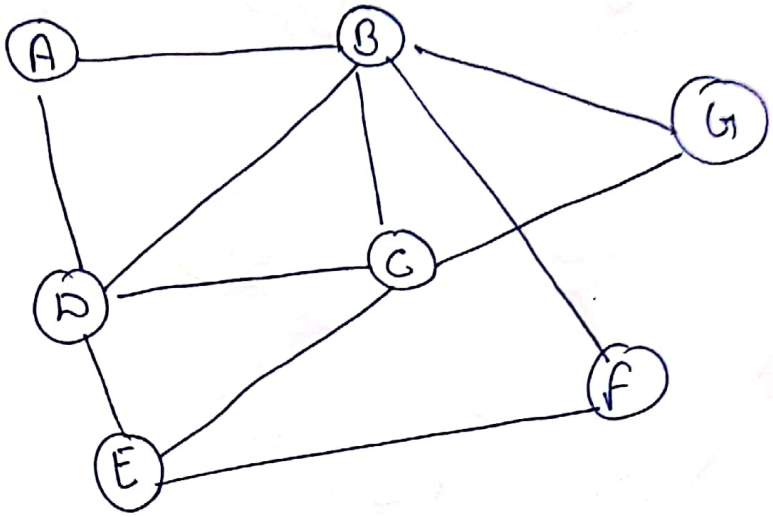
Depth first Search

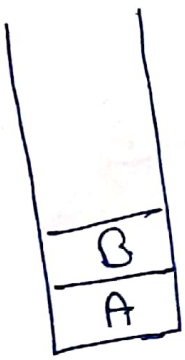


Stack data structure is used.

Starting Vertex = A

Step 1

Push the starting vertex (A) onto stack. Then next visited vertex is pushed onto stack

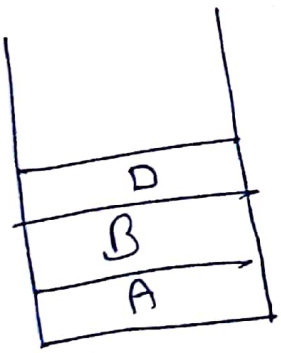


Result : A

Step 2.

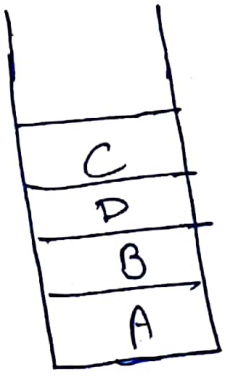
Now, visit B & that Push the next adjacent vertex of B which is unvisited is pushed onto stack and B is in the result set.

Result A B



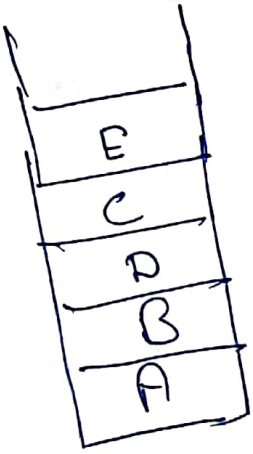
Step 3. Then, visit D and push one of its unvisited vertex onto stack which is 'C' and D onto result set i.e.

Result: A B D



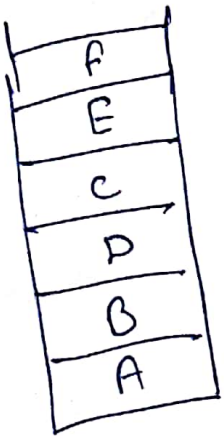
Step 4 Now visit C and push its unvisited vertex which is E onto stack and insert C onto result set.

Result: A B D C.



