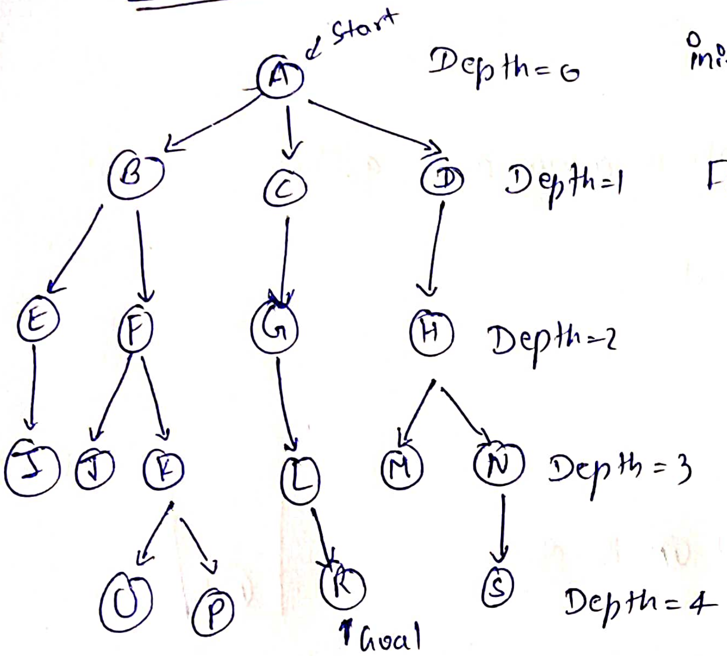
ACTIVITY-3

Aman Kalla
RA1911003010640

Artificial Intelligence

IDDFS



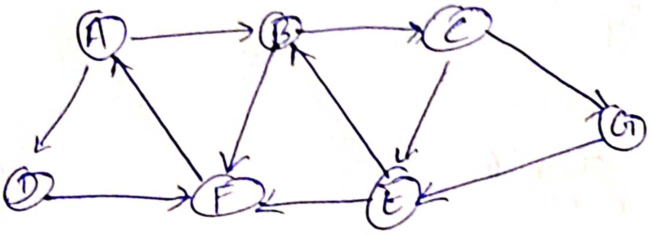
initial State = Ⓐ
or node

Final node / Goal Node = Ⓡ

Depth	Iterative deepning Search.
0	A
1	A B C D
2.	A B E F C G D H
3	A B E I F J K G L H M N
4	A B E I F J K O P C G L Ⓡ

↑ Goal Node occurs at depth ~~5~~ 4

BFS



Make a queue and a Stack

Start Node ⇒ A

put A in queue

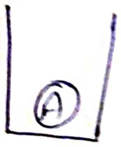
Rear Front

A

adjacent of A ⇒ B and D

put B D in queue.

and pop A and put A in stack



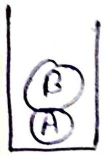
D B

adj of B ⟶ C F ⇒ put in queue

Pop B and put in stack

~~Fron~~ Rear Front

F C D

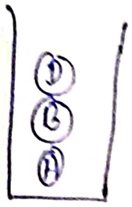


adj of D is F but F is already visited.

Pop D ⇒ put in Stack

Rear Front

F C

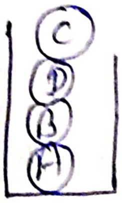


adj. of C ⇒ E and G

Pop C and put in stack

Rear Front.

G E F

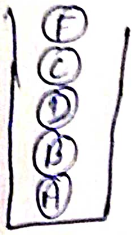


adj. of F ⇒ A but A is already visited

Pop F → Put in Stack

Rear Front.

G E



adj- of E ⇒ F and B

Both are visited

Pop E and put in stack

E
F
C
D
B
A

G

adj- of G ⇒ E but E is already visited

Pop G and put in stack.

G
E
F
C
D
B
A

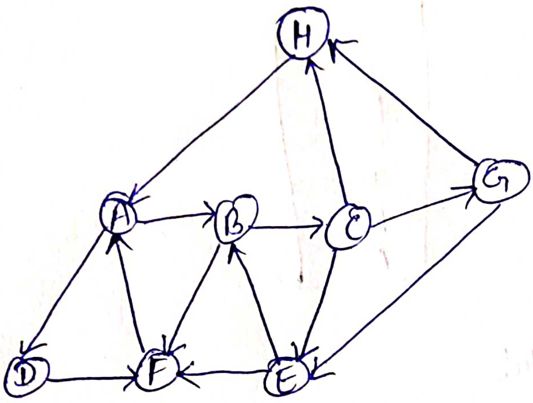
Now queue is empty

Pop element from stack. G E F C D B A

Reverse elements ⇒ A B D C F E G.

