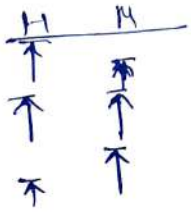
Artificial Intelligence

07/04/25

AI - mimic human behaviour

ANI - weak AI
AGI - strong AI } hypothetical
ASI -



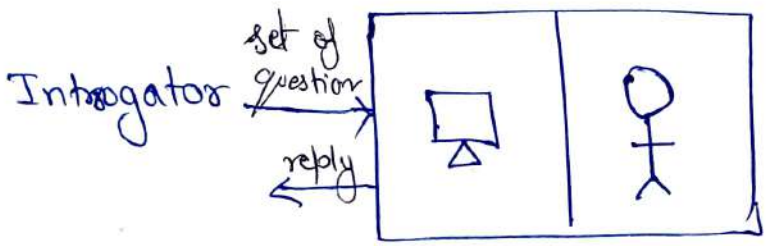
AI is a branch of computer science concerned with the study and creation of computer systems that exhibit some form of intelligence :

Systems that learn new concepts and ask system that can reason and draw useful conclusion about the world around us, systems that can understands a Natural Language or perceived and comprehend a visual see and the systems that performed other types of feats that require human type of intelligence.

(1955-56) Mid 1950s John Mc Carthy (coined the term AI)
1949 Alan Turing

↓

Turing Test (Imetation Game)



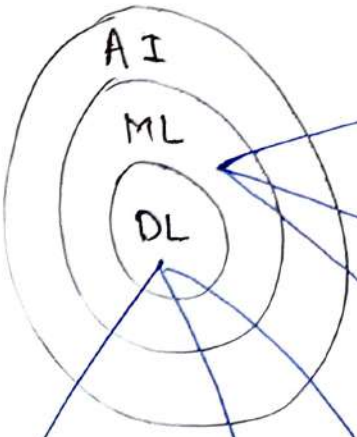
The Turing test is a method for evaluating a machine's ability to exhibit human-like intelligence.

According to Turing test, In a room there are two chamber, In one there is a machine and on another there is a person such that they can't communicate with each other.

from outside a ~~std~~ set of question is passed by the intrrogator and then machine and human send their responses/replies.

If Intrrogator is able to Differenciate between the replies between ~~two~~ machine and human, Then that ~~th~~ machine failed the Turing Test ~~an~~ else Machine passed the Turing Test.

If Machine failed that means machine need more practice.



supervised (labelled data)
(classification)

unsupervised (unlabelled data)
(form cluster)

Re-inforcement learning
(semi-supervised)

DL
|
Neural Network
↓
Nerrons

ANN (Artifical Neural Network)
- Text data
- e.g spam

CNN (convolutional Neural Network)
- image

RNN (Recurrent Neural Network)
- time series data

LISP (List Processing) — developed by: John Macarthy

Data collect — kaggle, uci repository

download

CSV load (comma seprated value)

clean data

Select feature

Split — 80% (use), 20% (test) (generally)

processor (Impute) — mean, median, mode — Bring value

Eliminate irrelevant data missing

08/04/25

* Types of AI

On the basis of Capabilities

→ 1) ANI / weak AI
(Artificial Narrow Intelligence)

→ 2) AGI / strong AI
(Artificial General Intelligence)

→ 3) ASI
(Artificial Super Intelligence)

On the basis of functionalities

→ 1) Reactive Machines
→ 2) Limited Memory
→ 3) Theory of Mind.
→ 4) Self-aware AI

1) ANI (Artificial Narrow Intelligence)

This type of AI is designed to perform Narrow task like face-recognition, ~~lang~~ Natral-language procesing or driving a car. Most curent AI system including those ~~who can~~ that can play complex game like chess, fall under this category. They operate under limited predefined Range or set of context.

2) strong AI / AGI (Artificial General Intelligence)

This type of AI posses humman like cognitive capablities which is able to tackle new and unfamilier task autonomously. Such a Robust AI framework passes the

capacity to anticipate, adapt and utilize its intelligence to resolve any challenge without meeting needing human guidance.

3) Artificial Super Intelligence (ASI):
This represents the future form of AI where machines could surpass human intelligence across all fields including creativity general wisdom and problem solving. Super Intelligence is currently a hypothetical situation.

1) Reactive Machines
Reactive Machines are AI systems that have no memory and are task specific. It means that an input always telling us the same output. The machine learning model tend to the reactive much because they take customer data. such as purchase or ~~serch~~ search history and use it to delivery Recommendation to the same customer. It performs more than the average human would as it is not possible to process huge amount of data such as customer's entire search history and feedback, customized recommendation.

Deep Blue — play chess
↓
1997 IBM super computer
Gareig Kas — world chess champion
computer defeated him.

e.g :- (1) Recommendation systems
(2) DeepBlue, IBM chess playing super computer Gareig Kaskhaw in late 1990s.

2) Limited Memory

This type of AI can store and use past data to make predication and forcasts. This system can make an informed and improved decisions by studing the past data they have collected. Most present day AI applications from chatbots to virtual Assitance and self-driving cars fall under this category. it imitates the way our brains neurons works together.

3) Theory of Mind

This is a more advanced type of AI that researchers are still working on. It would entail understand and remember emotions, believe, meaning and depending on those make decision. This type requires machines to understand human truly.

4) Self- Aware AI

This represents the future of AI, where machines will have their own consciousness, sentiments, and self- awareness. This type of AI is still a theoritical concepts and would be capable of understanding and possessing emotions, which could lead to form believes and desires.

* Application of AI

1) Machine vision - detection of object, find defected object from vision, color detection, sorting on basis of color. [focus on h/w aspect].

2) computer vision - focus of h/w aspect and DL Recognize object, crowd- find a person.

3) pattern Recognition - face-recognition, fingerprint recognition, character recognition (handwriting), medical image (MRI).

4) Natural Language Processing (NLP) :- understanding of language either in written or spoken format.

NLP

- voice: authentication using voice.
- speech: detect voice convert it to text. e.g :- Hey, google.

6) Game playing
7) Data Mining - search large amount of data, legal system uses. study large amt of data and bring important information
8) Robotics.
9) Expert System - Multiple experts knowledge is kept/store on a database (expertise)
eg - Mycin
9) Searching
10) Driver-less vehicals.

15/04/25

* Production System

The Production system in AI is a framework that consists in developing computer programs to automate wide range of task. Production system, serves as a cognitive architecture it encompasses rules representing declarative knowledge, allowing machines to make decisions and act based on different conditions. In production system knowledge is stored as rules within knowledge base. These rules consists of condition-action statements that define how the system should behave or react when presented with certain conditions. A Rule have two components if-then rules.

example -

if (age = youth & student = Yes) then (Buys.computer = Yes)

LHS — Antecede

RHS — consequence

Knowledge store in KB (Knowledge Base)
Also, it is in the form of these rules.

Production system uses knowledge in the form of rules to provide diagonsis or advice on the basis of input data. A production system consists of —

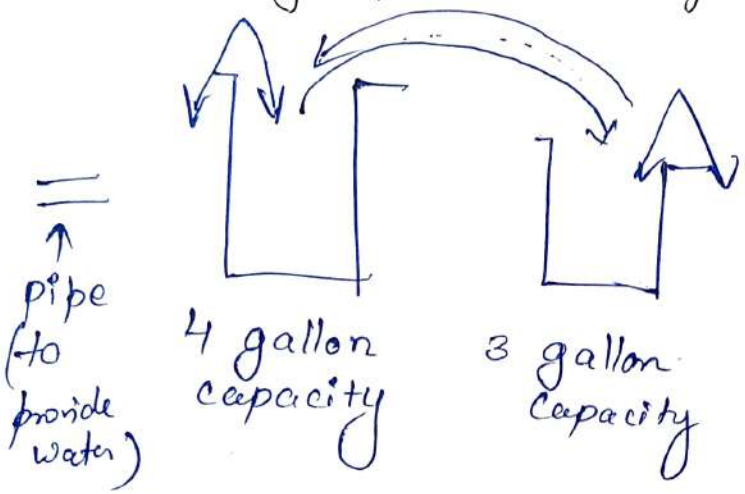
1) A set of Rules consisting of a LHS (Left-hand side) that determine the applicability of the rule. and the RHS (Right-hand side) that describes the operation to be performed if the rule is applied.

2) One or more knowledge base that contains whatever information is appropriate for particular task. some part of the database may be permanent while otherpart of the end may pertain only to the solution of current problem. The information in these databases may be structured in any appropriate way.

3) A control strategy that specifies the order in which the rules will be compared to a database and a way of resolving conflicts that arise when several rules are matched at once.

4) A Rule Applier - which is the computational system that implements control strategy and applies the rules.

example of production system - "Water - Jug Problem".



Task -
fill 4 gallan Jug half.
i.e, 2 gallan water should be on 4 gallon Jug.

Statement :-

you are given two jugs, A 4 galloan one and a 3 gallaon one, neither has any measuring marker on it, there is a pump that can be used to fill the jugs with water how can you get exactly two gallons of water into the 4 gallon jug.

Solution :-

The state space for this problem can be described as the set of ordered pair of integers. (x,y) such that $x = 0, 1, 2, 3, 4$ and y represents the quantity of water in 3 gallon Jug the start state is (0, 0) the goal state is (2, n) for any value of n.

x - quantity of water in 4 gallon jug value lies from 0 — 4
y " " " " 3 " " " " " 0 - 3

production rule for water-Jug problem..

Rule 1) Fill the 4 gallon Jug

$(x, y) \longrightarrow (4, y)$

if $x < 4$

Rule 2) Fill the 3 gallon Jug

$(x, y) \longrightarrow (x, 3)$

if $y < 3$

Rule 3) Pour some water out of 4 gallon Jug

$(x, y) \longrightarrow (x-d, y)$

pour water = d

if $(x > 0)$

Rule 4) pour some water out of 3 gallon Jug

$(x, y) \longrightarrow (x, y-d)$

if $(y > 0)$

Rule 5) Empty 4 gallon Jug on the ground.

$(x, y) \longrightarrow (0, y)$

if $(x > 0)$

Rule 6) Empty 3 gallon Jug on the ground

$(x, y) \longrightarrow (x, 0)$

if $y > 0$

Rule 7) pour water from 3 gallon Jug into the 4 gallon Jug until 4 gallon Jug is full.

~~$(x,y) \longrightarrow (4,y)$~~

~~$(x,y) \longrightarrow (y, x+y)$~~

~~$y \neq 4$~~

$(x,y) \longrightarrow (4, y-(4-x))$

if $x+y \geq 4$
& $y > 0$

Rule 8) Pour water from 4 gallon Jug into 3 gallon until 3 gallon Jug is full.

$(x,y) \longrightarrow (x-(3-y), 3)$

if $x+y \geq 3$
& $x > 0$

17/04/25

Rule 9) pour all water from 3 gallon jug into 4 gallon Jug.

$(x,y) \longrightarrow (y+x, 0)$

if $y+x \leq 4$
& $y > 0$

Rule 10) pour all water from 4 gallon jug into 3 gallon Jug.

$(x,y) \longrightarrow (0, x+y)$

if $x+y \leq 3$
& $x > 0$

Rule 11) pour the two gallons from 3 gallon jug into 4 gallon jug.

$(x, y) \longrightarrow (2, 0)$

if $x + y \geq 4$
& $y = 2$
$x = 0$

Rule 12) empty two gallons in the four gallon jug on the ground.