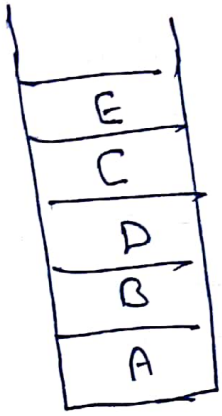
Step 5 Now, visit E and push its unvisited vertex which is F onto stack and E onto result set

Result: A B D C E



Step 6. Now visit F, There is no unvisited vertex of F i.e. no adjacent vertex which is unvisited. Push F onto resultset

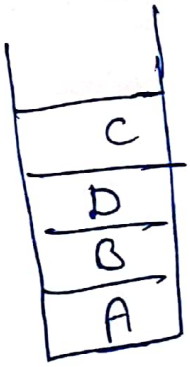
Result A B D C E F



Now Pop f from stack and backtrack from f to E

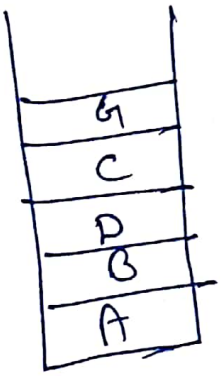
Step 7 Since all adjacent vertex of E are processed. So Pop, E from stack and backtrack at C.

Result A B D C E f

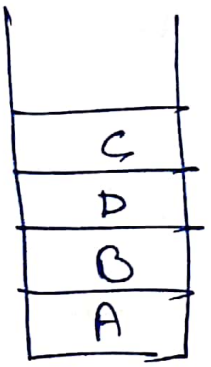


Step 8 Now visit C, and Push its unvisited vertex which is G onto stack.

Result A B D C E F G



Step 9. Now visit G, all its adjacent vertex are processed. So pop G from stack and backtrack at C



Step 10 Now visit C, all its adjacent vertex are processed. So pop C and backtrack at D.

Result A B D C E f G

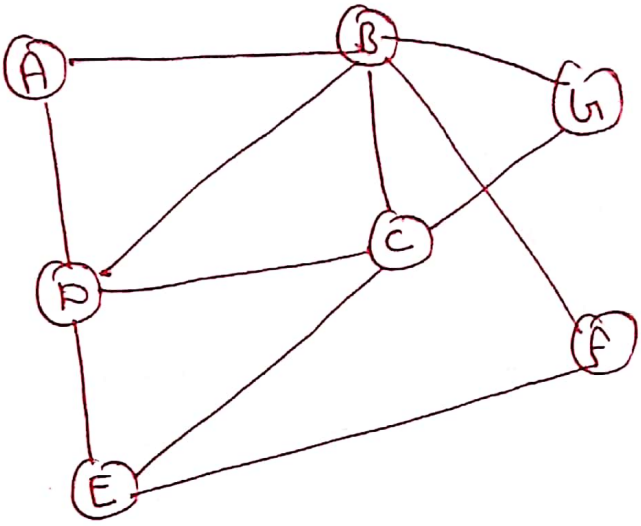
Step 11 Visit D, all its adjacent vertex are processed. So pop D, backtrack at B.

Step 12 Visit B, all its adjacent vertex are processed. So pop B and backtrack A.

Step 13. visit A, all its adjacent vertex are processed. Now pop A. Now stack is empty which is the terminating criteria.

Result A B D C E f G

Breadth First Search

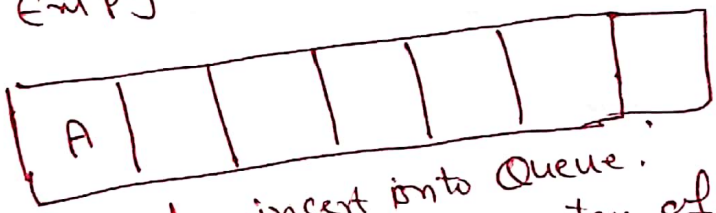


Here Queue data structure is used.

Starting vertex = A.

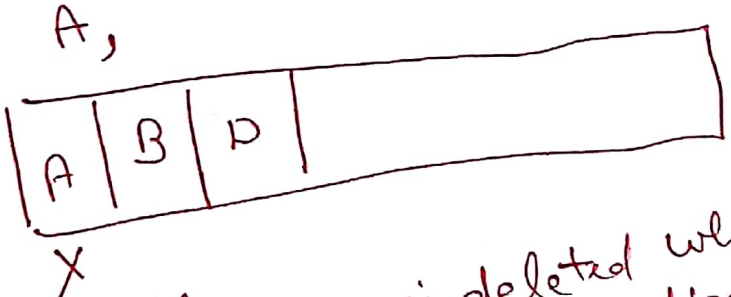
Initially Queue is Empty.

Queue:



Step 1: Since A is the Source node, insert into Queue.
Explore all the adjacent vertex of 'A'.
Remove A from Queue and insert all its adjacent vertex onto Queue.

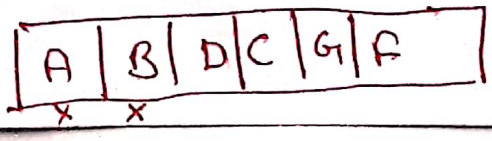
Result : A,



Step 2: front element of Queue is deleted which is B. And Insert all unvisited adjacent vertex of B onto Queue.

Result : A, B

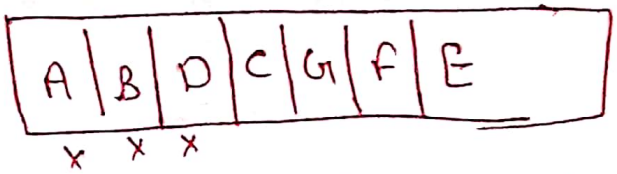
Queue:



Step 3. Front element of Queue which is D is removed and all its adjacent vertices are inserted onto queue.

Result A, B, D

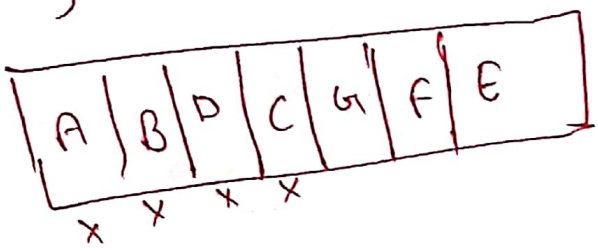
Queue:



Step 4. Front element C is removed and insert its unvisited adjacent vertex to Queue.

Result A, B, D, C

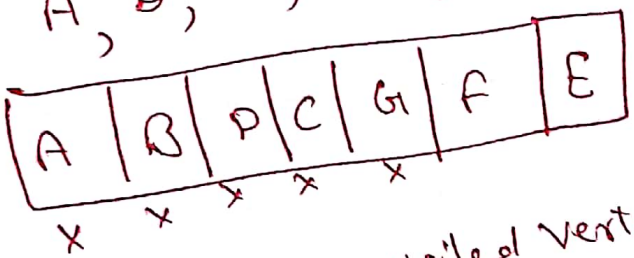
Queue:



Step 5. Remove G and insert its unvisited vertex onto Queue.

Result: A, B, D, C, G

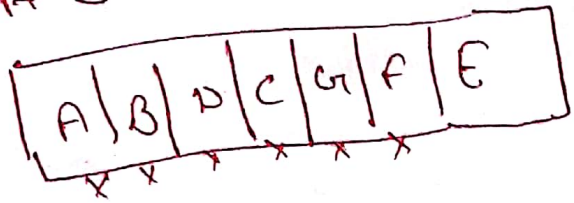
Queue



Step 6. Remove F and Insert its unvisited vertex onto Queue.

Result: A B D C G F

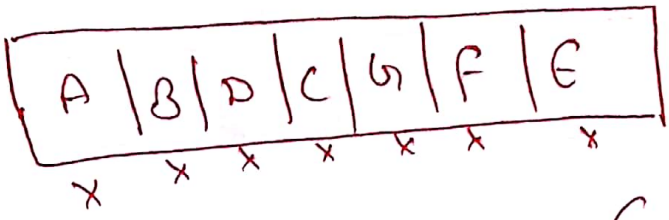
Queue.



Step 7 Remove E from Queue and insert its unvisited vertex onto Queue.

Result : A B D C G F E

Queue:



The BFS traversal of given Graph is

→ A B D C G F E.

