Depth First Search

DFS stands for Depth First Search is a type of uninformed state-space searching algorithm and a general graph-searching algorithm.

Q) Why it is uniformed search algorithm?

Ans: This search algorithm does not know when it will reach its goal state. It is also known as unguided search or blind search. It works on basis of Brute Force technique.

Brute Force Technique: It checks each elements along its way to get the solution.

There are two type of states obtained 1) non-final state and 2) final state.

In Non-final state when goal state is not reached but at each state it obtain a new state and proceed to get the final goal state.

In Final State when goal state is reached.

Some terminologies:

1. Start State: This is also known as goal-driven agent . This is the state in which the program starts to find the solution. It is the initial condition of the agent.

2. Goal state: This the output of the search.

3. **State space:** **This refers to all the states that can be reached from the start state by a series of actions. It can also be thought of as all the states in which the agent can exits.**

4. **Action space:** **This refers to the list of all possible actions that can be executed by the agent. More specifically, it tells us about all the actions that can be executed in a particular state.**

Note: "Each state can move to another state by transition, how that transition occurs is called action."

5. **Transition model:** **This gives us a description of the outcomes of the actions performed in a state.**

6. **Goal Test:** **This is the method to check whether the current state is goal state or not.**

7. **Path cost: This is a function that is used in assigning vales to a path, which can be thought of as cost with respect to performance.**

8. **Path: The route along which we can reach goal state.**

Q) Why Depth First Search algorithm is called Graph Searching Algorithm?

Ans: In graph searching technique, also known as traversal technique visits every node exactly one in a systematic fashion. Same technique used in DFS or Depth first search.

Q) HOW DEPTH FIRST SEARCH ALGORITHM WORKS?

PROCEDURE:

1. IT EXPANDS THE DEEPEST NODE IN THE CURRENT FRONTIER OF THE SEARCH TREE.

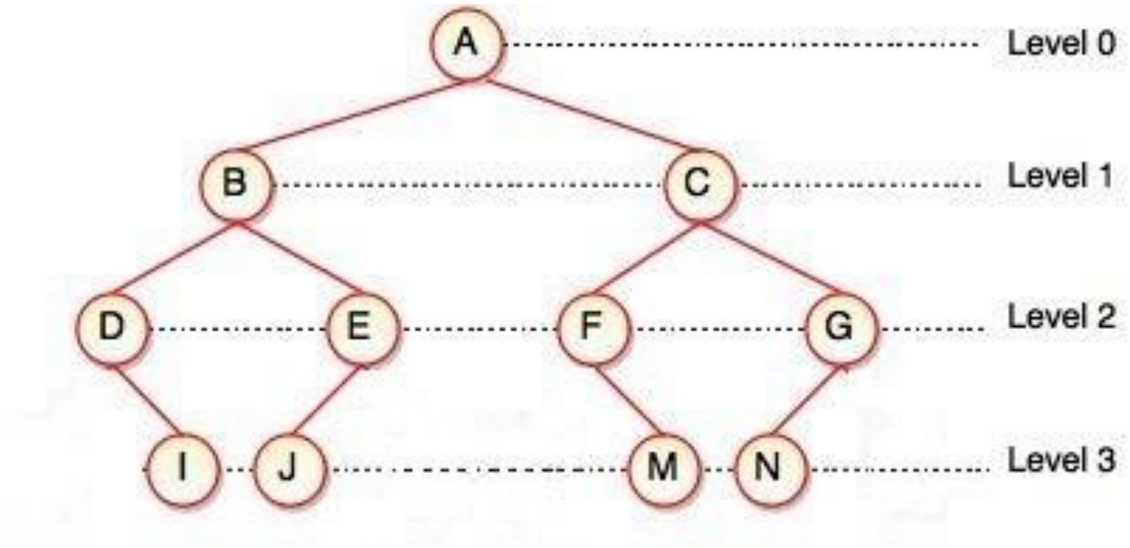
2. THE SEARCH PROCEEDS IMMEDIATELY TO THE DEEPEST LEVEL OF THE SEARCH TREE, WHERE NODES HAVE NO SUCCESSOR.

3. DEPTH FIRST USES A LIFO (LAST IN FIRST OUT) OR FILO(FIRST IN LAST OUT)STACK DATA STRUCTURE USING PUSH AND POP METHOD.

THAT IS THE SEARCH START FROM A STARTING NODE , IF THE GOAL STATE IS NOT OBTAINED ,THEN IT CHOOSE A DEEPER NODE OF THE STARTING NODE TO SEARCH AND THEN IT GO AND SEARCH DEEPER NODES OF THAT NODE TILL IT REACH THE LEAF WHICH IS THE DEEPEST NODE AND IF THE GOAL STATE IS NOT OBTAINED THEN AGAIN IT COME BACK TO THE EARLIEST NODE AND CONTINUE THE SEARCH THROUGH SAME PROCESS.

COMING BACK TO THE EARLIEST NODE IS CALLED BACK TRACKING.

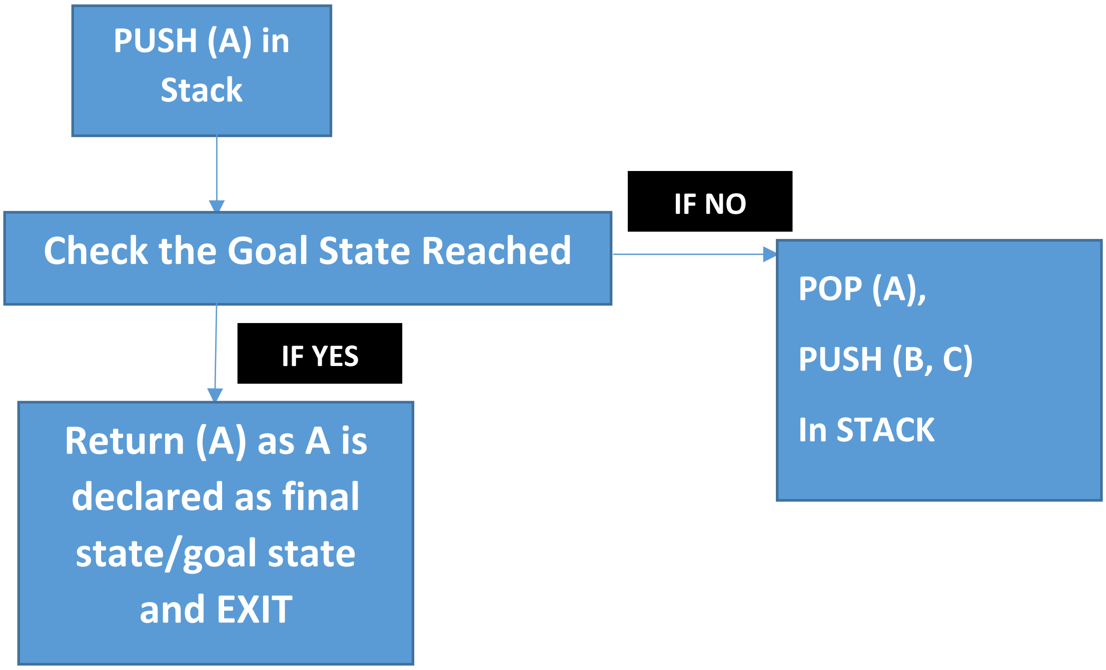
Lets say we have a tree:



1st in Stack we will push 'A' inside:

A

NOW,



PUSH (A) in Stack

Check the Goal State Reached

IF NO

POP (A),

PUSH (B, C)

In STACK

IF YES

Return (A) as A is declared as final state/goal state and EXIT

AS B,C ARE SUCCESSORS OF A.

NOTE B,C CAN BE PUSHED IN STACK IN ANY ORDER SUCH AS B,C

C
B

OR C,B

B
C

I.E. 2_{p_2} **= 2! WAYS.**

"IMPORTANT : ***NOW POP FUNCTION WILL DEPEND UPON PUSH FUNCTION ON STACK,***

IF

C
B

THEN 1ST POP WILL BE C, HENCE TRAVERSAL WILL TAKE PLACE FROM A TO C, TO POP C(LIFO- LAST IN FIRST OUT OR FILO- FIRST IN LAST OUT).