$(x, y) \longrightarrow (0, y)$

if $x = 2$

Solution 1. for water jug problem

gallons of water in 4 gallon jug	gallons in 3 gallon jug	Rule Applied
0	0	
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5
2	0	11

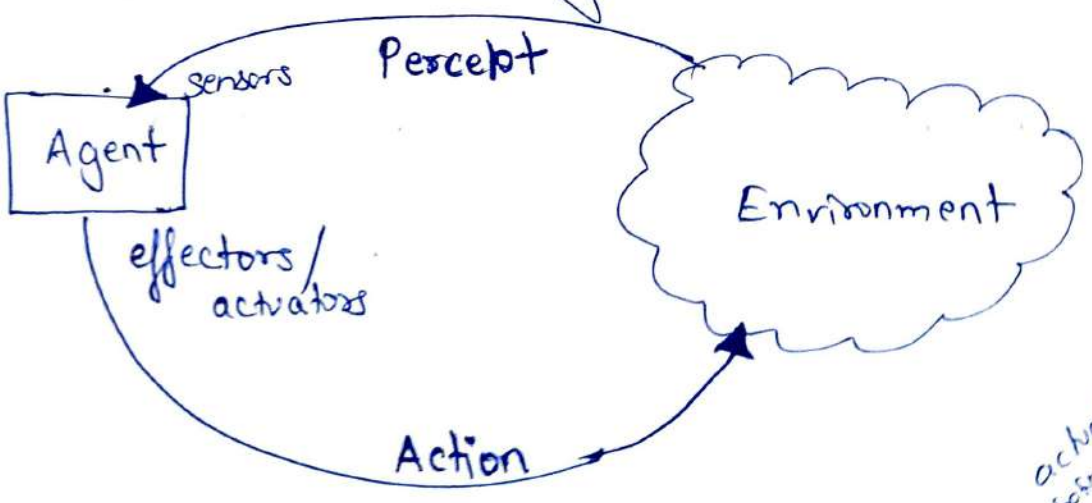
Solution

gallons of water in 4 gallon Jug	gallons in 3 gallon Jug	Rule Applied.
0	0	
4	0	1
1	3	8
1	0	6
0	1	10
4	1	1
2	3	8

- **Intelligent Agent :**

Intelligent Agent is an Autonomous entity which observes through sensors and act upon an environment using actuators. and directs its activity towards achieving a goal.

Intelligent Agent may also learn or use knowledge to achieve their goals they may be very simple or very complex.



actuators = robots are like

Agents have sensors, actuators and have goals. Agent program implements mapping from percept sequence to actions. An agent perceives its environment through sensors.

Russel & Norvig in 2003 grouped agents into five classes based on their degree of perceived intelligence and capability.

1. simple Reflex agent
2. Model based Reflex agent
3. Goal based agent
4. utility based agent
5. learning agent

22/04/25

1) simple Reflex Agent

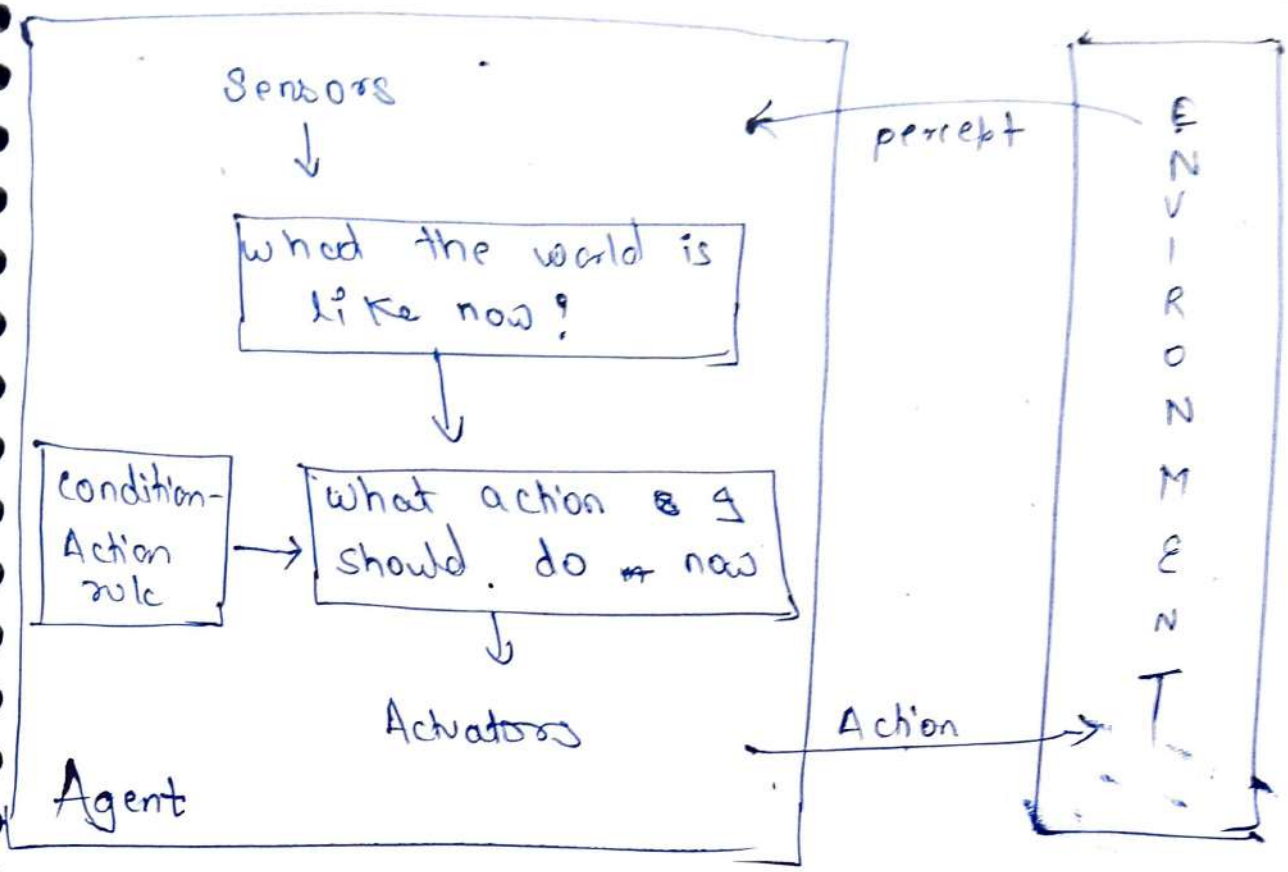
simple Reflex agent acts only on the basis of current percept, ignoring rest of the percept history. The agent function is based on condition action rule: " if condition then action "

example -

Automatic doors and smart thermostat

air-bag of car

smart thermostat - maintain constant temperature of the room, irrespective outside temperature increases or decreases.



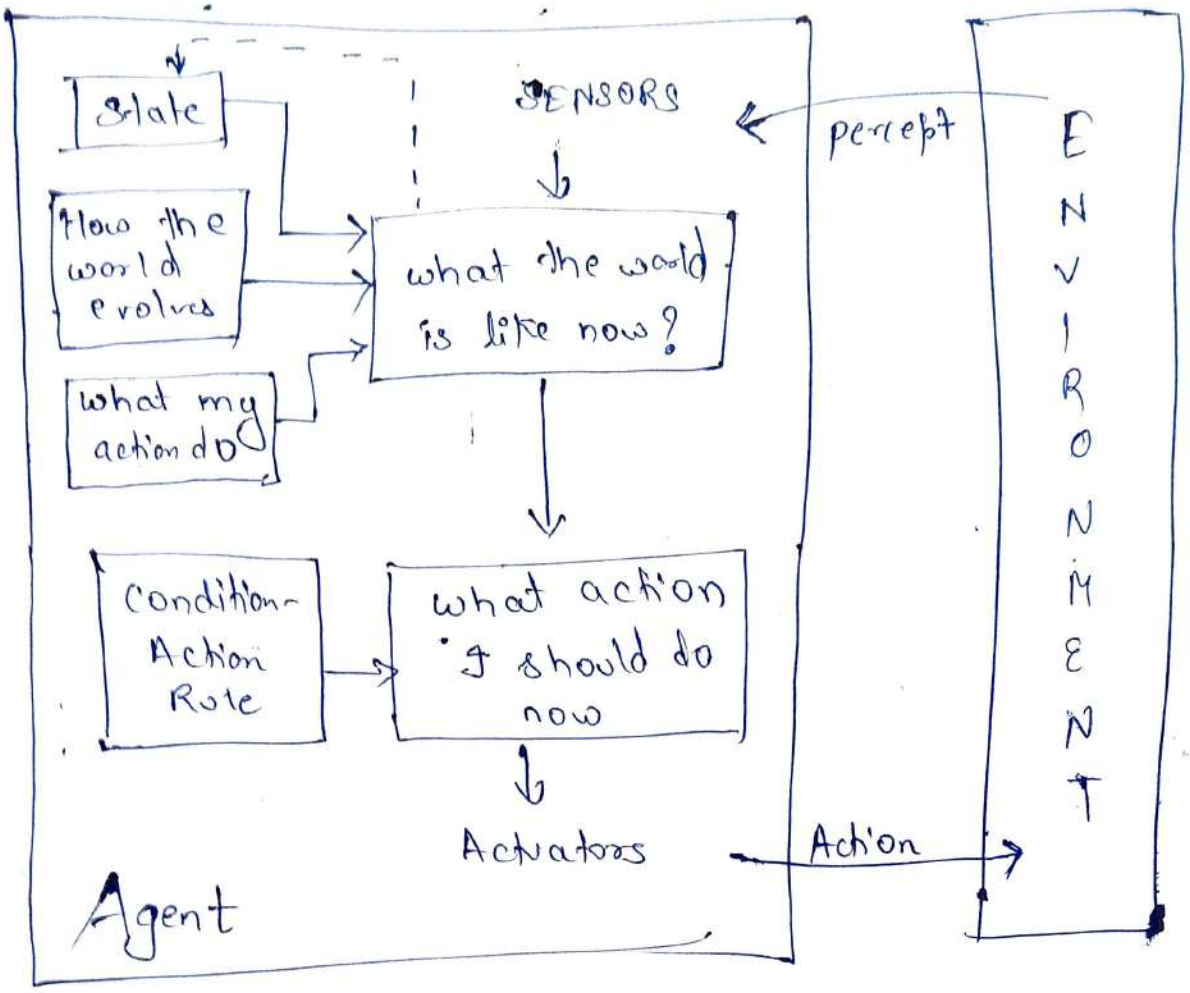
2. Model based Reflex Agent

The Model based reflex ~~of~~ agent can handle partially observable environment, its current state is stored inside the agent maintaing some kind of structure which describes the part of the world which cannot be seen. This knowledge is about "how the world works." is called the Model of the world. Hence the name is Model based Agent.

Example :

self driving car which respond to present road condition as well as also takes into account its knowledge of traffic rules, road maps and past experience.

auto-pilot

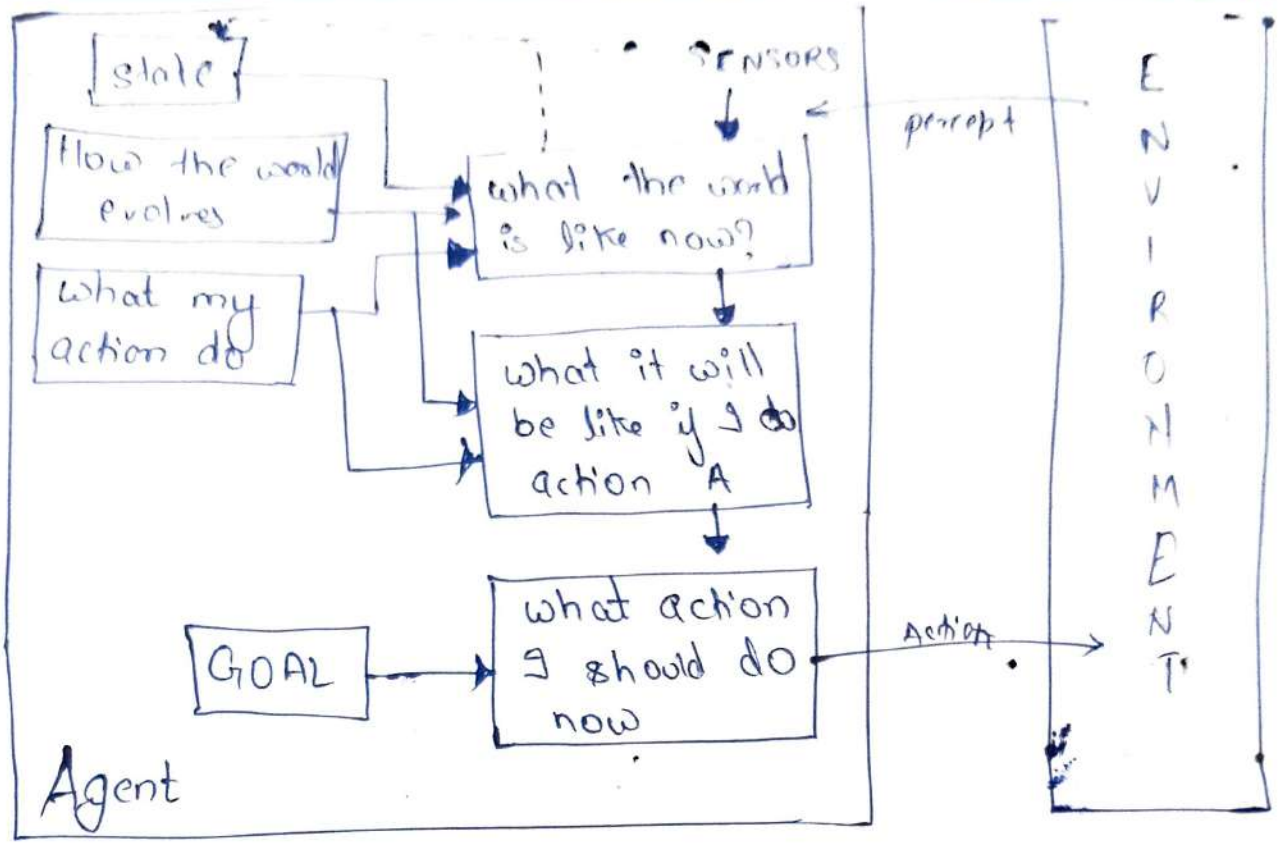


3. Goal-based agent

Goal based agent further expand on the capabilities of model based agent by using "goal" information. goal information describes situation that are desirable. This allows the agent a way to choose among multiple possibilities, selecting the one which reach a goal state. search and planning are the sub-fields of AI devoted to finding action sequence then achieve the agents goal.

example - chatbot or Robat as a delivery agent.

