Unit - I

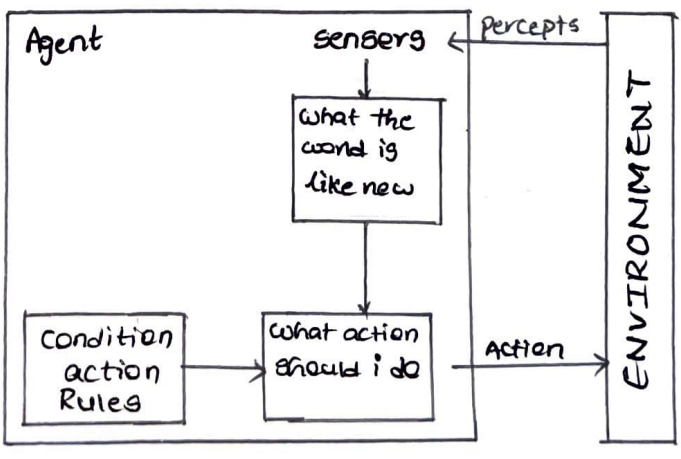
1. Types of agent:

Agents can be grouped into five classes based on their degree of perceived intelligence and capability. All these agents can improve their performance and generate better action over the time. These are given below.

1. Smart reflex agents
2. Model based Reflex agent
3. Goal based Agent
4. ultility based Agent
5. Learning agent

1. Simple Reflex agent:

* These are the simplest agents.
* These agents take decisions on the basis of the current percepts and ignore the rest of percepts history.
* The agents suceed at fully observable environment
* It doesn't consider any percept history which taking decision and action process.
* problems: → Limited intelligence
  → too big to generate
  → too big to store
  → Not adaptive to change in environment



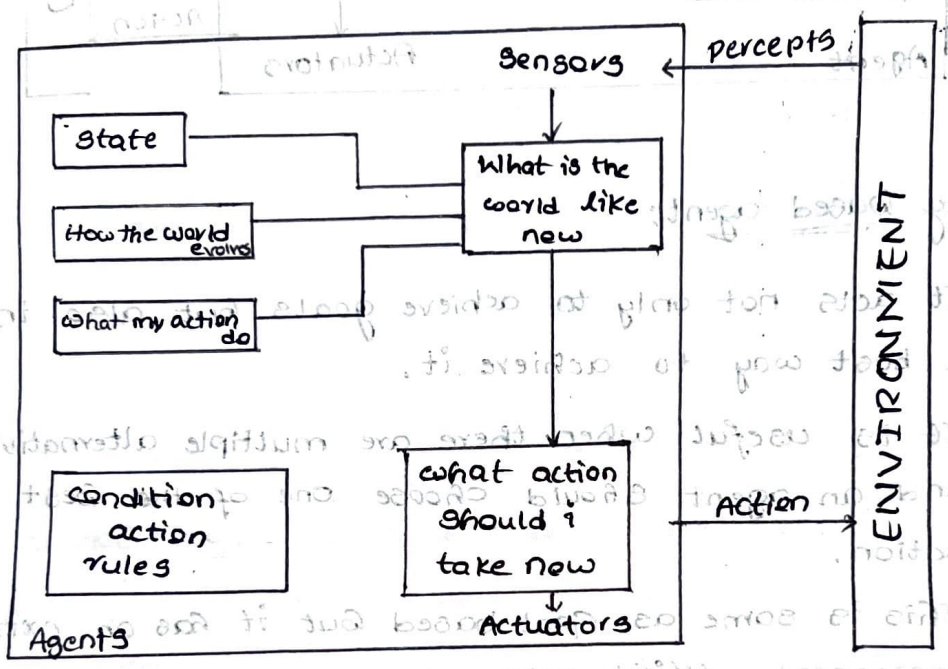
2. Model Based reflex agent:

The model based reflex agent partially coorks in observable environment and track the situation.

