Iterative deepening DFS / Iterative deepening search

- It is general strategy
- often used in combination with DFS, that finds the best depth limit.
- it does this by gradually increasing the depth limit like first 0, then 1, then 2 & so on until goal is found.

How to perform ID DFS ?

- First perform DFS to depth d
  i.e treat start/root node as having no successors
- Then, if no solution/goal node found
  - perform DFS to depth 1
- Repeat until goal found

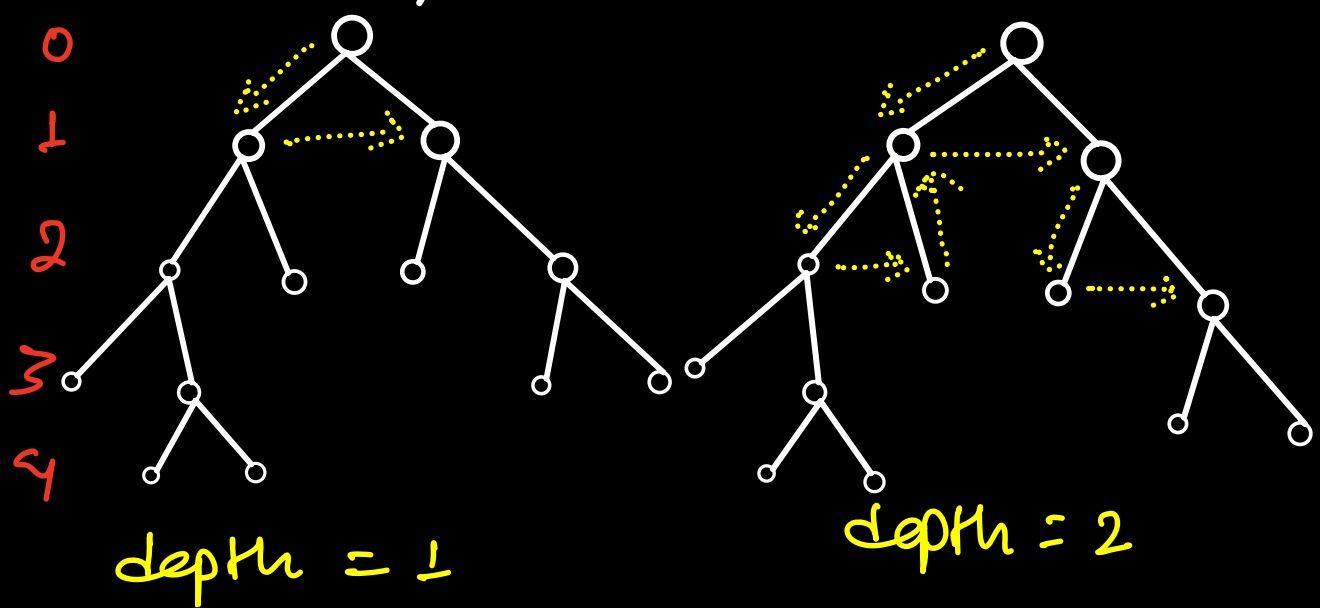
Algorithm

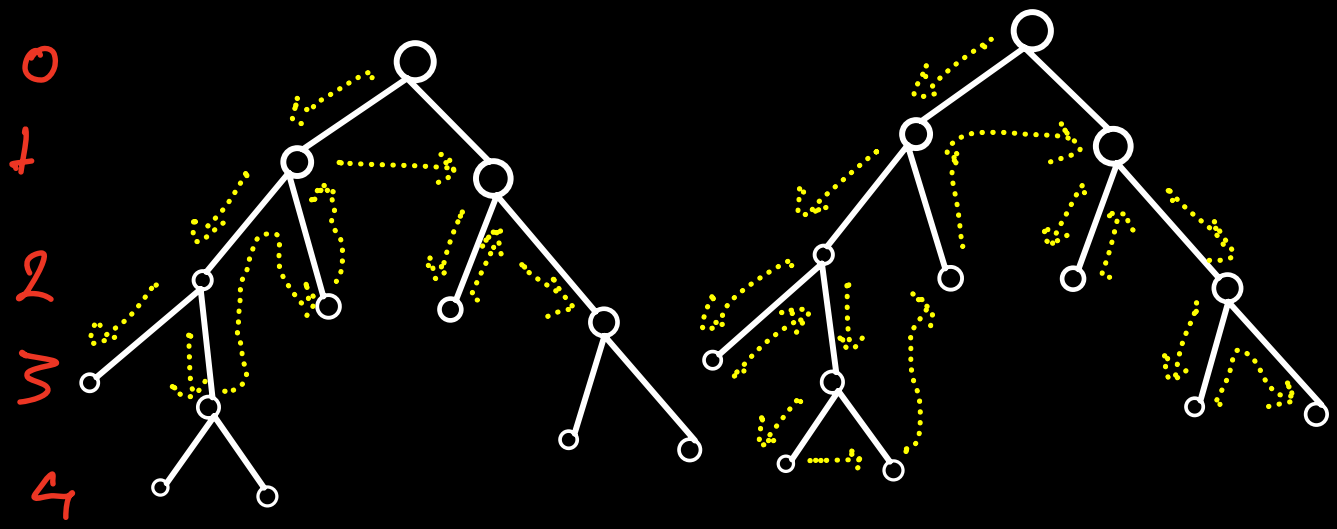
```
until solution/goal found do
    DFS with depth cutoff d
    d = d+1
```

Merits:

- Linear memory requirements of DFS.
- Guarantee for goal node of minimal depth.

Procedure / Illustration





depth = 3

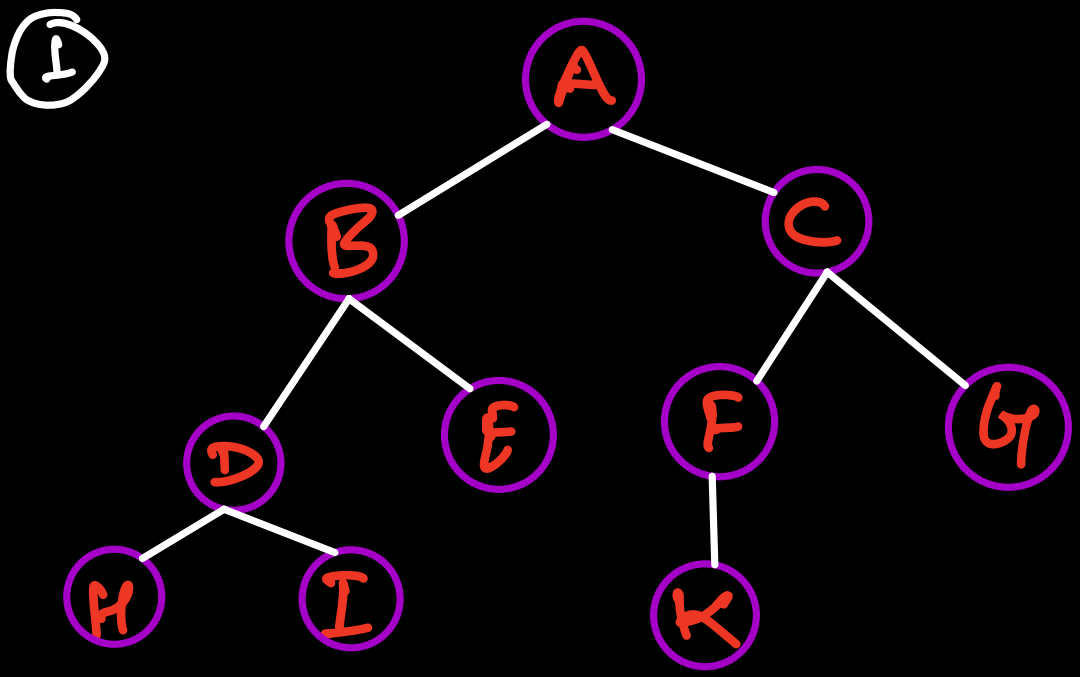
depth = 4

Evaluations:

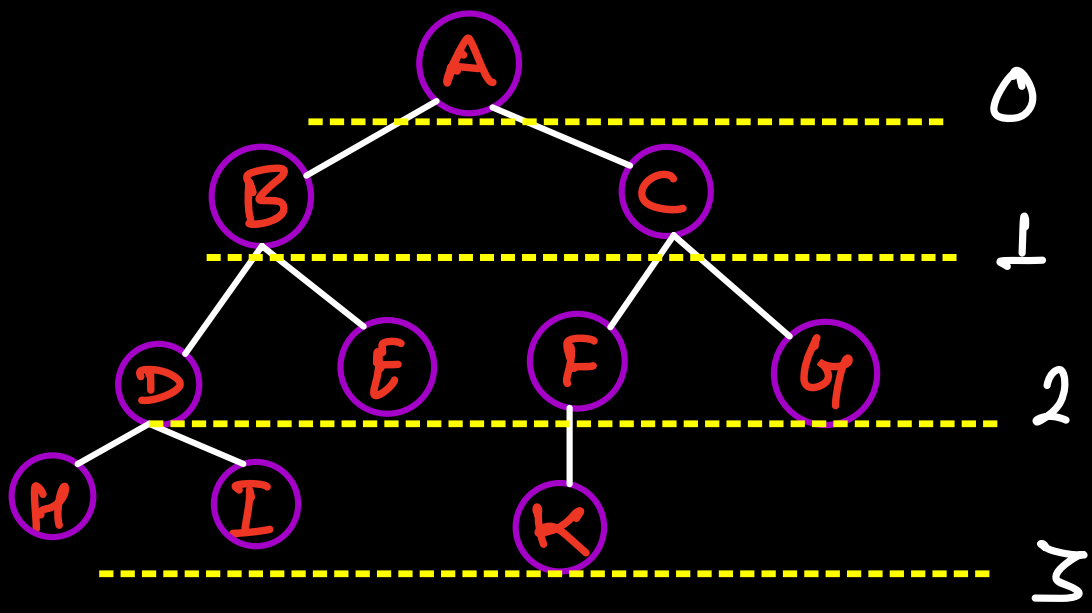
- Complete
- optimal / Admissible if all operators have uniform cost.
- Exponential time complexity i.e. $O(b^d)$

