2: Searching Techniques (30 mk)

A: Uninformed Searching

B: Informed Searching

C: Adeversial Search

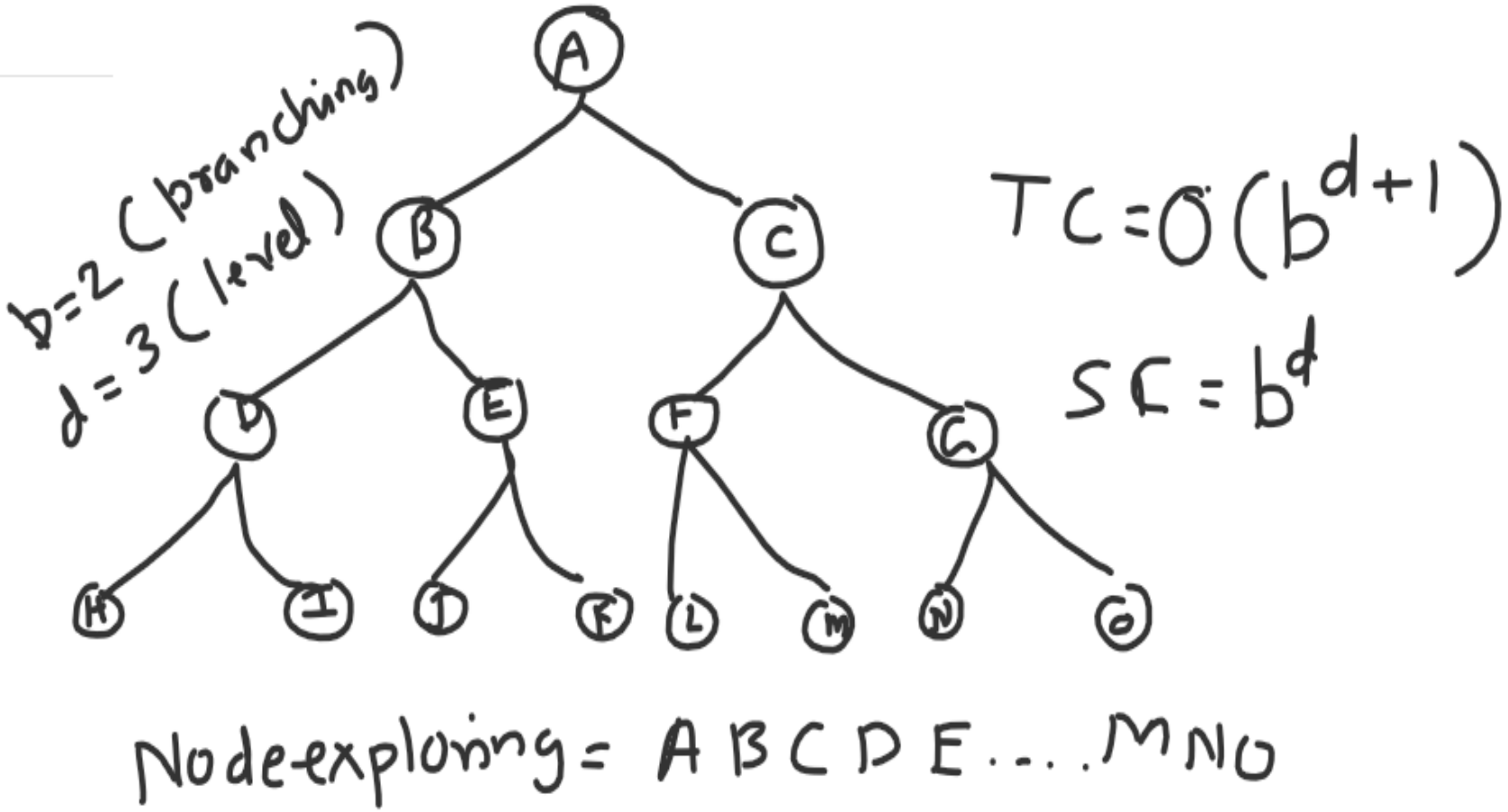
D: Constrain based Search (CSP)

A: Uninformed Search (Brute Force Search)

- If no additional knowledge of state is given then it is uninformed search or **Blind search** .
- Here information of START state, GOAL state and search space is given. How **far we are from goal OR are we searching in right direction,** we can predict till we reach goal
- This technique keep on exploring successor (next) state till a solution(goal state) is located.

