CO-2

Search Algorithm in AI

→ **Search algorithms** are one of the most important areas of Artificial Intelligence. **Search techniques** are universal problem-solving methods in AI.

→ **Artificial Intelligence** is the study of building agents that act rationally. Most of the time, these agents perform some kind of search algorithm in the background in order to achieve their tasks.

→ **Problem solving agents or rational agents** in AI mostly used these search strategies or algorithms to solve a specific problem and provide the best result/solution. Problem-solving agents are the goal-based agents and use atomic representation.

→ A simple **problem solving agent** in a goal-based agent. It decides what to do by finding different possible sequences of actions that lead to desirable states, and then choosing the best sequence.

* **Reflex Agents** do not consider the future consequences of their actions. Choose action based on current percept.

* **Planning Agents** searching এর সাথে খুবই related. কোনো action নেওয়ার পর next এ action এর consequences গুলো কি হতে পারে সেগুলো বের করে future consequences গুলো। planning agent এইভাবেই কাজ করে। So তার future consequences দেখে how the world would be. তারপরে agents কোনো action নেওয়ার আগে how the world would be. তারপরে agents কোনো action নিলে, তাহলে environment এ state এ change কি হতে পারে, কোনো action নিলে, আগে থেকেই plan করে এবং তার consequences গুলো consider করে।

Note: Uninformed search এ all possible solutions এ কোন্ কোন্ node visite করা যাবে। node এত বেশি হলে তত slow এবং কাজ করা, মানে cost and time বেড়ে যাবে।

*Search Algorithm Terminologies:

→**Search**: Searching is a step by step procedure to solve a search-problem in a given search space.

→**Search Space**: Search space represents a set of possible solutions, which a system may have.

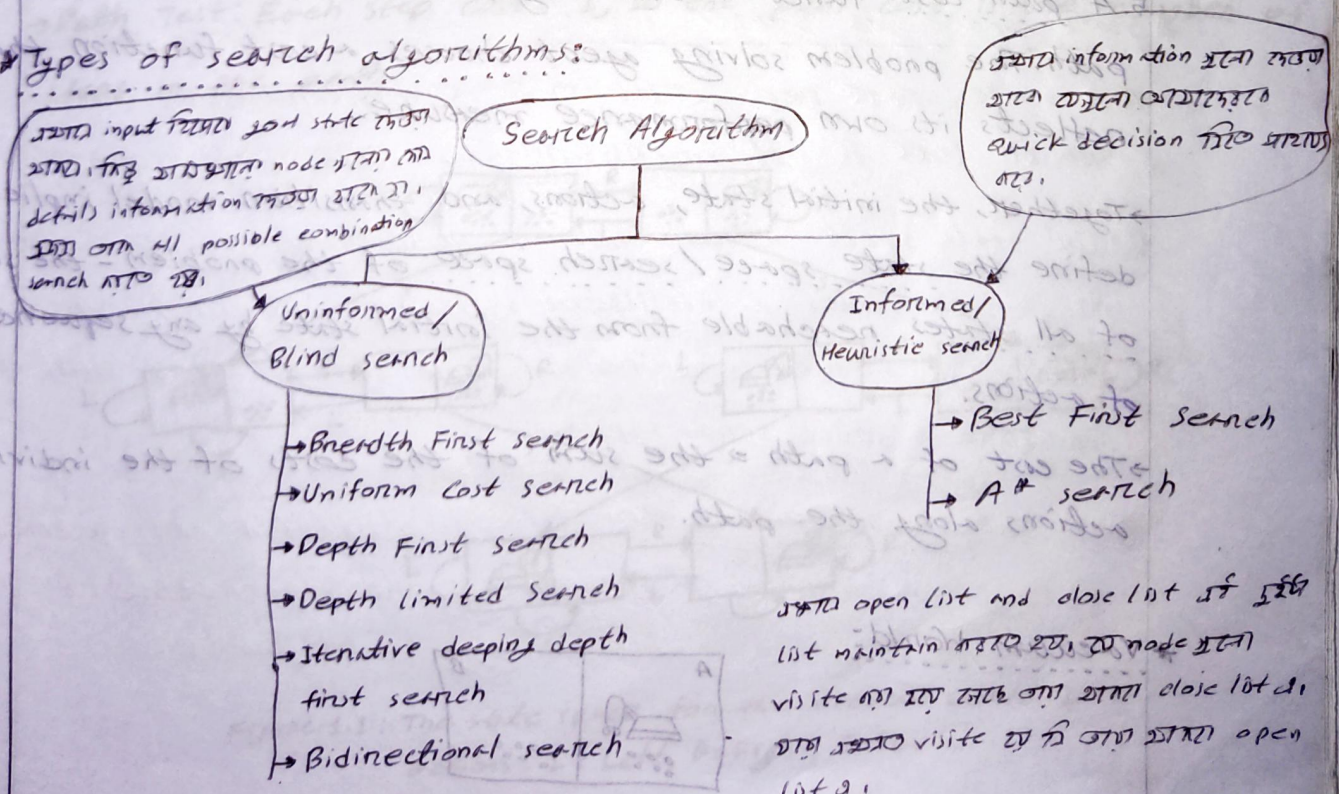
→**Start ~~Space~~ State**: It is a state from where agent begins the search.

