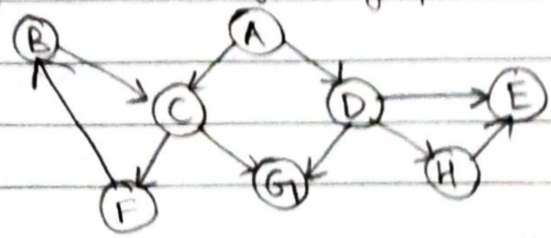
24.03.2020.

1. (i) (a) Write an algorithm for DFS.

(b) Using the algorithm, find the DFS of the graph starting from vertex A.



– (a) Array visited[n] initially set to false, where n is the no. of vertices.

```
dfs( int v)
{
        visited [v] = true;
        print (v)
        for each vertex w adjacent to v, do:
            if not visited [w], then
                dfs(w);

}
```

(b) Visited

A	B	C	D	E	F	G	H
0	0	0	0	0	0	0	0

~~Result : A → C~~

① dfs(A)

visited[A] = true

Result: A

dfs(C)

dfs(D)

② dfs (C)

visited [C] = true

Result A → C

dfs(F)

dfs(G)

③ dfs(F)

visited [F] = true

A → C → F

dfs(B)

④ dfs (B)

Visited [B] = true

Res: A → C → f → B

Adj. node = C → already visited. So pop B. (frame completes).

⑤ Adj. node : B already visited.

∴ Pop F

⑥ adj. node: G.

dfs (G)

visited [G] = true

Result: A → C → F → B → G

no adj node. Pop G.

⑦ no other unvisited adj. node for C.

∴ Pop C.

⑧ dfs(D)

Visited (D) = true

adj: E H.

A → C → f → B → G → D

⑨ dfs(E)

visited(E) = true

A → C → F → B → G → D → E

(10) no neighbors.
∴ frame returns
Pop E

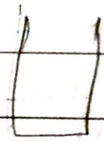
(11) dfs (H)
visited (H) = true

A → C → F → B → G → D → E → H

(12) node adj to H: E.
→ already visited
∴ pop

(13) no unvisited edge next to D. ∴ Pop D

(14) No unvisited edge adj to A. ∴ Pop A



RESULT: A → C → F → B → G → D → E → H.