

CS 112: Data Structures

Assignment Project Exam Help

<https://powcoder.com>
Sesh Venugopal

Add WeChat powcoder

Depth-first Traversal (DFS) on Graphs

Traversals are a means to explore the topology of the graph

There are two standard graph traversal methods:

Depth-first search (DFS)

Breadth-first search (BFS)

(The word "search" does not mean searching for a specific vertex or edge - it actually means "scan")

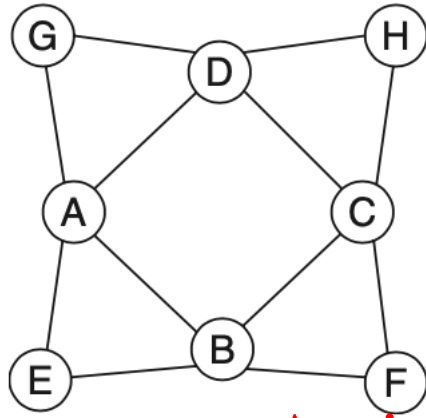
<https://powcoder.com>

Neither of these traversals answers a specific question – typically an actual graph application might use one or the other as a building block to solve a practical problem

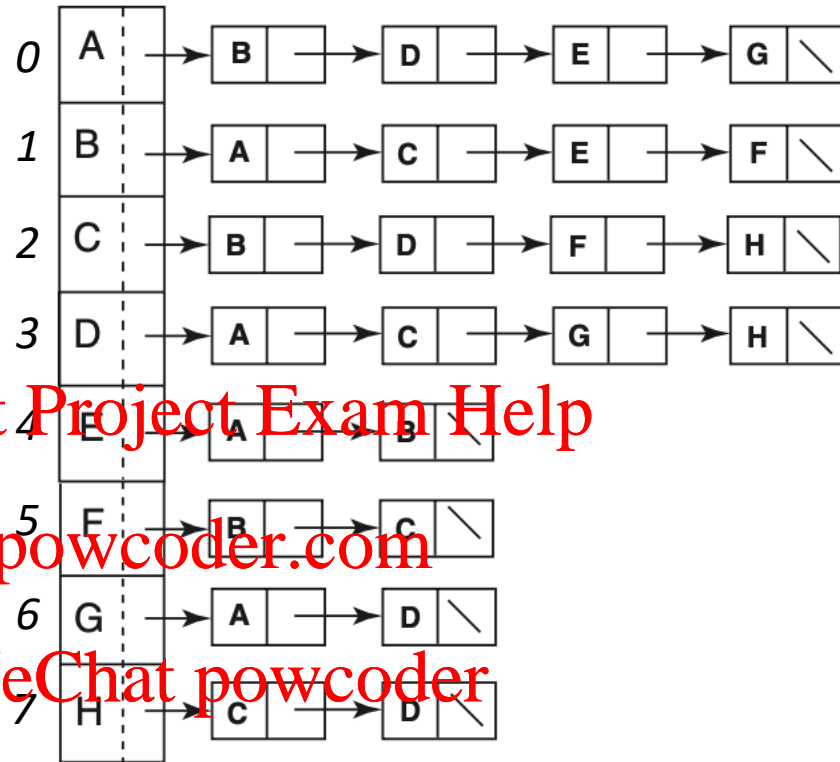
For example, LinkedIn might use BFS to tell how many links separate you from someone else e.g. 3rd connection

Or, a routing algorithm might use DFS to determine how many different paths there are to get from point A to point B

Depth-first search (DFS)



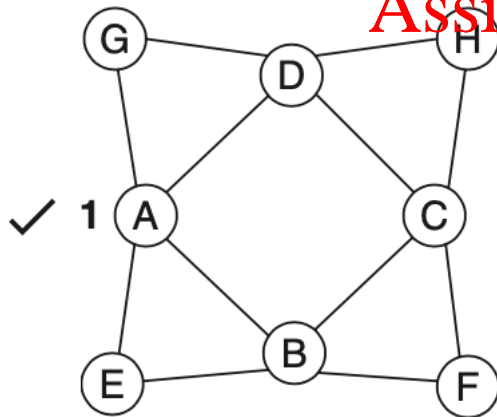
Vertices have been numbered in alphabetical order for ease of exposition. In practice, they can be numbered (placed in the adjacency array) in any order