→**Goal Test**: It is a function which observe the current state and returns whether the goal state is achieved or not.

→**Search Tree**: A tree representation of search problem is called search tree. The root of the search tree is the root node which is corresponding to the initial state.

→ Actions, Transition Model, Path cost, solution, optimal solution.

*Types of search algorithms:



* A problem can be defined formally by five components:

1. The initial state that the agent starts in. For example, the initial state for our agent in Romania might be described as In(Arad).

2. A description of the possible actions available to the agent. For example, from the state In(Arad), the applicable actions are {Go(Sibiu), Go(Timisoara), Go(Zerind)}.

3. Transition model: A description of what each action does RESULT(In(Arad), Go(Zerind)) = In(Zerind).

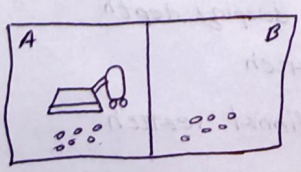
4. The goal test, which determines whether a given state is a goal state. The agent's goal in Romania is the singleton set {In(Bucharest)}.

5. A path cost function that asigns a numeric cost of each path. The problem solving agent chooses a cost function that reflects its own performance mesure.

→ Together, the initial state, actions, and transition model implicitly define the state space / search space of the problem - the set of all states reachable from the initial state by any sequence of actions.

→ The cost of a path = the sum of the costs of the individual actions along the path.

* Vacuum World:



→States/State space: The state is determined by both the agent location and the dirt locations. The agent is in one of two locations, each of which might or might not contain dirt. Thus, there are $2 \times (2 \times 2) = 8$ possible world states. A larger environment with n locations has $n \times 2^n$ states.

→Initial state: Any state can be designated as the initial state.

→Actions: In this simple environment, each state has just three actions: Left, Right, and Suck. Larger environments might also include Up and Down.

→Transition Model: The actions have their expected effects, except that moving Left in the leftmost square, moving Right in the rightmost square, and sucking in a clean square have no effect. The complete state space is shown in Figure-1.1.

→Goal Test: This checks whether all the squares are clean.

→Path Test: Each step costs 1, so the path cost is the number of steps in the path.

