GRAPHS: SEARCH

David Kauchak
CS 62 – Spring 2020

Admin

Assignment 8

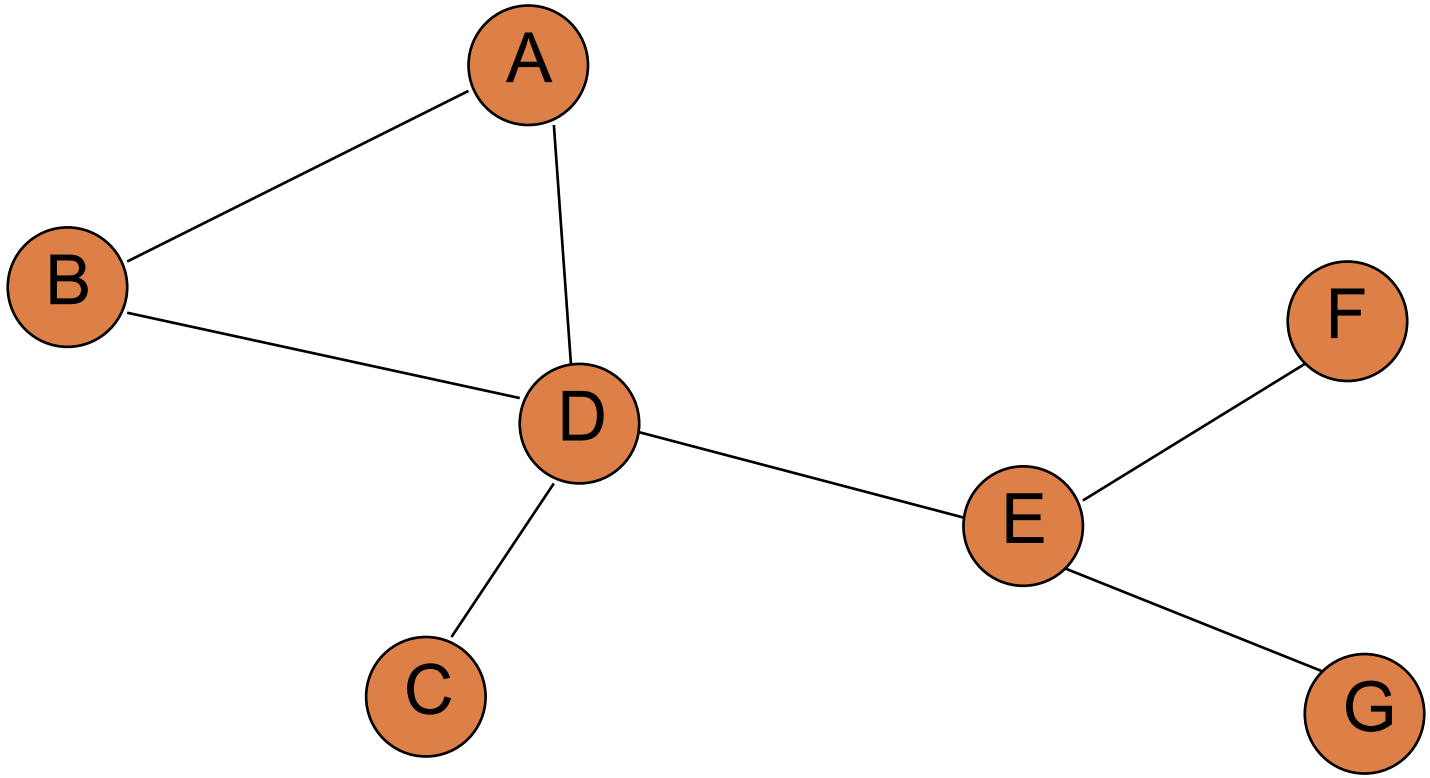
Lab tomorrow:

- Look at Java code examples of graph representations and graph algorithms
- Work session for assignment

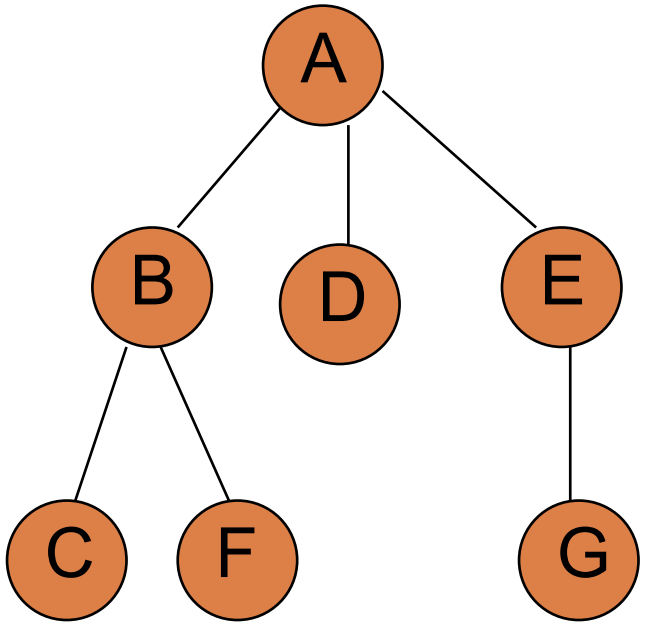
Final

Graphs

A graph is a set of vertices V and a set of edges (u,v) $\in$ E where u,v $\in$ V

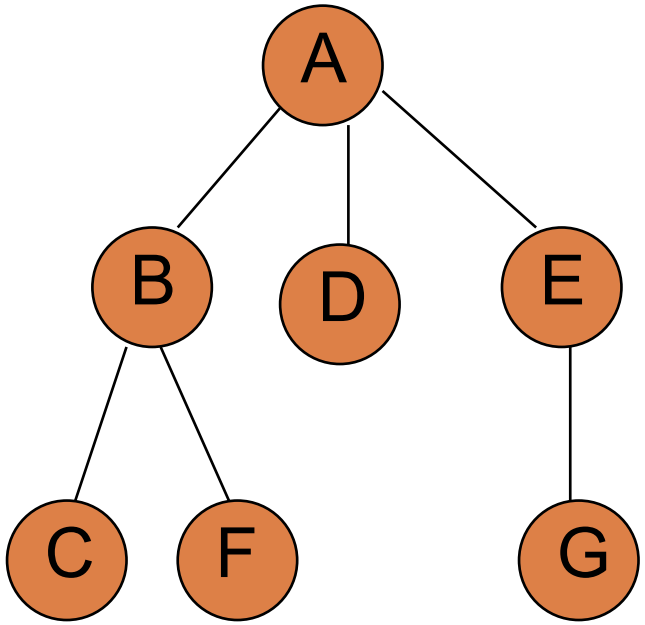


Searching a tree



How can we print out all of the vertices in a tree?

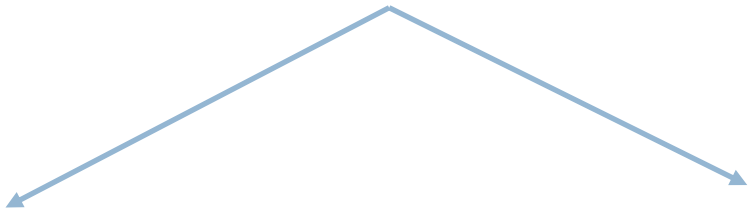
Searching a tree



We could do any of the traversals we saw before:
pre-order, in-order, post-order

A flash from the past

```
public interface Linear<E> {
    public void add(E item);
    public E remove();
    public E peek();
    public boolean empty();
}
```



Stack:

- LIFO
- Add to the back
- Remove from the **back**

Queue:

- FIFO
- Add to the back
- Remove from the **front**

Searching a tree

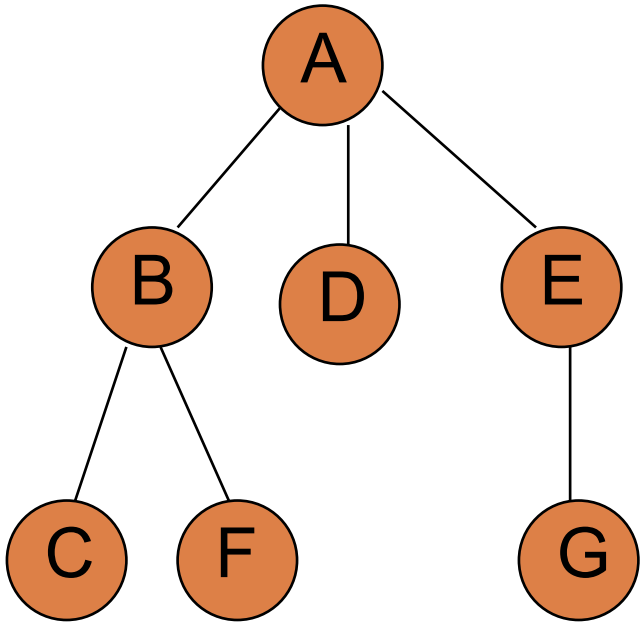
```
treeBFS( start )
    q = new Queue()
    q.add(start)
    treeSearch(q)
```

```
treeDFS( start )
    s = new Stack()
    s.add(start)
    treeSearch(s)
```

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

Tree BFS

```
treeBFS( start )
    q = new Queue()
    q.add(start)
    treeSearch(q)
```

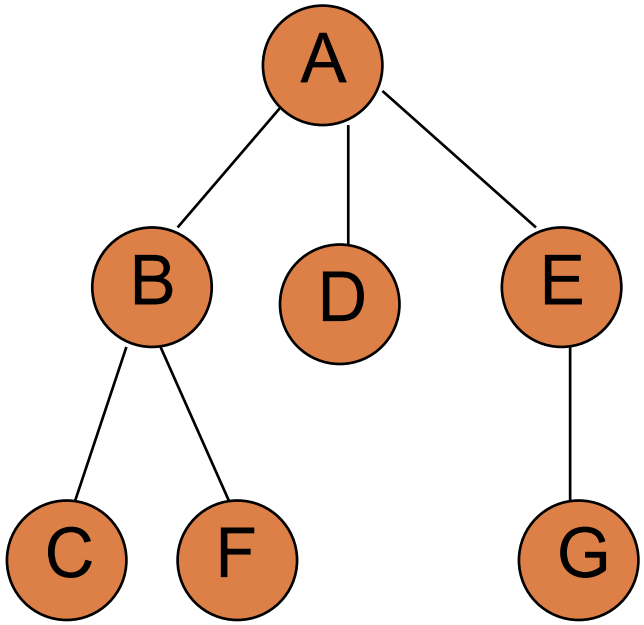


q:

Visited:

Tree BFS

```
treeBFS( start )
    q = new Queue()
    q.add(start)
    treeSearch(q)
```

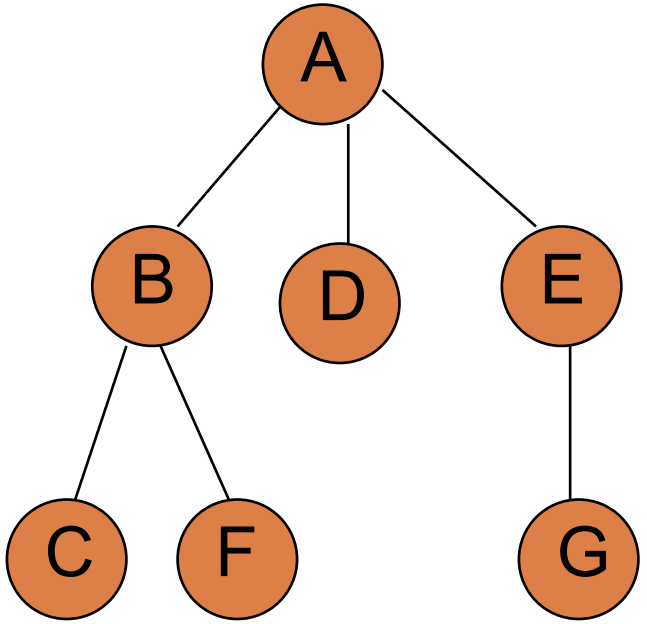


q: A

Visited:

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



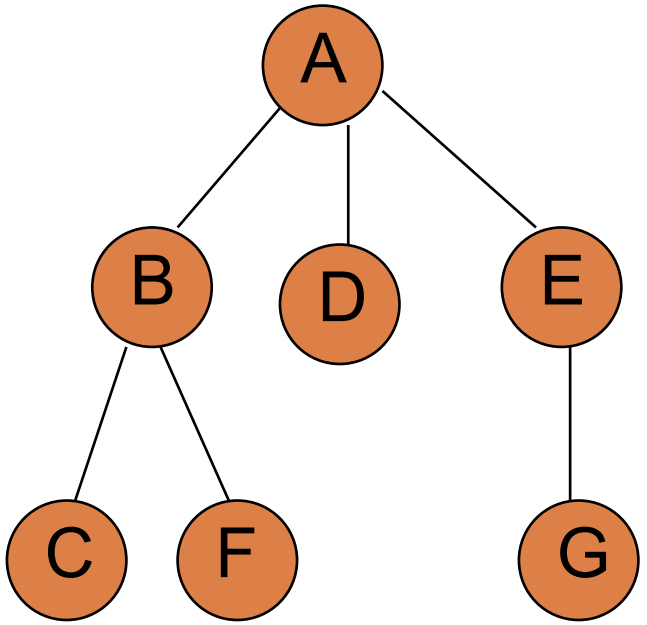
toVisit-queue: A

printed:

What order will the nodes get printed out?
Assume children are traversed left to right.

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

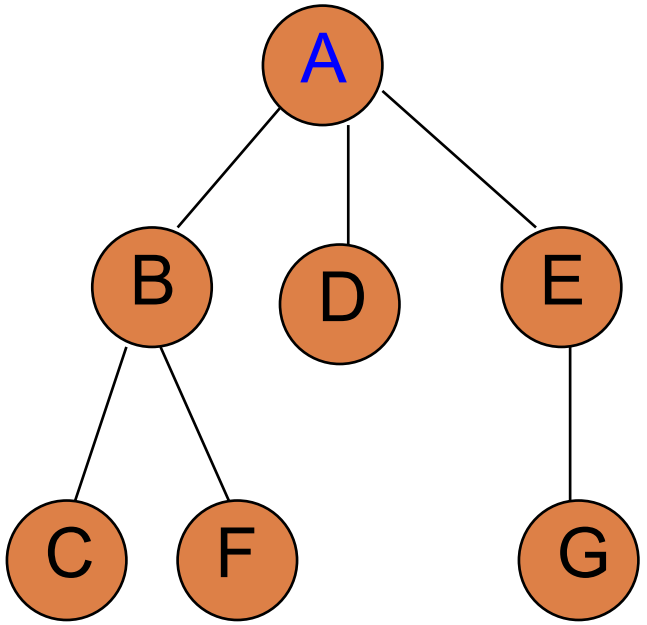


toVisit-queue: A

printed:

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

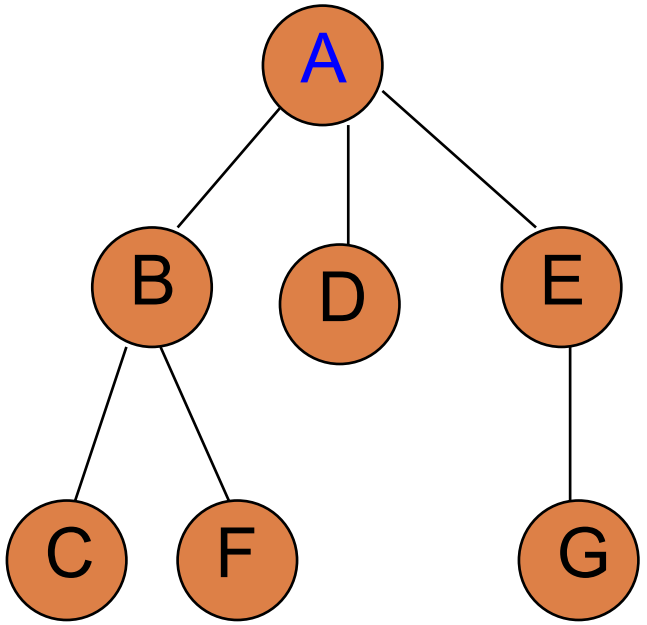


toVisit-queue:

printed: A

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

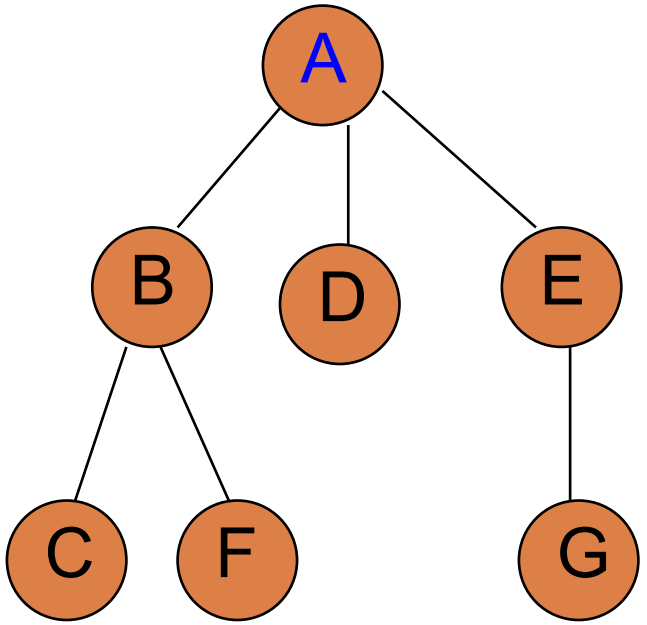


toVisit-queue:

printed: A

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

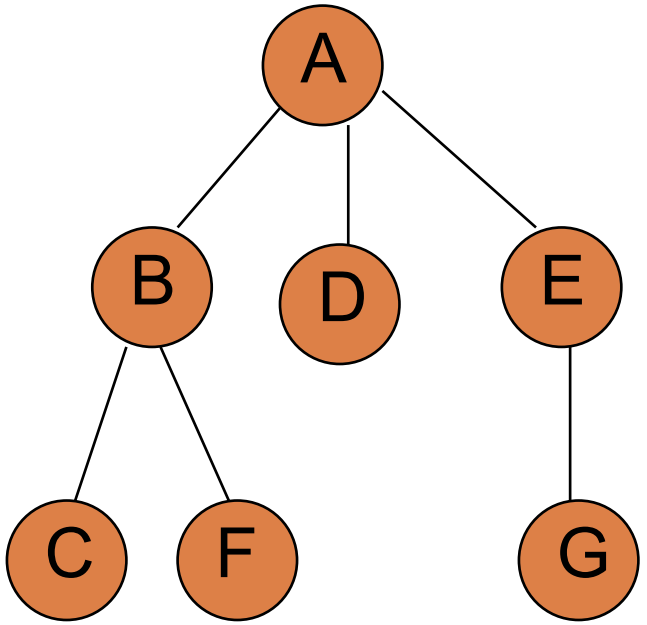


toVisit-queue: B D E

printed: A

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

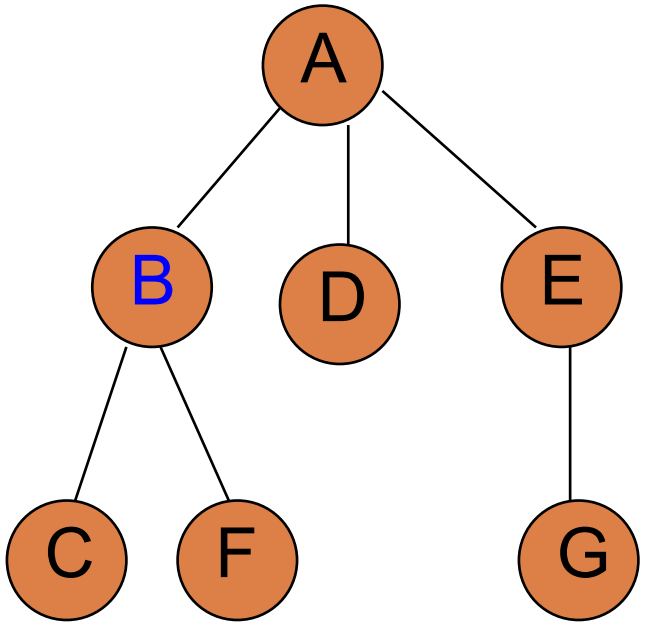


toVisit-queue: B D E

printed: A

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

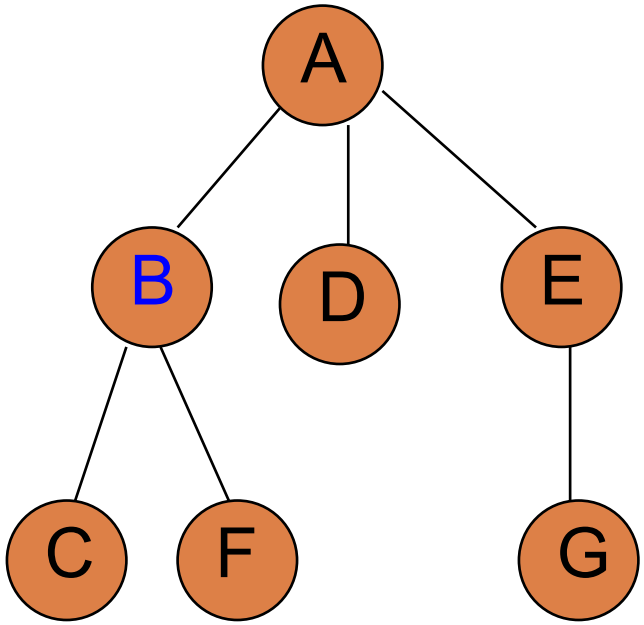


toVisit-queue: D E

printed: A B

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

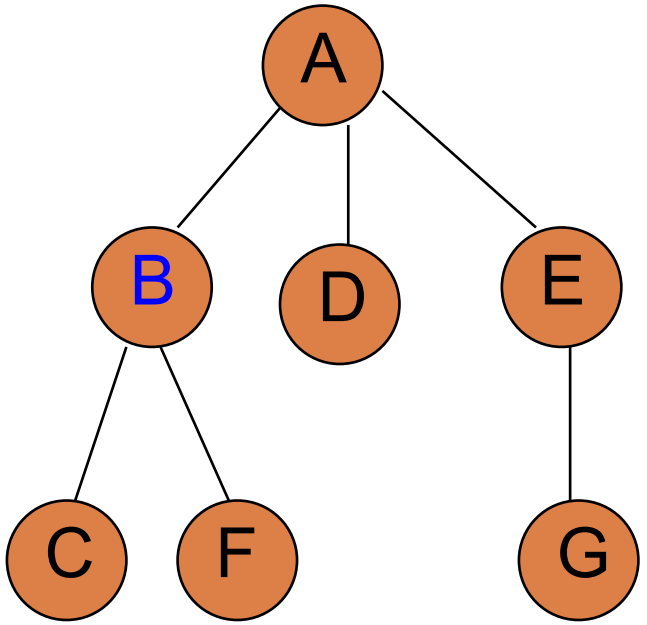


toVisit-queue: D E

printed: A B

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

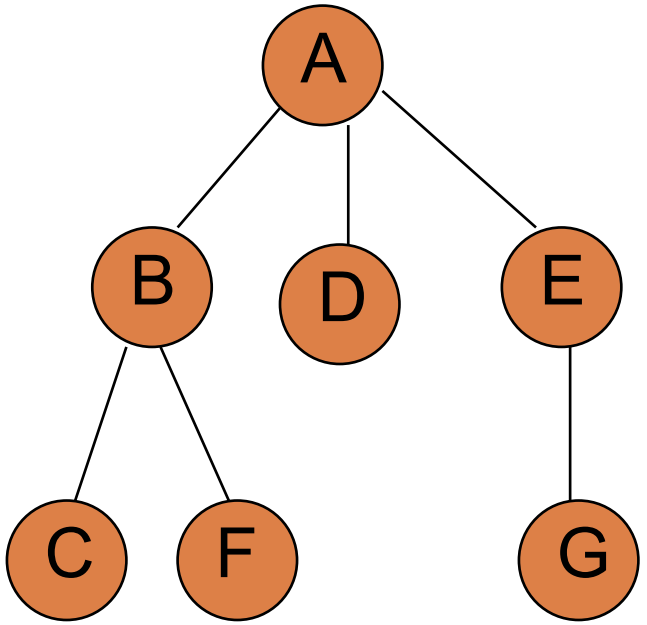


toVisit-queue: D E C F

printed: A B

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

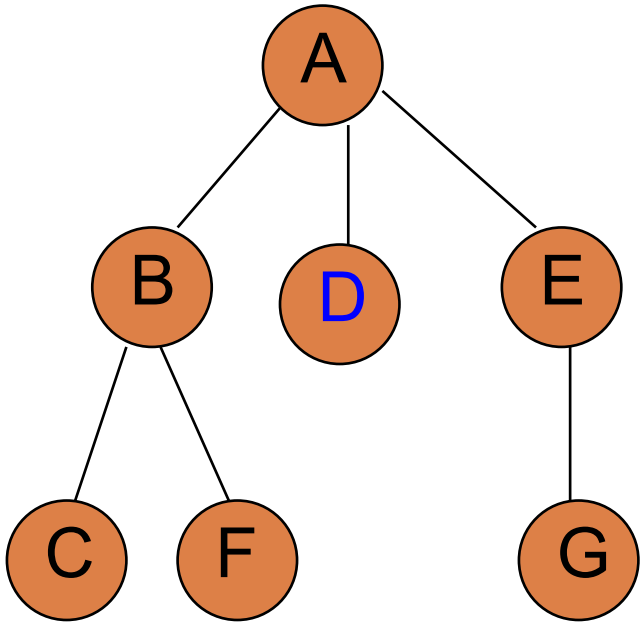


toVisit-queue: D E C F

printed: A B

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

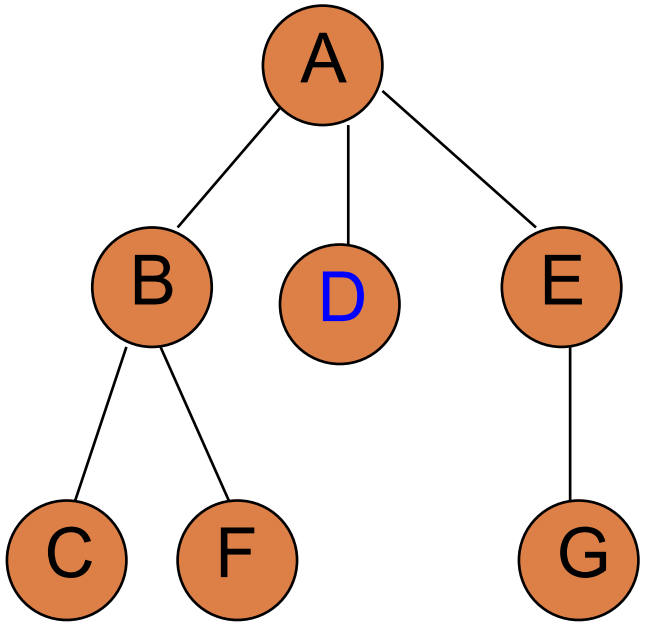


toVisit-queue: E C F

printed: A B D

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

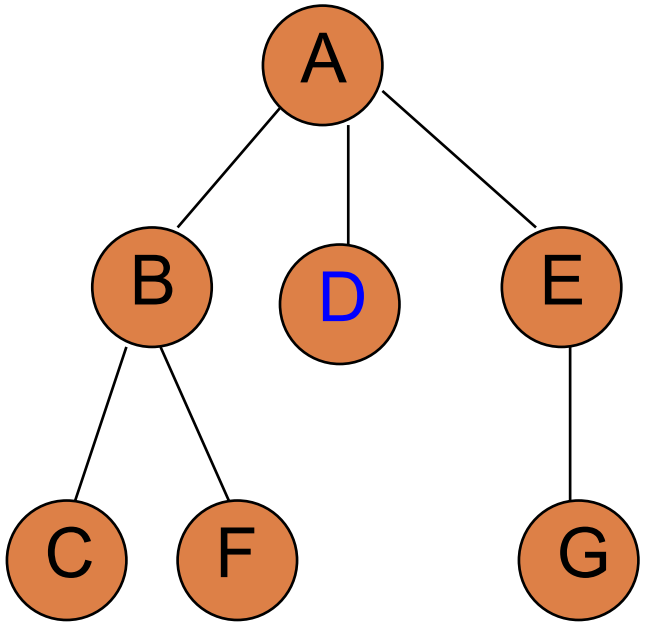


toVisit-queue: E C F

printed: A B D

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

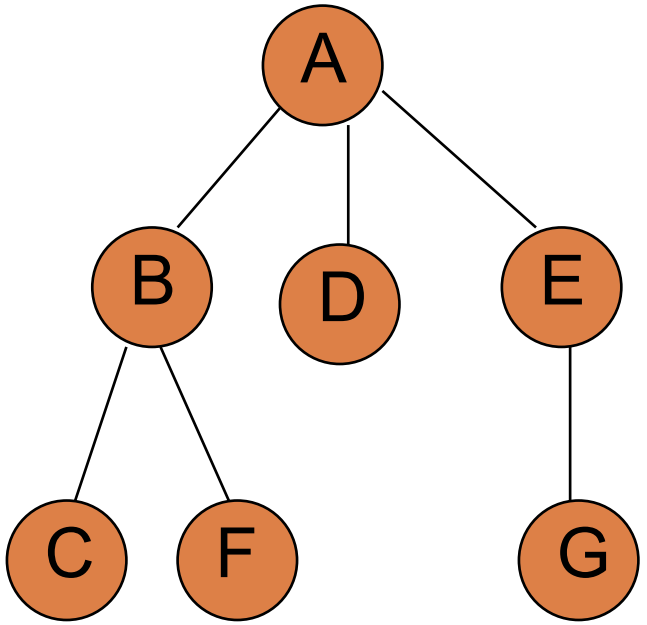


toVisit-queue: E C F

printed: A B D

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

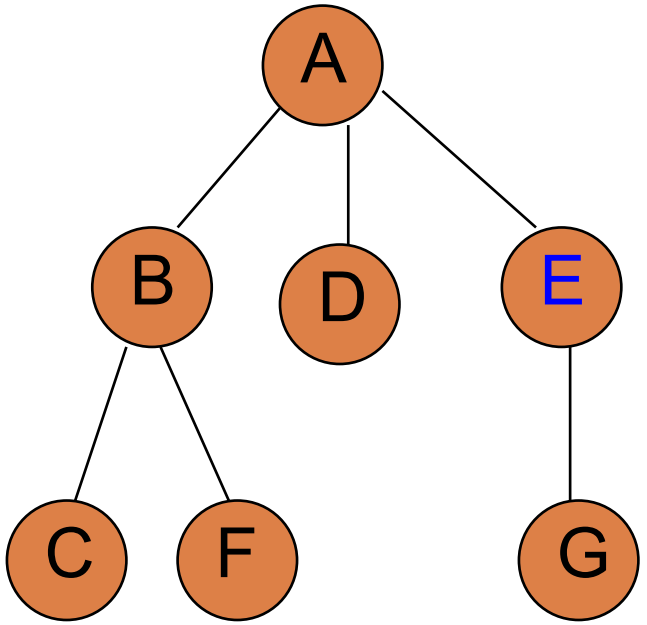


toVisit-queue: E C F

printed: A B D

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

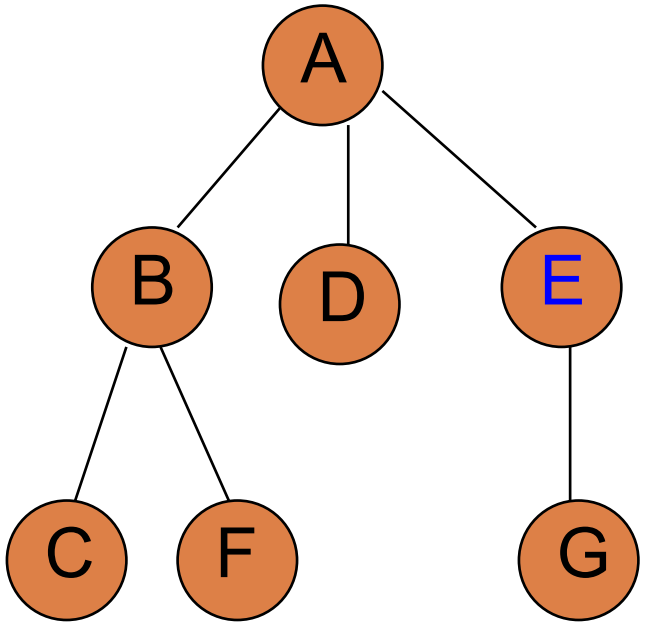


toVisit-queue: C F

printed: A B D E

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

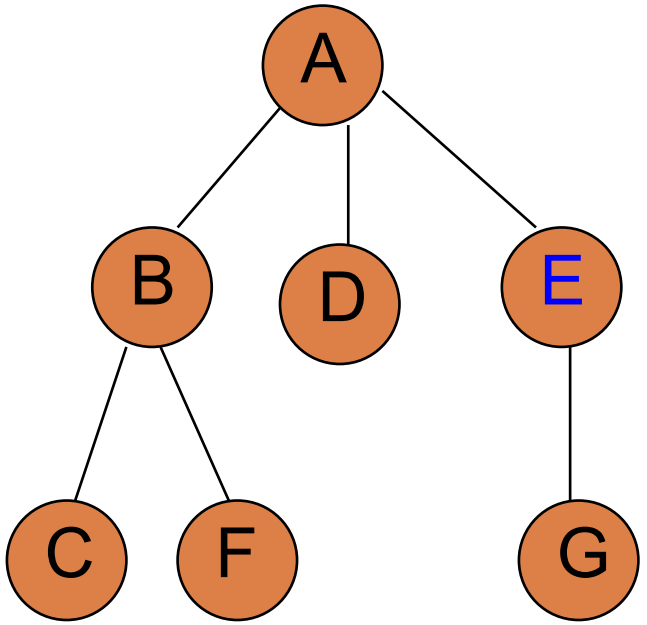


toVisit-queue: C F

printed: A B D E

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

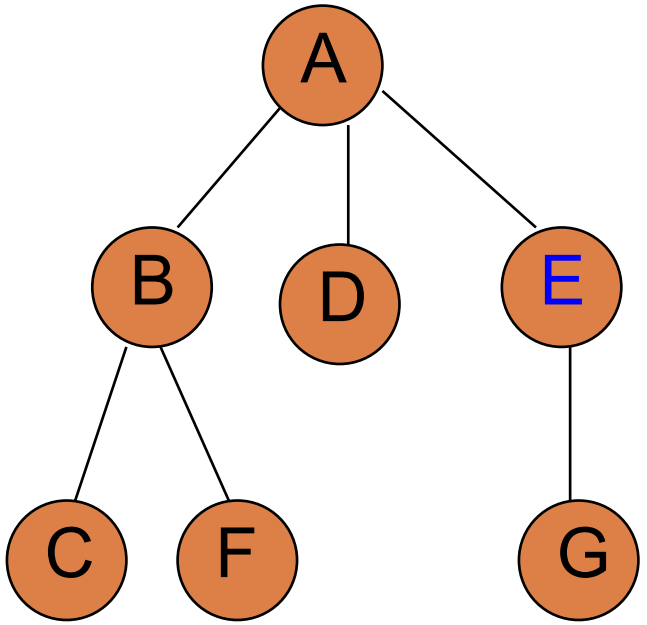


toVisit-queue: C F G

printed: A B D E

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



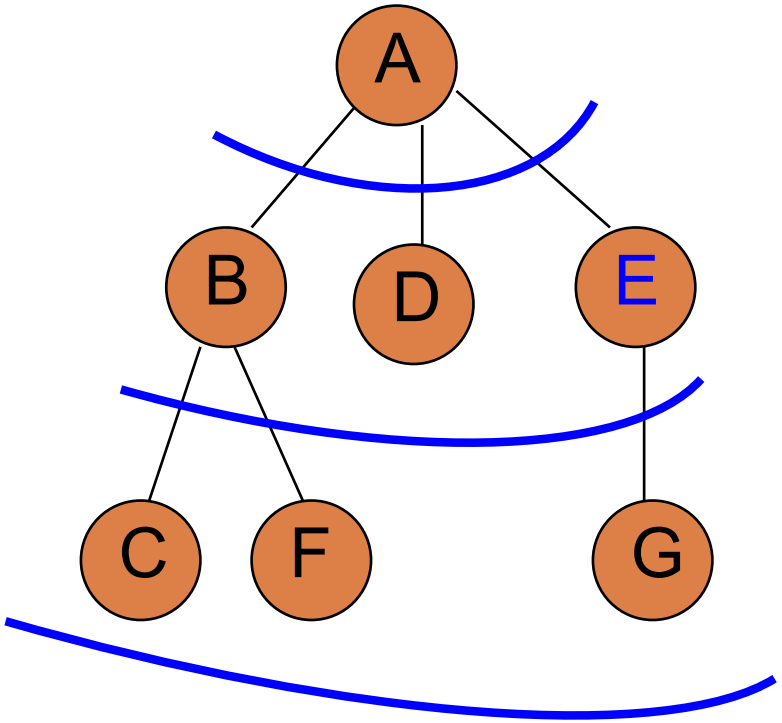
toVisit-queue: C F G

printed: A B D E

How are we exploring the vertices?

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



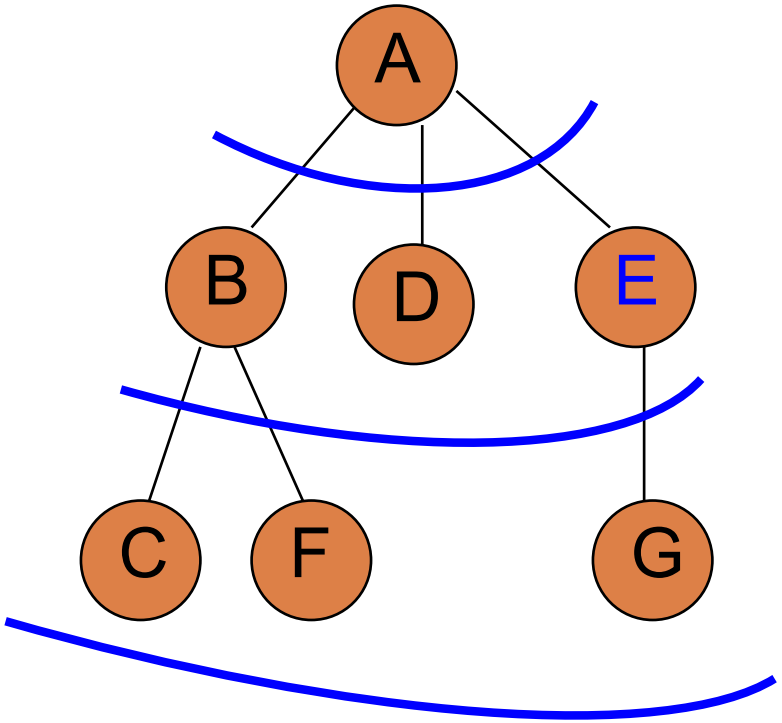
toVisit-queue: C F G

printed: A B D E

Frontier: all vertices a given number of edges from the start/root

Tree BFS = Tree breadth first search

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



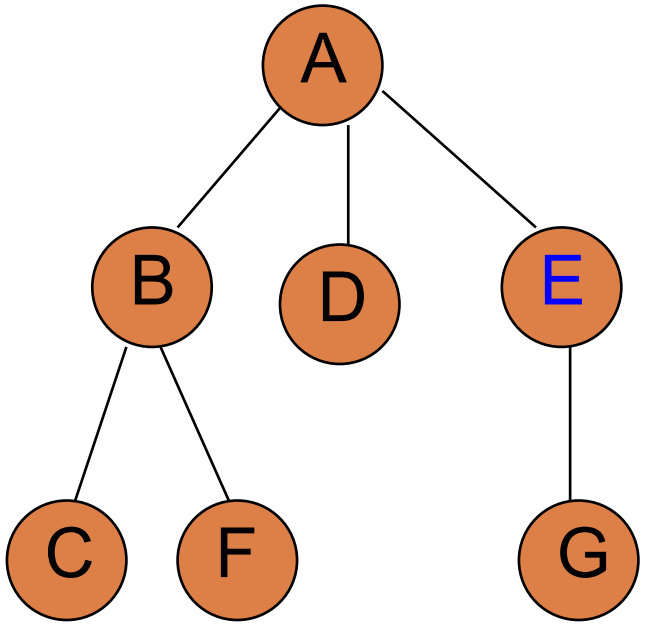
toVisit-queue: C F G

printed: A B D E

Frontier: all vertices a given number of edges from the start/root

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

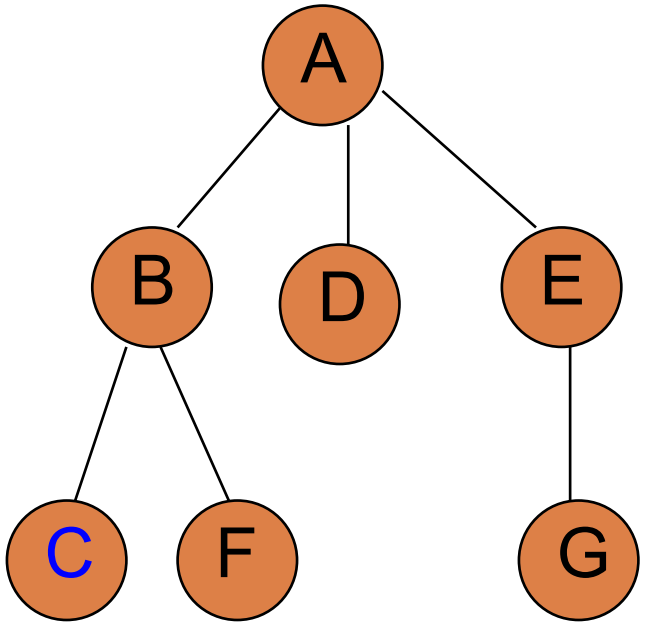


toVisit-queue: C F G

printed: A B D E

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

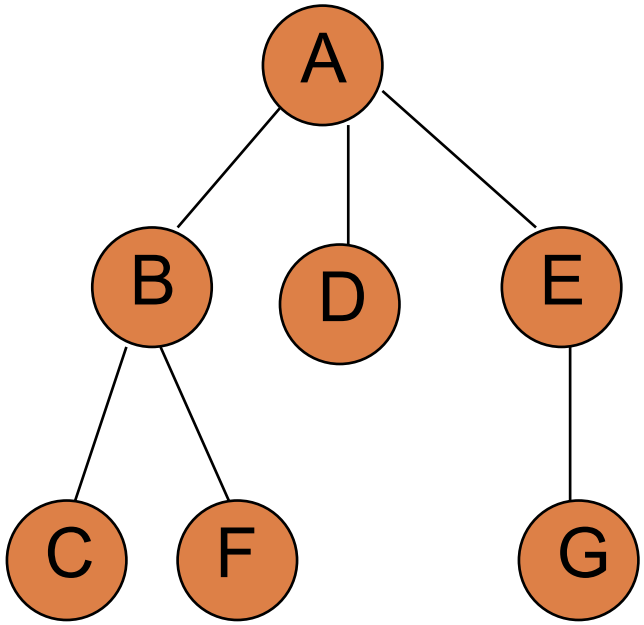


toVisit-queue: F G

printed: A B D E C

Tree BFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

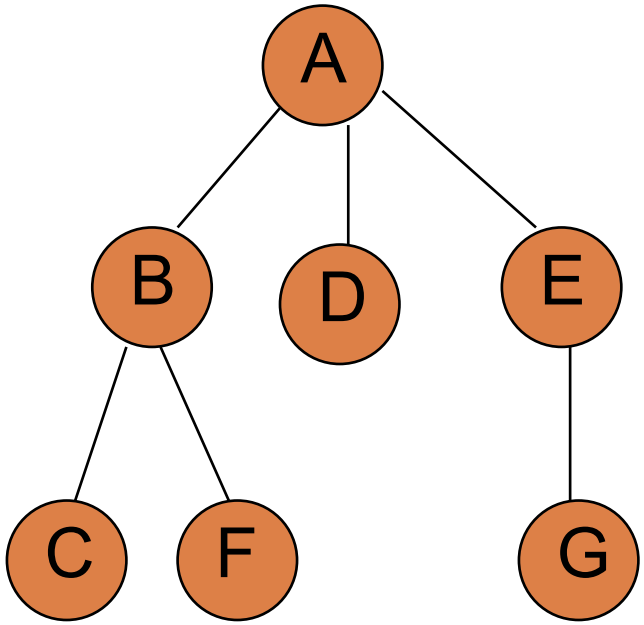


toVisit-queue:

printed: A B D E C F G

Tree DFS

```
treeDFS( start )
    s = new Stack()
    s.add(start)
    treeSearch(s)
```

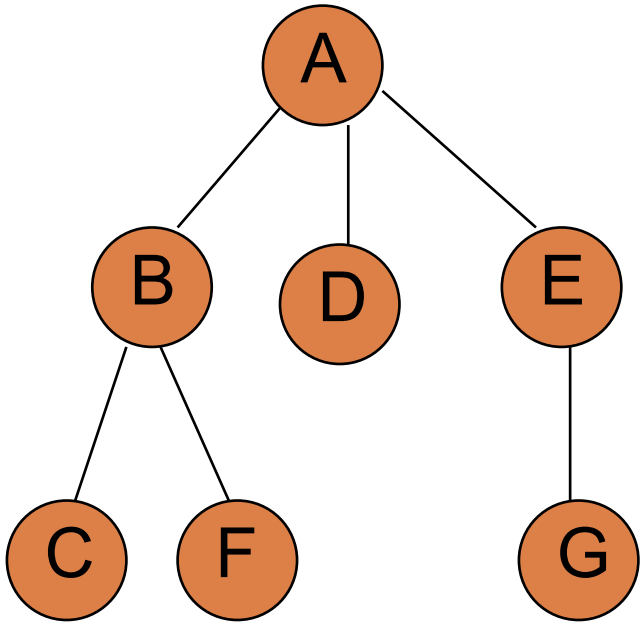


s:

printed:

Tree DFS

```
treeDFS( start )
    s = new Stack()
    s.add(start)
    treeSearch(s)
```

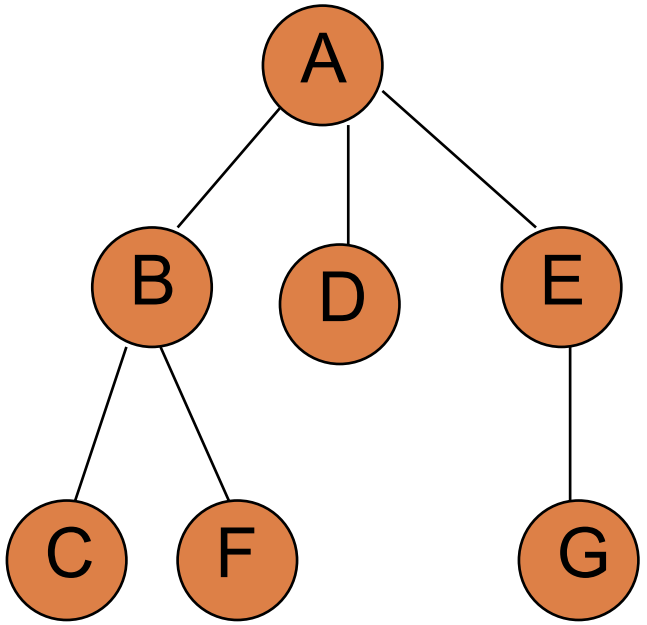


s: A

printed:

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



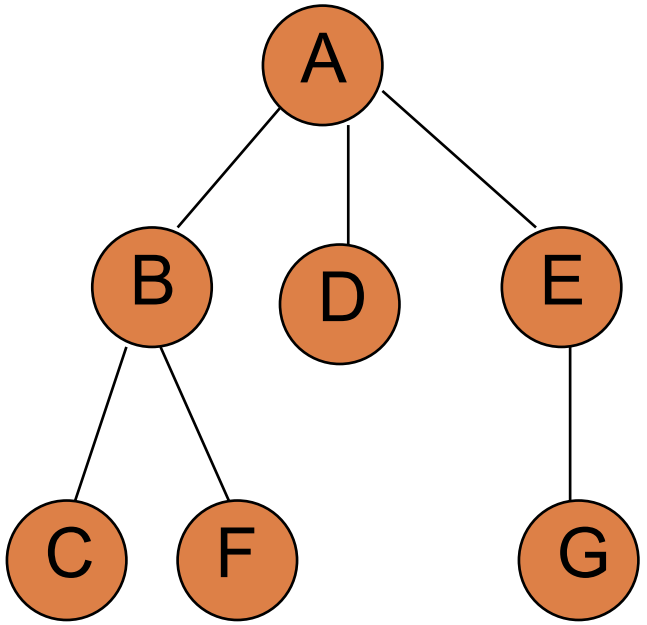
toVisit-stack: A

printed:

What order will the nodes get printed out?
Assume children are traversed left to right.

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

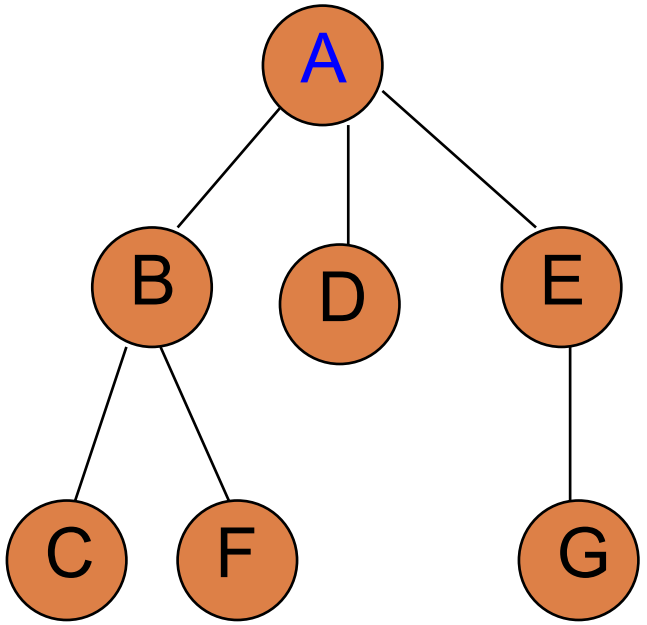


toVisit-stack: A

printed:

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

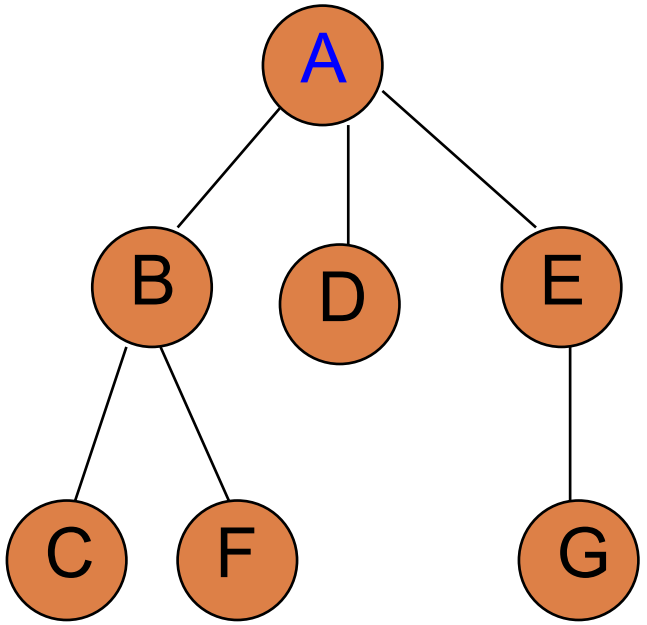


toVisit-stack:

printed: A

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

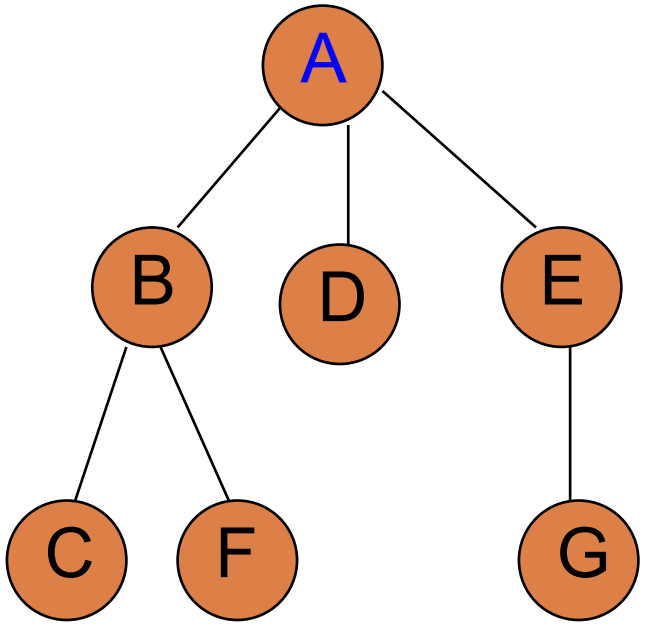


toVisit-stack:

printed: A

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

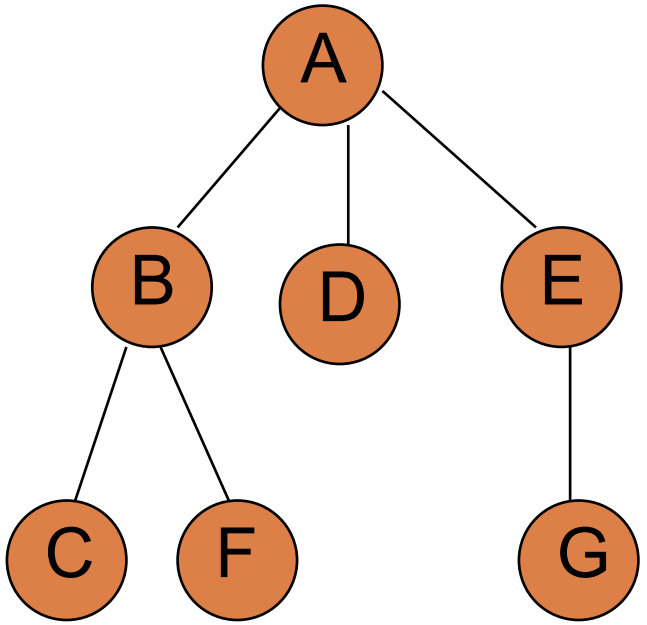


toVisit-stack: B D E

printed: A

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

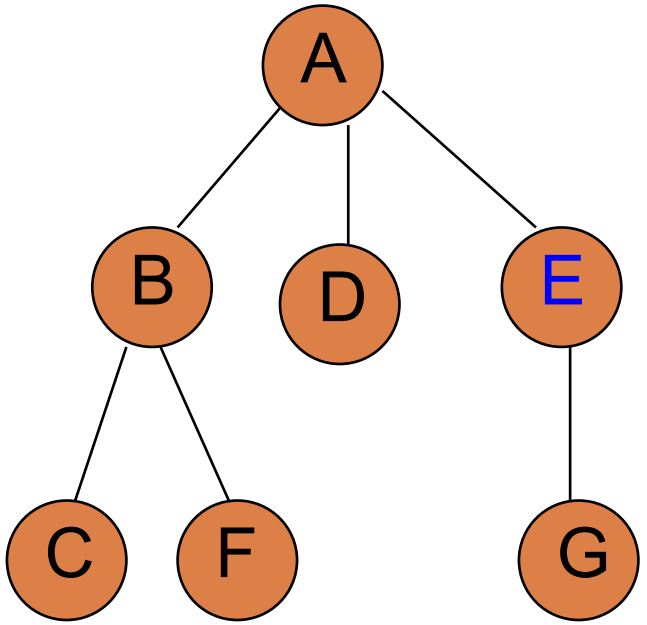


toVisit-stack: B D E

printed: A

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

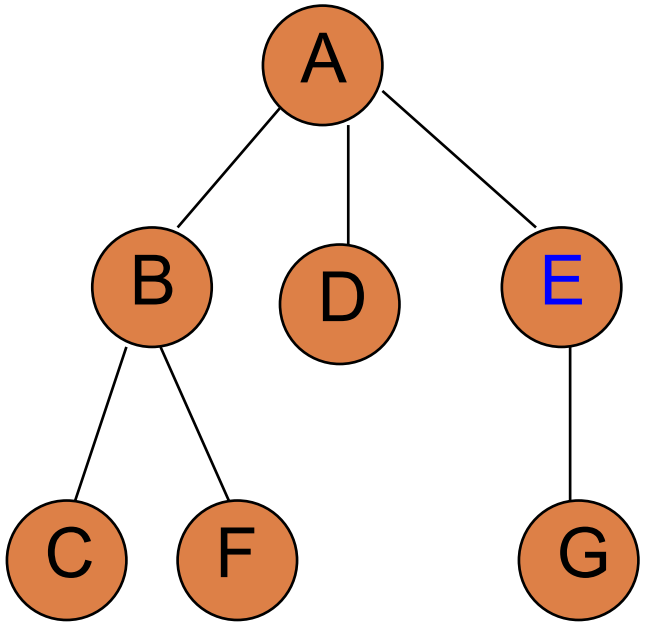


toVisit-stack: B D

printed: A E

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

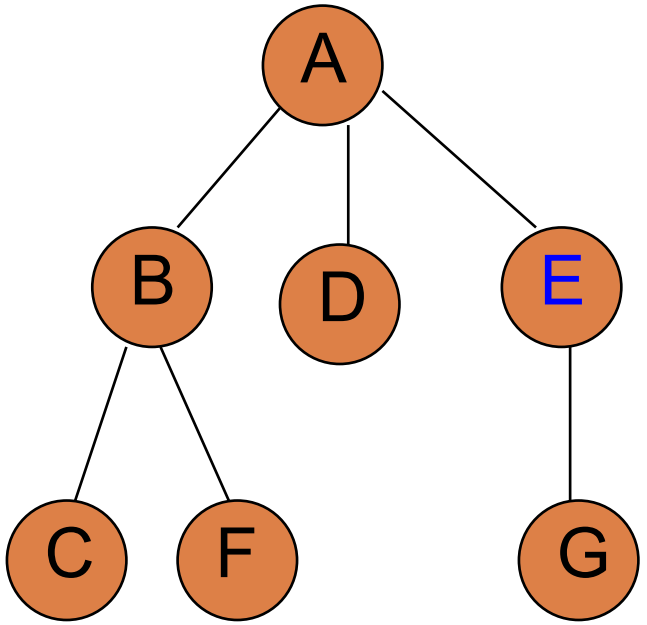


toVisit-stack: B D

printed: A E

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

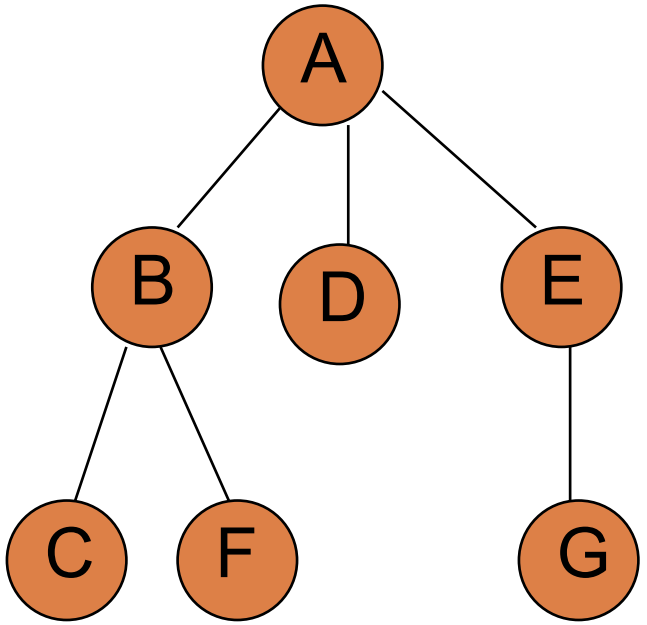


toVisit-stack: B D G

printed: A E

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

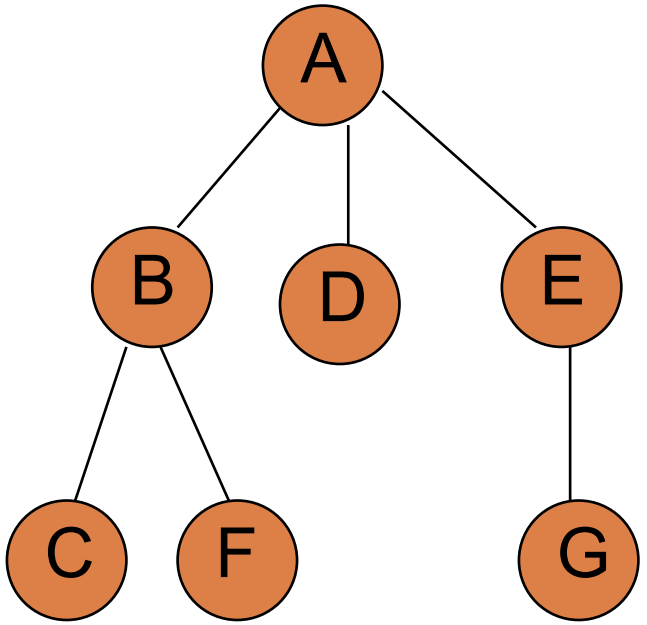


toVisit-stack: B D G

printed: A E

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

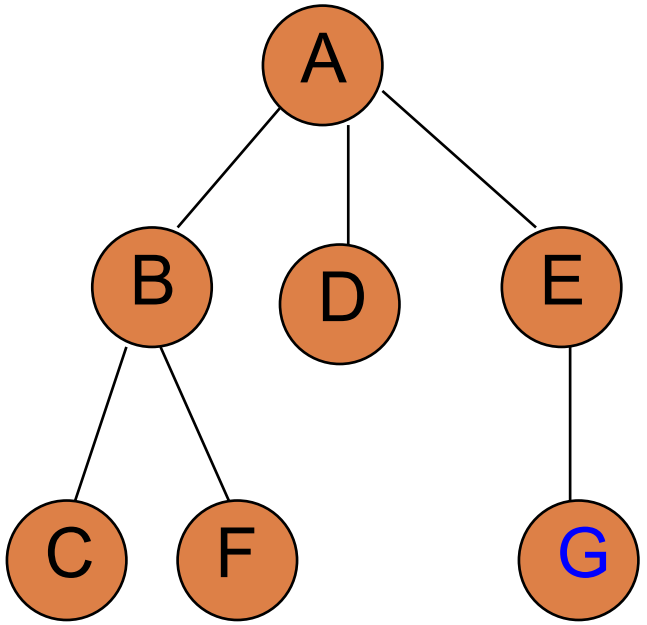


toVisit-stack: B D

printed: A E G

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

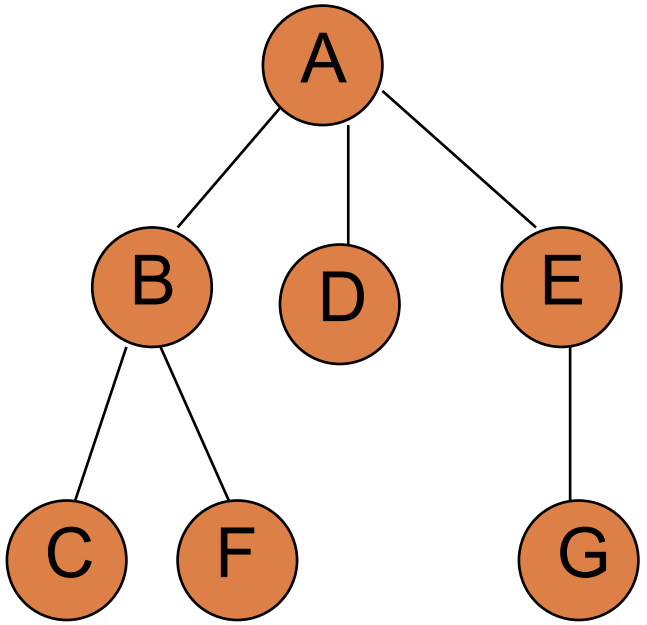


toVisit-stack: B D

printed: A E G

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

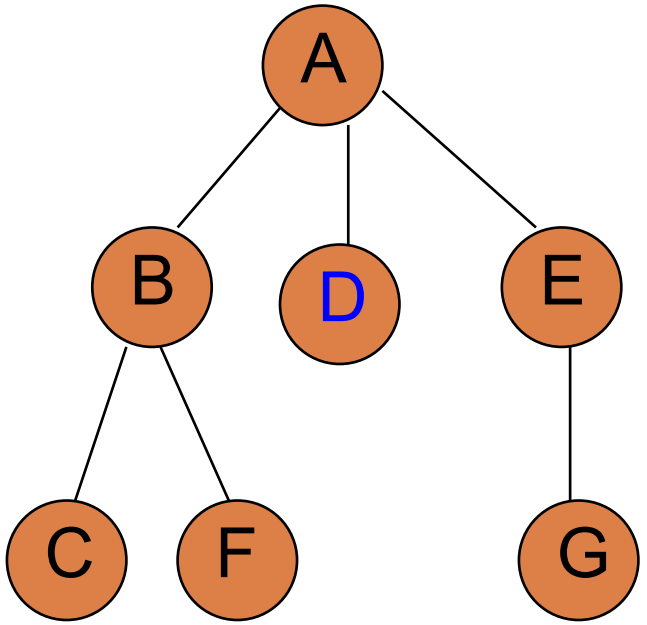


toVisit-stack: B D

printed: A E G

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

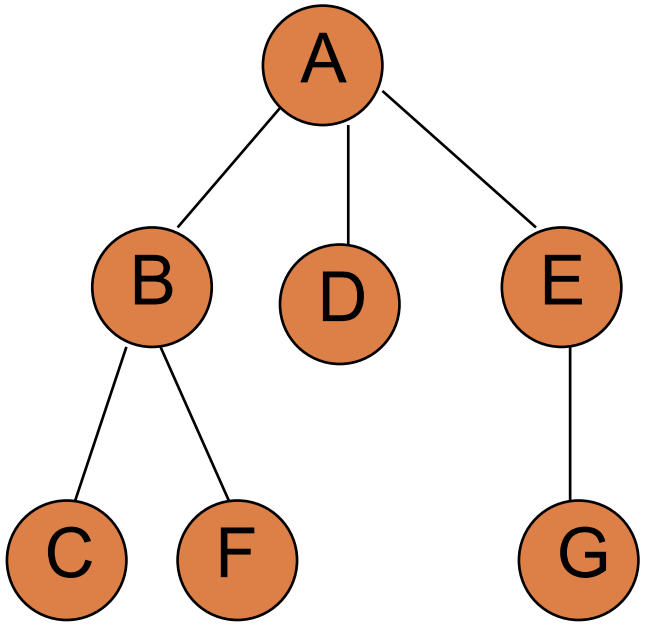


toVisit-stack: B

printed: A E G D

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



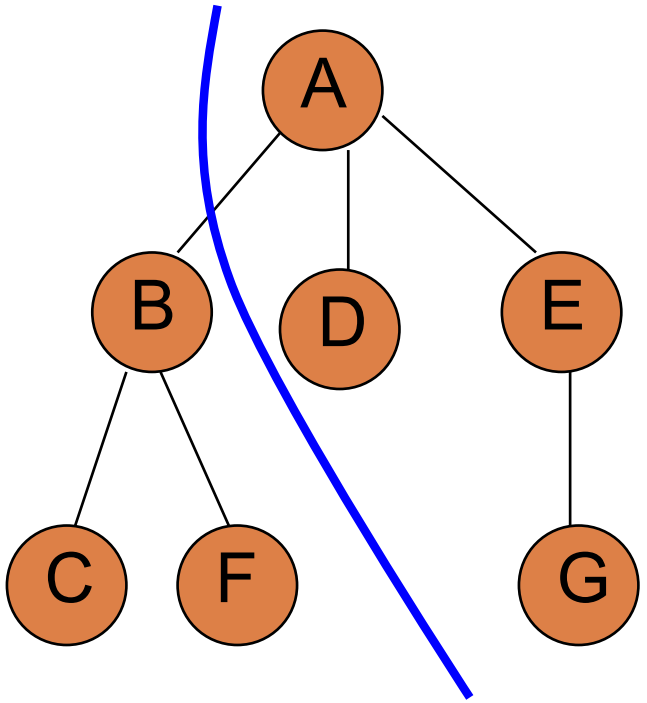
toVisit-stack: B

printed: A E G D

How are we exploring the vertices?

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



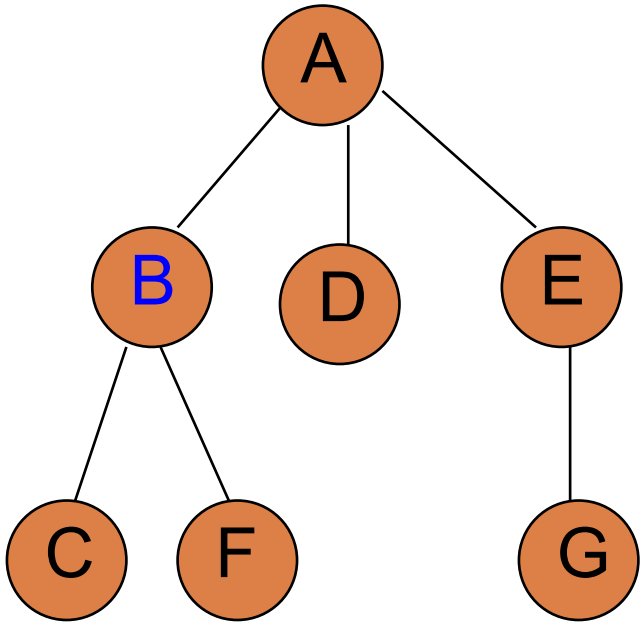
toVisit-stack: B

printed: A E G D

Frontier: go as far down one branch as possible, working right to left

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

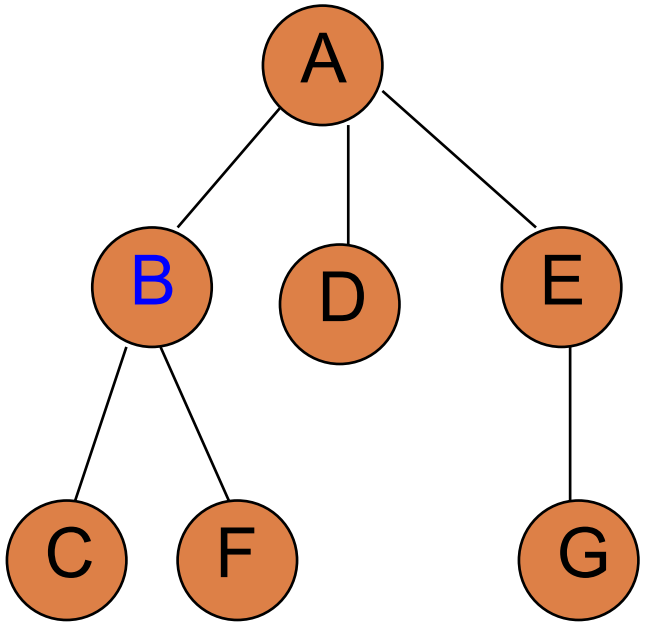


toVisit-stack:

printed: A E G D B

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

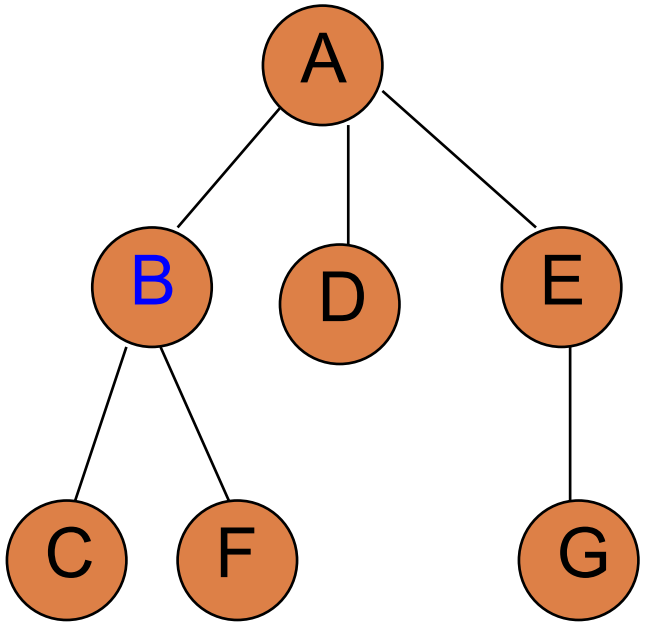


toVisit-stack:

printed: A E G D B

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

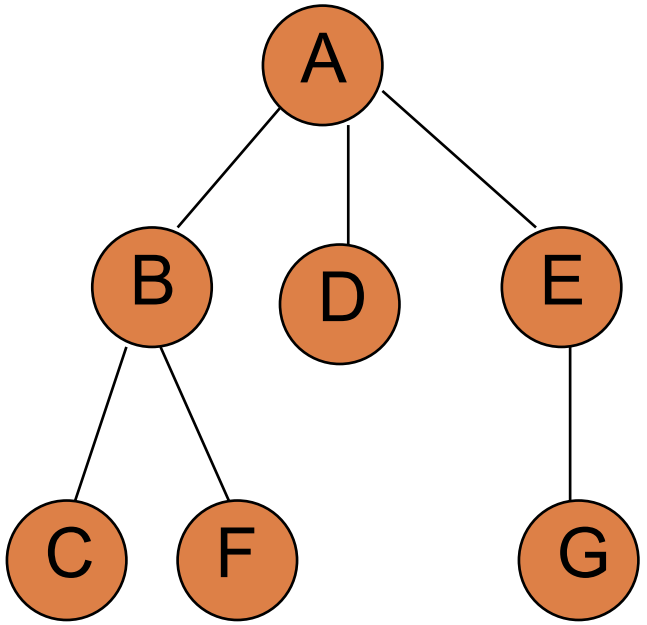


toVisit-stack: C F

printed: A E G D B

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

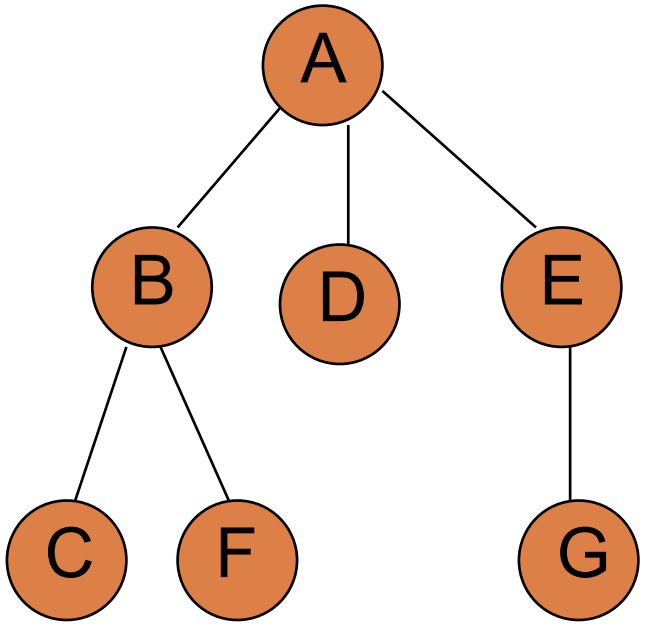


toVisit-stack: C F

printed: A E G D B

Tree DFS

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



toVisit-stack:

printed: A E G D B F C

Run-time of graph algorithms

A graph is a set of vertices V and a set of edges $(u,v) \in E$ where $u,v \in V$

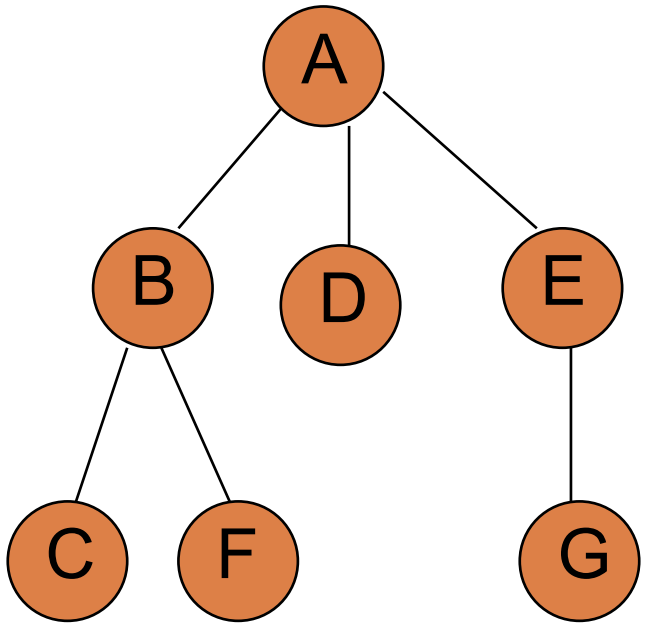
When we analyze graph algorithms, the run-time often includes both the number of vertices **AND** the number of edges:

- $|V|$ = number of vertices
- $|E|$ = number of edges

Sometimes, in big-O notation, we'll use just V and E to represent these to simplify notation

treeSearch run-time

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



What is the big-O run-time of teeSearch?

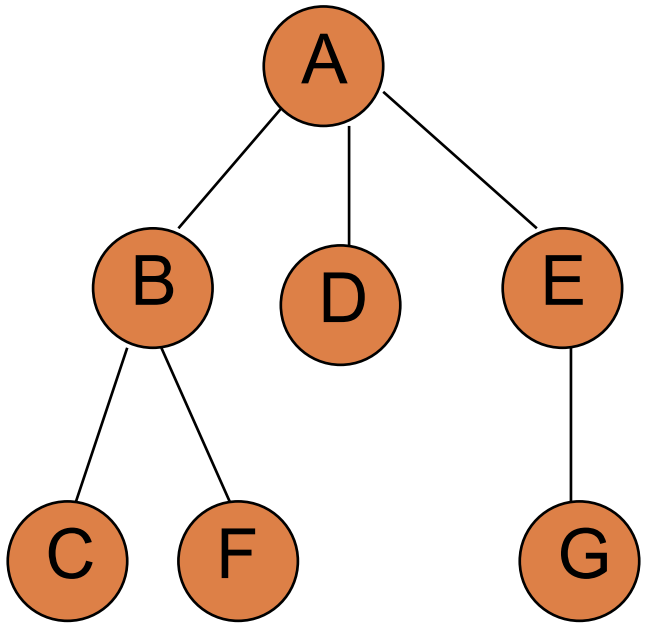
Assume all of the stack/queue operations are constant.

How many times do we visit each vertex?

How many times do we traverse each edge (vis the for loop)?

treeSearch run-time

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



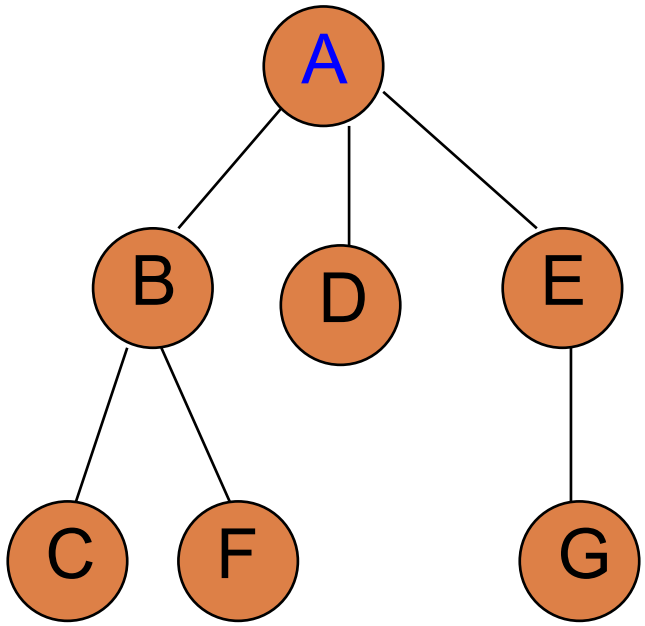
How many times do we visit each vertex? Exactly once

How many times do we traverse each edge (vis the for loop)? Exactly once

What is the big-O run-time of teeSearch? O(|V| + |E|). Linear algorithm.

What algorithm is this?

```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```

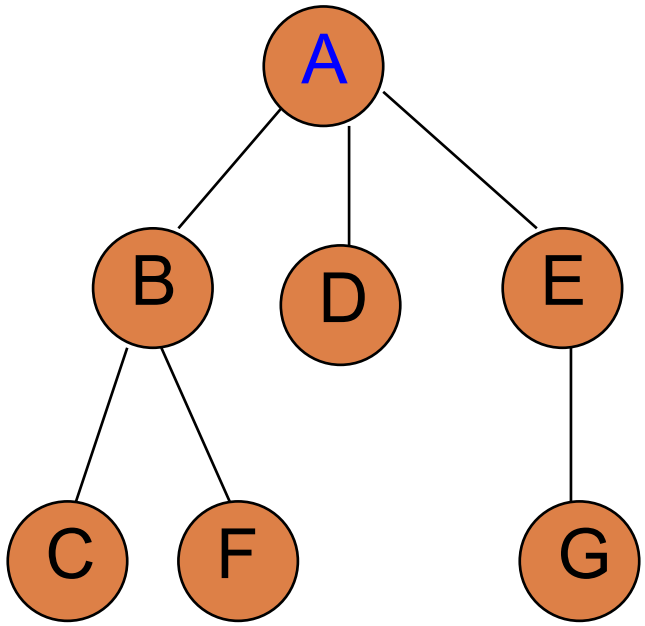


What order will the vertices get printed out?

Assume children are traversed left to right.

What algorithm is this?

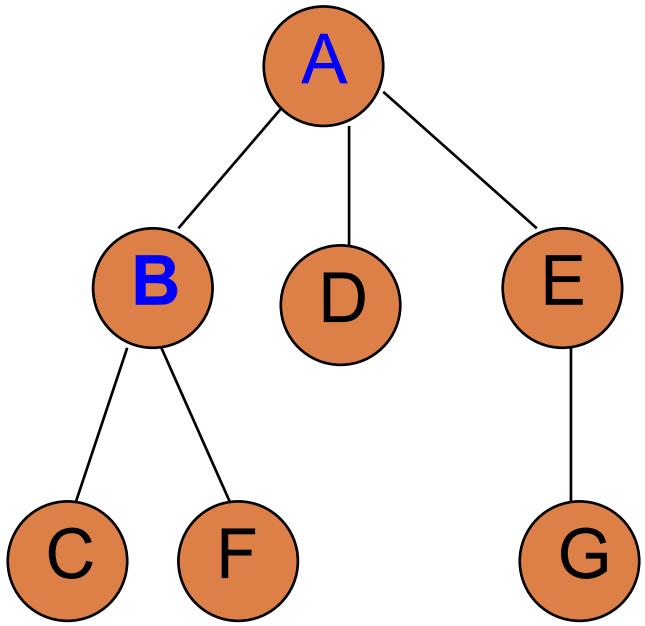
```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```



Visited: A

What algorithm is this?

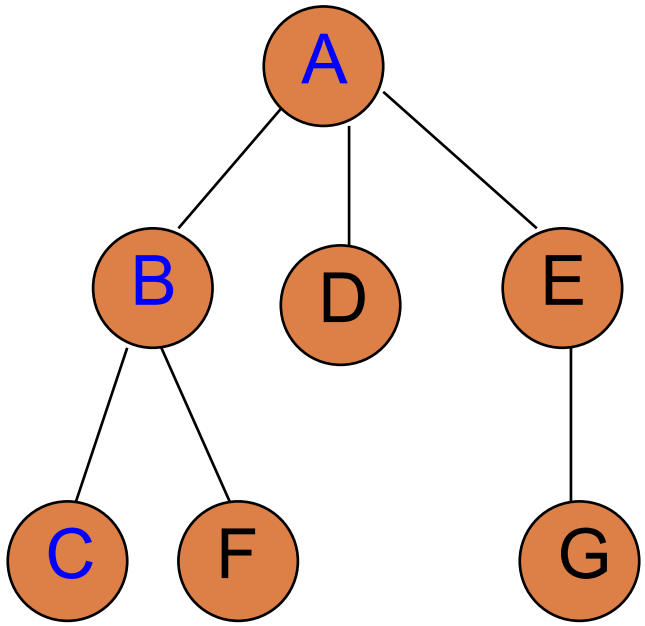
```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```



Visited: A B

What algorithm is this?

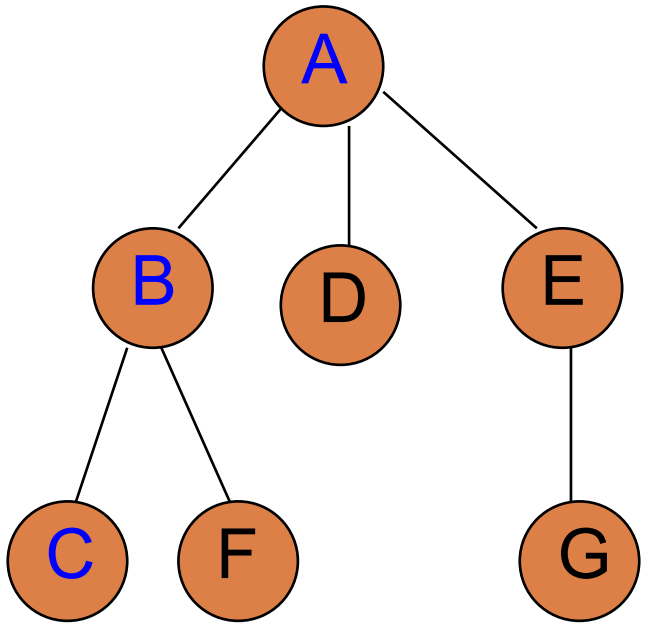
```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```



Visited: A B C

What algorithm is this?

```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```

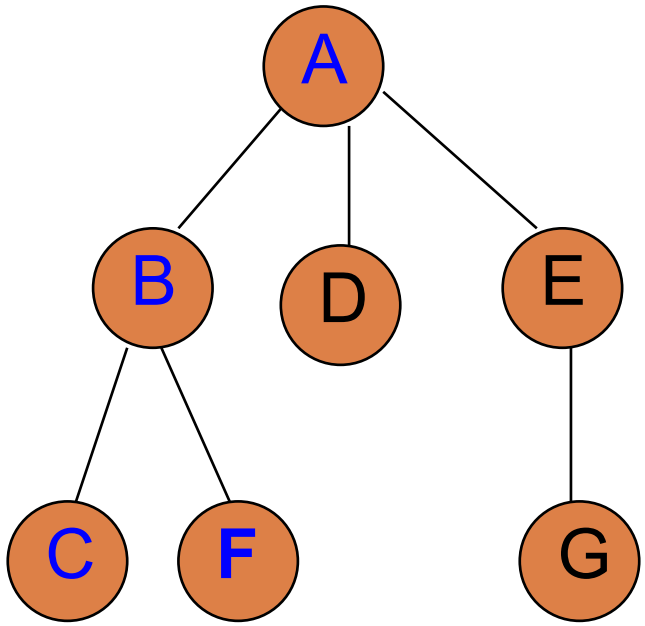


Visited: A B C

Now where?

What algorithm is this?

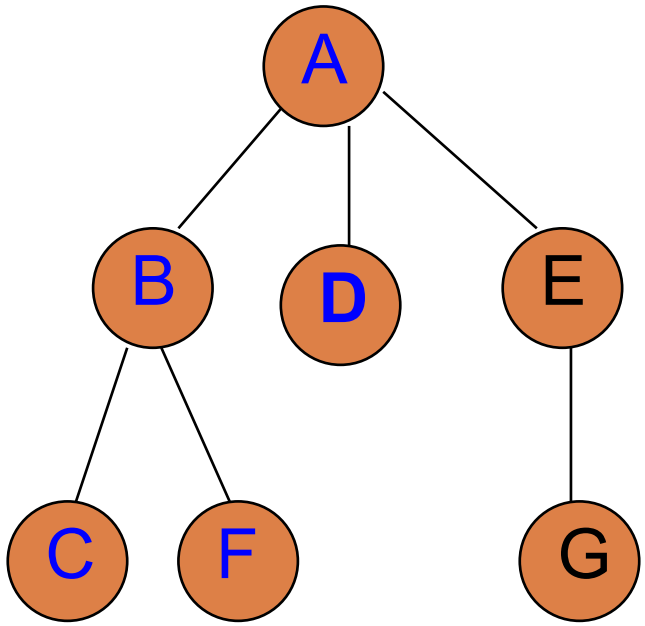
```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```



Visited: A B C F

What algorithm is this?

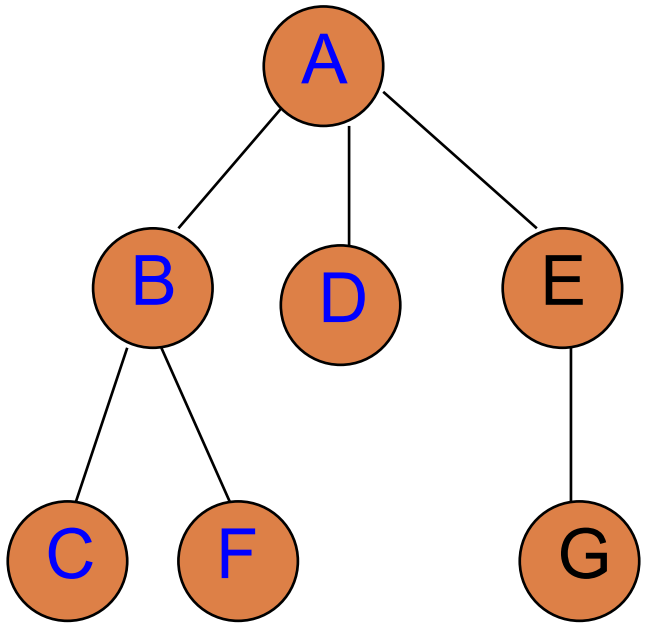
```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```



Visited: A B C F D

What algorithm is this?

```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```

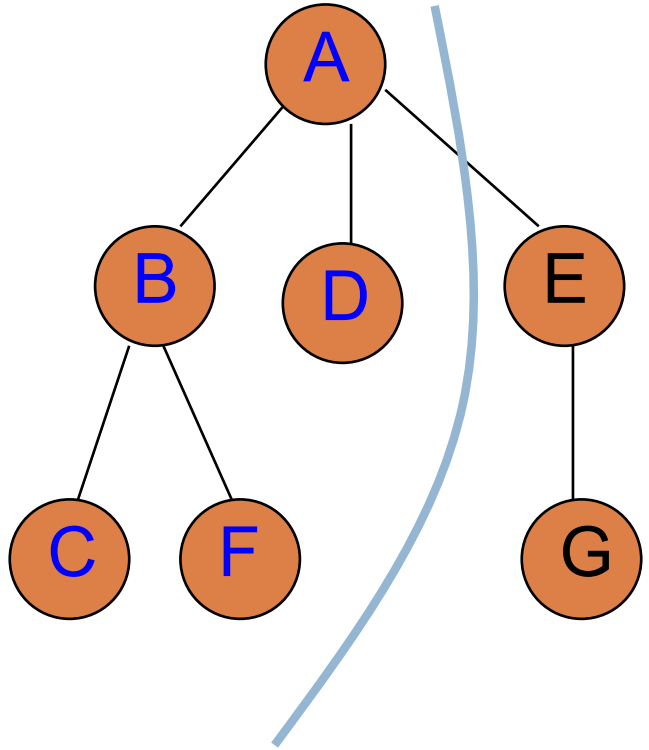


Visited: A B C F D

What algorithm is this?

What algorithm is this?

```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```

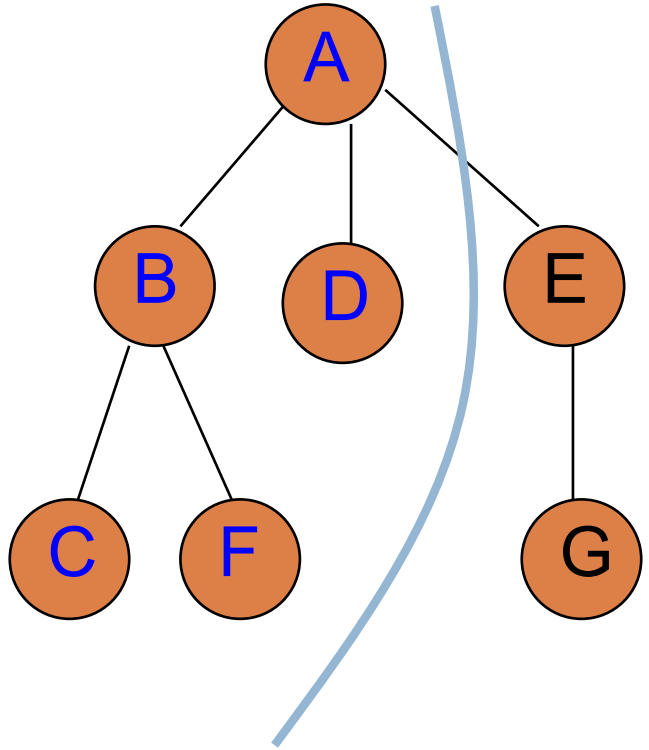


Depth first search!

Visited: A B C F D

What algorithm is this?

```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```

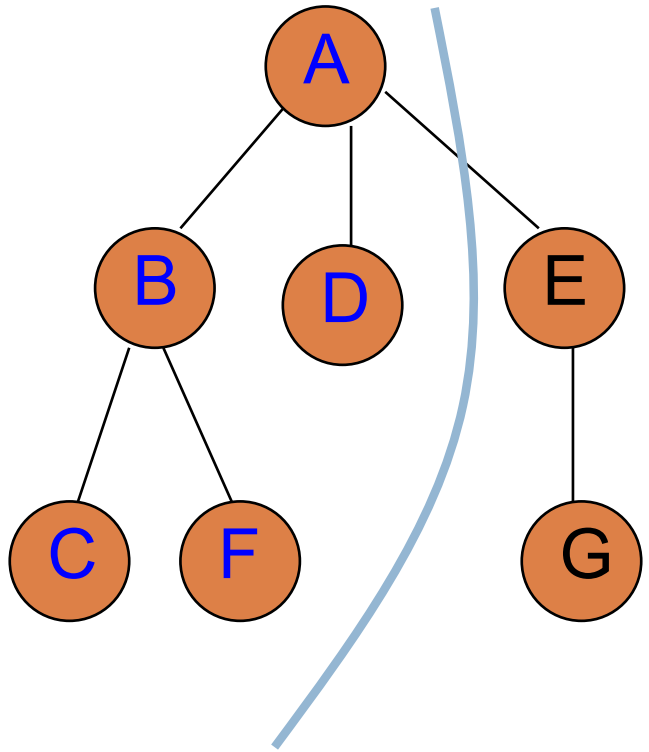


Any difference between this version and the stack version?

Visited: A B C F D

What algorithm is this?

```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```



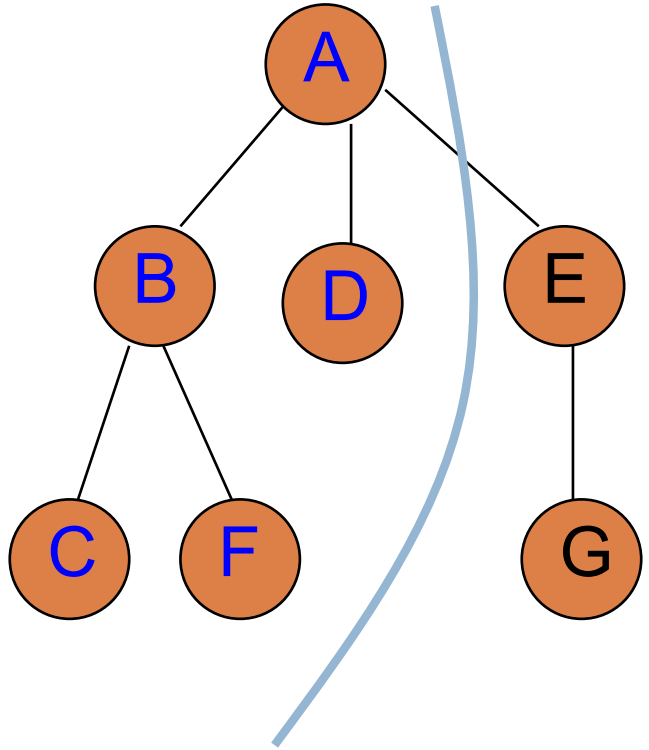
Any difference between this version and the stack version?

Visited: A B C F D

Traverses in the other direction (left to right in this case).

What algorithm is this?

```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```



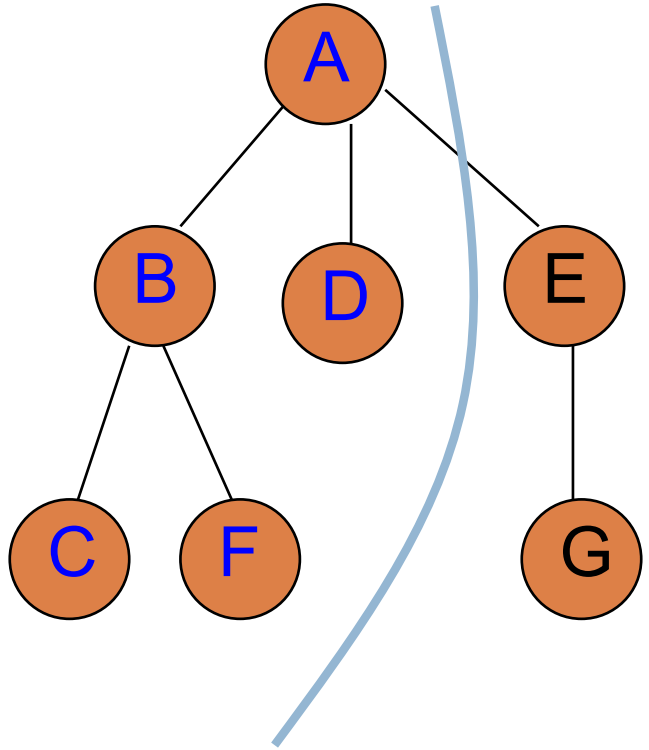
treeDFS used a stack.

Is there a stack here?

Visited: A B C F D

What algorithm is this?

```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```



treeDFS used a stack.

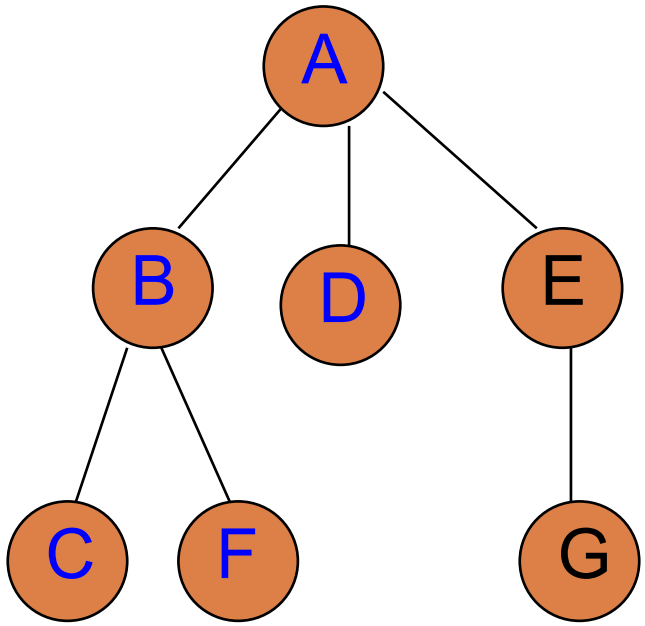
Is there a stack here?

Visited: A B C F D

The run-time stack keeping track of recursive calls!

What algorithm is this?

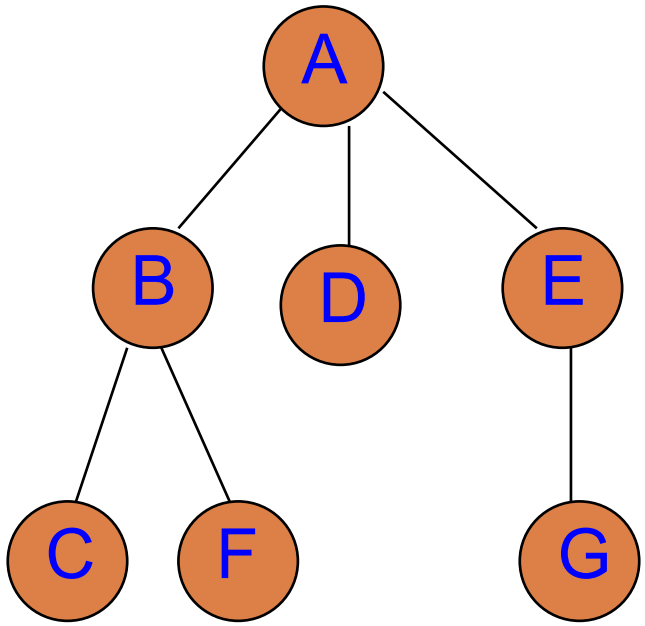
```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```



Visited: A B C F D

What algorithm is this?

```
search( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```



Visited: A B C F D E G

DFS versions

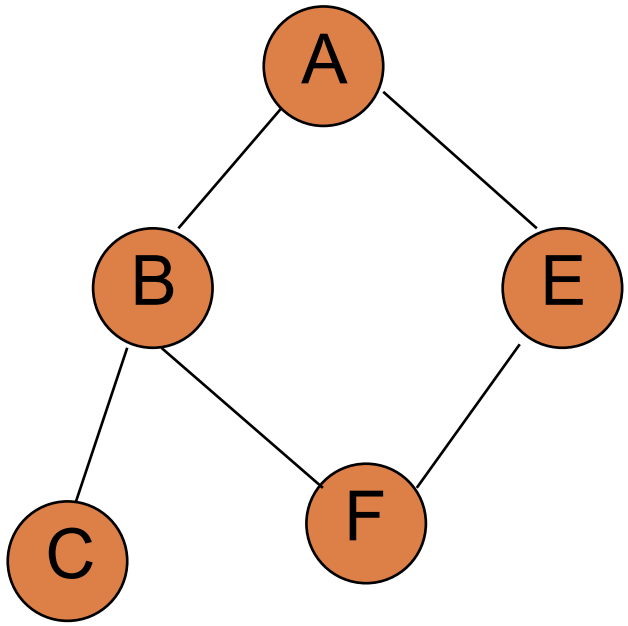
```
treeDFS( start )
    s = new Stack()
    s.add(start)
    treeSearch(s)


treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

```
treeRecursiveDFS( v )
    // visit v, e.g., print it out
    for c in v.getChildren()
        treeRecursiveDFS(c)
```

treeSearch on graphs

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

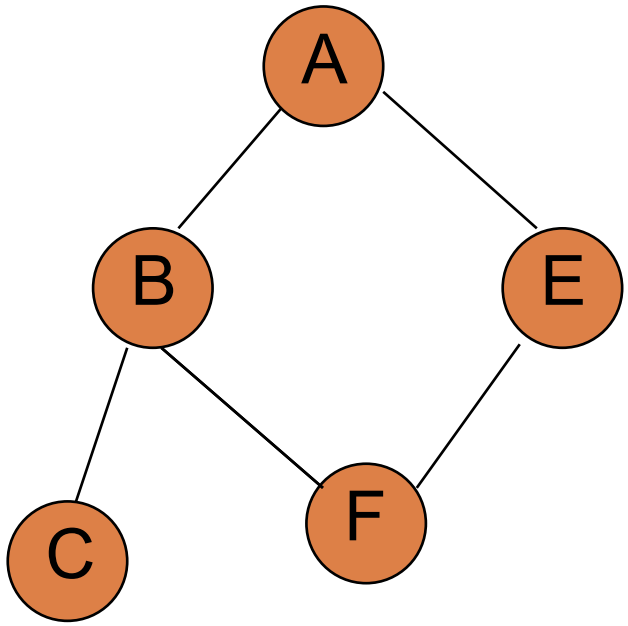


What would happen if we ran treeSearch on this graph?

Won't ever end!

treeSearch on graphs

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```



How can we fix this?

Keep track of the vertices that we've visited

Searching on graphs

```
treeSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

Searching on graphs

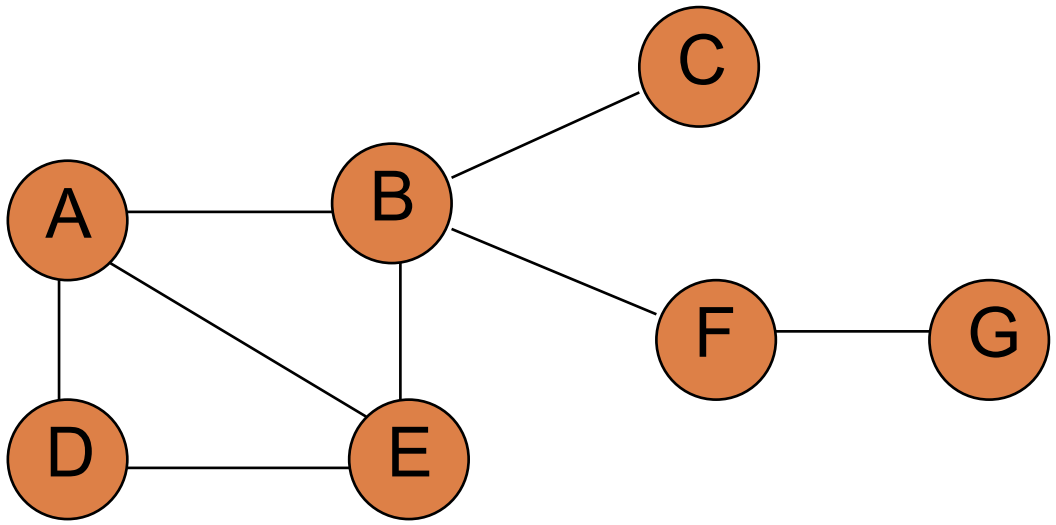
```
graphBFS( start )
    q = new Queue()
    q.add(start)
    treeSearch(q)
```

```
graphDFS( start )
    s = new Stack()
    s.add(start)
    treeSearch(s)
```

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

BFS

```
graphBFS( start )
    q = new Queue()
    q.add(start)
    treeSearch(q)
```

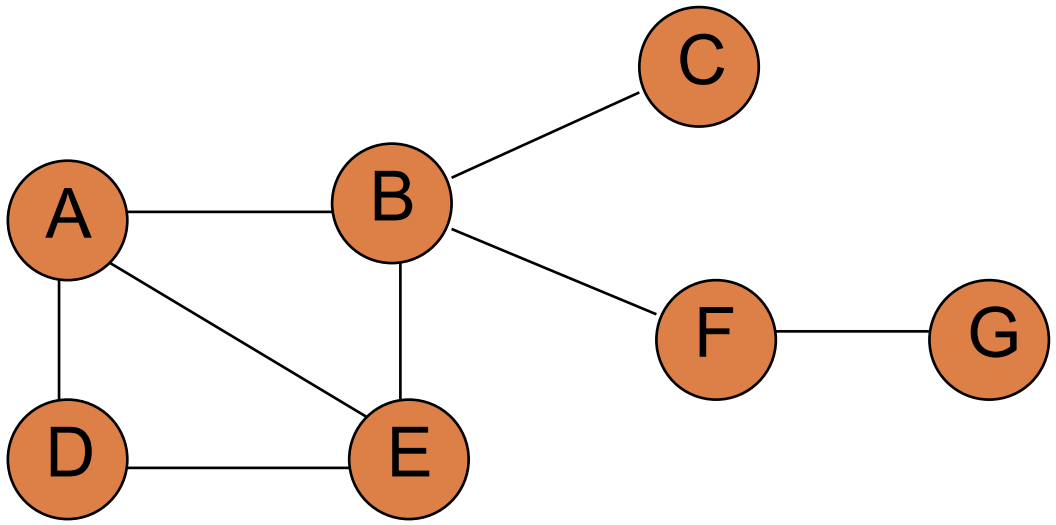


toVisit-queue: A

visited:

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```



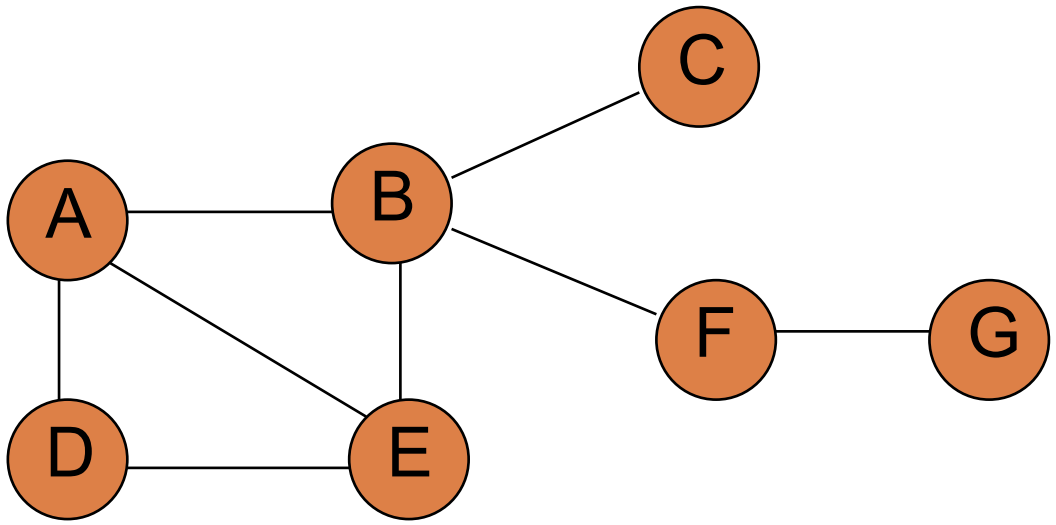
toVisit-queue: A

visited:

What order will the nodes get printed out?
Assume edges are traversed alphabetically.

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

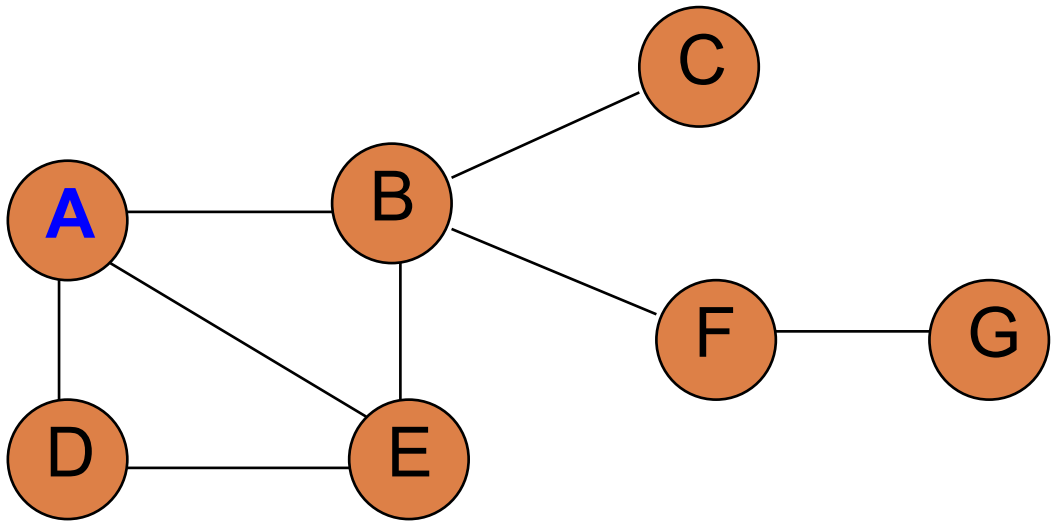


toVisit-queue: A

visited:

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

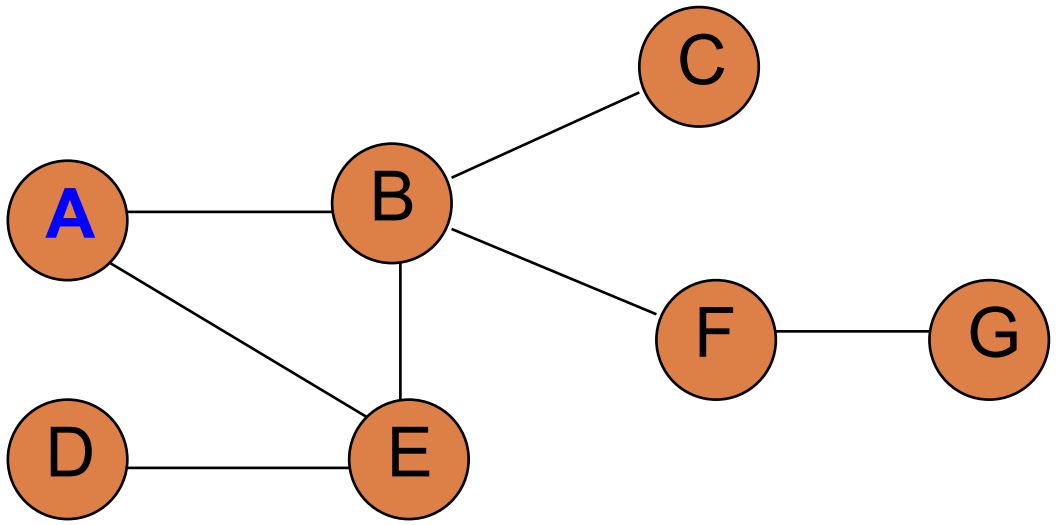


toVisit-queue:

visited: A

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

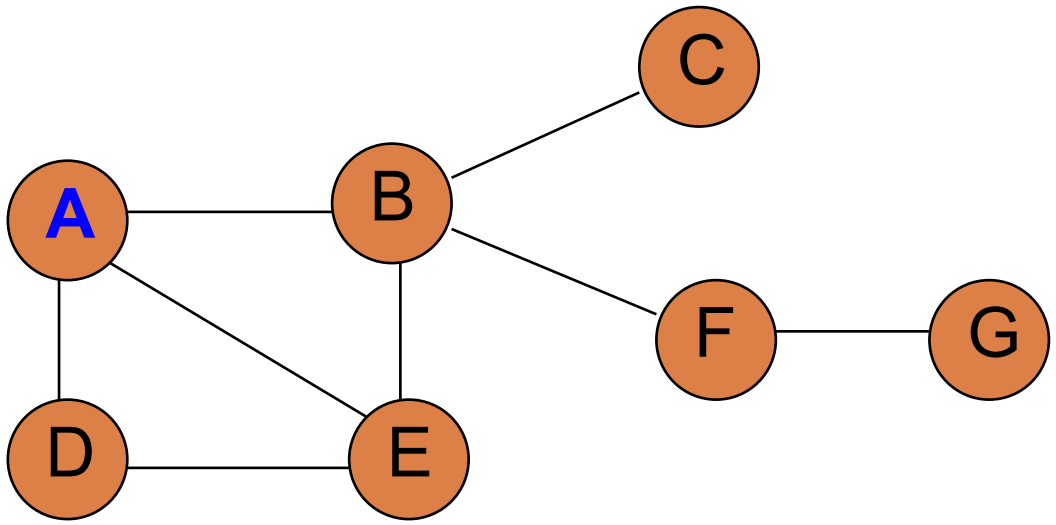


toVisit-queue:

visited: A

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

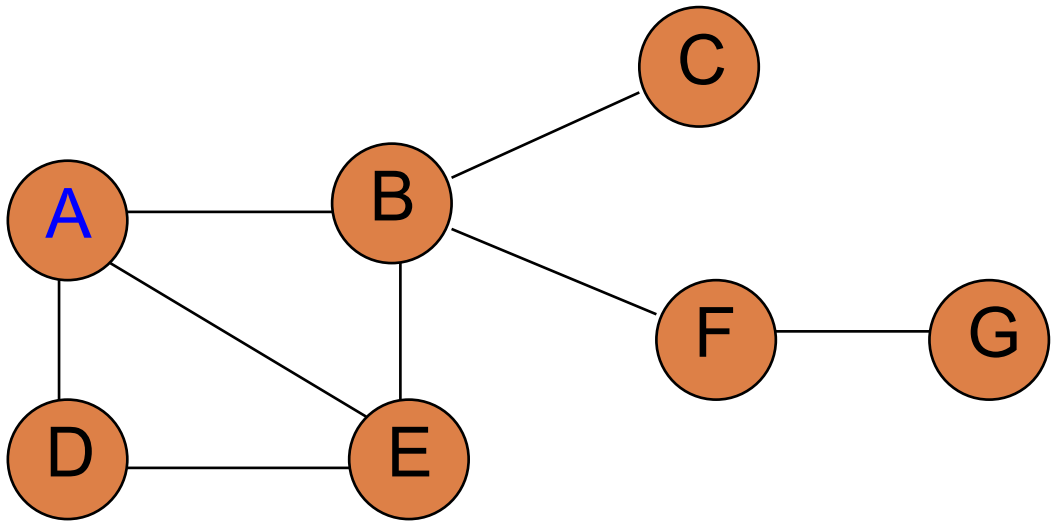


toVisit-queue: B D E

visited: A

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

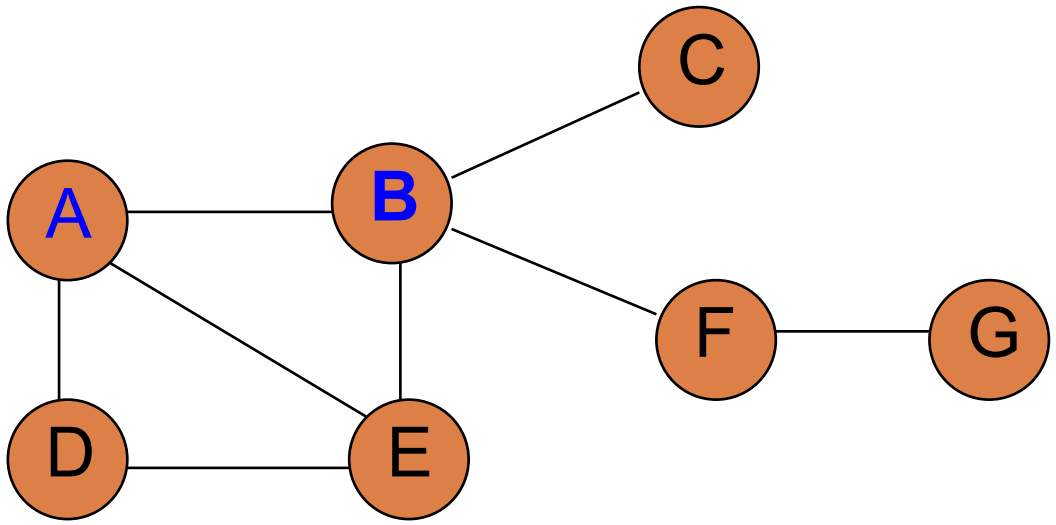


toVisit-queue: B D E

visited: A

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

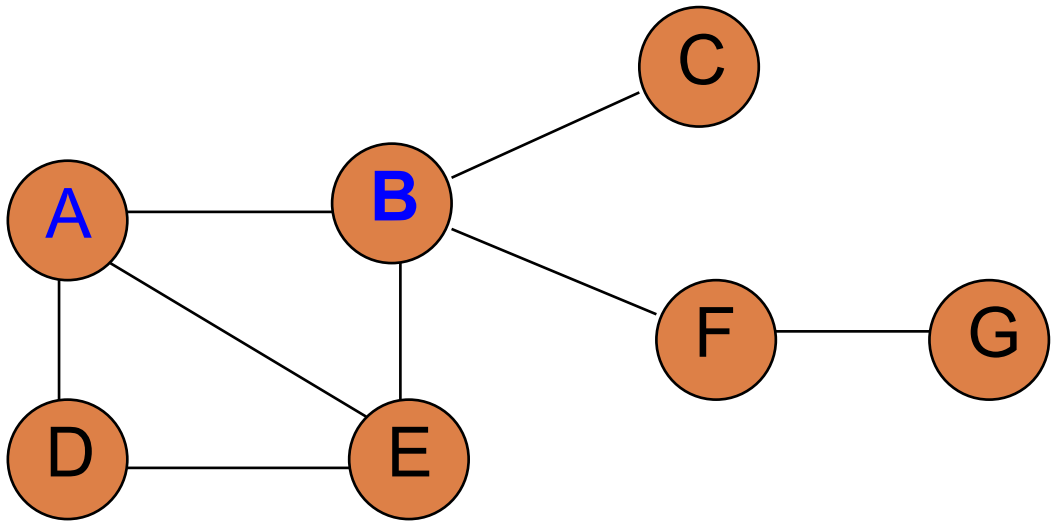


toVisit-queue: D E

visited: A B

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

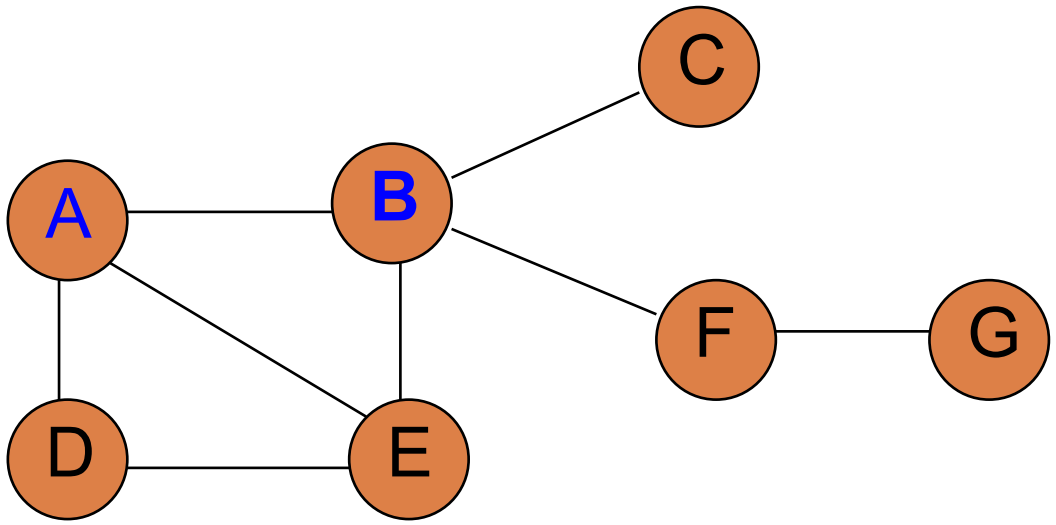


toVisit-queue: D E

visited: A B

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```



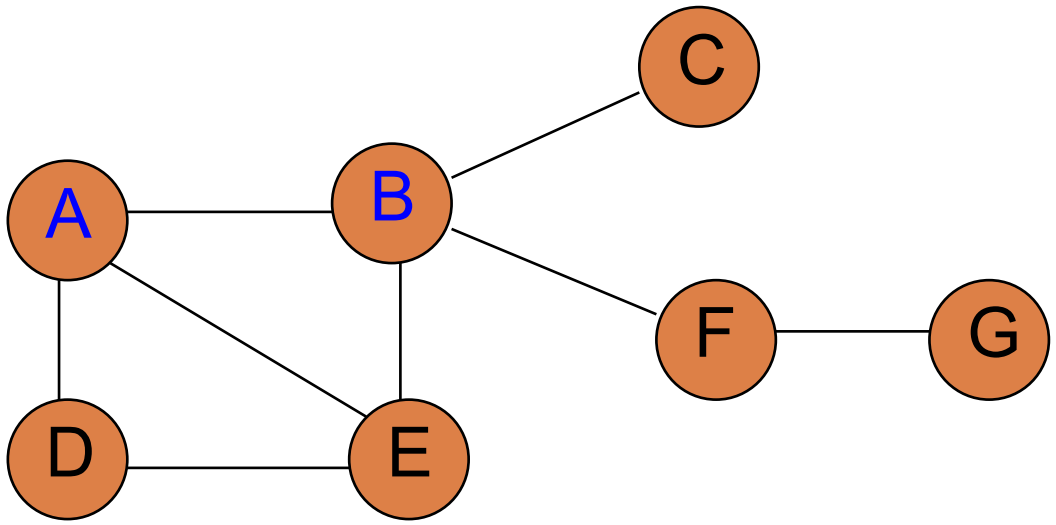
toVisit-queue: D E C E F

visited: A B

Notice that we do add E again to toVisit since we haven't visited it yet

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

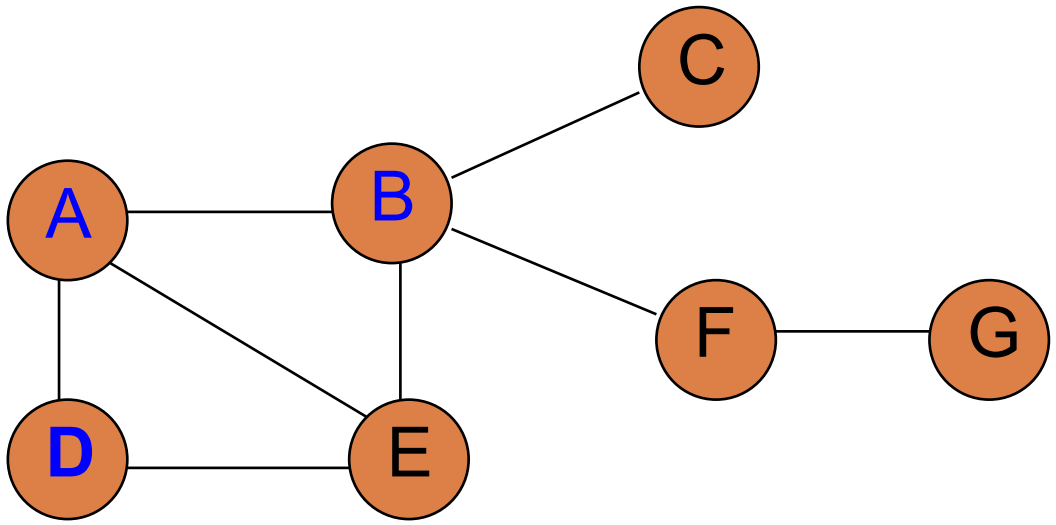


toVisit-queue: D E C E F

visited: A B

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

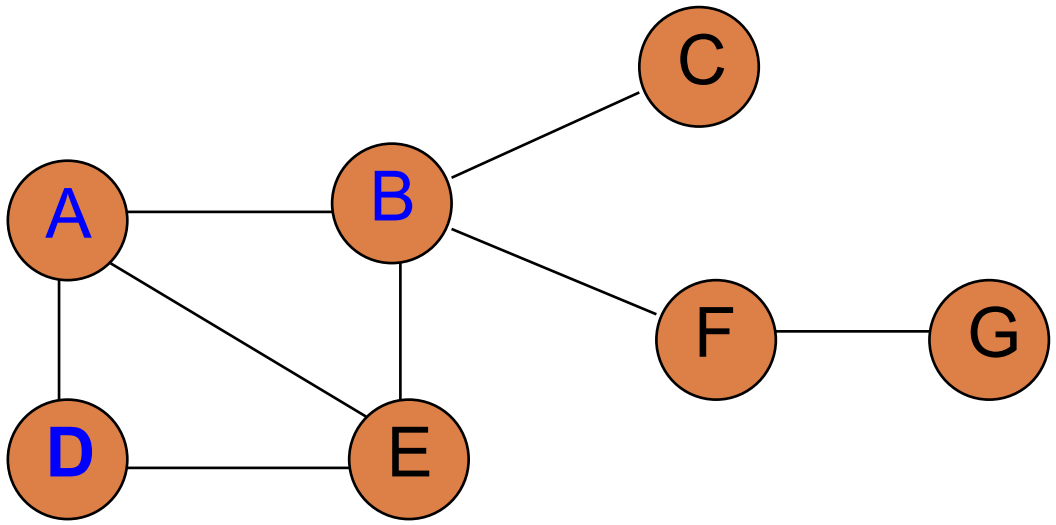


toVisit-queue: E C E F

visited: A B D

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

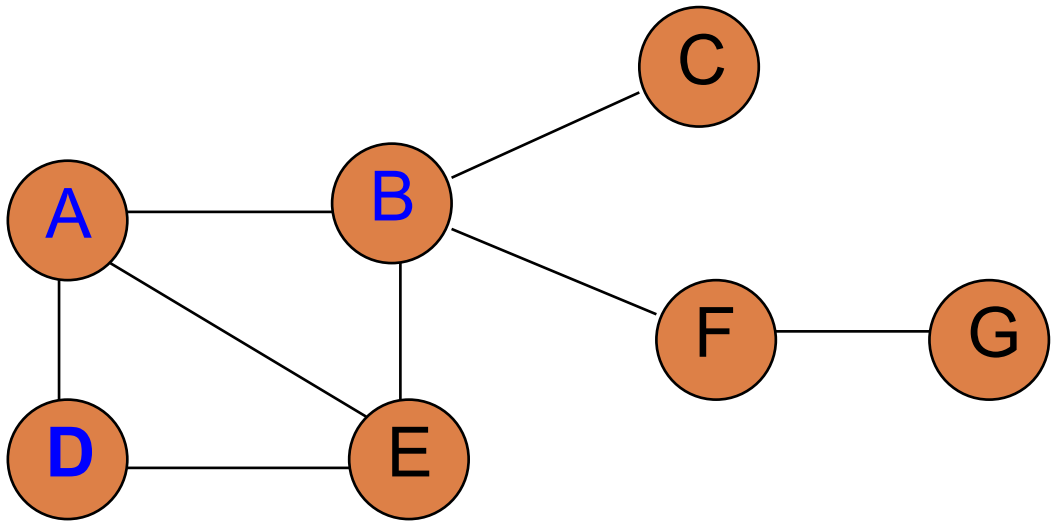


toVisit-queue: E C E F

visited: A B D

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```



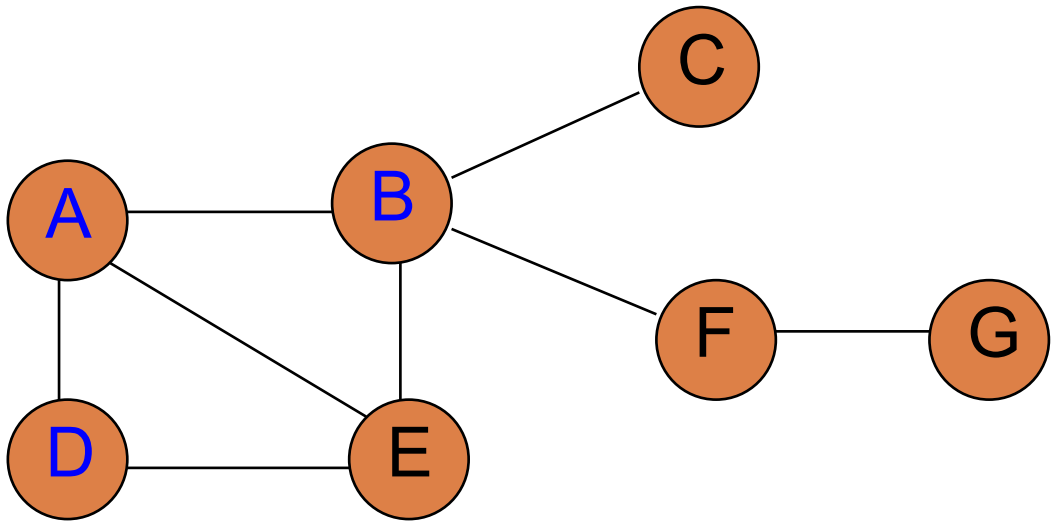
toVisit-queue: E C E F E

visited: A B D

We add E again to toVisit since we still haven't visited it yet

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

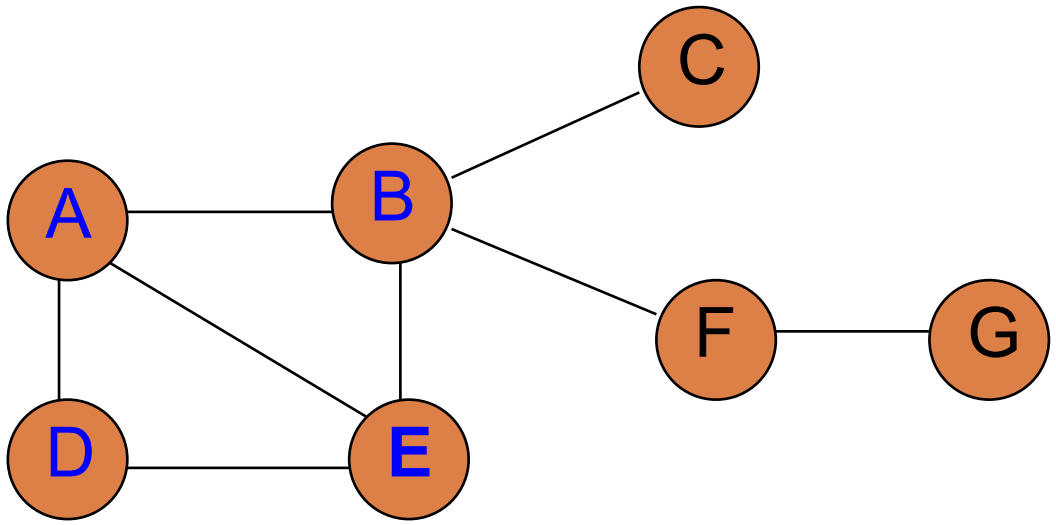


toVisit-queue: E C E F E

visited: A B D

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

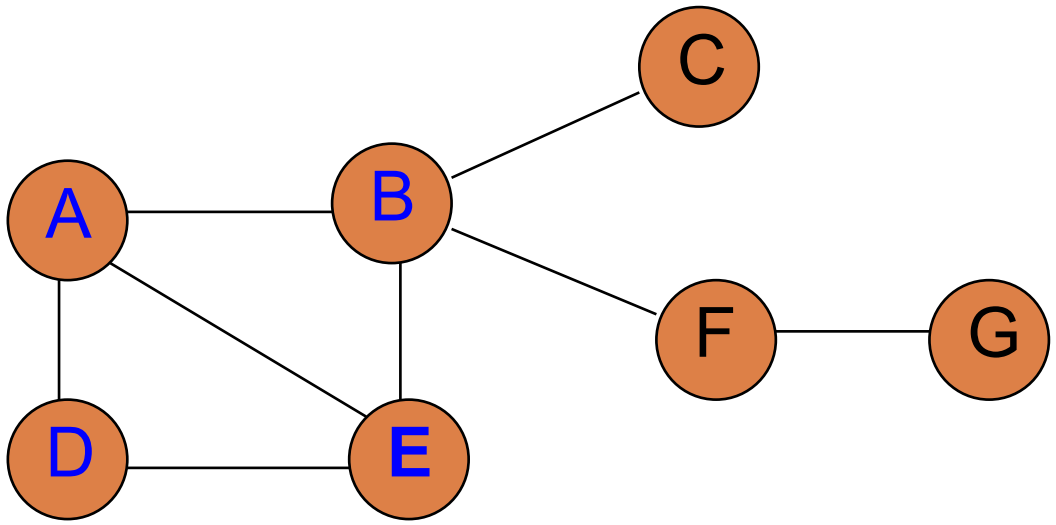


toVisit-queue: E C E F E

visited: A B D E

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

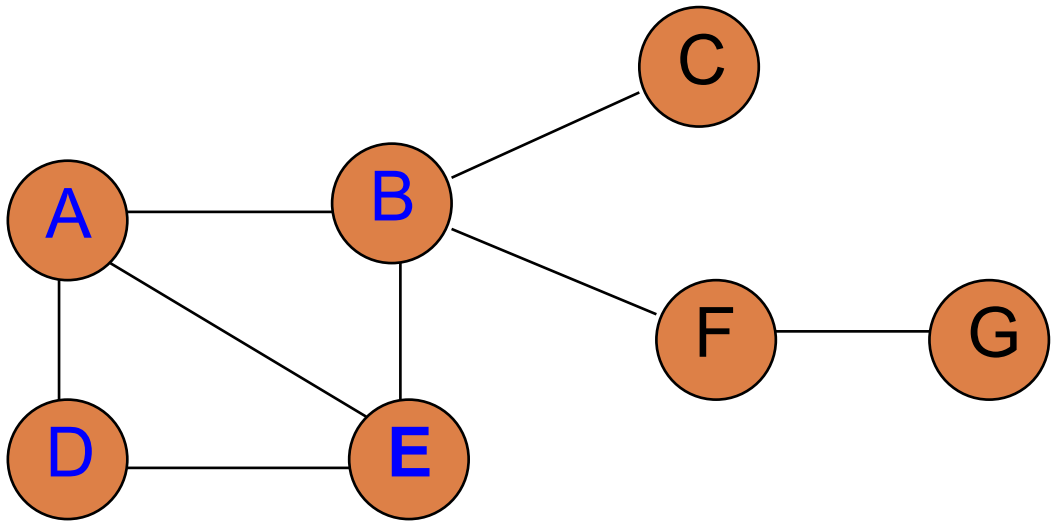


toVisit-queue: E C E F E

visited: A B D E

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```



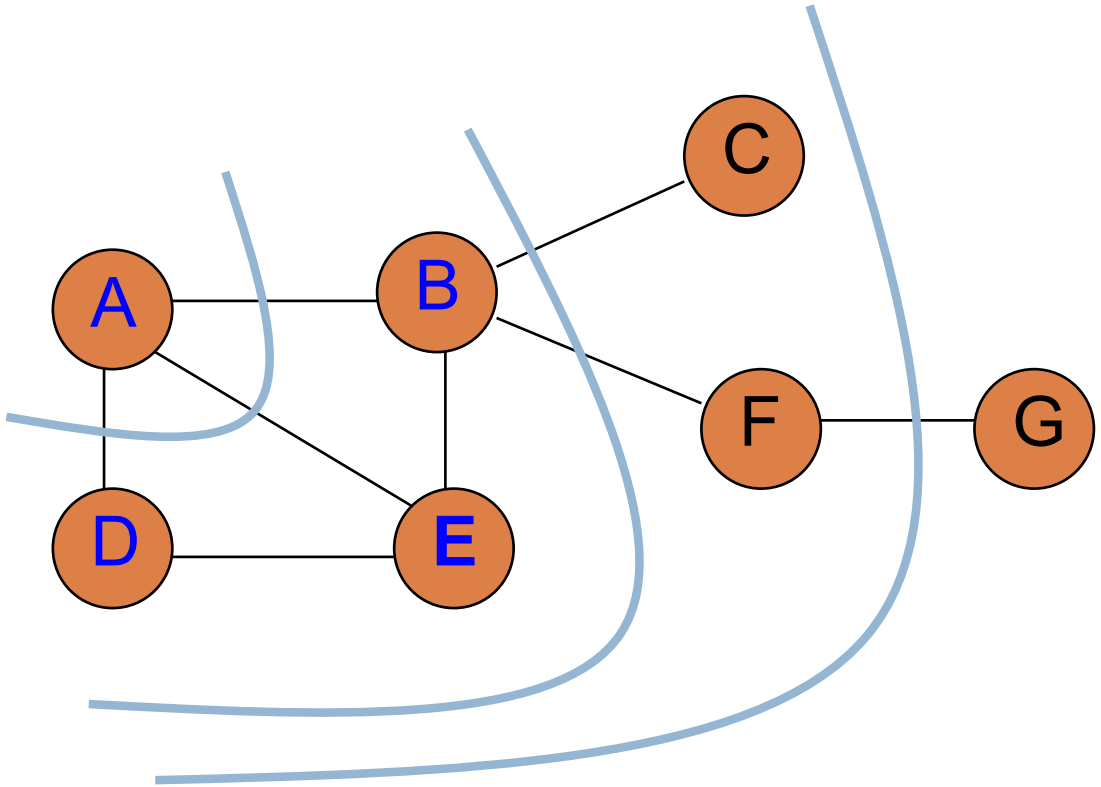
toVisit-queue: E C E F E

visited: A B D E

No adjacent vertices that haven't been visited

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```



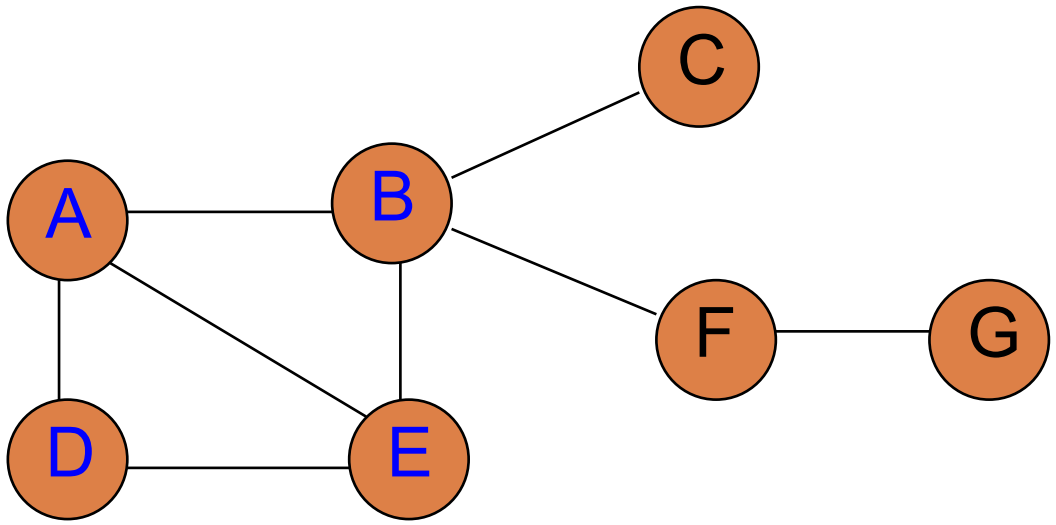
toVisit-queue: E C E F E

visited: A B D E

Frontier: all vertices a given number of edges from the start/root

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

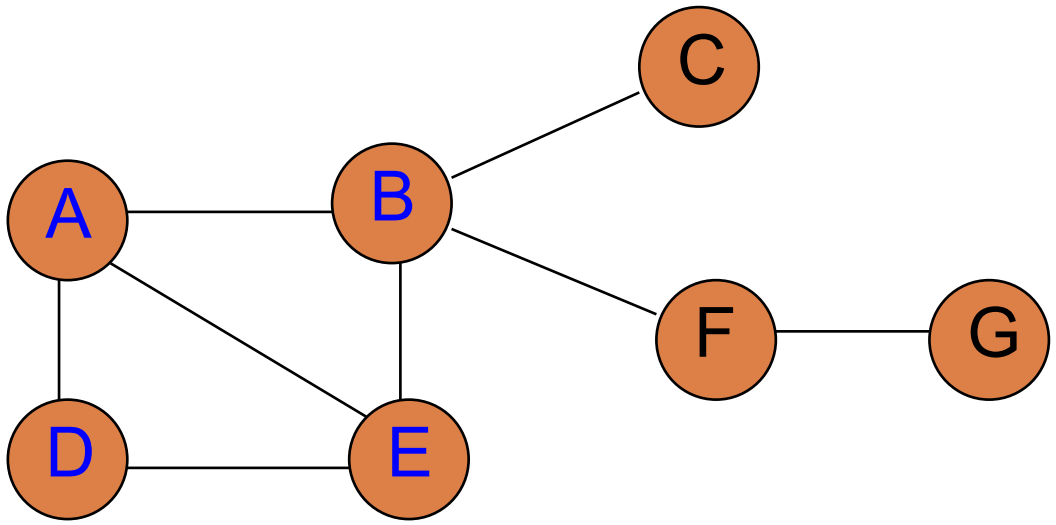


toVisit-queue: E C E F E

visited: A B D E

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```



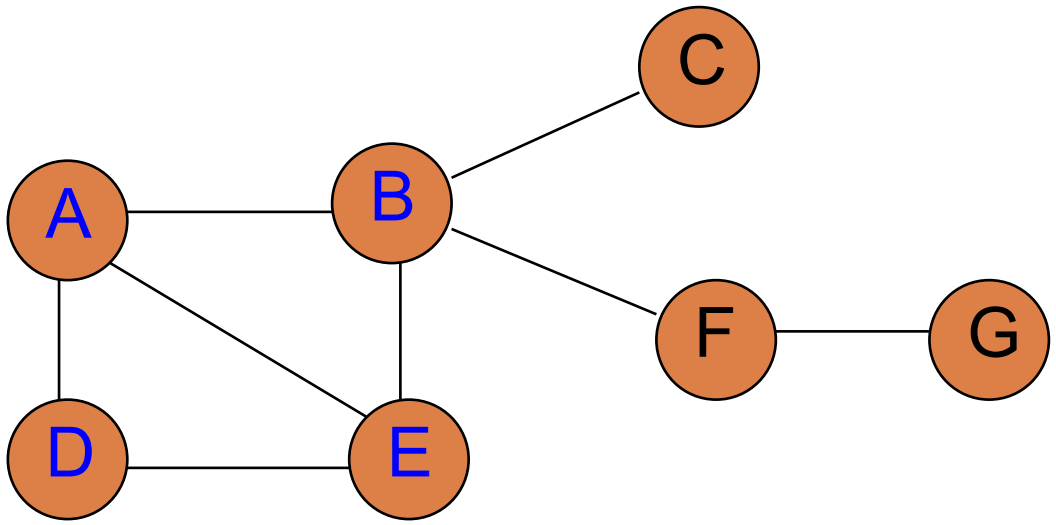
toVisit-queue: C E F E

visited: A B D E

E has already been visited

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

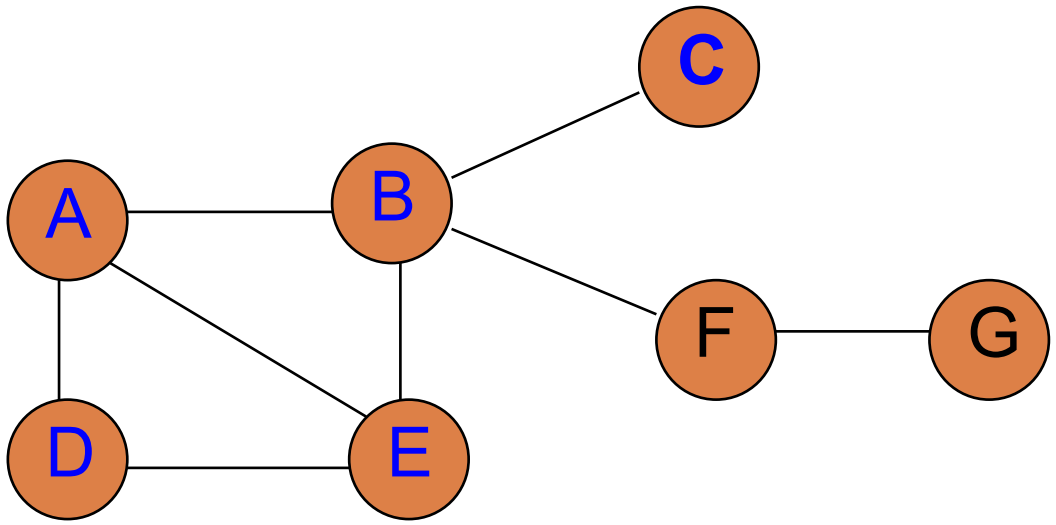


toVisit-queue: C E F E

visited: A B D E

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

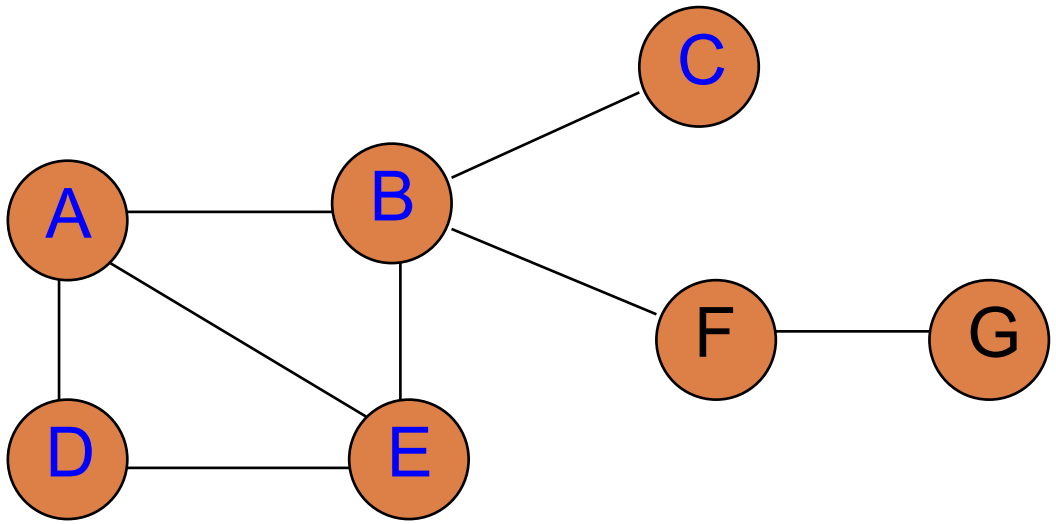


toVisit-queue: E F E

visited: A B D E C

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

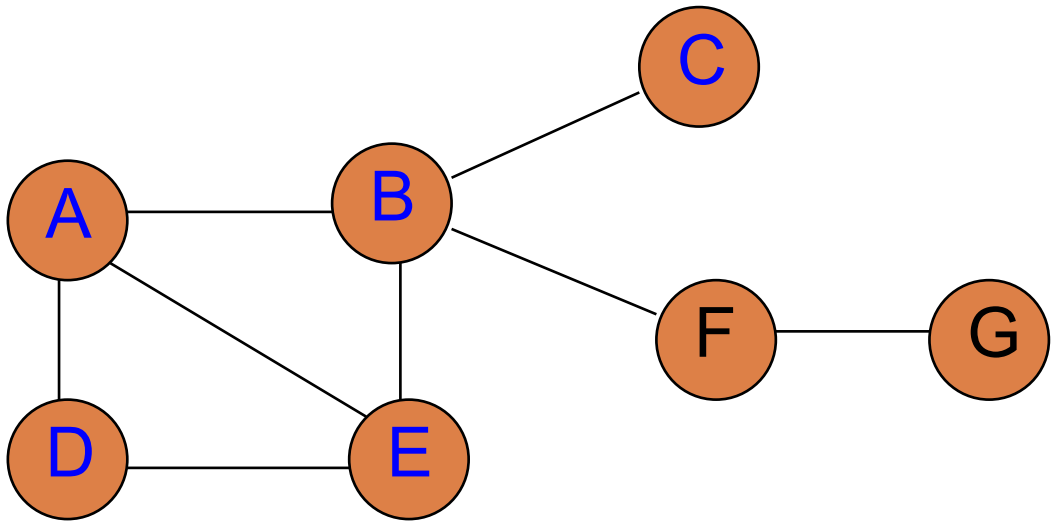


toVisit-queue: E F E

visited: A B D E C

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```



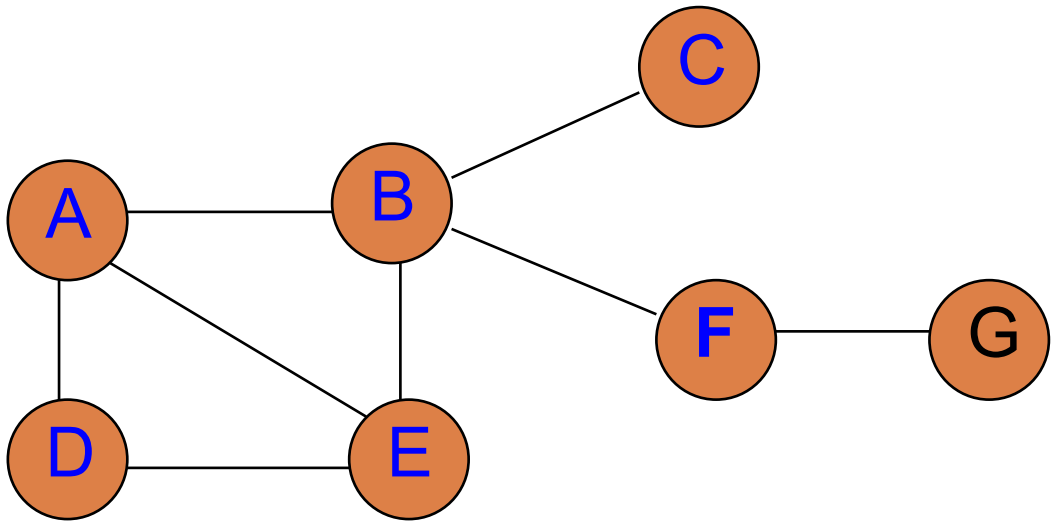
toVisit-queue: F E

visited: A B D E C

E has already been visited

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

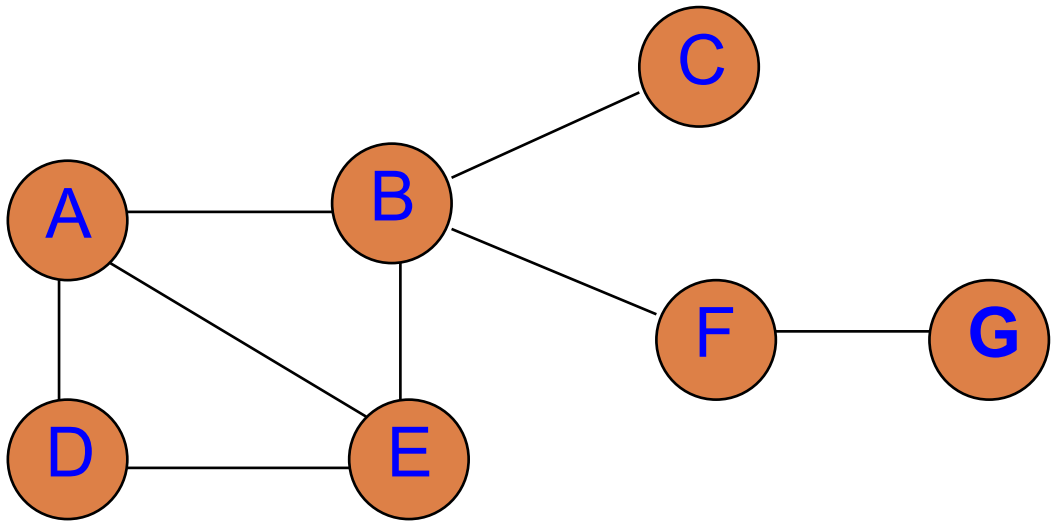


toVisit-queue: E G

visited: A B D E C F

BFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

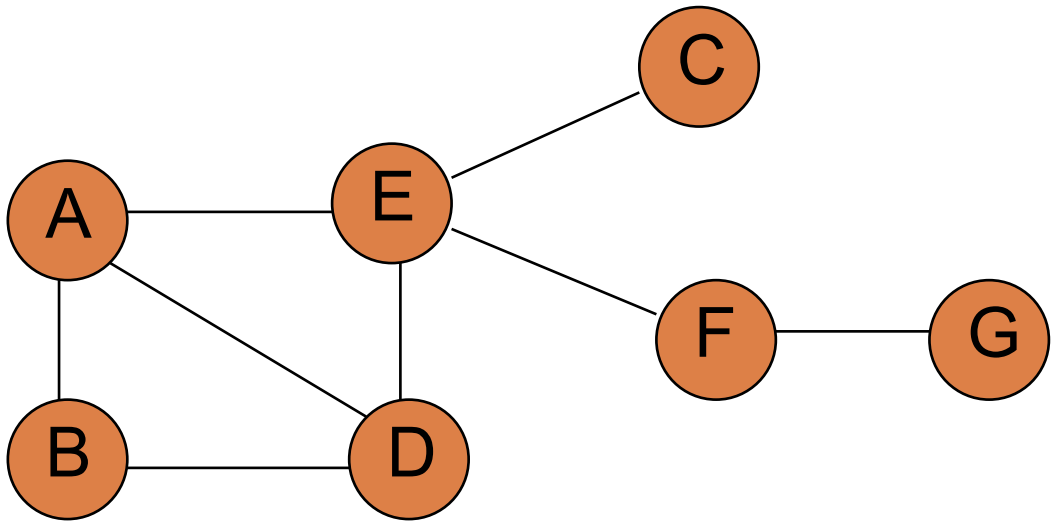


toVisit-queue:

visited: A B D E C F G

DFS

```
graphDFS( start )
    s = new Stack()
    s.add(start)
    treeSearch(s)
```

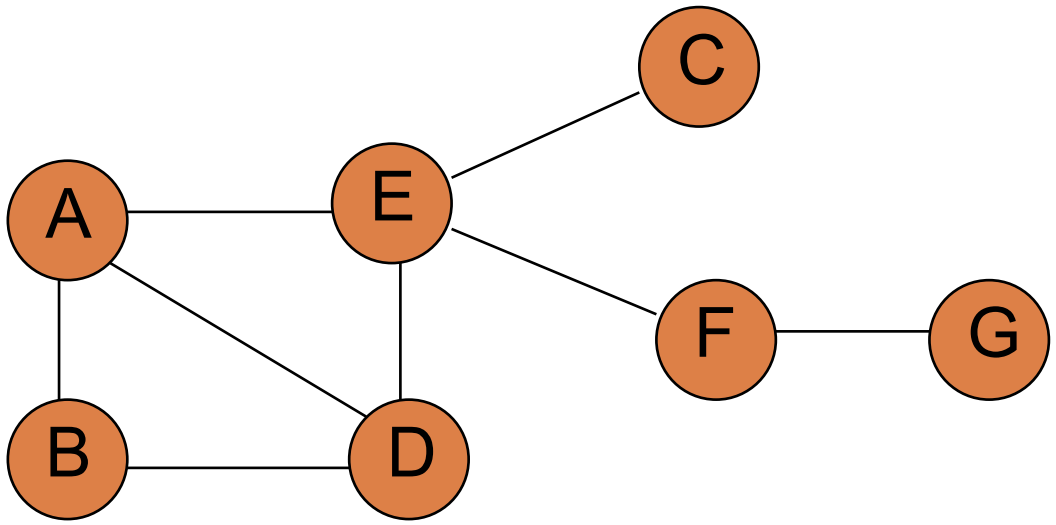


toVisit-stack: A

visited:

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```



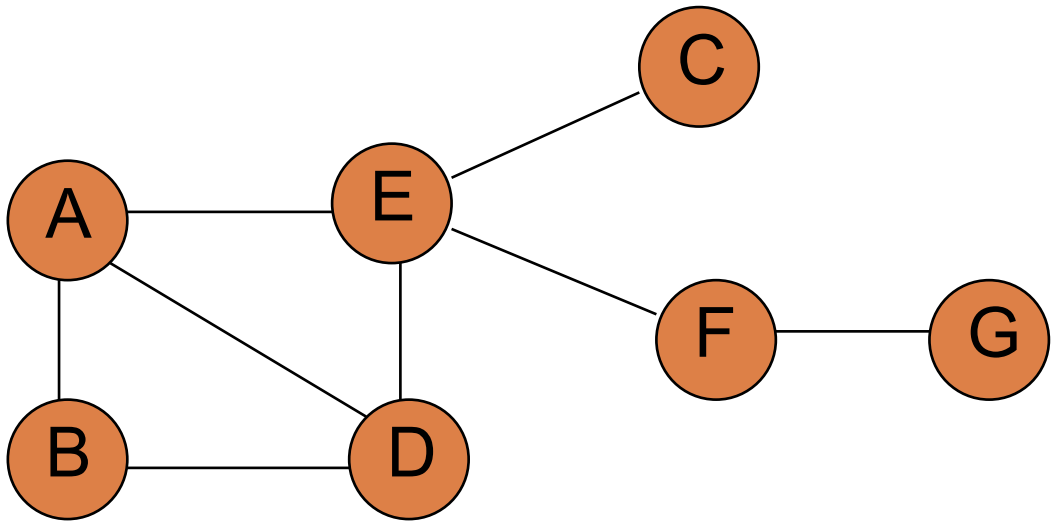
toVisit-stack: A

visited:

What order will the nodes get printed out?
Assume edges are traversed alphabetically.

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

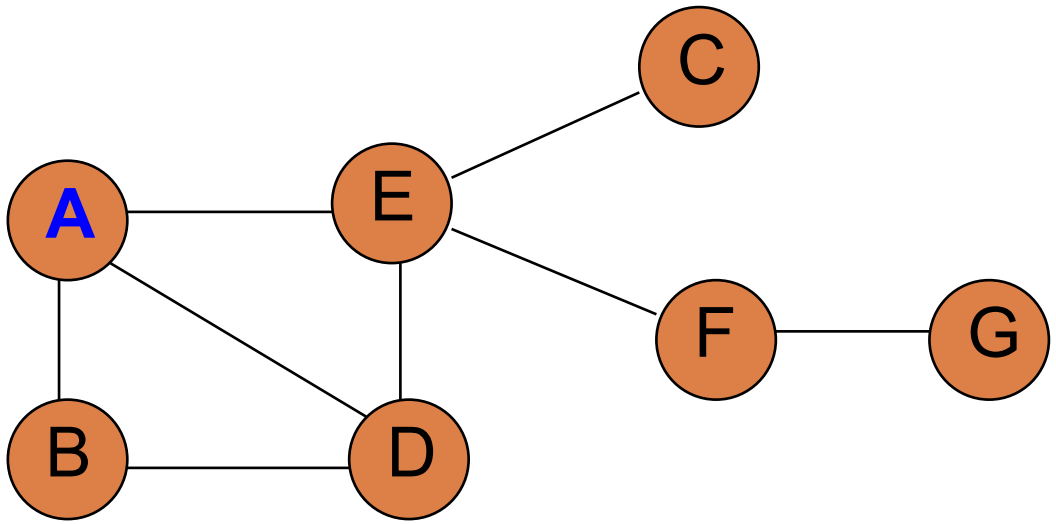


toVisit-stack: A

visited:

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

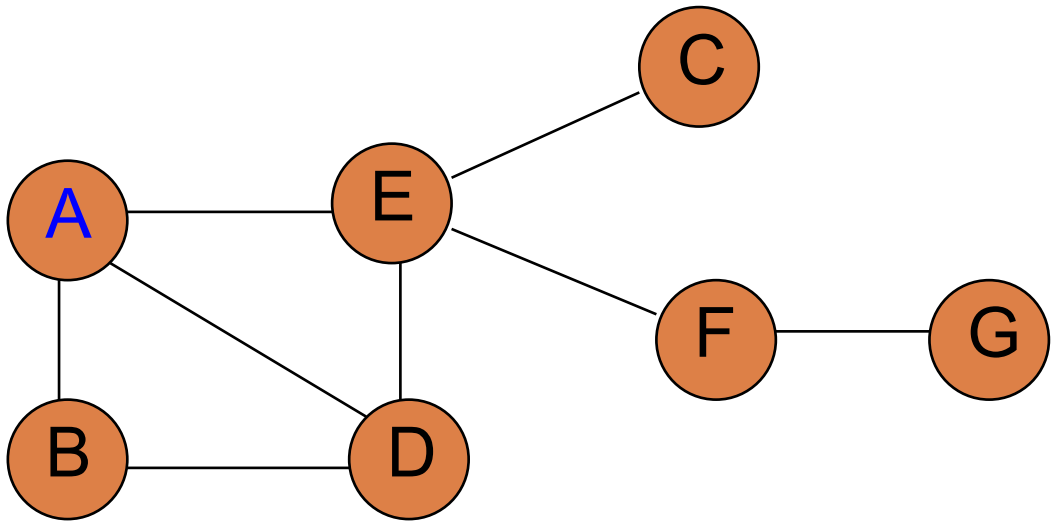


toVisit-stack: B D E

visited: A

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

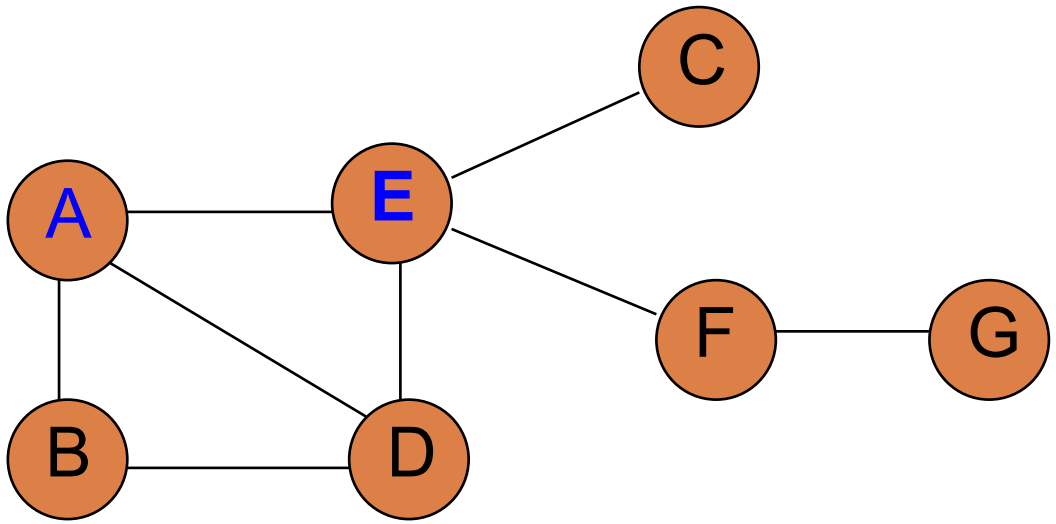


toVisit-stack: B D E

visited: A

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

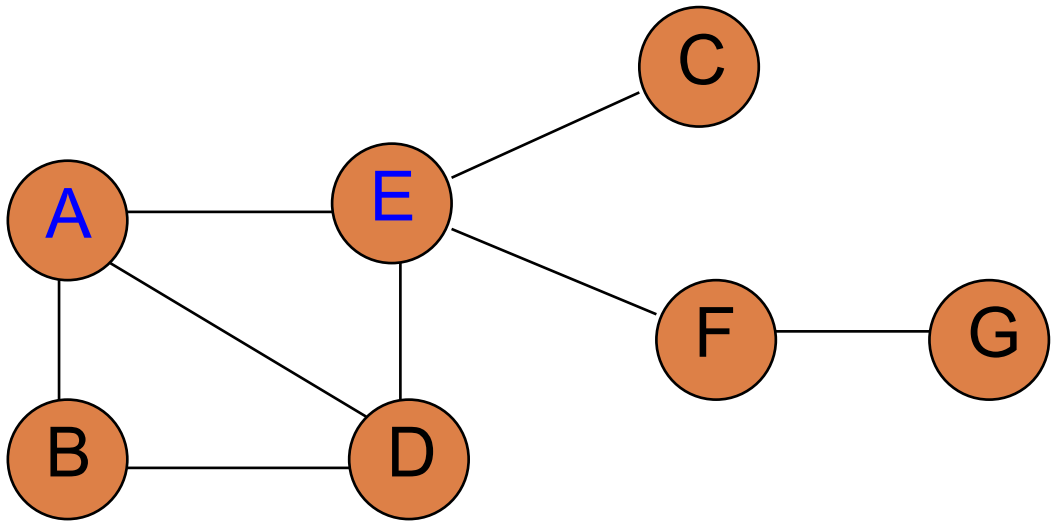


toVisit-stack: B D C D F

visited: A E

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

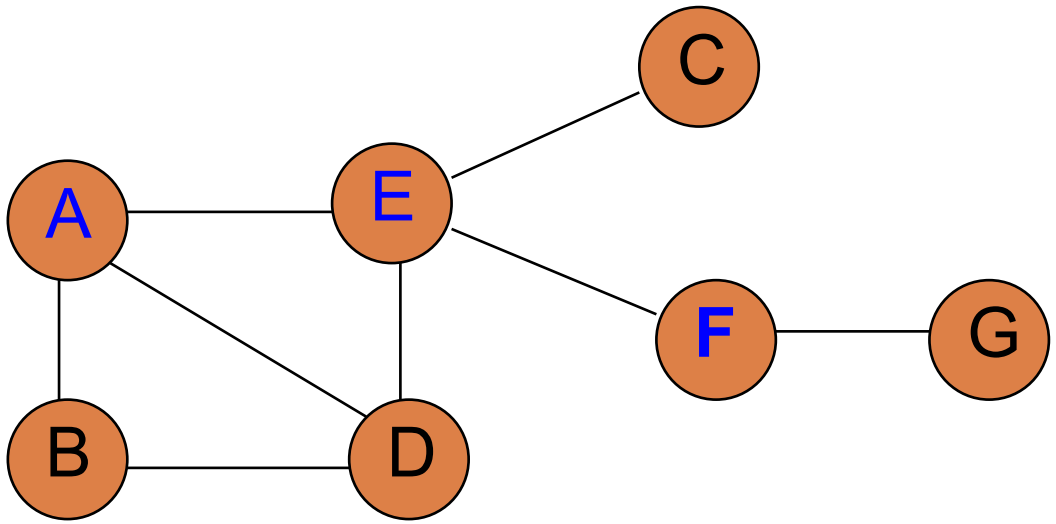


toVisit-stack: B D C D F

visited: A E

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

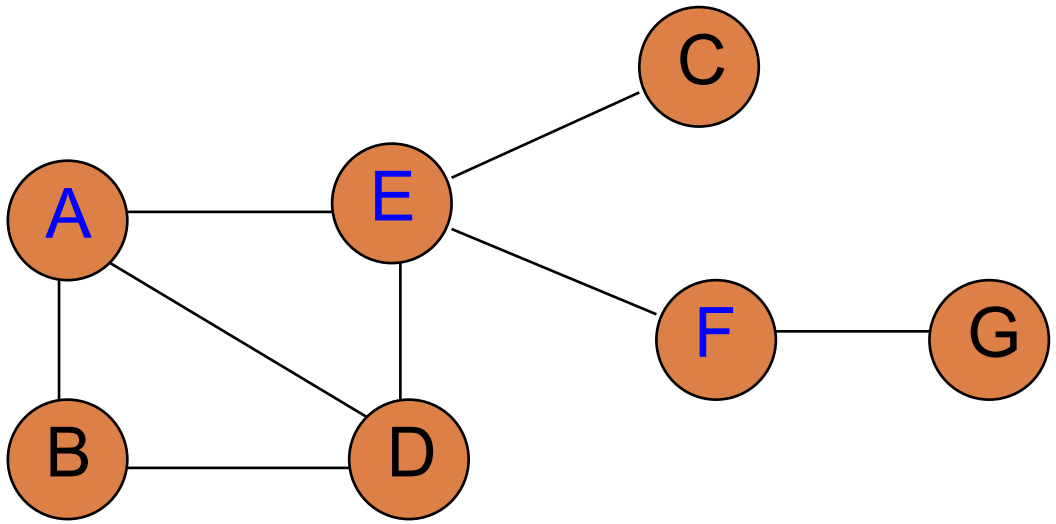


toVisit-stack: B D C D G

visited: A E F

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

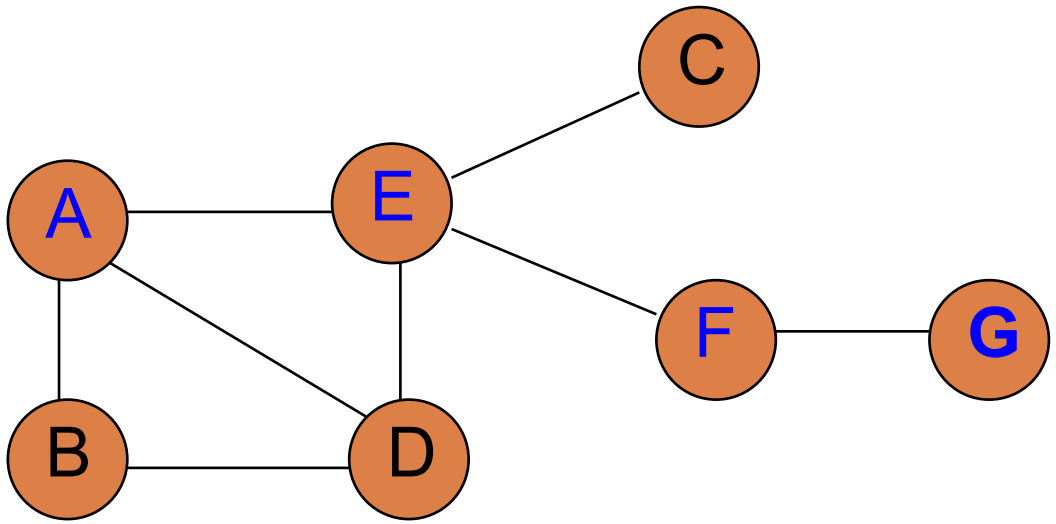


toVisit-stack: B D C D G

visited: A E F

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

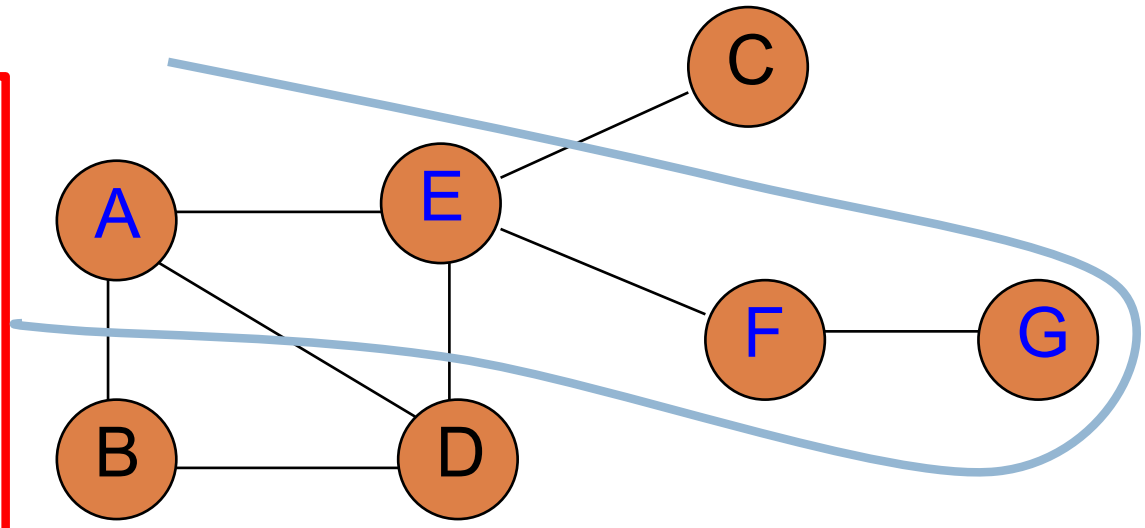


toVisit-stack: B D C D

visited: A E F G

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```



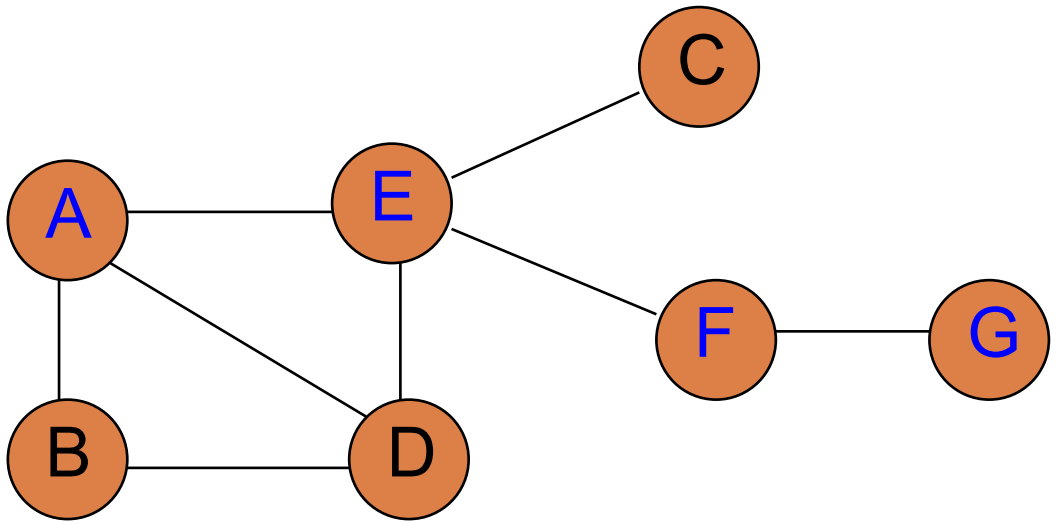
toVisit-stack: B D C D

visited: A E F G

Frontier: Go as far down one path as possible

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

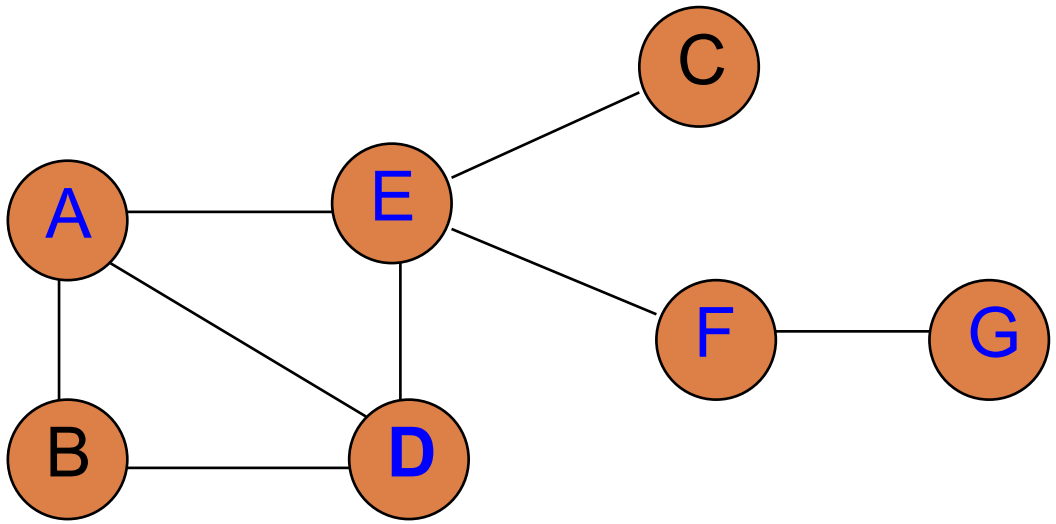


toVisit-stack: B D C D

visited: A E F G

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

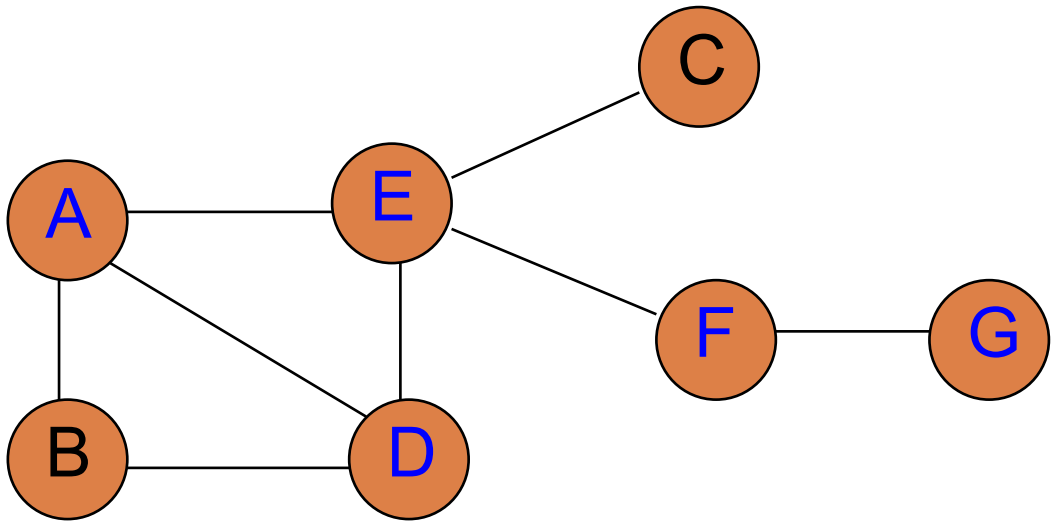


toVisit-stack: B D C B

visited: A E F G D

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

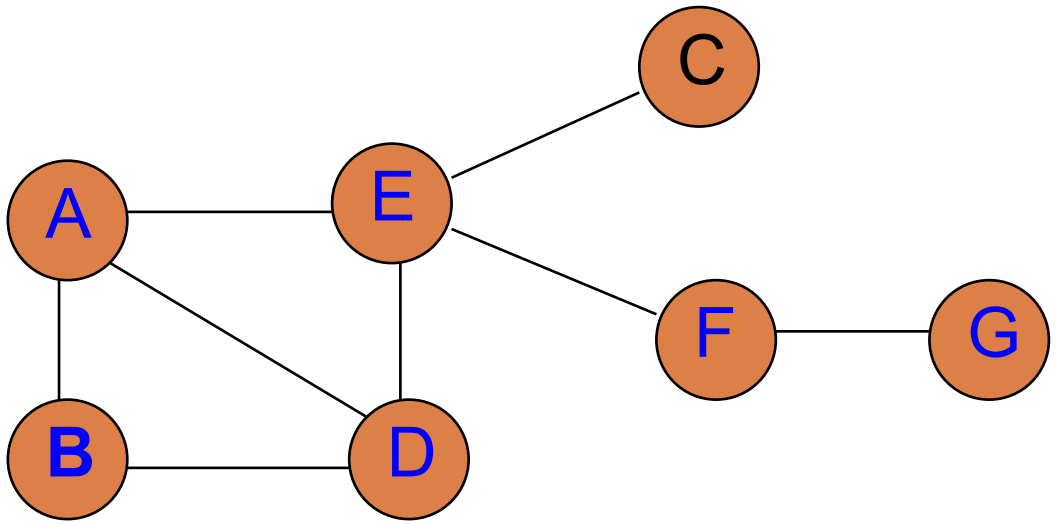


toVisit-stack: B D C B

visited: A E F G D

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

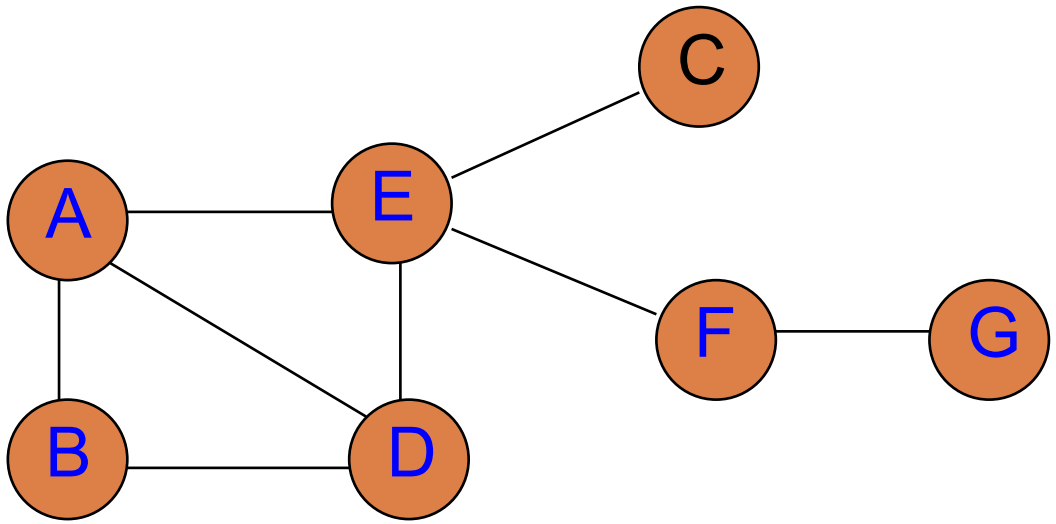


toVisit-stack: B D C

visited: A E F G D B

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

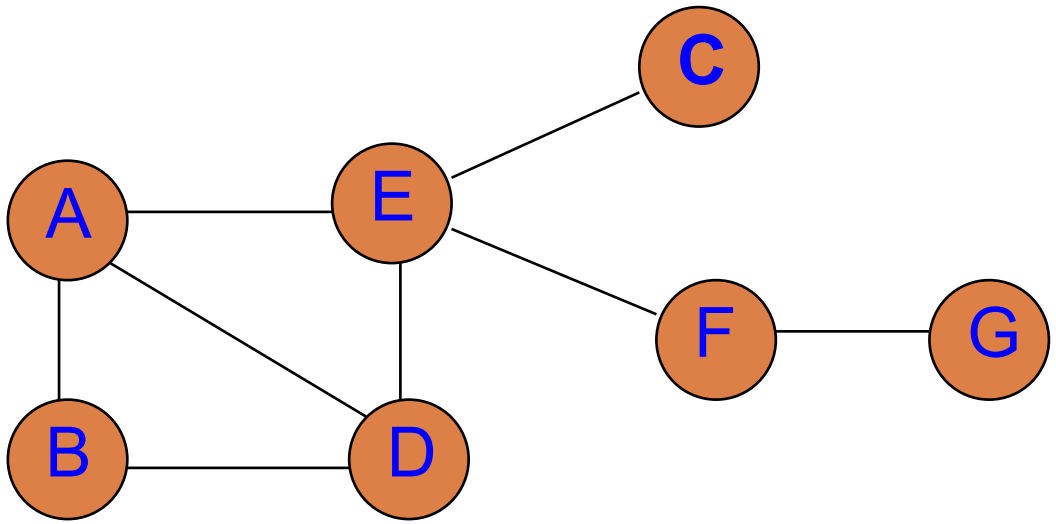


toVisit-stack: B D C

visited: A E F G D B

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

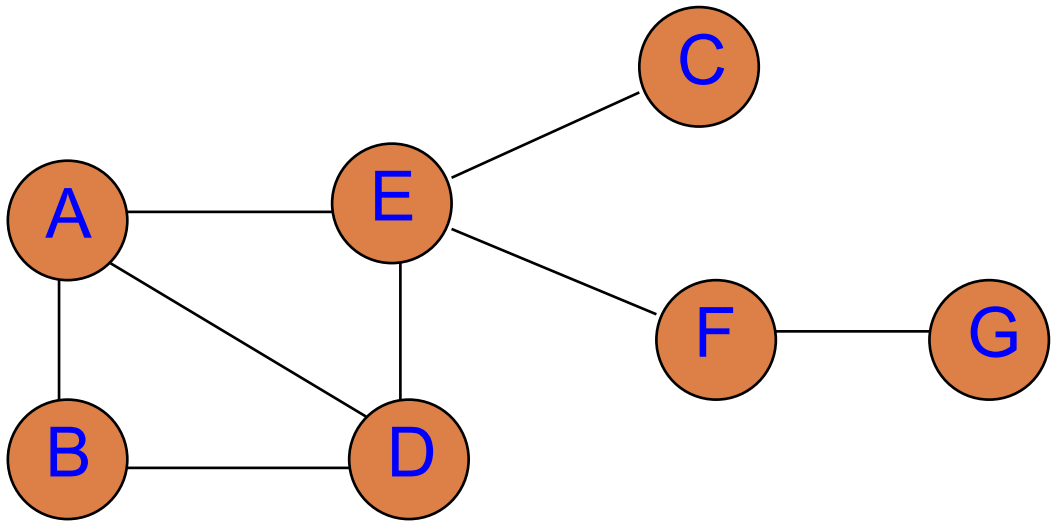


toVisit-stack: B D

visited: A E F G D B C

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

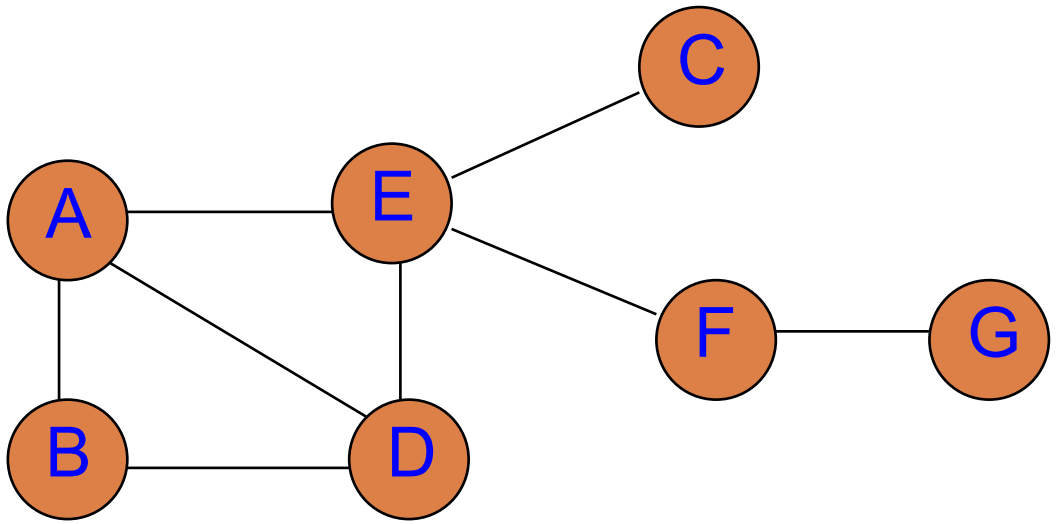


toVisit-stack: B D

visited: A E F G D B C

DFS

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```



toVisit-stack:

visited: A E F G D B C

graphSearch run-time

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

What is the big-O run-time of graphSearch?

Assume all of the stack/queue operations are constant.

How many times do we visit each vertex?

How many times do we traverse each edge (vis the for loop)?

graphSearch run-time

```
graphSearch( toVisit )
    while !toVisit.empty()
        v = toVisit.remove()
        if !visited[v]
            visited[v] = true
            for c in v.getAdjacent()
                if !visited[c]
                    toVisit.add(c)
```

How many times do we visit each vertex? Exactly once

How many times do we traverse each edge (vis the for loop)? Exactly once

What is the big-O run-time of teeSearch? O(|V| + |E|). Linear algorithm.

Nothing changes from treeSearch!