D B G

Stack



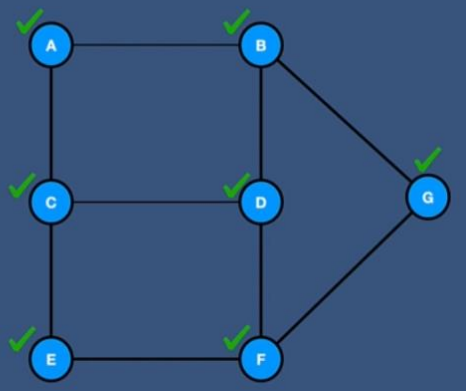
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
  p = pop()
  if p is unvisited
     mark it visited
     Push all adjacent
     unvisited vertices of p
```

A C E F D B G

Stack



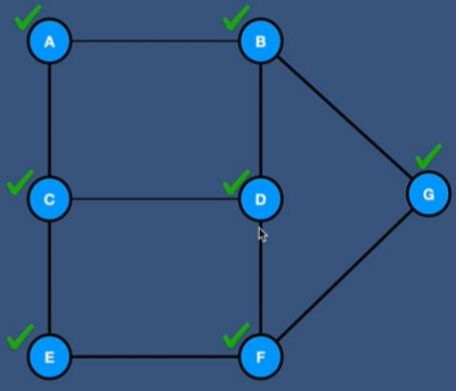
Depth First Search Algorithm

DFS

```
push any starting vertex
while stack is not empty
  p = pop()
  if p is unvisited
     mark it visited
     Push all adjacent
     unvisited vertices of p
```

A C E F D B G

Stack



Time and Space Complexity of DFS

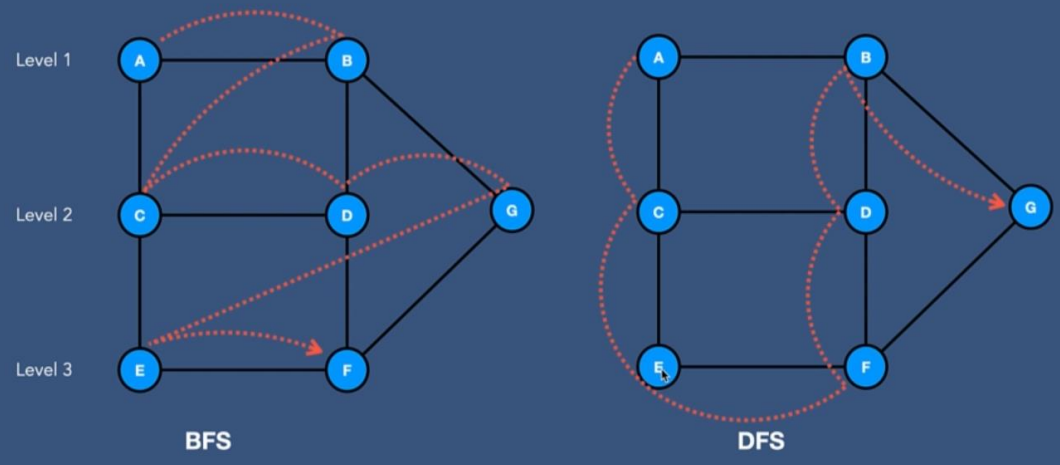
```
DFS

while all vertices are not explored ..... O(V)
  push any starting vertex ..... O(1)
  while stack is not empty ..... O(V)
   p = pop()  ..... O(1)
   if p is unvisited ..... O(1)
     mark it visited ..... O(1)
     push unvisited adjacent vertices of p ..... O(Adj)
```

O(Adj) and O(E)

Time Complexity : O(V+E)

Space Complexity : O(V+E)

BFS vs DFS



BFS vs DFS

	BFS	DFS
How does it work internally?	It goes in breath first	It goes in depth first
Which DS does it use internally?	Queue	Stack
Time Complexity	O(V+E)	O(V+E)
Space Complexity	O(V+E)	O(V+E)
When to use?	If we know that the target is close to the staring point	If we already know that the target vertex is buried very deep