* A model based agent has two important factors:

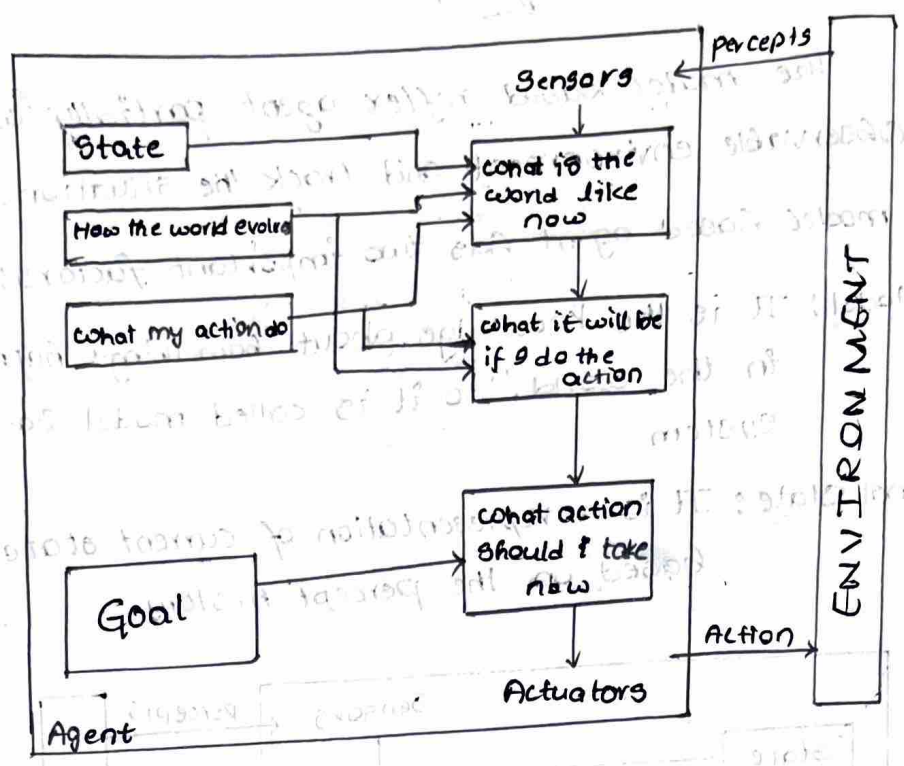
Model: It is the knowledge about "how things happen in the coorld," so it is called model based system

Internal state: It is a representation of current state based on the percept history.



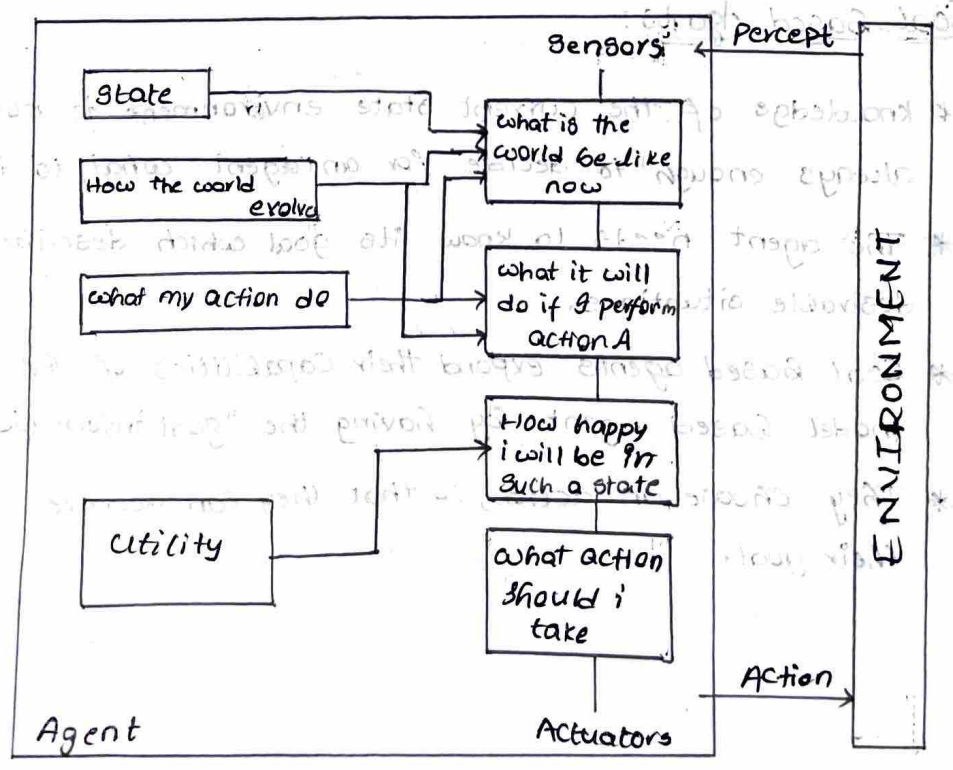
3. Goal-Based Agents:

* knowledge of the current state environment is not always enough to decide for an agent what to do.
* The agent needs to know its goal which describes desirable situations.
* Goal based agents expand their capabilities of the model based agent by having the "goal information".
* They choose an action, so that they can achieve their goal.



