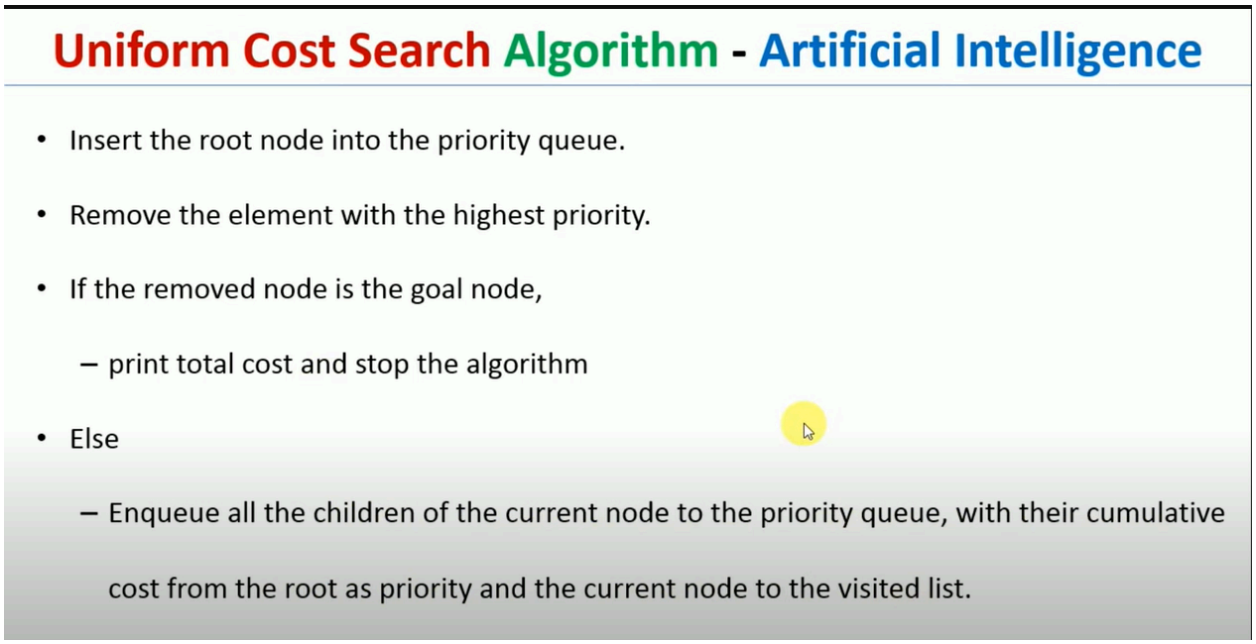
Sums 👇
Module 1
- PEAS Analysis

Module 2

1. Uniform Cost Search Algorithm

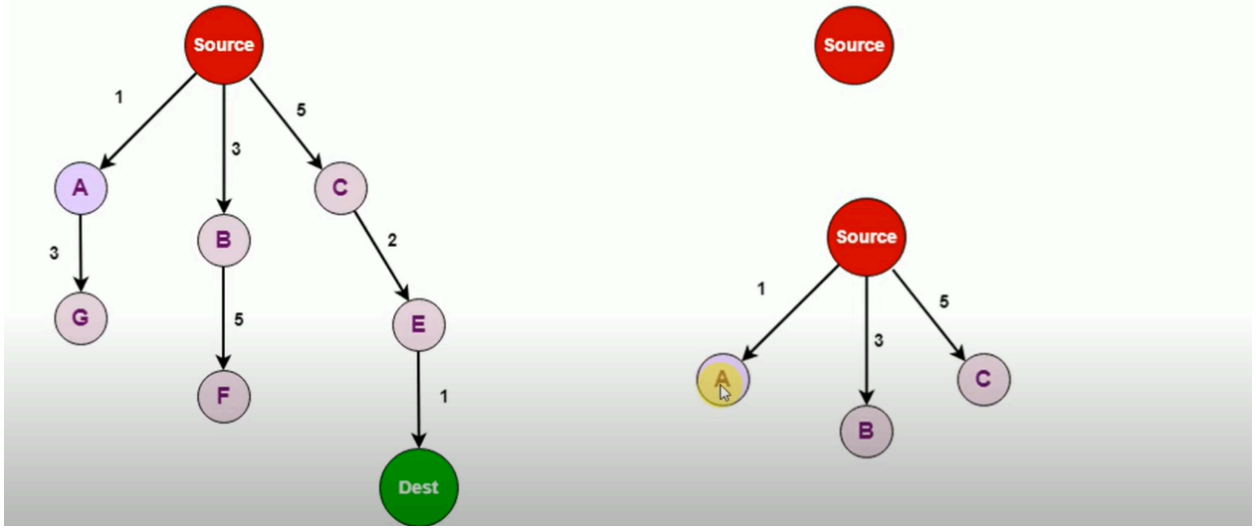
- Branch & bound
- Uninformed search algorithm
- Uses the lowest cumulative cost out of all paths/ costs
- Implemented using priority Queue

