VASANTDADA PATIL PRATISHTHAN'S
COLLEGE OF ENGINEERING & VISUAL ARTS

Anurag. Choeghe
VU1F20210007
Page No.
TE-Comps-A
Batch-A

# Assignment 03

Q1] Compare search strategies in terms of the four evaluation criteria
→ There are variety of problem solving methods & algorithms available in AI. They can be compared or evaluated using the following four criteria

1) Completeness: If an algorithm is ~~[illegible]~~ able to produce the solution, if one exists, then it satisfies completeness
2) Optimality: If the solution produced is the minimum cost solution, ~~the~~ ~~the~~ ~~number of nodes generated during the~~ ~~search~~ the algo is said to be optimal
3) Time complexity: It depends on the time taken to generate solution. It is the number of nodes generated during the search
4) Space complexity: Memory required to store the generated nodes while performing the search.

Complexity of algorithms is expressed in terms of three quantities
i) b: Branching factor represents maximum number of successors a node can have the search tree
ii) d: Stands for depth of the shallowest goal node
iii) m: It is the maximum depth of any path in S.T

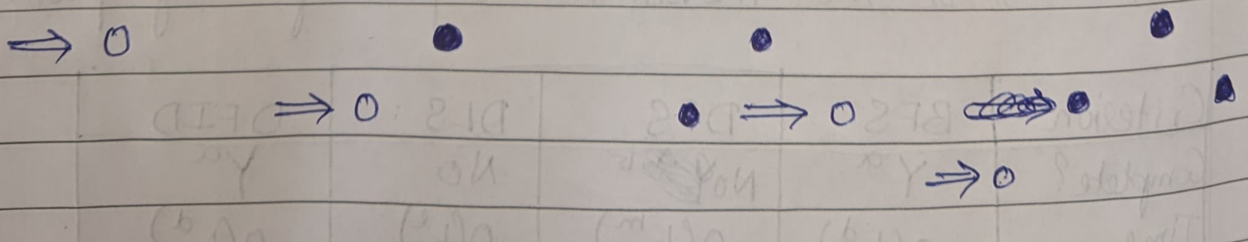
| Criterion | BFS | DFS | DLS | DFID |
|---|---|---|---|---|
| Complete? | Y$^{a}$ | No$^{b}$ | No | Y$^{a}$ |
| Time | $O(b^{d})$ | $O(b^{m})$ | $O(b^{l})$ | $O(b^{d})$ |
| Space | $O(b^{d})$ | $O(bm)$ | $O(bl)$ | $O(bd)$ |
| Optimal? | Yes$^{c}$ | No | No | Yes$^{c}$ |

Y$^{a}$ is complete if b is finite.
$l$ = depth limit
Yes$^{c}$ is optimal if step cost are

Q2] What are different uninformed search strategies? Explain any one with suitable example

→ Uninformed search strategies have only information about what is the initial/start state & final/end state along with problem definition. These techniques can generate successor states & can distinguish a goal state from non-goa
There are four different uninformed search strategies
i) Breadth First Search (BFS)
ii) Depth First Search (DFS)
iii) Depth Limited Search (DLS)
iv) Depth First Iterative Deepening (DFID)

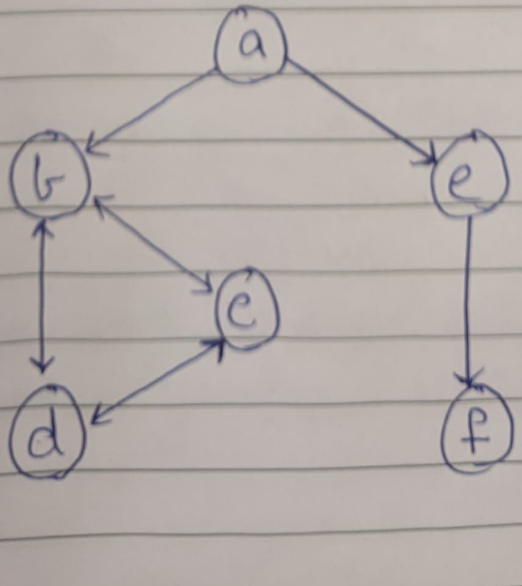
BFS : Breadth First Search is applied on the tree & expands breadth wise. The rootnode is expanded first, then all the successors of the root node are expanded, then their successors, and so on. In turn, all the nodes at a particular depth in the Search Tree (S.T) are expanded first & then the search will be proceeded for the next level node expansion. Thus, the shallowest unexpanded node will be chosen for expansion.
The search process of BFS is illustrated below



○ → Unexpanded nodes / Not visited nodes
● → Expanded nodes / Visited nodes

Q4] What is meant by search algorithm completeness & optimal
Discuss for BFS strategies
→ Completeness: If an algorithm is complete, it means that if at least one solution exists then the algorithm is guaranteed find a solution in a finite amount of time
Optimality: If a search algorithm is optimal, then when it finds a solution it finds the 'best' solution.
In BFS we use a FIFO queue for the fringe. Because of which the newly inserted nodes in the fringe will automatically be placed after their parents Thus, children nodes, which are deeper than their parents, go to the back of the queue, & old nodes, which are shallower, get expanded first. This makes BFS more complete & optimal when compared to DFS. If there are more than one solⁿ for a given problem, then the BFS will provide solⁿ that requires minimal cost or steps.

Q5] Consider the search problem represented in Figure 1, where 'a' is the start node & 'f' is the goal node. Would you prefer DFS or BFS for this problem? Why?



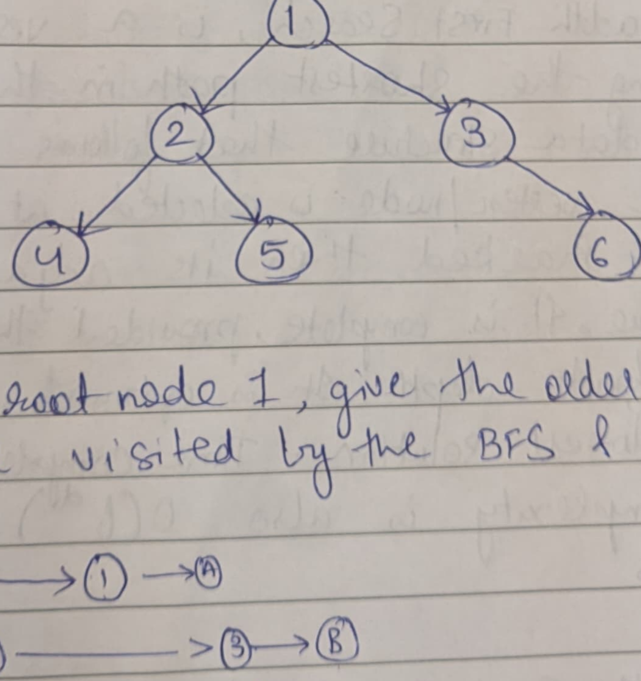
→ If we were just running vanilla DFS (no pruning or loop checking) the we would prefer BFS, because DFS could get stuck in an infinite loop. ~~whereas~~ ~~if it exp~~ Note that BFS is sensitive to the ordering of nodes. If it explores to the left first it will get stuck in the loop, whereas if explores to right it will find goal very quickly. Vanilla DFS explores a → b → d → b & keeps oscillating between b & d. Whereas BFS first adds a to FIFO queue the a → b & a → e. It expands 'ab' & add 'abd' and 'abc' to FIFO queue. Then path ae is expanded adding 'aef'. Both abd & abc are expanded but rejected as aef cannot be expanded & hence reaching goal state.

