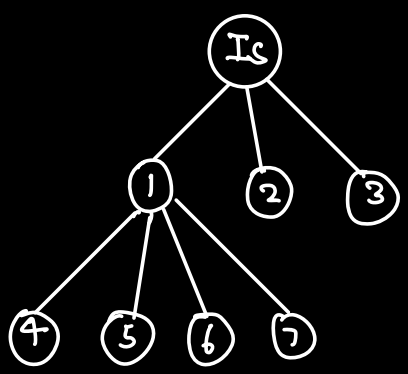
The Search Process

Terminology:

- Generate Children
- Expand
  - (1) Check goal
  - (2) Generate Children
- Fringe / frontier : The unexplored nodes



Fringe: {2, 3, 4, 5, 6, 7}

Expand a fringe, say 2

↳ Check goal → not the Es

↳ Generate Children {8, 9}.

Idea: 1. Choose a node in fringe

2. Expand it

3. Repeat.

> We check the goal during expansion not Generation!

SEARCH STRATEGIES:

A search strategy is a criteria for deciding which node on the fringe to **expand next**.

EVALUATING SEARCH STRATEGIES:

1. COMPLETENESS:

   Does it always find a solution if one exists.

2. TIME COMPLEXITY:

   Number of nodes generated.

3. SPACE COMPLEXITY:

   Maximum number of nodes in memory.

4. OPTIMALITY:

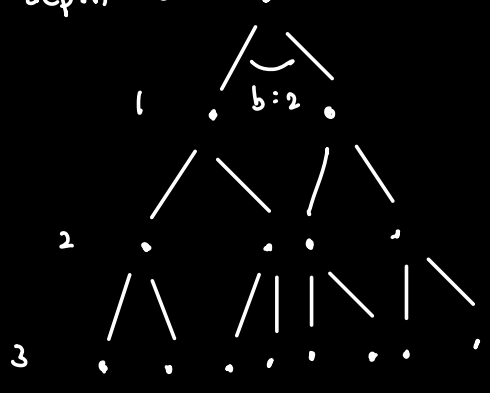
   Does it find a least-cost solution.

PARAMETERS FOR EXPRESSING COMPLEXITY:

1. b : Branching factor
2. d : Depth of least_cost solution.
3. m : maximum depth of the search tree (could be infinite)

OBSERVATION ABOUT TREES:

1. depth = 0

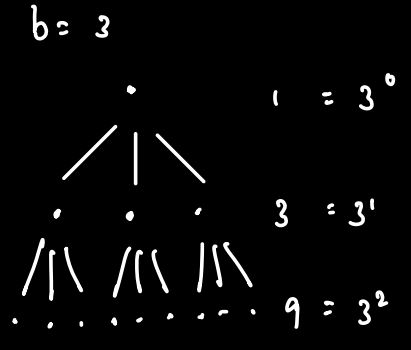


$1 = 2^0$

$2 = 2^1$

$4 = 2^2$

$8 = 2^3$



$$N(b,d) = 1 + b + \dots + b^d$$

$$bN(b,d) = b + b^2 + \dots + b^{d+1}$$

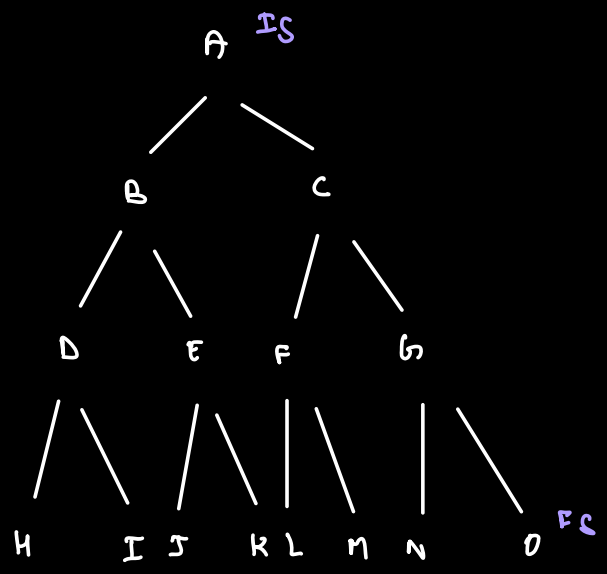
$$(b-1)N(b,d) = b^{d+1} - 1$$

$$N(b,d) = \frac{b^{d+1} - 1}{b-1} \simeq \frac{b^{d+1}}{b-1} = \left(\frac{b}{b-1}\right) b^d$$