Example : BFS, DFS, Depth Limit Search, Iterative Deeping Search(IDS), Uniform Cost Path, Bidirectional Search .

BFS


b=2 (branching)
d=3 (level)
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
TC=O(b^{d+1})
SC=b^d
Node exploring = A B C D E .... M N O

Explain BFS & DFS with advantage and drawback?(Univ Q)

BFS:

- Here root node is expanded first then all successors of the root node are expanded next, then their successors and so on.
- In general all the nodes are expanded at a given depth in the search tree before any nodes at the next level,are expanded.
- We can also say that this tree is expanded breadth wise(horizontally)
- It is implemented by tree search using FIFO queue as shallowest unexplored nodes is chosen for expansion.
- If every state has b successors(Branching factor) then till depth(d) total number of nodes generated are:- ($b+ b^2 + b^3 + b^4 + \ldots\ldots + b^d$)

Explain BFS & DFS with advantage and drawback?(Univ Q)

Advantage BFS:

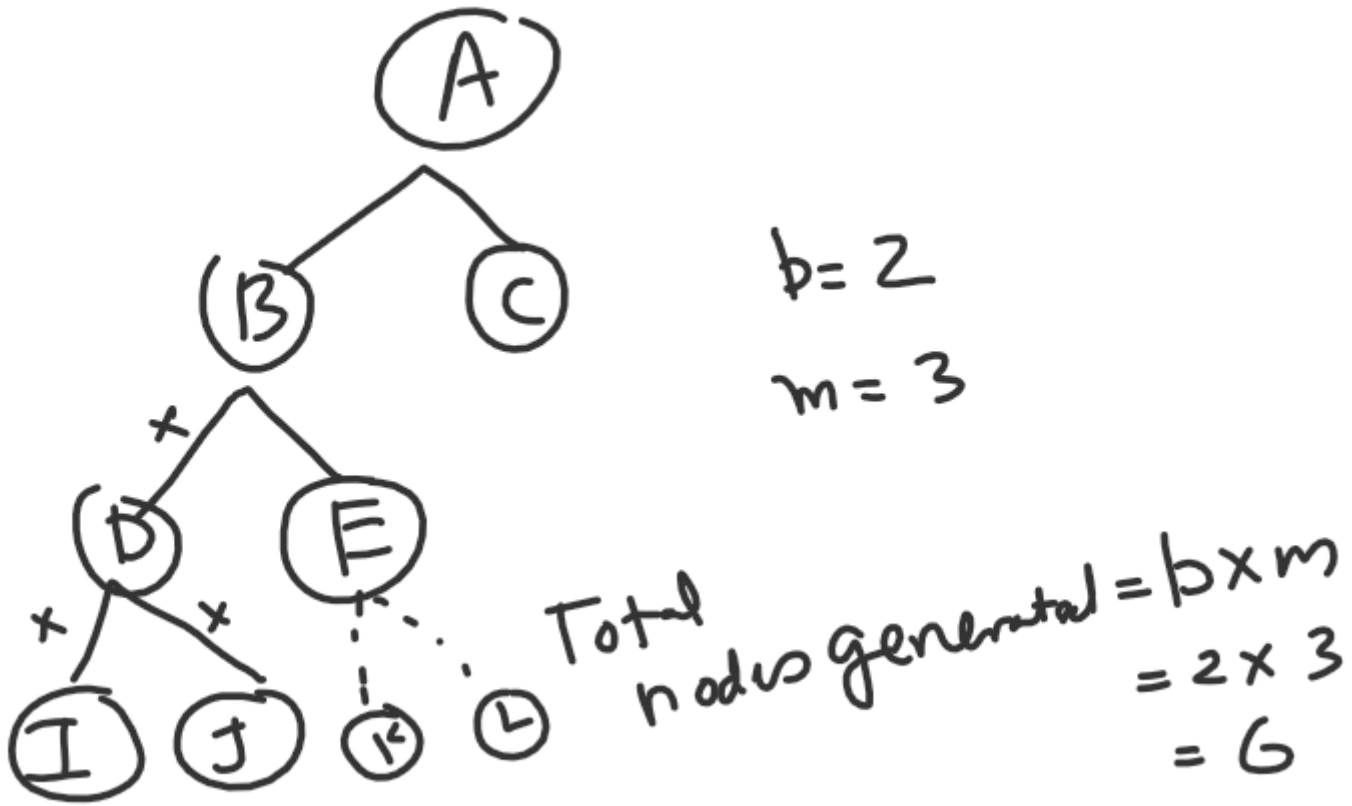
- If there is a finite solution BFS guarantees it.
- If there is more than one solution then solution that requires minimum number of steps will be found first.
- BFS does not get trapped by exploring wrong path.
- It is good when number of branches in a node are less.