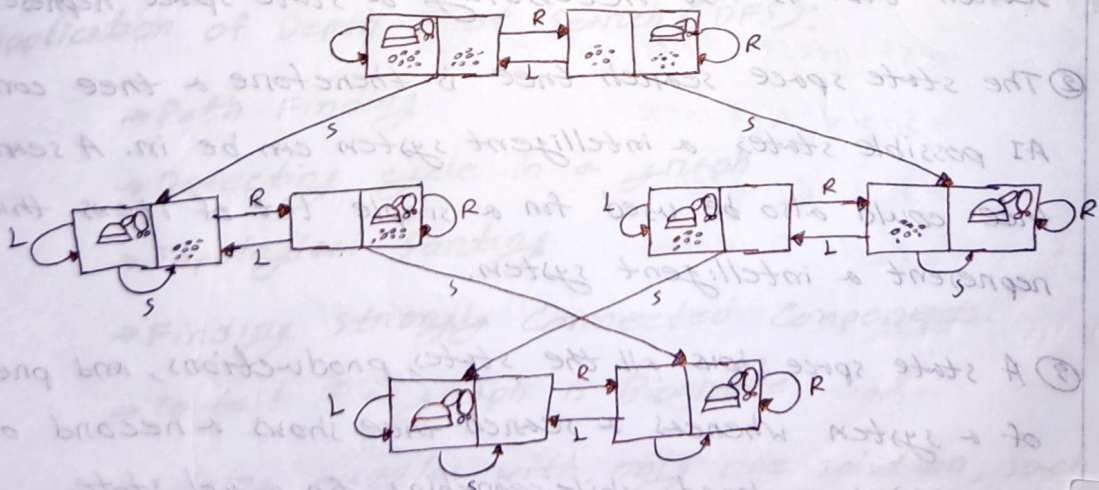
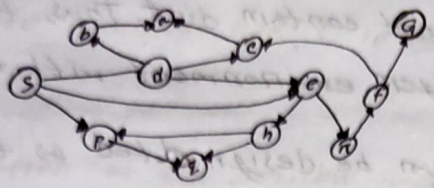


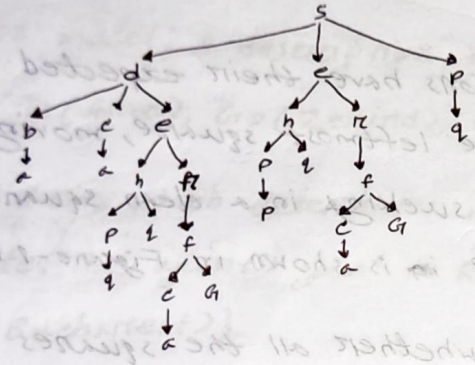
Figure-1.1: The state space for the vacuum world. Links denote actions: L = Left, R = Right, S = Suck.

* A state space forms a graph in which the nodes are states and the arcs between nodes are actions.



In a search graph, each state occurs only once!

* A tree representation of search problem is called search tree.



यहाँ node से node जोड़ा जाता है।

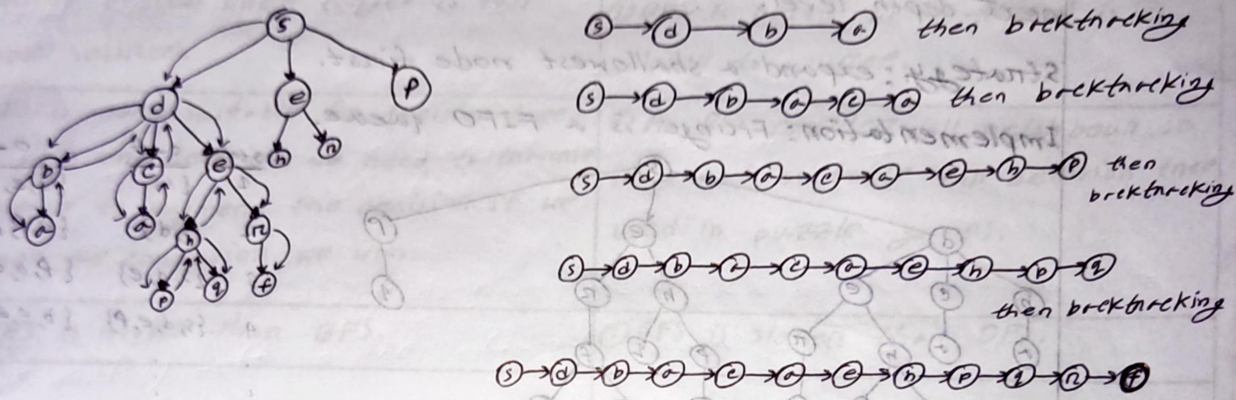
* State Space vs Search Tree:

① A state space representation is a search tree, but a search tree is not necessarily a state space representation.

