8 Puzzle Problem: { Without Heuristic }.

Start State

1	2	3
	4	6
7	5	8

1	2	3
4	5	6
7	8	

goal state

L R U D

L

1	2	3
4		6
7	5	8

	2	3
1	4	6
7	5	8

1	2	3
7	4	6
-	5	8

L R U D

1	2	3
	4	6
7	5	8

1	2	3
4	6	
7	5	8

1		3
4	2	6
7	5	8

1	2	3
4	5	6
7		8

L R U D

1	2	3
4	5	6
7	8	

Heuristic in AI (Rule of Thumb): { What, why, how?.

Search Space Possible	15 puzzle problem	24 puzzle prob.
8 puzzle problem	10^{13}	10^{24}
3^{20}		

(Non polynomial) → Polynomial.

Exponential time.

How: 1. Euclidian Distance
2. Manhatten Distance
3. Misplaced tiles

#A Heuristic func^{n} is a technique used to solve problems quickly by making assumptions/guesses. It helps to find a quick sol^{n} without exploring every possibility. In maths, heuristics are used to simplify problems like assuming values to solve questions faster. Ex → Assuming x = 100 for cost - price calculations.

Heuristics Values in AI:

Heuristics help make uninformed decisions by providing a value that indicates how close a particular state is to be goal-state. Heuristics values are used to guide such process towards the most promising space in a search space.

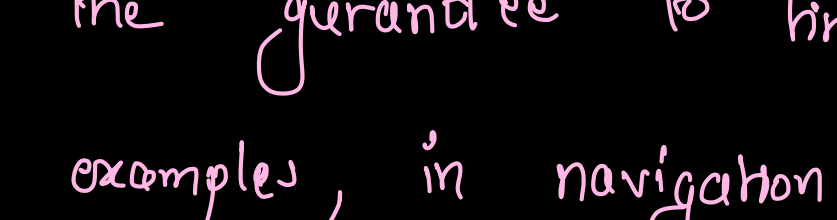
Why do we use Heuristics:

1. Unlike uninformed search (blind search), Heuristics provide a guide to solve the problem more quickly.
2. It gives the guarantee to find out the good soln.
3. Real life examples, in navigation Heuristics can guide you towards the shortest route without having to explore all possible paths
4. It converts non-polynomical into polynomical time.

How do we calculate Heuristic values:

* Several methods are used to calculate Heuristic values:

1. Euclidean Distance (Straight line distance)



$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

used when movement is allowed in any dirn, eg: maps.

2. Manhattan Distance:

8 puzzle problem.

S:

1	3	2
6	5	4
	8	7

G =

1	2	3
4	5	6
7	8	

Measures the total vertical and horizontal distance.
Used in problems where movement is restricted to grid like paths.

3. Misplaced tiles

Calculate the Heuristic values →

S:

1	3	2
6	5	4
	8	7

G =

1	2	3
4	5	6
7	8	

$\Rightarrow 0+1+1+2+0+2+2+0 = 8$

8 ↓ Heuristic value.

* Counts how many tiles are in wrong position compared to goal state. Example given above.

limitations of Heuristic:

- Heuristics guranttee a good soln but they do not always guranttee a optimal soln. for ex, in some cases the best heuristic path might not be optimal one. Eg, obstacles like mountain might make the straight line path in physical.

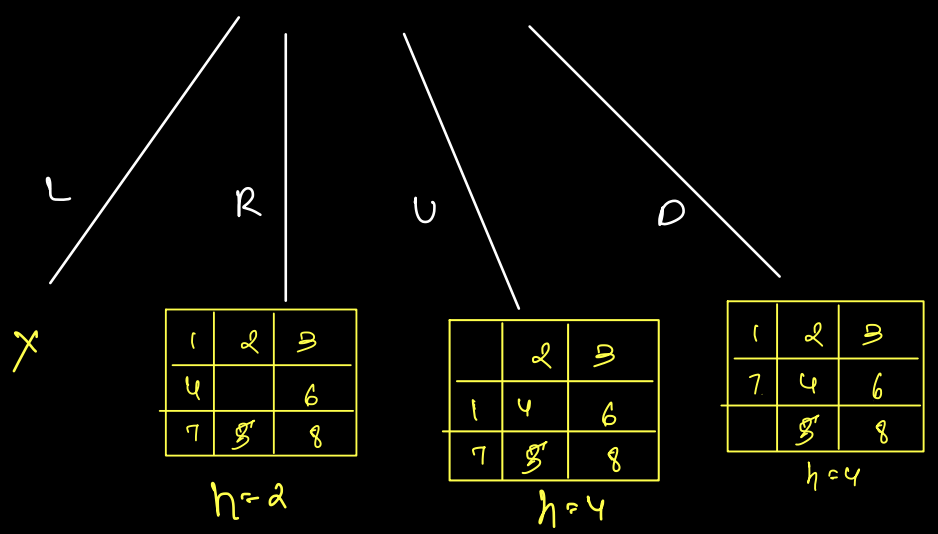
8 Puzzle Problem with Heuristic :