OR IF

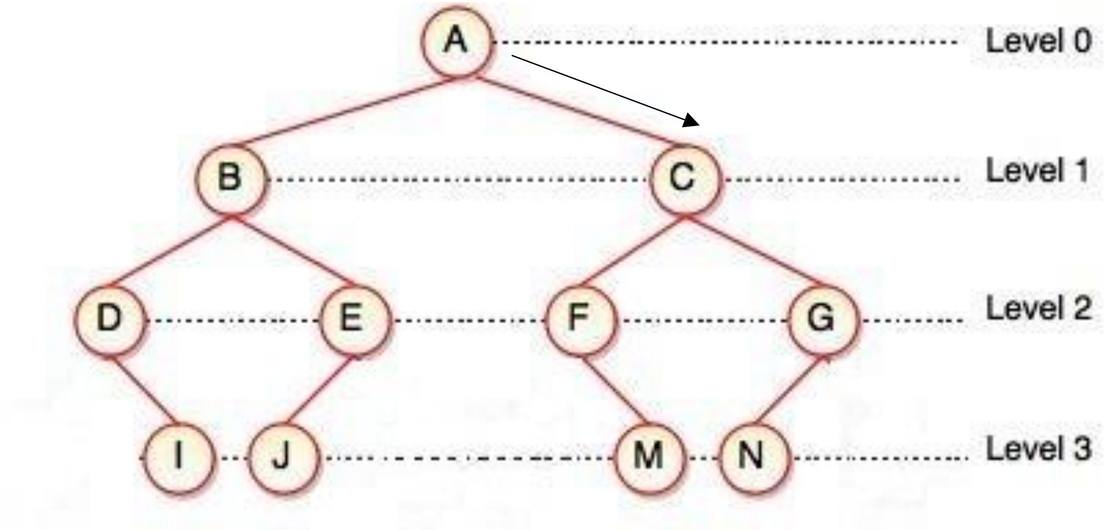
B
C

THEN 1ST POP WILL BE B (LIFO), HENCE TRAVERSAL WILL TAKE PLACE FROM A TO B, TO POP B(LIFO- LAST IN FIRST OUT OR FILO- FIRST IN LAST OUT).”

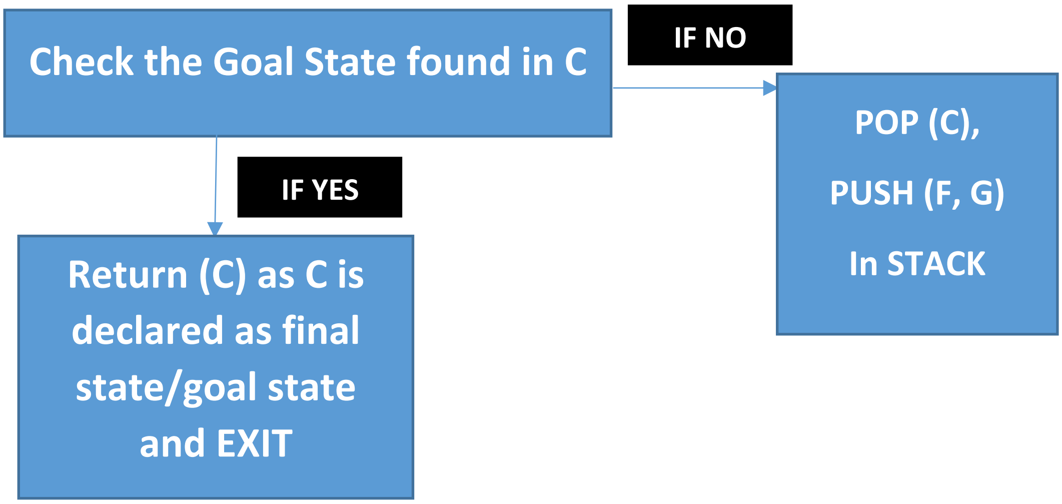
CONSIDERING B INSERTED FIRST THEN C ,WE HAVE IN STACK:

C
B

NOW TRAVERSAL OCCUR FROM A TO C, C BEING DEEPER NODE AND FOLLOWING THE PATH.



NOW C WILL BE CHECKED FOR GOAL STATE:


Check the Goal State found in C

IF NO

POP (C),

PUSH (F, G)

In STACK

IF YES

Return (C) as C is declared as final state/goal state and EXIT

NOW TRAVERSAL MAY OCCUR FROM C TO G OR C TO F.

NOTE IF THE STACK IS:

G
F
B

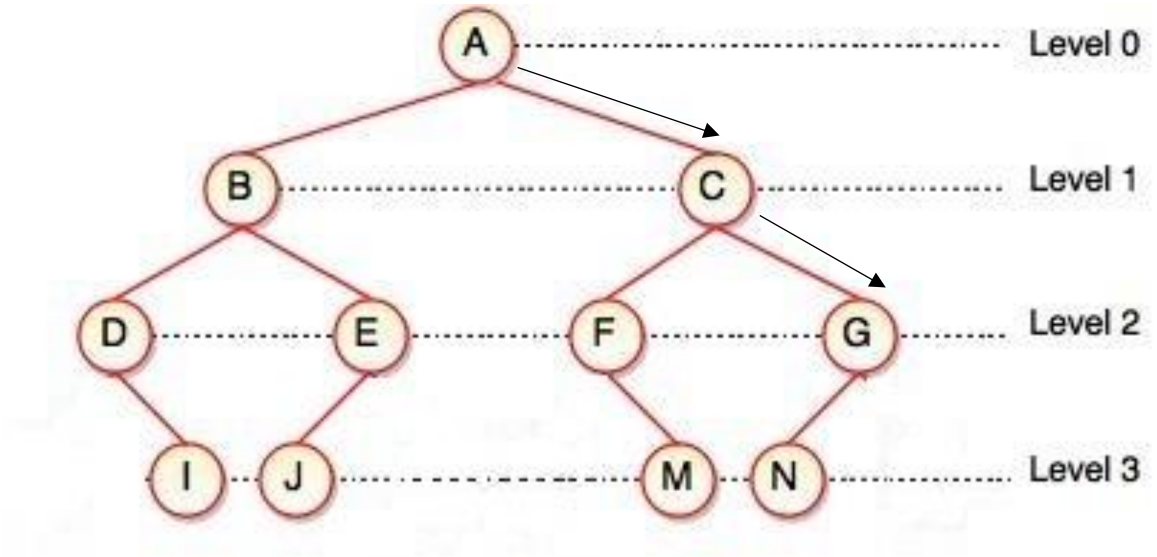
THEN 1ST POP WILL BE G, HENCE TRAVERSAL WILL TAKE PLACE FROM C TO G, TO POP G(LIFO- LAST IN FIRST OUT OR FILO- FIRST IN LAST OUT).

OR IF

F
G
B

THEN 1ST POP WILL BE F, HENCE TRAVERSAL WILL TAKE PLACE FROM C TO F, TO POP F(LIFO- LAST IN FIRST OUT OR FILO- FIRST IN LAST OUT).

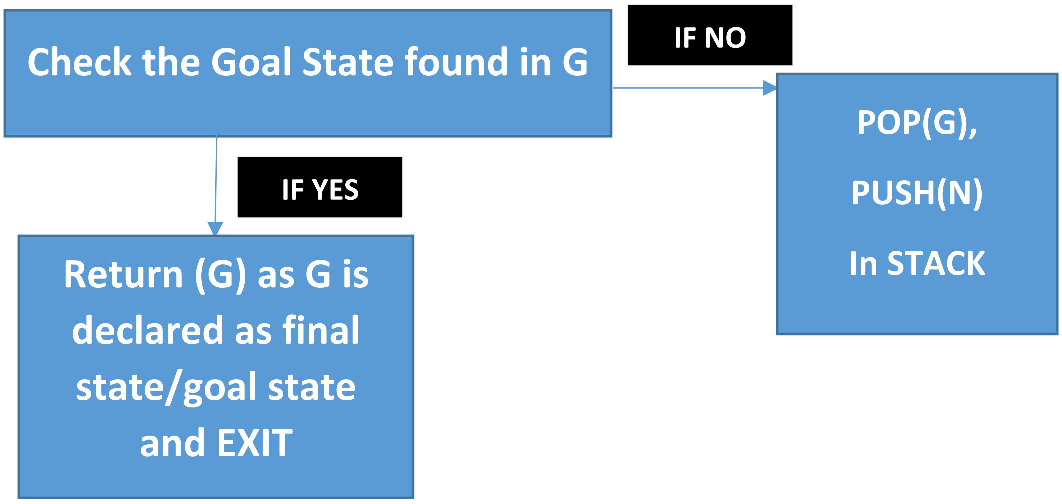
CONSIDERING THAT TRAVERSAL HAD TAKEN PLACE FROM C TO G(DEEPER NODE) AND FOLLOWING THE PATH.