② The state space search tree is therefore a tree containing AI possible states a intelligent system can be in. A search tree could also be used for a simple list of items that don't represent a intelligent system.

③ A state space shows all the states, productions, and preconditions of a system whereas a search tree shows a record of state transitions explored while searching for a goal state.

* **Depth First Search:** Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node and explores as far as possible along each branch before backtracking.



→Note: কোন একটি node হতে একদম depth এ চলে যেতে হবে। তারপর যদি goal test না হলে then backtracking করতে হবে।

→Time complexity : $O(b^n)$.

→Space complexity : $O(bn)$.

→ Inplementation : fringe : LIFO (Stack).

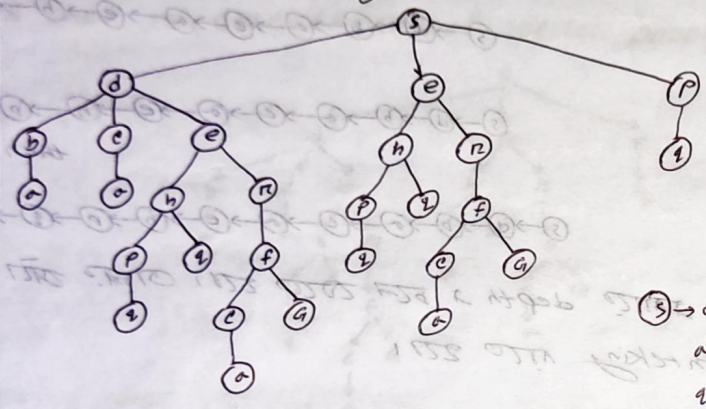
→Application of Depth-First Search (DFS):

- →Path Finding.
- →Detecting cycle in a graph
- →Topological Sorting.
- →Finding strongly Connected Components.
- →To test if a graph is Bipartite.
- →Solving puzzles with only one solution, such as 'mazes'.

* Breadth-first search (BFS): BFS is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root, and explores all of the neighbour nodes at the present ~~graph~~ depth prior to moving on to the nodes at the next depth level.

Strategy: expand a shallowest node first.

Implementation: Fringe is a FIFO queue.



step	e-f	o-f
1	{s}	{d, e, p}
2	{s, d}	{e, p, b, c, e}
3	{s, d, e}	{p, b, c, e, h, n}
4	{s, d, e, p}	{b, c, e, h, n, q}
⋮	⋮	⋮

(S) → d → e → p → b → c → e → h → n → q →
a → a → h → n → p → q → f → p → q → f →
q → c → (G)

→ Time complexity: $O(b^s)$.

→ optimal: Only if costs are all 1 (more on costs later).

* Application of Breadth-First Search:

→ Shortest Path in a Graph

→ Social Network

→ Web Crawler

→ Cycle Detector

→ To test if a graph is Bipartite

→ Broad casting in a Network

→ Ford-Fulkenson Algorithm.

* DFS vs BFS:

DFS	BFS
① DFS stands for Depth First search.	① BFS stands for Breadth First search.
② DFS uses stack data structure for finding the shortest path.	② BFS uses Queue data structure for finding the shortest path.
③ DFS is better when target is far from source.	③ BFS is better when target is closer to source.
④ DFS is more suitable for decision tree. As with one decision, we need to traverse further to augment the decision. If we reach the conclusion, we won.	④ As BFS consider all neighbour so it is not suitable for decision tree used in puzzle games.
⑤ DFS is faster than BFS.	⑤ BFS is slower than DFS.

