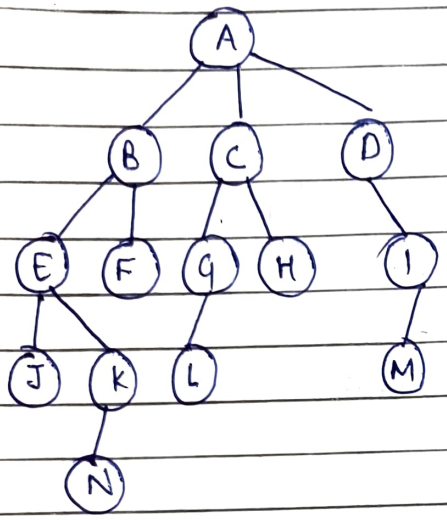
Unit 2 : State Space Search

PAGE NO.
DATE: / 202

State space search is used to explore all potential configurations or states of an instance until one w/ the necessary feature is found.

1. BFS (Breadth First Search) – Uninformed Search

- Inspects nodes on a FCFS basis, ie, nodes form a queue to be inspected.
- Also called level search technique. (moves level by level)



queue

A
BCD
CDEF
DEFGH
EFGHI
FGHIJK
GHIJK
HIJKL
IJKL
JKL
KLM
LMN
MN
N

Completeness : Complete for finite state space ($\neq \infty$)

Time Complexity: $N_{BFS} = \frac{b^d - 1}{b - 1}$ ~~$+ \frac{1 - b^d}{2}$~~

The search arrives at level d after examining the entire subtree above it.

If the final state is the leftmost node, it picks that node.
rightmost, b^d nodes more.

PAGE NO.
DATE: / 202

~~Space complexity~~:

Leftmost : $1 + b + \dots + b^{d-1} = \frac{b^d - 1}{b-1}$

Rightmost : $1 + b + \dots + b^{d-i} + b^d = \frac{b^{d+1} - 1}{b-1}$

$$N_{BFS} = \frac{\frac{b^d - 1}{b-1} + \frac{b^{d+1} - 1}{b-1}}{2}$$

$$\approx \frac{b^d (b+1)}{2(b-1)}$$

$$\approx \frac{b^d}{2} \quad \text{(for large } b\text{)}$$

Space complexity : BFS pushes into the tree level by level. As it enters each level at depth d, it sees all b^d nodes ahead of it in OPEN. When it enters $(d+1)$, it sees b^{d+1} nodes in OPEN.

∴ Size of OPEN grows exponentially, and hence space complexity is exponential.

Quality of solution : Where BFS loses out on space complexity, it makes up on the quality of the soln. as it always finds the shortest solution.

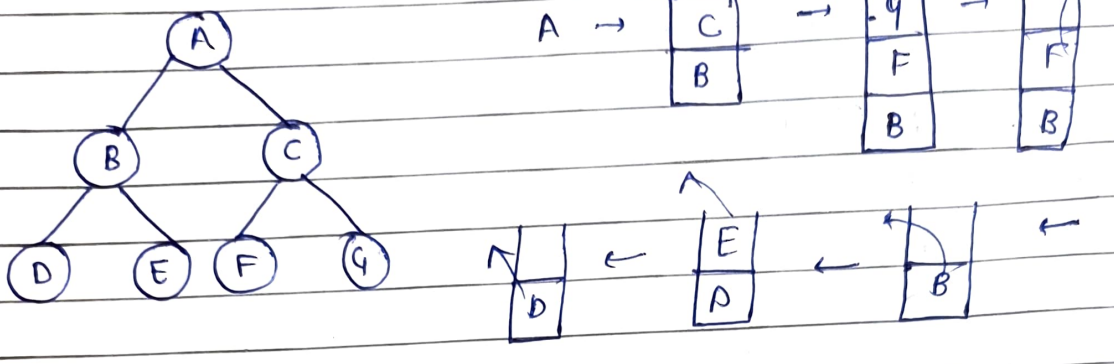
(pushes into the tree level by level, inspects soln in ↑ order of their length).

∴ It is optimal

PAGE NO.
DATE: / 202

2. DFS (Depth First Search) : Uninformed

- Treats OPEN like a stack
- Selects the deepest node from the start node. Hence called depth first.
- Candidates are inspected in LIFO order.



order of visit : ACGFBED

Completeness : complete for finite state space
Incomplete for ∞

Time complexity :

~~Left~~ If goal is in the leftmost branch at depth d, it is found after examining d nodes.

If goal is on the ~~rig~~ extreme right, we have to search the entire tree.

Leftmost : $d+1$

Rightmost : $\frac{b^{d+1}-1}{b-1}$

$$N_{DFS} = \frac{d+1 + \frac{b^{d+1}-1}{b-1}}{2}$$

$$\approx \frac{b\, b^{d+1}}{2(b-1)} \qquad b : \text{v large}$$

$$\approx \frac{b^d}{2}$$

PAGE NO.
DATE: / 202

Space Complexity : $(b-1)$ nodes to OPEN at every depth. add

$$O_{DFS} = (b-1)(d-1) + b$$
$$= d(b-1) + 1$$
$$\sim O(bd)$$

Quality of soln : DFS returns the first soln found, which holds no guarantee it will be the shortest one. It may return a non-optimal soln.

3. Depth Bounded DFS (DBDFS)

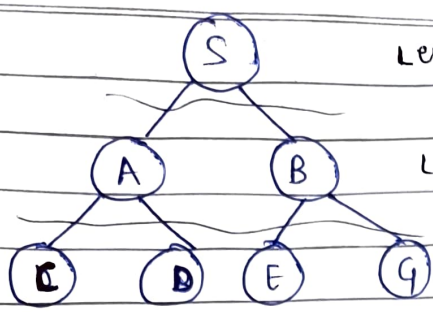
Similar to DFS but w/ predetermined limit
→ Memory efficient (adv)
→ can be terminated w/o finding soln (disadv)

Completeness : since the depth bound is artificially imposed, the algo is not complete in the general case.

Rest same as DFS

4. Depth First Iterative Deepening (DFID)
→ Combination of BFS & DFS.
→ Best depth limit is found by gradually ↑ limit
→ In every cycle, it implements DFS w/ bound incremented by 1.

PAGE NO.
DATE: / / 202



Level 0 — 1st itⁿ : d=0, [S]

Level 1 — 2nd : d=1 [S → A → B]

Level 2 — 3rd : d=2 [S → A → C → D]
[G ← E ← B]

Completeness : complete

~~Time complexity : since it explores performs DFS on each cycle, TC = DFS Nodes = NDFID = NDFS~~

Time complexity : For b = 5 (b → branching factor)

depth	nodes visited
0	$5^0 = 1$
1	$5^0 + 5^1 = 6$
2	$5^0 + 5^1 + 5^2 = 31$

∴ Nodes covered at depth d = $b^0 + 1 + b + b^2 + \dots + b^d$

$= \frac{b^{d+1} - 1}{b - 1} \approx b^d$

∴ TC = $O(b^d)$

Space complexity: Since it performs DBDFS in every iteration, space complexity is that of DBDFS.

$O(bd)$

Quality of soln : Same as DBDFS

PAGE NO.
DATE: / 202

Additional cost of DFID:

In each cycle, it explores a new level of nodes. But for this, it has to generate the tree all over again.

↑ regeneration of tree

Internal nodes (extra cost)

Leaf nodes (new nodes)

↳ new level of nodes

$$= \frac{1 + b + b^2 + \cdots + b^{d-1}}{b^d}$$

$$= \frac{b^d - 1}{b-1(b^d)} \approx \frac{1}{b-1}$$