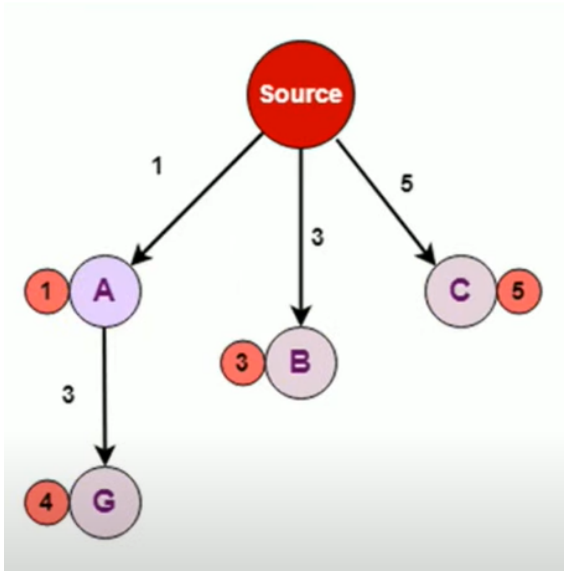
Algorithm

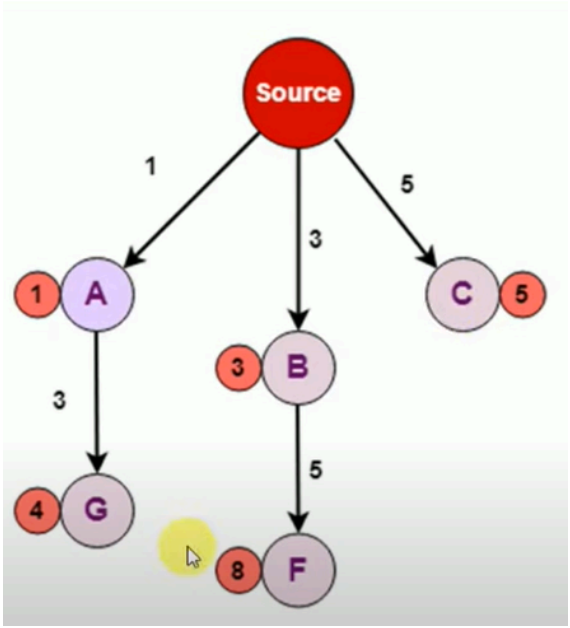


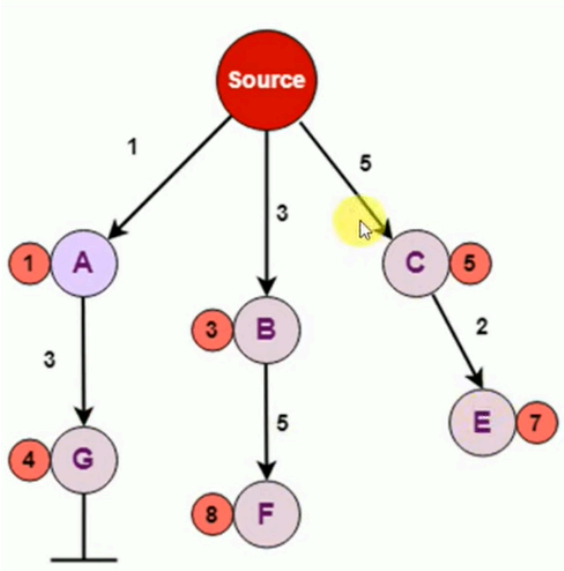
Example 1 : https://youtu.be/8ofimg8cnRE?feature=shared

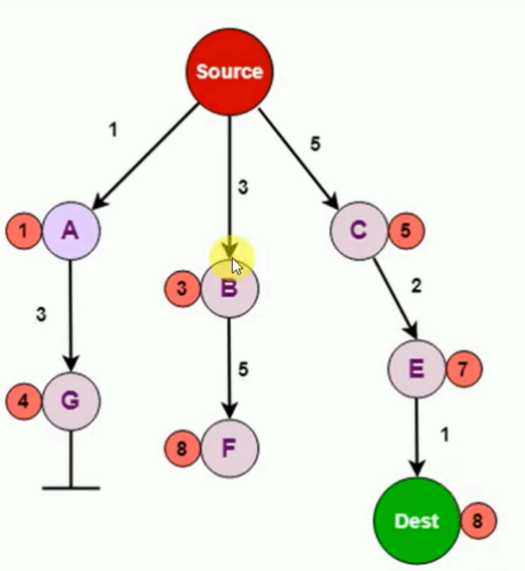
Uniform Cost Search Algorithm – Solved Example



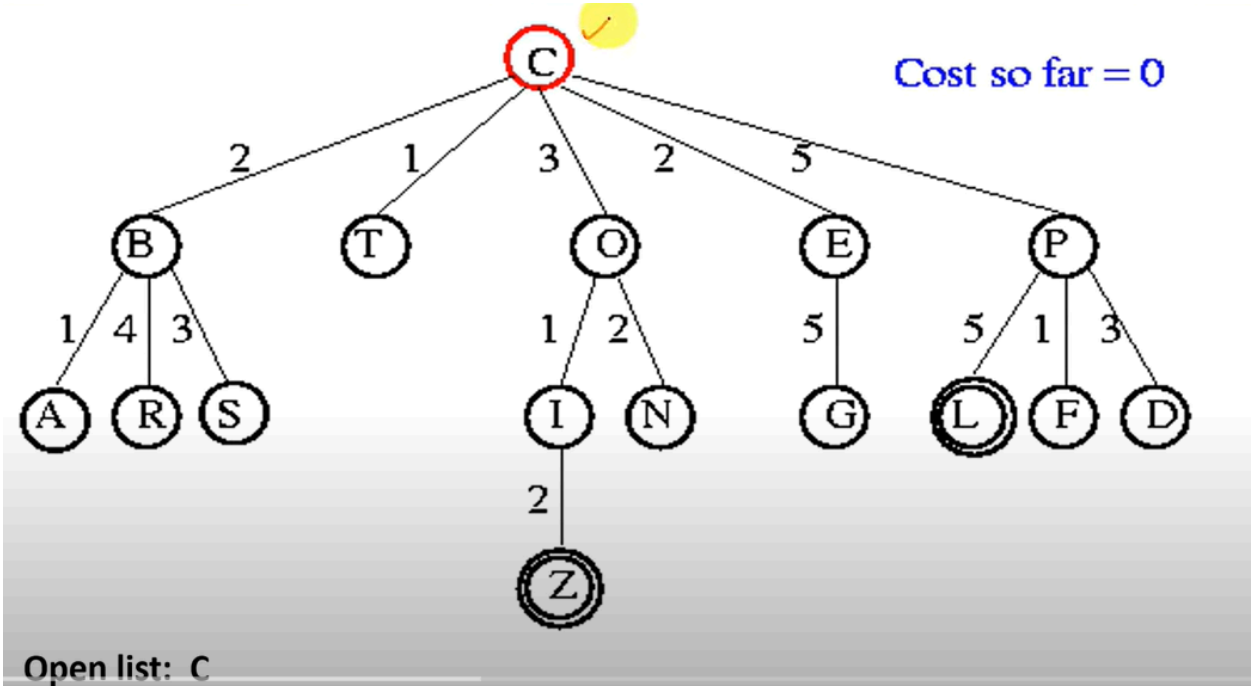




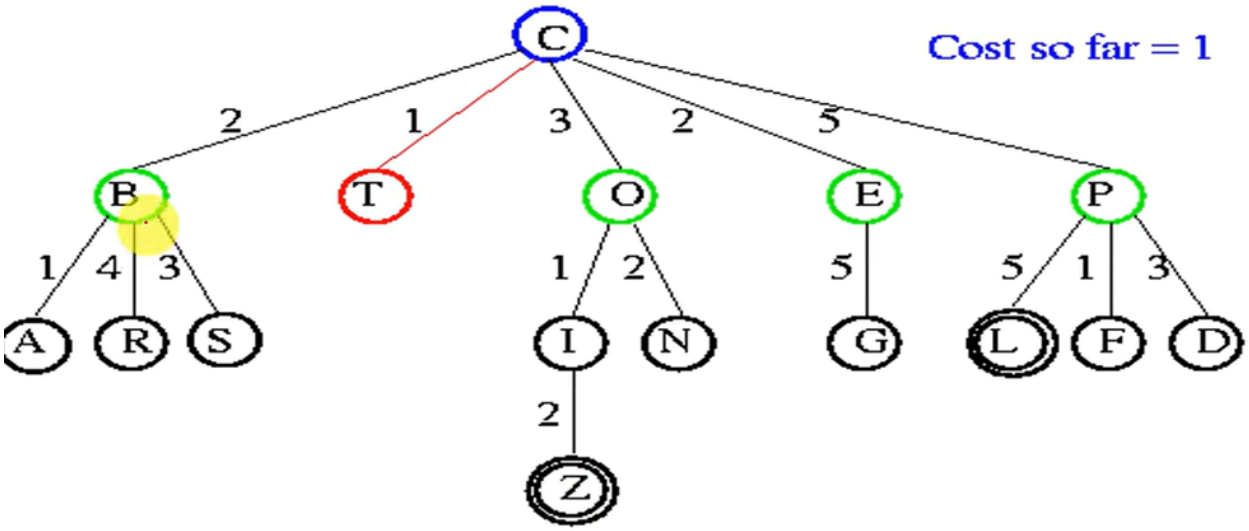




Example 2 : https://youtu.be/xkcR8-NEI1g?feature=shared



Open list: T(1) B(2) E(2) O(3) P(5)