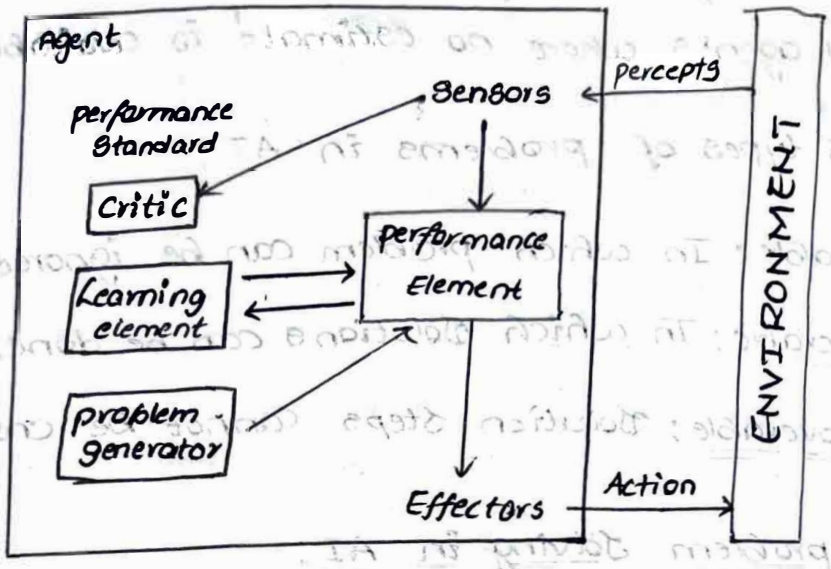
4. Utility based agent:

* It acts not only to achieve goals but also in a best way to achieve it.
* It is useful when there are multiple alternatives and an agent should choose one of the best action.
* This is same as Goal based but it has an extra component utility measurement.



5. Learning agent:

* It is an AI type of agent which can learn from its past experiences or it has learning capabilities.
* It starts to act with basic knowledge and then able to act and adapt automatically through learning.



Problem Solving agents:

An agent that tries to come up with a sequence of actions that will bring the environment into desired state. Such agents is called as problem solving agents.

* Informed agents in which the agents can estimate how far it is from the goal.
* uniformed agents where no estimate is available.

There are 3 types of problems in AI:

1. Ignorable: In which problem can be ignored.
2. Recoverable: In which solutions can be done.
3. Irrecoverable: Solution steps cannot be undone.

Steps of problem solving in AI:

* The problem of AI is directly associated with the nature of humans and their activities.
* So we need a finite no of steps to solve a problem

1. problem definition
2. problem analysis
3. knowledge Representation
4. problem Solving

1. problem definition:

Detailed specification of the problem

2. problem analysis:

Analyze the problem throughly

3. knowledge Representation:

collect the detailed info about the problem and define all possible Techniques

4. problem solving

Select the best technique

* The problem can be solved (or) defined using 5 components:

1. Initial state
2. Action
3. Transition model
4. Goal test
5. path test

1. Initial state:

This state is called an initial state because the problem starts here and tends towards specified goal.

2. Action:

This state main objective is to function with a specific class from initial state and all possible actions done in this stage.

3. Transition:

It integrated the actual action done by the previous stage and collects the final stage to forward into the next stage.