There are various valid BFS traversal of given Graph depending on the unvisited adjacent vertex of source vertex (selected vertex) inserted onto Queue.

Assignment 1.

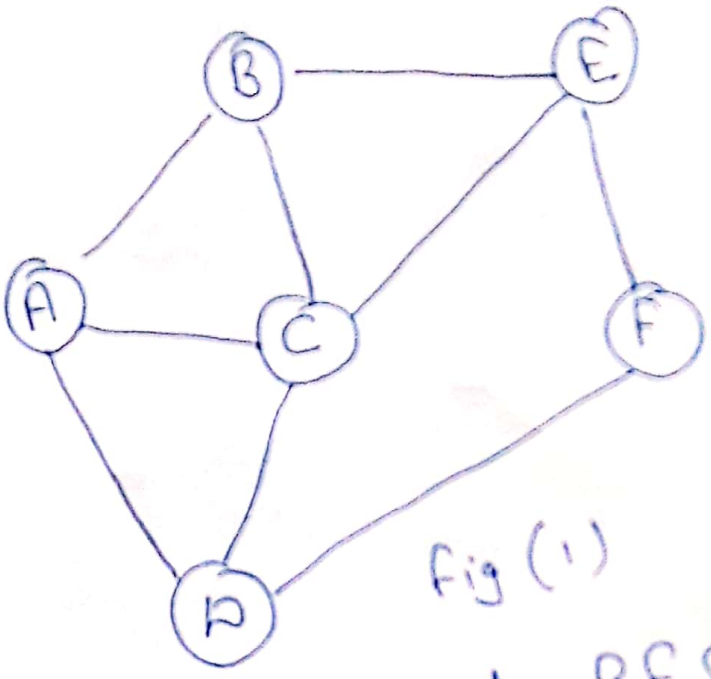


fig (i)

Apply DFS and BFS on graph shown in fig (i) and fig (ii)

Assignment 2.

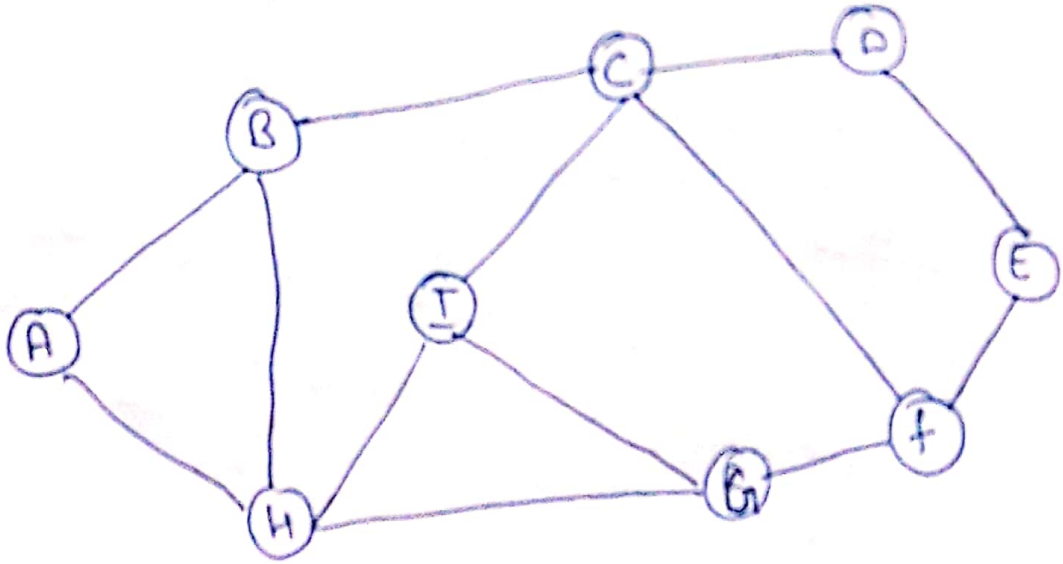


fig (ii)