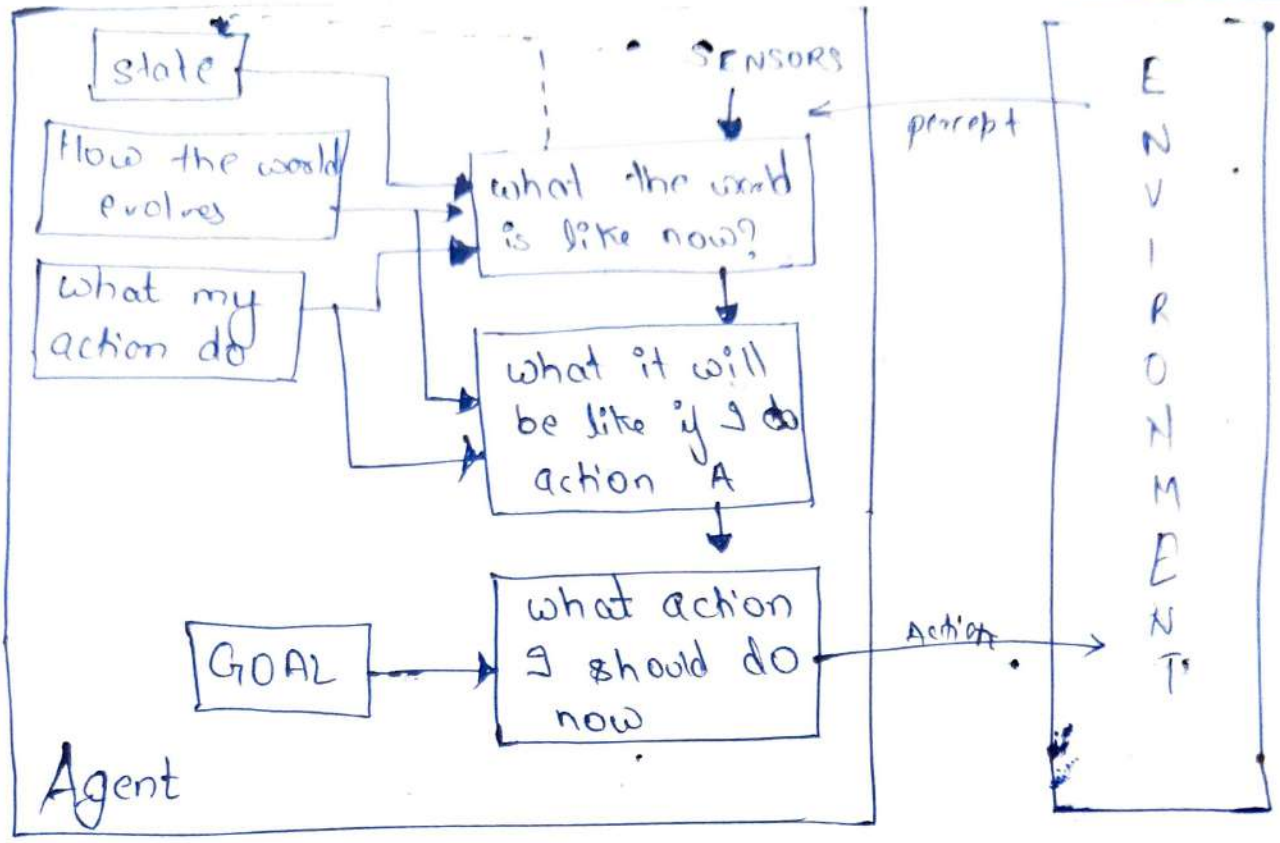
delivery agent



4. utility based agent

Goal based agent only distinguish between goal states and non-goal state. It is possible define a measure of how deserable a particular state is. This measure can be obtained through the use of utility function which maps a state to a measure of the utility of the state. A more general performace measure should allow the comparison of different world state according to exactly how happy they would make the agent. the term utility can be describe how "happy" the agent is.

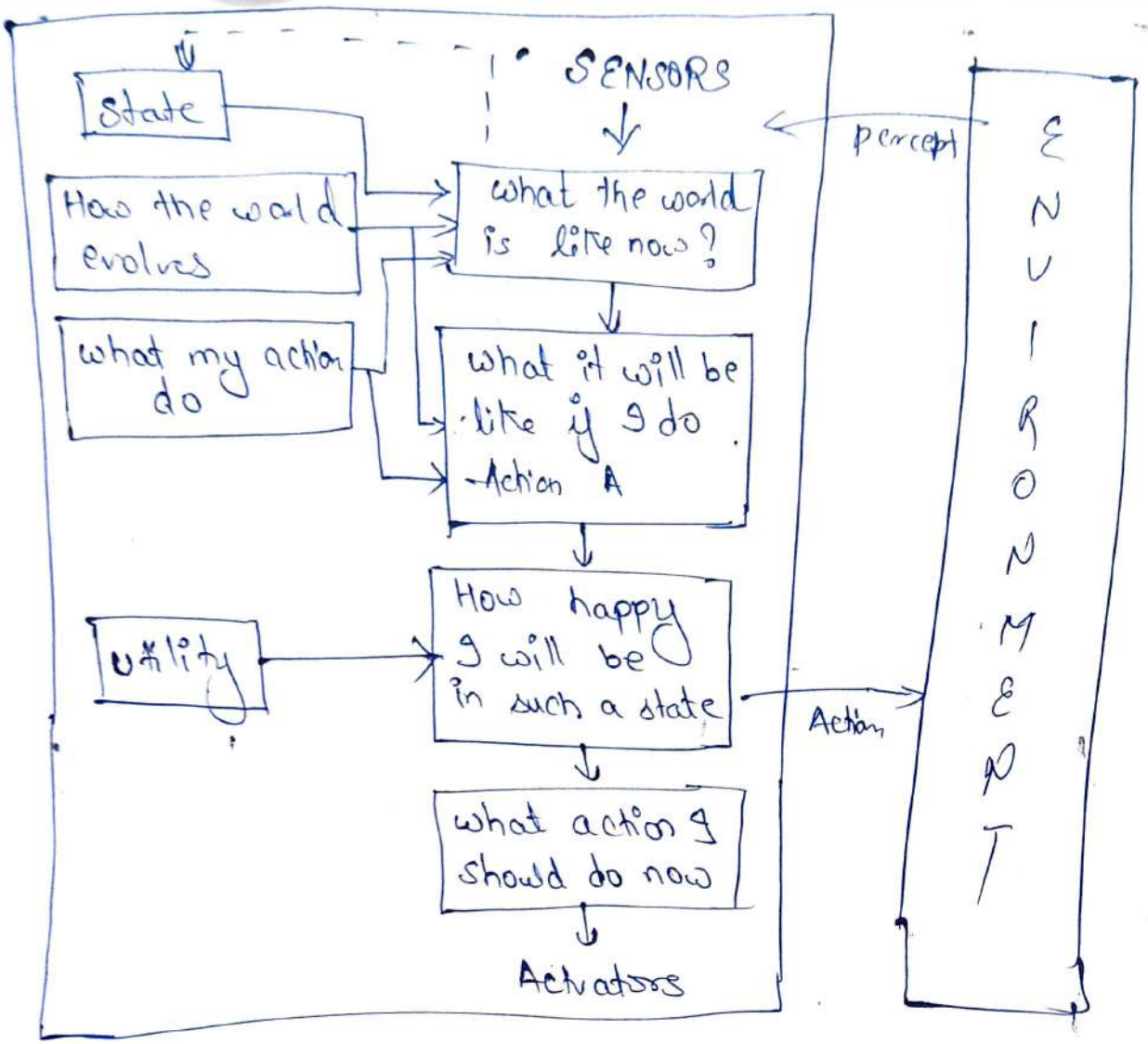
example - stock trading bot with the goal of maximizing the investors long term friancial satisfication.



4. utility based agent

Goal based agent only distinguish between goal states and non-goal state. It is possible define a measure of how deserable a particular state is. This measure can be obtained through the use of utility function which maps a state to a measure of the utility of the state. A more general perfosmace measure should allow the comparison of different world state according to exactly how happy they would matte the agent. the term utility can be describe how "Happy" the agent is.

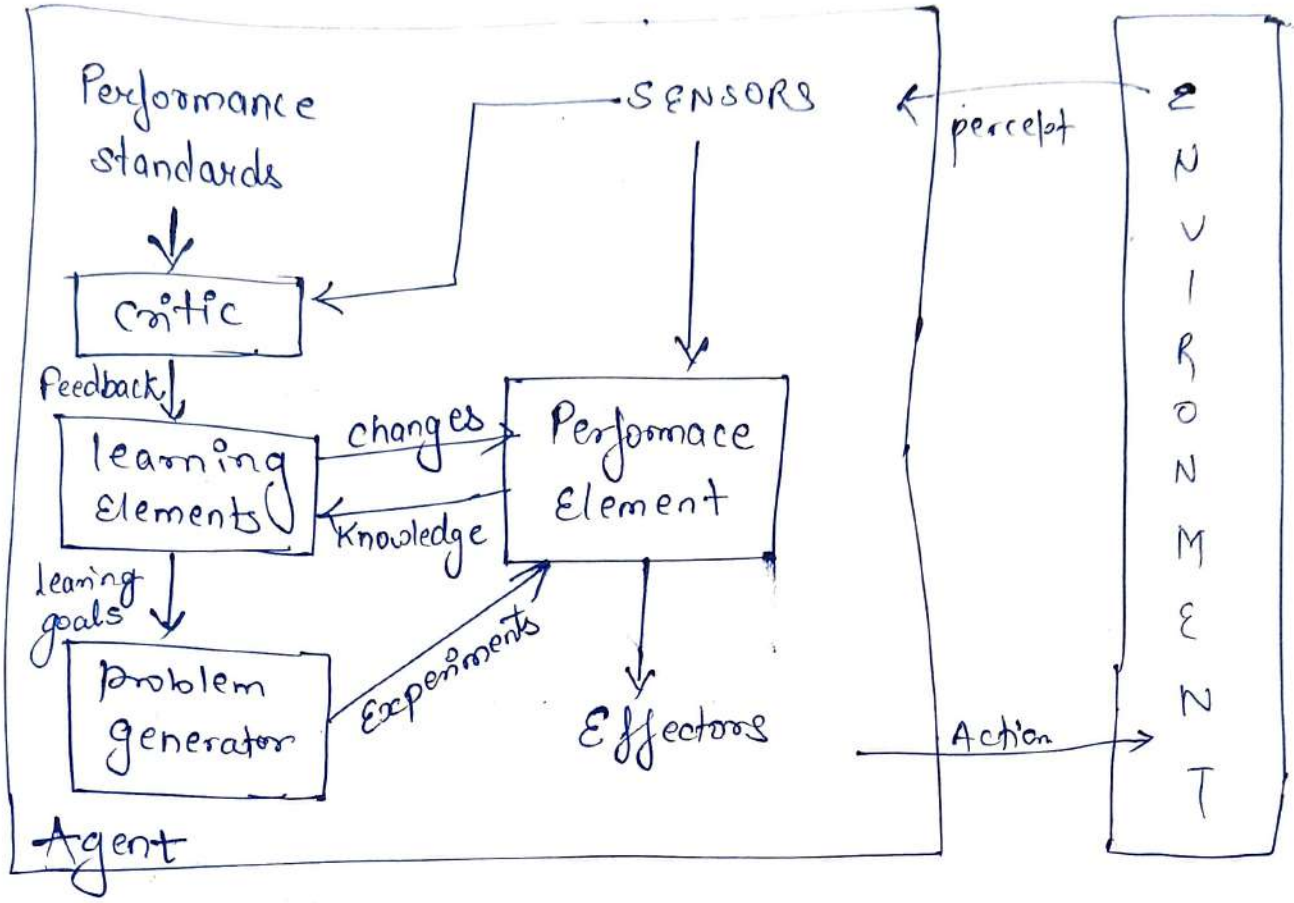
example - stock trading bot with the goal of maximizing the investors long term friancial satisfication.