4. Goal test:

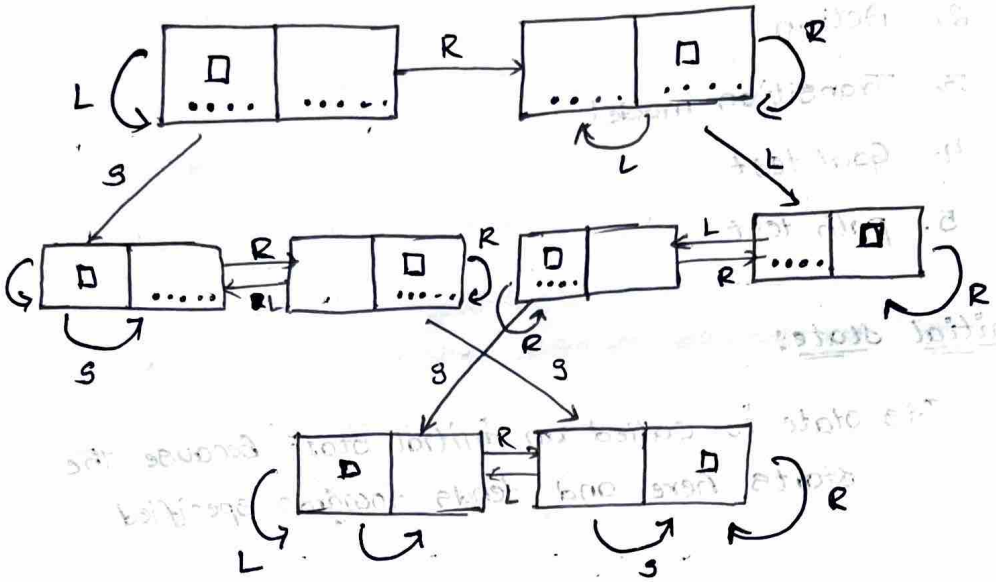
This stage determines that specific goal achieved by the integrated transition model or not.

5. path test:

* It determines the total cost required to achieve a particular goal.
* It includes hardware, software and human work cost.

Ex: 1) Toy problem:

Statement: There are two dirty square blocks and we need to clean this dirty square box using Vaccum cleaner.



Stages

2 cells, 3 actions

$$g^3 = 2 \times 2 \times 2$$

$$= 8 \text{ states}$$

Initial State: Any state can be designated at Initial state.

Actions: MOV LEFT
MOV RIGHT
SUCK

Transition Model:

SUCK removes the dirt present in the agent cell.

Goal State:

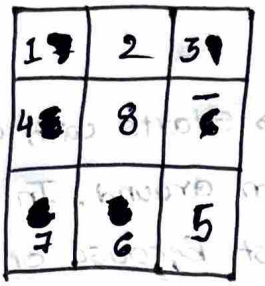
The state in which every cell is cleaned

Action cost: Each action costs

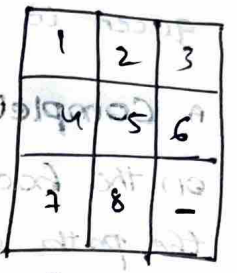
2. The 8-puzzle:

An 8-puzzle consists of 3x3 board with eight numbered tiles and a blank space. A tile adjacent to the blank space can slide into the space. The object is to reach the goal state as shown in figure

Ex:



Start State



Goal state

we can use → { Down, Up, Left, Right }

Initial = { (1,2,3) (4,8,0) (7,6,5) }

= { (1,2,3) (4,8,5) (7,6,0) }

= { (1,2,3) (4,8,5) (7,0,6) }

= { (1,2,3) (4,0,5) (7,8,6) }

= { (1,2,3) (4,5,0) (7,8,6) }

= { (1,2,3) (4,5,6) (7,8,0) }

5

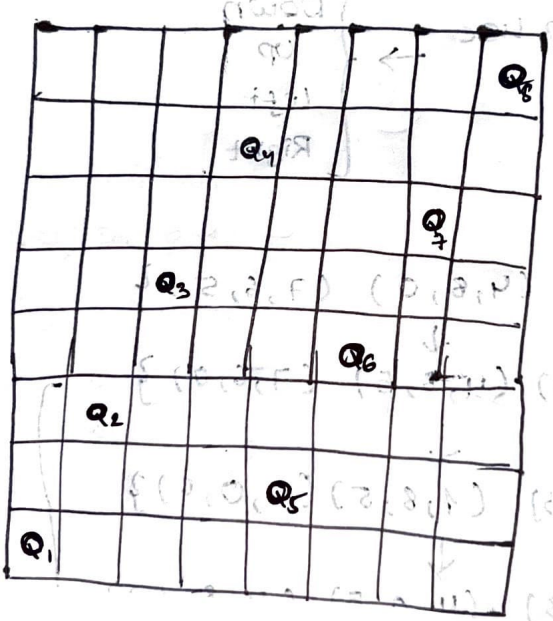
Path cost = "5"

3. 8-queens problem:

The goal of 8 queens problem is to place 8 queens in a chess board such that no queens attacks any other.