S :

1	2	3
	4	6
7	5	8

G =

1	2	3
4	5	6
7	8	



Difference blw Blind Search and Heuristic Search in terms of T.C:

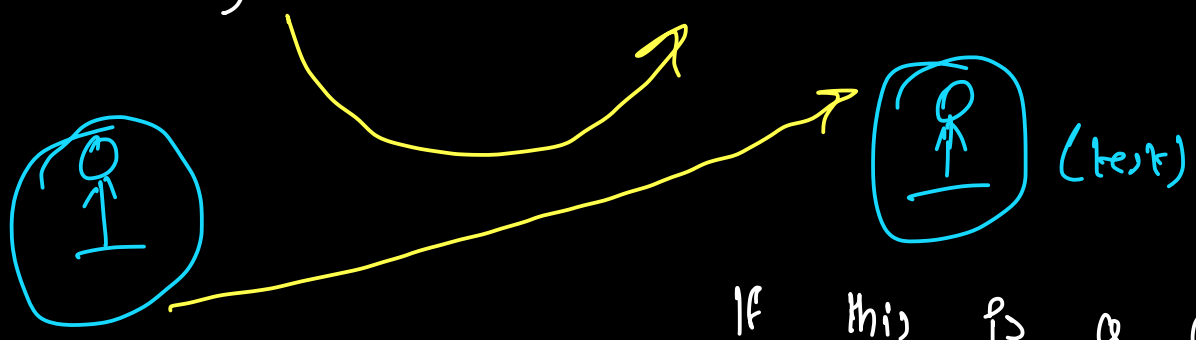
* Blind Search explores four possible states without any Heuristic guidance leading to higher T.C. $O(b^d)$ (where b → branch factor, depth).

* Heuristic focuses on most promising state i.e. the state with the lowest Heuristic value reaching the search space and improving efficiency.

Generate and Test Algorithm: { Informed Search, DFS with backtracking }

GENERATE + TEST

(It generates a possible soln) + (It tests the generated soln)



If this is a actual soln then it quits otherwise generate a next possible soln.

- It is the simplest method than any other methods.

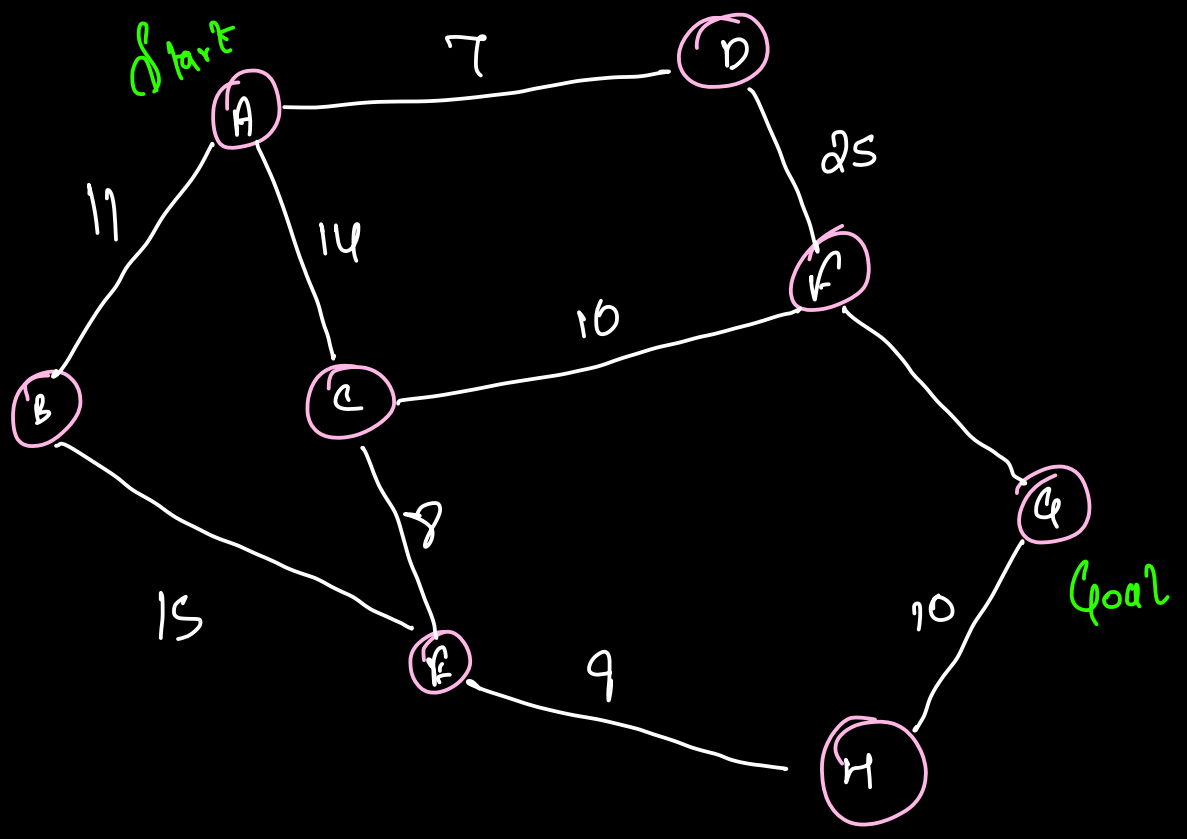
Algo:

1. Generate a possible soln.
2. Test to see if its a actual soln.
3. If a soln is found then it quits otherwise go to step 1.

Properties of Good generates:

1. Complete. (It gives the definite soln)
2. Non-redundant
3. Informed search

Best First Search Algo: {Informed Search, Heuristic}



A → G = 40
B → G = 32
C → G = 25
D → G = 35
E → G = 19
F → G = 17
H → G = 10
G → G = 0

} h

Algorithm:

Let 'OPEN' be a priority queue containing initial state.
Loop
    If OPEN is empty then return false
        Node ← Remove ← First (open)
If Node is a goal then return the path from initial to Node
else generate all successors of Node & put the newly generated Node into OPEN according to their f values (heuristic value)

END LOOP.

Hill Climbing Algo:

- It is based on the greedy method.
- Best first search algo, greedy for the heuristic value.
- It has the guarantee for giving a good soln but it may or may not give the ophimal soln.

Complexity:

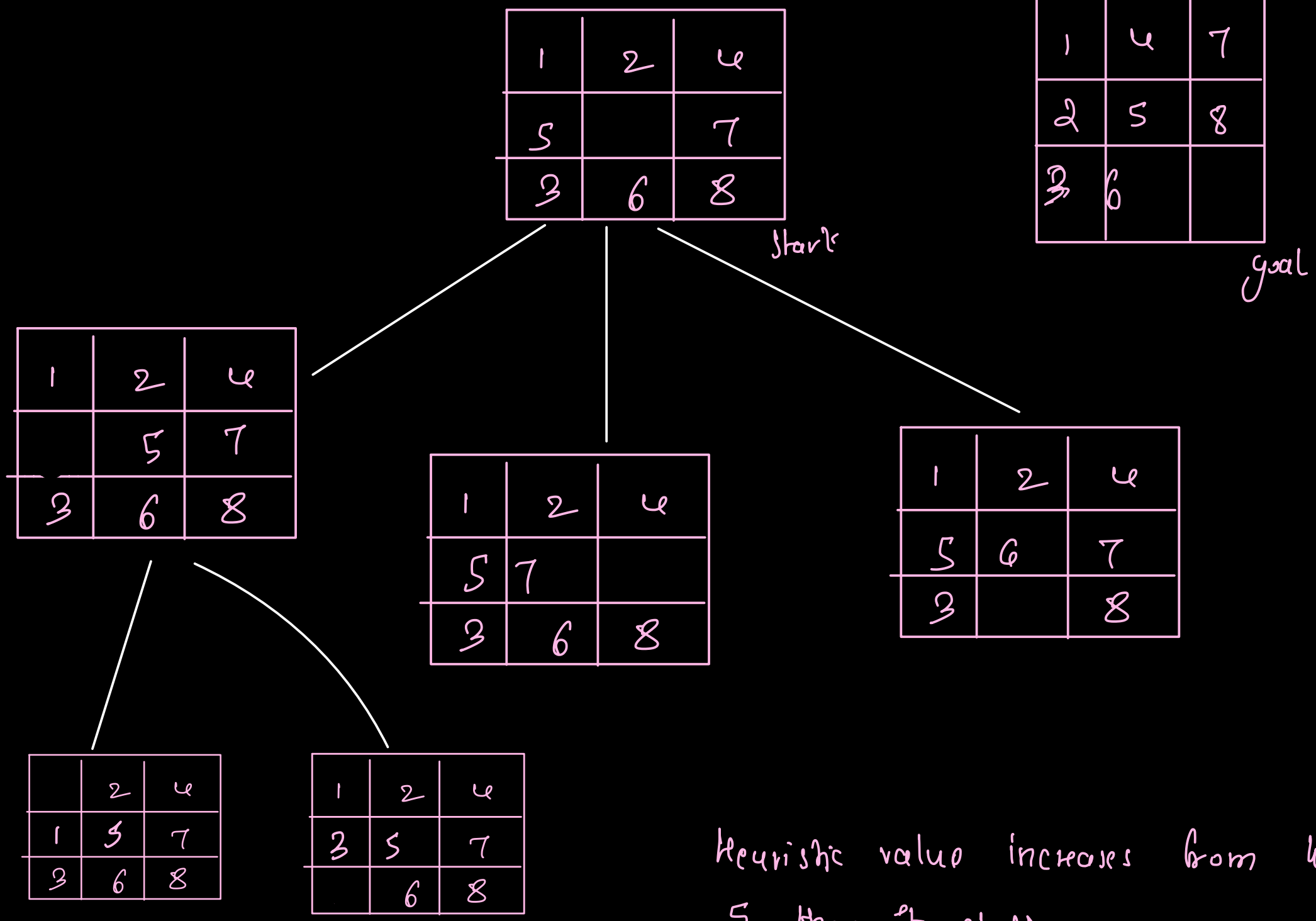
- Best Case or Avg Case or Nrml Case → Its time and space complexity performs much better than uninformed search algo
- Worst Case → Its some as uninformed search algo.