5. Learning Agent

learning has the advancement that it allows the agents to initially operate in ~~unk~~ a unknown environment and to become more ~~conf~~ competent then its initial knowledge alone might allow. The most important distinction is between the "learning elements" which is responsible for making improvements and "performace element" which responsible for selecting external actions.

example - Personalized Recommendation system.



P – Performance measure
E – Environment
A – Actuators
S – Sensors

Self driving car

Path cover, destination reach, safety
Road, traffic, pedestrian
Steering, speedometer
camera, gps

peas represent the foundational ~~component~~ component that define AI agent behaviour

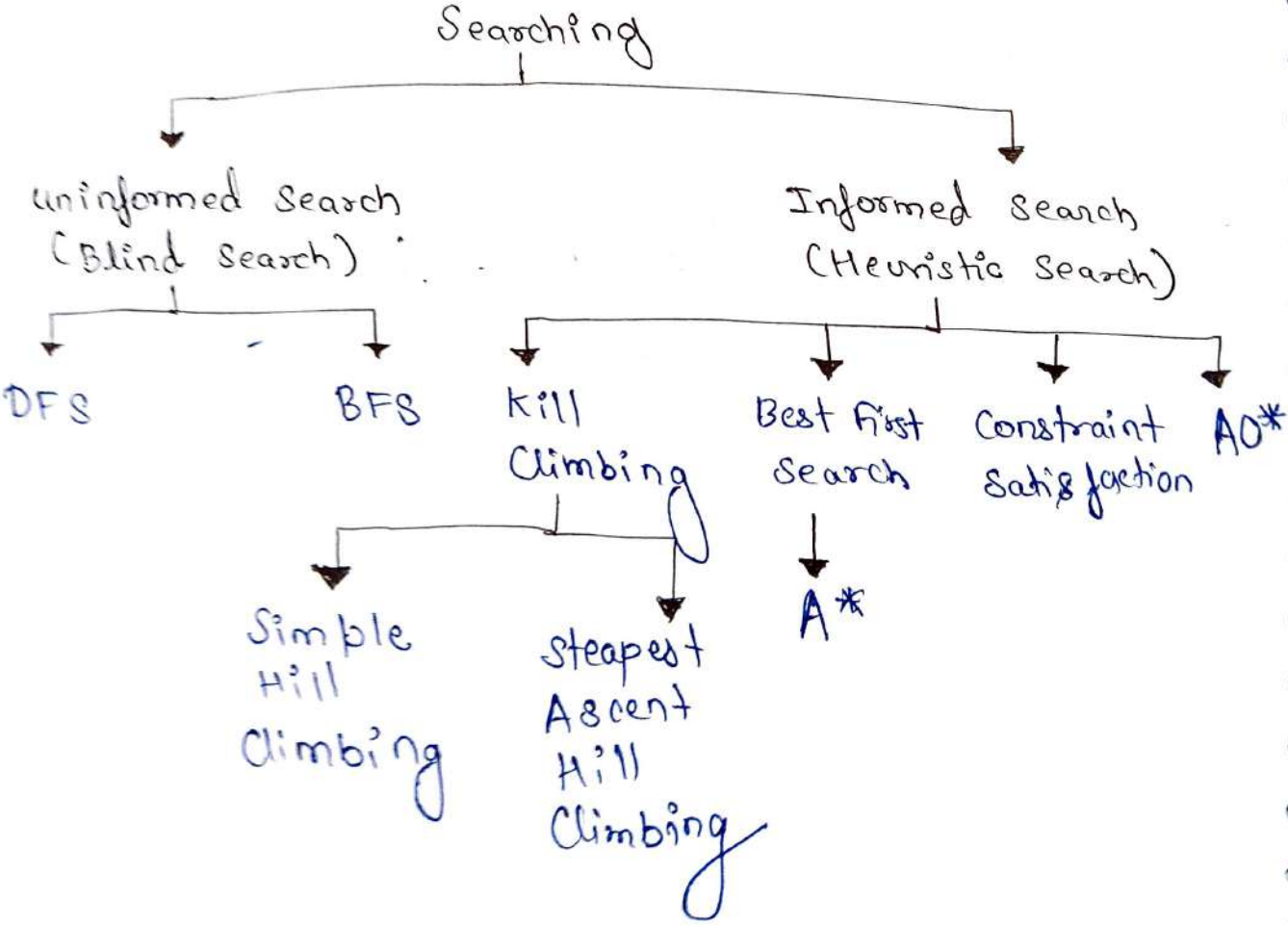
where P stands for performace measure

UNIT 3 5/4/23

* Searching

Search is a systematic examination of states to find path from state start state to the goal state. It is a search for a solution in a problem space search is fundamental to a problem space search is fundamental to a problem solving process.

Any problem can be solved by using rules in combination with an appropriate control strategic, to move to the problem space until a path from initial state to a goal state is formed. In worst case search explore all possible paths between initial and final state.



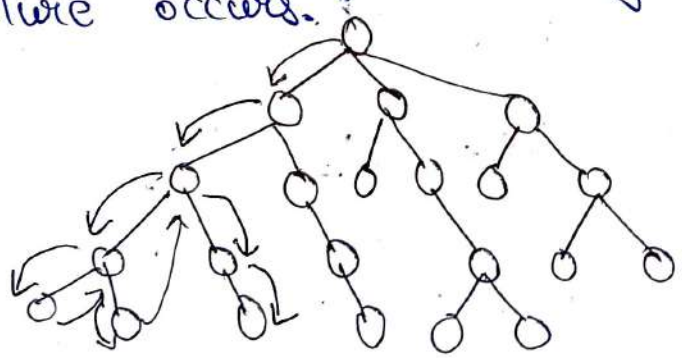
* Uninformed Search - Blind search or uninformed search uses no information about the problem to guide the search and therefore may not be very efficient. In this search, no performance is given to the order of successor node generation and selection. The path selected is blinding and mechanically follow. No information is used to determine the performance of one child over another

- DFS - Depth First search

DFS are perform by ~~dividing~~ diving downword into a tree as quickly as possible In DFS one has to examine all the children on the particular state and their descendant before coming to any of its siblings.

DFS always expand one node till the deepest level of the tree. Only one of the search hits dead end it goes back to the most recently expanded node and gurantees another of its children.

This process continues until a goal is found or failure occurs.



DFS uses stack as a data structure :

Algorithm →

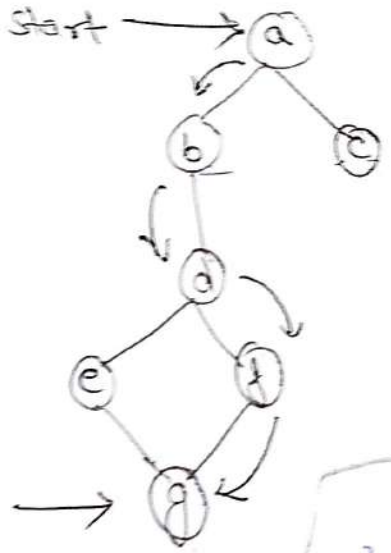
Step 1 : place the starting node of S on the stack.

Step 2 : If the stack is empty, return failure and stop.

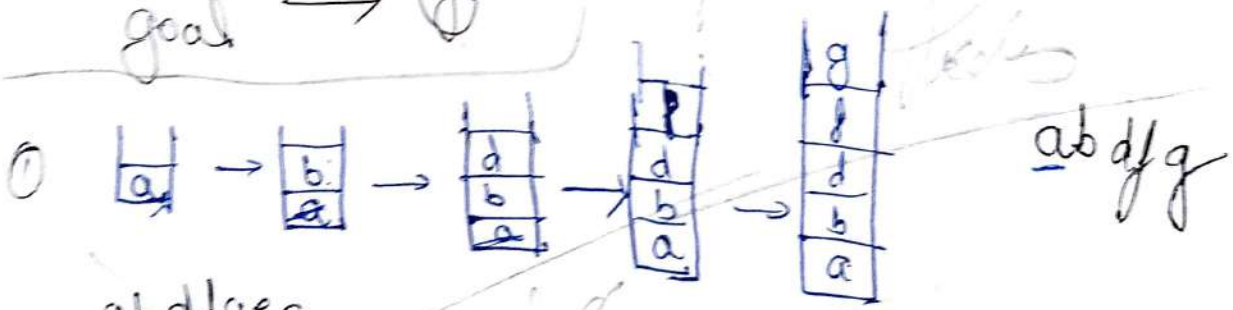
Step 3 : If the first element of the stack is the goal node g return success and stop.

Step 4 : Remove and expand the first element and place the children at the top of the stack.

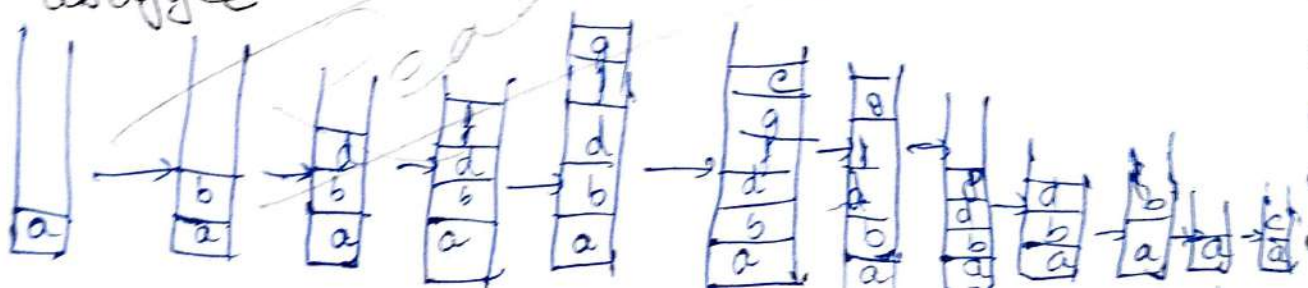
Step 5 : Return to step 2.



Same configuration



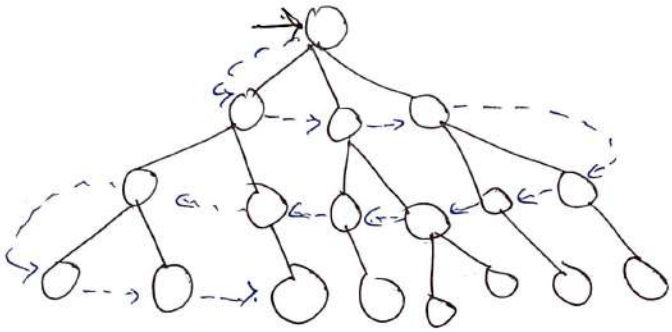
(2) abdfgee



02/05/25

• Breadth First Search (BFS):

BFS is performed by exploring all nodes at a given depth before proceeding to the next level. BFS systematically proceeds testing each node that is reachable from a parent node before it expands to any child of those nodes. BFS search the space in a ~~label~~ level by level manner. it explores nodes nearest to the root before exploring nodes that are further avail.



can start from Left/Right

till goal state is reached.

All the nodes at depth d in search ~~in~~ tree are expanded before those at depth d+1 using DFS one can find the shortest path from start state to goal state.

Queue is used as a data-structure to hold all generated but still ~~ex~~ unexplored nodes in BFS.

Algorithm

Step I: place the starting node s on the queue

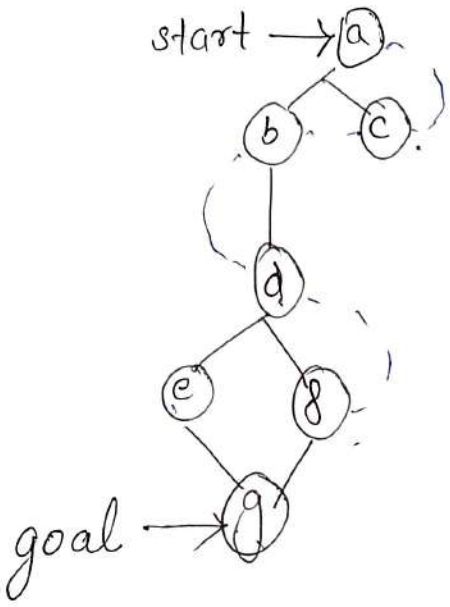
Step II: If queue is empty return failure and stop.

Step III: If the first element on the queue is the goal

node g retrn success and stop. Otherwise,

Step IV : Remove and expand the first elemt from the queue and place all the children at the end of the queue.

Step V : Retrn to step II.



FIFO

abcdefg

~~I. Step I.~~ [a]

II. [b|c]

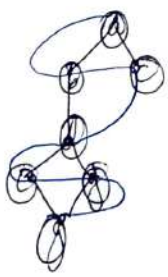
III. [~~b~~|d]

IV. [d]

V. [e|f]

VI. [~~f~~|g]

VII. [g]



acbdfeg

I. [a]

II. [c|b]

III. [b]

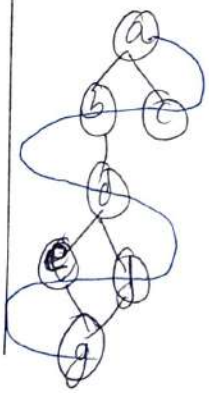
IV. [d]

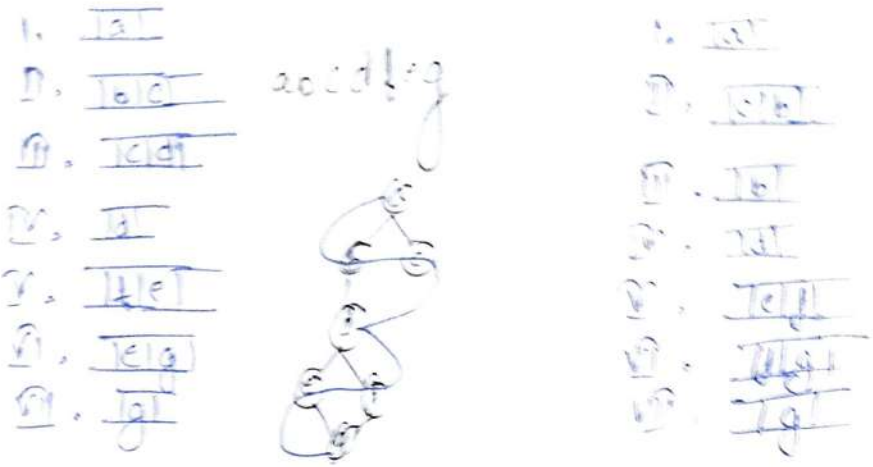
V. [f|e]

VI. [e|g]

VII. [g]

~~I. [a]~~





* Informed Search / Heuristic Search

Heuristic is a technique design for solving a problem more quickly when classical methods are too slow or for finding an approximate solution when classical method fails to find any exact solution. A Heuristic is only an informed guess of the next step to be taken for solving a problem it is often based on past experience or intuition.

It do not guarantee optimal solution but it offers solution which are good enough most of the time.

• Heuristic function :

A Heuristic function is a function that maps from problems state descriptions to measure desirability. It usually represented as number. The value of Heuristic function at a given node gives the estimate of whether a path is good or not. The purpose of Heuristic function is to guide the search process in the most profitable direction. By suggesting which path to follow when

more than one path is available.

if sol exist you get sol. but optimal not guaranteed

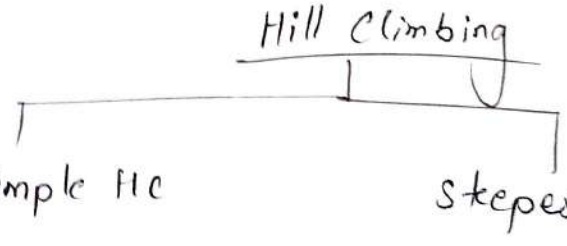
- Hill Climbing

03/05/25

Hill climbing is the varient of generate and Test in which feedback from the test procedure is used to help the generator decide which direction to move in the search space.

In a pure generate and test procedure the test function responds with only Yes or NO. But if the test function is augumented with a heuristic function that provides an estimate of how close a given state is to a goal state. the generate procedure can exploit it ~~to~~ reach the goal state.

Hill-climbing is often used when a good heuristic function is ~~et~~ available for evaluating states but no other useful knowledge is available.



(1) Simple HC :-

Step I: Evaluate the initial state, if it is a goal state then return it and quit.
Otherwise continue with initial state as the current state.

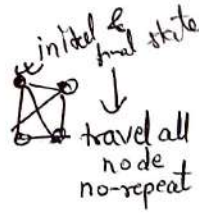
current state

Step2: look until a solution is found or until there are no new operators left to be applied in the current state:

a) Select an operator that has not yet been ~~appk~~ applied to the current state and apply it to produce a new state.

b) ~~to~~ ~~ge~~ evaluate the new state
   1) if it is a goal state, then return it and quit.
   2) if it is not goal state but better then the current state then make it current state.
   3) if it is not better then the current state, then ~~to~~ continue in the loop.

Absolute soln (goal predefined)
Relative " (travelling salesman problem)



• Variation of simple Hill Climbing

A useful variation of simple Hill Climbing ~~configures all~~ consider all the moves from the current state and selects the best one as the next state this method is called steepest Ascent Hill Climbing OR Gradient Search.

(2) ~~Ste~~ Steepest Ascent HC :-

~~Algorithm~~

Step 1: Evaluate the initial state, if it is also a goal

state then return it and quit.
otherwise continue with the initial state as the current state

Step 2: loop until a solution is found or until a complete iteration produces no change to the current state :

a) let SUCC (successor node) be a state such that any possible successor of the current state will be better then SUCC

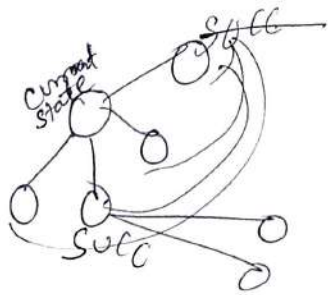
b) for each operator that applies to the current state to :

1) Apply the operator and generate a new state

2) Evaluate the new state.

- If it is a goal state, return it and quit.
- If not, compare it to SUCC. If it is better, then ~~sent~~ set SUCC to this state.
- If it is not better leave SUCC alone.

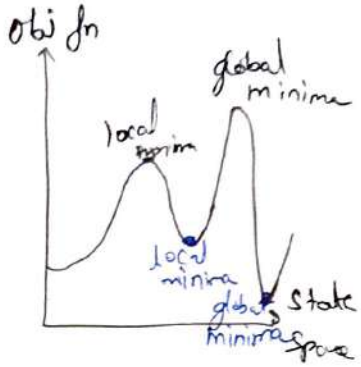
c) If the SUCC is better then the current state then set current state to SUCC.



* Drawbacks of Hill Climbing

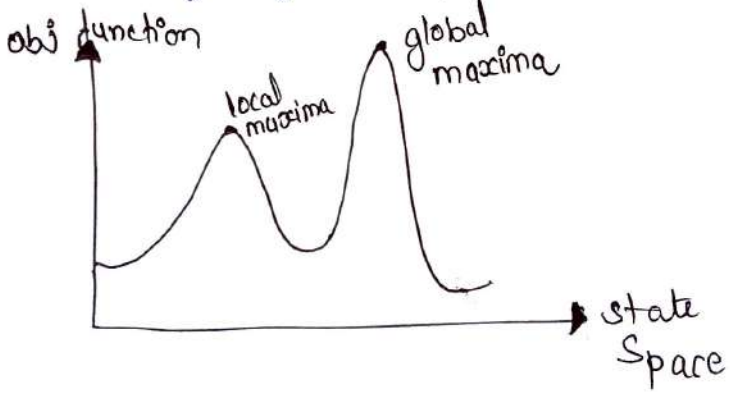
Hill Climbing suffers from some serious drawbacks which are as follows

1) Local Maxima
2) Plateau
3) Ridge



1) Local Maxima

A local Maximum is a state that is better then all its ~~neghiboori~~ neighbouring state but it is not better then some other state further away. At a local Maximum all moves appear to make things worse. Local Maxima are particularly frustating because they often occur almost within site of a solution, in this case they are called ~~footfills~~ foothills.



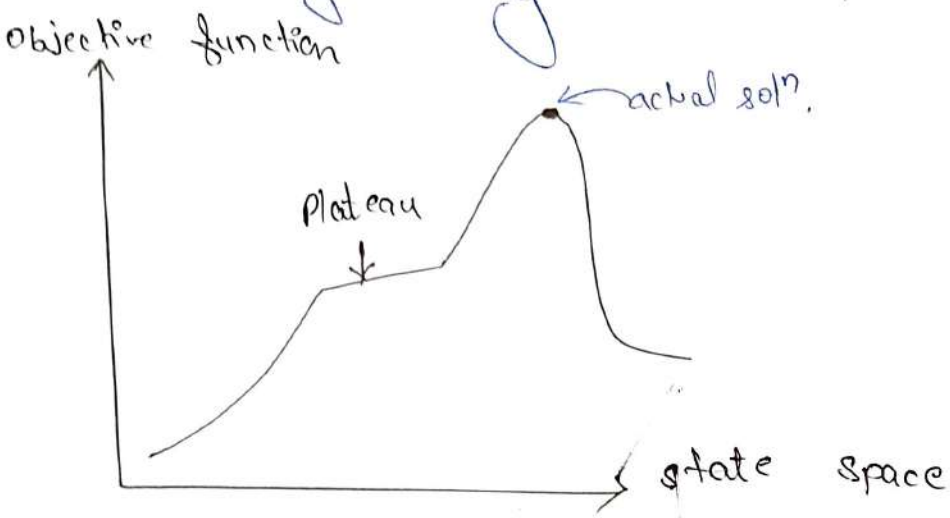
06/05/25

Solution:

Backtrack ~~too~~ to some earlier node and ~~try~~ going in a different direction.

2) Plateau : (Mesa problem)

A plateau is a flat area of search space in which a goal set of neighbouring states have same value. On a plateau it is not possible to determine the best direction in which to move by making local comparisons.



Solution : Make a big jump in some direction to try to get a new section of search space.

3) Ridge :

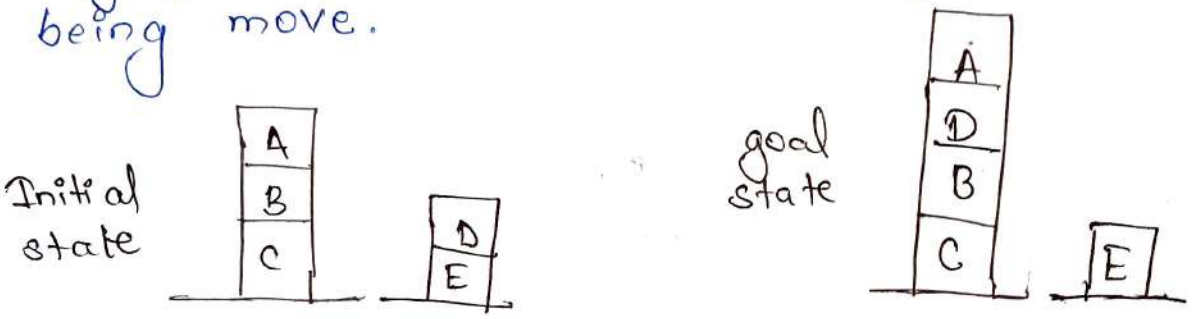
Ridge = Range of mountains

Ridge is a special kind of local maximum. It is an area of search space that is higher than surrounding areas and that itself has a slop. but the orientation of the high region compared to the set of available moves, makes it impossible to traverse a Ridge by single move.

Solution : Apply two or more rules before doing the test this corresponds to moving in several directions at once.