- Linear space complexity like DFS i.e. $O(bd)$

Example: Perform IDDFS



Hints: Depth limit



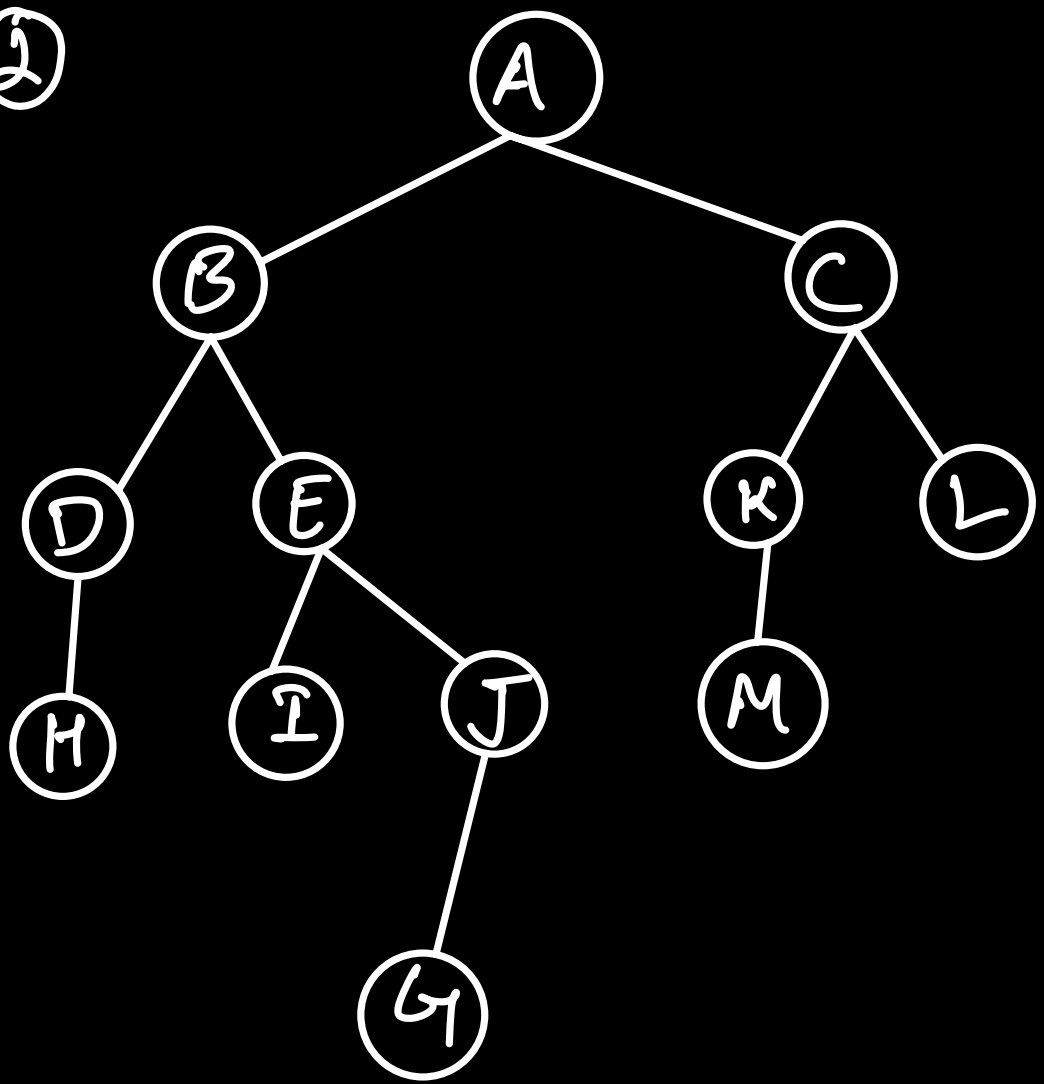
1st iteration: A

2nd iteration: A, B, C