An **Incremental** formation → starting with an empty state for 8 queens problem, this means each action adds a queen to the state.

A **Complete-State** formation → starts with all 8-queens on the board and moves them around. In either case the path cost is of no Interest because only the final state counts.



States: Any arrangement of 0-8 queens on board is a state

Initial state: No queens on the board

Successor function: Add a queen to any empty square.

Goal test: 8 queens are on the board, none attacked

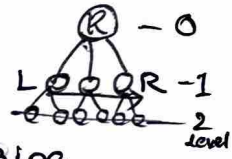
1. Uninformed Search Strategies:

Uninformed search is a class of general purpose search algorithms which operates in brute force way. Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree, so it is also called Blind search

Types of Uniformed Search Algorithms:

1. Breadth-first search
2. Depth-first search
3. Depth limited search
4. Iterative deepening depth first search
5. Uniform cost search
6. Bi directional search

1. Breadth-first search:



* Most common search strategy for traversing a tree or a graph.
* This algorithm searches breadthwise in a tree or graph so it is called breadth-first search
* BFS algo starts searching from the root node of the tree and expands all successor node at the current level before moving to the node of the next level
* BFS algorithm is an example of general graph search algorithm
* BFS implemented using FIFO queue data structure

Advantages:

→ BFS will provide a solution if any solution exists.

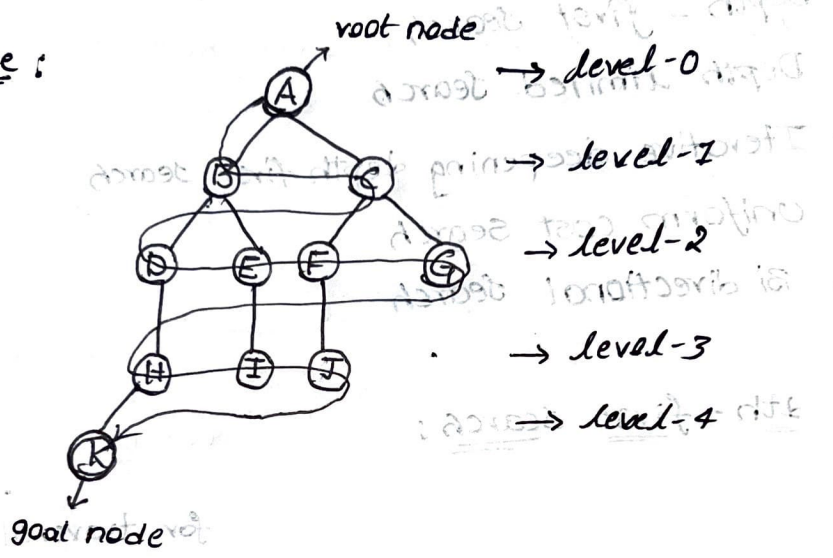
→ If there are more than one solutions for a problem, the BFS will provide minimal solution which requires least no. of steps.

Disadvantages:

→ It requires a lot of memory since each level of tree must be saved into them to expand the next level.

→ BFS need a lots of time if the solution is faraway from the root node.

Example:



Time Complexity:

d = depth of shallowness solution

b is a node at every state

$T(b) = 1 + b^2 + b^3 + \dots + b^d$

$= O(b^d)$

Space Complexity: Memory size which is $O(b^d)$.

Completeness: BFS is Complete.

Optimality: BFS is optimal if path cost is non decreasing function of the depth of the node.

2. Depth-first search:

DFS

BFS

* It is a recursive algorithm for traversing a tree or a graph in data structures.
* It is called DFS because it starts from the root and follows each path to its greatest depth node before moving to the next.
* DFS uses a stack ds for its implementation
* This process is similar to BFS

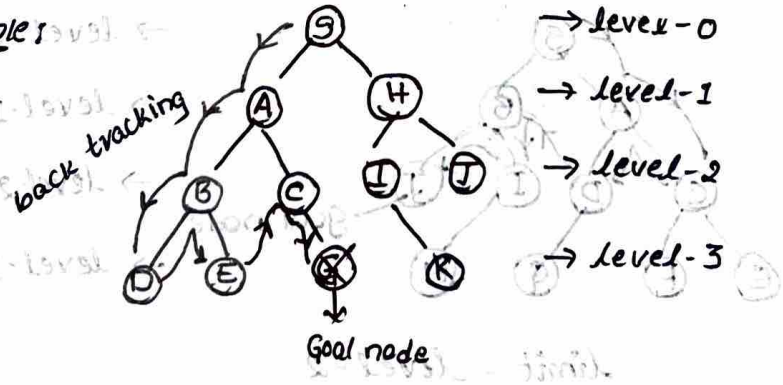
Advantages:

* It requires very less memory.
* It takes less time to reach the goal node compared to BFS algorithm.