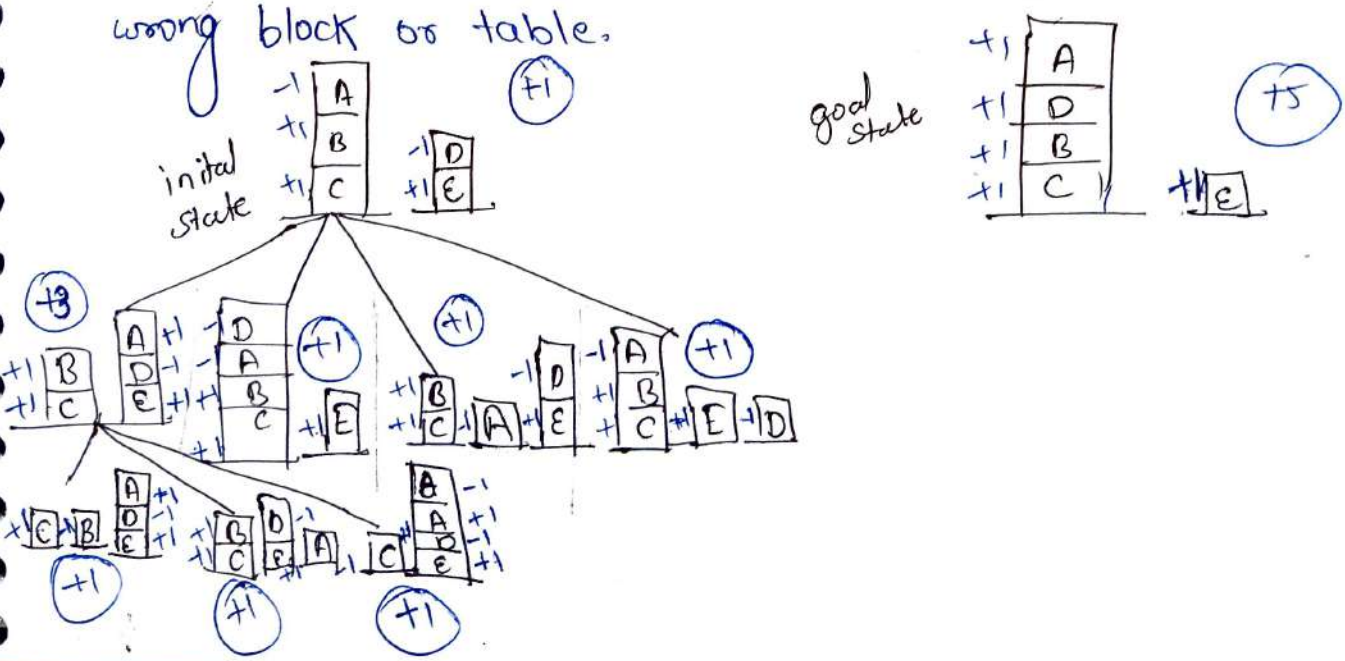
• Block World Problem

The Block world problem is a classic AI planning problem where the goal is to rearrange a set of blocks on a table to achieve desired configuration. A Robot arm can pick up and place blocks but only one at a time, only if no other block is on top of the one being move.



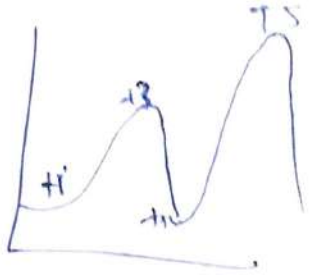
Heristi function

$h_1(n)$ = Add One if block is on correct block and or table and substract one if it is on wrong block or table.



since we stuck on local maxima.

Hence, back track, and work with next heuristic fn.



$h_2(n)$ = add 1 for every block in a correct structure that the block is setting on and substract 1 for every wrong structure.

✓

07/05/25

- Best-first search —— DS = priority queue

Best First search also depends on the use of Heuristic to select most promising path to goal nodes. Unlike hill-climbing this algorithm retains all estimates computed for previously generated nodes and makes its selection based on the best among them all. Thus at any point in the search process best first moves forward from the most promising of all the nodes generated so far. In doing so, it avoids the potential traps encountered in hill-climbing.

Algorithm

Step I: place the starting node s on the queue

Step II: If the queue is empty return failure and stop.

Step III: If the first element on the queue is the goal node g, return success and stop. Otherwise,

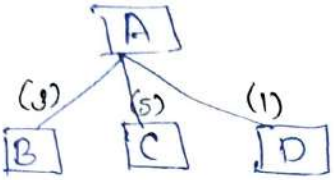
Step IV: Remove the first element from the queue expand it and compute estimated goal distances for each child. place the children on the queue and arrange all queue elements in ascending order corresponding to goal distances from the front of the queue.

Step V: Return to step II.

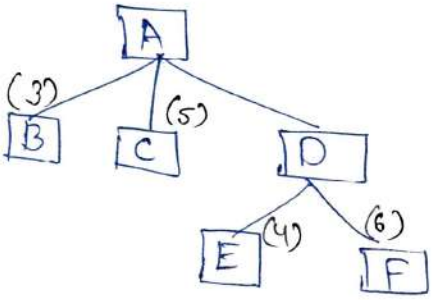
example

Step I:

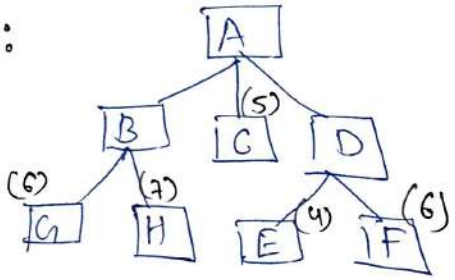
Step II:



Step III:



Step IV:



- **A* search**

A star or A* algorithm combines the feature of uniform first search and pure heuristic search to efficiently compute optimal solutions. A* algorithm is a best-first search algorithm in which the cost associated with ~~the~~ a node is $f(n) = g(n) + h(n)$

where,
- $g(n)$ is the cost of path from initial state to node n and
- $h(n)$ is the heuristic estimate of the cost of a path from node n to a goal.

Thus, $f(n)$ ~~Thus~~ ~~f(n)~~ estimates the lowest total ~~cost~~ cost of any solution path going through node n.

At each point a node with lowest f value is choosen for expansion. Ties among nodes of equal f value should be broken in favour of nodes with lower h value.

The algorithm terminates when a goal is choosen for expansion.

Algorithm

Step I: create a search graph G consisting of start node n_0. put n_0 on a list called OPEN.

Step 2: Create a list called CLOSE that is initially empty.

step 3: if OPEN is empty, exit with failure

step 4: Select the node on OPEN. remove it from OPEN and put it on CLOSE. Call this node n

step 5: if n is goal node, exit succesfully with the solution obtained by tracing a path along the point s from n to n_0 in G.

step 6: expand node n, generating the set M of its successors that are not already ancestors of n in G. List all these member of M of successor of n in G

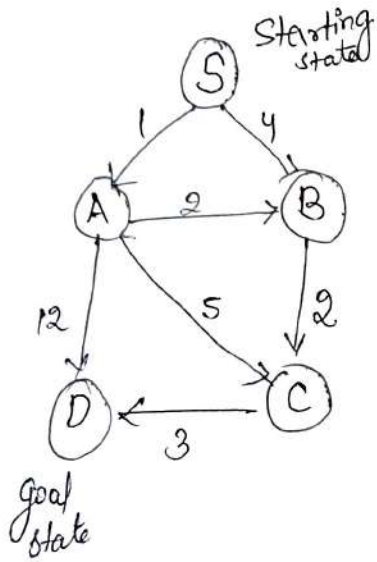
step 7: establish a pointer to n from each of those members of M that were not already in G. Add these member to M to open.

for each member m of M that was already on open or closed redirect its pointer to n if the best path to m found so far is through m. for each member of M already on CLOSED redirect the pointers of its desendents in G so that they point backward along the best path found so far to these desendents.

Step8 : Reorder the list OPEN in order of increasing f value

Step9 : go to step 3.

Example :



$f(n) = g(n) + h(n)$

heuristic value

S	7
A	6
B	2
C	1
D	0

$S = 0+7 = 7$

$S \to A = 1+6 = 7$

$S \to B = 4+2 = 6$

$(S \to B) \to C = (4+2)+1 = 7$

$(S \to A) \to B = (1+2)+2 = 5$

$(S \to A) \to C = (1+5)+1 = 7$

$(S \to A) \to D = (1+12)+0 = 13$

OPEN	CLOSE
~~S~~	~~S~~
$S \to A = 7$ ✗	$S \to B$
$S \to B = 6$ ✗	$S \to A$
$(S \to B) \to C = 7$	$(S \to A) \to B$
$(S \to A) \to B = 5$ ✗	
$(S \to A) \to C = 7$	
$(S \to A) \to D = 13$	

$(S \to A \to B) \to C = (1+2+2)+1 = 6$

$(S \to A \to B \to C) \to D = (1+2+2+3)+0 = 8$

OPEN	close
bd	$(S \to A \to B) \to C$

09/05/25

* Minimax search

Minimax algorithm is a backtracking algorithm which is used in decision making and game playing. It provide an optimal move for the player assuming that oppent is also playing optimally. Minimax algorithm uses recursion to search through the game tree It is mostly used for game playing in AI such as chess, checkers, tic-tac-toe and various 2-player games. This algorithm amputes the minimax decision for the current state. In this algorithm there are two players One is called MAX and the other is called MIN. both players fight it as the oppenent player gets the minimum benefit while they get the maximum benefit. Both the players are oppenent of each other where MAX will select maximised value and MIN will select minimised value. This algorithm performs a depth first search for the exploration of complete game tree. The min-max proceeds all the way down to the terminal node of a tree and then backtrack the tree.

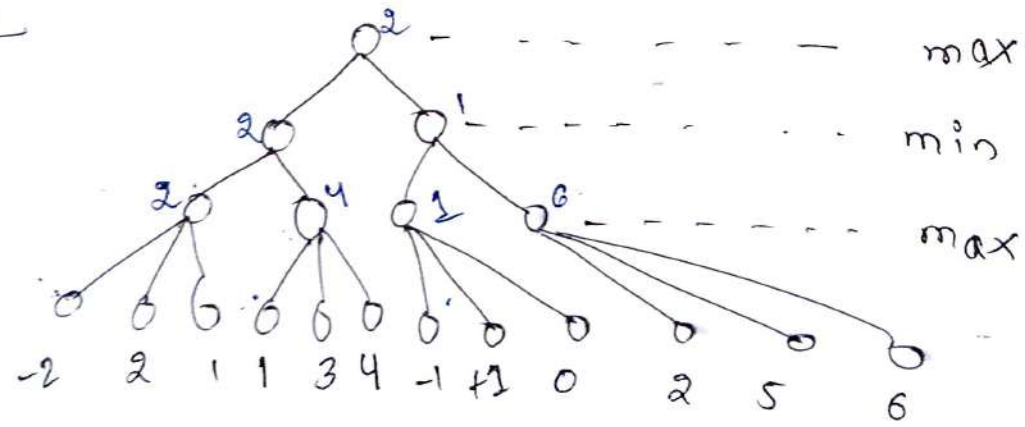
Working

In the tree, there are two types of nodes maximizing node and minimizing node. maximizer will try to get the maximum possible score and minizer will try to get the minimum possible score.

steps used for picking a next move:

1. Since, its turn to move for player 1, the start node is max node with current board configuration.
2. Expand nodes down to some depth of look-ahead in the game.
3. Apply evaluation function at each of the leaf node.
4. Backups value for each non-leaf node until computed for Root node.
5. At mean node the backup value is the minimum of values associated with its children.
6. At max node the backup value is the maximum of the value associated with its children.

Eg:-



1st max
1st min
Compare with other only

max = $-\infty$

min = ∞

max $(-\infty, -2) = -2$

max $(-2, 2) = 2$

max $(2, 1) = 2$

max $(-\infty, 1) = 1$

max $(1, 3) = 3$

max $(3, 4) = 4$

max $(-\infty, -1) = -1$

max $(-1, +1) = 1$

max $(1, 0) = 1$

max $(-\infty, 2) = 2$

max $(2, 5) = 5$

max $(5, 6) = 6$

min $(\infty, 2) = 2$

min $(2, 4) = 2$

min $(\infty, 1) = 1$

min $(1, 6) = 1$

~~max $(-\infty, 2) = 2$~~

max $(2, 1) = 2$

- Alpha - Beta pruning :

The problem with minimax algorithm is that the no. of game states it has to examine is exponential in the number of mover. The Alpha-beta pruning helps to arrive at correct minmax algorithm decision without looking at every node of the game tree.