NOW G WILL BE CHECKED FOR GOAL STATE:

NOW STACK WE HAVE, IF GOAL STATE NOT FOUND IN C:

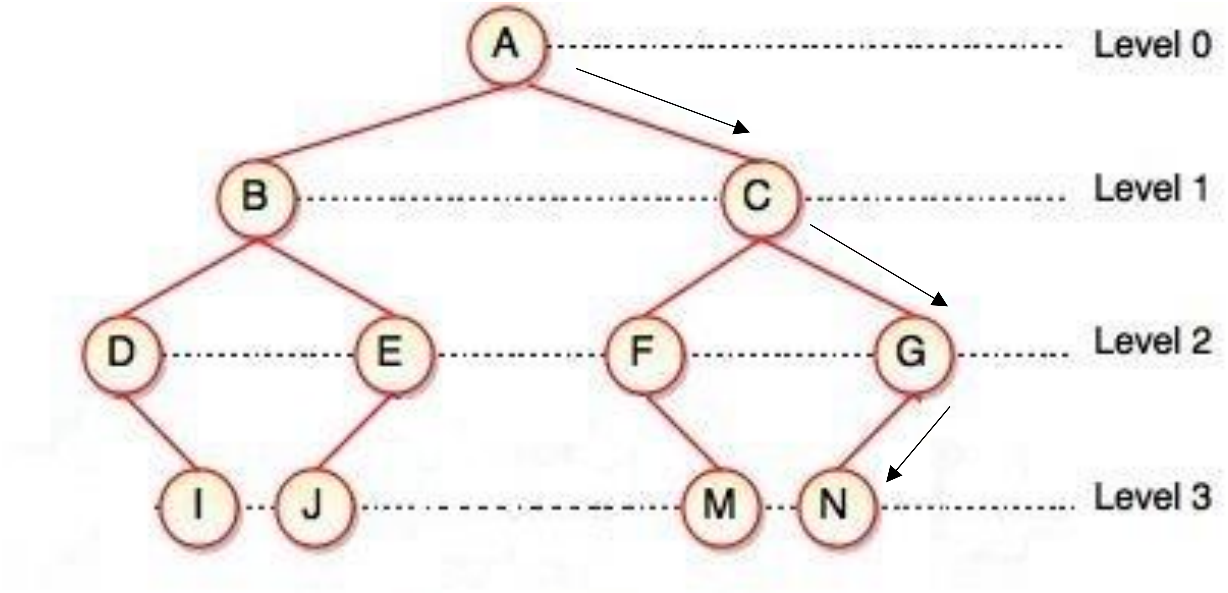
G
F
B

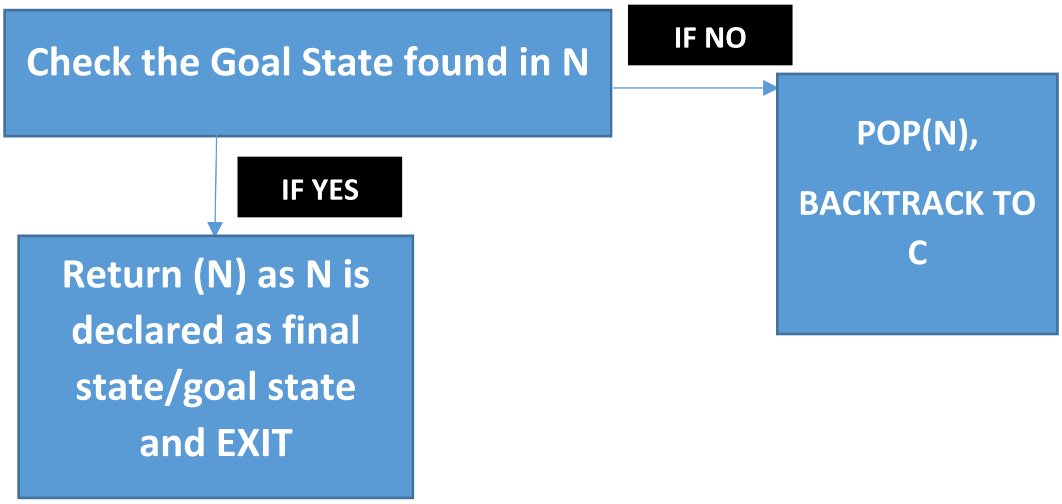

Check the Goal State found in G
IF NO
POP(G),
PUSH(N)
In STACK
IF YES
Return (G) as G is declared as final state/goal state and EXIT

STACK:

N
F
B

NOW N WILL BE CHECKED FOR GOAL STATE:




Check the Goal State found in N
IF NO
POP(N),
BACKTRACK TO C
IF YES
Return (N) as N is declared as final state/goal state and EXIT

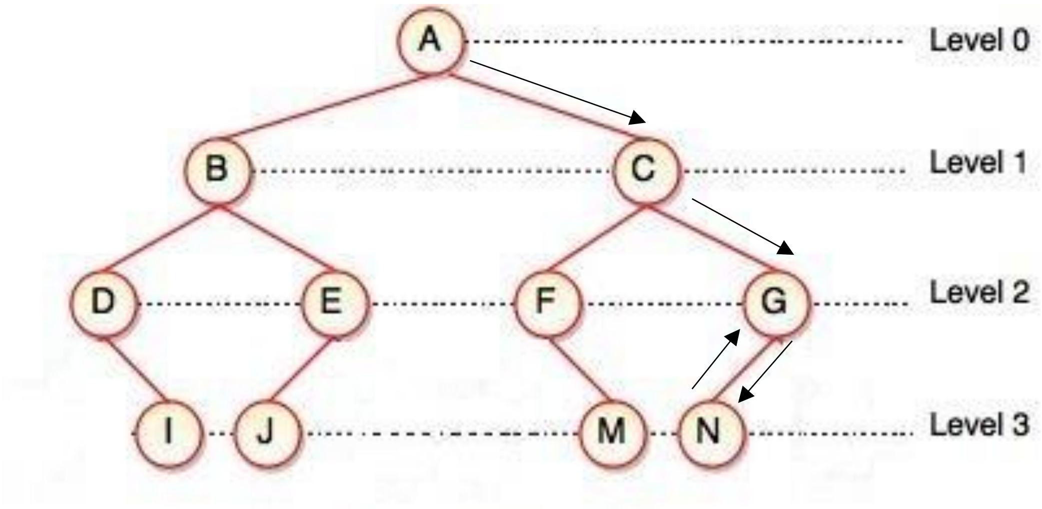
NOTE N IS THE DEEPEST NODE.

NOW STACK WE HAVE, IF GOAL STATE NOT FOUND IN N:

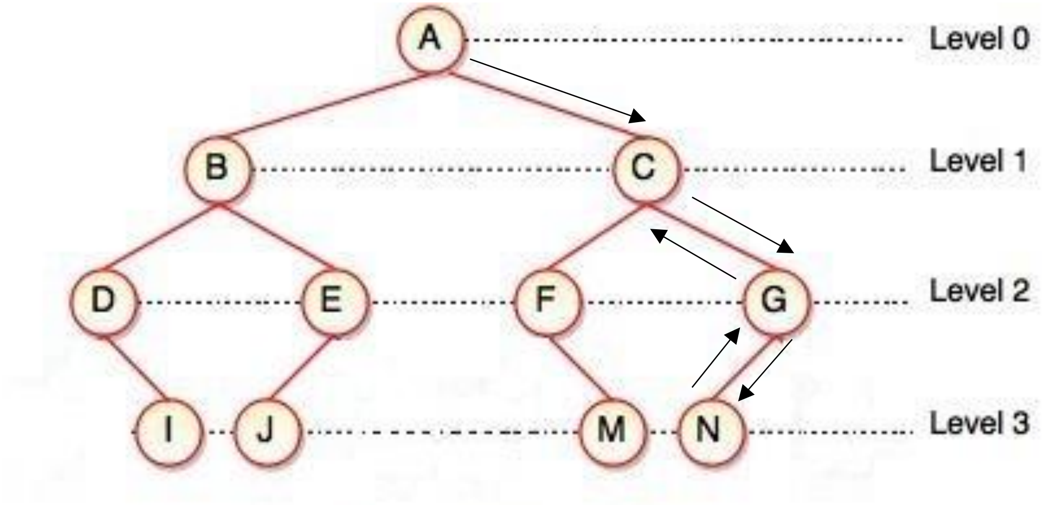
F
B

NOW WE WILL BACKTRACK TO C TO CHECK GOAL STATE IN F.

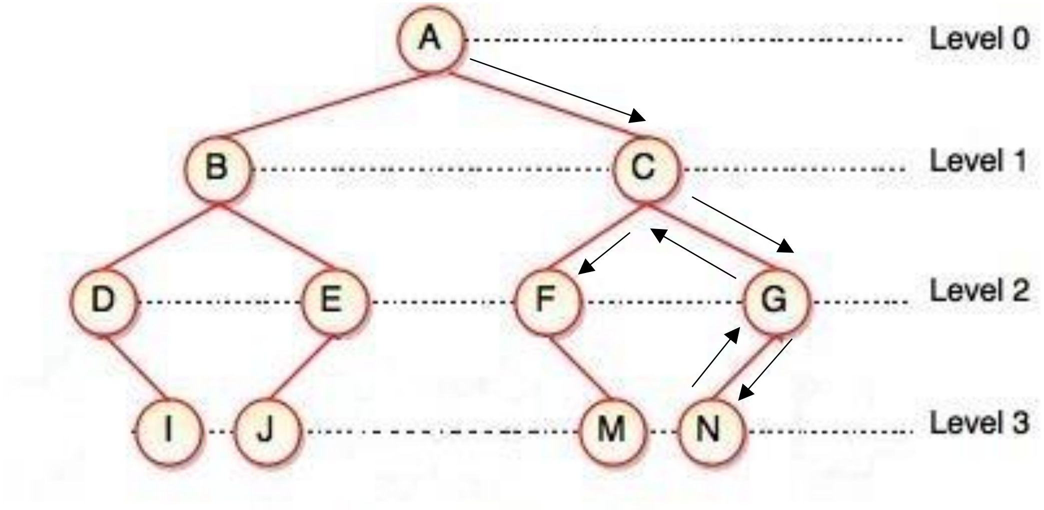
BACKTRACK 1:

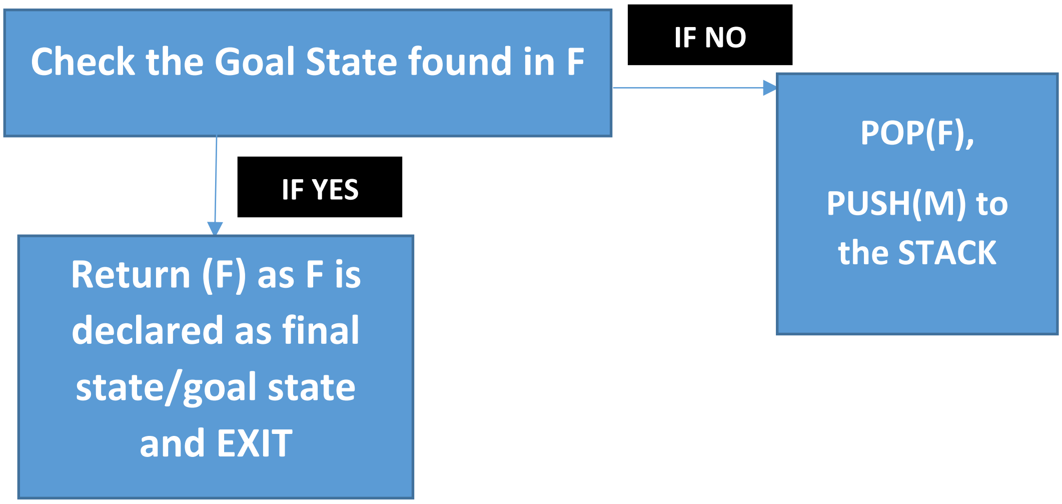


BACKTRACK 2:



NOW F WILL BE CHECKED FOR GOAL STATE:





Check the Goal State found in F

IF NO

POP(F),

PUSH(M) to the STACK

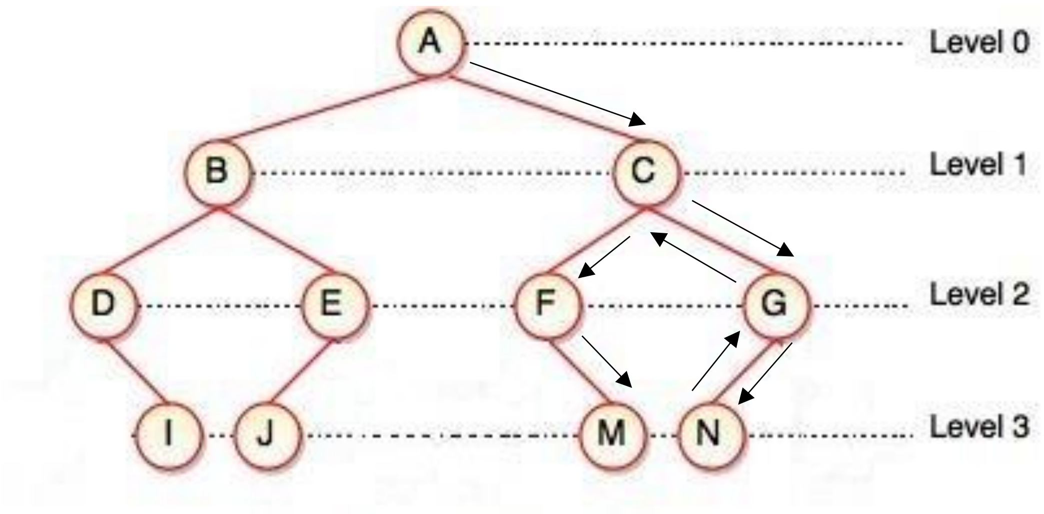
IF YES

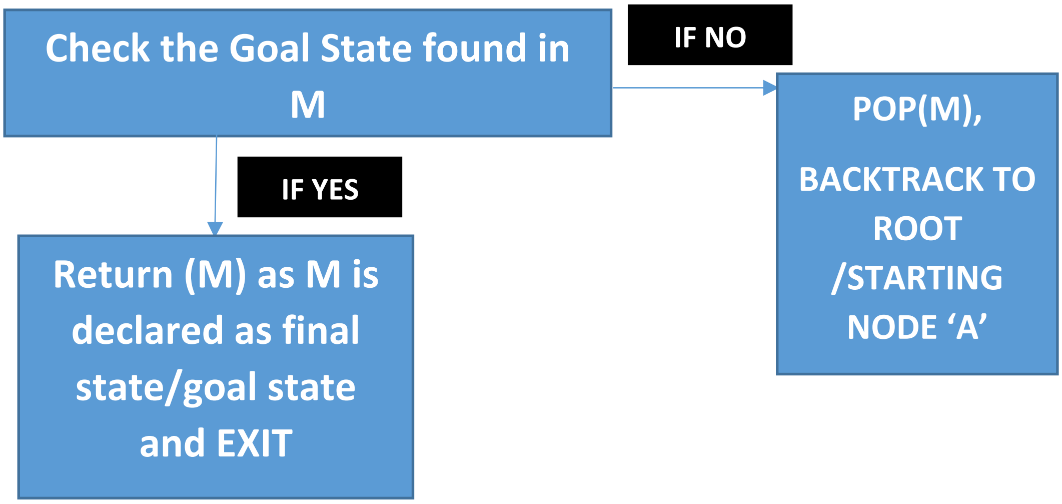
Return (F) as F is declared as final state/goal state and EXIT

NOW STACK WE HAVE, IF GOAL STATE NOT FOUND IN F:

M
B

NOW M WILL BE CHECKED FOR GOAL STATE, M BEING THE DEEPEST NODE:





Check the Goal State found in M

IF NO

POP(M),

BACKTRACK TO ROOT /STARTING NODE 'A'

IF YES

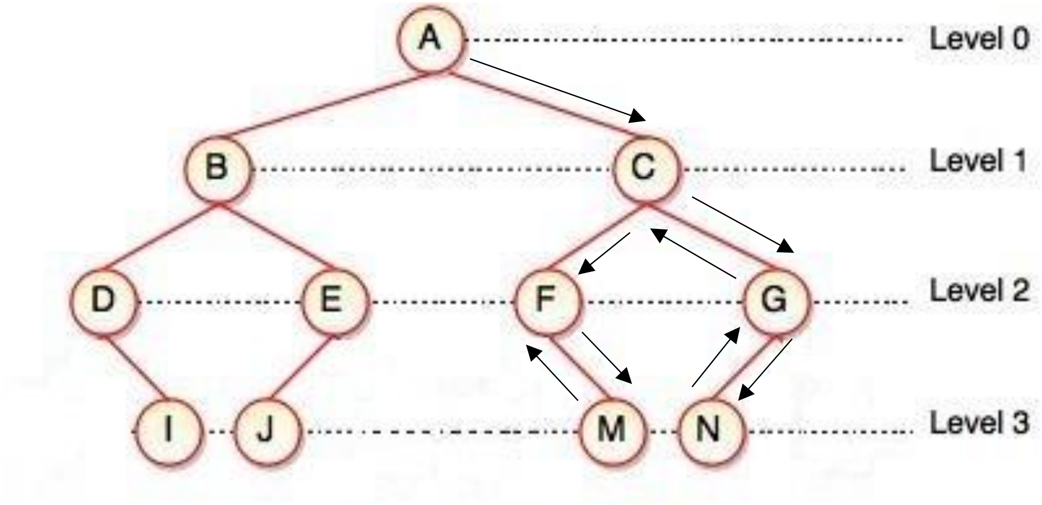
Return (M) as M is declared as final state/goal state and EXIT

NOW STACK WE HAVE, IF GOAL STATE NOT FOUND IN M:

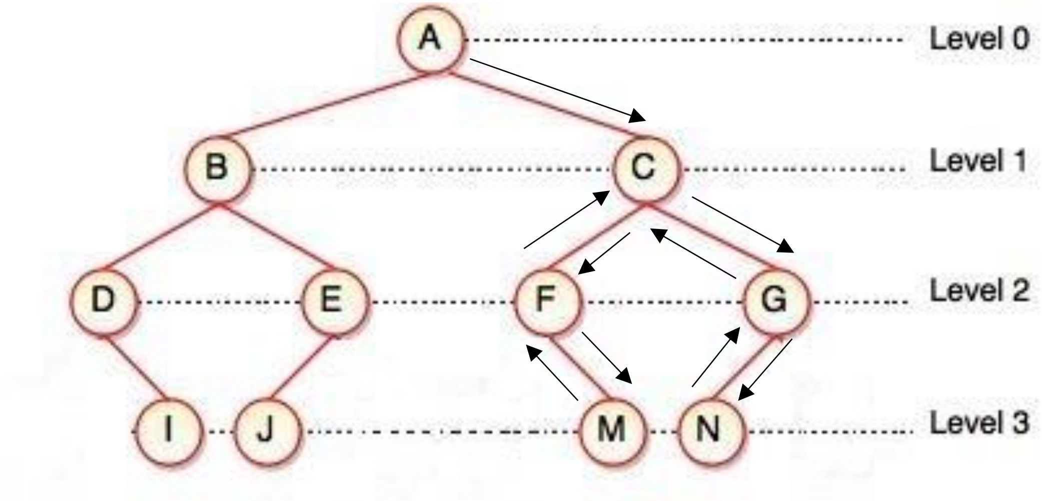
B

NOW WE WILL BACKTRACK TO A TO CHECK GOAL STATE IN B.