Hill Climb Algorithm:

1. Local Search Algorithm
2. Greedy Approach [Best Search]
3. No backtrack

* Algorithm:

1. Evaluate the initial state
2. Loop until a soln is found or there are no operators left - select and apply a new approach.
3. Evaluate the new state, if goal then quit.
4. If better than current stage then it is a new current stage.
5. It is blindfolded



Problems:

1. local maximum – Algo gets stuck at a peak which is not globally optimal. It thinks it has achieved a max value.

- All neighbor states have the same heuristic value.
- No clear dirⁿ to move algo halts.

- Higher value states exist but lies directly or indirectly
- Hill climbing can't reach it bcuz it only explores one dirⁿ

A* Algorithm [Informed Searching]:

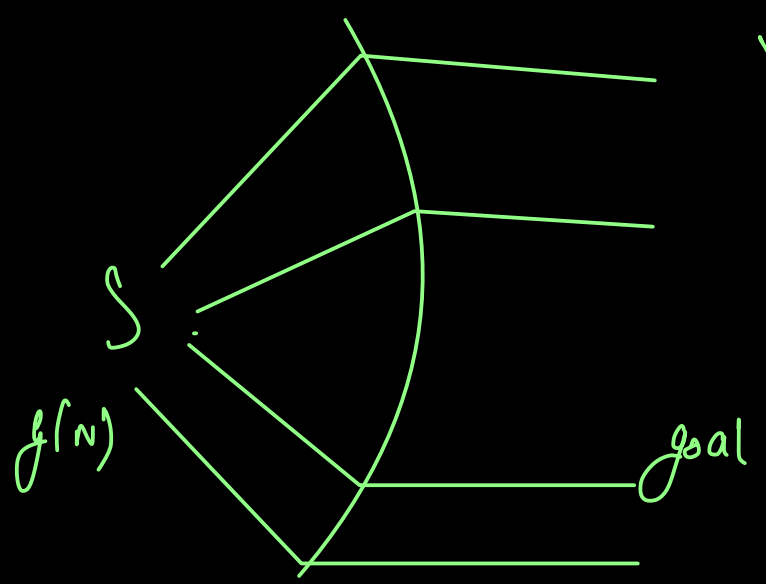
* → Admissible

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

$h(N)$ → Estimation cost from n to goal node.