Explain BFS & DFS with advantage and drawback?(Univ Q)

Drawback BFS:

- Time complexity is $O(b^{d+1} - b) = O(b^{d+1})$ and space complexity is $O(b^d)$
- It need more memory space since all the nodes of tree so far generated need to be stored.
- It is wasteful when all the goal state are at same level.

DFS



Explain BFS & DFS with advantage and drawback?(Univ Q)

DFS:

- It expands deepest node first and reaches the deepest level(leaf node) and then backtrack to next shallow node that has unexplored successors.
- As the nodes are expanded they are dropped from memory and using backup next deepest unexplored nodes are searched.
- This can be implemented using LIFO queue (stack) or recursive function(calling its child recursively)
- If b is branching factor and m is maximum depth then total number of nodes generated is: bm
- Space complexity is O(bm) and time complexity is $O(b^m)$
- Extension of DFS is Depth limit search where we can limit the search by considering tree till certain level only.

Explain BFS & DFS with advantage and drawback?(Univ Q)

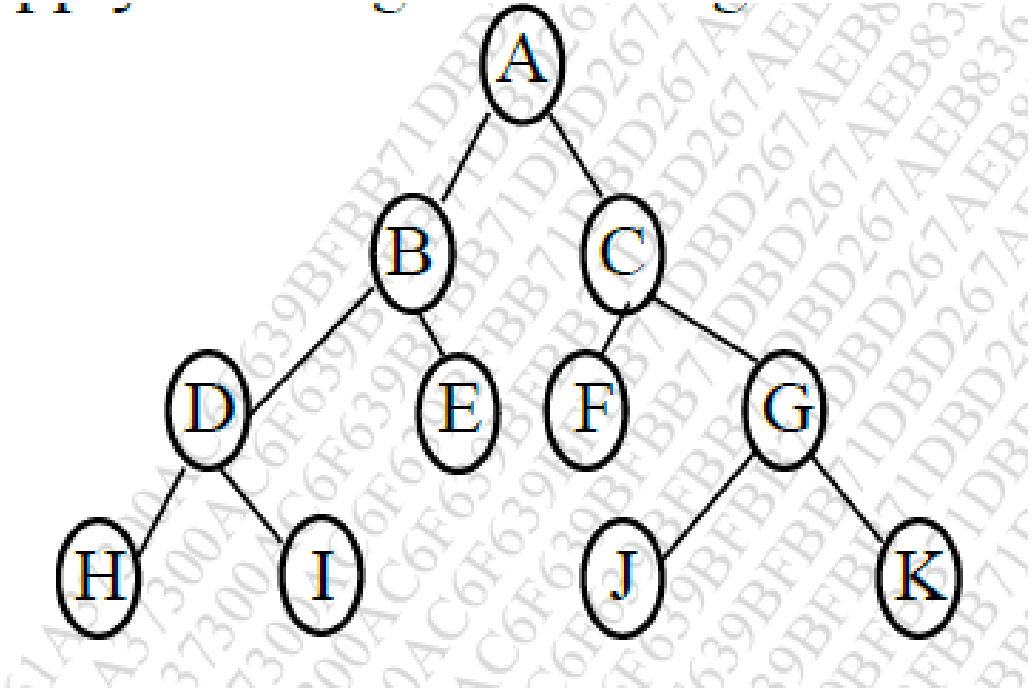
Advantage DFS:

- DFS need less memory since only nodes to current path are stored.
- By chance DFS may find a solution without searching much of search tree this will save good amount of time.

Drawback DFS:

- DFS can be trapped by making wrong choice and get stuck down very long path
- It does not guarantee minimal solution.