Dis-advantages:

* There is a possibility of many states keep re-occuring and there is no guarentee in finding the solution
* DFS algorith goes for deep searching and sometimes it may go to the infinite loop.

Example:



Time complexity: $T(n) = 1+n^2+n^3+\cdots+n^m = O(n^m)$

M → Max depth of any node

Completeness: It is complete within finite state space

Space Complexity: $S(C) = O(bm)$ only store single path

Optimal: DFS is not-optimal as it may generate a large number of steps or high cost to reach to the goal node.

3. Depth-Limited Search Algorithm: (DLS ≃ DFS)
↳ limit

* It is similar to DFS with a predetermined limit.

* Depth limited search can solve the drawback of infinite path in DFS

* The node at the depth limit will treat as it has no successor nodes further.

Two conditions of failure:

1. Standard failure value: It indicates the problem does not have any solution.

2. cut off failure value: It defines no solution for the problem in given depth limit.

Advantages:

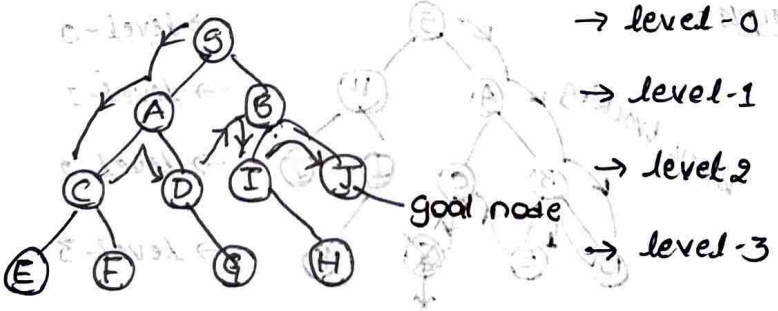
* It is memory efficient

Dis-advantages:

* Incompleteness

* It may not be optimal (problem has more than 1 solution).

Example:



limit = level-2

Time Complexity: TC = O(b^l) l → level

Space complexity: SC = O(b × l)

Optimal: It is not optimal.

4. Iterative deepening depth-first search:

* It is the combination of BFS and DFS algorithms.
* This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found
* This search combines the benefits of BFS and DFS for memory efficiency.
* This search is useful ucs when search space is large and depth of goal node is known.

Advantages:

It combines the benefits of BFS & DFS search algorithm term of fast search and memory efficiency.

Dis-advantages:

The main drawback of IDDFS is that repeats all the work of the previous phase.

Completeness:

Completes branching factor is finite

Time complexity:

Let's suppose 6 is the branching factors & depths then worstcase time complexity is $O(6^d)$.

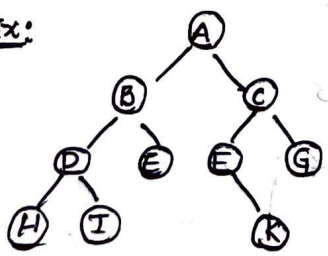
Space Complexity:

$S(c) = O(6d)$

Optimal:

Optimal if path cost is non-decreasing function depth of node.

Ex:



1st iteration ----> A
2nd iteration ----> A, B, C
3rd iteration ----> A, B, D, E, C, F, G
4th iteration ----> A, B, D, H, I, E, C, F, K, G

5. Uniform-cost Search Algorithm (UCS):

* UCS is used for traversing a weighted tree or graph.
* This algorithm comes into play when a different cost is available for each edge.
* The primary goal of the UCS is to find a path to the goal node which has the lowest cummulative cost.
* UCS expands nodes according to their path costs form the root node.
* It can be used to solve any graph / tree where the optimal costs is in demand.
* UCS is implemented by priority queue.
* It gives maximum priority to the lowest cummulative cost.
* UCS equivalents to BTS algorithm if path cost of all edges is the same.