$g(N)$ → Actual cost from start node N

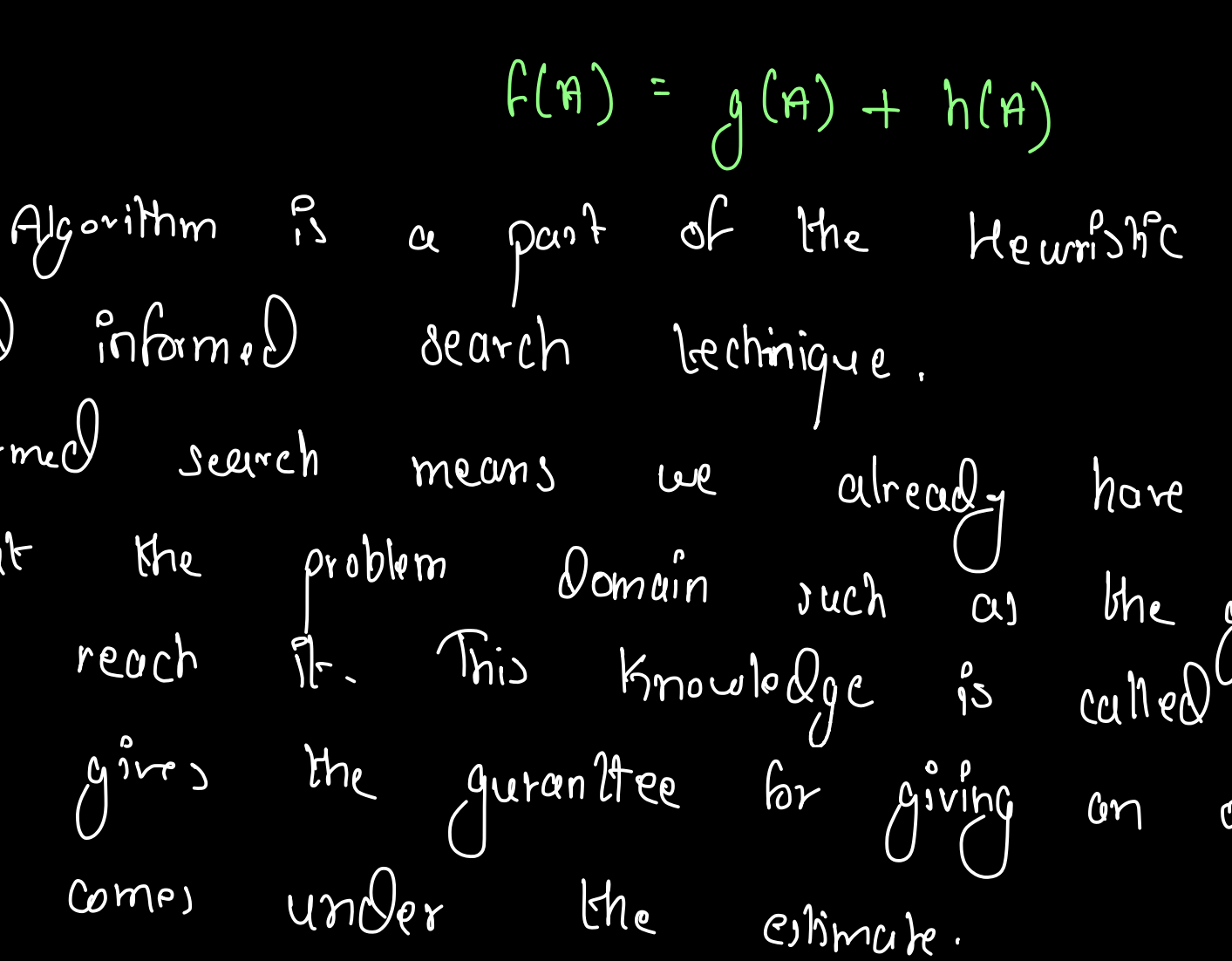
$f(N)$ → Total cost func^n.



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

- A* Algorithm is a part of the Heuristic Search Technique and informed search technique.
- Informed search means we already have some knowledge about the problem Domain such as the goal state or how to reach it. This knowledge is called the heuristic value.
- It gives the guarantee for giving an optimal soln when it comes under the estimate.

Example:



{ The value at edges = actual cost

Table :

$S \to G = 14$

$B \to G = 12$

$C \to G = 11$

$E \to G = 4$

$F \to G = 11$

$D \to G = 6$

$G \to G = 0$

S	14
B	12
C	11
E	4
F	11
D	6
G	0

* Step (1) ⇒ • $S \to B$

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

$= 4 + 12 = 16 \to$ Total cost

(4 ↓ Actual, 12 ↓ Estimated)