Apply DFS on the given tree and write sequence of node expansion(May18)



DLS Special case of DFS

Depth Limit search:

- The problem of unbounded tree is solved by Depth limit search as here node at depth (L) are treated as they have no successor
- It may lead to incomplete solution if we choose L<<<d(very small such that it has no solution in it)

DLS

Uniform cost search.

- The best solution found by BFS may not be least expensive in term of path cost.
- So uniform cost path searches which node to expand by looking at the path cost of each node and one with least cost is expanded first.
- It may not look very efficient but it gives weightage to low cost nodes.
- It do not take care of number of step a path has but only focus on their total cost.
- It give optimal solution as it explore lower path cost first.
- **It is equivalent to BFS it path cost is uniform or cost path is not available.**

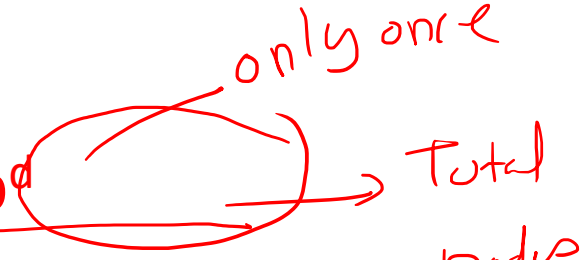
Iterative Deeping depth First Search(IDDFS)(Univ Q) OR IDS

- This technique uses BFS but in combination with DFS to find the best depth limit.
- It is Depth Limit Search(DLS) with iteration
- Depth limit start from 0(only root node) then goes to 1(only up to first level DFS) then goes to 2(only up to two level DFS) and so on.
- It has benefits of both DFS(less memory need) and BFS(give optimal solution).
- It explores all nodes present in current limit(BFS) before going to next limit(DFS)

Iterative Deeping depth First Search(IDDFS)(Univ Q)

- Here the nodes at the bottom level (depth d) are generated(scanned) once those next(parent) to bottom are generated twice and so on up to the root node generated d times.
- Total number of nodes generated are:

  $N(IDS)=d.\ b+(d-1)\ b^2+(d-2)b^3+\ldots\ldots+(1)\ .\ b^d$



→ BFS

→ DFS

- Time complexity=$O(b^d)$
- Space complexity=O(bd)
- It may look like IDS is waste because state are generated multiple time but its not true because search tree with same branching factor most of the nodes are at bottom level

Iterative Deeping depth First Search(IDDFS)(Univ Q)

- It is preferred when there is a large search space and depth of solution is not known.

Bidirectional Search(Univ)

- Here the idea is to run two simultaneous search one from initial state and other backward from the goal and stopping when two searches meet in the middle.
- The benefit of this searching technique is its search Time complexity $=O(b^{d/2} + b^{d/2})$ which is better than time complexity compared to IDDFS give by $O(b^d)$
- But space complexity is issue because at a time one of the search tree must be kept in memory so space complexity is $O(b^{d/2})$ while IDDFS has space complexity as O(bd).

Bidirectional
$b^{d/2} + b^{d/2}$
$= 3^2 + 3^2$

18

Suppose $b=3$ $d=4$ ⇒ IDS ⇒
$TC = 3^4$
$= 81$

Bidirectional Search(Univ)

It has severe limitations

- predecessors must be generated, which is not always possible (generating root node from child node)
- search must be coordinated between the two searches i.e search starting from root and search starting from goal must meet somewhere in middle
- Track of both search need to be maintained in memory i.e how both search are progressing need to be monitored.

Example: In chess game from goal state i.e “checkmate” its difficult to reach middle state here backward search need to construct all possible move to reach its predecessor.

Comparing Uninformed search technique?(Univ Q)

Criteria	BFS	DFS	DLS	IDS	Bidirectional Seach
Complete	Yes, if b is finite	No	No	Yes, if b is finite	Yes, if b is finite and both direction use BFS
Time Complexity	$O(b^{d+1})$ d is depth of shallowest solution	$O(b^m)$ m is max depth of search tree	$O(b^l)$ l is depth limit	$O(b^d)$ D is depth of tree in particular iteration.	$O(b^{d/2})$
Space complexity	$O(b^d)$	O(bm)	O(bl)	O(bd)	$O(b^{d/2})$
Optimal	Yes if step cost are identical	No	No	Yes if step cost are identical	Yes if step cost are identical & both direction use BFS