

COMP 250

INTRODUCTION TO COMPUTER SCIENCE

Assignment Project Exam Help

<https://powcoder.com>

Week 14-1: Graphs 2

Giulia Alberini, Fall 2020

Slides adapted from Michael Langer's

WHAT ARE WE GOING TO DO IN THIS VIDEO?



- Recursive graph traversal
 - depth first
- Non-recursive graph traversal
 - depth first
 - breadth first

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

RECALL: TREE TRAVERSAL (RECURSIVE)

```
depthFirst_Tree (root) {  
    if (root is not empty) {  
        visit root // preorder  
        for each child of root  
            depthfirst_Tree ( child )  
    }  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

GRAPH TRAVERSAL (RECURSIVE)

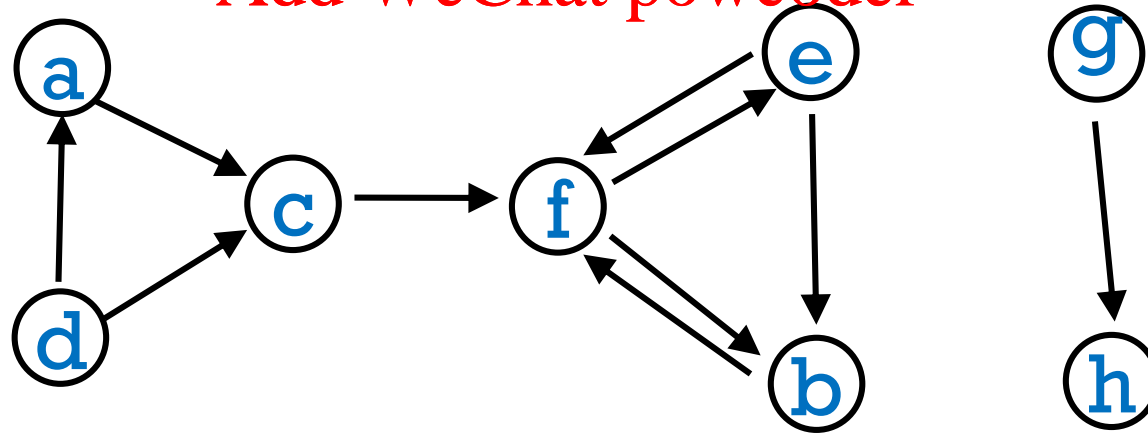
Need to specify a starting vertex.

Visit all nodes that are “reachable” by a path from a starting vertex.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



GRAPH TRAVERSAL (RECURSIVE)

```
depthFirst_Graph (v) {
```

```
    v.visited = true
```

```
    for each w such that (v,w) is in E
```

```
        // i.e. v.adjList.contains(w) returns true
```

```
        ??
```

```
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

GRAPH TRAVERSAL (RECURSIVE)

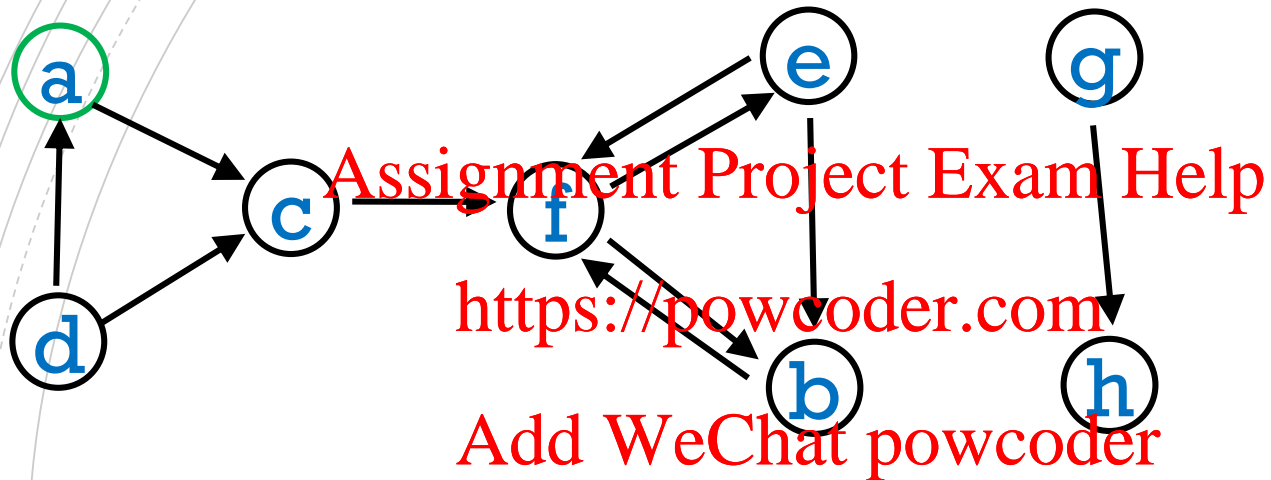
```
depthFirst_Graph (v) {  
    v.visited = true  
    visit v // do something with v  
    for each w such that (v,w) is in E  
    // i.e. for each w in v.adjList  
        if !(w.visited) // avoid cycles!  
            depthFirst_Graph(w)  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

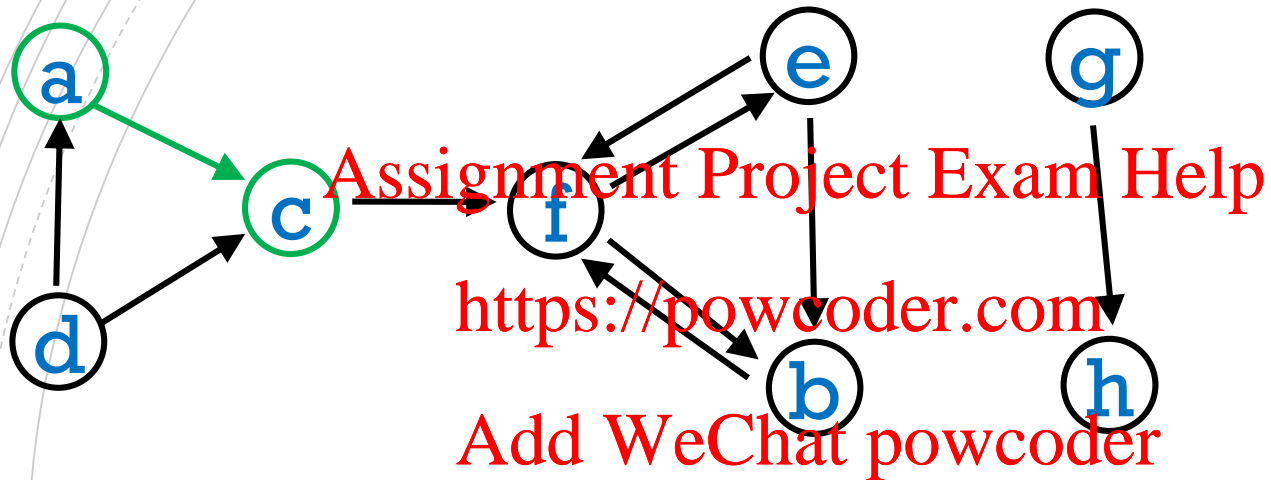
CALL STACK FOR depthFirst(a)



```
depthFirst_Graph (v) {  
    v.visited = true  
    for each w s.t. (v,w) is in E  
        if !(w.visited)  
            depthFirst_Graph(w)  
}
```

a

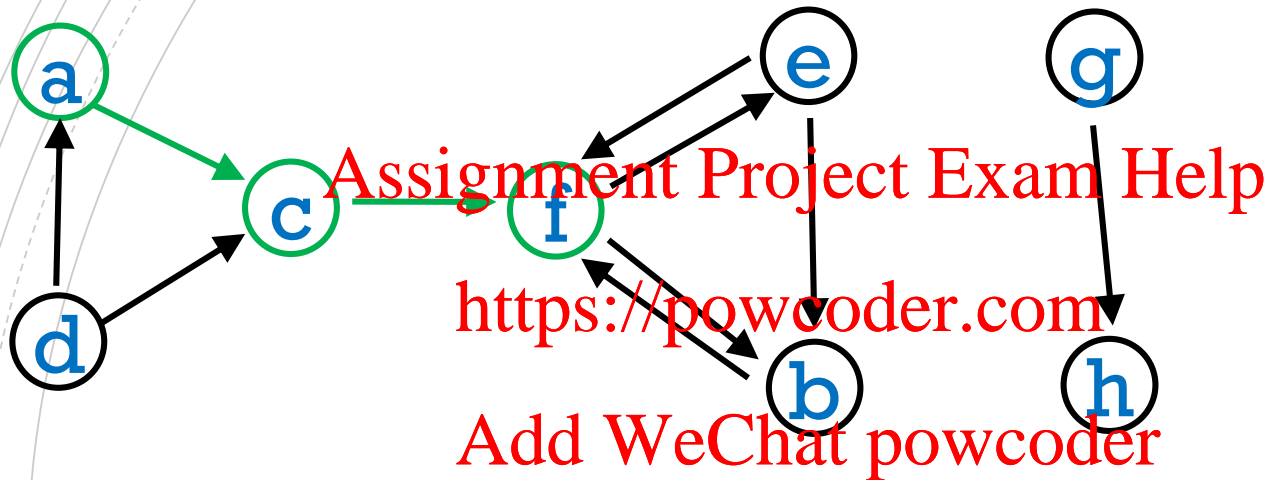
CALL STACK FOR depthFirst(a)