• $S \to C$

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

$= 3 + 11 = 14$

{Select the min Total cost. ie. 14}

• Step 2 => $SC \to D$

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

$= 7 + 3 + 6 = 16$

$SC \to E$

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

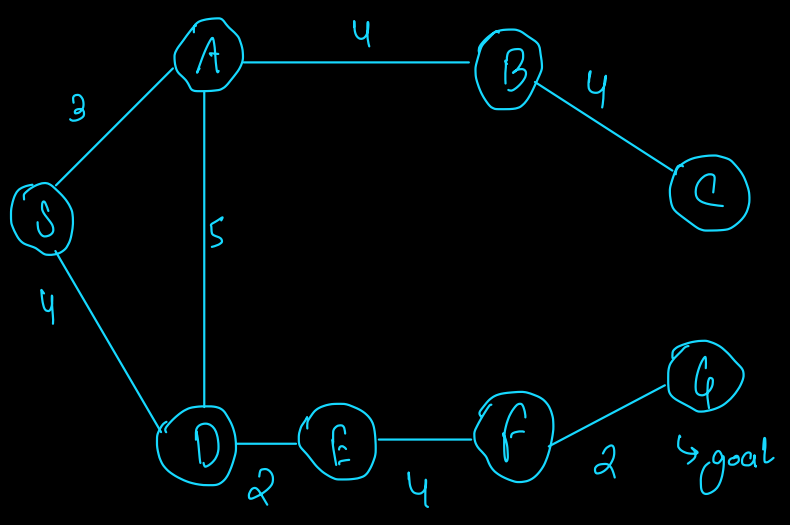
$= 10 + 3 + 4 = 17$

Step 3 => SCD → E

$$f(n) = g(n) + h(n)$$
$$= 12 + 4 = 16$$

Step 4 => SCDE → G

$$f(n) = g(n) + h(n)$$
$$= 17 + 0$$
$$= 17$$



S	11.5
A	10.1
B	8.0
C	3.4
D	9.2
E	7.1
F	3.5
G	

Step 1 → S → A

$$f(n) = g(n) + h(n)$$
$$= 3 + 10.1$$
$$= 13.1$$

S → D

$$f(n) = g(n) + h(n)$$
$$= 4 + 9.2 = 13.2$$

Step 2: SA → B

$$f(n) = g(n) + h(n)$$
$$= 3+4 + 8.0$$
$$= 15.0$$

SA → D

$$f(n) = g(n) + h(n)$$
$$= 3 + 5 + 9.2$$
$$= 17.2$$

Step 3 → JDE → F

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

=

Step 4 → SDEF → G

Final path → S → D , D → E , E → F , F → G

Total value = 12.

How to make base class Admissible:

$h(n) \leq h^*(n)$ ⟶ Underestimation

$h(n) \geq h^*(n)$ ⟶ Overestimation

$h(n)$ → Estimated value

$h^*(n)$ → Actual value / optimal value

S → n → G

S ↓ Start

G ↳ Goal

Underestimation:

If the heuristic value $h(n) \leq$ the actual cost to the goal node is called underestimation.

Overestimation:

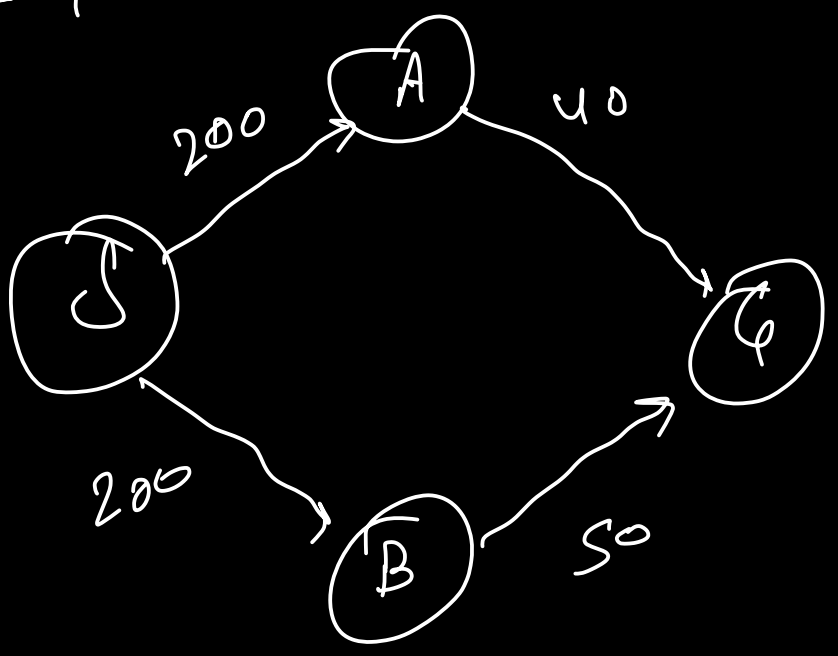
If the heuristic value $h(n) \geq$ the actual cost to the goal node is called overestimation. Overestimation can lead to incorrect paths. and suboptimal soln. In this case, it may or may not give an optimal soln. but in underestimation it

guarantees an optimal soln.