In the adjacency list

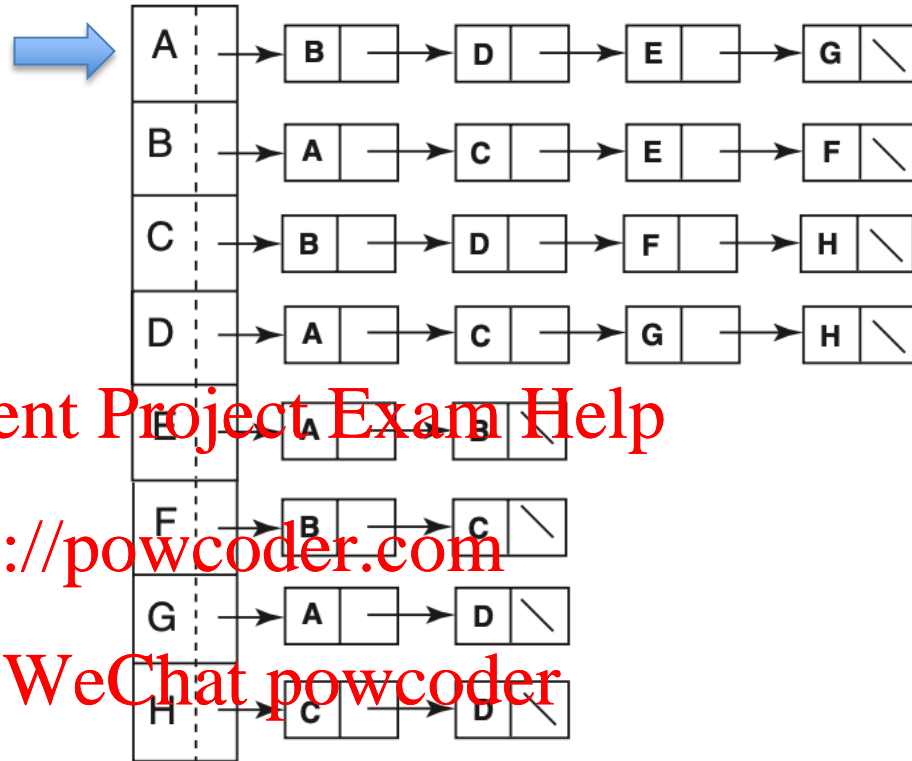
1. Neighbors of a vertex are stored in alphabetical order only for ease of exposition. In practice, they could be in any order – it all depends on how they are stored in file
2. The linked list nodes show names of vertices. Again this is for ease of readability, in practice they would be vertex number. So, imagine 1 in place of B, 3 in place of D, etc.

Suppose the traversal starts at vertex A. It visits A - the traversal keeps track of which vertices have been visited by appropriately “marking” a vertex when it is visited (in the boolean visited array)



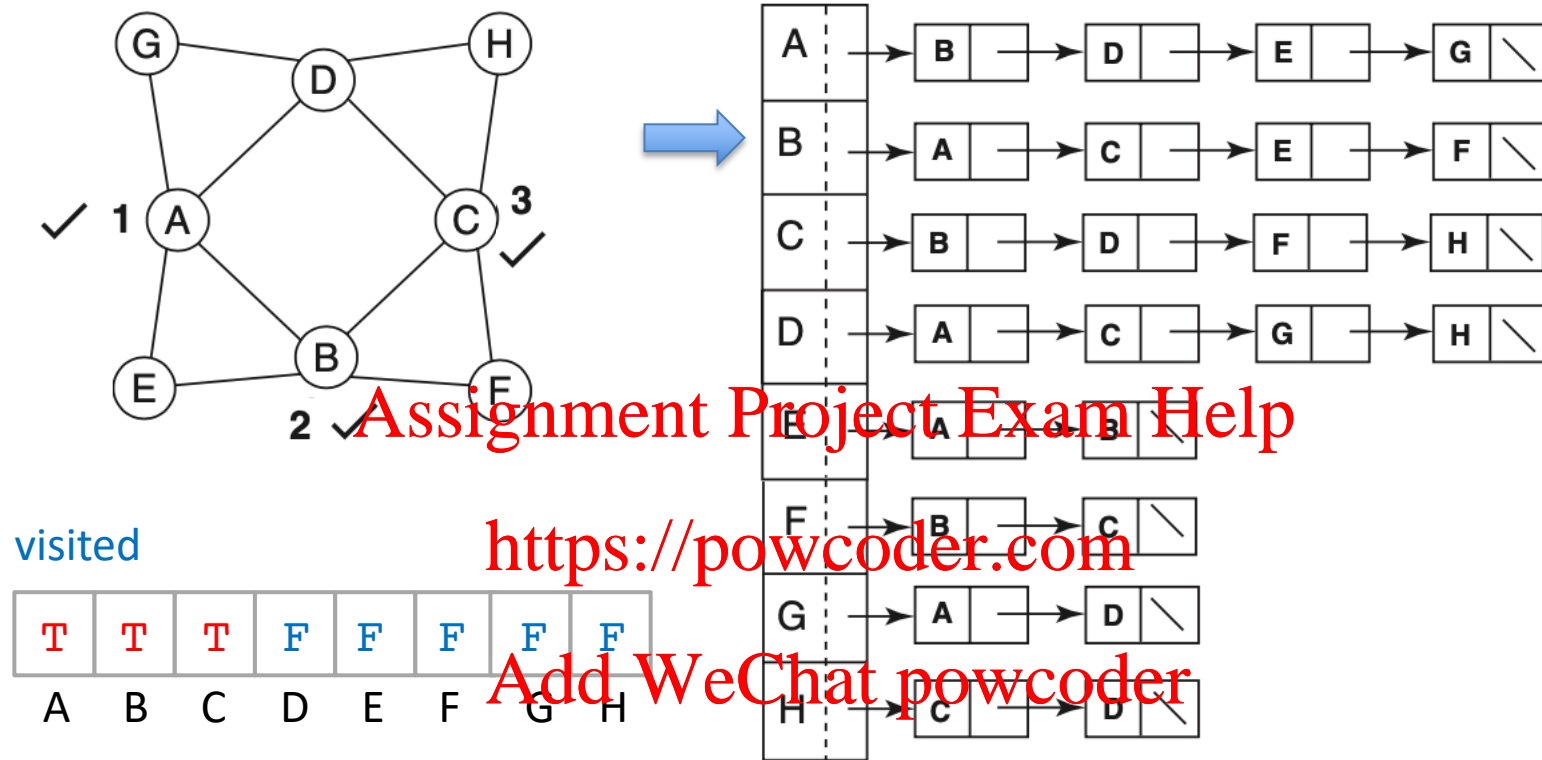
visited

T	F	F	F	F	F	F	F
A	B	C	D	E	F	G	H



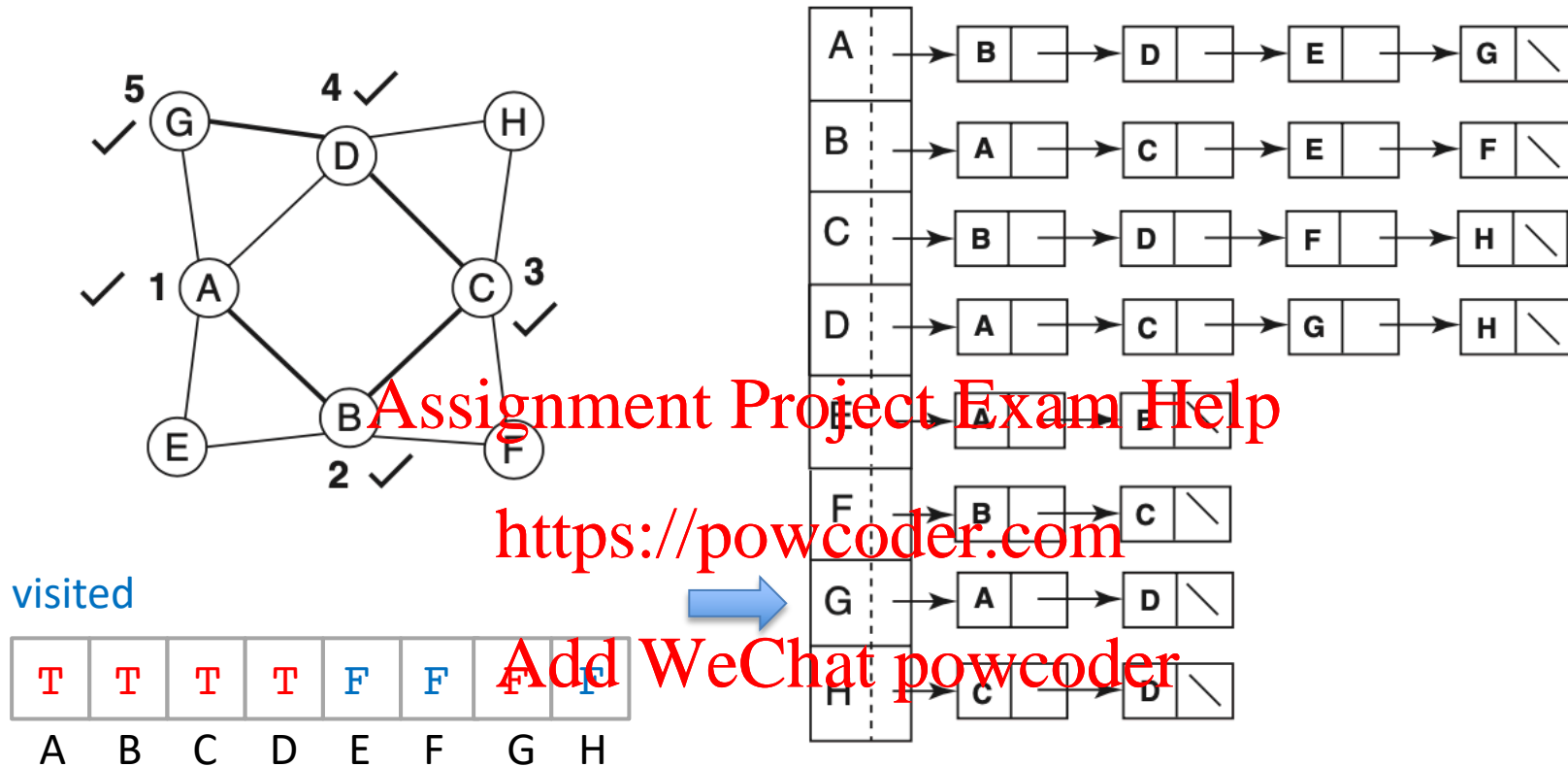
After visiting A, the traversal has to choose a vertex to visit next. A has four neighbors: B, D, E, and G. Any one of these can be picked as the next vertex to visit without affecting the correctness of the traversal. Which neighbor is picked next depends on the order in which the neighbors are stored. Here, B is the first vertex in A’s neighbor list, so it is picked next to visit

Depth-first search (DFS)



After visiting *B*, in order to decide which vertex to visit next, it first examines *B*'s neighbor *A*. Since *A* has been visited, it is skipped. The next vertex in the neighbors list, *C*, has not yet been visited. So *C* is visited next.

Depth-first search (DFS)

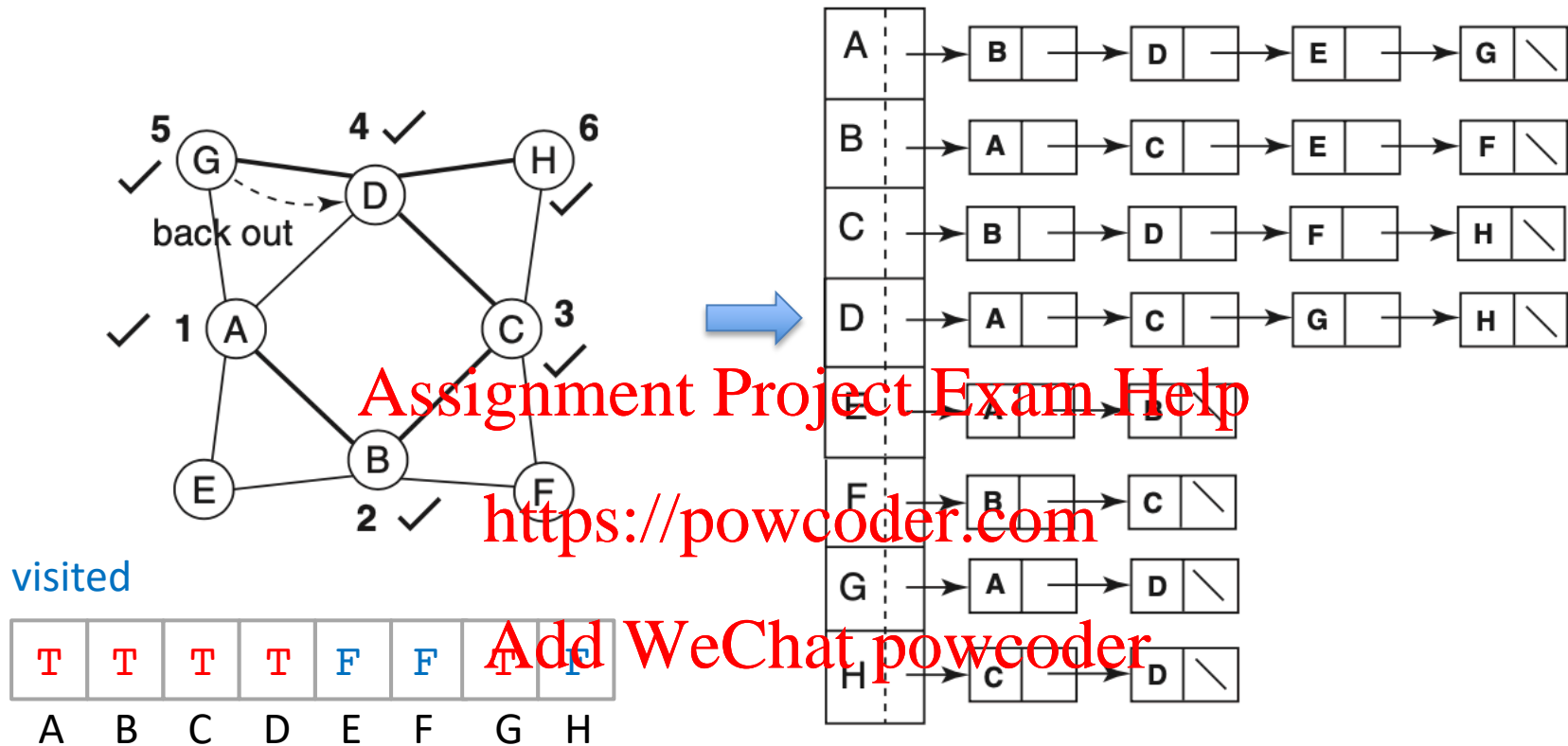


Continuing in this manner, the traversal next goes to D, then to G.

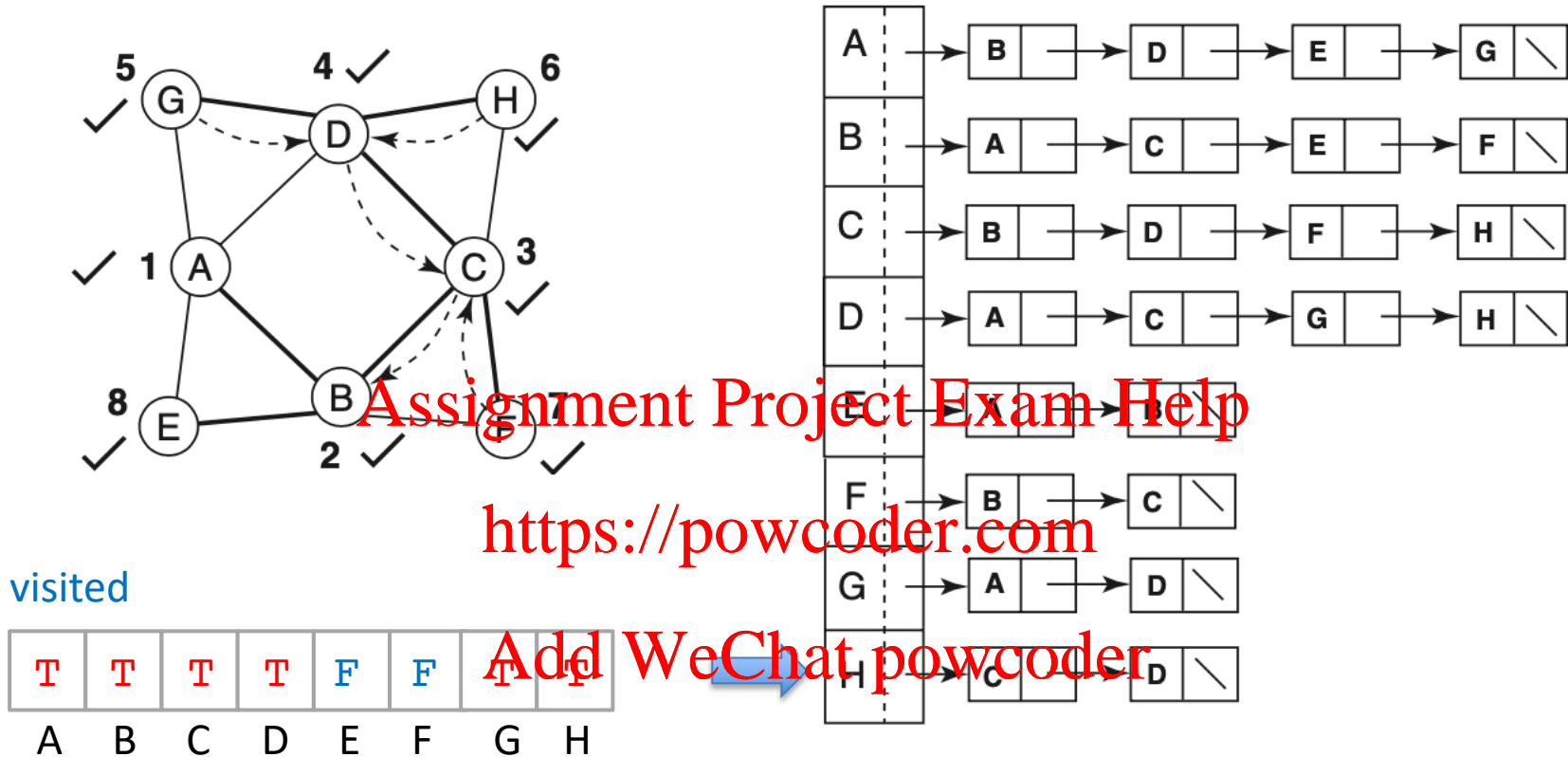
It examines the neighbors of G to select the next vertex to visit, but finds that both neighbors, A and D, have already been visited.