* If we are facing an infinitely deep graph with a high branching factor, we can use Iterative Deepening (IDS) algorithm.

* Iterative Deepening Search (IDS):

3. (a) Yes, I can. Intelligent means it does things based on reasoning, while rational means it doest the best action for a given situation. Intelligent agent is a program that can make decisions on perform a service based on its environment, user input and experience. Whereas, rational agent is an agent which has clear preferences models uncertainty, and acts in a way to maximize its performances measure which all possible action.

(b) Here, my registration ID: 1620 1043

$\therefore i = (43 \% 3) + 1$

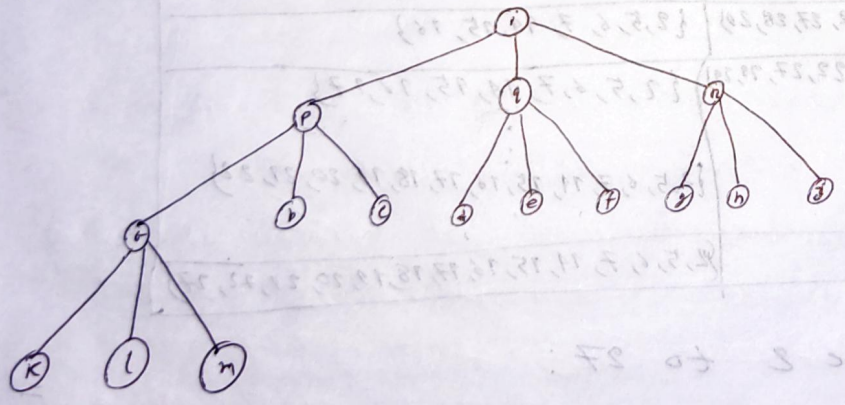
$= 1 + 1 = 2$

$\therefore p = 3i - 1 = 3 * 2 - 1 = 6 - 1 = 5$

$\therefore q = 3i = 3 * 2 = 6$

$\therefore n = 3i + 1 = 3 * 2 + 1 = 6 + 1 = 7$

(i) Search tree from root node value i to leaf node value (i+26).



$a = 3p - 1 = 3 * 5 - 1 = 14$

$b = 3p = 3 * 5 = 15$

$c = 3p + 1 = 3 * 5 + 1 = 16$

$d = 3q - 1 = 3 * 6 - 1 = 17$

$e = 3q = 3 * 6 = 18$

$f = 3q + 1 = 3 * 6 + 1 = 19$

$g = 3n - 1 = 3 * 7 - 1 = 20$

$h = 3n = 3 * 7 = 21$

$j = 3n + 1 = 3 * 7 + 1 = 22$

$k = 3a - 1 = 3 * 14 - 1 = 27$

$l = 3a = 3 * 14 = 28$

$m = 3n + 1 = 3 * 14 + 1 = 29$

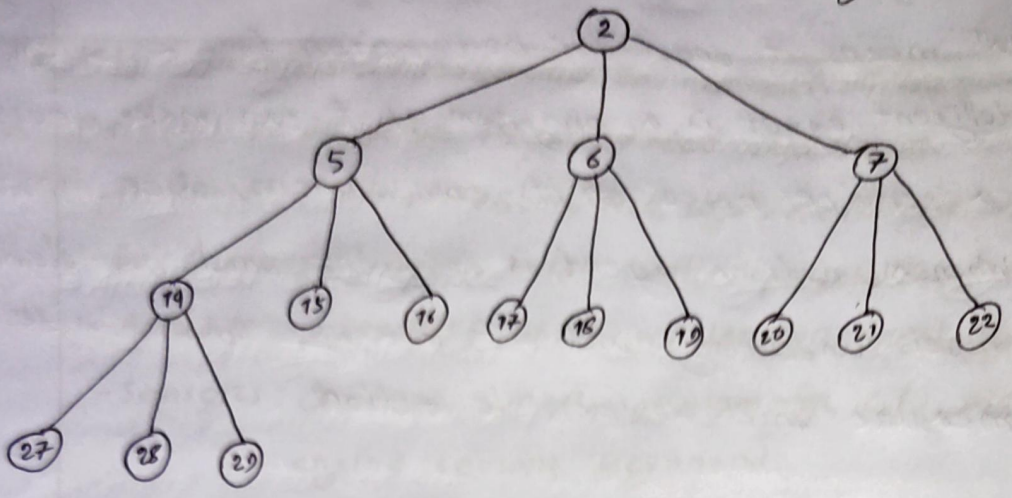
Hence, leaf node value $= i + 26 = 2 + 26$