Open list: B(2) E(2) O(3) P(5)

Open list: E(2) O(3) A(3) S(5) P(5) R(6)

Open list: A(3) I(4) S(5) N(5) P(5) R(6) G(7)

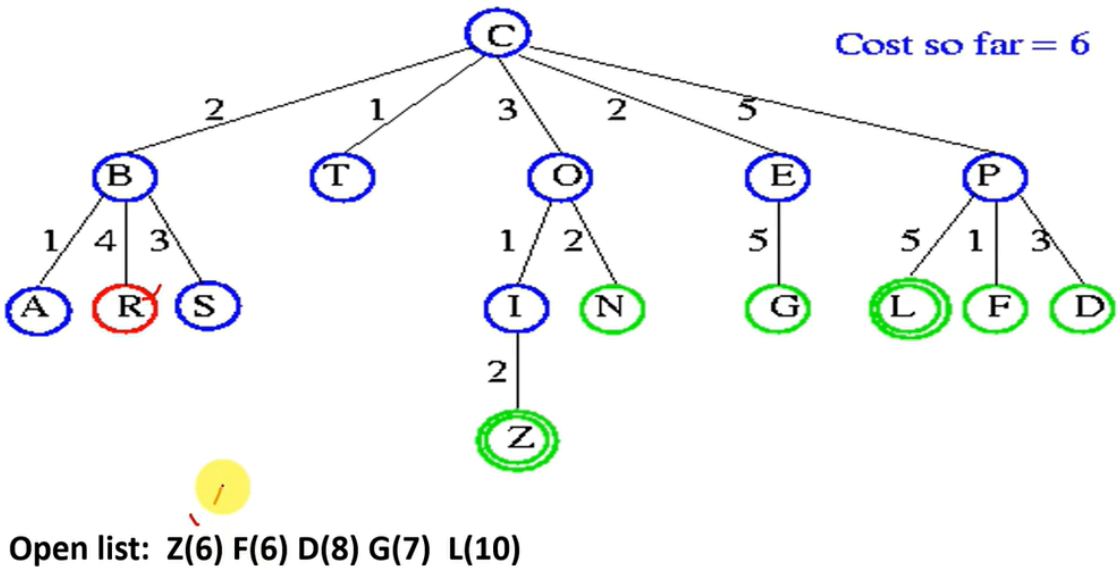
Open list: I(4) P(5) S(5) N(5) R(6) G(7)

Open list: P(5) S(5) N(5) R(6) Z(6) G(7)

Open list: S(5) N(5) R(6) Z(6) F(6) G(7) D(8) L(10)

Open list: N(5) R(6) Z(6) F(6) G(7) D(8) L(10)

Next R is selected but we don't have anything

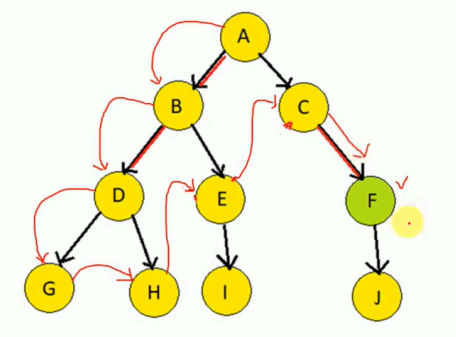


Open list: Z(6) F(6) D(8) G(7) L(10)