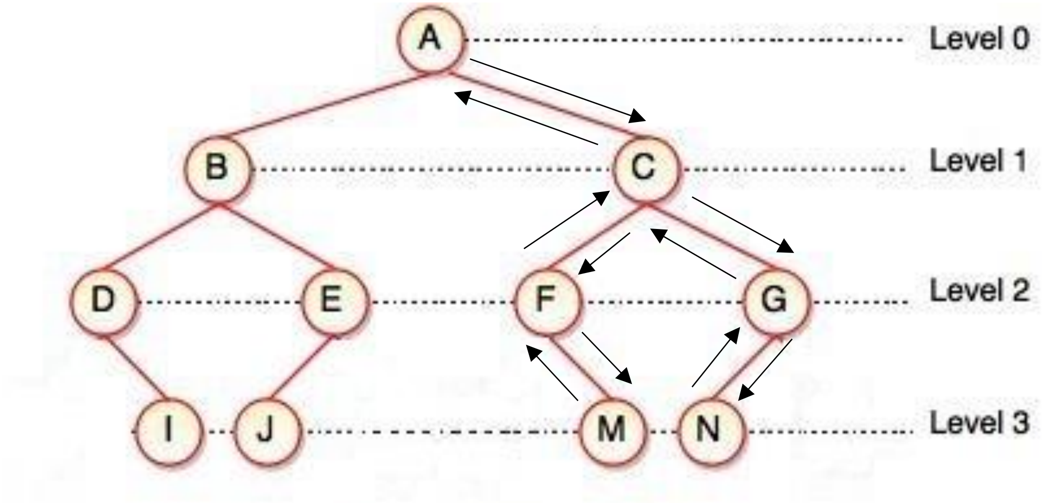
BACKTRACK 1:



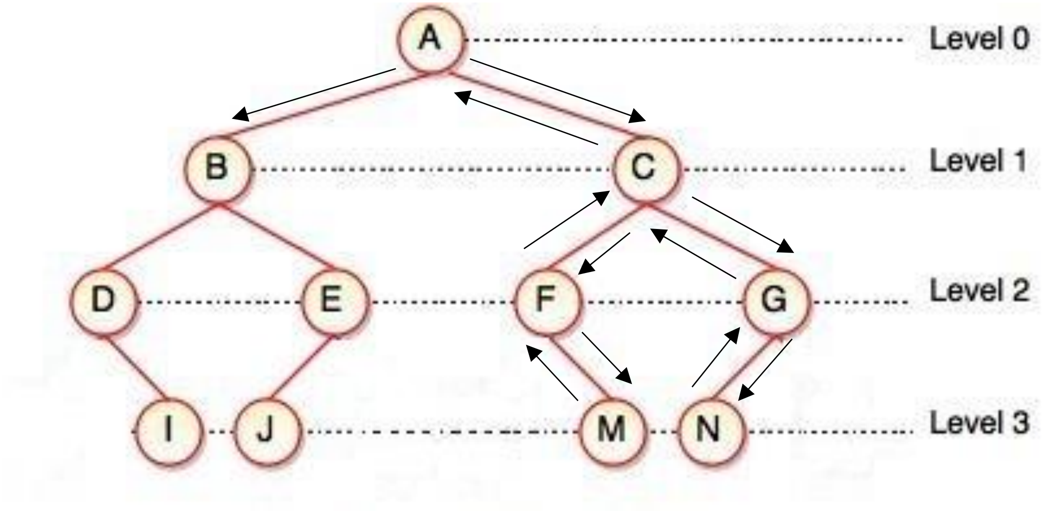
BACKTRACK 2:

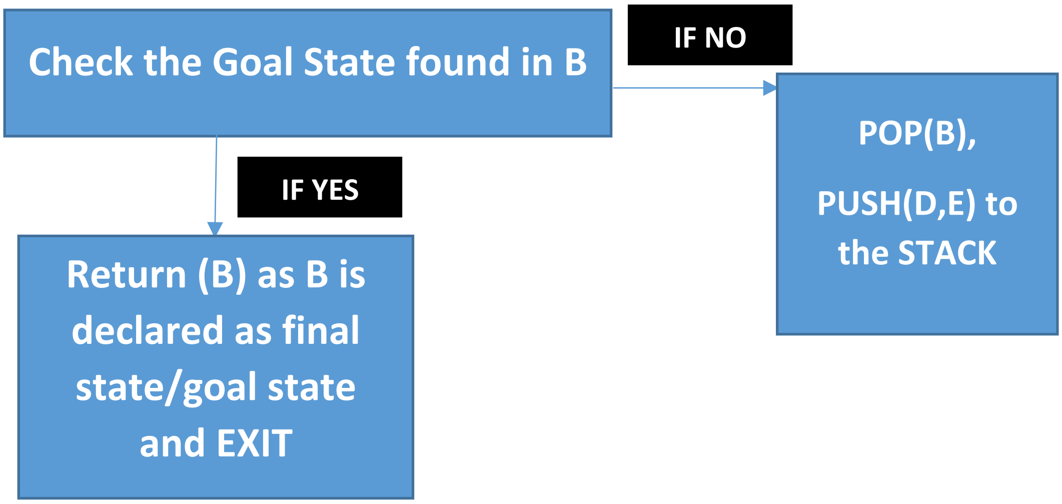


BACKTRACK 3:



NOW B WILL BE CHECKED FOR GOAL STATE:




Check the Goal State found in B

IF NO

POP(B),

PUSH(D,E) to the STACK

IF YES

Return (B) as B is declared as final state/goal state and EXIT

NOW STACK WE HAVE, IF GOAL STATE NOT FOUND IN B:

:

D
E

AS E,D ARE SUCCESSORS OF B.

NOTE E,D CAN BE PUSHED IN STACK IN ANY ORDER SUCH AS E,D

D
E

OR D,E

E
D

I.E. 2_{p_2} = 2! WAYS.

NOW TRAVERSAL MAY OCCUR FROM B TO D OR B TO E.

"IMPORTANT : *NOW POP FUNCTION WILL DEPEND UPON PUSH FUNCTION ON STACK,*

IF

D
E

THEN 1ST POP WILL BE D HENCE TRAVERSAL WILL TAKE PLACE FROM B TO D, TO POP D(LIFO- LAST IN FIRST OUT OR FILO- FIRST IN LAST OUT).

OR IF

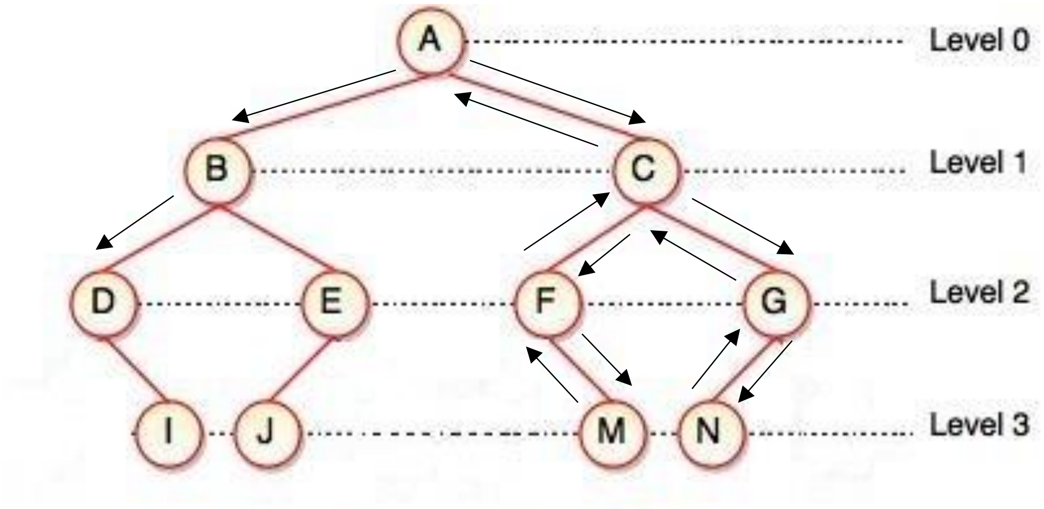
E
D

THEN 1ST POP WILL BE E HENCE TRAVERSAL WILL TAKE PLACE FROM B TO E, TO POP E (LIFO- LAST IN FIRST OUT OR FILO- FIRST IN LAST OUT)."