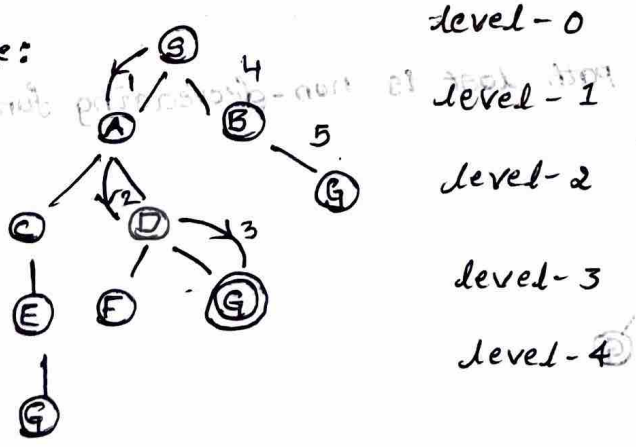
Advantages:

* UCS is optimal because at every state the path with the least cost is choosen.

Dis-advantages:

* It doesnot care about the number of steps involve in searching loop concerned about pathcost.
* Due to which this algorithm may be stuck in an infinite loop

Example:



Time Complexity:

Let C^* is cost of optimal solution and "ε" is each step to get closer to the goal node. Then the number of steps is $= C^*/\varepsilon + 1$, taken +1 as start 0 & end to $C^*/2$

Worst case - $O(b^{1+[C^*/\varepsilon]})$.

Space Complexity:

$O(b^{1+[C^*/\varepsilon]})$

optimal: always optimal.

6. Bidirectional Search Algorithm:

* Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward search. and other form goal node called as backward search to find the goal node.
* BDS replaces one single search graph with two small subgraph in which one starts the search from an initial vertex & other start from goal vertex.
* It search stops when these two graphs intersect each other.
* BDS technique such as BFS, DFS, DLS etc...,

Advantages:

* BDS is fast
* BDS is less memory

Dis-advantages:

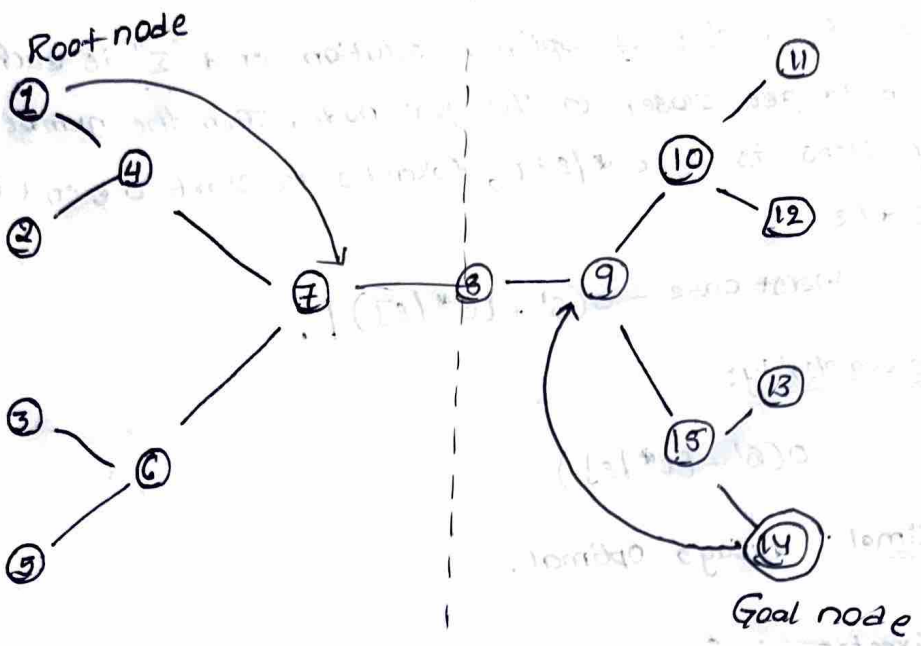
* Implementation of bidirectional search tree is difficult.
* BDS, one should know the goal state in advance

Completeness: BDS is complete, if we use BFS is both search

Time complexity: BFS is $O(b^d)$

Space complexity: $O(b^d)$

Optimal: yes it is optimal.


Root node
1
4
2
7
8
9
10
11
12
3
6
5
13
15
14
Goal node