Reached Goal Node

2. Depth Limited Search : https://youtu.be/P7WQUBLKDmo?feature=shared

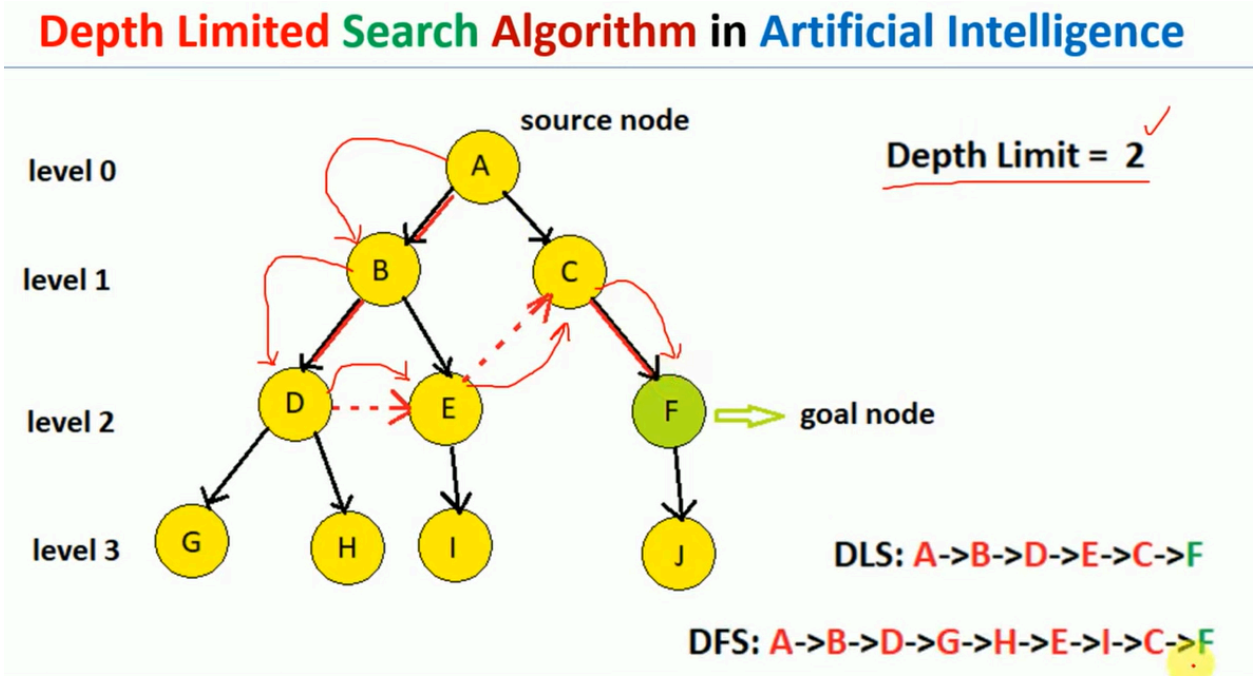
- Modified Version of DFS Algo
- DFS : Start at root node, go to the left subtree and search for goal node, if not found go to the right subtree



- If goal node is in Right Subtree a lot of time & memory is wasted to search the left subtree
- Hence we use Depth Limited Search

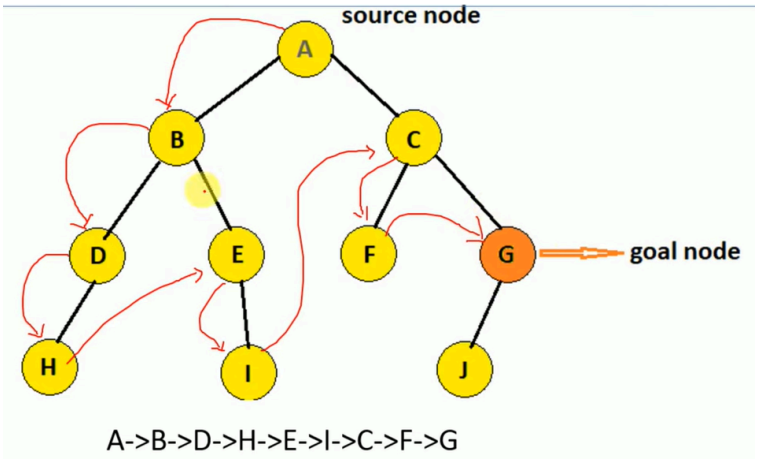
Algorithm
Same as DFS only we limit the depth

Example



3. Iterative Deepening : https://youtu.be/BK8cEWKHCkY?feature=shared

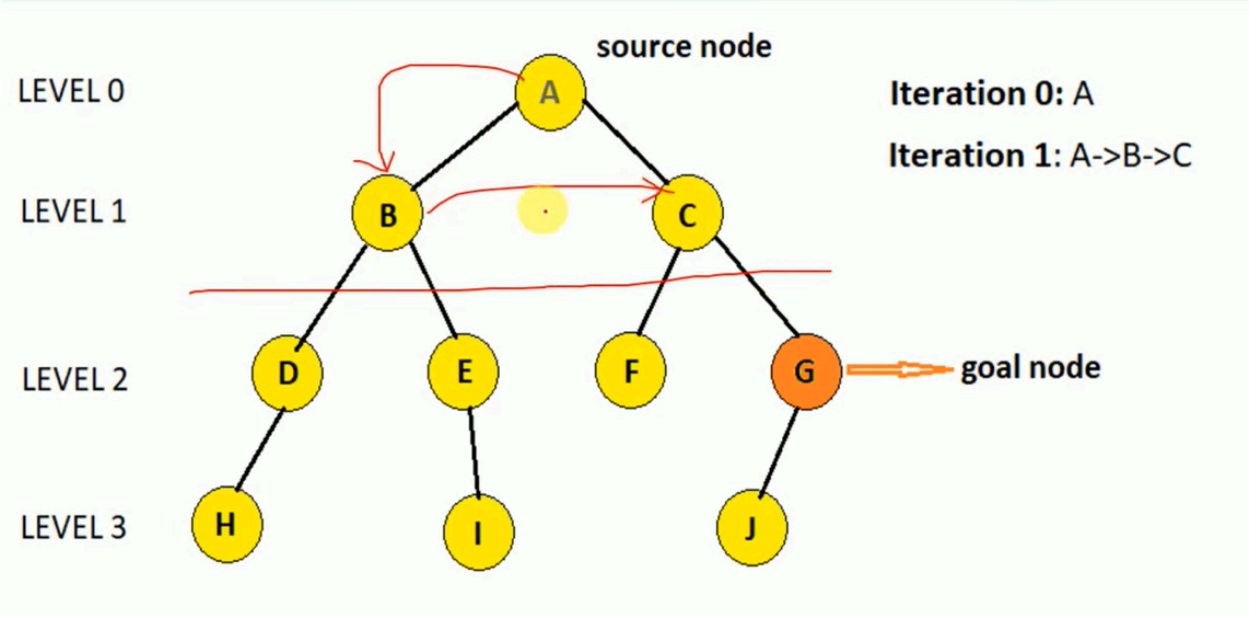
- Modified version of Depth Limited which is modified version of DFS
- DFS : Start at root node, go to the left subtree and search for goal node, if not found go to the right subtree