BFS Traversal ⇒ A B D C F E G Ans

DFS



Make a vector and a stack.

Start node = A.

push A into stack

push A into vector
Vector ⇒ Ⓐ

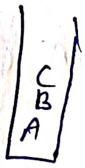
adj of A ⇒ B and D

Take B and push into Stack

Push B into vector
Vector ⇒ Ⓐ Ⓑ

adj of B ⇒ C and F

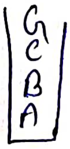
Take C and push into Steck



push C into vector
vector ⇒ Ⓐ Ⓑ Ⓒ

adj. of C ⇒ H, G, E

Take G and push into stack



Push G into vector
Vector ⇒ Ⓐ Ⓑ Ⓒ Ⓖ

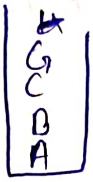
adj of G ⇒ H and E

Take H and push into Stack

Push H into vector
Vector ⇒ Ⓐ Ⓑ Ⓒ Ⓖ Ⓗ

adj. of H → A. But A is already visited
so. H does not have unvisited node.
so pop H from stack.

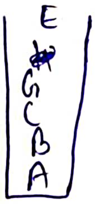
Pop (H) ⇒



Vector⇒ A B C G H

St. top() ⇒ G

Unvisited node of G ⇒ E so push E into stack



Push Ⓔ into vector

Vector ⇒ Ⓐ Ⓑ ~~Ⓖ~~ Ⓒ Ⓖ Ⓗ ~~Ⓔ~~ Ⓔ

Un~~visited nod of E~~ ⇒

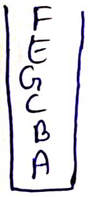
adj. node of E ⇒ Ⓑ Ⓕ

But unvisited node is Ⓕ

so push F into stack

Push Ⓕ into vector

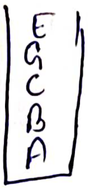
Vector ⇒ Ⓐ Ⓑ Ⓒ Ⓖ Ⓗ Ⓔ Ⓕ



St. top() ⇒ F

adj. node of F ⇒ A but A is visited.

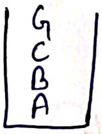
So, Pop(F) ⇒



~~Ve~~

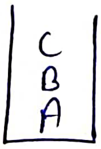
adj node of E ⇒ Ⓑ Ⓕ but both are Visited

so pop (E)



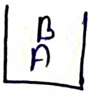
adj node of G ⇒ H and E buth both visited.

So pop G



adj. node of C ⇒ E, G, H but all are visited

so, pop C



St. top() = B

Vector ⇒ A, B C G H E F

adj. node of B ⇒ C, F but both visited

So pop (B)

~~St top()~~ = A

adj ~~node of A ⇒ B, D but both visited~~ ~~visited~~.

~~So, Pop(A)~~

~~Now Stack is~~ empty.

St. top() = A

adj. node of A ⇒ B, D

but D is unvisited

put D in Stack

puch D in vector

Vector A B C G H E F D

adj node of D = F. but F is visited

Pop (D)

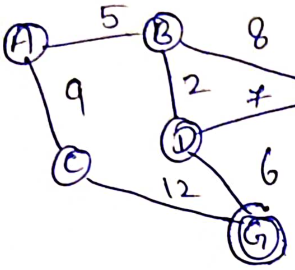
adj node of A ⇒ B, D but both visited.

Pop (A)

Now, Stack empty. So, return vector.

DFS Traversal ⇒ Ⓐ Ⓑ Ⓒ Ⓖ Ⓗ Ⓔ Ⓕ Ⓓ

UCS



Start state = A

Goal State = G

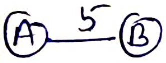
Took the node which has minimum path cost.

Start State A

A → B : 5

A → C : 9

Choose A → B

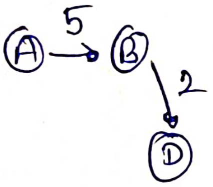


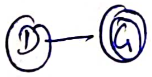
Now,

B → E : 8

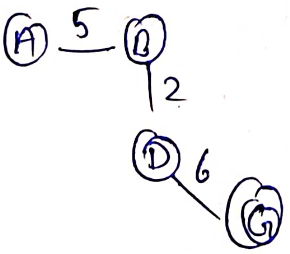
B → D : 2

Choose B → D



Now  G is Goal state

So.



Path cost ⇒ 5 + 2 + 6

= 13 Ans.