```
depthFirst_Graph (v) {  
    v.visited = true  
    for each w s.t. (v,w) is in E  
        if !(w.visited)  
            depthFirst_Graph(w)  
}
```

a c
a a

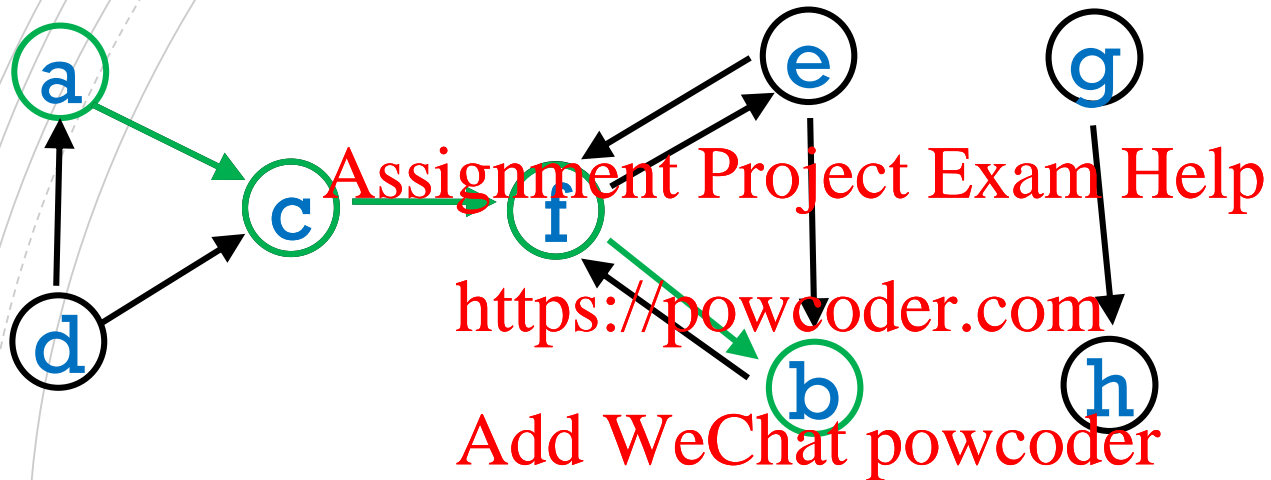
CALL STACK FOR depthFirst(a)



a c f
a a c
a a a

```
depthFirst_Graph (v) {  
    v.visited = true  
    for each w s.t. (v,w) is in E  
        if !(w.visited)  
            depthFirst_Graph(w)  
}
```

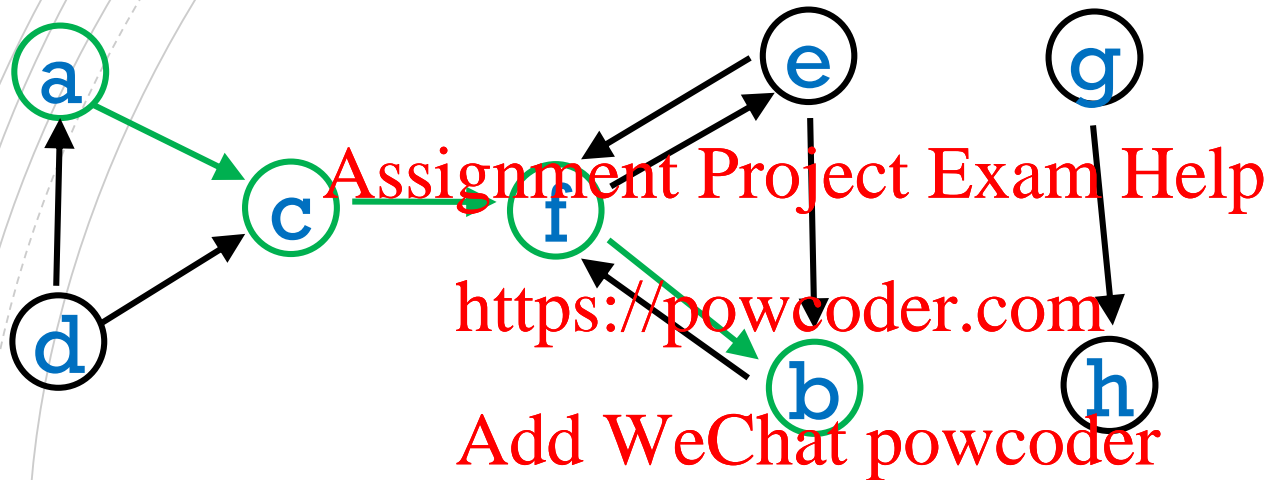
CALL STACK FOR depthFirst(a)



			b
		f	f
	c	c	c
a	a	a	a

```
depthFirst_Graph (v) {  
    v.visited = true  
    for each w s.t. (v,w) is in E  
        if !(w.visited)  
            depthFirst_Graph (w)  
}
```

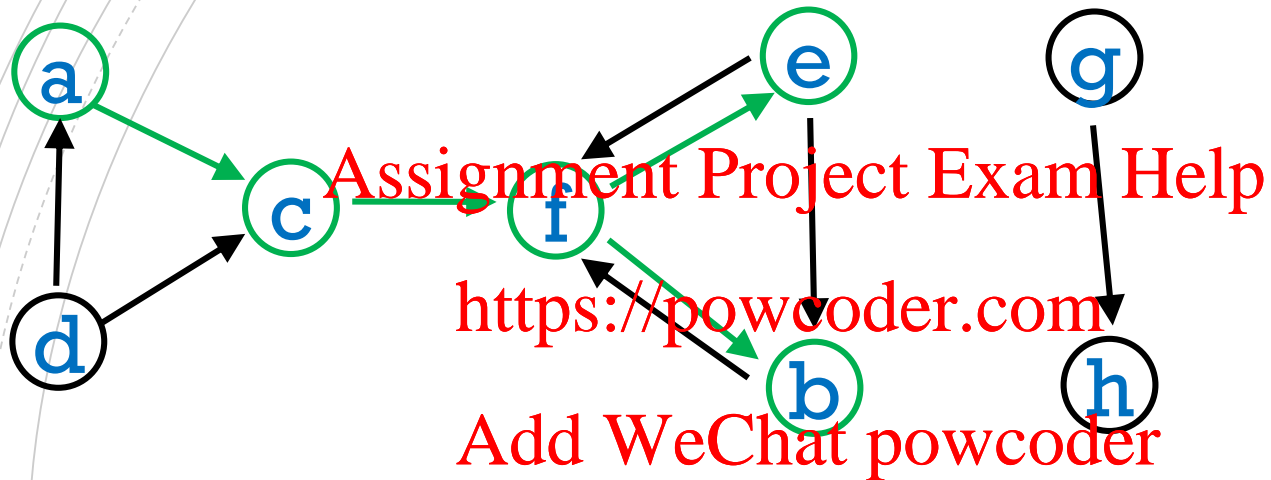
CALL STACK FOR depthFirst(a)



			b	
		f	f	f
	c	c	c	c
a	a	a	a	a

```
depthFirst_Graph (v) {  
    v.visited = true  
    for each w s.t. (v,w) is in E  
        if !(w.visited)  
            depthFirst_Graph(w)  
}
```

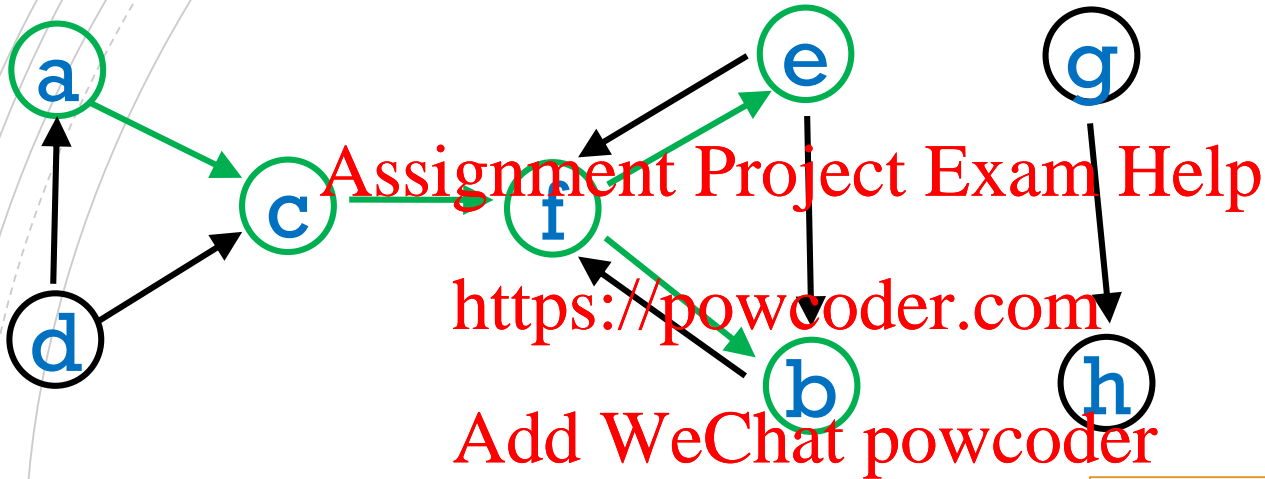
CALL STACK FOR depthFirst(a)



			b		e
		f	f	f	f
	c	c	c	c	c
a	a	a	a	a	a

```
depthFirst_Graph (v) {  
    v.visited = true  
    for each w s.t. (v,w) is in E  
        if !(w.visited)  
            depthFirst_Graph(w)  
}
```

CALL STACK FOR depthFirst(a)



Assignment Project Exam Help

<https://powcoder.com>

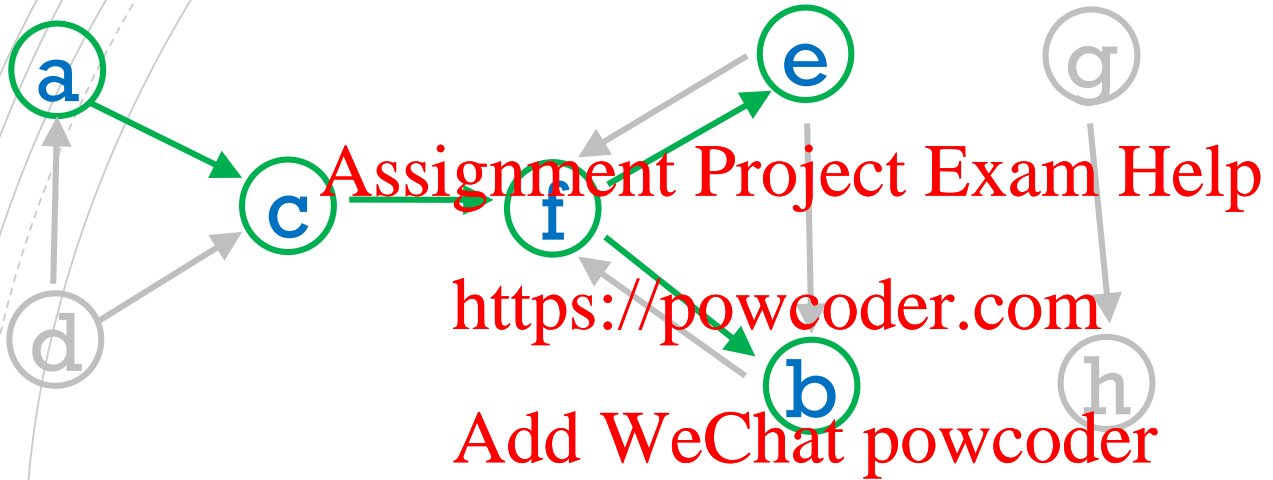
Add WeChat powcoder