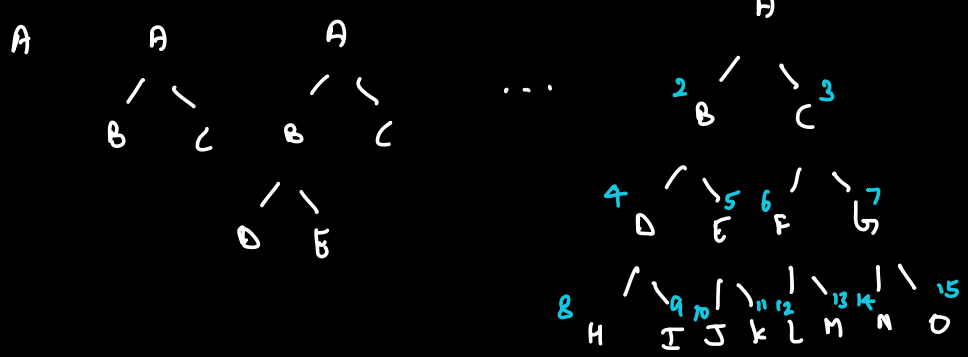
$$= O(b^d) \;//$$

2. Last layer has more nodes than all other layers combined.

(BFS) BREADTH FIRST SEARCH: "Shallowest node first"

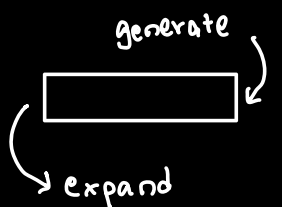


1. Number nodes according to expand
2. Implementation Note
3. Evaluation Criteria



② Implementation note:

Use Queue



A

AC

ADE

ADEFB

③ EVALUATION CRITERIA:

1. Complete :

Expands all nodes at depth i before any node at i+1.

2. Optimality :

We expand lower levels before going to higher. Hence, it is optimal.

3. Space Complexity;

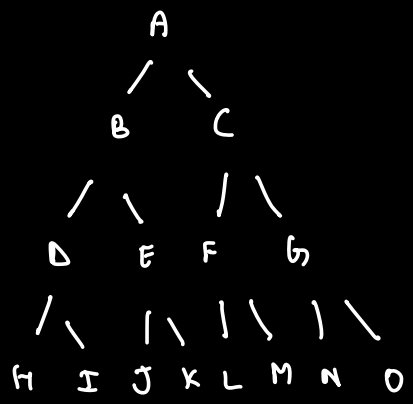
$O(b^d)$. Need full tree to check answer.

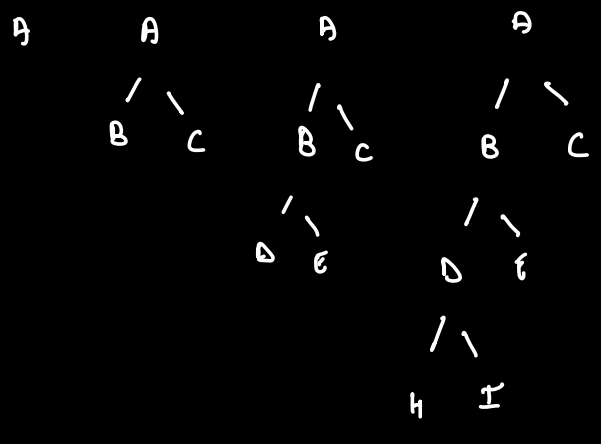
4. Time Complexity

$O(b^d)$. Sum of nodes.

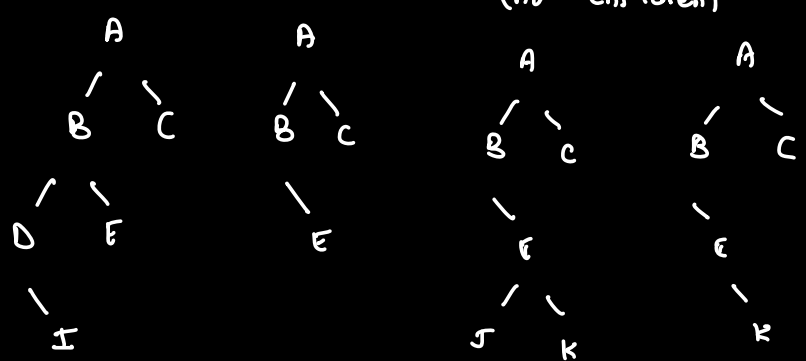
$b^0 + b^1 + \dots + b^d$.

(DFS) Depth-First Search : "deepest node first"

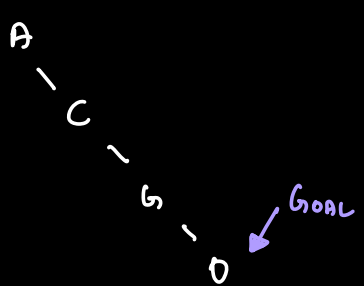




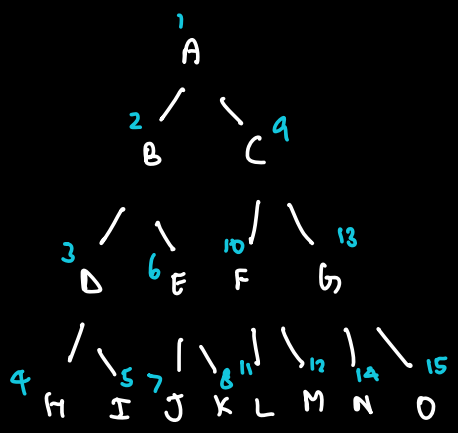
expand, not an answer (no children)



...



1. Order of expansion:



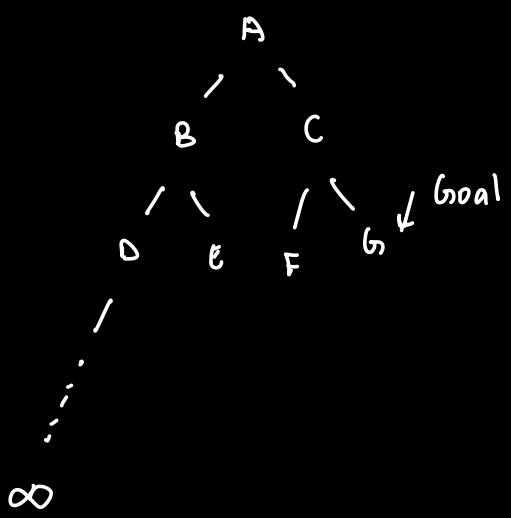
2. Implementation note:

Use a stack

3. Evaluation Metrics

1. Complete:

No, we might just infinitely expand (infinite tree).



2. Optimal:

No

3. Space Complexity:

$O(bm)$

4. Time Complexity:

$O(b^m)$

Limited Depth Search:

Limit depth to l

1. Complete :

Yes if $d \leq l$

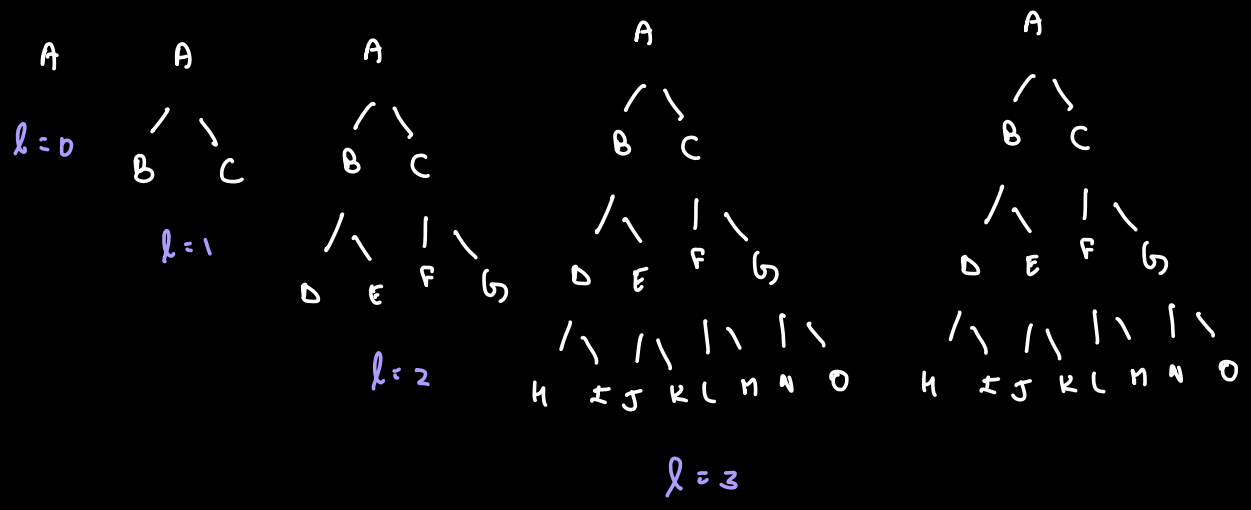
2. Optimal :

No

3. Time Complexity : $O(b^l)$

4. Space Complexity : $O(bl)$

Iterative Deepening Search:



1. Complete :

   Yes

2. Optimal :

   Yes

3. Time Complexity : $O(b^d)$

4. Space Complexity : $O(bd)$

Time Complexity of ID:

$l = 0$ $\quad b^0 (b/b-1)$

$l = 1$ $\quad b^1 (b/b-1)$

$l = 2$ $\quad b^2 (b/b-1)$

$\vdots$

$l = d$ $\quad b^d (b/b-1)$

$$\text{Sum} = (b/b-1)(b/b-1)b^d$$

$$= (b/b-1)^2 b^d$$

For $b = 10, d = 3$

Tree: b, d

$b^d (b/b-1)$ $\rightarrow$ 111,111

ID: $b^d (b/b-1)^2$ $\rightarrow$ 123,456

} very close.

Ratio: $b/b-1$

$b = 2 \Rightarrow 2$

$b = 3 \Rightarrow 1.5$