The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does but it removes all the nodes which are not really affecting the final decision by making algorithm slow hence by pruning these nodes it makes the algorithm fast.

The two parameters can be defined as alpha – The highest value choice we have found so far at any point along the path of maximizer. The initial value of alpha is $-\infty$.

beta – The lowest value choice we have found so far at any point along the path of minizer. The initial value of beta is $+\infty$.

The main condition which is required for Alpha beta pruning is $\alpha \geq \beta$. the max player will only update the value of alpha and the min player will only update the value of beta.

While Backtracking the tree the node value will be passed to upper nodes instead of Alpha and beta. It will pass only the alpha-beta values to the child node.

Working:

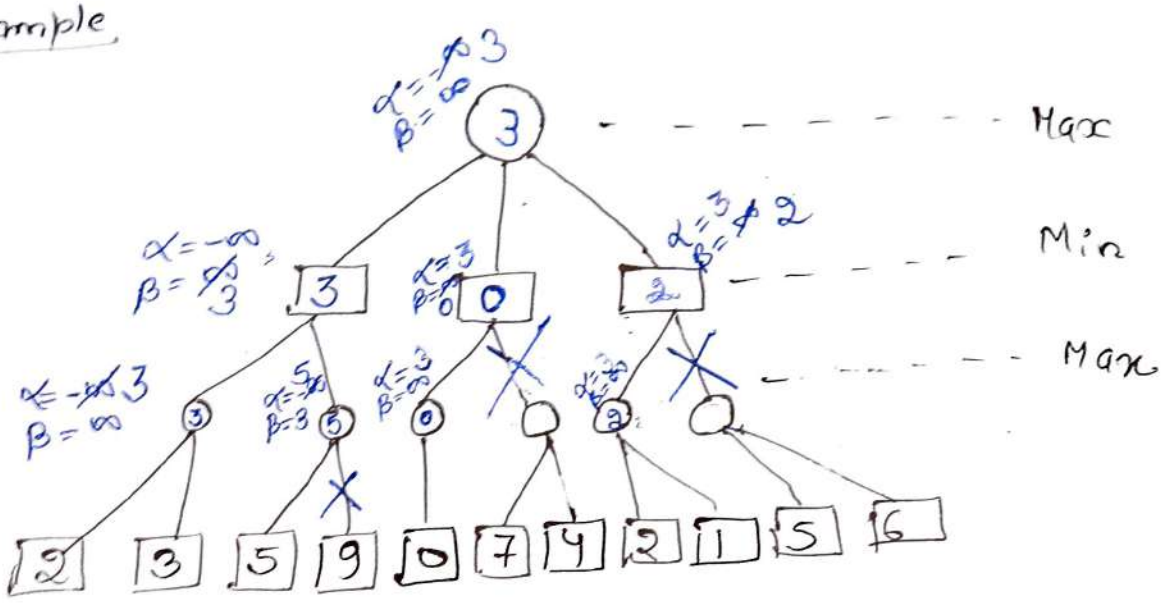
1) Copy the alpha and beta values from the parent node. If no parent node exist then initialize the value of alpha and beta to $-\infty$ and $+\infty$ respectively.

2) For every branch:
   1) Recursively process the branch
   2) update the current alpha or beta values depending on the values of branch after processing
   3) If $\alpha \geq \beta$ then prun rest of the branch

3) Get the value of the node to the max node or min node value.

10/05/25

Example



Max = alpha = $-\infty$

Min = Beta = ∞

Condition

$\alpha \geq \beta$

α cutoff = 1

β cutoff = 2

$\max(-\infty, 2) = 2$

$\max(2, 3) = 3$

$\max(-\infty, 5) = 5$

$\max(-\infty, 0) = 0$

~~$\max(-\infty, 7) = 7$~~

~~$\max(7, 4) = 7$~~

$\max(-\infty, 2) = 2$

$\max(2, 1) = 2$

~~$\max(-\infty, 5) = 5$~~

$\min(\infty, 3) = 3$

$\max(-\infty, 3) = 3$

$\min(\infty, 0) = 0$

$\max(3, 0) = 3$

$\min(\infty, 2) = 2$

$\max(3, 2) = 3$

* AO* (And-OR star)

AO* algorithm is a best first search method. It explores multiple path simultanously. Its this makes it more efficient for problems that involve AND-OR nodes.

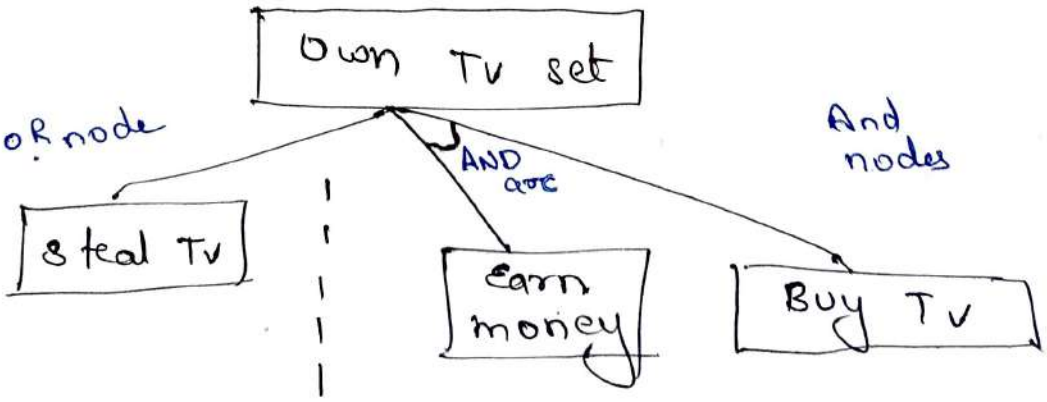
And AND Nodes –

It represents states where all child nodes must be satisfied to achieve a goal. If a task require multiple conditions to be met, it would be represented as an AND node.

OR nodes –

It represents states where atleast one child must be satisfied to achieve a goal. this type is useful in scenerios where multiple paths can lead to a solution.

AND-OR graph is useful for representing the solution of problems that can be solved by decomposing them into a set of smaller problems all of which must then be solved.



each node in the graph will have a heuristic value associated with it.

$$f(n) = g(n) + h(n)$$

where, $f(n)$ = cost function
$g(n)$ = actual cost or edge value
and $h(n)$ = heuristic / estimated value of node.

The AO* algorithm involves first calculating heuristic values for each level and then updating these values from the leaf nodes upward to the root node.

Assignment

submission date : 22nd May
file + Handwritten

1. What ~~is AI~~ do you understand by AI? what are its different application areas. Explain.
2. What do you mean by Intelligent agent Explain different types of Intelligent agent.
3. Discuss the concept and uses of ANN. and ~~su~~ make suitable differences between. ANN and BNN.
4. Write ~~a~~ the concept of frame and conceptual dependency with ~~favarable~~ suitable examples.

Algorithm of AO*

step1 : place the starting node s on open

step2 : using the search tree constructed, thus far compute the most promising solution tree T_o

step3 : select the node n, that is both on open and a part of T_o.
Remove n from OPEN and place it on CLOSE

step4 : If n is a terminal goal node, label n as solved. If the solution of n results in any of n's ancestor being solved, label all the ancestor and solved,
If the start node s is solved exit with success where T_o is the solution tree.

Remove from open all nodes with a solved ancestor

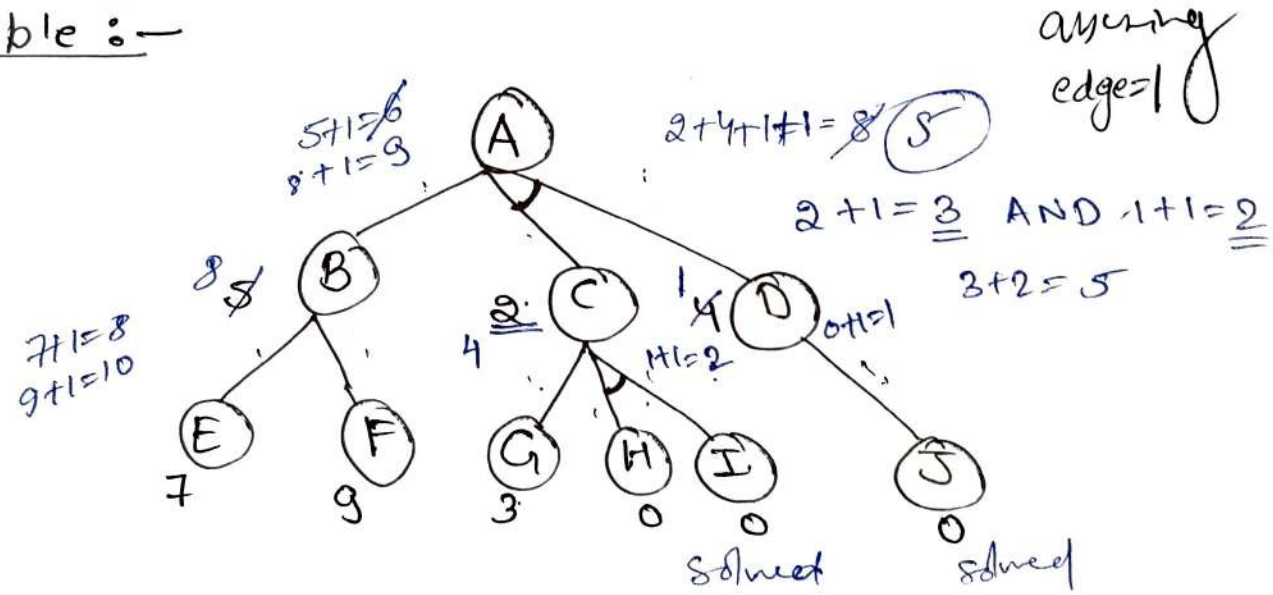
step5 : If n is not a solvable node, label n as unsolvable. If the start node is labelled as unsolvable, exit with failure.
If any of n's ancestor become unsolvable because n is, labelled them unsolvable as well.
Remove from OPEN all nodes with unsolvable ancestors.

Step 6 : Otherwise, expand nod n generating all of its successors for each such successor node that contains more than one sub-problem generate them

successor to give individual subproblems.
Attach to each newly generated node a back pointer to its predecessor.
Compute the cost estimate h^* for each newly generated node and place all such nodes on OPEN.
Next, Recompute the value of h^* and n and each ancestor of n.

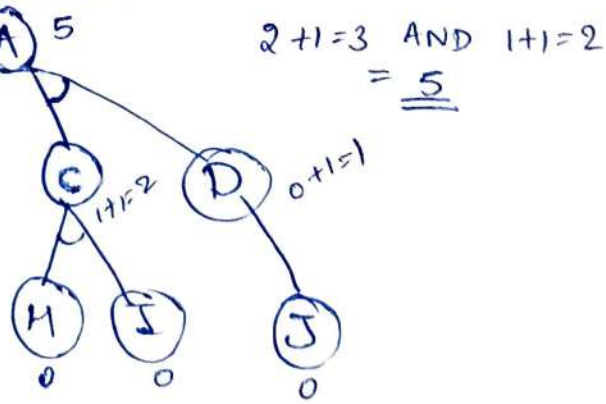
Step 7: Return to step 2.

Example :—



min (6, 8) = 6
min (8, 10) = 8
min (4, 2) = 2
min (1, 4) = 1
min (9, 5) = 5

Best path



A → (C+D) → (H+I)

Why ? — because, minimum cost

14/05/25

* Genetic algorithm (GA) :-

Darwin's theory - "survival of fittest"

genetic algorithm is the undermine search and optimization technique guided by the principle of natural genetic system. gentic algorithm learning methods are based on models of natural adaptation and evaluation.

It is a part of evoluationary computing, rapidly growing area of AI genetic algorithm are adaptive heuristic search algorithm based on Evolutionary ideas of natural selection and genetics. It is inspired by darwin's theory about evolution — "survival of fittest."

In nature competition among individuals for scandy (limited) resources results in the fittest individuals dominating over weaker ones.

Genetic Algorithm are intelligent exploitation of random search used in the optimization problems It also uses historical information to direct the search into the region of better performance within search space.

In Genetic algorithm the proces of finding solutions generates other points or other possible solution

as evolution proceeds. It do not break easily even if input changes slightly. It offers significant benefits over many other typical search optimization technique.