```
depthFirst_Graph (v) {
    v.visited = true
    for each w s.t. (v,w) is in E
        if !(w.visited)
            depthFirst_Graph (w)
}
```

The diagram illustrates a sequence of points labeled 'a' through 'a' along a horizontal axis. Above the axis, there are several curved lines representing a path or boundary. The letters 'b', 'e', 'f', 'c', and 'a' are placed at various points along the axis and above it, indicating a specific sequence or mapping.

CALL TREE

root



The diagram illustrates the construction of a Huffman tree for the sequence "a b f f f f f a a a a a a a". The tree is built from the bottom up, starting with individual characters and merging them into nodes labeled 'a', 'c', 'f', and 'b'.

- Level 1 (Bottom):** Individual characters: a, a, a, a, a, a, a, a, a, a.
- Level 2:** Pairs of 'a' characters are merged into nodes labeled 'a'. Specifically, the first two 'a's merge into an 'a' node, the next two 'a's merge into an 'a' node, and the next two 'a's merge into an 'a' node. The remaining three 'a's are not yet merged.
- Level 3:** The 'a' nodes from Level 2 are merged into nodes labeled 'c'. Specifically, the first two 'a' nodes merge into a 'c' node, and the next two 'a' nodes merge into a 'c' node. The remaining 'a' node is not yet merged.
- Level 4:** The 'c' nodes from Level 3 are merged into nodes labeled 'f'. Specifically, the first 'c' node and the second 'c' node merge into an 'f' node. The remaining 'a' node is not yet merged.
- Level 5:** The 'f' node from Level 4 and the remaining 'a' node from Level 3 merge into a node labeled 'b'.
- Level 6 (Top):** The 'b' node and the remaining 'a' node from Level 1 merge into the root node labeled 'e'.

GRAPH TRAVERSALS

- Unlike tree traversal for rooted tree, a graph traversal started from some arbitrary vertex does not necessarily reach all other vertices.

Assignment Project Exam Help

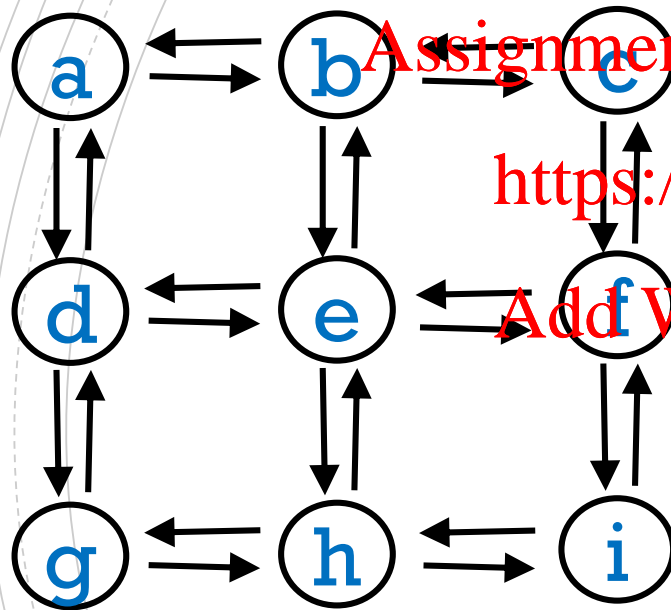
- *Knowing which vertices can be reached by a path from some starting vertex is itself an important problem. You will learn about such graph 'connectivity' problems in COMP 251.*

<https://powcoder.com>

Add WeChat powcoder

- The order of nodes visited depends on the order of nodes in the adjacency lists.

EXAMPLE 2



Adjacency List

a - (b,d)

b - (a,c,e)

c - (b,f)

d - (a,e,g)

e - (b,d,f,h)

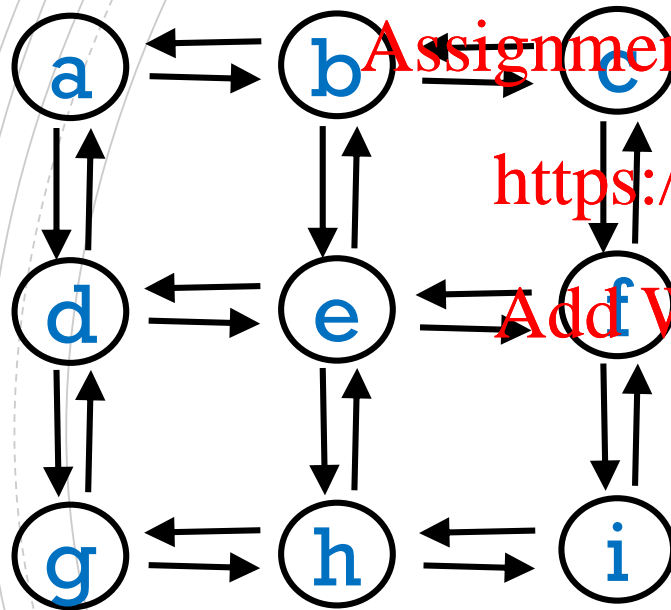
f - (c,e,i)

g - (d,h)

h - (e,g,i)

i - (f,h)

EXAMPLE 2



Assignment Project Exam Help

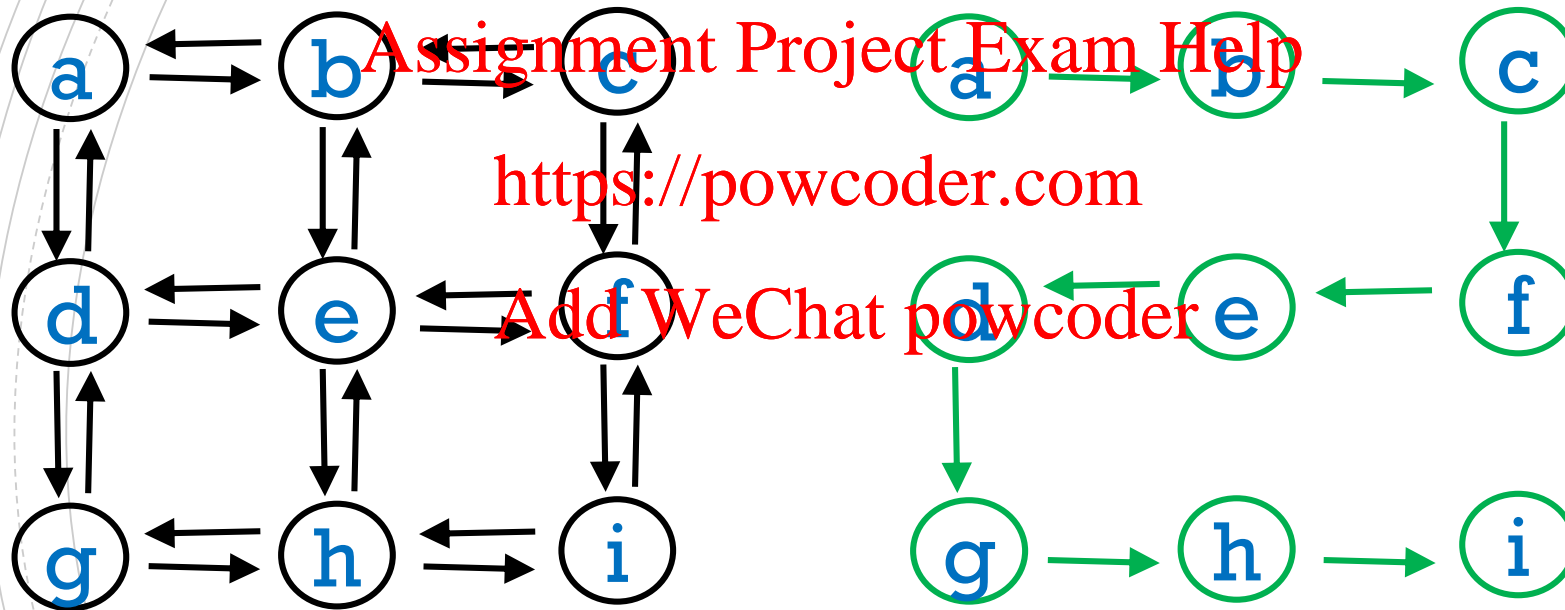
<https://powcoder.com>

Add WeChat powcoder

What is the call tree
for **depthFirst**(a) ?

(Do it in your head)

EXAMPLE 2



call tree for `depthFirst(a)`

GRAPH TRAVERSALS

- Q: Can we do non-recursive graph traversals?

<https://powcoder.com>

Add WeChat powcoder

GRAPH TRAVERSALS

- Q: Can we do non-recursive graph traversals?

<https://powcoder.com>

- A: Yes, similar to tree traversal; use a stack or a queue.

RECALL: DEPTH FIRST TREE TRAVERSAL (WITH A SLIGHT VARIATION)

```
treeTraversalUsingStack(root) {  
    initialize empty stack s  
    s.push(root)  
    while s is not empty {  
        cur = s.pop()  
        visit cur  
        for each child of cur {  
            s.push(child)  
        }  
    }  
}
```

Visit a node *after*
popping it from the
stack.

Every node in the tree
gets pushed, and
popped, and visited.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

GENERALIZE TO GRAPH

```
graphTraversalUsingStack(v) {  
    initialize empty stack s  
    v.visited = true  
    s.push(v)  
    while s is not empty {  
        cur = s.pop()  
        visit cur // do something  
        for each w in cur.adjList  
            if(!w.visited) {  
                w.visited = true  
                s.push(w)  
            }  
        }  
    }  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

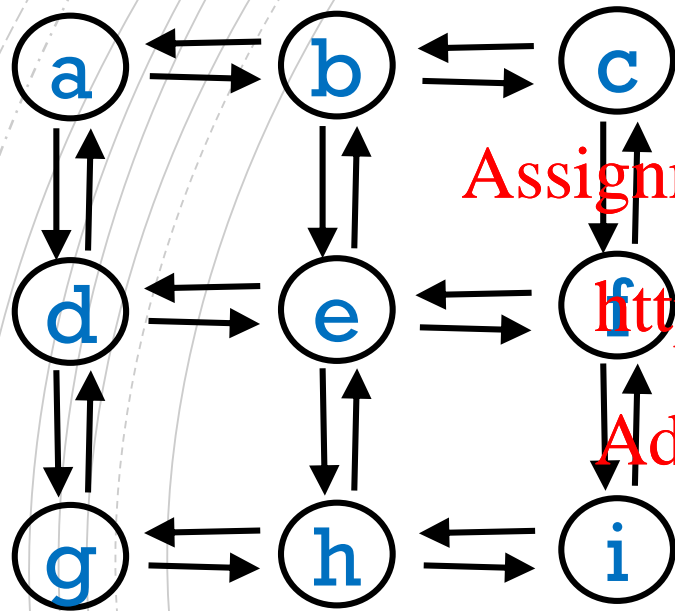
Indicate as “reached” a node *before pushing* it onto the stack. We do that by updating the field `visited`.