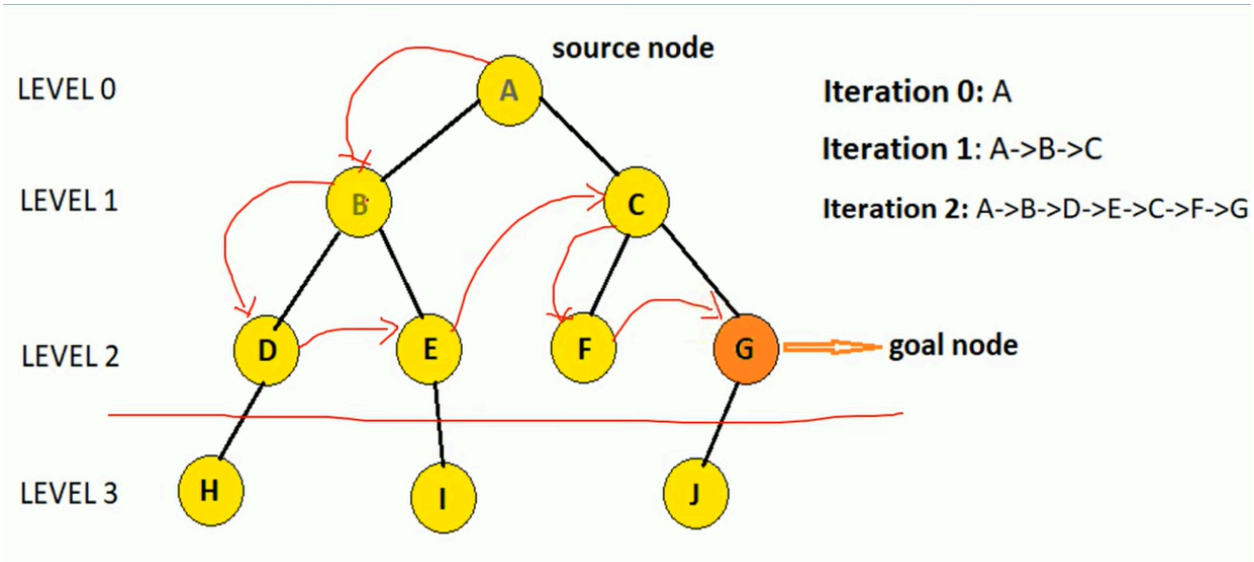


- Hence, in Iterative Deepening DFS, rather than going till the leaf node H, we put a restriction on level that can be searched

- If goal node is not found in that level, we increase the level limit and search the entire tree till that level