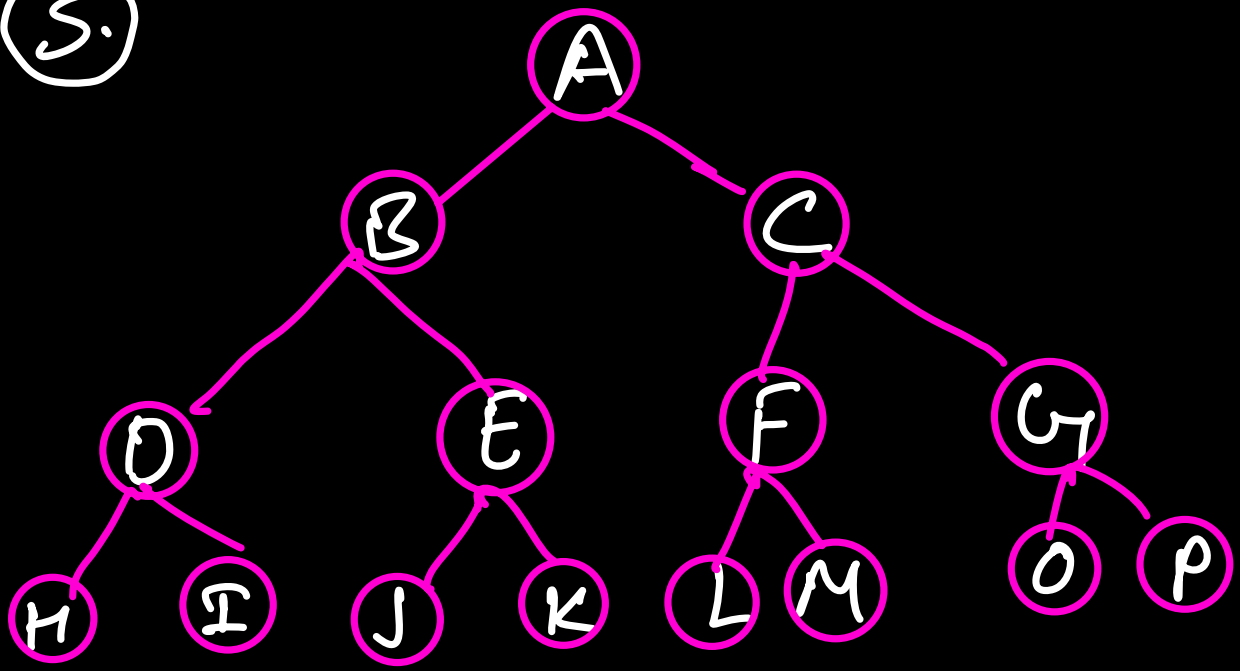
3rd iteration: A, B, D, E, C, F, G

4th iteration:
A, B, D, H, I, E, C, F, K, G

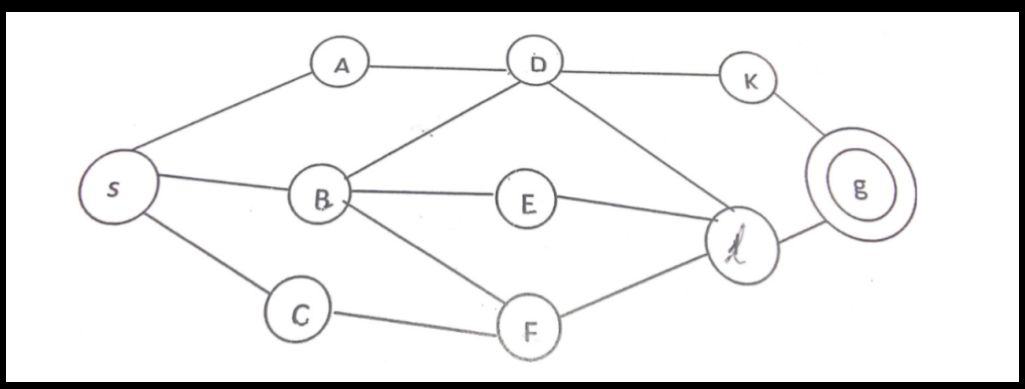
(2)



3.

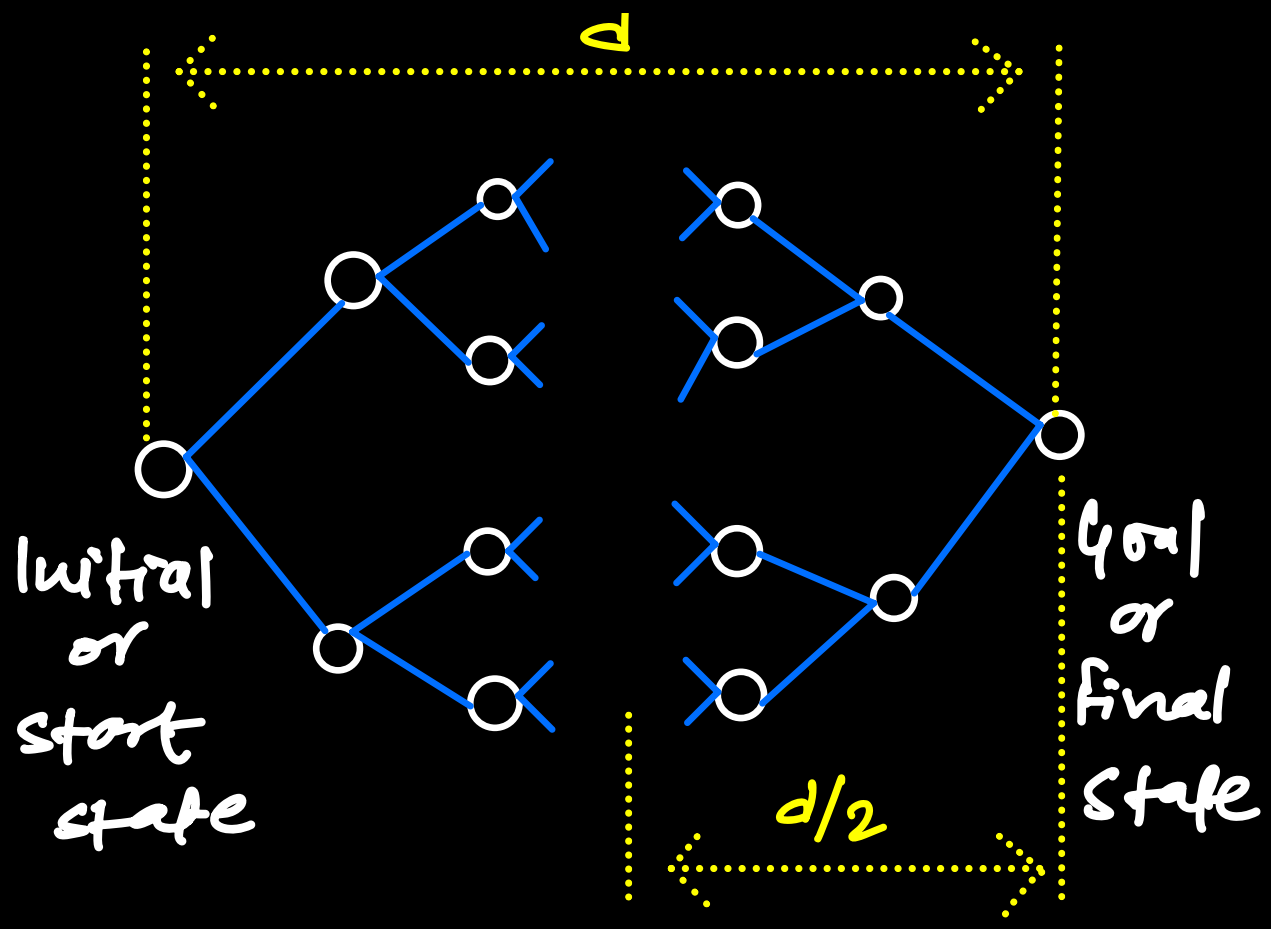


4



Bidirectional search

- Consider a search problem with bidirectional arcs.
- Like, there is an operator that maps from start state to goal state & another operator which maps from goal node to start node.
- Example:
  path finding problem