Visit the node (perform some operations) after it gets popped from the stack.

Every node in the graph gets *reached*, pushed, popped, and visited.

EXAMPLE: graphTraversalUsingStack(a)

Order of visit: **a**



a

Assignment Project Exam Help

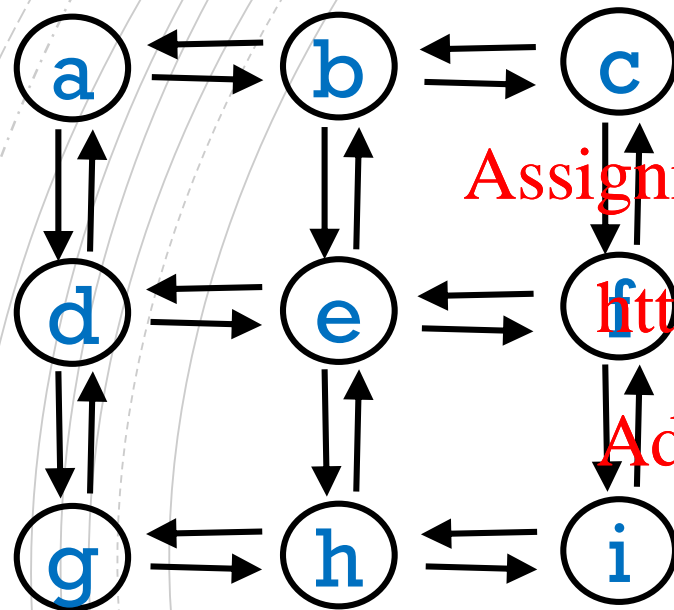
<https://powcoder.com>

Add WeChat powcoder

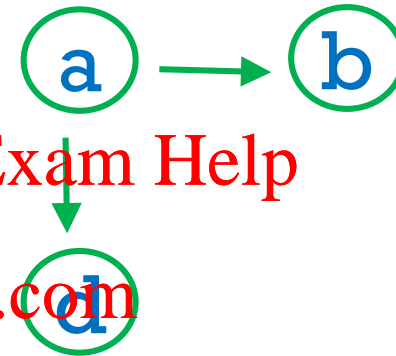
a



EXAMPLE: graphTraversalUsingStack(a)



Order of visit: a



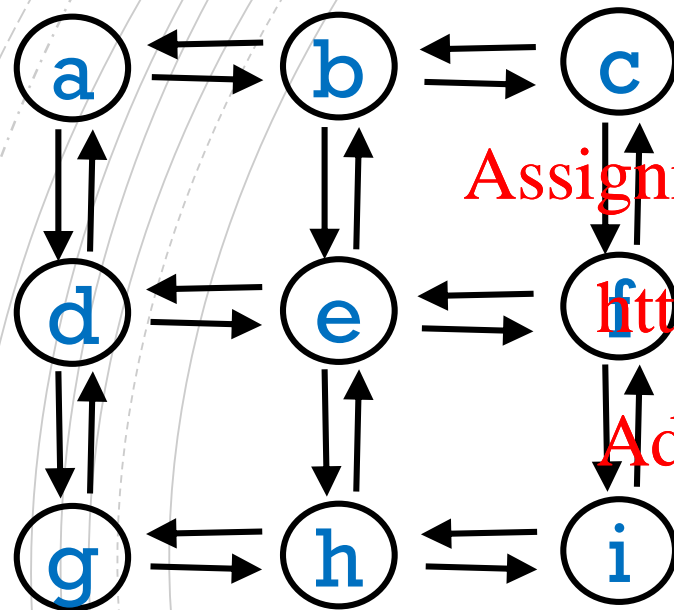
Assignment Project Exam Help

<https://powcoder.com>

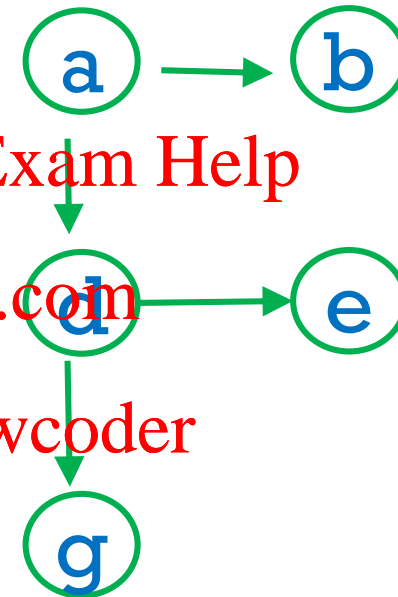
Add WeChat powcoder

a _ b b d

EXAMPLE: graphTraversalUsingStack(a)



Order of visit: **ad**



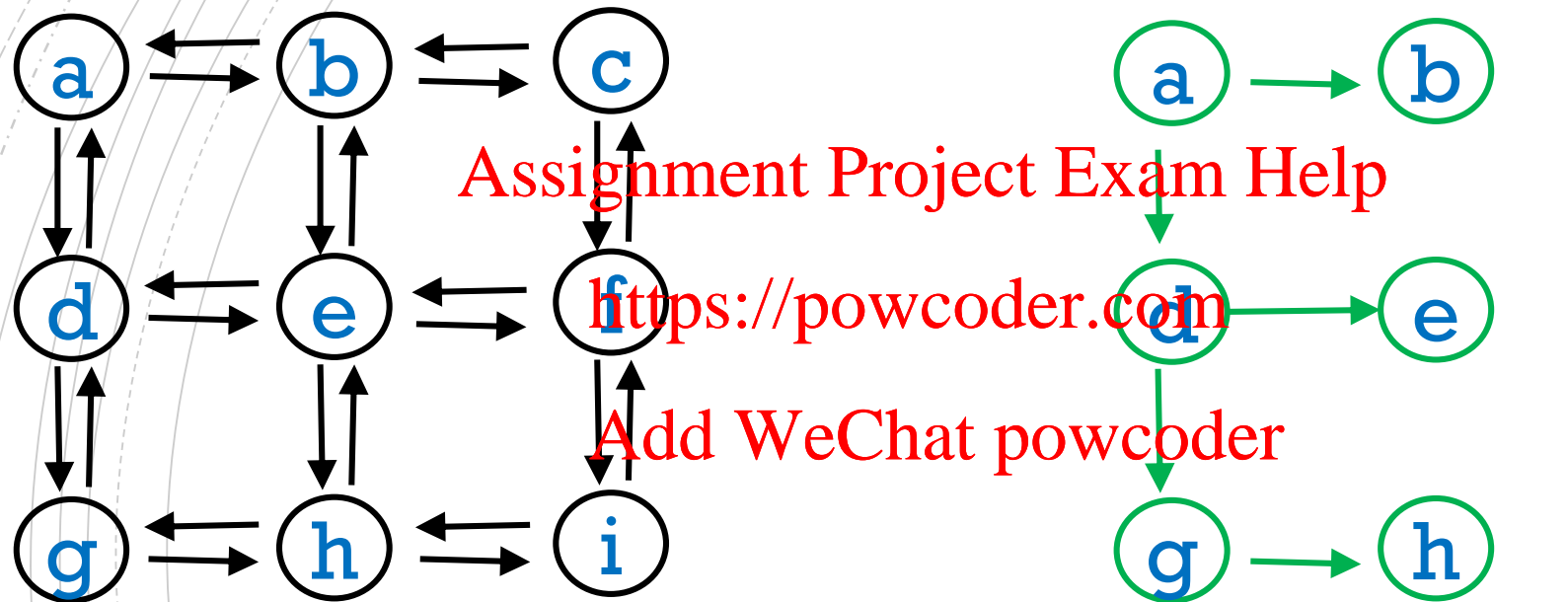
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

a _ b d g
b b e
b b

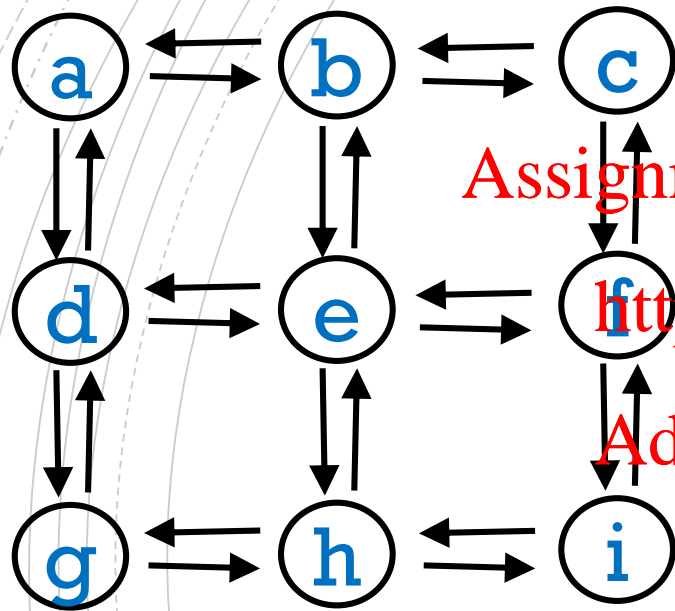
EXAMPLE: graphTraversalUsingStack(a)



a _ b d g h
b b e e e
b b b b b

EXAMPLE: graphTraversalUsingStack(a)