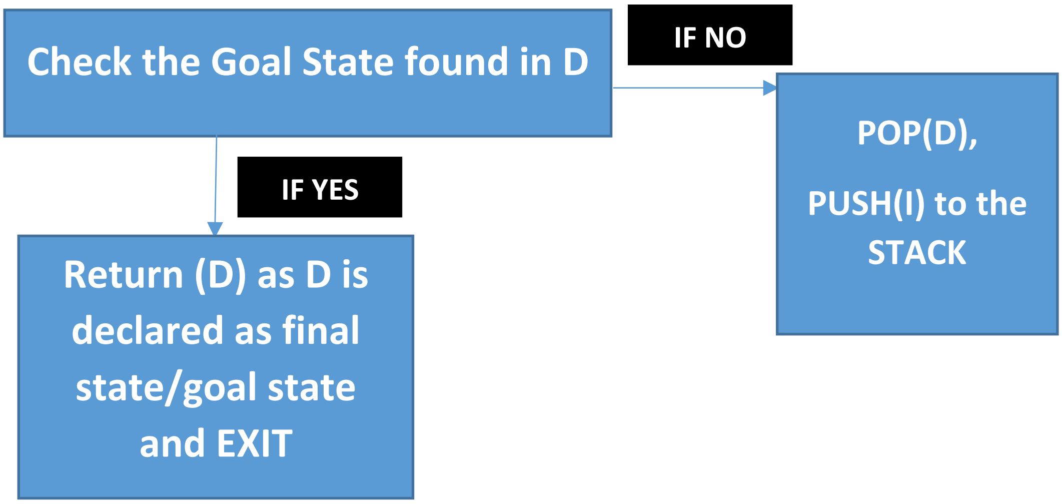
CONSIDERING E INSERTED FIRST THEN D ,WE HAVE IN STACK:

D
E

THEN TRAVERSAL HAD TAKEN PLACE FROM B TO D, D BEING THE DEEPER NODE AND FOLLOWING THE PATH.



NOW D WILL BE CHECKED FOR GOAL STATE:



Check the Goal State found in D

IF NO

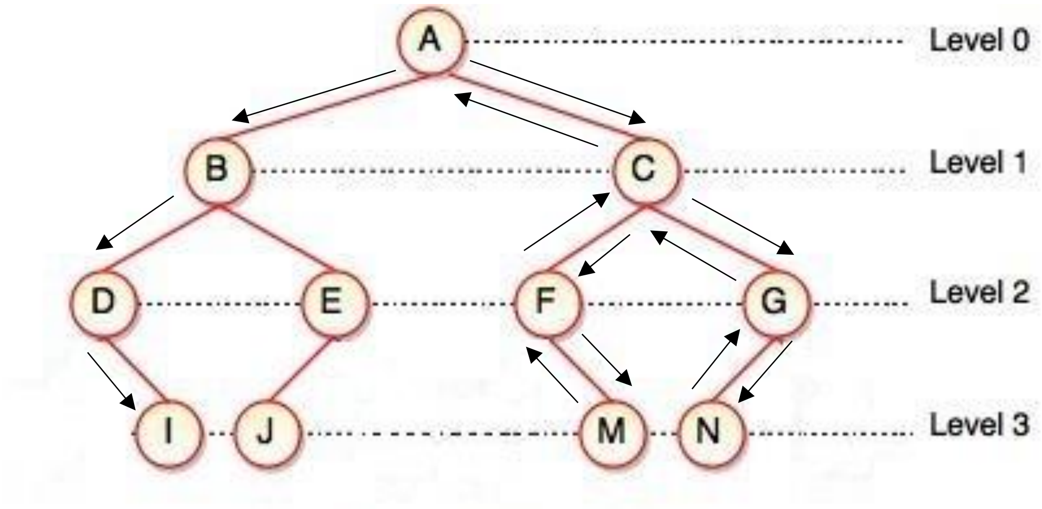
POP(D),

PUSH(I) to the STACK

IF YES

Return (D) as D is declared as final state/goal state and EXIT

AS I BEING THE SUCCESSOR OF D.

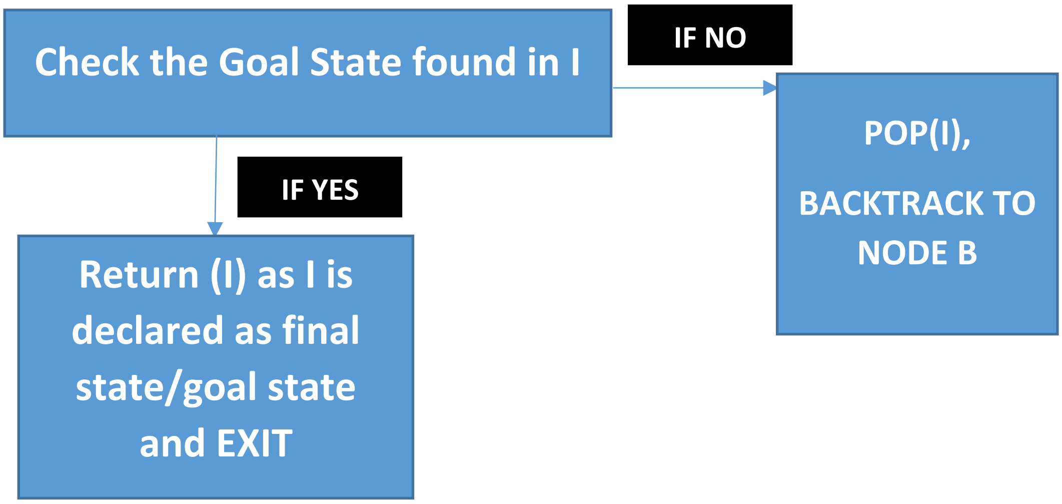


NOW STACK WE HAVE, IF GOAL STATE NOT FOUND IN D:

:

I
E

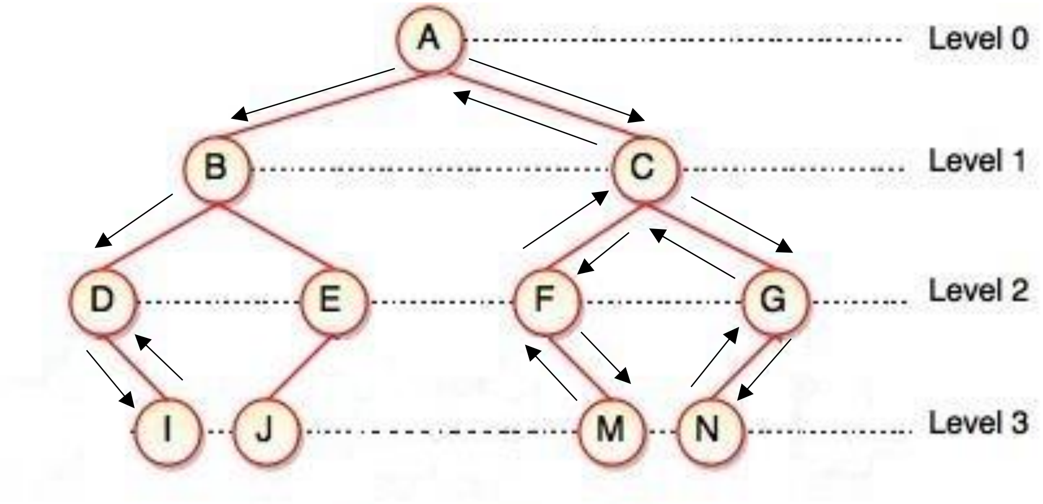
NOW I WILL BE CHECKED FOR GOAL STATE, I BEING THE DEEPEST NODE



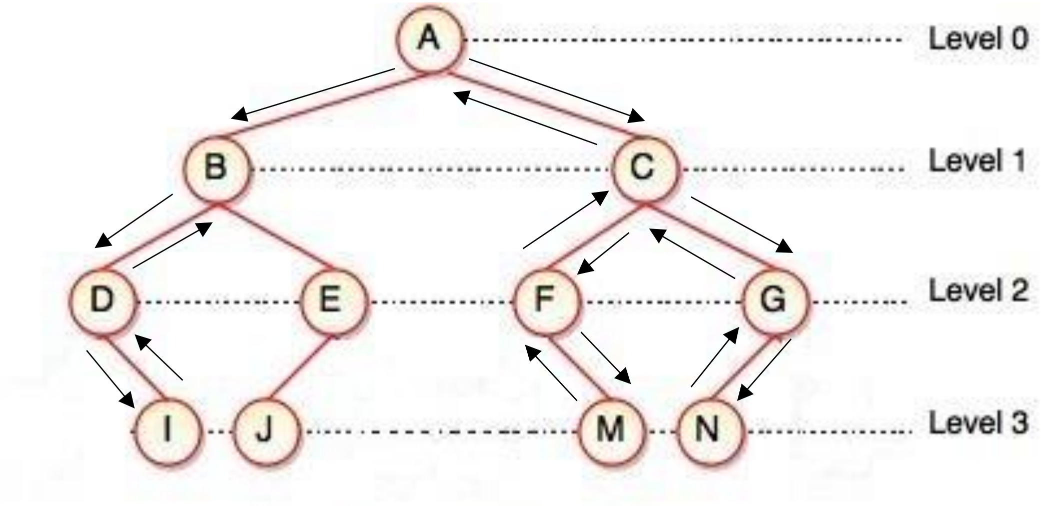
Check the Goal State found in I
IF NO
POP(I),
BACKTRACK TO NODE B
IF YES
Return (I) as I is declared as final state/goal state and EXIT

IF I IS NOT THE GOAL STATE BACKTRACK TO NODE B.