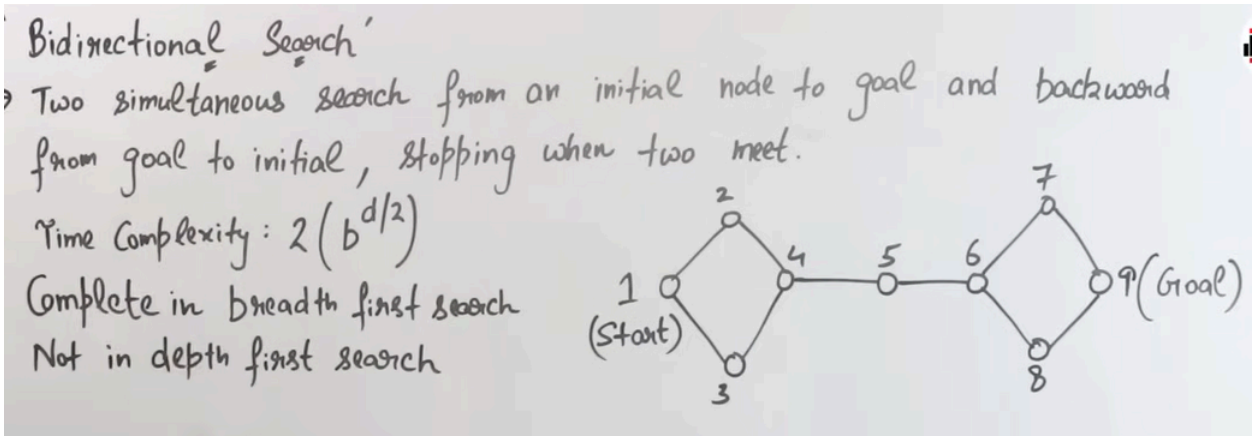
Example

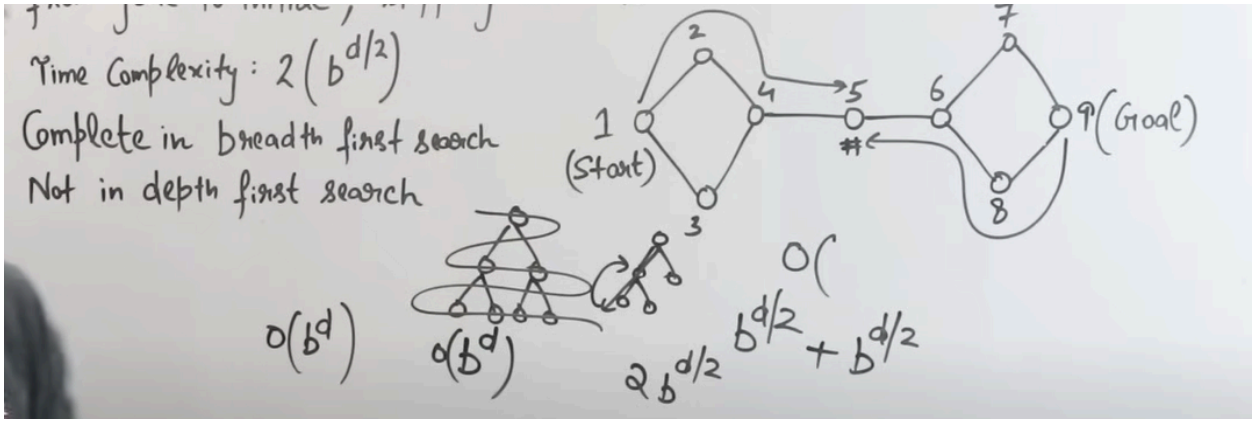


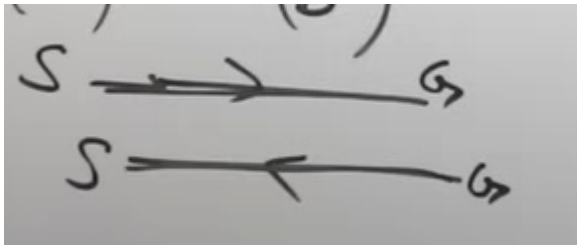


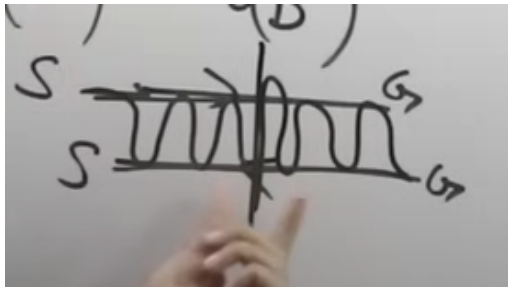
4. Bidirectional search : https://youtu.be/rEema9uQ02c?feature=shared

- Special extension to BFS & DFS

- 

- 

- FOR DFS: 

- FOR BFS: 

- Best First Search

Best-First Search in Artificial Intelligence

- Best First Search algorithm combines the advantages of both DFS and BFS into a single method.
- At each step of the BFS search process, we select the most promising of the nodes we have generated so far.
- This is done by applying an appropriate heuristic function to each of them.
- We then expand the chosen node by using the rules to generate its successors
- To implement such a graph-search procedure, we will need to use two lists of nodes:
  - **OPEN** — nodes that have been generated and have had the heuristic function applied to them, but which have not yet been examined (i.e., had their successors generated).
  - **CLOSED** — nodes that have already been examined. We need to keep these nodes in memory if we want to search a graph rather than a tree, since whenever a new node is generated, we need to check whether it has been generated before.