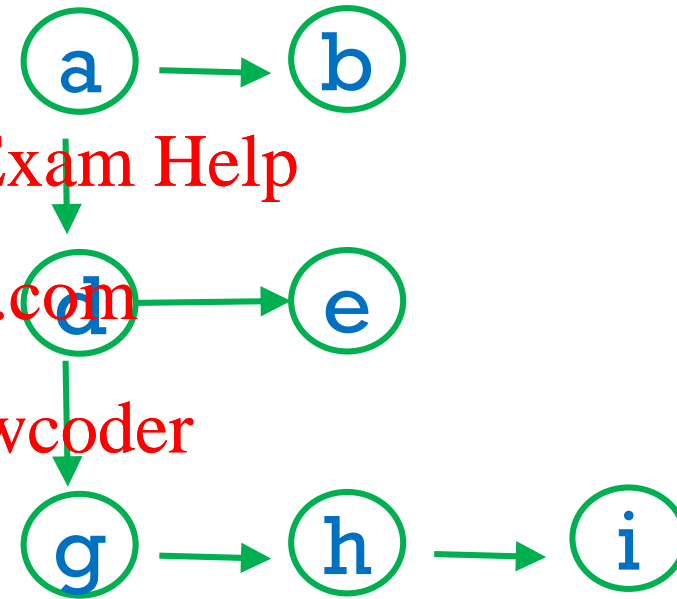
Order of visit: **adgh**



Assignment Project Exam Help

<https://powcoder.com>

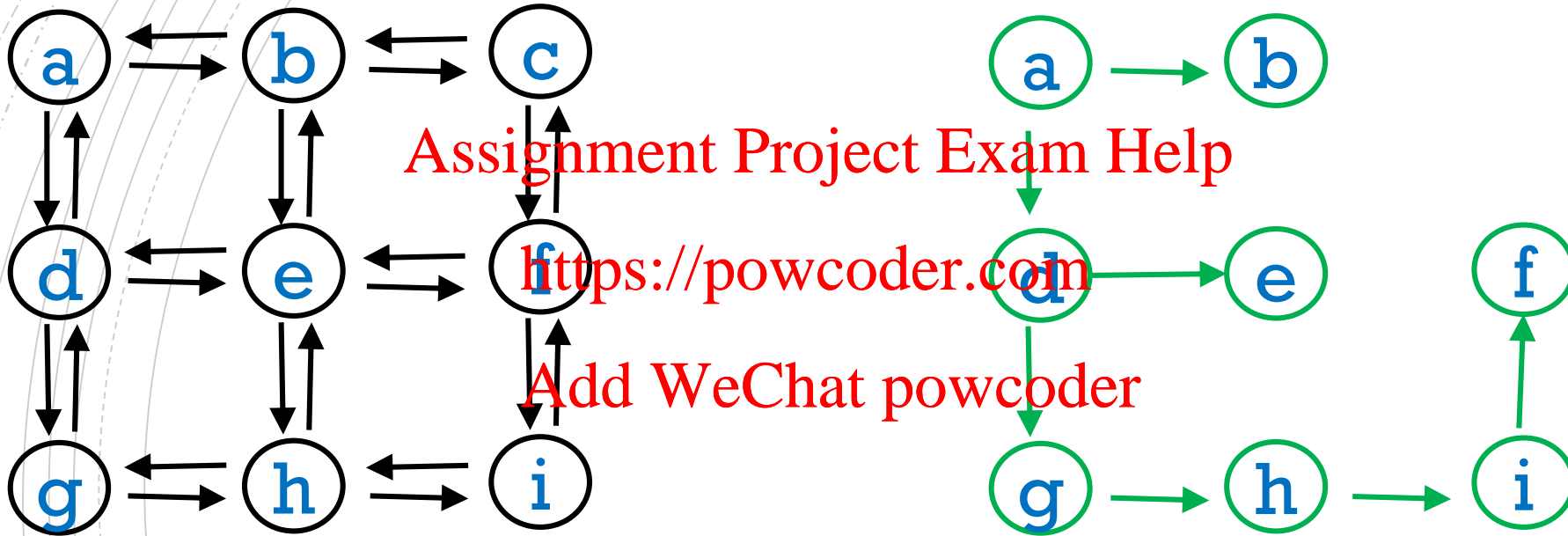
Add WeChat powcoder



				g		h		i
		d		e	e	e	e	e
a	_	b	b	b	b	b	b	b

EXAMPLE: graphTraversalUsingStack(a)

Order of visit: **adghi**



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

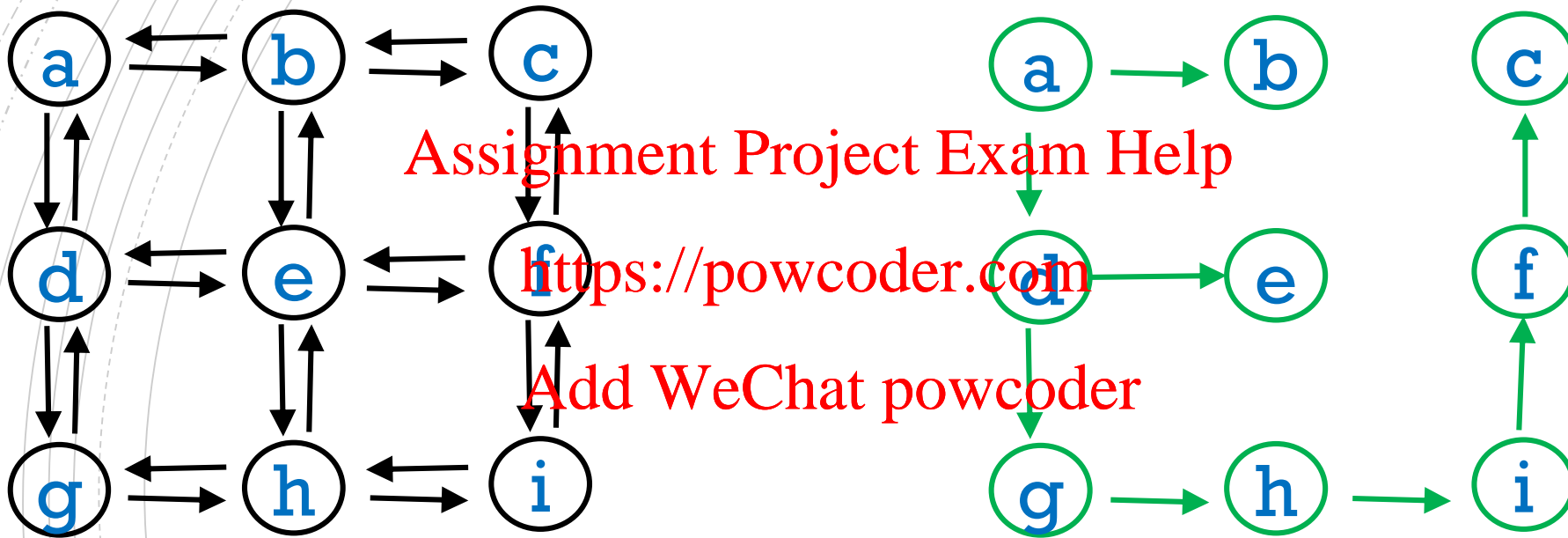
g h i f

d e e e e e e e

a b b b b b b b b b

EXAMPLE: graphTraversalUsingStack(a)

Order of visit: **adghif**



Assignment Project Exam Help

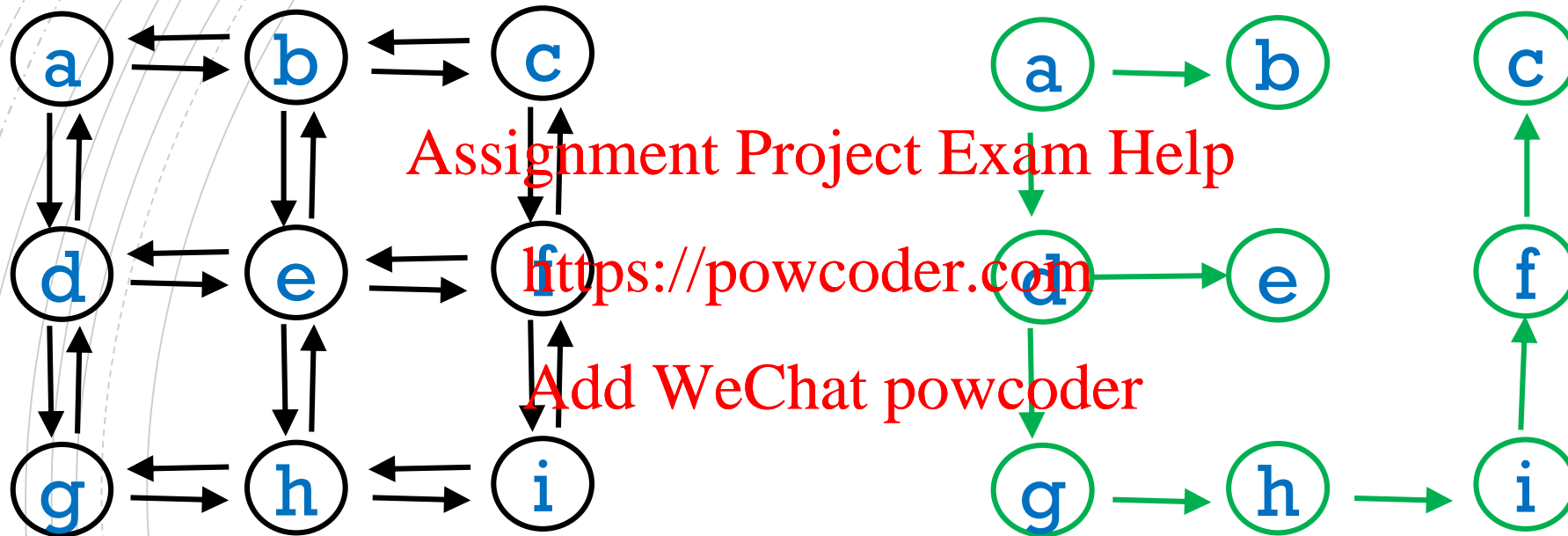
<https://powcoder.com>

Add WeChat powcoder

The diagram illustrates a 3D coordinate system with three axes: a vertical axis labeled 'a', a horizontal axis labeled 'b', and a diagonal axis labeled 'c'. A point is marked with coordinates (g, h, i, f, c) in the 5D space. The coordinates are arranged in a grid-like fashion, with 'g' at the top, followed by 'h', 'i', 'f', and 'c' in the next row. Below these, the letters 'd', 'e', 'e', 'e', 'e', 'e', 'e', 'e', 'e', 'e' are arranged in a single row. At the bottom, the letters 'a', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b', 'b' are arranged in a single row. The 'a' axis is represented by a solid line, while the 'b' and 'c' axes are represented by dashed lines.