Real life example:

- Suppose the actual price of a laptop is 30k. but you had an expected price of 40k. You might quickly purchase if the shopkeeper offers it for 30k.

Graph:



h(A) = 80 , h(B) = 70

A → f(n) = g(n) + h(n)

⇒ f(A) = 200 + 80 = 280

f(B) = 200 + 70 = 270

* Examples of:

Tic-tac-toe, Chess, 8 Puzzle Game

In game playing two major algos are used → min-max algo, α-β

These algos work on a

Game tree Concepts:

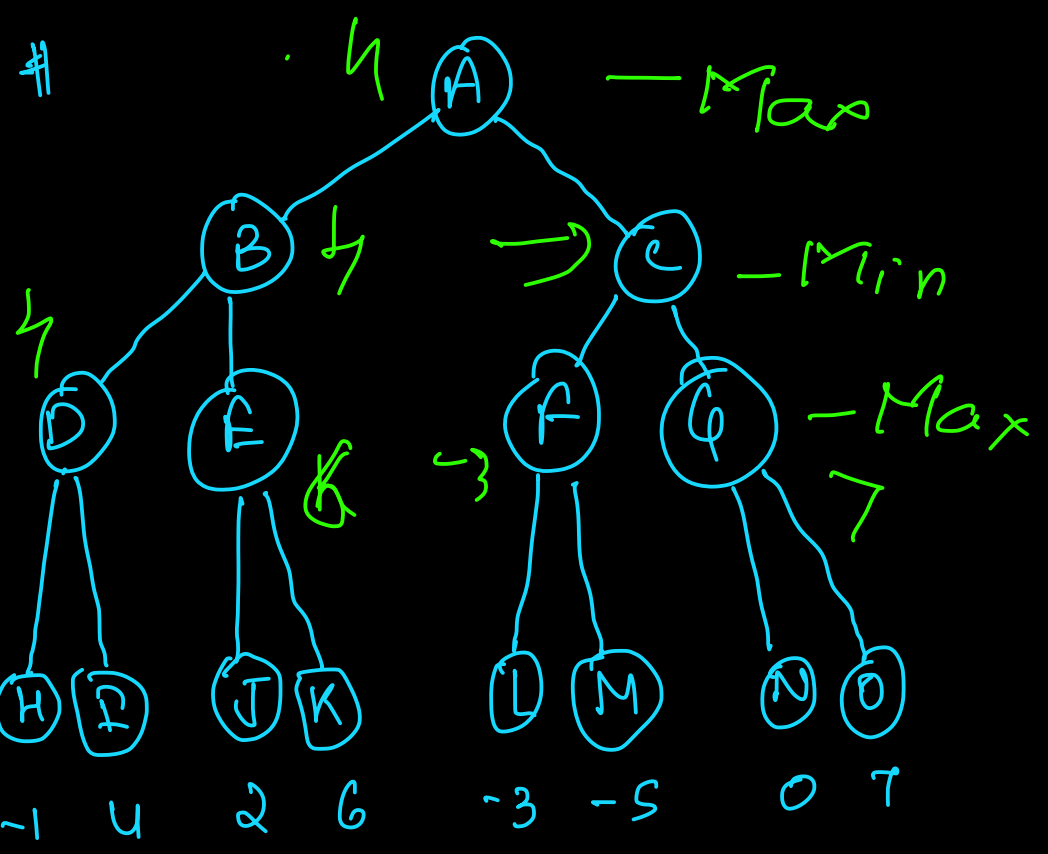
- The tree shows all possible moves players can make.
- The player trying to maximise their chance of winning is called max.
- The opponent trying to minimise max chance of winning is called min.
- The game alternates btw max and min moves at diff. levels of the tree.
- Complexity → $O(b^d)$