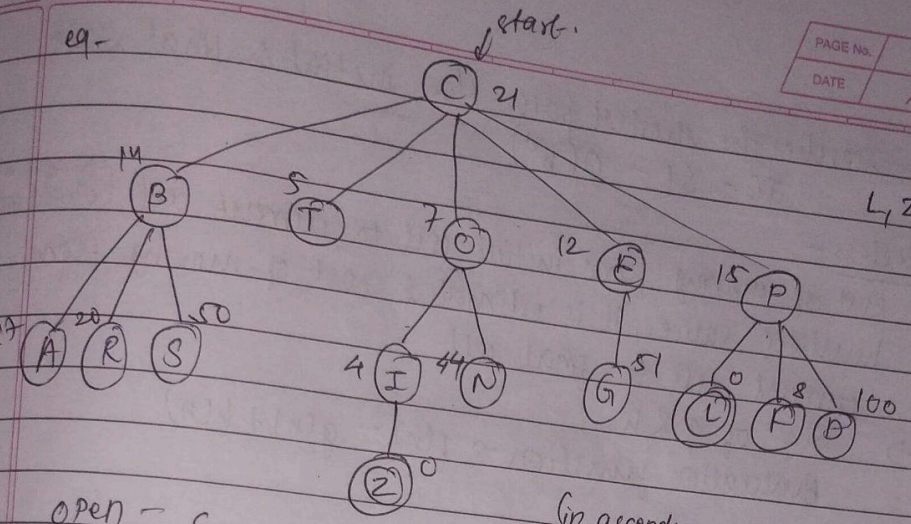
Algorithm: Best-First Search

1. Start with OPEN containing just the initial state.
2. Until a goal is found or there are no nodes left on OPEN do:
   a) Pick them best node on OPEN.
   b) Generate its successors.
   c) For each successor do:
      i. if it has not been generated before, evaluate it, add it to OPEN, and record its parent.
      ii. If it has been generated before, change the parent if this new path is better than the previous one. In that case, update the cost of getting to this node and to any successors that this node may already have.

Solved eg- Mahesh ke hi video ka hai bas utne ss nahi dalne the to solve kara hua dal raha, Arrows follow karo wo seq me execute ho rahe!!

EG1
EG2

eg-

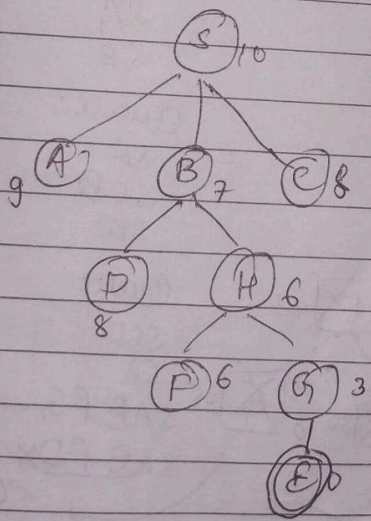


L, Z → goal

open - C
closed - [ ]
→ open - (I, O, E, B, P) (in ascending order) → open - O, E, B, P
closed - C
closed - C, I,

open - E, B, P
closed - C, I, O
←
open - I, E, B, P, N
closed - C, I, O,
←
open : Z, E, B, P, N
closed - C, I, O, I

now 'Z' goal node is found so stop algo C → O → I → Z. Anal path.



open - S
closed - [ ]
→ open - B, C, A
closed - S

open - H, C, D, A
closed - S, B
→ open - G, F, C, D, A
closed - S, B, H

open - E, F, C, P, A
closed - S, B, H, G

Goal node found
Path → S → B → H → G → E

- A*

- Algorithm A* (Hart et al., 1968):

$$f(n) = g(n) + h(n)$$

- h(n) = cost of the cheapest path from node n to a goal state.
- g(n) = cost of the cheapest path from the initial state to node n.

1 & 2 EG dekhlo enough hai, 3rd EG UT wala Q hai

EG1
EG2
EG3

- Hill Climbing

- Crypto-Arithmetic Problem

Rules for Solving Cryptarithmetic Problems

1. Each Letter,Symbol represents only one digit throughout the problem.
2. Numbers must not begin with zero i.e. 0567 (wrong), 567 (correct).
3. The aim is to find the value of each letter in the Cryptarithmetic problems
4. There must be only one solution to the Cryptarithmetic problems
5. The numerical base, unless specifically stated, is 10.
6. After replacing letters with their digits, the resulting arithmetic operations must be correct.
7. Carryover can only be 1 in Cryptarithmetic problems involving 2 numbers.

EG1
EG2

- Water Jug

- Graph Coloring

- Min-Max Search
Bas Min and Max value according to the node nikalte jate leftmost followed by right ez
EG1
EG2

- Alpha Beta Pruning
EG1
EG2

Module 3
- FOPL
Ex
- FOL to CNF conversion
Ex
- Forward and Backward Chaining

Module 5
- Covariance and Correlation
- ANOVA
https://www.cuemath.com/anova-formula/
https://www.statisticshowto.com/tables/f-table/