EXAMPLE: graphTraversalUsingStack(a)

Order of visit: **adghifceb**



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

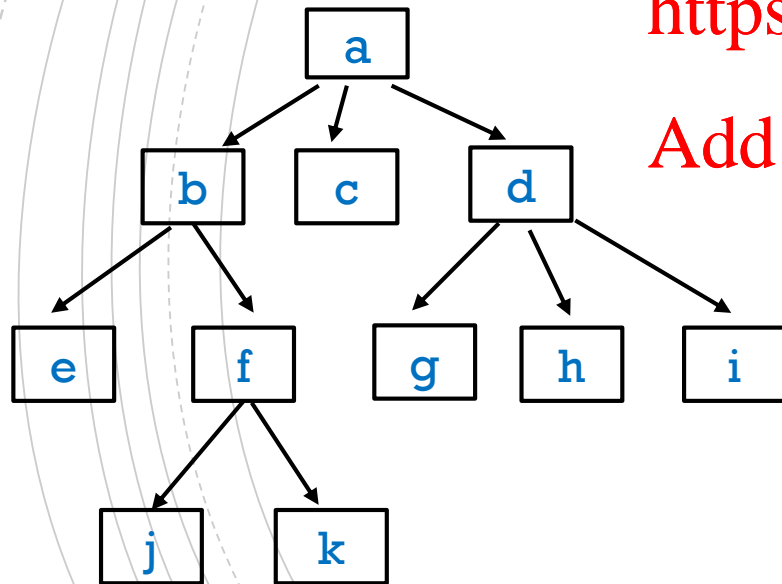
a _ b b b b b b b b b b b b b _

g h i f c

d e e e e e e e e e e

RECALL: BREADTH FIRST TREE TRAVERSAL

for each level i
visit all nodes at level i



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
treeTraversalUsingQueue(root) {  
    initialize empty queue q  
    q.enqueue(root)  
    while q is not empty {  
        cur = q.dequeue()  
        visit cur  
        for each child of cur  
            q.enqueue(child)  
    }  
}
```

BREADTH FIRST GRAPH TRAVERSAL

Given an input vertex, visit all vertices that can be reached by paths of length 1, 2, 3, 4,.....

<https://powcoder.com>

Add WeChat powcoder

BREADTH FIRST GRAPH TRAVERSAL

```
graphTraversalUsingQueue(v) {  
    initialize empty queue q  
    v.visited = true  
    q.enqueue(v)  
    while q is not empty {  
        cur = q.dequeue()  
        for each w in cur.adjList {  
            if(!w.visited) {  
                w.visited = true  
                q.enqueue(w)  
            }  
        }  
    }  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

EXAMPLE

graphTraversalUsingQueue(c)

queue

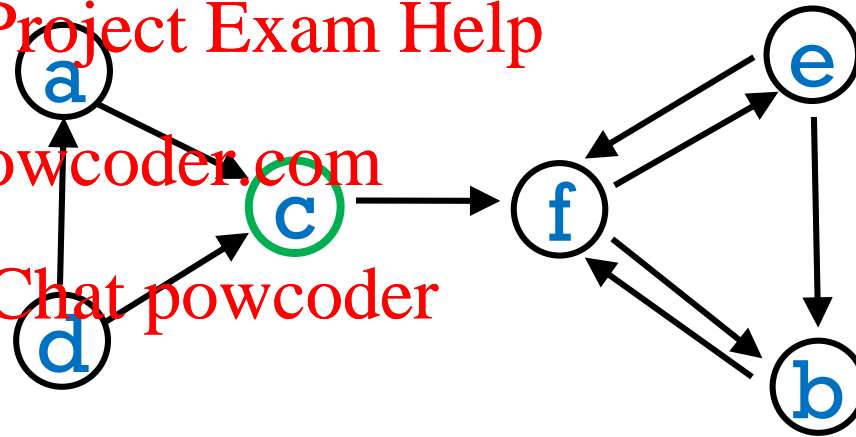
c



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



EXAMPLE

graphTraversalUsingQueue(c)

queue

c

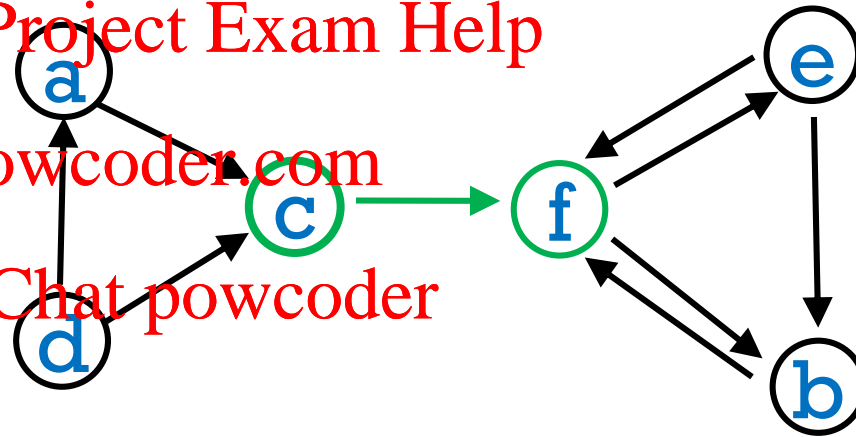
f



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



EXAMPLE

graphTraversalUsingQueue(c)

queue

c

f

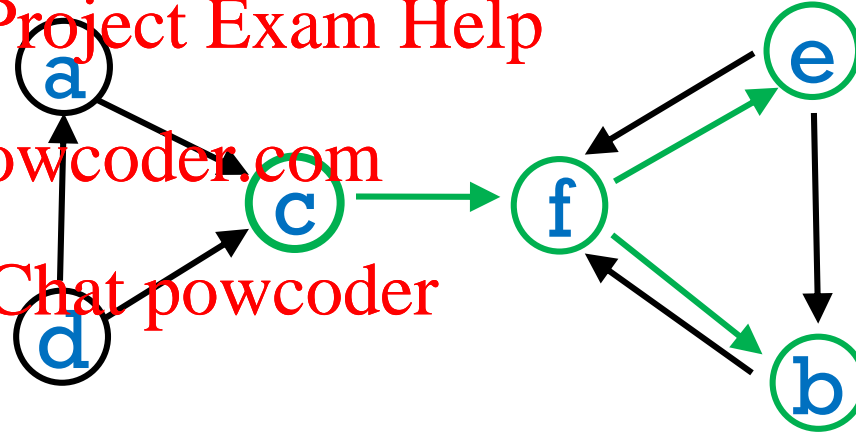
b
e

Both 'b', 'e' are visited
and enqueued before
'b' is dequeued.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



EXAMPLE

graphTraversalUsingQueue(c)

queue

c

f

b

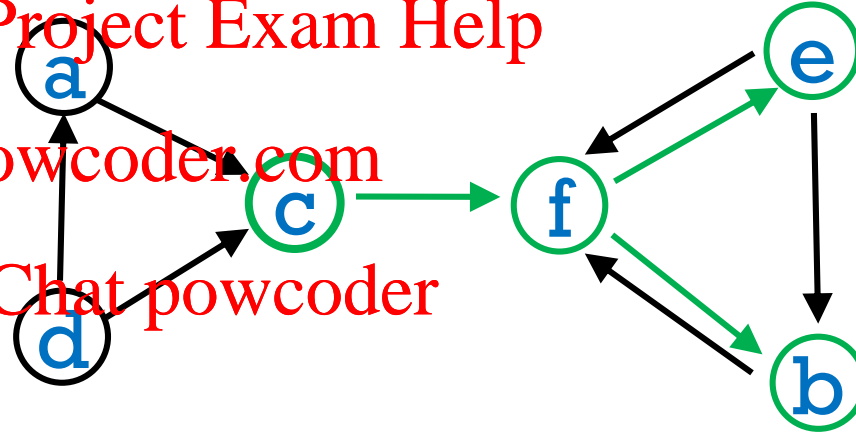
e

—

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



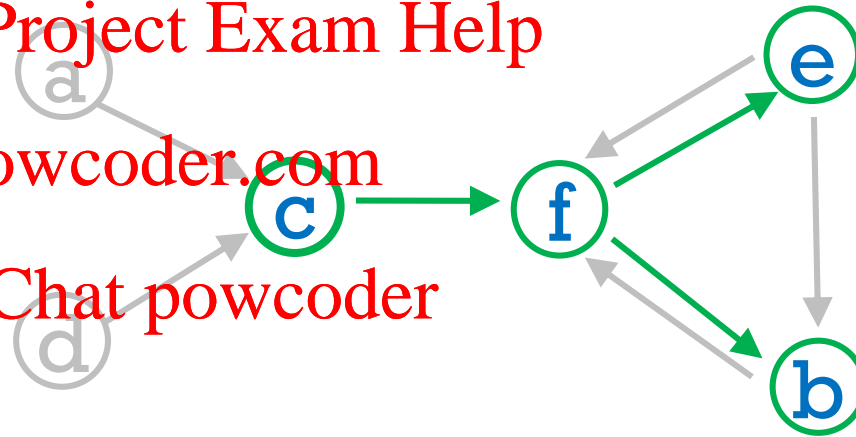
EXAMPLE

`graphTraversalUsingQueue(c)`

Assignment Project Exam Help

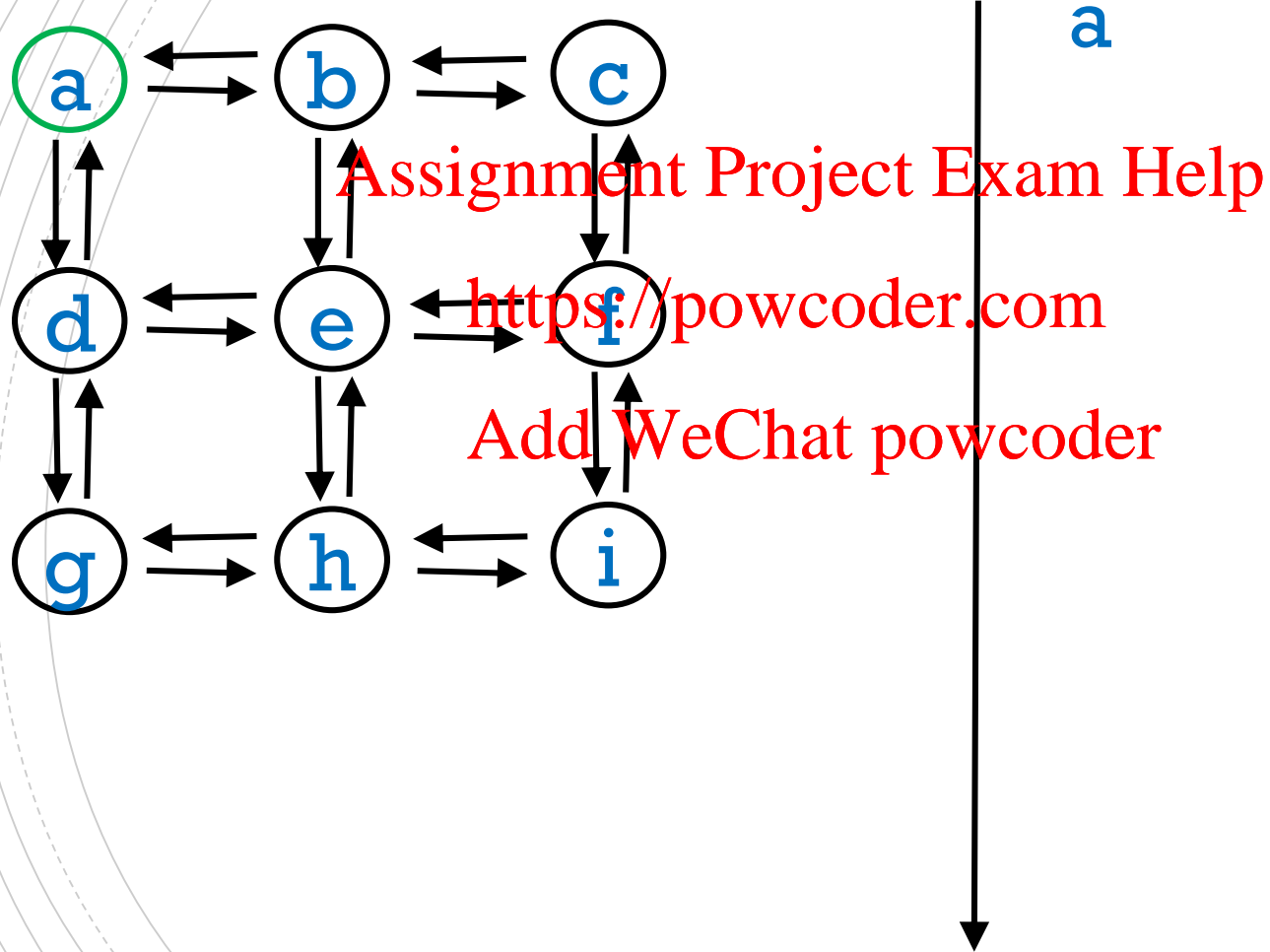
<https://powcoder.com>

Add WeChat powcoder

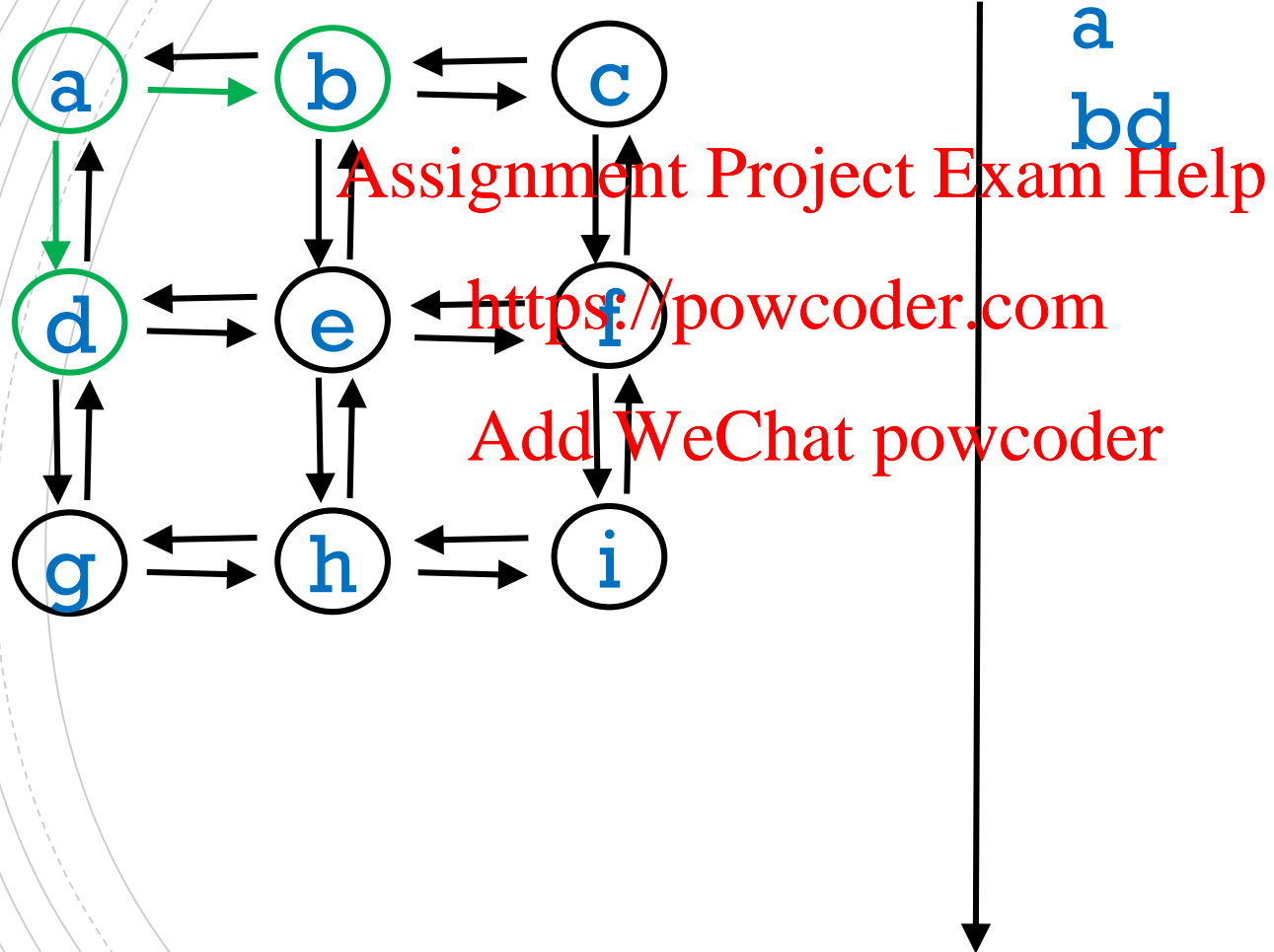


It defines a tree whose root is the starting vertex.
It finds the shortest path (number of edges) to all vertices reachable from the starting vertex.

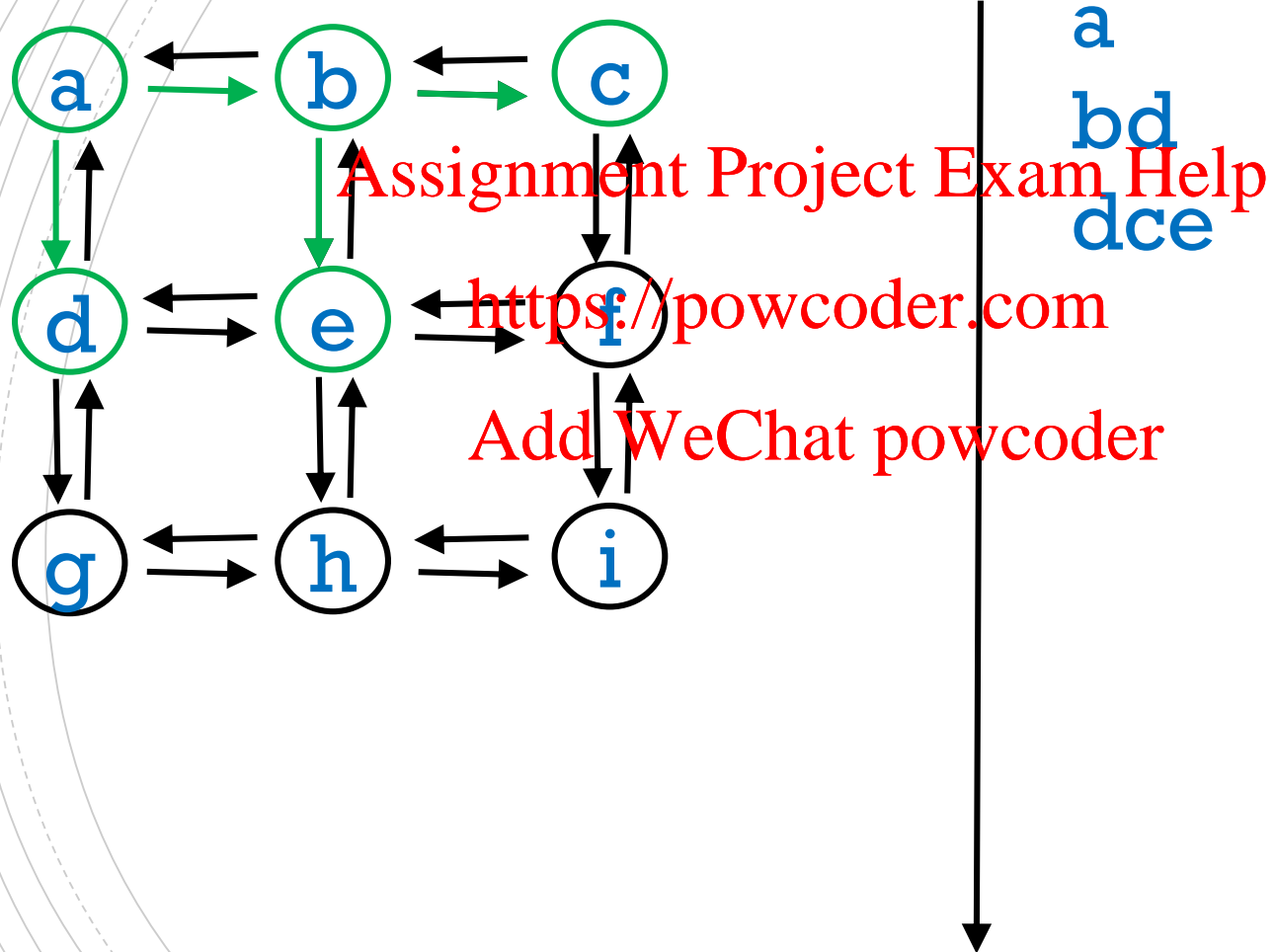
EXAMPLE: graphTraversalUsingQueue(a)