1) Operators of genetic Algorithm.
   1) Selection - select the best & discard the rest!!
   2) Crossover
   3) Mutation

1) Selection : The process that determine which solution are to be preserved and allowed to reproduce and which one deserves to die out. The primary objective of selection operator is to emphasize the good solution and eliminate bad solution in a population by ~~pick~~ keeping the population size constant.

"select the best & discard the rest"

~~Sect~~ Selection means extract a subset of genes from an existing population, according to any defination of quality every gene has a meaning so one can derive from the gene a kind of quality measurement called fitness function. selection can be performed by fitness value.

fitness function quantifies the optimality of the a solution so that the particular solution can be rank against all the other solution.

2) crossover : crossover is a genetic operator that combines two chromosomes to produce a new chromosomes. The idea behind crossover is that new chromosome may be better than both of the parents if it takes best characteristics from each of the parent.

Mating between two string is accomplised with crossover operation which randomly selects a bit positions in a string of bits and concatenates the head of one parent to the tail of another parent to produce offspring.

Suppose,

parent 1 : xxx|xxxxx
parent 2 : yyy|yyyyy

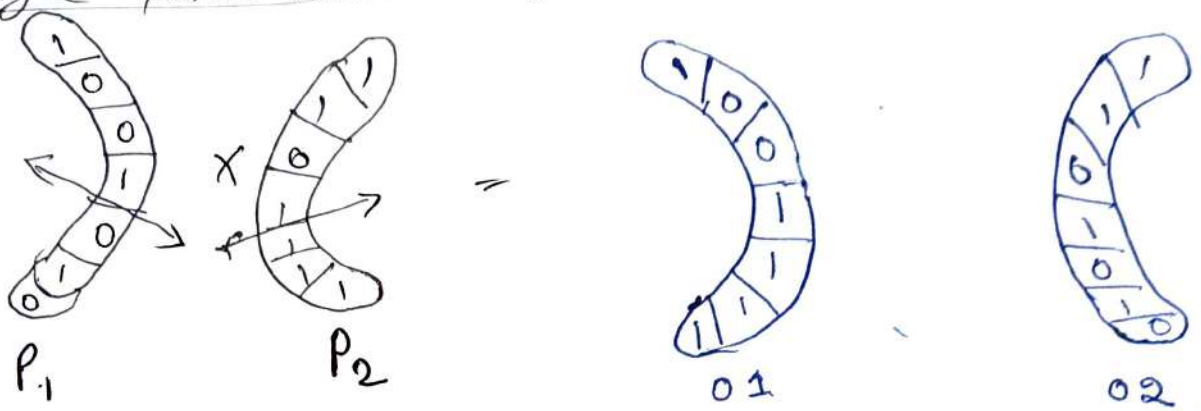
third bit position has been selected as a crossover point.

offspring 1 : xxxyyyyy
offspring 2 : yyyxxxxx

both parents no of bit should be same

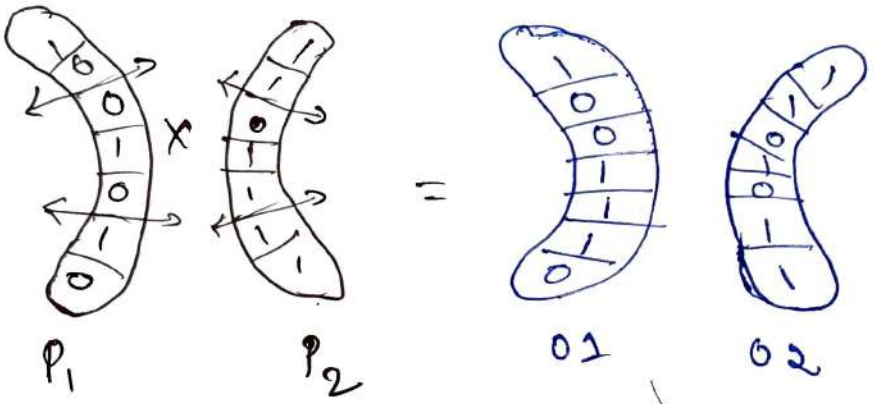
single point crossover

- single point crossover :-

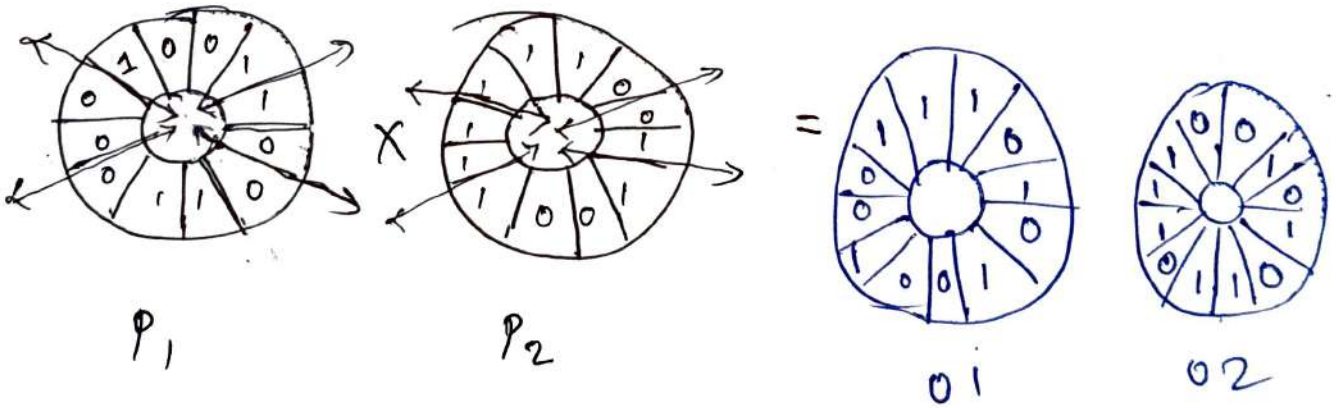


Structure of offspring should be, that of similar to their parent.

- Double point crossover :-



- multi point crossover :-



16/05/25

3) Mutation :-

Mutation is the occasional introduction of new feature into the solution string of the population pool. to maintain diversity in a population.

Mutation is used to ensure that all ~~mutation~~ of the new place are reachable, they every potential groom in the sole space is available for evolution. This ensures that the selection process does not get caught in a local minima or local optima.

It may happen that crossover, and inversion operator will only produce a set of structures that are better then all local members but not optimal in global sense. so mutation of can overcome this by simply selecting any bit position in a string at random and ~~strangeably~~ changing it. This operator simply inverts the value of choosen/selected gene that is 0 close to 1 and 1 close to 0.

example -

original offspring :- 0 1 1 0 1 1 0 1

Mutated " :- 0 1 1 1 1 1 0 1

Inverted or muted gene.

Application :-

1) RNA structre Algo
2) Building Bioo. plogentic gene
3) learning Robot behaviour using genetic algorithm
4) Biology and computational. chemeistry
5) Static mutation testing
6) construction of facial composite of supects of by eye witnes in forensic science.

Algorithm :-

Step I :- Initialize :
create a ~~composition~~ population of n element

Step II :- Selection :
Evaluate the fitnes of each element of the population by fitnes function.

Step III :- ~~Int~~ Reproduction :
Repeat 10 times
a) pick two parents with probability according to relative fitnes
b) crossover
c) Mutation
d) Add new child to new population

Step IV :- Replace the ~~of~~ old population with new population. and turn to step 3.

Q-> consider the problem finding global maximum of the following function.

$$f: \{0, \ldots 31\} \rightarrow R$$

$$x \rightarrow x^2$$

Inital random population is 13, 24, 8 and 19

1. selecting parent :-

No. of Individual	String (genotype)	x value (phenotype)	$f(x)=x^2$	Pselect $P_i = \frac{f_i}{\Sigma f_i}$	Expected Count $P_i \times 4$
1	01101	13	169	0.14	0.56
2	11000	24	576	0.49	1.97
3	01000	8	64	0.06	0.22
4	10011	19	361	0.31	1.23
			1160		

$1160\overline{)1690}(0.14$
1160
5300
4640
660

$1160\overline{)5760}(0.49$
5800
4640
11200
10440
760

$1160\overline{)6400}(0.055$
5800
6000
5800
200

$1160\overline{)3610}(0.031$
2320
1290
3480
1300
1160

Pi * Total no. of individual

2. Cross over:

No. of Individual	String (genotype)	Crossover point	New offspring (string) (genotype)	x value phenotype	fitness $f(x)=x^2$
1	0 110|1	4	01100	12	144
2	1100|0	4	11001	25	625
2	11|000	2	11011	27	729
4	10|011	2	10000	16	256
					1754

3. Mutation:

Obj no.	Offspring after Crossover	offspring after Mutation	x-value phenotyl	$f(x)=x^2$
1	01100	11100	26	676
2	11001	11001	25	625
3	11011	11011	27	729
4	10000	10100	18	324
Sum				2354
Avg				588.5
Max				729

17/05/25

* Iterative Deepening Search (IDS) or Iterative Deepening Depth First search.

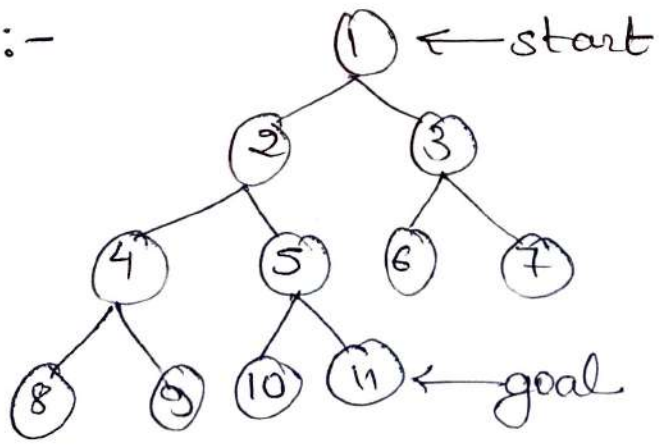
Iterative deepening search is a general strategy, often use in combination with depth first search, that finds the best depth limit. It does this by ~~rationally~~ gradually increasing dynamic first 0 then 1 then 2 and so on. until a goal is found. This will occur when the depth limit reach d, the depth of the shawllest goal node.

Algorithm :-

Step I: set ~~search~~ SEARCH-DEPTH = 1

Step II: conduct a Depth first search to a depth of search depth. if the solution path is found then return it. Otherwise, increment search depth by 1 and goto step II.

example :-



depth 1 - 1

depth 2 - 1 2 3

depth 3 - 1 2 4 5 3 6 7

depth 4 - 1 2 4 8 9 5 10 11 6

* Simulated annealing :- variation of hill climbing

annealing = change metal shape by high heating and then putting in desired shape and slowly cooling it

Simulated annealing is a variation of hill climbing in which at the begining of the proces, some down hill moves may be made. The idea is to do enough exploration of the whole space early on so that the final solution is relatively insensitive to the starting state. This should lower the chances of getting caught at local maximum, plateau or a ridge.

Here, we attempt to minimize rather than maximize the value of objective function thus the actual process is ~~valid~~ valley descending rather than hill climbing.

Algorithm -

The algorithm for simulated annealing is only slightly different from hill climbing the differences are :

1) The annealing schedule must be maintain
2) Moves to worse state maybe accepted
3) It is a good idea to maintain, in addition to the current state the best state count ~~far~~. then if the final state is ~~far~~ the worse then the earlier state, The earlier state is still available.

* 8-puzzle problem (EPP) :-

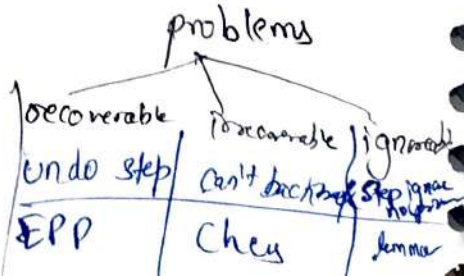
The 8 puzzle problem, is a square tray in which 8 square tiles are placed the remaining 9 square is uncovered. Each time has a number on it. The ~~tiles~~ tile that is ~~to~~ adjacent to the Blank space can be slide into that space. A game consisting of starting position and a specified goal position the goal is to transform starting position into the goal position by sliding the tiles around.

inital configration

1	2	3
5	6	
7	8	4

goal configuration

1	2	3
5	8	6
	7	4



$f(n) = g(n) + h(n)$

$g(n)$ = step/height.
$h(n)$ = how many tiles are wrongly placed