So the traversal has hit a dead end.

Depth-first search (DFS)

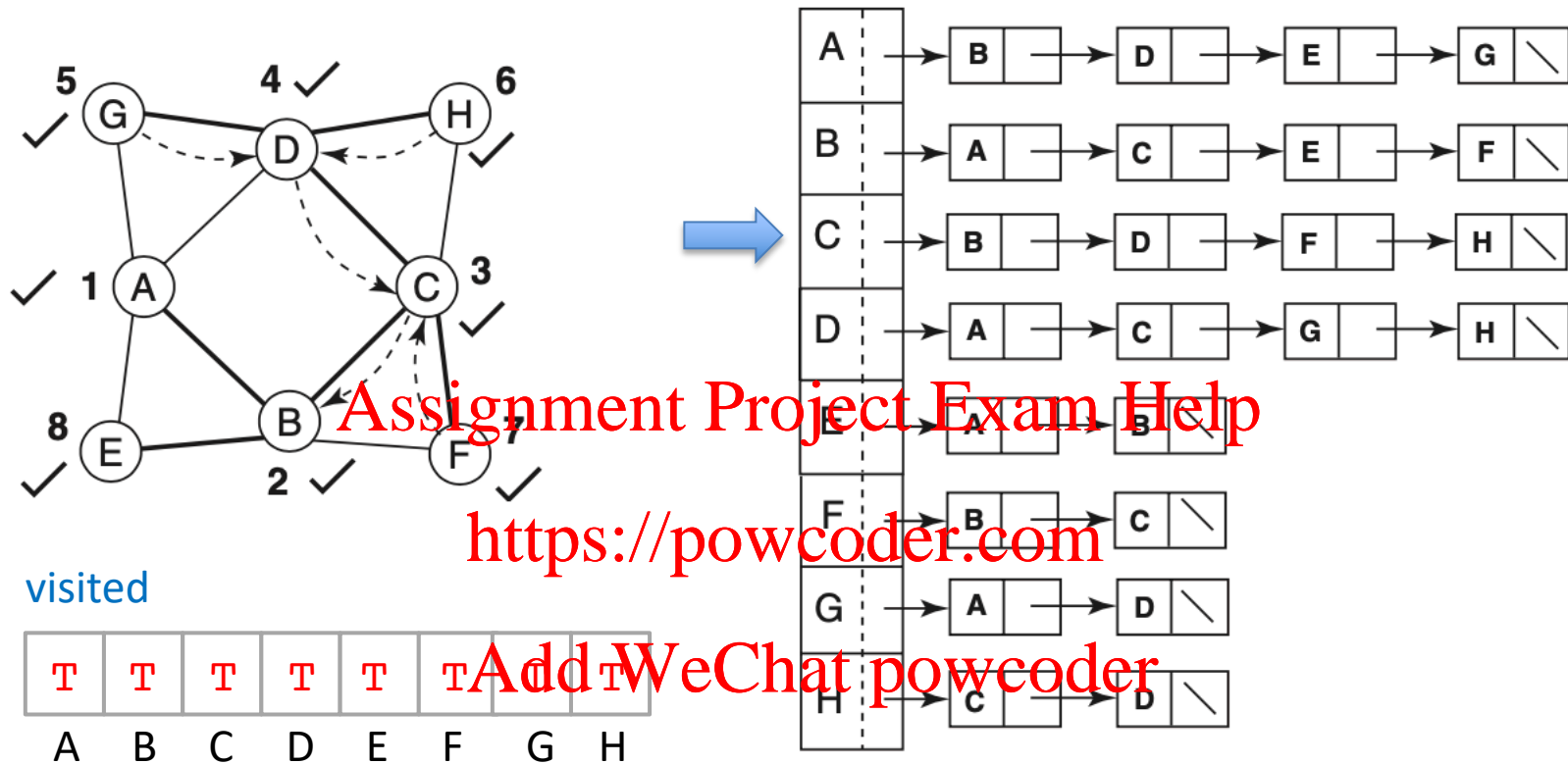


With no way forward from G, the traversal backs up to the previously visited vertex D. This is shown by the dashed arc with an arrow – this is basically a return to the previous step in the recursion, **not an edge in the graph**.



At *D*, the traversal now examines *H*. Since *H* has not been visited, the traversal moves forward and visits *H*.

After visiting *H* it finds that both neighbors of *H*, *C* and *D*, have been visited. So the traversal is at a dead-end again, and it backs out to *D* only to come to a dead end once more, since all the neighbors of *D* have been visited.



The traversals backs up from D to C, from where it visits F, and then finds itself at a dead end yet again – both of F's neighbors, B and C, have already been visited. So from F it backs up to C, and then to B, from where it moves forward to visit E. It then backs up to B and then to A, upon which it terminates.

The final sequence of visits is: A, B, C, D, G, H, F, E

Depth-first search (DFS)

```
// recursive dfs
private void dfs(int v, boolean[] visited) {
    visited[v] = true;
    for (Neighbor e=adjLists[v].adjList; e != null; e=e.next) {
        if (!visited[e.vertexNum]) {
            dfs(e.vertexNum, visited);
        }
    }
}
```

Assignment Project Exam Help

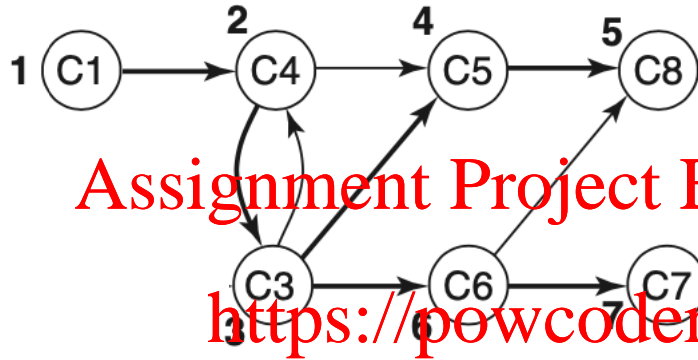
Starting at some vertex, depth-first search follows a path that goes as deep or far down the graph as possible, visiting each vertex along this path.

<https://powcoder.com>
Add WeChat powcoder

When the traversal cannot proceed any farther along the current direction, it backs up and tries to take another direction for traversal, again going forward as deep as possible before backing up and repeating.

Depth-first search (DFS)

The process (and code) is the exactly same for a directed graph.



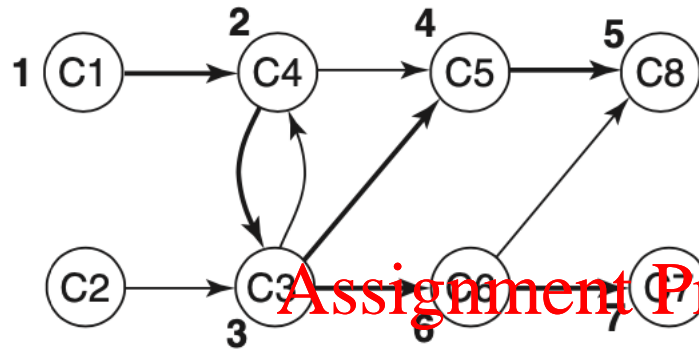
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Depth-first search (DFS)

What if there was another vertex, C2, in the graph, with an edge to C3?



If the traversal starts at C1, there is no way to reach C2. (So DFS can be used to find out all vertices reachable from a given vertex.)

Assignment Project Exam Help

<https://powcoder.com>

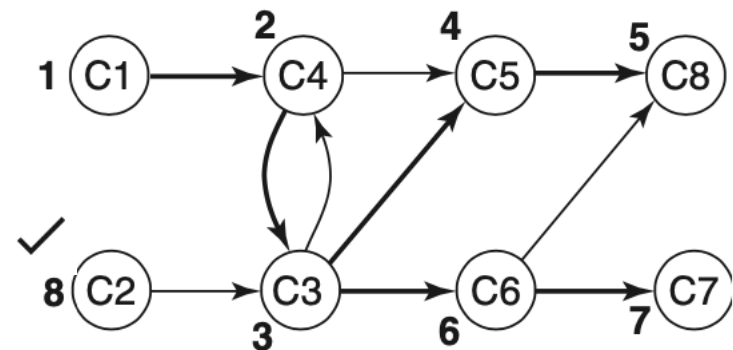
However, if we want to do a complete traversal, and get to every single vertex, the only way to do this is to restart the traversal.

Add WeChat powcoder

And the restart is done at the first unvisited vertex in the visited array

visited

T	F	T	T	T	T	T	T
C1	C2	C3	C4	C5	C6	C7	C8



Depth-first search (DFS)

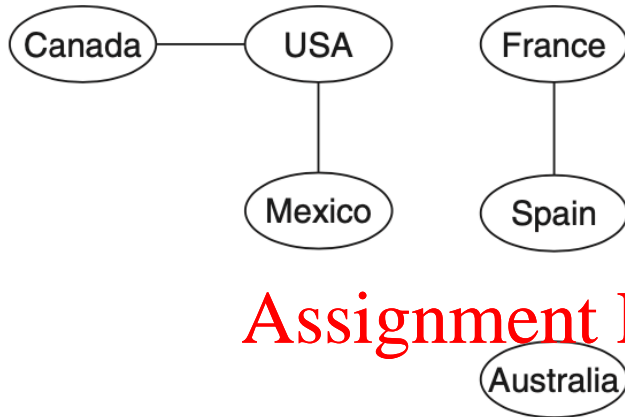
We need a "driver" to restart the traversal every time it stalls, and we haven't finished visiting all vertices.

```
// driver (this is the method called from any application)
public void dfs() {
    boolean[] visited = new boolean[adjLists.length];
    for (int v=0; v < visited.length; v++) {
        visited[v] = false;
    }
    for (int v=0; v < visited.length; v++) {
        if (!visited[v]) { // start/restart at v
            dfs(v, visited);
        }
    }
}

// recursive dfs
private void dfs(int v, boolean[] visited) {
    visited[v] = true;
    for (Neighbor e=adjLists[v].adjList; e != null; e=e.next) {
        if (!visited[e.vertexNum]) {
            dfs(e.vertexNum, visited);
        }
    }
}
```

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

Restarts can happen with undirected graphs as well



This graph has islands – it is “unconnected”

No matter where we start the traversal, 2 restarts would be needed

Assignment Project Exam Help

<https://powcoder.com>

Start at US

F	F	F	F	F	F
US	SP	AU	CA	MX	FR

Restart at SP

T	F	F	T	T	F
US	SP	AU	CA	MX	FR

Restart at AU

T	T	F	T	T	T
US	SP	AU	CA	MX	FR

Done

T	T	T	T	T	T
US	SP	AU	CA	MX	FR

DFS big O Running Time

What to count towards running time?

$O(n+e)$ Recursive dfs

n - Visiting a vertex : $O(1)$

(The $O(1)$ includes marking cell in visited array as true, plus whatever is done in visiting, e.g. printing the vertex name)

<https://powcoder.com>

e (directed) - Checking whether a neighbor is visited: $O(1)$

$2e$ (undirected) (The $O(1)$ includes indexing into the visited array, as well as pushing pointer to next neighbor in adjacency linked list)

$O(n)$ Driver
- Scanning the visited array

Total: $O(n+e)$

There is no worst case/best case separation of running times since the traversal runs unconditionally through the entire graph