BACKTRACK 1:



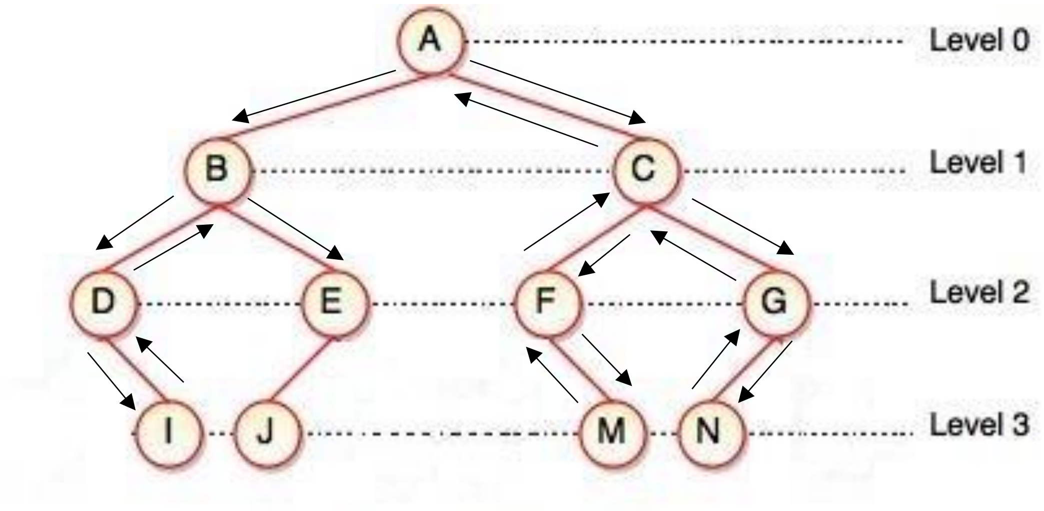
BACKTRACK 2:



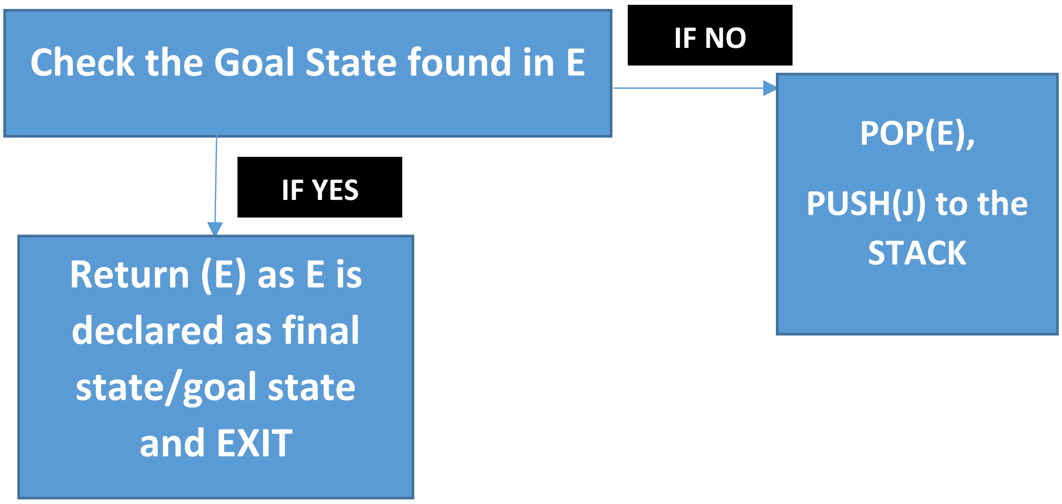
NOW IN STACK WE HAVE:

E

SO WE WILL TRAVERSE FROM B TO E NOW TO CHECK E:



NOW E WILL BE CHECKED FOR GOAL STATE:


Check the Goal State found in E

IF NO

POP(E),

PUSH(J) to the STACK

IF YES

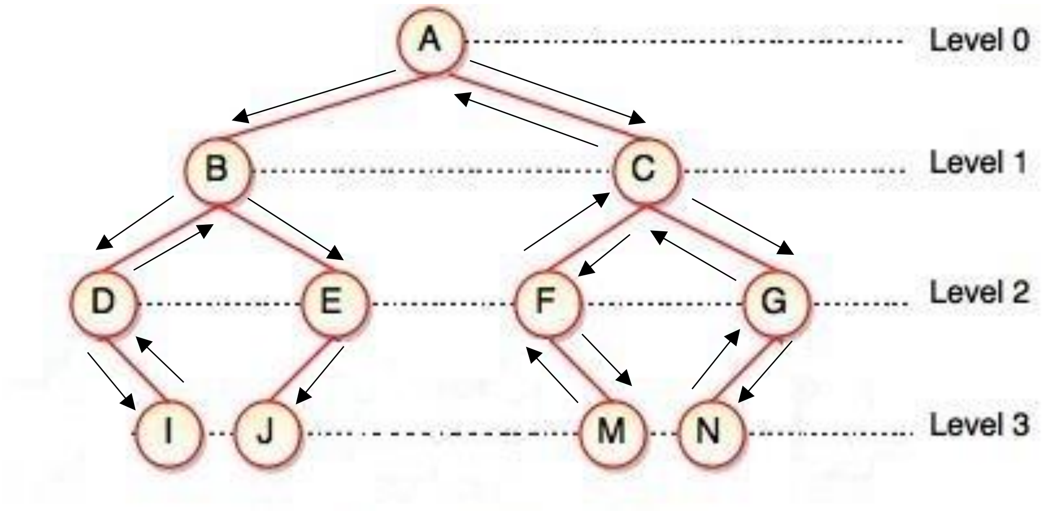
Return (E) as E is declared as final state/goal state and EXIT

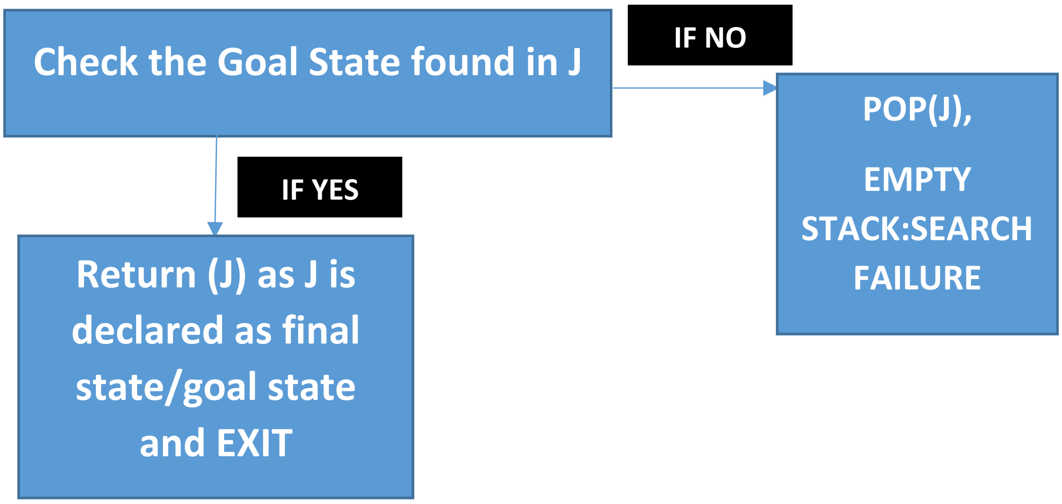
NOW J WILL BE CHECKED FOR GOAL STATE IF E WAS NOT THE GOAL STATE, J BEING THE DEEPEST NODE.

NOW IN STACK WE HAVE:

J

SO WE WILL TRAVERSE FROM E TO J NOW TO CHECK J:





Check the Goal State found in J

IF NO

POP(J),

EMPTY STACK:SEARCH FAILURE

IF YES

Return (J) as J is declared as final state/goal state and EXIT

THEREFORE, IF J IS THE GOAL STATE IT WILL RETURN J OTHERWISE IT WILL RETURN AN EMPTY STACK WILL BE THE FAILURE OF THE SEARCH.

