~~Travel~~

Traversing a graph

→ BFS

→ DFS

BFS (Breadth first search)

(i) Derive adjacency list of a given graph G.

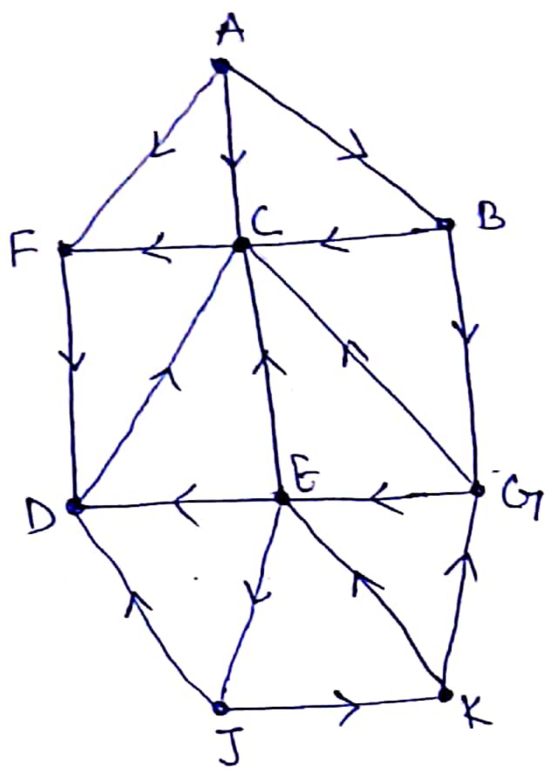
(ii) Now starting node A, kept in a queue with its origin.

(iii) Now remove the node from front of queue and place its neighbors, from adjacency list, at the rear end of queue.

(iv) If any neighbor has been processed or has come in queue, then do not add again it.

(v) Repeat steps (iii) & (iv) until destination node is not reached.

(vi) Back track from destination node to starting node with the help of origin.



Nodes	Neighbors
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

(i) Queue : A

ORIG : Ø

(ii) Remove A.

Queue : ~~A~~, ~~F~~, C, B, D

ORIG : Ø, A, A, A, F

(iii) Remove F & C

Queue : ~~A~~, ~~F~~, ~~C~~, B, D, !

ORIG : Ø, A, A, A, F,

(iv) Remove B

Queue : ~~A~~, ~~F~~, ~~C~~, B, D, G

ORIG : Ø, A, A, A, F, B

(v) Remove D

Queue : ~~A~~, ~~F~~, ~~C~~, ~~B~~, ~~D~~, G

ORIG ; Ø, A, A, A, F, B

(vi) Remove G

Queue : ~~A~~, ~~F~~, ~~C~~, ~~B~~, ~~D~~, ~~G~~, E

ORIG : Ø, A, A, A, F, B, G

Start

(vii) Remove E

Queue : A, F, C, B, D, G, E, J]

ORIG : Ø, A, A, A, F, B, G, E

End.

~~G~~ $A \longrightarrow B \longrightarrow G \longrightarrow E \longrightarrow J$

DFS (Depth-First Search)

Steps

(i) Find out adjacency list of a given graph

(ii) Push starting element into ~~graph~~ Stack

(iii) Pop the top element & print it & push all its ~~neighb~~ neighbor in stack.

(iv) Repeat step (iii) until stack is not empty.

Previous Example

(i) Initially push J into Stack

(ii) Pop, Top element from Stack & print it & push it's neighbours.

STACK : D, K.

Print J

(iii) Now, pop, top element (K) from Stack & print & push its neighbours.

STACK : D, E, G.

Print K

(iv) Now, pop, top element (G) from Stack & print it's neighbours.

STACK : D, E, C.

Print G

(v) Pop the C and print it & push it's neighbour

Print C   STACK: D, E, F

(vi) Pop the F from Stack & print it & push its neighbour.

Print F . STACK: D, E.

Since D is already in STACK.

(vii) Pop the element E from Stack & print it & push its neighbour.

print E   STACK: D

Since D, C, J ~~and~~ all are processed.

(viii) pop D from Stack & print it & push its neighbour.

print D   STACK: —

Since C is already processed.

Now, sequence are

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