$= 28$

Goal node value $= i + 25 = 2 + 25$

$= 27$

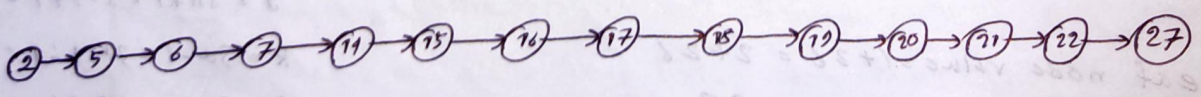
Now, put the value on the search tree and get a new search tree.



(ii) For Breadth-First search (BFS):

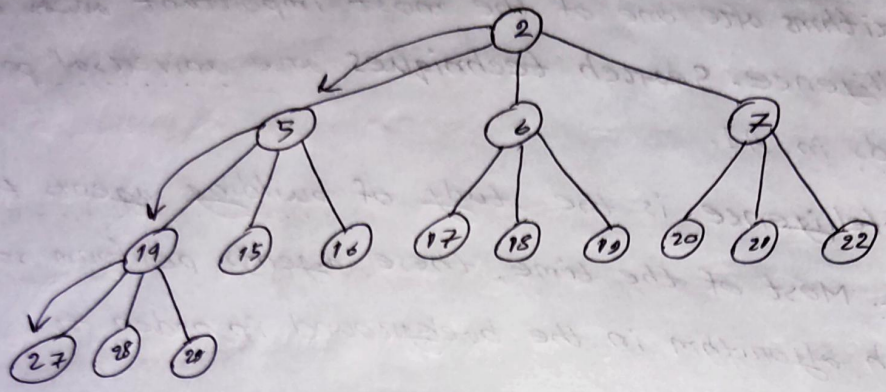
Iteration	Open fringe	close fringe
1	{2}	{ }
2	{5, 6, 7}	{2}
3	{6, 7, 14, 15, 16}	{2, 5}
4	{7, 14, 15, 16, 17, 18, 19}	{2, 5, 6}
5	{14, 15, 16, 17, 18, 19, 20, 21, 22}	{2, 5, 6, 7}
6	{15, 16, 17, 18, 19, 20, 21, 22, 27, 28, 29}	{2, 5, 6, 7, 14}
7	{16, 17, 18, 19, 20, 21, 22, 27, 28, 29}	{2, 5, 6, 7, 14, 15}
8	{17, 18, 19, 20, 21, 22, 27, 28, 29}	{2, 5, 6, 7, 14, 15, 16}
9 ⋮ 14	{18, 19, 20, 21, 22, 27, 28, 29} ⋮ {27, 28, 29}	{2, 5, 6, 7, 14, 15, 16, 17} ⋮ {2, 5, 6, 7, 14, 15, 16, 17, 18, 19, 20, 21, 22}
15	{28, 29}	{2, 5, 6, 7, 14, 15, 16, 17, 18, 19, 20, 21, 22, 27}

So, the path from node 2 to 27:



(iii) For Iterative Deepening Search (IDS):

Let, the maximum search depth = 4



So, the path from node 2 to 27: