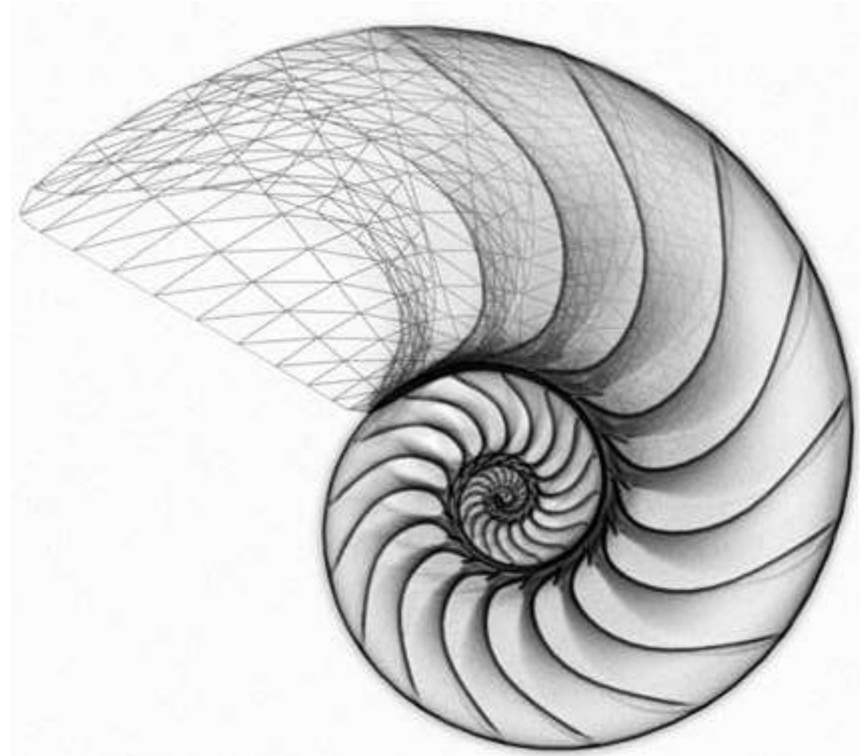


Graphs (Continued...)

CSE225: Data Structures and Algorithms

Depth-First Search

- ***Depth-first search*** is a strategy for exploring a graph and find path between two vertices
  - Explore "deeper" in the graph whenever possible
  - Edges are explored out of the most recently discovered vertex ***v*** that still has unexplored edges
  - When all of ***v***'s edges have been explored, backtrack to the vertex from which ***v*** was discovered

Depth-First Search

Set found to false

stack.Push(startVertex)

do

stack.Pop(vertex)

***if** vertex = endVertex*

Write final vertex

Set found to true

else if vertex is unvisited

Write this vertex

Push all unvisited adjacent vertices onto stack

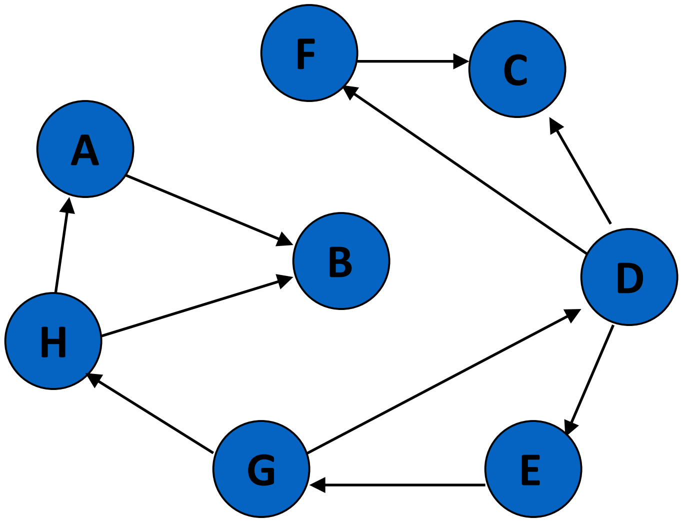
***while** !stack.IsEmpty() AND !found*

if(!found)

Write "Path does not exist"

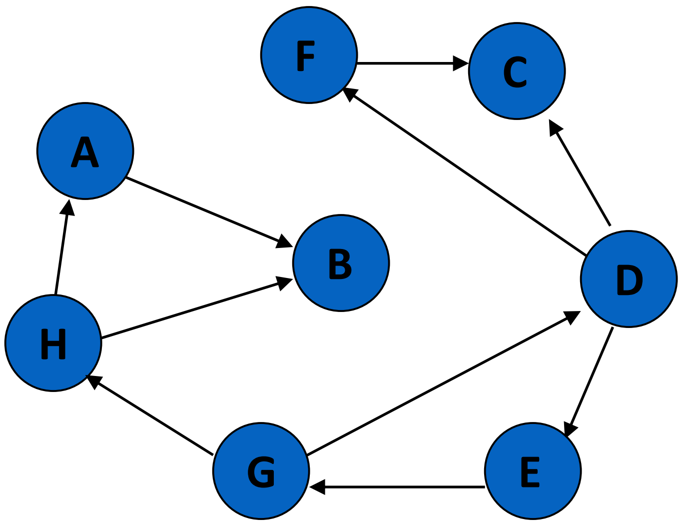
Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B

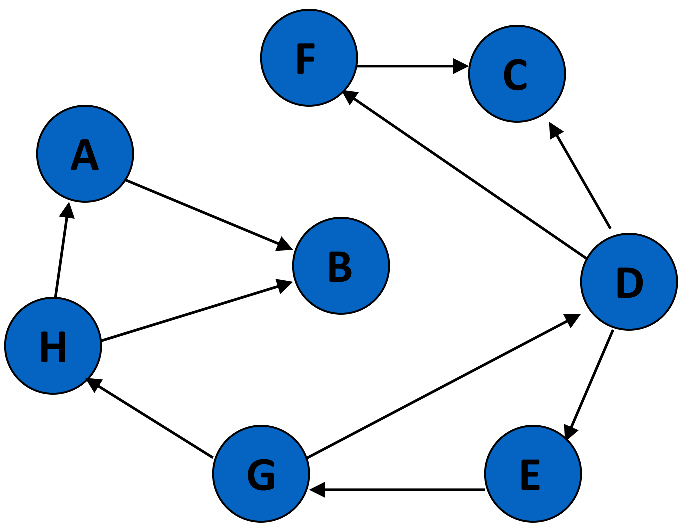


Visited nodes:

	vertices		marks	stack	found
[0]	A	[0]			
[1]	B	[1]			
[2]	C	[2]			
[3]	D	[3]			
[4]	E	[4]			
[5]	F	[5]			
[6]	G	[6]			
[7]	H	[7]			

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



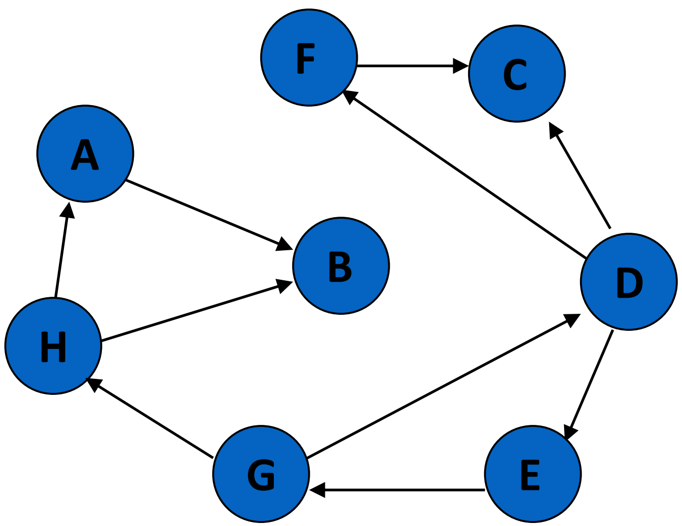
	vertices		marks	stack	found
[0]	A	[0]			
[1]	B	[1]			
[2]	C	[2]			
[3]	D	[3]			
[4]	E	[4]			
[5]	F	[5]			
[6]	G	[6]			
[7]	H	[7]			

Visited nodes:

Clear the marks (set to false). Push D onto the stack. Set *found* to false.

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



Visited nodes:

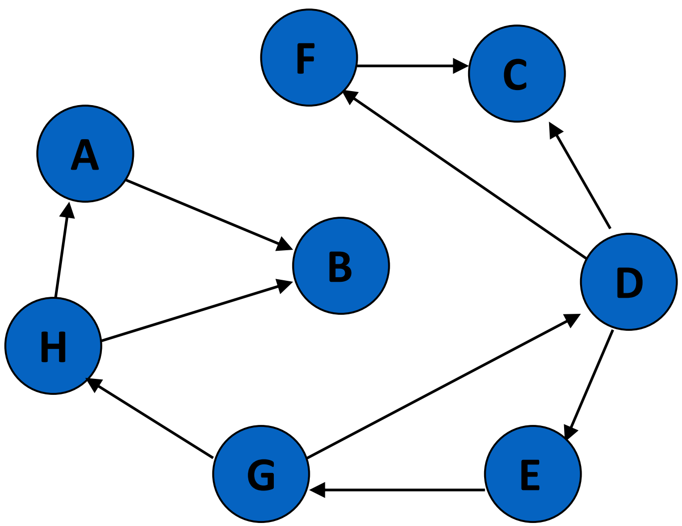
	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	0	
[4]	E	[4]	0	
[5]	F	[5]	0	
[6]	G	[6]	0	
[7]	H	[7]	0	D

found
false

Clear the marks (set to false). Push D onto the stack. Set *found* to false.

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	0	
[4]	E	[4]	0	
[5]	F	[5]	0	
[6]	G	[6]	0	
[7]	H	[7]	0	D

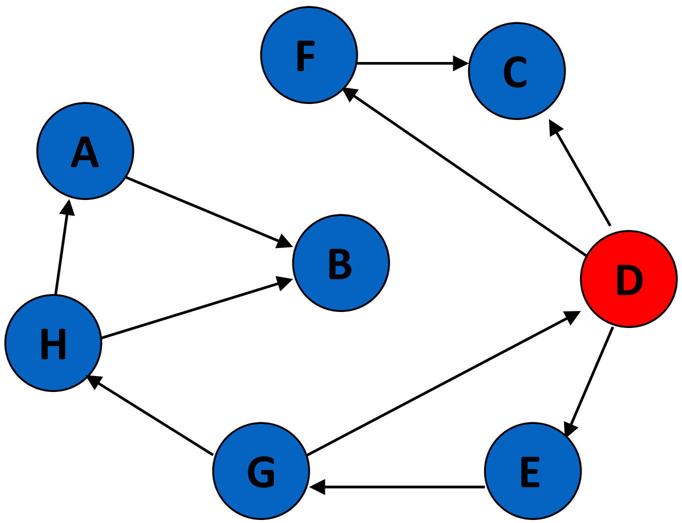
found
false

Visited nodes:

Pop from stack (D is popped). D is not visited yet (unmarked). So, visit D (set D as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	0	
[6]	G	[6]	0	
[7]	H	[7]	0	

found

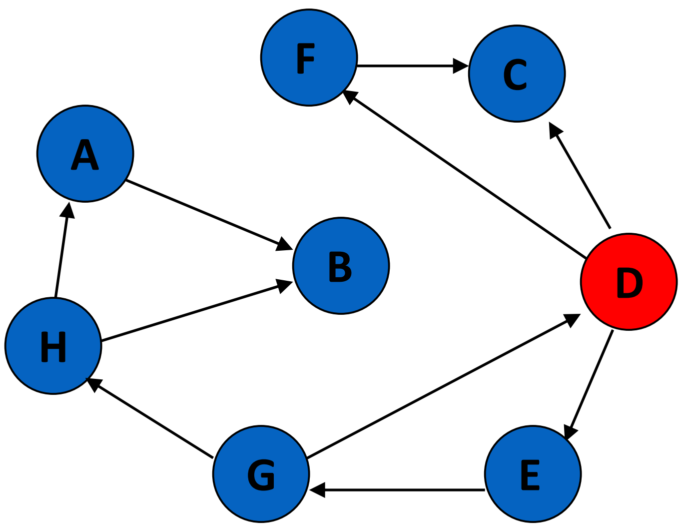
false

Visited nodes:
D

Pop from stack (D is popped). D is not visited yet (unmarked). So, visit D (set D as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	0	
[6]	G	[6]	0	
[7]	H	[7]	0	

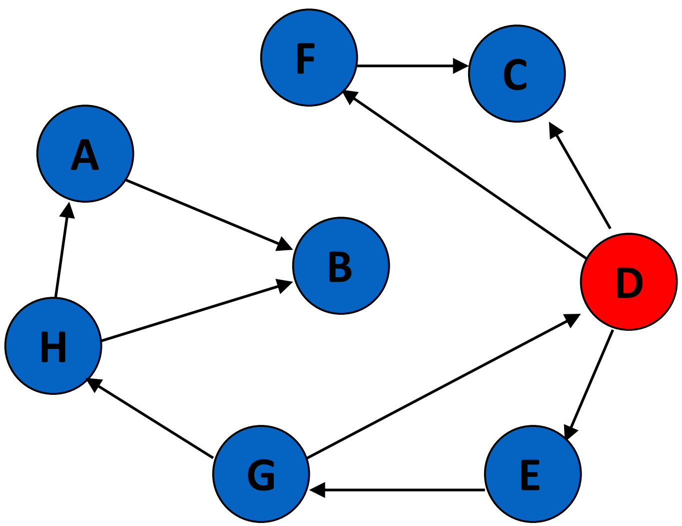
found
false

Visited nodes:
D

Push all the vertices that are adjacent to D and unvisited (unmarked) onto the stack (C, E and F are pushed).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	0	F
[6]	G	[6]	0	E
[7]	H	[7]	0	C

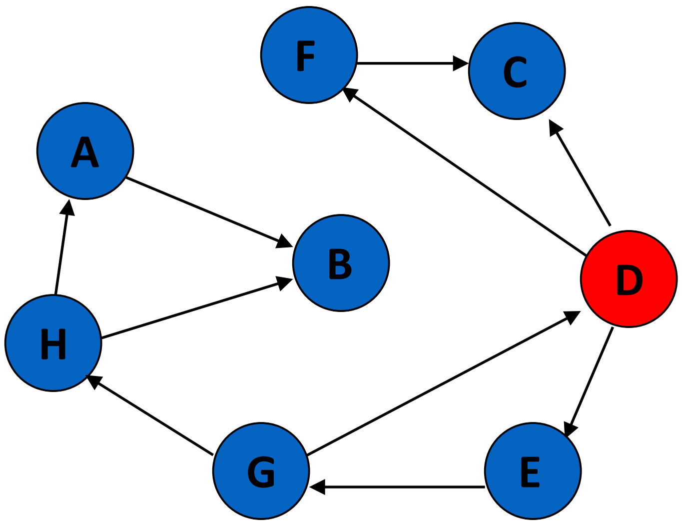
found
false

Visited nodes:
D

Push all the vertices that are adjacent to D and unvisited (unmarked) onto the stack (C, E and F are pushed).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	0	F
[6]	G	[6]	0	E
[7]	H	[7]	0	C

found

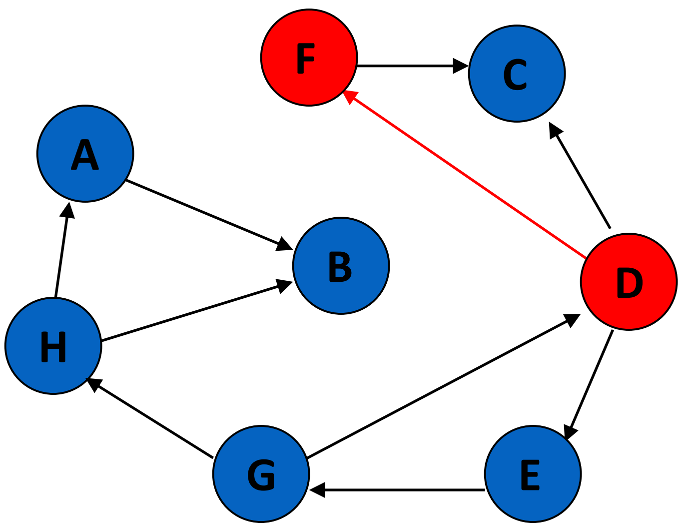
false

Visited nodes:
D

Pop from stack (F is popped). F is not visited yet (unmarked). So, visit F (set F as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



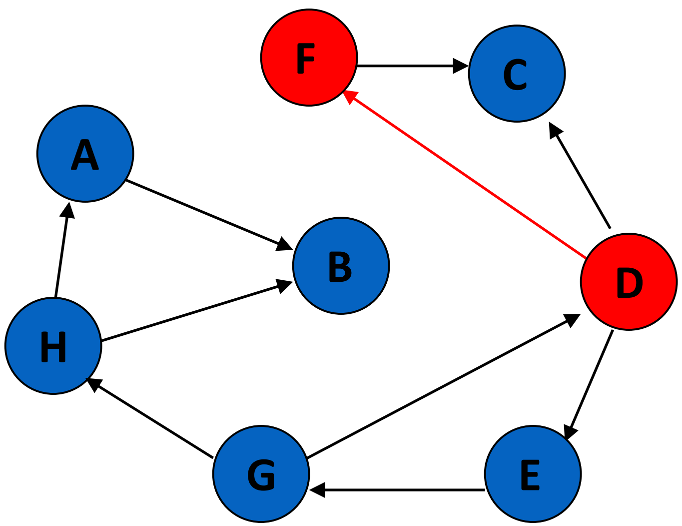
	vertices		marks	stack	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	0		
[3]	D	[3]	1		
[4]	E	[4]	0		
[5]	F	[5]	1		
[6]	G	[6]	0	E	
[7]	H	[7]	0	C	

Visited nodes:
D F

Pop from stack (F is popped). F is not visited yet (unmarked). So, visit F (set F as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	1	
[6]	G	[6]	0	E
[7]	H	[7]	0	C

found

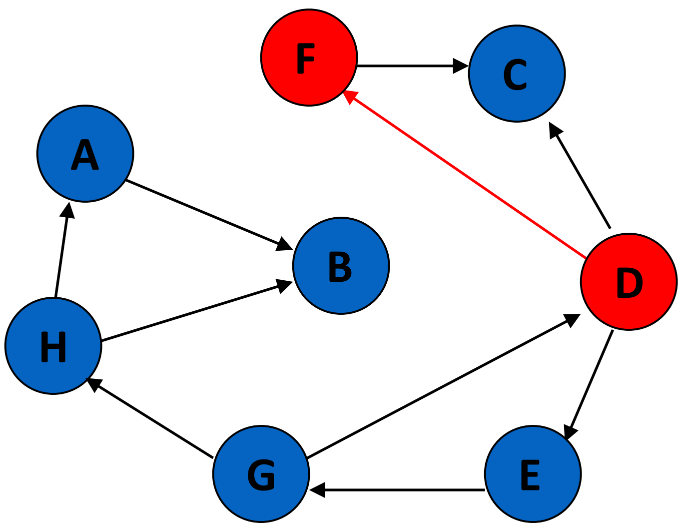
false

Visited nodes:
D F

Push all the vertices that are adjacent to F and unvisited (unmarked) onto the stack (C is pushed).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	1	C
[6]	G	[6]	0	E
[7]	H	[7]	0	C

found
false

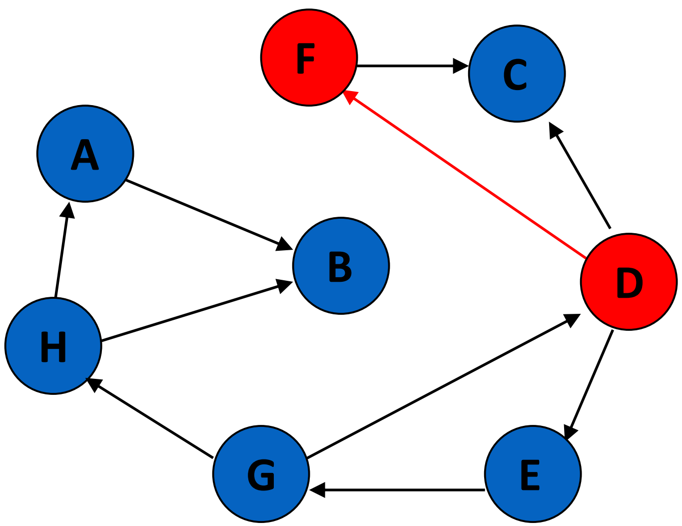
Visited nodes:

D F

Push all the vertices that are adjacent to F and unvisited (unmarked) onto the stack (C is pushed).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	1	C
[6]	G	[6]	0	E
[7]	H	[7]	0	C

found

false

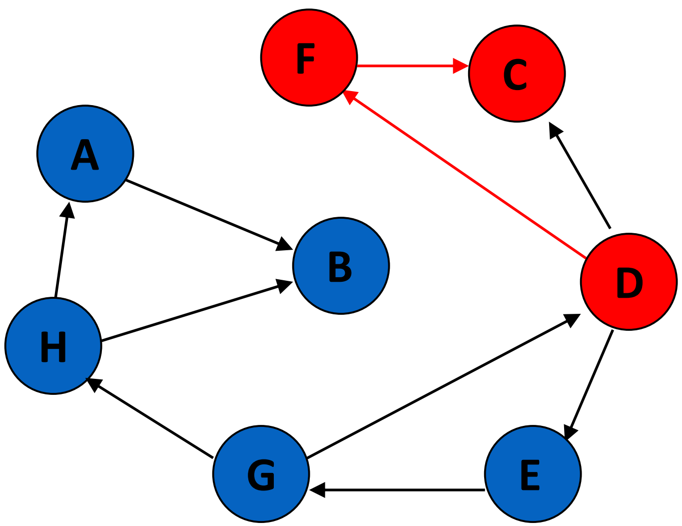
Visited nodes:

D F

Pop from stack (C is popped). C is not visited yet (unmarked). So, visit C (set C as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	1	
[6]	G	[6]	0	E
[7]	H	[7]	0	C

found

false

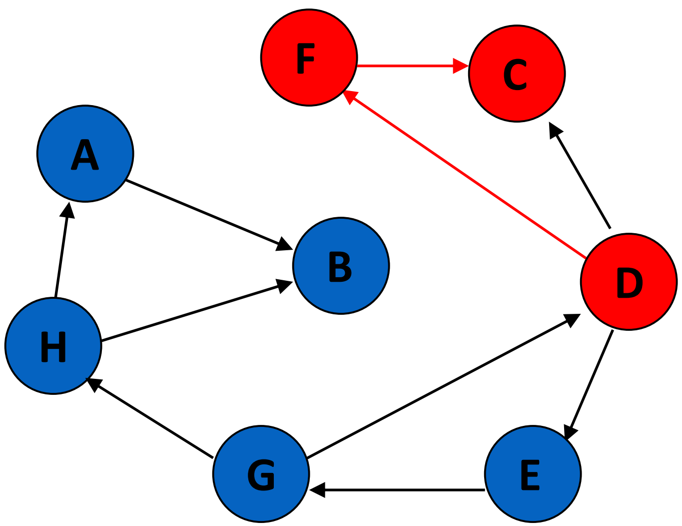
Visited nodes:

D F C

Pop from stack (C is popped). C is not visited yet (unmarked). So, visit C (set C as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	1		
[3]	D	[3]	1		
[4]	E	[4]	0		
[5]	F	[5]	1		
[6]	G	[6]	0	E	
[7]	H	[7]	0	C	

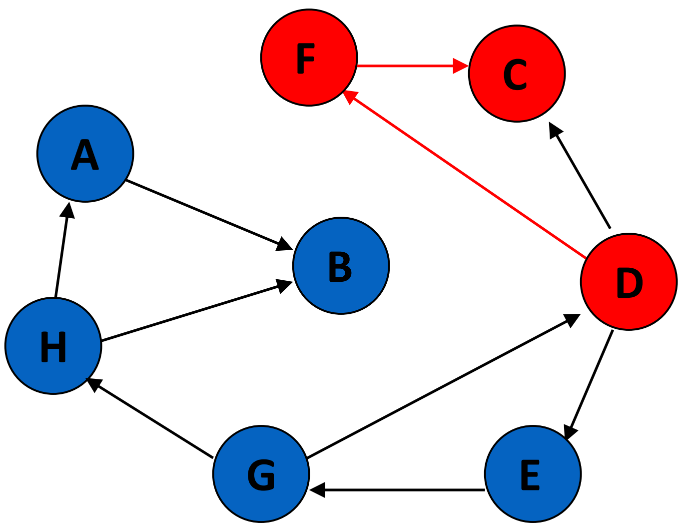
Visited nodes:

D F C

Push all the vertices that are adjacent to C and unvisited (unmarked) onto the stack (nothing is pushed).

Depth-First Search

Example: **Conduct a depth-first search in the graph and find if there is a path from D to B**



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	1	
[6]	G	[6]	0	E
[7]	H	[7]	0	C

found

false

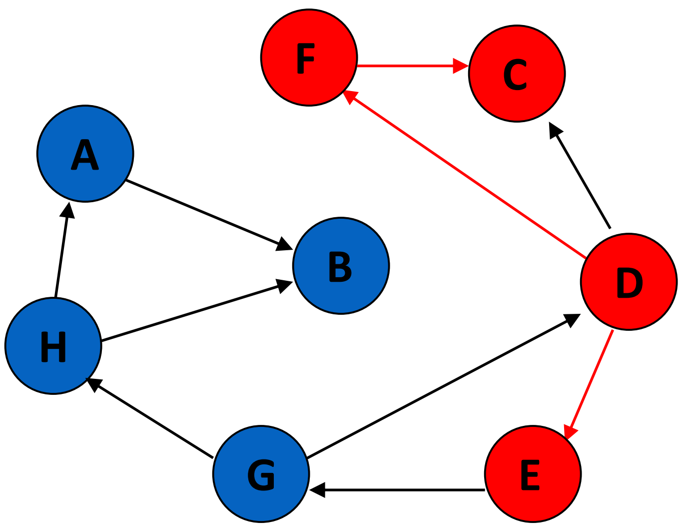
Visited nodes:

D F C

Pop from stack (E is popped). E is not visited yet (unmarked). So, visit E (set E as marked).

Depth-First Search

Example: **Conduct a depth-first search in the graph and find if there is a path from D to B**



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	0	
[7]	H	[7]	0	C

found

false

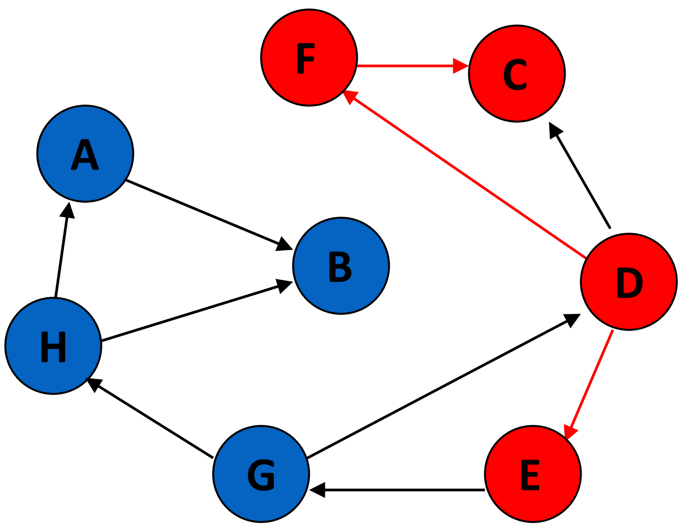
Visited nodes:

D F C E

Pop from stack (E is popped). E is not visited yet (unmarked). So, visit E (set E as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



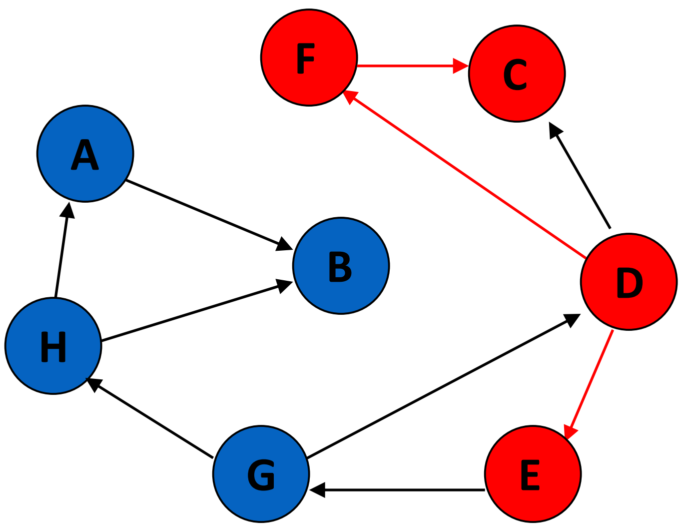
	vertices		marks	stack	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	1		
[3]	D	[3]	1		
[4]	E	[4]	1		
[5]	F	[5]	1		
[6]	G	[6]	0		
[7]	H	[7]	0	C	

Visited nodes:
D F C E

Push all the vertices that are adjacent to E and unvisited (unmarked) onto the stack (G is pushed).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	1		
[3]	D	[3]	1		
[4]	E	[4]	1		
[5]	F	[5]	1		
[6]	G	[6]	0	G	
[7]	H	[7]	0	C	

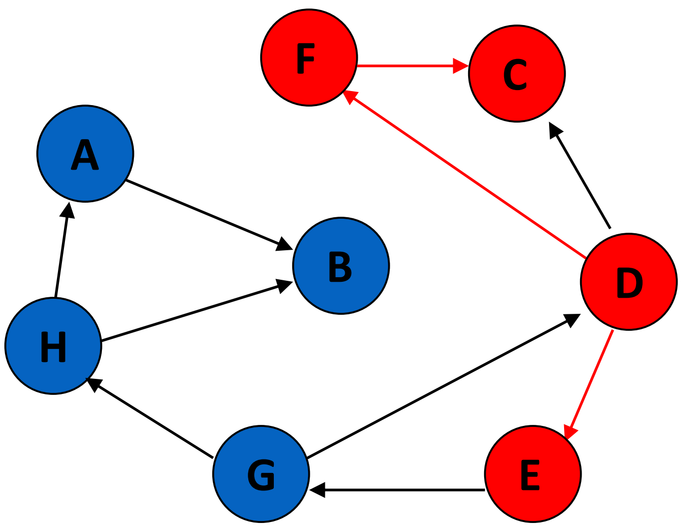
Visited nodes:

D F C E

Push all the vertices that are adjacent to E and unvisited (unmarked) onto the stack (G is pushed).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	0	G
[7]	H	[7]	0	C

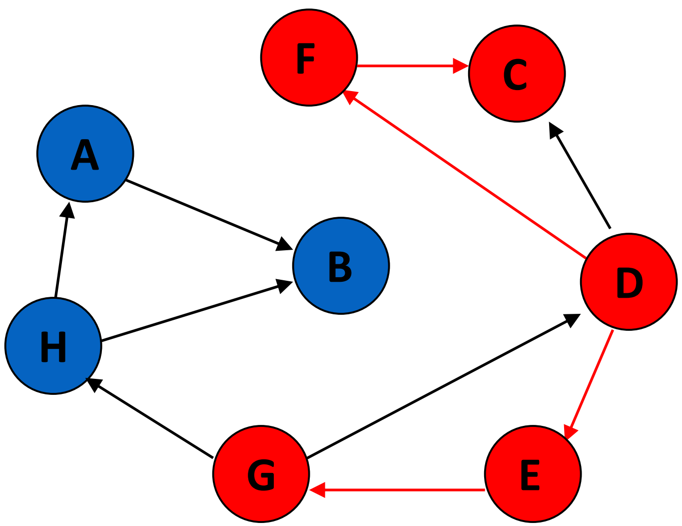
found
false

Visited nodes:
D F C E

Pop from stack (G is popped). G is not visited yet (unmarked). So, visit G (set G as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



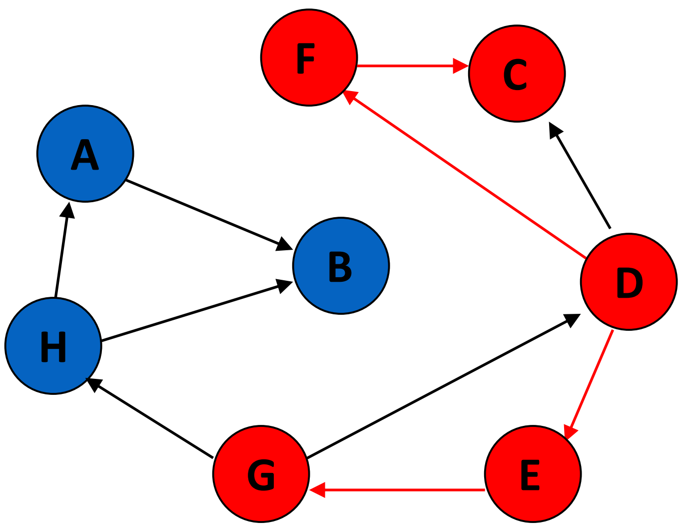
	vertices		marks	stack	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	1		
[3]	D	[3]	1		
[4]	E	[4]	1		
[5]	F	[5]	1		
[6]	G	[6]	1		
[7]	H	[7]	0	C	

Visited nodes:
D F C E G

Pop from stack (G is popped). G is not visited yet (unmarked). So, visit G (set G as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	0	C

found

false

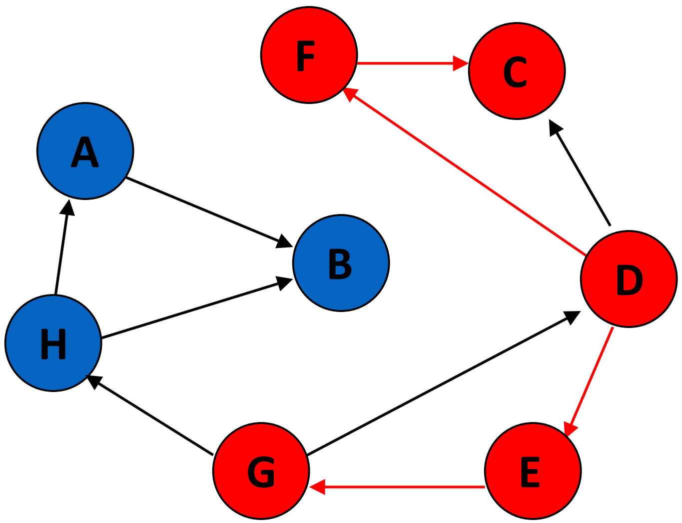
Visited nodes:

D F C E G

Push all the vertices that are adjacent to G and unvisited (unmarked) onto the stack (H is pushed).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	H
[7]	H	[7]	0	C

found

false

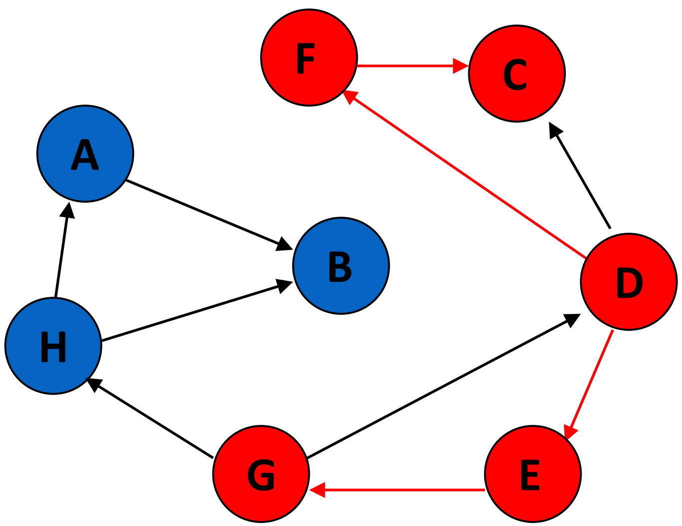
Visited nodes:

D F C E G

Push all the vertices that are adjacent to G and unvisited (unmarked) onto the stack (H is pushed).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	H
[7]	H	[7]	0	C

found

false

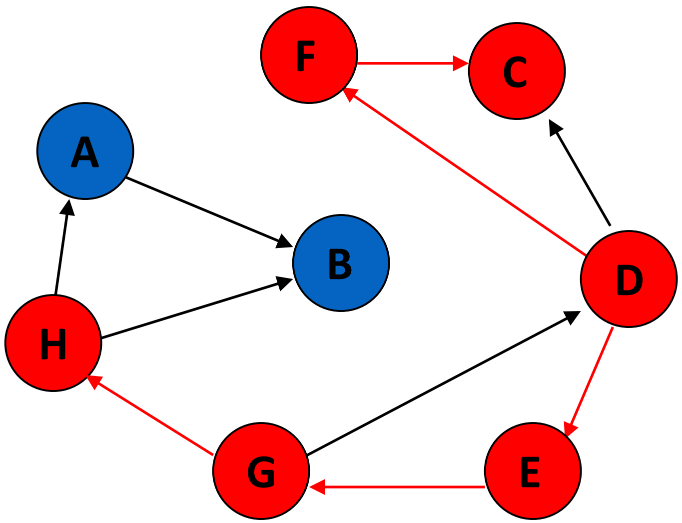
Visited nodes:

D F C E G

Pop from stack (H is popped). H is not visited yet (unmarked). So, visit H (set H as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	1	C

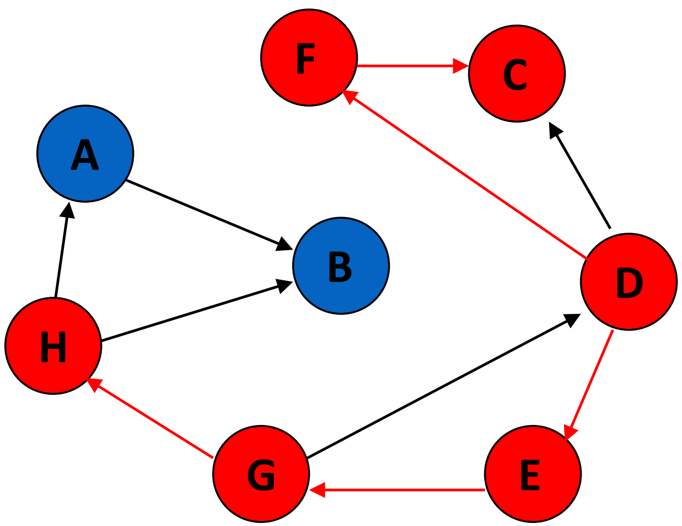
found
false

Visited nodes:
D F C E G H

Pop from stack (H is popped). H is not visited yet (unmarked). So, visit H (set H as marked).

Depth-First Search

Example: **Conduct a depth-first search in the graph and find if there is a path from D to B**



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	1	C

found

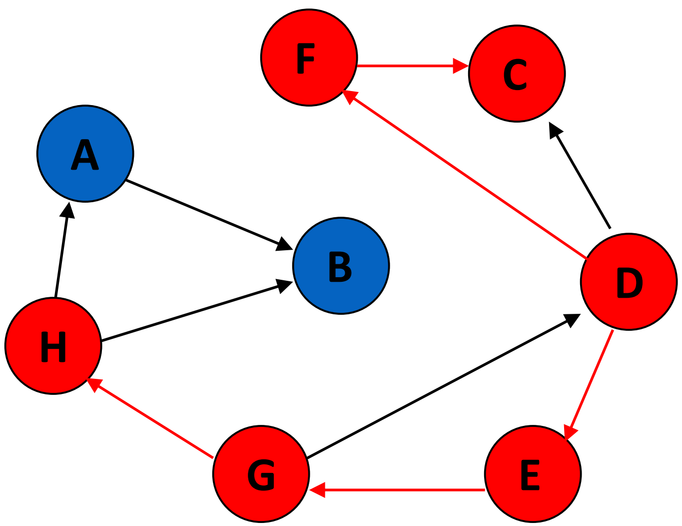
false

Visited nodes:
D F C E G H

Push all the vertices that are adjacent to H and unvisited (unmarked) onto the stack (A and B are pushed).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	B
[6]	G	[6]	1	A
[7]	H	[7]	1	C

found

false

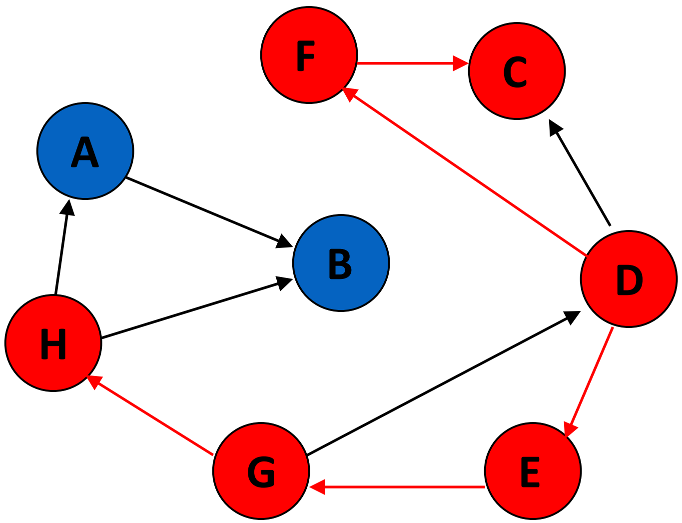
Visited nodes:

D F C E G H

Push all the vertices that are adjacent to H and unvisited (unmarked) onto the stack (A and B are pushed).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	B
[6]	G	[6]	1	A
[7]	H	[7]	1	C

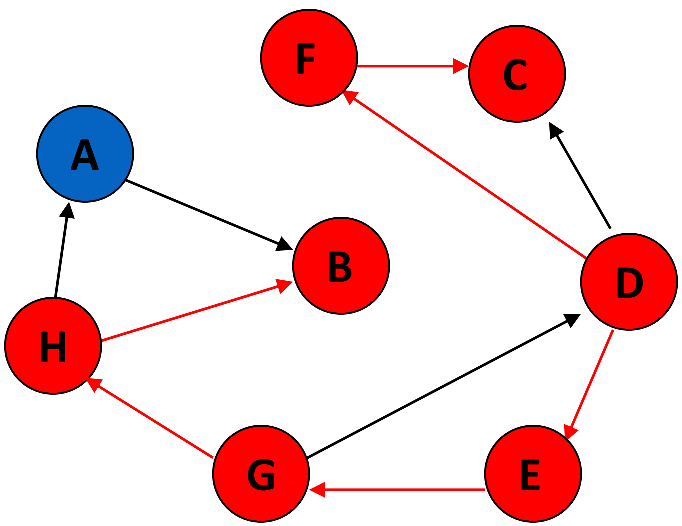
found
false

Visited nodes:
D F C E G H

Pop from stack (B is popped). B is not visited yet (unmarked). So, visit B (set B as marked).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	1	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	A
[7]	H	[7]	1	C

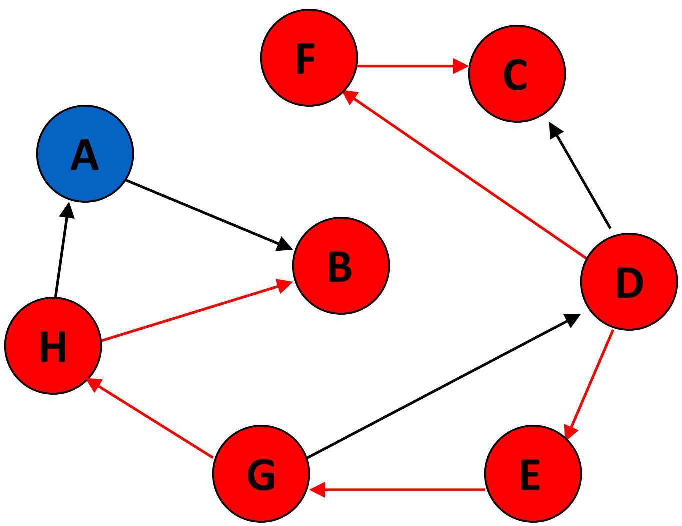
found
false

Visited nodes:
D F C E G H B

Pop from stack (B is popped). B is not visited yet (unmarked). So, visit B (set B as marked).

Depth-First Search

Example: **Conduct a depth-first search in the graph and find if there is a path from D to B**



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	1	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	A
[7]	H	[7]	1	C

found
false

Visited nodes:

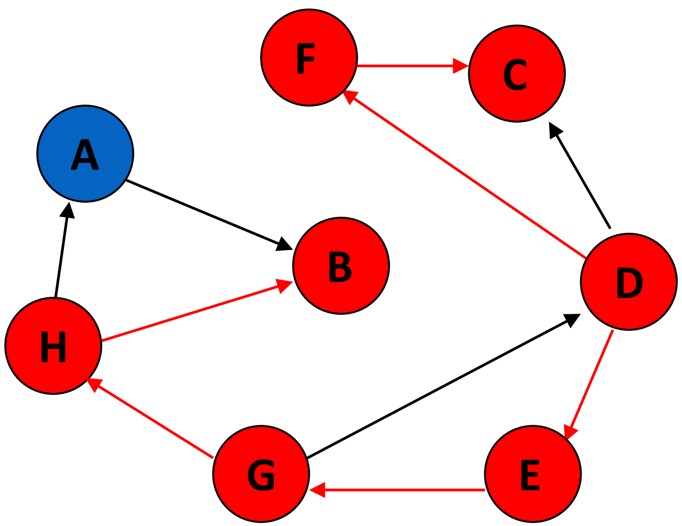
D F C E G H B

B is the destination vertex. So set *found* to true (there is a path). Search is complete.

Note: We can still carry on with the search (until the stack is empty).

Depth-First Search

Example: Conduct a depth-first search in the graph and find if there is a path from D to B



	vertices		marks	stack
[0]	A	[0]	0	
[1]	B	[1]	1	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	A
[7]	H	[7]	1	C

found: true

Visited nodes:

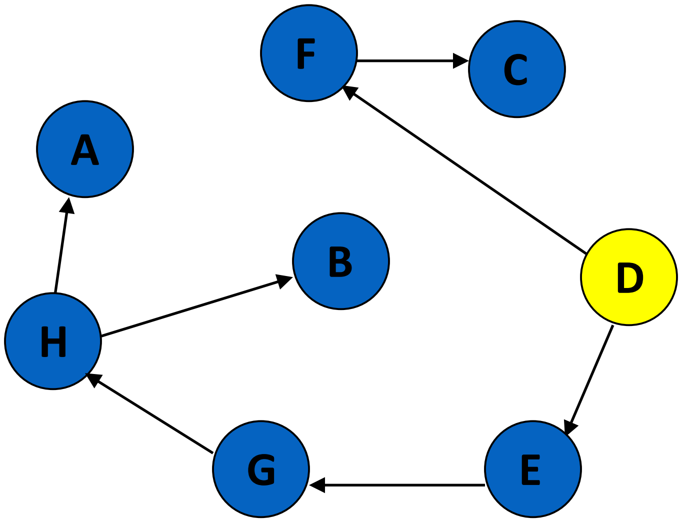
D F C E G H B

B is the destination vertex. So set *found* to true (there is a path). Search is complete.

Note: We can still carry on with the search (until the stack is empty).

Depth-First Search

Depth-First Search yields a tree (the path taken during the search) also known as the Depth-First Tree.



Depth-First Search

```
template<class VertexType>
void DepthFirstSearch(GraphType<VertexType> graph,
     VertexType startVertex, VertexType endVertex)
{
  StackType<VertexType> stack;
  QueType<VertexType> vertexQ;

  bool found = false;
  VertexType vertex;
  VertexType item;

  graph.ClearMarks();
  stack.Push(startVertex);
  do
  {
    stack.Pop(vertex);
    if (vertex == endVertex)
    {
           cout << vertex);
           found = true;
    }
```

Depth-First Search

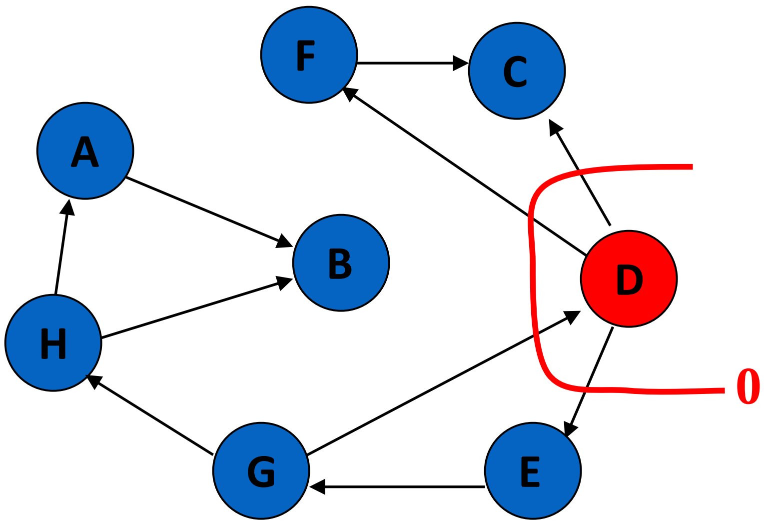
```
    else
    {
        if (!graph.IsMarked(vertex))
        {
            graph.MarkVertex(vertex);
            cout << vertex;
            graph.GetToVertices(vertex, vertexQ);

            while (!vertexQ.IsEmpty())
            {
                vertexQ.Dequeue(item);
                if (!graph.IsMarked(item))
                stack.Push(item);
            }
        }
    }
  } while (!stack.IsEmpty() && !found);
  if (!found)
  cout << "Path not found." << endl;
}
```

Breadth-First Search

- ***Breadth-first search*** is a strategy for exploring a graph and find path between two vertices
  - Explore "level by level" in the graph

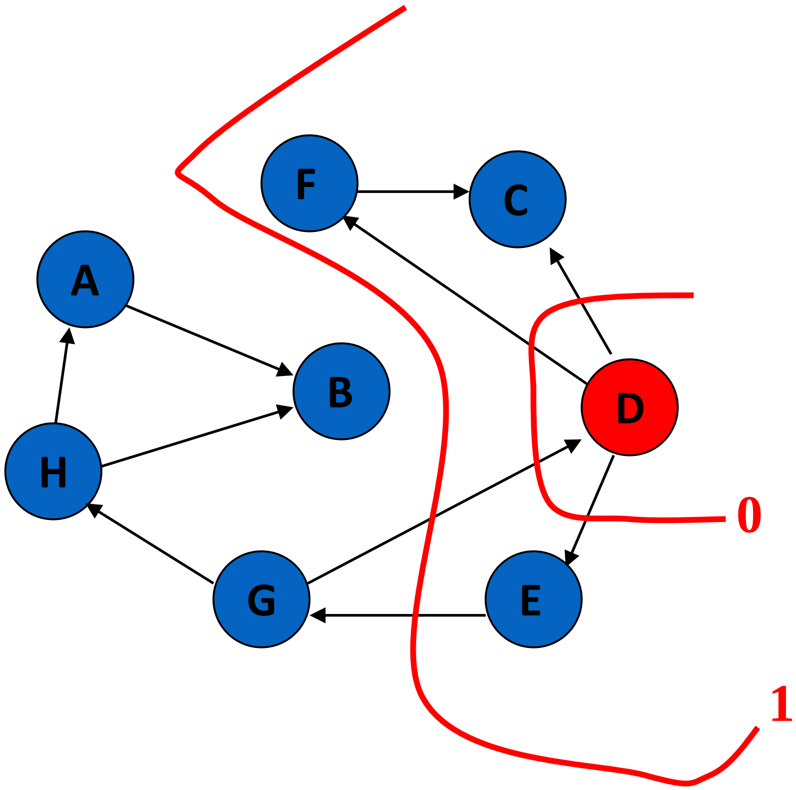
Breadth-First Search



Breadth-first search starts with given node

Task: Conduct a breadth-first search of the graph starting with node D

Breadth-First Search



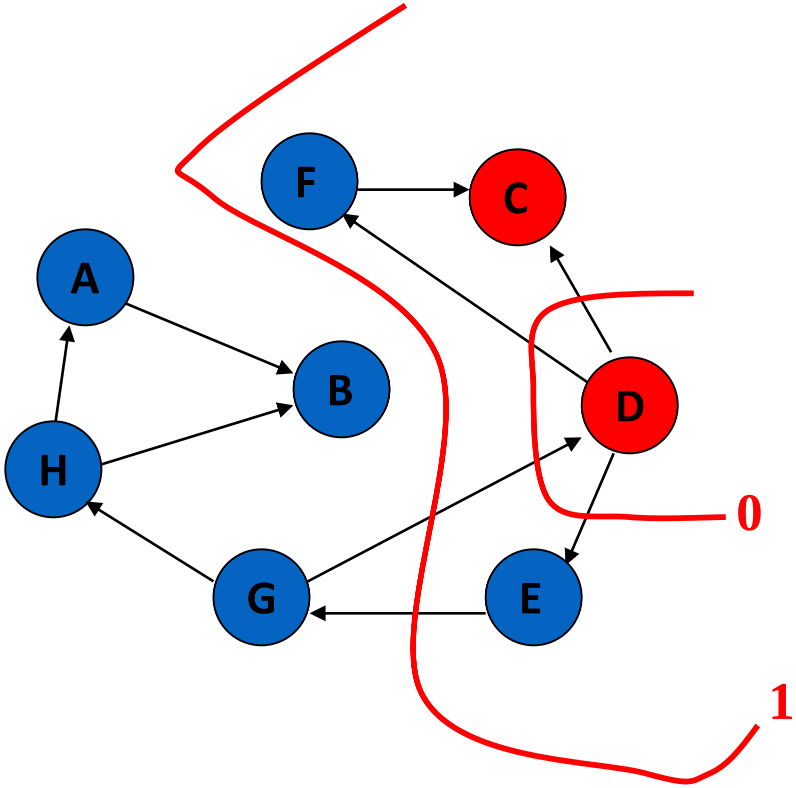
Breadth-first search starts with given node

Then visits nodes adjacent in some specified order (e.g., alphabetical)

Like ripples in a pond

Nodes visited: D

Breadth-First Search



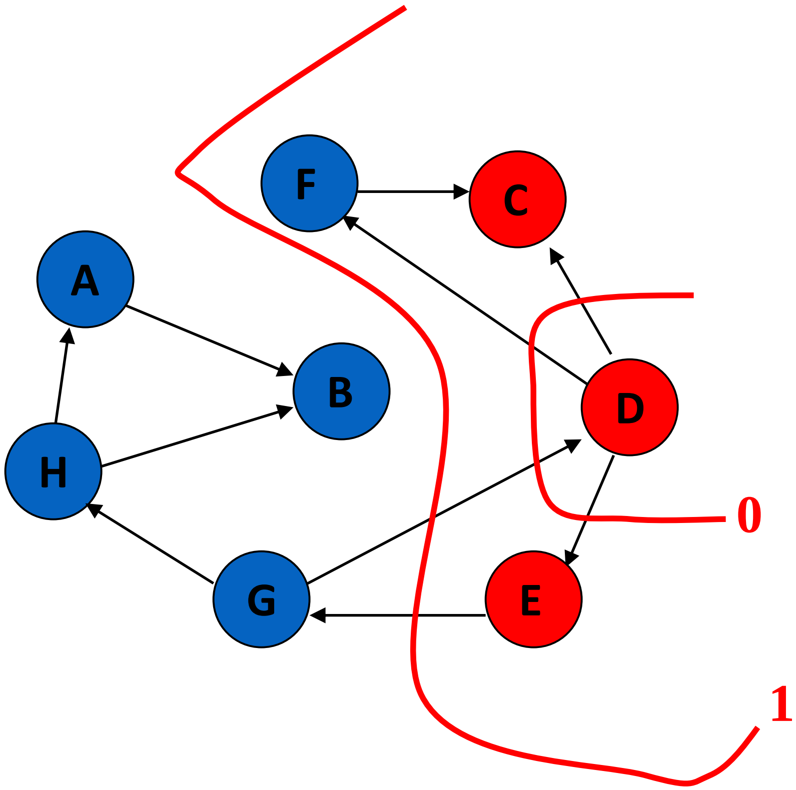
Breadth-first search starts with given node

Then visits nodes adjacent in some specified order (e.g., alphabetical)

Like ripples in a pond

Nodes visited: D, C

Breadth-First Search



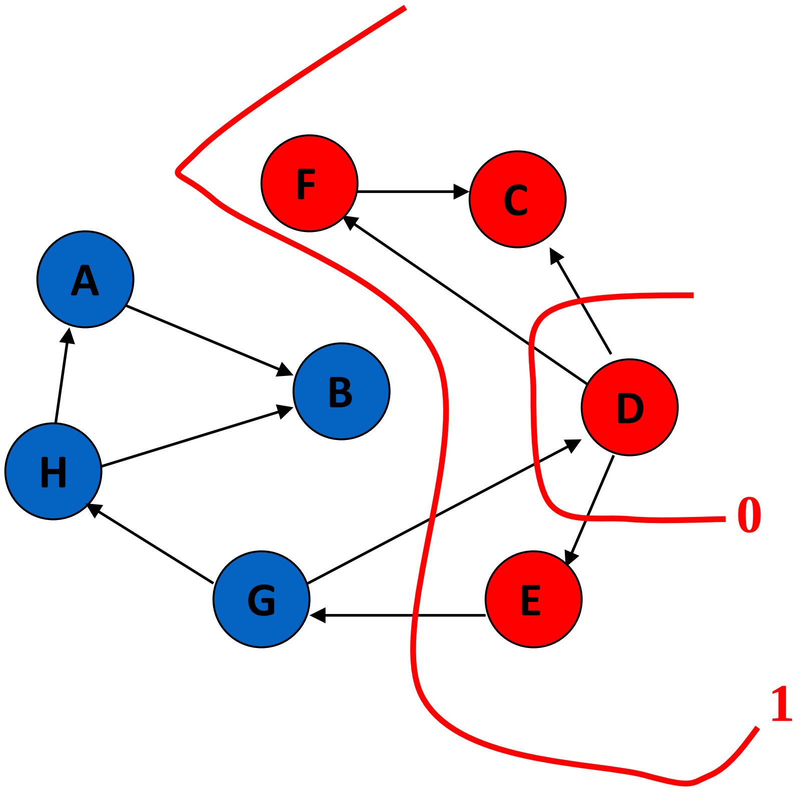
Breadth-first search starts with given node

Then visits nodes adjacent in some specified order (e.g., alphabetical)

Like ripples in a pond

Nodes visited: D, C, E

Breadth-First Search



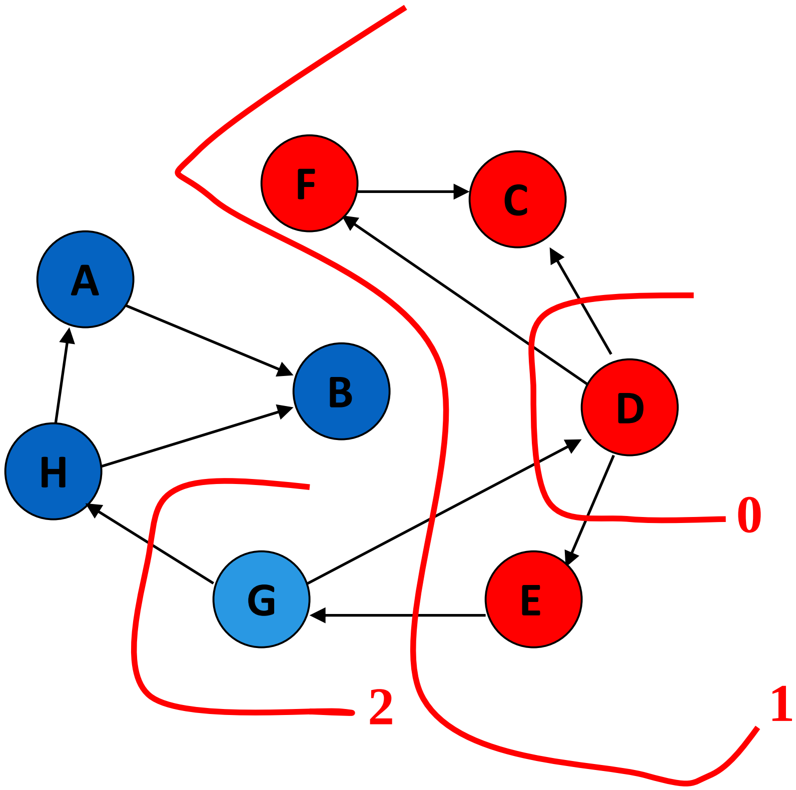
Breadth-first search starts with given node

Then visits nodes adjacent in some specified order (e.g., alphabetical)

Like ripples in a pond

Nodes visited: D, C, E, F

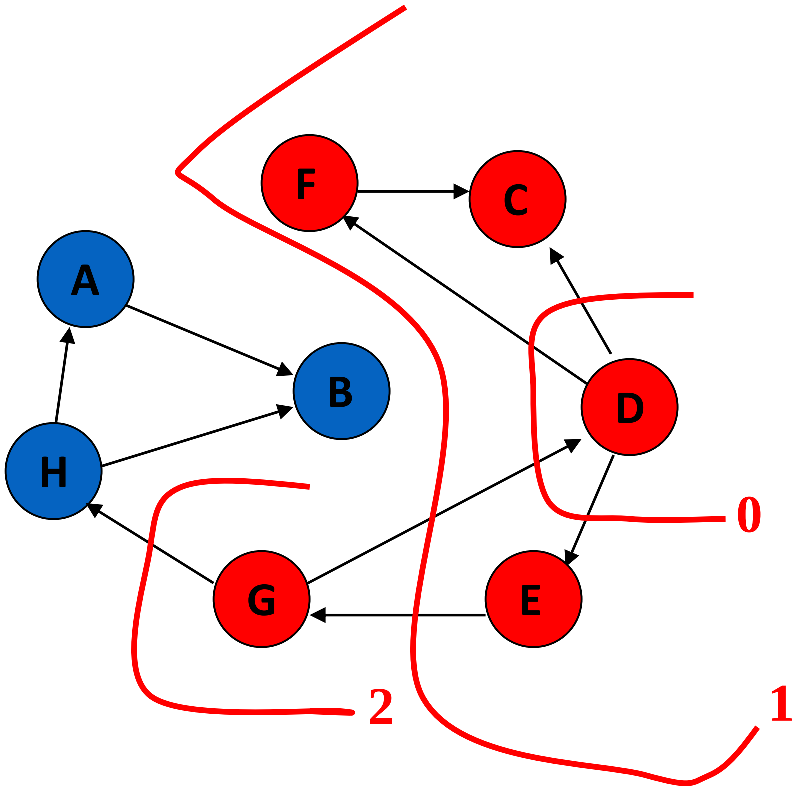
Breadth-First Search



When all nodes in ripple are visited, visit nodes in next ripples

Nodes visited: D, C, E, F

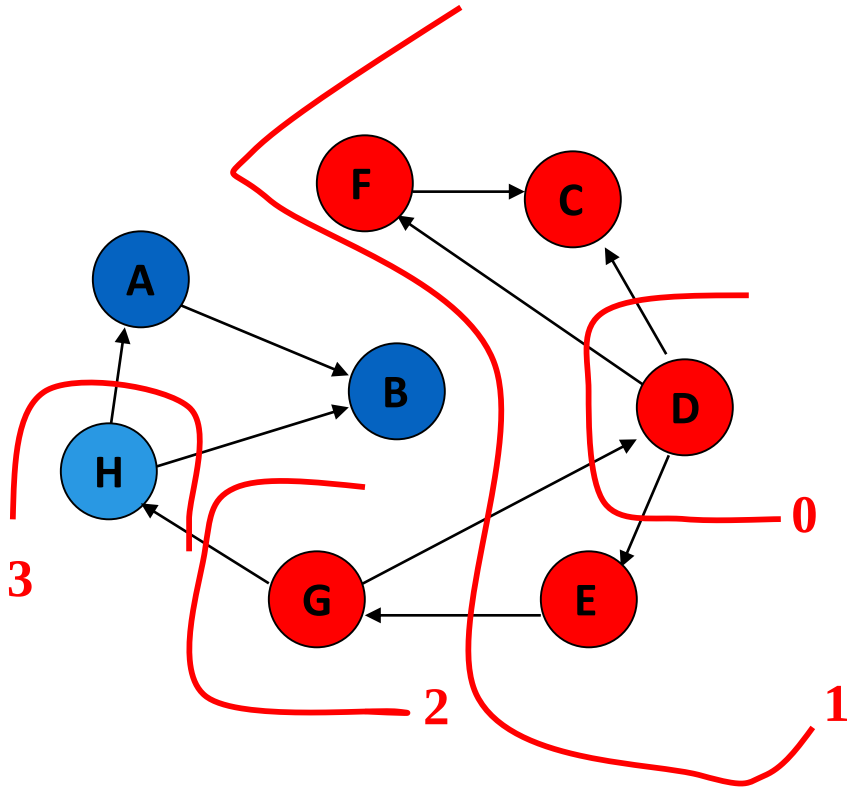
Breadth-First Search



When all nodes in ripple are visited, visit nodes in next ripples

Nodes visited: D, C, E, F, G

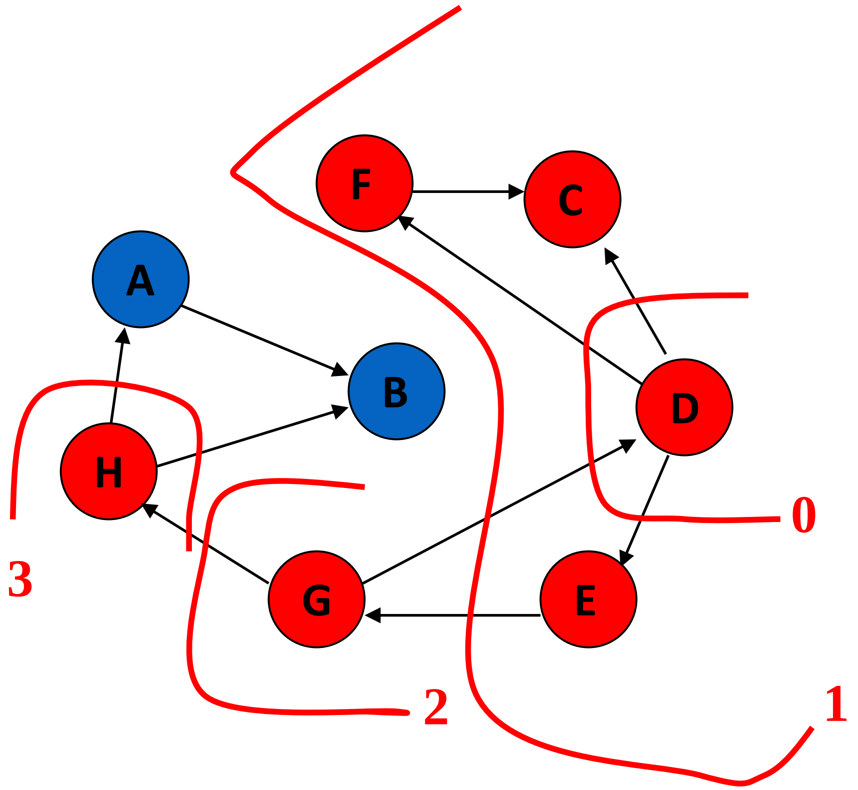
Breadth-First Search



When all nodes in ripple are visited, visit nodes in next ripples

Nodes visited: D, C, E, F, G

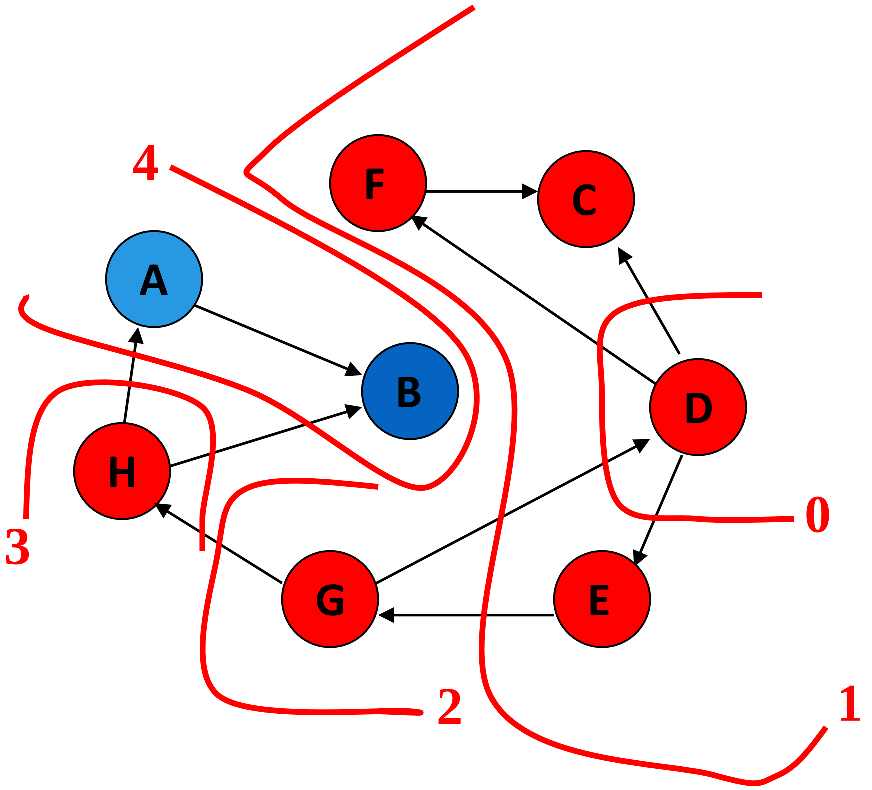
Breadth-First Search



When all nodes in ripple are visited, visit nodes in next ripples

Nodes visited: D, C, E, F, G, H

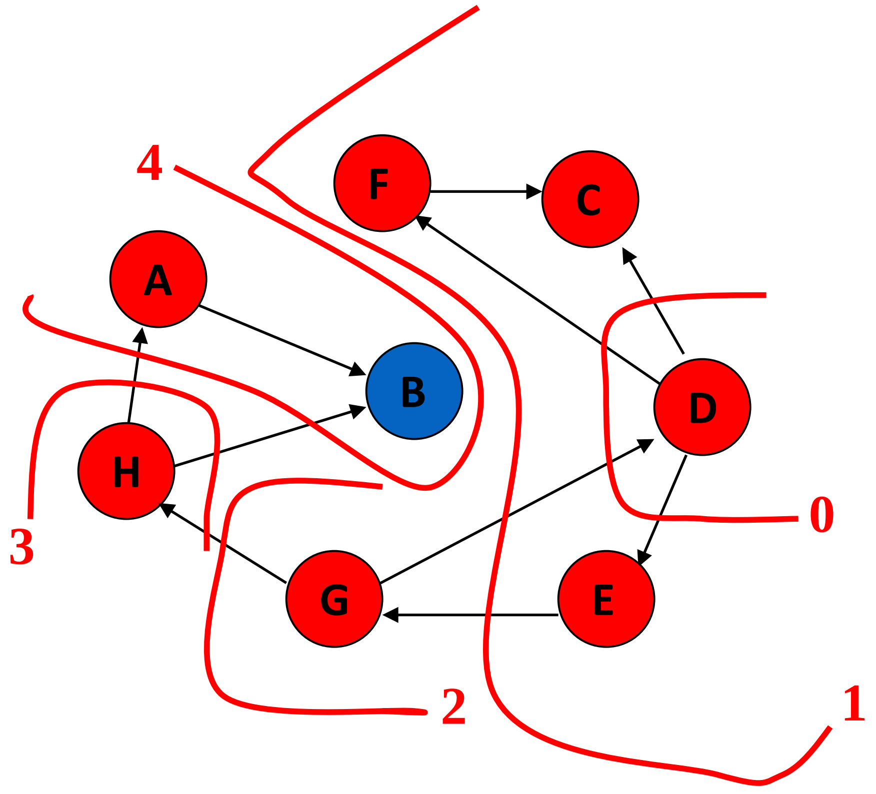
Breadth-First Search



When all nodes in ripple are visited, visit nodes in next ripples

Nodes visited: D, C, E, F, G, H

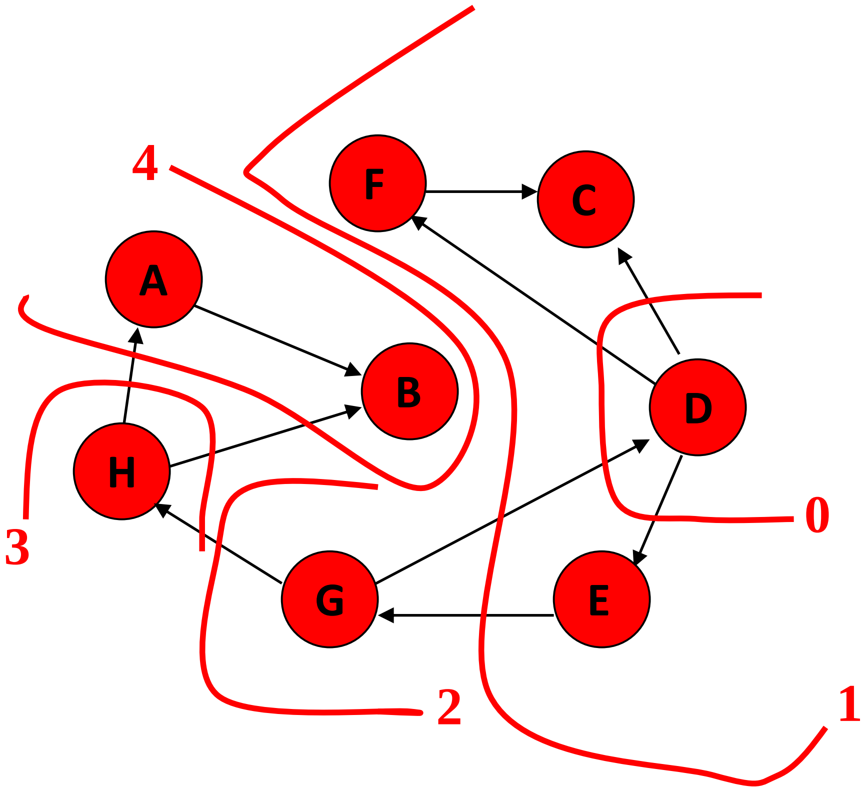
Breadth-First Search



When all nodes in ripple are visited, visit nodes in next ripples

Nodes visited: D, C, E, F, G, H, A

Breadth-First Search



When all nodes in ripple are visited, visit nodes in next ripples

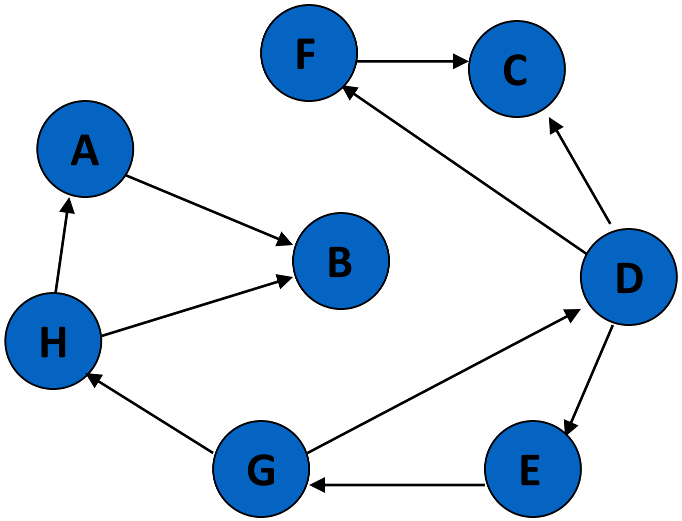
Nodes visited: D, C, E, F, G, H, A, B

Breadth-First Search

```
Set found to false
queue.Enqueue(startVertex)
do
   queue.Dequeue (vertex)
   if vertex = endVertex
      Write final vertex
      Set found to true
   else if vertex is unvisited
      Write this vertex
      Enqueue all unvisited adjacent vertices onto queue
while !queue.IsEmpty() AND !found
if(!found)
      Write "Path does not exist"
```

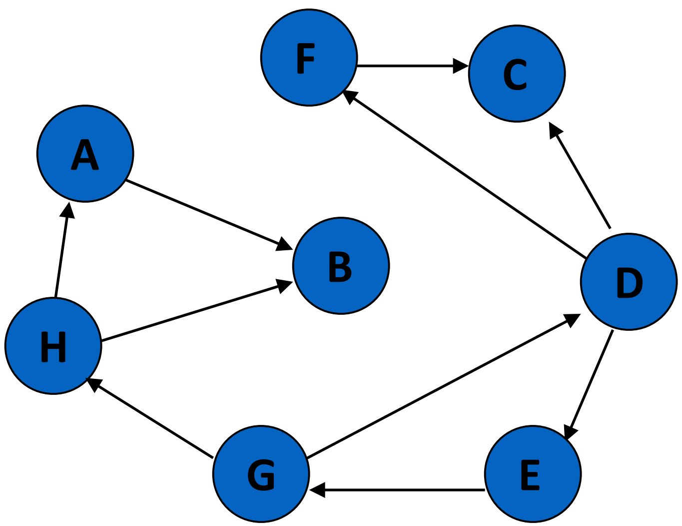
Breadth-First Search

Example: **Conduct a breadth-first search in the graph and find if there is a path from D to B**



Breadth-First Search

Example: **Conduct a breadth-first search in the graph and find if there is a path from D to B**



Visited nodes:

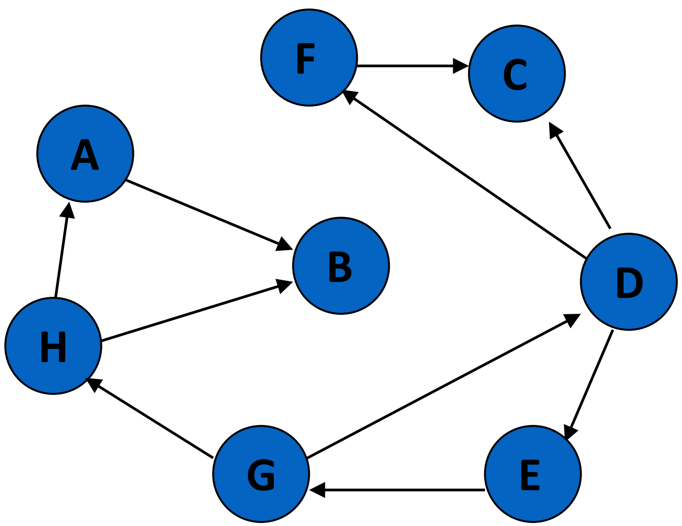
	vertices		marks	queue
[0]	A	[0]		
[1]	B	[1]		
[2]	C	[2]		
[3]	D	[3]		
[4]	E	[4]		
[5]	F	[5]		
[6]	G	[6]		
[7]	H	[7]		

found

Clear the marks (set to false). Enqueue D. Set *found* to false.

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



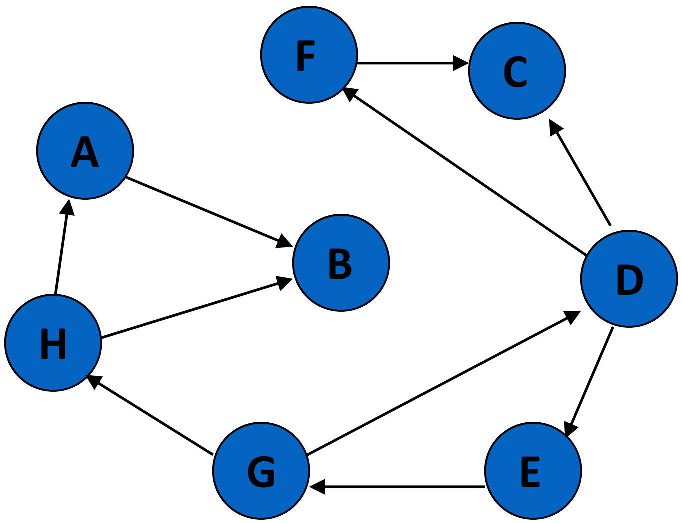
	vertices		marks	queue	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	0		
[3]	D	[3]	0		
[4]	E	[4]	0		
[5]	F	[5]	0		
[6]	G	[6]	0		
[7]	H	[7]	0	D	

Visited nodes:

Clear the marks (set to false). Enqueue D. Set *found* to false.

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	0	
[4]	E	[4]	0	
[5]	F	[5]	0	
[6]	G	[6]	0	
[7]	H	[7]	0	D

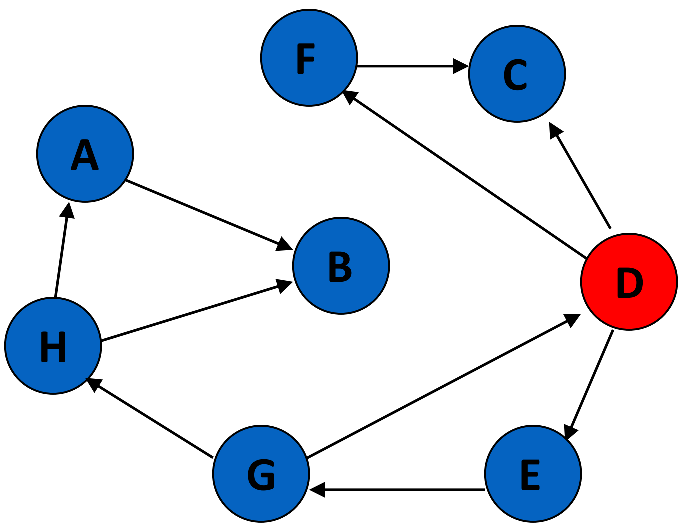
found
false

Visited nodes:

Dequeue (D is dequeued). D is not visited yet (unmarked). So, visit D (set B as marked).

Breadth-First Search

Example: **Conduct a breadth-first search in the graph and find if there is a path from D to B**



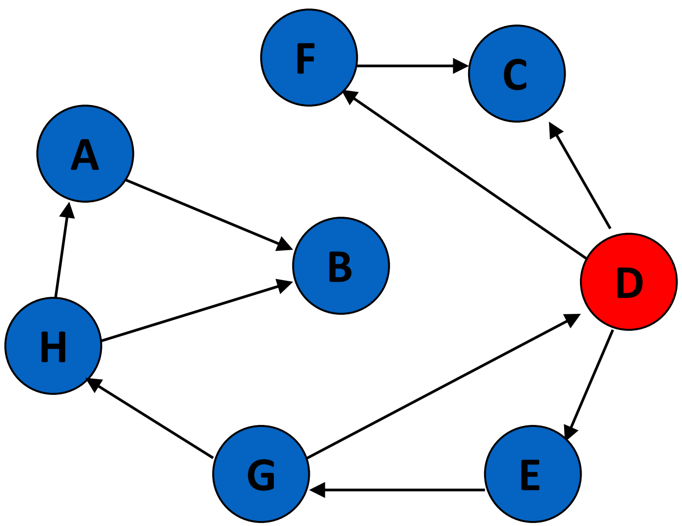
	vertices		marks	queue	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	0		
[3]	D	[3]	1		
[4]	E	[4]	0		
[5]	F	[5]	0		
[6]	G	[6]	0		
[7]	H	[7]	0		

Visited nodes:
D

Dequeue (D is dequeued). D is not visited yet (unmarked). So, visit D (set B as marked).

Breadth-First Search

Example: **Conduct a breadth-first search in the graph and find if there is a path from D to B**



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	0	
[6]	G	[6]	0	
[7]	H	[7]	0	

found

false

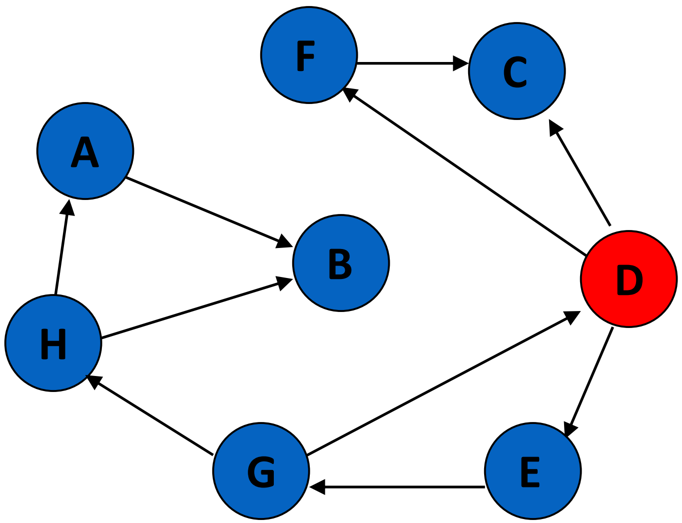
Visited nodes:

D

Enqueue all the vertices that are adjacent to D and unvisited (unmarked) (C, E and F are enqueued).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



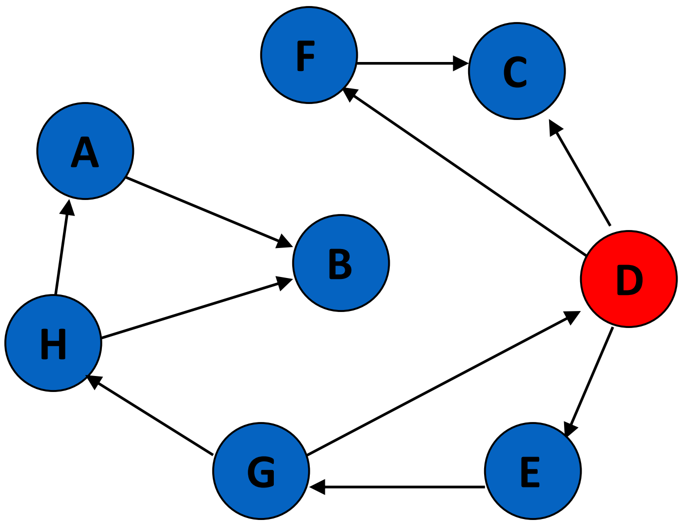
	vertices		marks	queue	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	0		
[3]	D	[3]	1		
[4]	E	[4]	0		
[5]	F	[5]	0	F	
[6]	G	[6]	0	E	
[7]	H	[7]	0	C	

Visited nodes:
D

Enqueue all the vertices that are adjacent to D and unvisited (unmarked) (C, E and F are enqueued).

Breadth-First Search

Example: **Conduct a breadth-first search in the graph and find if there is a path from D to B**



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	0	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	0	F
[6]	G	[6]	0	E
[7]	H	[7]	0	C

found
false

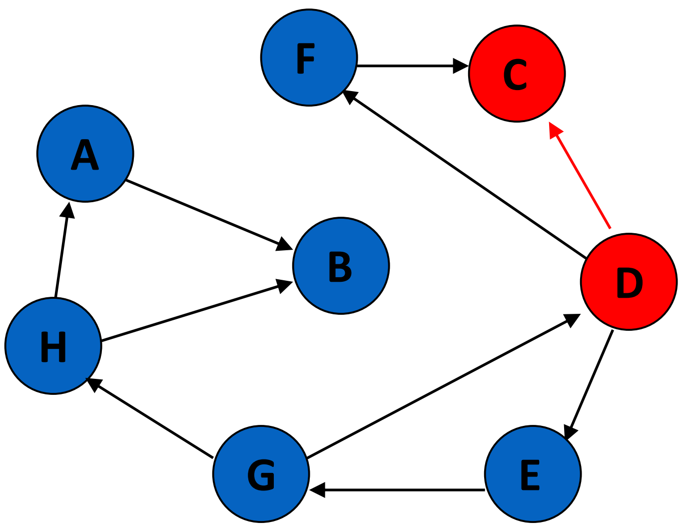
Visited nodes:

D

Dequeue (C is dequeued). C is not visited yet (unmarked). So, visit C (set C as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	1		
[3]	D	[3]	1		
[4]	E	[4]	0		
[5]	F	[5]	0		
[6]	G	[6]	0	F	
[7]	H	[7]	0	E	

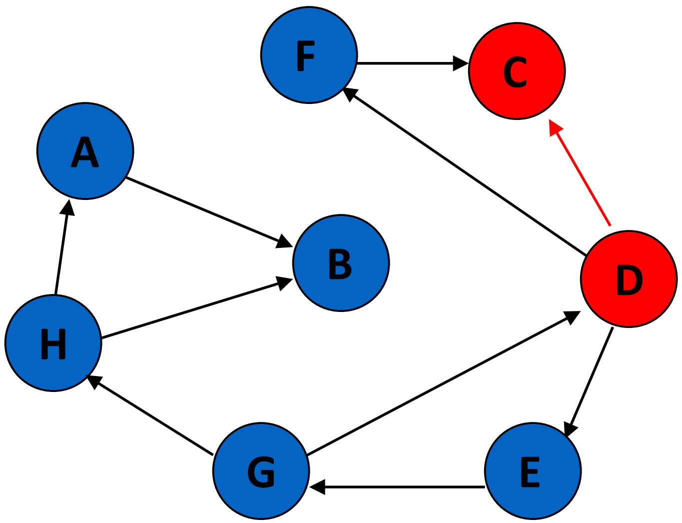
Visited nodes:

D C

Dequeue (C is dequeued). C is not visited yet (unmarked). So, visit C (set C as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	0	
[6]	G	[6]	0	F
[7]	H	[7]	0	E

found

false

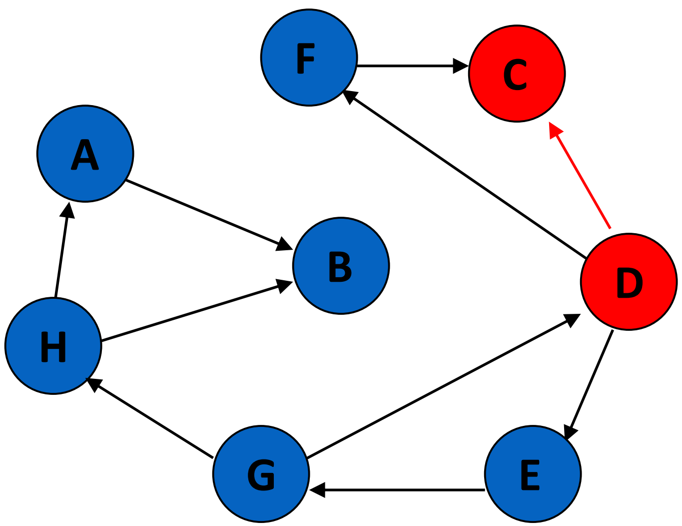
Visited nodes:

D C

Enqueue all the vertices that are adjacent to C and unvisited (unmarked) (nothing is enqueued).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	0	
[5]	F	[5]	0	
[6]	G	[6]	0	F
[7]	H	[7]	0	E

found

false

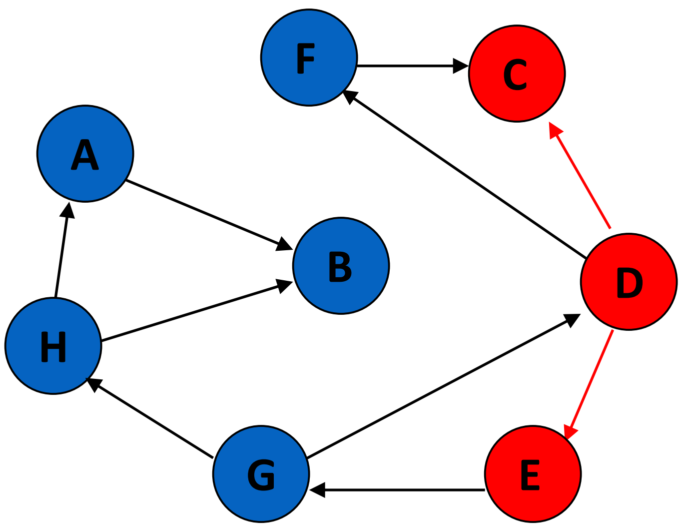
Visited nodes:

D C

Dequeue (E is dequeued). E is not visited yet (unmarked). So, visit E (set E as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	0	
[6]	G	[6]	0	
[7]	H	[7]	0	F

found: false

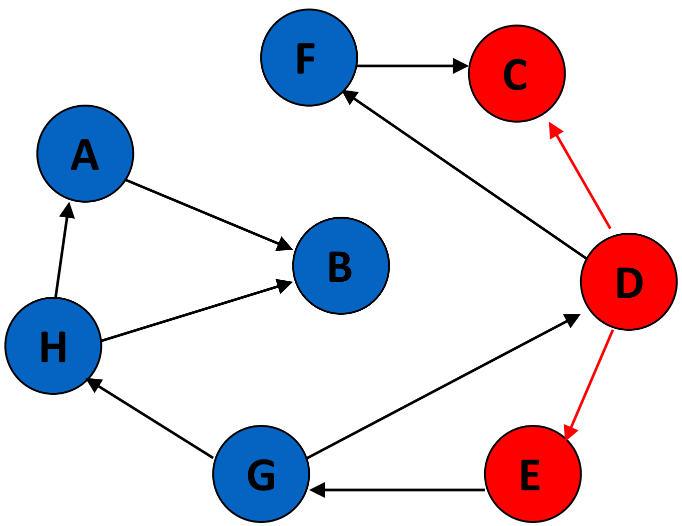
Visited nodes:

D C E

Dequeue (E is dequeued). E is not visited yet (unmarked). So, visit E (set E as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



vertices		marks		queue	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	1		
[3]	D	[3]	1		
[4]	E	[4]	1		
[5]	F	[5]	0		
[6]	G	[6]	0		
[7]	H	[7]	0	F	

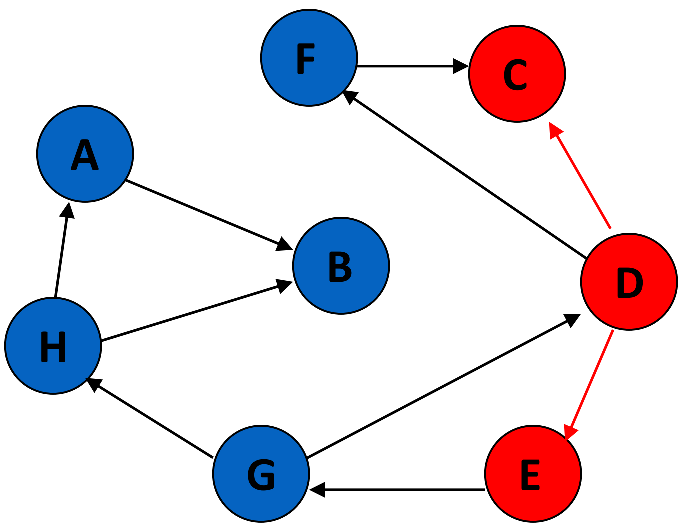
Visited nodes:

D C E

Enqueue all the vertices that are adjacent to E and unvisited (unmarked) (G is enqueued).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



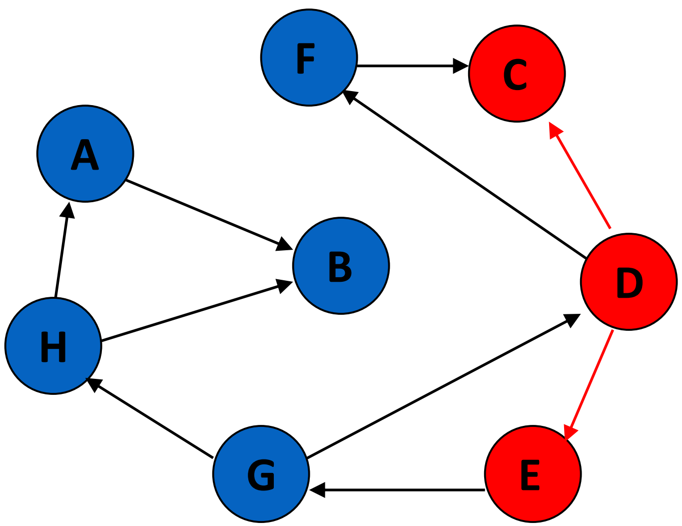
	vertices		marks	queue	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	1		
[3]	D	[3]	1		
[4]	E	[4]	1		
[5]	F	[5]	0		
[6]	G	[6]	0	G	
[7]	H	[7]	0	F	

Visited nodes:
D C E

Enqueue all the vertices that are adjacent to E and unvisited (unmarked) (G is enqueued).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	0	
[6]	G	[6]	0	G
[7]	H	[7]	0	F

found

false

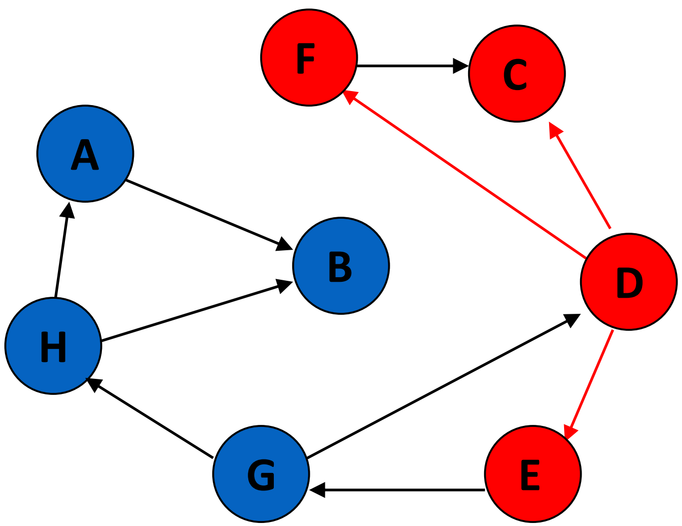
Visited nodes:

D C E

Dequeue (F is dequeued). F is not visited yet (unmarked). So, visit F (set F as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	0	
[7]	H	[7]	0	G

found

false

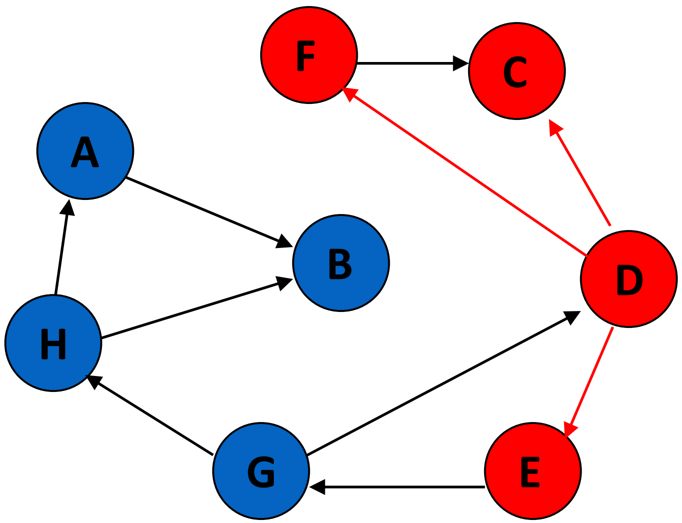
Visited nodes:

D C E F

Dequeue (F is dequeued). F is not visited yet (unmarked). So, visit F (set F as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	0	
[7]	H	[7]	0	G

found
false

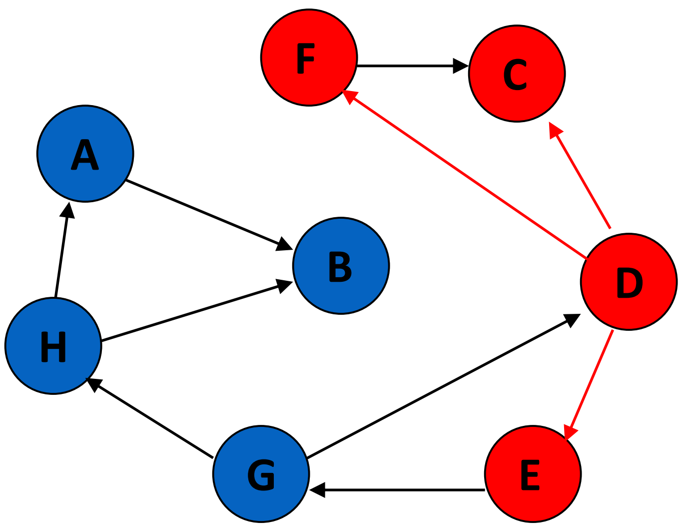
Visited nodes:

D C E F

Enqueue all the vertices that are adjacent to F and unvisited (unmarked) (nothing is enqueued).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	0	
[7]	H	[7]	0	G

found

false

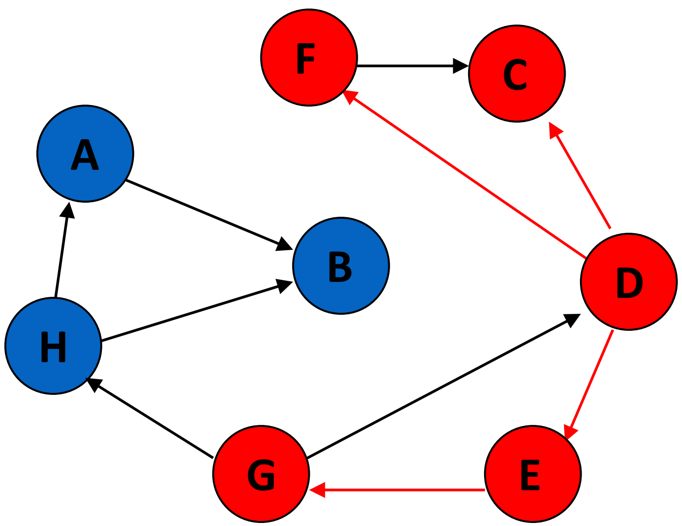
Visited nodes:

D C E F

Dequeue (G is dequeued). G is not visited yet (unmarked). So, visit G (set G as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	0	

found

false

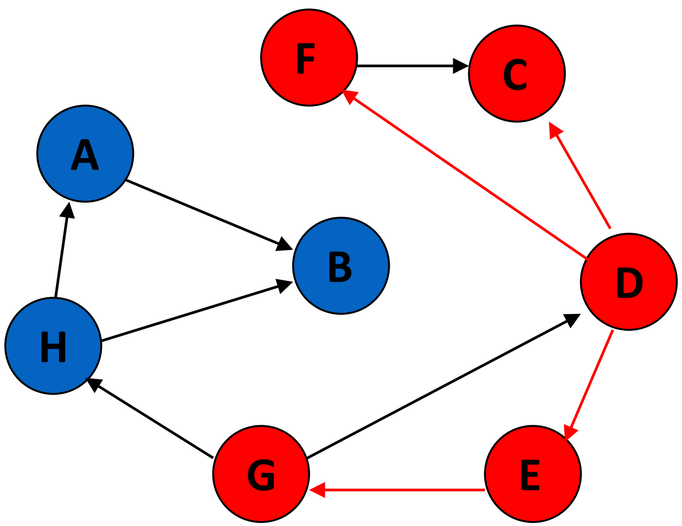
Visited nodes:

D C E F G

Dequeue (G is dequeued). G is not visited yet (unmarked). So, visit G (set G as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	0	

found

false

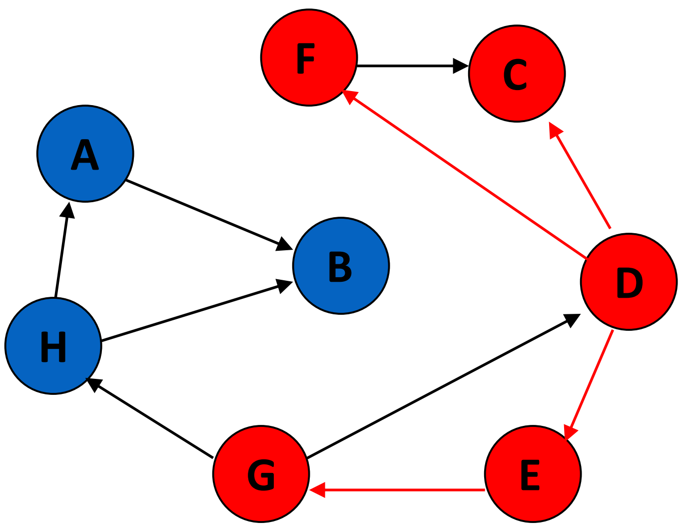
Visited nodes:

D C E F G

Enqueue all the vertices that are adjacent to G and unvisited (unmarked) (H is enqueued).

Breadth-First Search

Example: **Conduct a breadth-first search in the graph and find if there is a path from D to B**



	vertices	marks	queue
[0]	A	0	
[1]	B	0	
[2]	C	1	
[3]	D	1	
[4]	E	1	
[5]	F	1	
[6]	G	1	
[7]	H	0	H

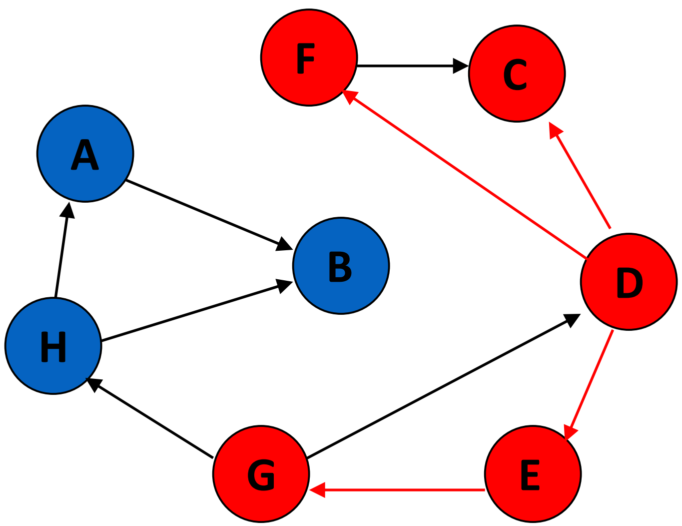
found: false

Visited nodes:
D C E F G

Enqueue all the vertices that are adjacent to G and unvisited (unmarked) (H is enqueued).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	0	H

found

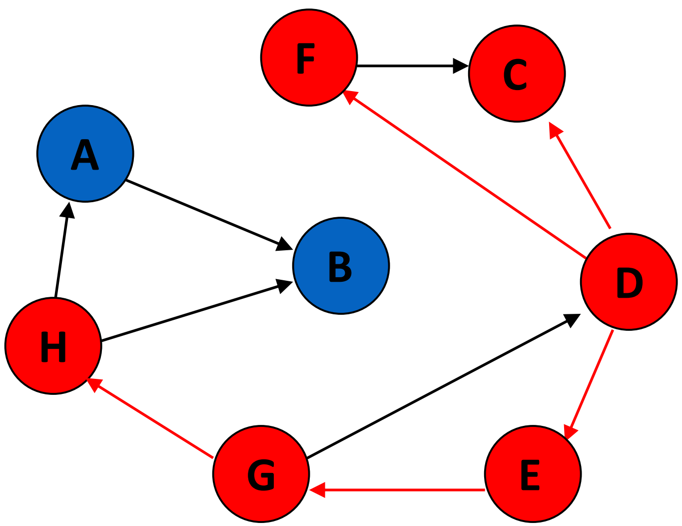
false

Visited nodes:
D C E F G

Dequeue (H is dequeued). H is not visited yet (unmarked). So, visit H (set H as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue	found
[0]	A	[0]	0		false
[1]	B	[1]	0		
[2]	C	[2]	1		
[3]	D	[3]	1		
[4]	E	[4]	1		
[5]	F	[5]	1		
[6]	G	[6]	1		
[7]	H	[7]	1		

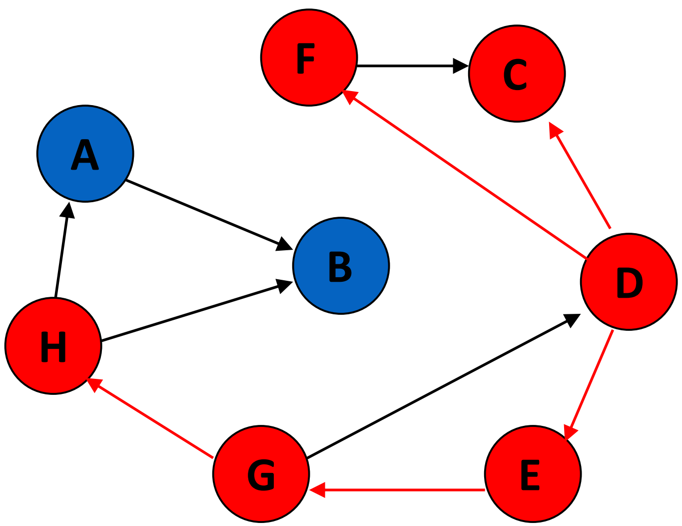
Visited nodes:

D C E F G H

Dequeue (H is dequeued). H is not visited yet (unmarked). So, visit H (set H as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	1	

found

false

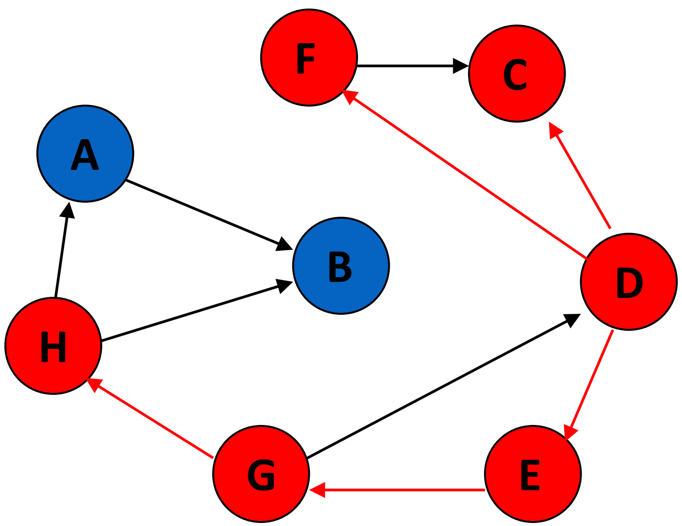
Visited nodes:

D C E F G H

Enqueue all the vertices that are adjacent to H and unvisited (unmarked) (A and B are enqueued).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	B
[7]	H	[7]	1	A

found
false

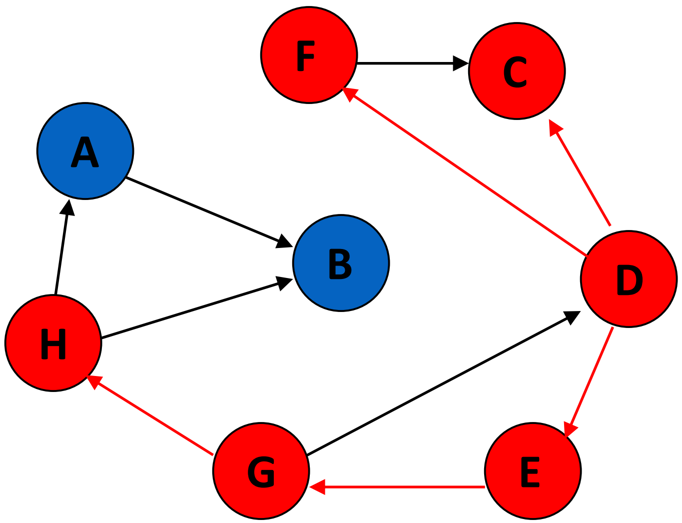
Visited nodes:

D C E F G H

Enqueue all the vertices that are adjacent to H and unvisited (unmarked) (A and B are enqueued).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	0	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	B
[7]	H	[7]	1	A

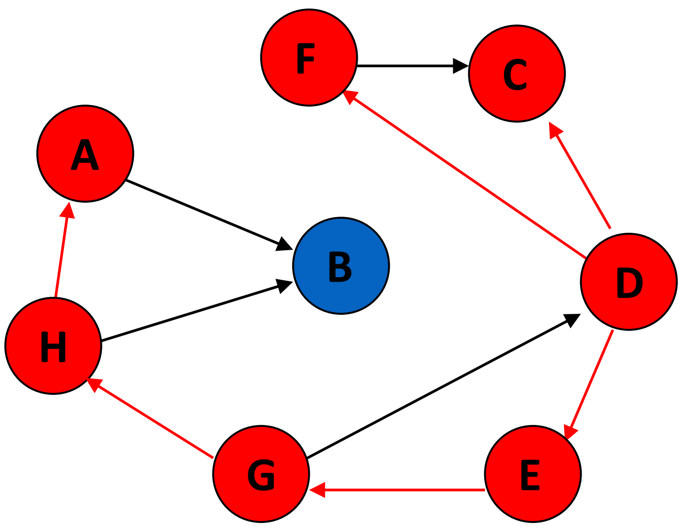
found: false

Visited nodes:
D C E F G H

Dequeue (A is dequeued). A is not visited yet (unmarked). So, visit A (set A as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue	found
[0]	A	[0]	1		false
[1]	B	[1]	0		
[2]	C	[2]	1		
[3]	D	[3]	1		
[4]	E	[4]	1		
[5]	F	[5]	1		
[6]	G	[6]	1		
[7]	H	[7]	1	B	

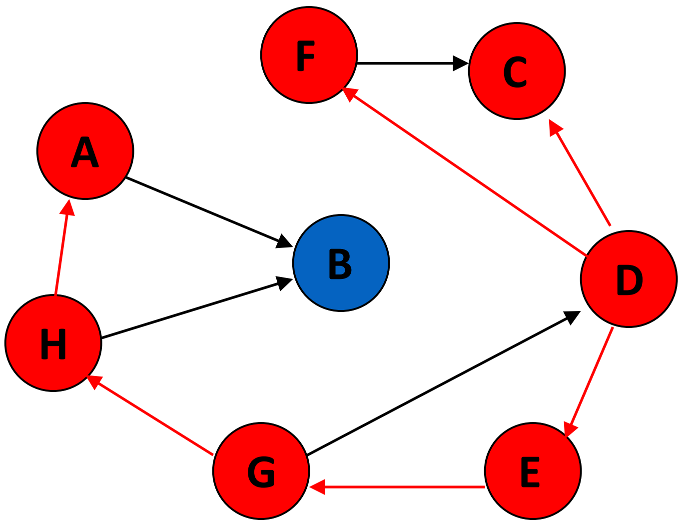
Visited nodes:

D C E F G H A

Dequeue (A is dequeued). A is not visited yet (unmarked). So, visit A (set A as marked).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



	vertices		marks	queue
[0]	A	[0]	1	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	1	B

found

false

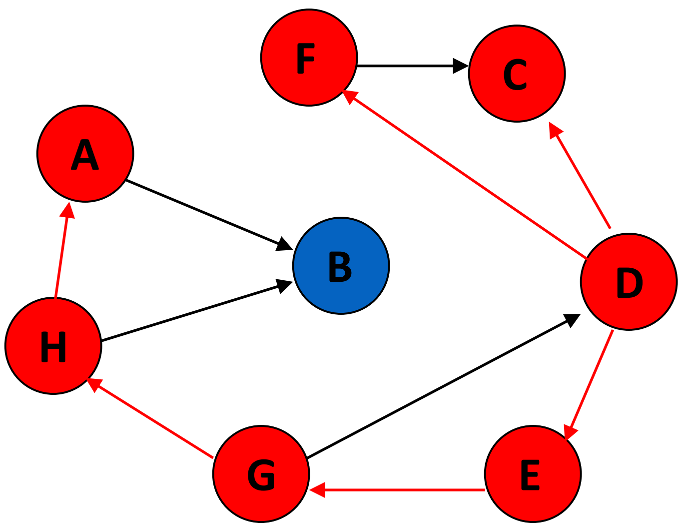
Visited nodes:

D C E F G H A

Enqueue all the vertices that are adjacent to A and unvisited (unmarked) (B is enqueued).

Breadth-First Search

Example: **Conduct a breadth-first search in the graph and find if there is a path from D to B**



	vertices		marks	queue
[0]	A	[0]	1	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	B
[7]	H	[7]	1	B

found

found
false

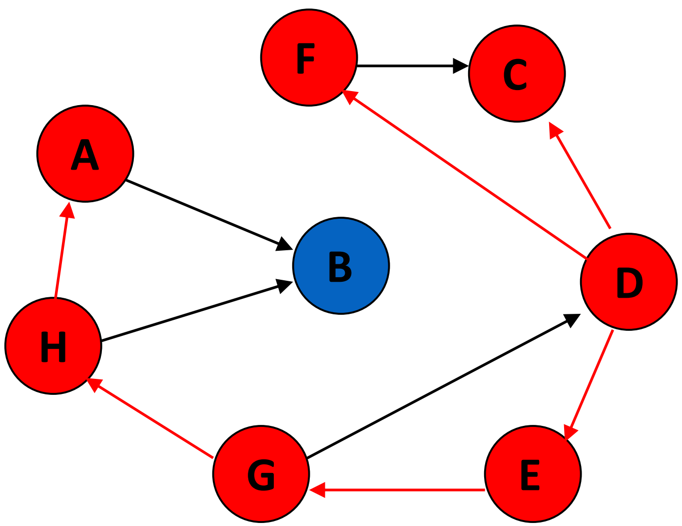
Visited nodes:

D C E F G H A

Enqueue all the vertices that are adjacent to A and unvisited (unmarked) (B is enqueued).

Breadth-First Search

Example: **Conduct a breadth-first search in the graph and find if there is a path from D to B**



	vertices		marks	queue
[0]	A	[0]	1	
[1]	B	[1]	0	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	B
[7]	H	[7]	1	B

found
false

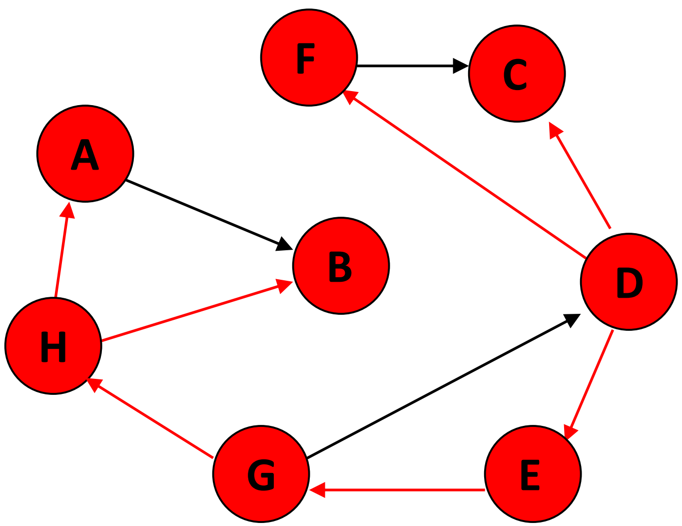
Visited nodes:

D C E F G H A

Dequeue (B is dequeued). B is not visited yet (unmarked). So, visit B (set B as marked).

Breadth-First Search

Example: **Conduct a breadth-first search in the graph and find if there is a path from D to B**



	vertices		marks	queue
[0]	A	[0]	1	
[1]	B	[1]	1	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	1	B

found

false

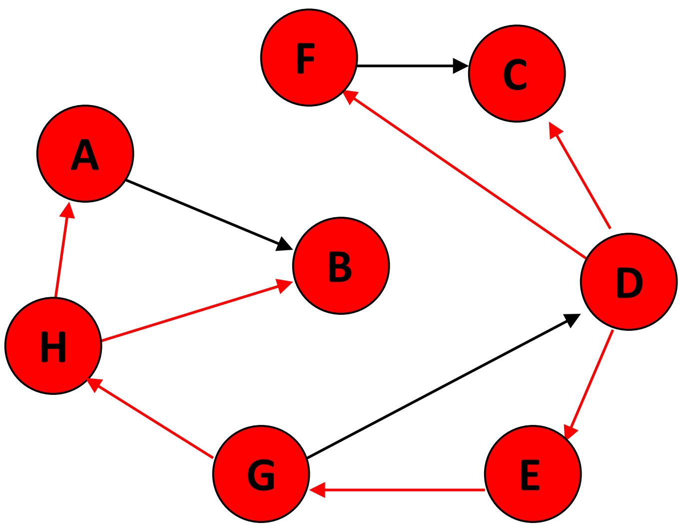
Visited nodes:

D C E F G H A B

Dequeue (B is dequeued). B is not visited yet (unmarked). So, visit B (set B as marked).

Breadth-First Search

Example: **Conduct a breadth-first search in the graph and find if there is a path from D to B**



Visited nodes:

D C E F G H A B

	vertices		marks	queue
[0]	A	[0]	1	
[1]	B	[1]	1	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	1	B

found

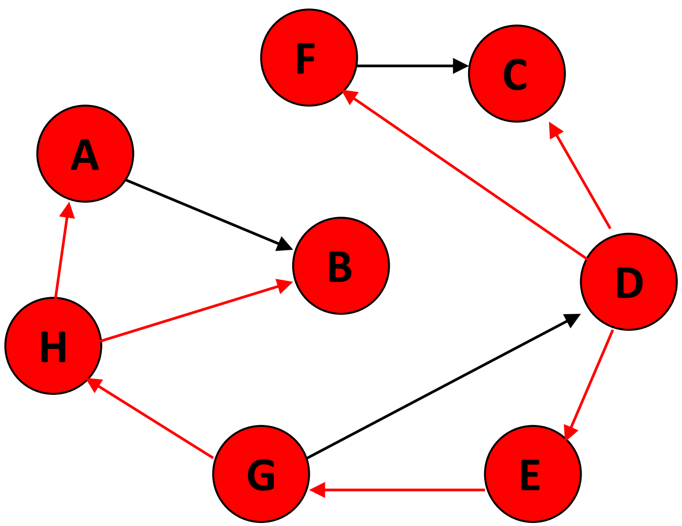
false

B is the destination vertex. So set *found* to true (there is a path). Search is complete.

Note: We can still carry on with the search (until the queue is empty).

Breadth-First Search

Example: Conduct a breadth-first search in the graph and find if there is a path from D to B



Visited nodes:

D C E F G H A B

	vertices		marks	queue
[0]	A	[0]	1	
[1]	B	[1]	1	
[2]	C	[2]	1	
[3]	D	[3]	1	
[4]	E	[4]	1	
[5]	F	[5]	1	
[6]	G	[6]	1	
[7]	H	[7]	1	B

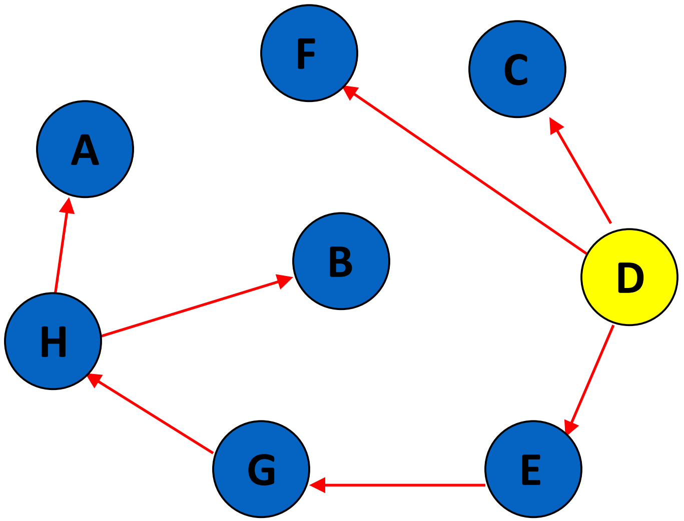
found
true

B is the destination vertex. So set *found* to true (there is a path). Search is complete.

Note: We can still carry on with the search (until the queue is empty).

Breadth-First Search

Breadth-First Search yields a tree (the path taken during the search) also known as the Breadth-First Tree.



Breadth-First Search

```
template<class VertexType>
void BreadthFirstSearch(GraphType<VertexType> graph,
VertexType startVertex, VertexType endVertex)
{
  QueType<VertexType> queue;
  QueType<VertexType> vertexQ;

  bool found = false;
  VertexType vertex;
  VertexType item;

  graph.ClearMarks();
  queue.Enqueue(startVertex);
  do
  {
    queue.Dequeue(vertex);
    if (vertex == endVertex)
    {
           cout  << vertex;
           found = true;
    }
```

Breadth-First Search

```
    else
    {
            if (!graph.IsMarked(vertex))
            {
                    graph.MarkVertex(vertex);
                    cout  << vertex;
                    graph.GetToVertices(vertex, vertexQ);

                    while (!vertexQ.IsEmpty())
                    {
                            vertexQ.Dequeue(item);
                            if (!graph.IsMarked(item))
                            queue.Enqueue(item);
                    }
            }
    }
  } while (!queue.IsEmpty() && !found);
  if (!found)
  cout << "Path not found." << endl;
}
```

Differences between the BFS and DFS

BFS	**DFS**
Stands for “Breadth-first search”	Stands for “Depth-first search”
The nodes are explored breadth wise level by level.	The nodes are explored depth-wise until there are only leaf nodes and then backtracked to explore other unvisited nodes.
BFS is performed with the help of queue data structure.	DFS is performed with the help of stack data structure.
Slower in performance.	Faster than BFS.
Useful in finding the shortest path between two nodes.	Used mostly to detect cycles in graphs.

Concluding Remarks

- There can be multiple paths from source vertex to destination vertex.
- Both Breadth First Search and Depth First Search ensures that the path will be found if there is any.
- Breadth First Search ensures that, if there are multiple paths from source vertex to destination vertex, the destination vertex is reached using minimum number of edges, i.e. it takes the shortest among all the paths, given that,
  - The edges do not have weights, OR
  - All the edges have equal weights
- Depth First Search does not ensure that the shortest path is taken.