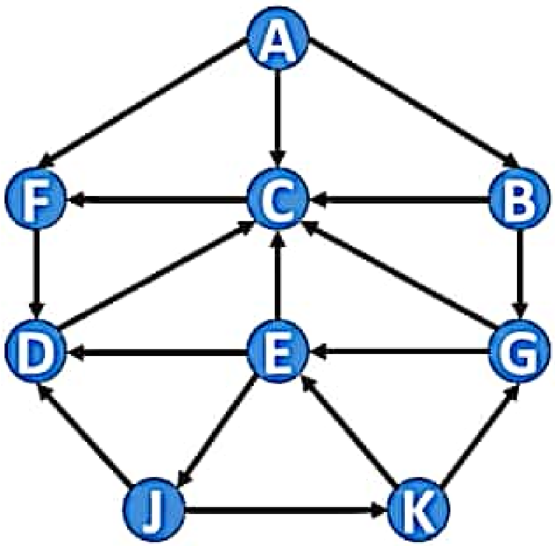
Depth-First Search (DFS)



Algorithm

```
1. Initialize all nodes to ready state (STATUS = 1)
2. Push the starting node A onto STACK and change its status to waiting state (STATUS = 2)
3. Repeat Steps 4 and 5 until STACK is empty:
4.         Pop the top node N of STACK. Process N and change its status to the processed state
   (STATUS = 3).
5.         Push onto STACK all the neighbors of N that are still in the ready state (STATUS = 1), and
   change their status to the waiting state (STATUS = 2).
   [End of Step 3 loop.]
6. Exit.
```

- Consider the graph G in Fig. A. (The adjacency lists of the nodes appear in Fig. B.)
- Suppose we want to find and print all the nodes reachable from the node J (including J itself). One way to do this is to use a DFS of G starting at the node J. The steps of our search follow.

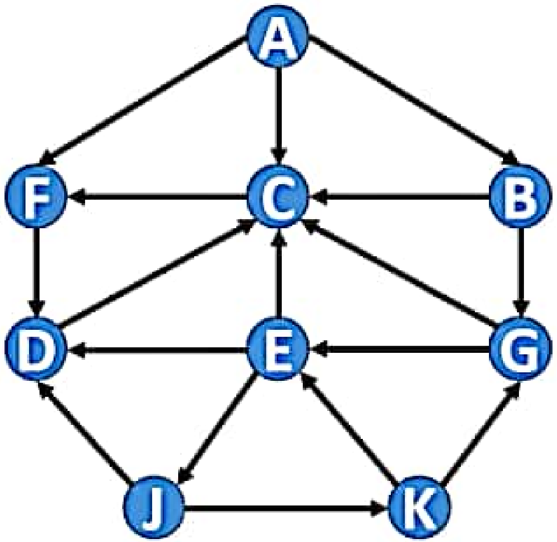
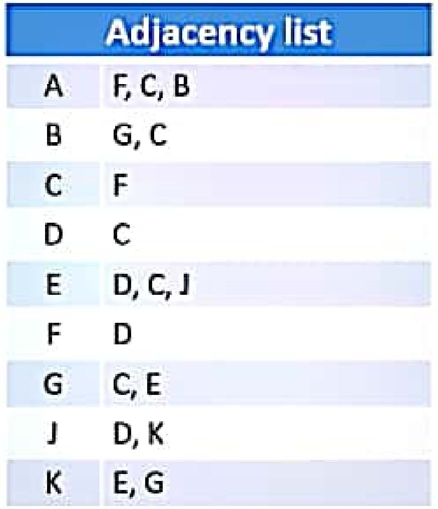


Fig. A



Adjacency list	
A	F, C, B
B	G, C
C	F
D	C
E	D, C, J
F	D
G	C, E
J	D, K
K	E, G

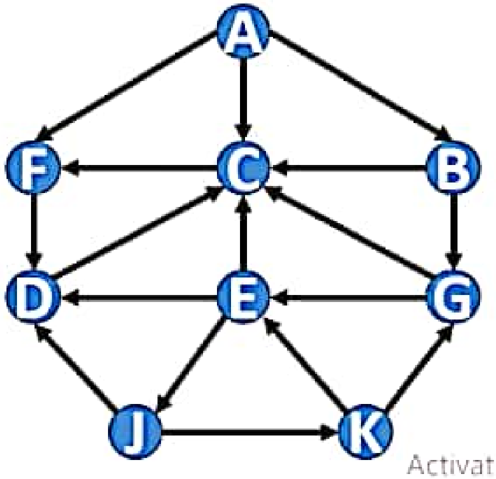
Fig. B

(A)

- Initially, push J onto the stack as follows:

✓STACK: J

```
1. Initialize all nodes to ready state (STATUS = 1)
2. Push the starting node A onto STACK and change its status to waiting state (STATUS = 2)
3. Repeat Steps 4 and 5 until STACK is empty:
4.     Pop the top node N of STACK. Process N and change its status to the processed state (STATUS = 3).
5.     Push onto STACK all the neighbors of N that are still in the ready state (STATUS = 1), and change their status to the waiting state (STATUS = 2).
   [End of Step 3 loop.]
6. Exit.
```



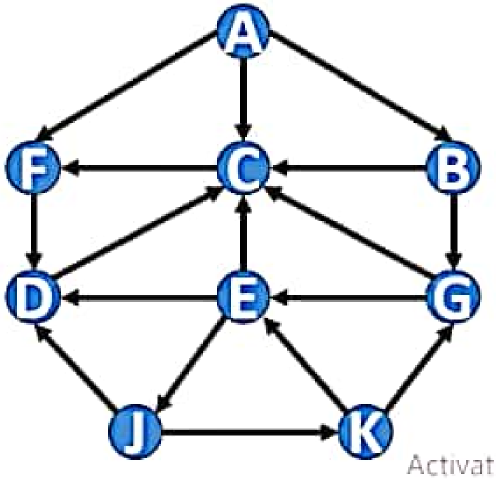
(B)

- Pop and print the top element J, and then push onto the stack all the neighbors of J (those that are in the ready state) as follows:

✓Print J

✓STACK: D, K

```
1. Initialize all nodes to ready state (STATUS = 1)
2. Push the starting node A onto STACK and change its status to waiting state (STATUS = 2)
3. Repeat Steps 4 and 5 until STACK is empty:
4.         Pop the top node N of STACK. Process N and change its status to the processed state (STATUS = 3).
5.         Push onto STACK all the neighbors of N that are still in the ready state (STATUS = 1), and change their status to the waiting state (STATUS = 2).
   [End of Step 3 loop.]
6. Exit.
```



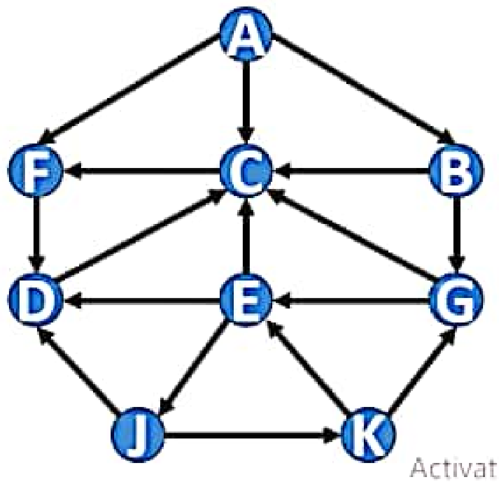
(C)

- Pop and print the top element K, and then push onto the stack all the neighbors of K (those that are in the ready state) as follows:

✓Print K

✓STACK: D, E, G

1. Initialize all nodes to ready state (STATUS = 1)
2. Push the starting node A onto STACK and change its status to waiting state (STATUS = 2)
3. Repeat Steps 4 and 5 until STACK is empty:
4. Pop the top node N of STACK. Process N and change its status to the processed state (STATUS = 3).
5. Push onto STACK all the neighbors of N that are still in the ready state (STATUS = 1), and change their status to the waiting state (STATUS = 2).
   [End of Step 3 loop.]
6. Exit.



(D)

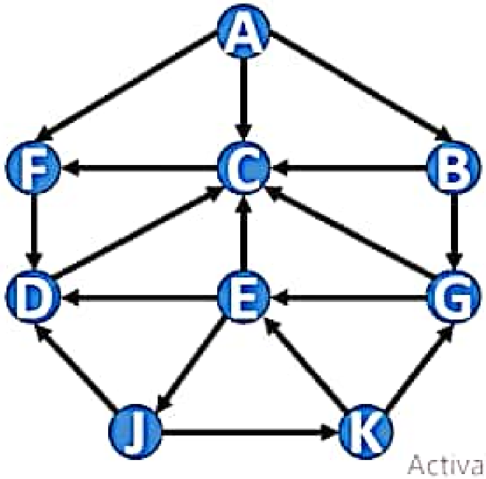
- Pop and print the top element G, and then push onto the stack all the neighbors of G (those that are in the ready state) as follows:

✓Print G

✓STACK: D, E, C

- Note that only C is pushed onto the stack, since the other neighbor, E, is not in the ready state (because E has already been pushed onto the stack).

```
1.  Initialize all nodes to ready state (STATUS = 1)
2.  Push the starting node A onto STACK and change its
    status to waiting state (STATUS = 2)
3.  Repeat Steps 4 and 5 until STACK is empty:
4.          Pop the top node N of STACK. Process N and
    change its status to the processed state (STATUS = 3).
5.          Push onto STACK all the neighbors of N that are
    still in the ready state (STATUS = 1), and change their
    status to the waiting state (STATUS = 2).
    [End of Step 3 loop.]
6.  Exit.
```



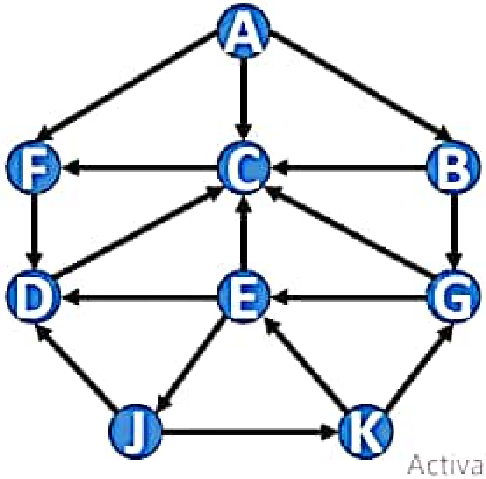
(E)

- Pop and print the top element C, and then push onto the stack all the neighbors of C (those that are in the ready state) as follows:

✓Print C

✓STACK: D, E, F

```
1. Initialize all nodes to ready state (STATUS = 1)
2. Push the starting node A onto STACK and change its status to waiting state (STATUS = 2)
3. Repeat Steps 4 and 5 until STACK is empty:
4.     Pop the top node N of STACK. Process N and change its status to the processed state (STATUS = 3).
5.     Push onto STACK all the neighbors of N that are still in the ready state (STATUS = 1), and change their status to the waiting state (STATUS = 2).
   [End of Step 3 loop.]
6. Exit.
```



(F)

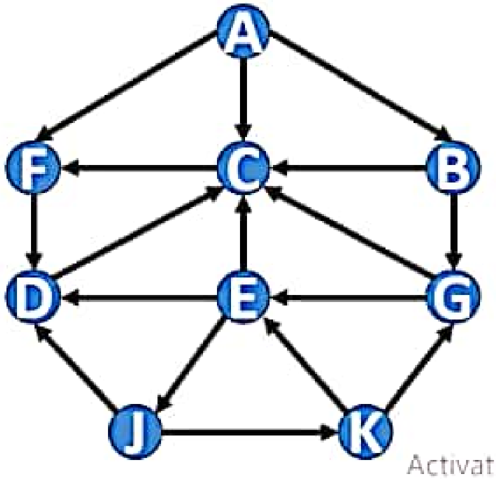
- Pop and print the top element F, and then push onto the stack all the neighbors of F (those in the ready state) as follows:

✓Print F

✓STACK: D, E

- Note that only neighbor D of F is not pushed onto the stack, since D is not in the ready state (because D has already been pushed onto the stack) .

1. Initialize all nodes to ready state (STATUS = 1)
2. Push the starting node A onto STACK and change its status to waiting state (STATUS = 2)
3. Repeat Steps 4 and 5 until STACK is empty:
4. Pop the top node N of STACK. Process N and change its status to the processed state (STATUS = 3).
5. Push onto STACK all the neighbors of N that are still in the ready state (STATUS = 1), and change their status to the waiting state (STATUS = 2).
[End of Step 3 loop.]
6. Exit.



(G)

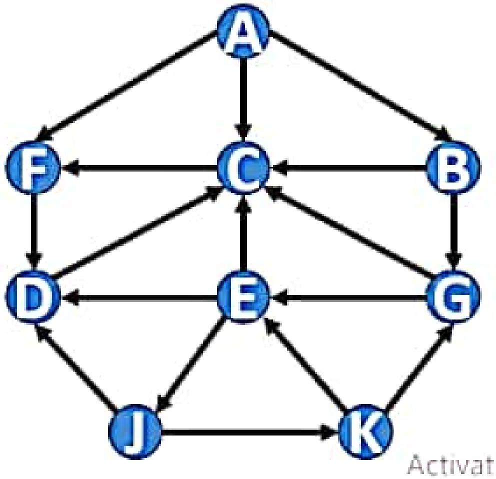
- Pop and print the top element E, and push onto the stack all the neighbors of E (those in the ready state) as follows:

✓Print E

✓STACK: D

- Note that none of the three neighbors of E is in the ready state.

1. Initialize all nodes to ready state (STATUS = 1)
2. Push the starting node A onto STACK and change its status to waiting state (STATUS = 2)
3. Repeat Steps 4 and 5 until STACK is empty:
4. Pop the top node N of STACK. Process N and change its status to the processed state (STATUS = 3).
5. Push onto STACK all the neighbors of N that are still in the ready state (STATUS = 1), and change their status to the waiting state (STATUS = 2).
   [End of Step 3 loop.]
6. Exit.



(H)

- Pop and print the top element D, and push onto the stack all the neighbors of D (those in the ready state) as follows:

✓Print D

✓STACK:

- The stack is now empty, so the DFS of G starting at J is now complete. Accordingly, the nodes which were printed,

J, K, G, C, F, E, D

are precisely the nodes which are reachable from J

```
1.  Initialize all nodes to ready state (STATUS = 1)
2.  Push the starting node A onto STACK and change its
    status to waiting state (STATUS = 2)
3.  Repeat Steps 4 and 5 until STACK is empty:
4.          Pop the top node N of STACK. Process N and
    change its status to the processed state (STATUS = 3).
5.          Push onto STACK all the neighbors of N that are
    still in the ready state (STATUS = 1), and change their
    status to the waiting state (STATUS = 2).
    [End of Step 3 loop.]
6.  Exit.
```