ALGORITHM FOR DEPTH FIRST SEARCH:

1. IF THE INITIAL STATE IS A GOAL STATE , QUIT AND RETURN SUCCESS,

2. OTHERWISE DO THE FOLLOWING UNTIL, SUCESS OR FAILURE IS SIGNALED:

   A. GENERATE SUCCESSOR E OF THE INITIAL STATE, IF THERE ARE NO MORE SUCCESSORS, SIGNAL FAILURE.

   B. CALL DEPTH-FIRST SEARCH WITH E AS THE INITIAL STATE.

   C. IF SUCCESS IS RETURNED , SIGNAL SUCCESS. OTHERWISE CONTINUE IN THIS LOOP.

ALTERNATIVELY,

- **STEP 1:** SET STATUS = 1 (READY STATE) FOR EACH NODE IN G
- **STEP 2:** PUSH THE STARTING NODE A ON THE STACK AND SET ITS STATUS = 2 (WAITING STATE)
- **STEP 3:** REPEAT STEPS 4 AND 5 UNTIL STACK IS EMPTY
- **STEP 4:** POP THE TOP NODE N. PROCESS IT AND SET ITS STATUS = 3 (PROCESSED STATE)
- **STEP 5:** PUSH ON THE STACK ALL THE NEIGHBOURS OF N THAT ARE IN THE READY STATE (WHOSE STATUS=1) AND SET THEIR STATUS=2(WAITINGSTATE)
  [END OF LOOP]
- **STEP 6:** EXIT

ADVANTAGE:

- DEPTH FIRST SEARCH REQUIRES LESS MEMORY SINCE ONLY THE NODES ON THE CURRENT PATH ARE STORED. THIS CONTRAST WITH BREADTH FIRST SEARCH, WHERE THE ENTIRE TREE THAT HAS SO FAR BEEN GENERATED MUST BE STORED.

- IT IS VERY RARE BUT STILL DEPTH FIRST SEARCH MAY FIND A SOLUTION WITHOUT EXAMINING MUCH OF THE SEARCH SPACE AT ALL. THIS CONTRASTS WITH BREADTH-FIRST SEARCH IN WHICH ALL PARTS OF THE TREE MUST BE EXAMINED TO LEVEL 'N' BEFORE ANY NODES ON LEVEL N+1 CAN BE EXAMINED.

  THIS IS PARTICULARLY SIGNIFICANT IF MANY ACCEPTABLE SOLUTIONS EXISTS. DEPTH FIRST SEARCH CAN STOP WHEN

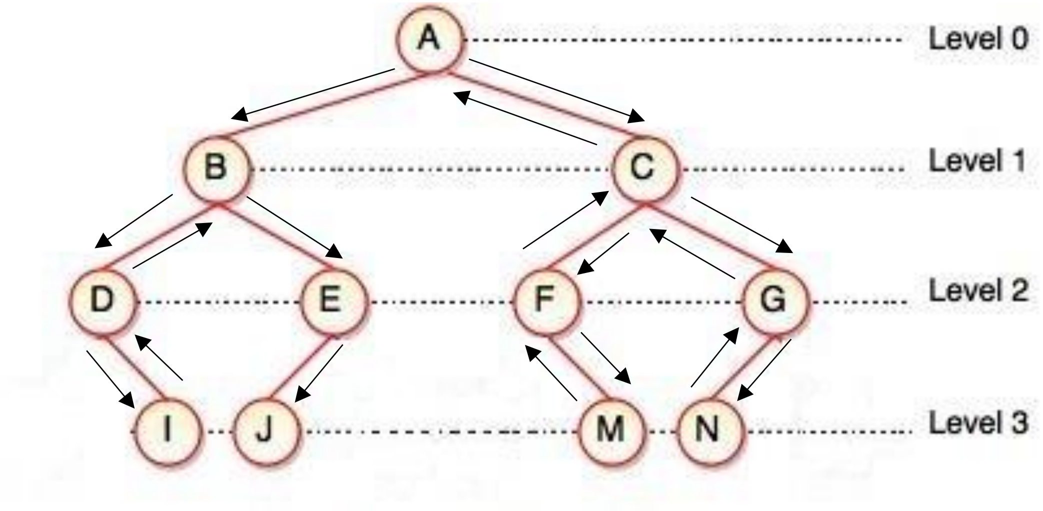
DISADVANTAGE:

- IF SEARCH STATE SPACE IS INFINITE, IT WILL MOVE IN ONE DIRECTION ONLY UNTIL IT GET THE DEEPEST NODE. ALSO IF IT GETS A CYCLE IN THE GRAPH THEN THERE IS A CHANCE THE ALGORITHM TRAP IN A LOOP , HENCE IT CAN GIVE AN INCOMPLETE SEARCHING .

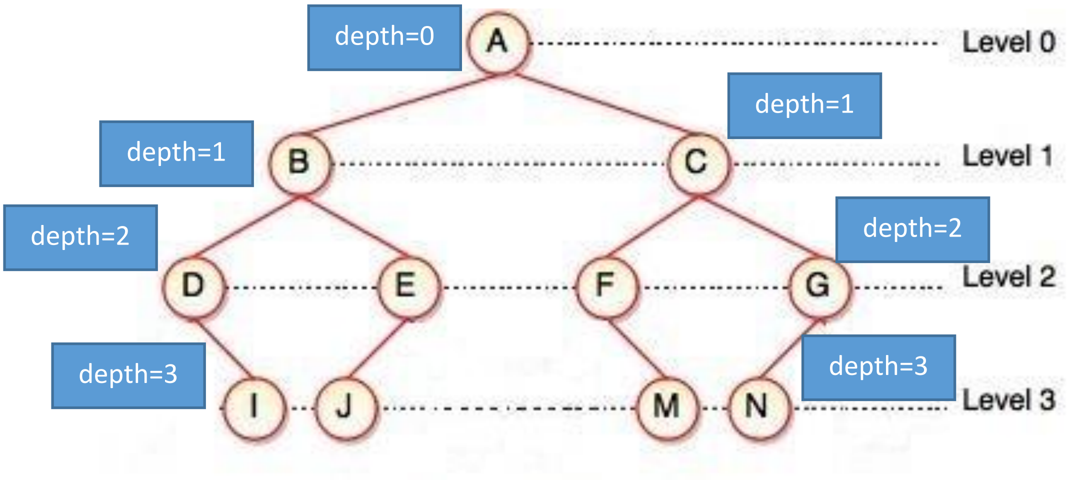
- IT CAN PROVIDE A NON-OPTIMAL OR UNDESIRED, NON-SATISFACTORY SOLUTION AS OUTPUT.

TIME COMPLEXITY

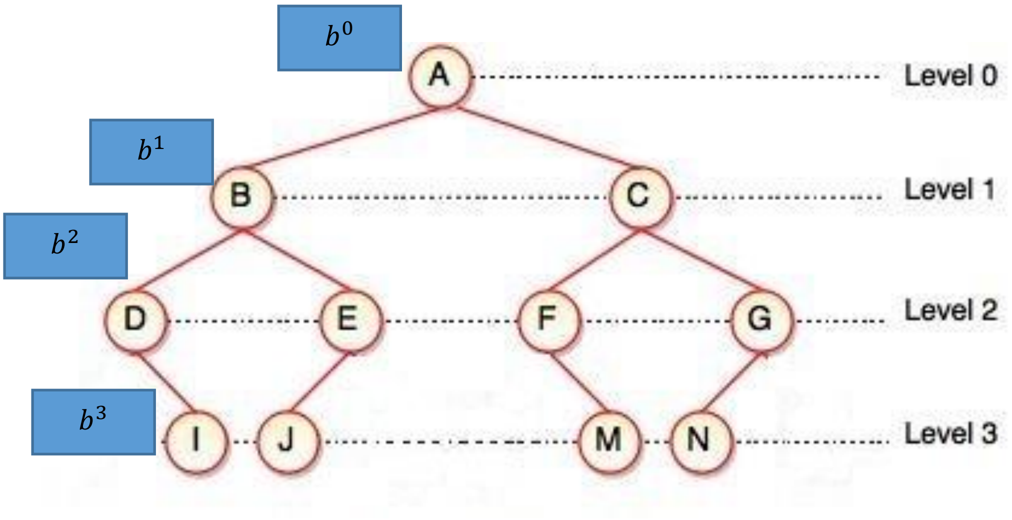
NOTE IF SEARCHING TRAVERSE AT EACH NODE THEN IF THE GOAL STATE IS SITUATED IN THE DEEPEST LAST NODE 'J':



SAY THEN:



IF GOAL STATE IS AT J THEN TRAVERSAL WILL OCCUR COVERING ALL THE DEPTHS INCLUDING ALL THE NODES.



$$b^0 + b^1 + b^2 + b^3 \ldots\ldots b^m = O(b^m)$$

$$where, m = maximum\ depth\ of\ any\ node,$$

$$b = branch\ factor \text{ i. e. how many children that a node has.}$$

AS SEARCHING MAY BE SHORTER BUT IT ALWAYS SEARCHES FOR DEEPEST NODE.

SPACE COMPLEXITY

FOR A STATE SPACE WITH BRANCHING FACTOR 'B' AND MAXIMUM DEPTH 'M', DEPTH FIRST SEARCH REQUIRES STORAGE ONLY :

$$O(bm)$$

NODES.