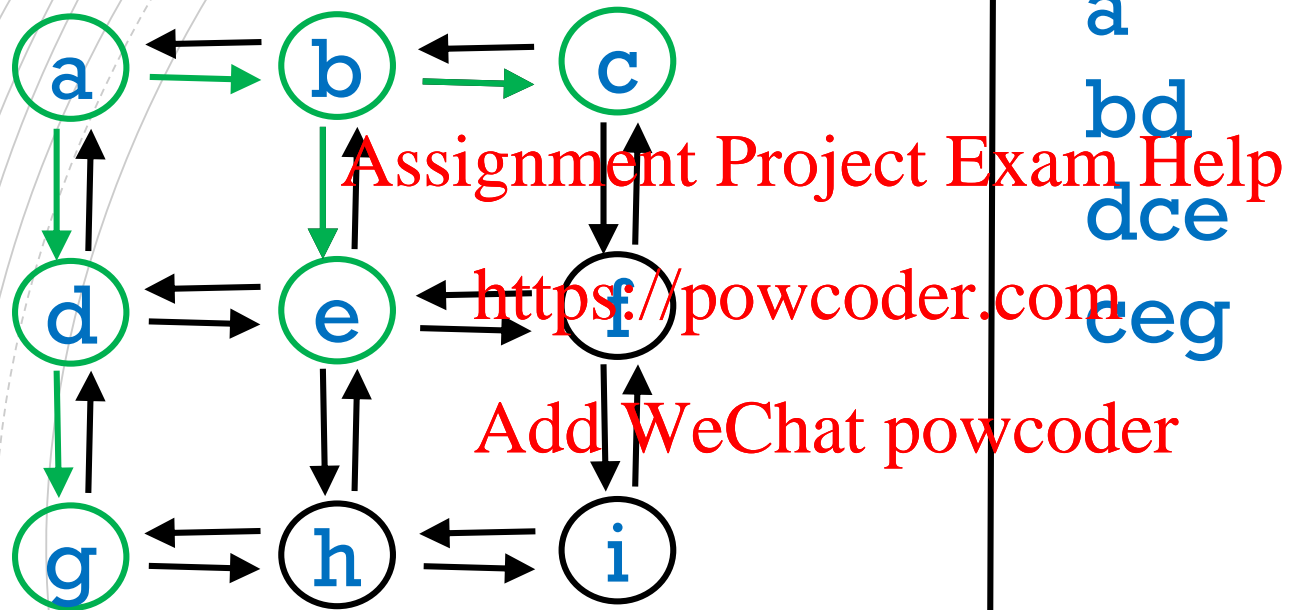
EXAMPLE: graphTraversalUsingQueue(a)



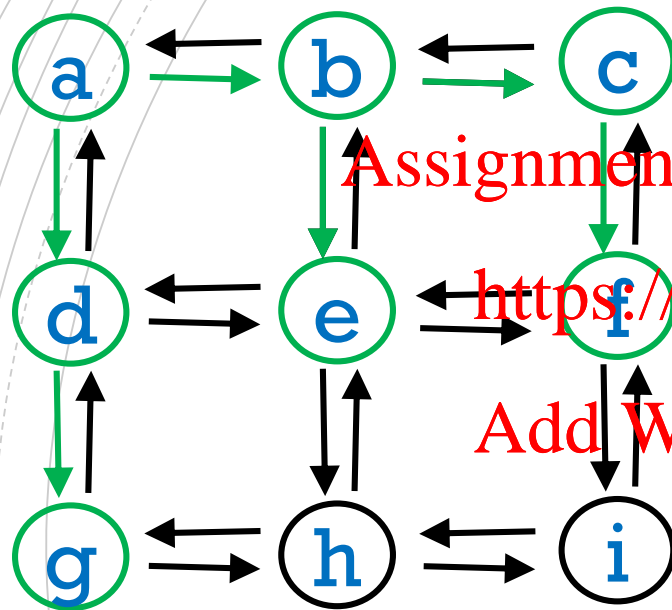
EXAMPLE: graphTraversalUsingQueue(a)



EXAMPLE: graphTraversalUsingQueue(a)



EXAMPLE: graphTraversalUsingQueue(a)



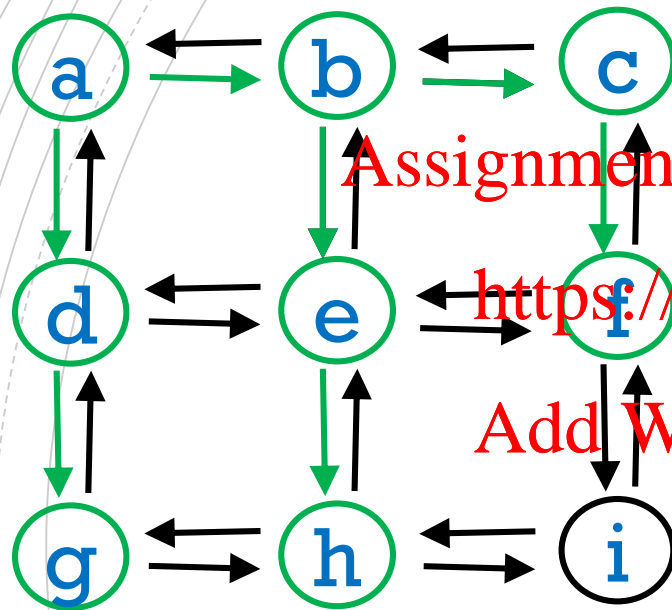
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

a
bd
dce
ceg
egf

EXAMPLE: graphTraversalUsingQueue(a)



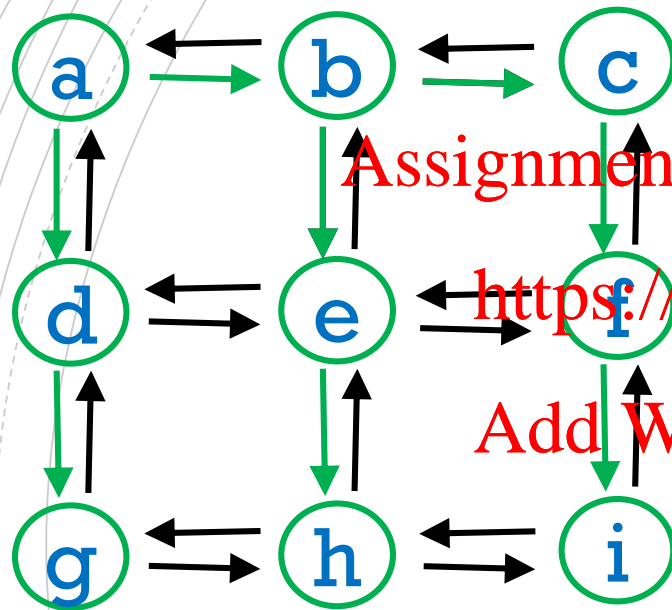
a
bd
dce
ceg
egf
gfh

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

EXAMPLE: graphTraversalUsingQueue(a)



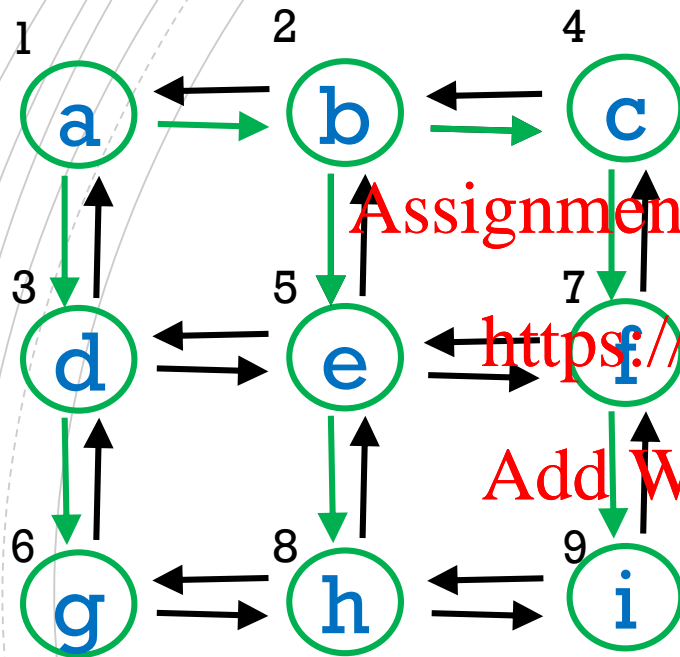
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

a
bd
dce
ceg
egf
gfh
fh
hi

EXAMPLE: graphTraversalUsingQueue(a)



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Note order of nodes visited: We get paths of length 1, the paths of length 2, etc. i.e. breadth first.

a
bd
dce
ceg
egf
gfh
fh
hi
i
_

RECALL: HOW TO IMPLEMENT A GRAPH CLASS IN JAVA?

```
class Graph<T> {  
    ArrayList<Vertex<T>> vetexList;  
  
    class Vertex<T> {  
        ArrayList<Edge> adjList;  
        T element;  
        boolean visited;  
    }  
  
    class Edge {  
        Vertex endVertex;  
        double weight;  
        :  
    }  
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

PRIOR TO TRAVERSAL!

```
for each w in V
```

```
  w.visited = false
```

Assignment Project Exam Help

<https://powcoder.com>

How should we implement this?

Add WeChat powcoder

PRIOR TO TRAVERSAL!

```
for each w in V
```

```
    w.visited = false
```

Assignment Project Exam Help

```
class Graph<T> {  
    ArrayList<Vertex<T>> vetexList;  
    :  
    public void resetVisited() {  
  
    }  
}
```

<https://powcoder.com>

Add WeChat powcoder

PRIOR TO TRAVERSAL!

```
for each w in V
```

```
    w.visited = false
```

Assignment Project Exam Help

<https://powcoder.com>

```
class Graph<T> {
```

```
    ArrayList<Vertex<T>> vertexList;
```

```
    :
```

```
    public void resetVisited() {
```

```
        for(Vertex<T> v : vertexList)
```

```
            v.visited = false;
```

```
    }
```

```
}
```

Add WeChat powcoder

Thank You

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