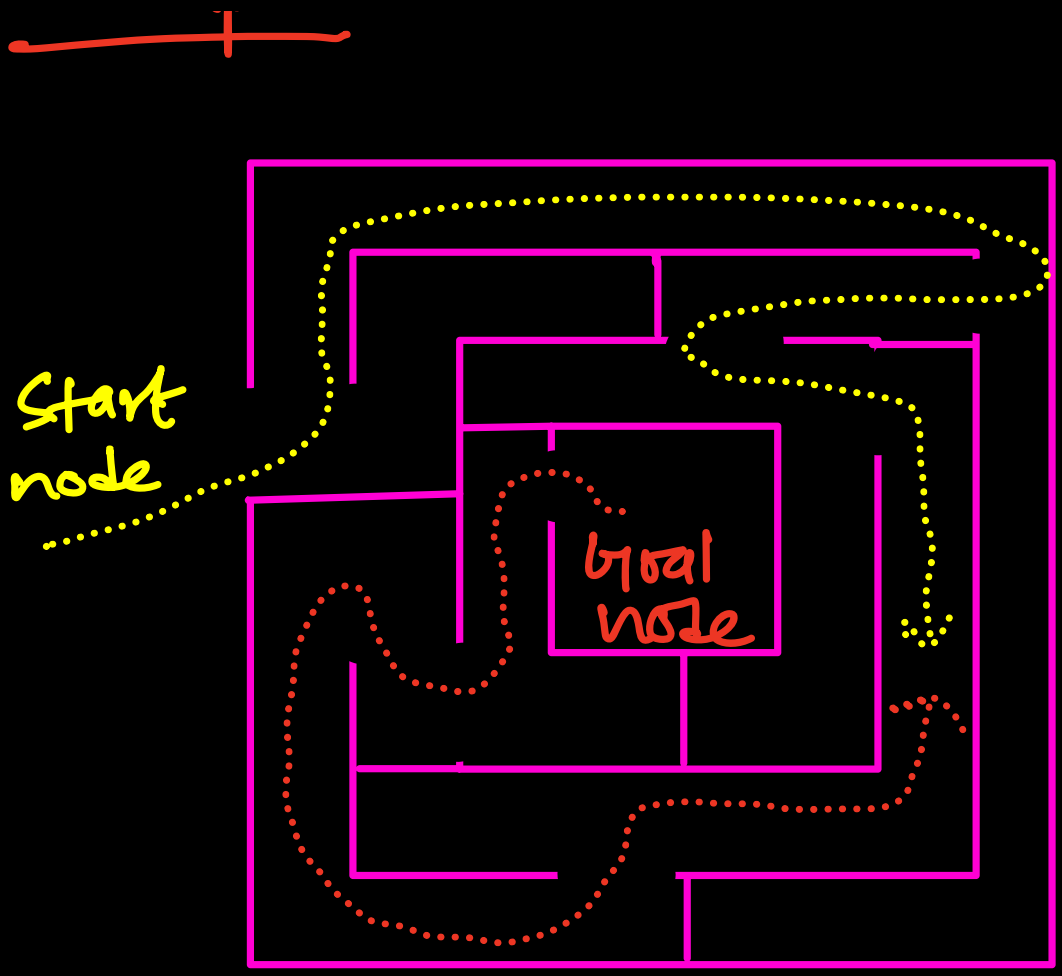
Evaluations

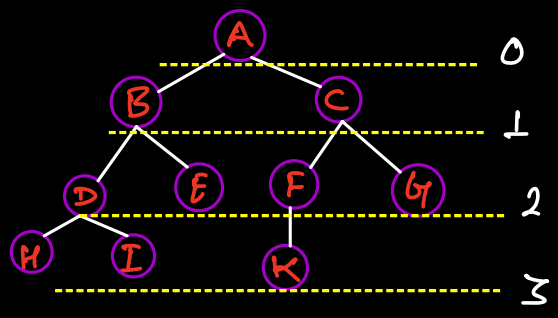
- Complete : Yes
- Optimal : Yes
- Time complexity $O(b^{d/2})$
- Space complexity $O(b^{d/2})$

Note:

- Bidirectional search involves alternate searching from
  - start node to goal node
  - goal node to start node
- Algorithms stops when frontiers intersect.

Example

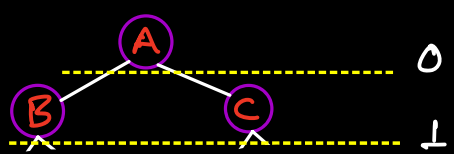

Start node
Goal node



1st iteration: depth 0

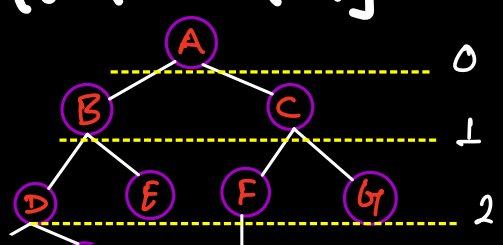
visited node: A

A is not a goal node, so, increase depth by 1.

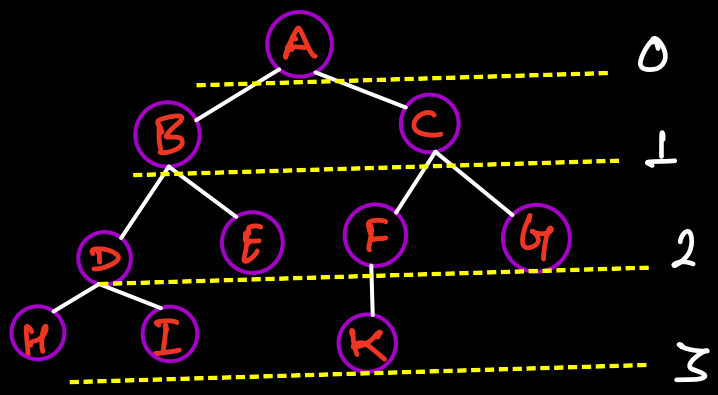


visited node: A, B, C

Goal not found at depth 1 so, increment depth by 1



visited node: A, B, D, E, C, F, G


A
B
C
D
E
F
G
H
I
K
0
1
2
3