Q6] Briefly describe BFS & DFS
BFS, Breadth First Search, is a vertex based technique for finding the shortest path in the graph. It uses a queue data structure that follows first in first out. In BFS, one vertex/node is selected at a time when it is visited & marked then its adjacent are visited & stored in the queue. It is complete, provided the shallowest goal node is at some finite depth. It is optimal, as it always finds the shallowest solution. Time complexity is $O(b^d)$ & space complexity is also $O(b^d)$. However it is slower than DFS

DFS, Depth First Search, is an edge-based technique i.e the deepest node in the current branch of the S.T is expanded. It uses LIFO fringe i.e stack data structure The most recently generated node, which is on the top in the fringe, is chosen first for expansion. As the node is expanded, it is dropped from the fringe & its successor

are added. So when there are no more successors to ad
to the fringe, the search 'back-tracks' to the next
deepest node until that is explored. It can be implemented
using iteration & recursion. It is only complete if m
is finite. Since it does not guaranteed a solution it is
not optimal. A DFS may generate all of the $O(b^m)$
nodes in the S.T, where m is the maximum depth of
any node; this can be much greater than size of
state space. For a ST with branching factor b & max
depth m, DFS requires storage of only $O(b^m)$
nodes, as at a time only one branch gets explored

Q7] Consider the following graph



i] Starting at root node 1, give the order in which the
nodes will be visited by the BFS & DFS

BFS : $\longrightarrow$ 1 $\rightarrow$ A

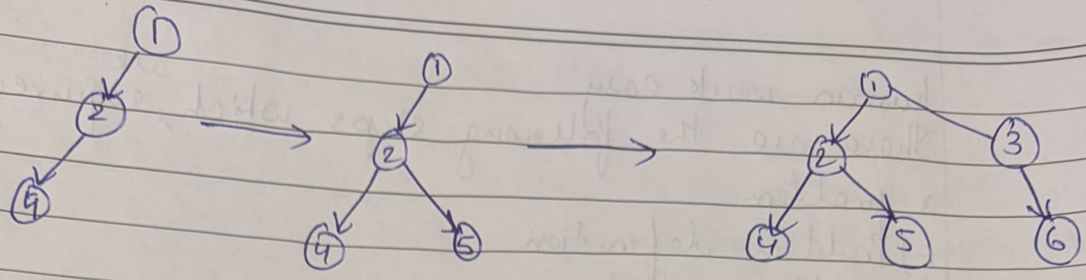
A $\longrightarrow$ 2 $\longrightarrow$ 3 $\rightarrow$ B

B $\longrightarrow$ 4 $\longrightarrow$ 5 $\longrightarrow$ 6

∴ 1 → 2 → 3 → 4 → 5 → 6

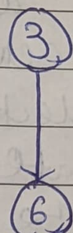
DFS : Since it explores all children nodes that are
expendable & not visited in a ~~[illegible]~~ ~~queue~~ stack

∴



1 → 2 → 4 → 5 → 3 → 6

ii] Starting at root node 3, give the order in which the nodes will be visited by the BFS & DFS algo.
→ Since 3 is root node, all branches of the S.T that are not a subsequent branch of '3' can be pruned
∴ the new S.T that is generated after pruning is



Since node '3' has only a singular node both BFS & DFS will have same output i.e 3 → 6

Q8] List & define components of problem solving
On the basis of the problem & their working domain, different types of problem-solving agents defined & ~~use~~ use at an atomic level without any internal state visible with problem solving algorithm. So we can say that problem solving is a part of AI that encompasses a number of techniques such as tree, B-tree, heuristic algo to solve problem. The problem of AI is directly associated with the nature of humans & their activities. So we need a number of ~~a~~ finite steps which ~~require~~ makes

human work easy
These are the following steps which are required to solve a problem
i) Problem definition
ii) Problem analysis
iii) Knowledge Representation
iv) Problem solving
Components to formulate associated problem
• Initial state : This state ~~[illegible]~~ is required which stores in the model/configuration of a problems start.
• Action : This stage of problem formulation works with function with a specific class taken from the initial state & all possible actions done in this stage
• Transition : It integrates the actual action done by the previous action state & collects the final state to forward it to their next state
• Goal test : It determines if the specified goal achieved by integrated transition model or not, whenever the goal achieves stop the action & forward into the next stage to determine the cost to achieve the goal
• Path costing : This component of problem-solving numerical assigned what will be the cost to achieve the goal. It requires all hardware, Software & human working cost.