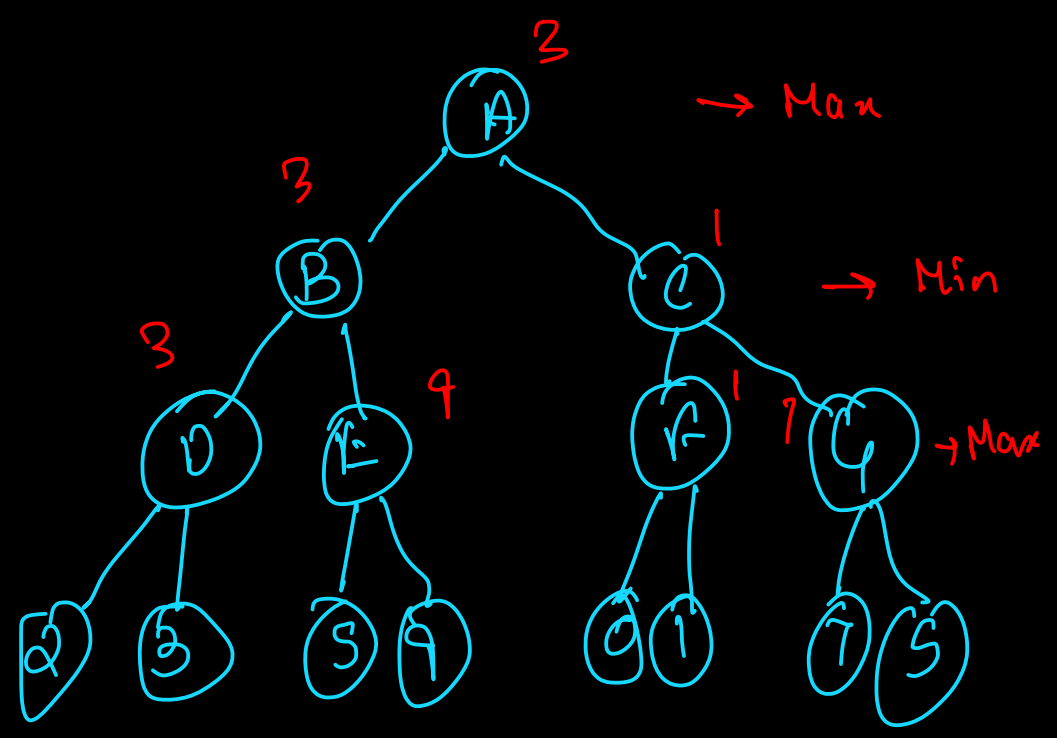
  b → branch factor

  d → depth
- The algorithm goes from root node to the node. Calculates values and terminals and then propagates these values back to the best move.

Best Move Strategy :

- Two players compete max and min to make the best move to maximise the chance of winning.
- Min tries to minimise max chances and prevent





A → B → D

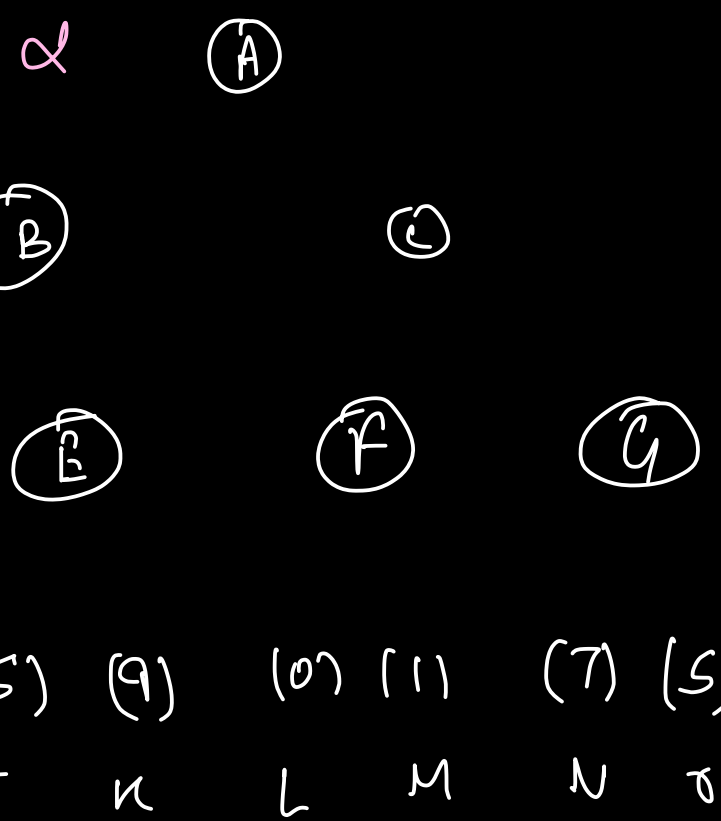
A → B → D → F

α-β Pruning :

- Its an advanced version of min-max algo.
- It aims to improve the T.C. and performance of min-max by pruning unnecessary branches in the game tree.
- Min-max explores all nodes in the game tree leading to a T.C. of $O(b^d)$
- Min-max evaluates every possible move which can be v. time consuming.
- But in α-β pruning, it cuts off the searech fewer nodes (pruning).

- It reduces the no. of nodes explored and increases efficiency.
. If a best path is alr formed, remaining path

Alpha-Beta Algo:



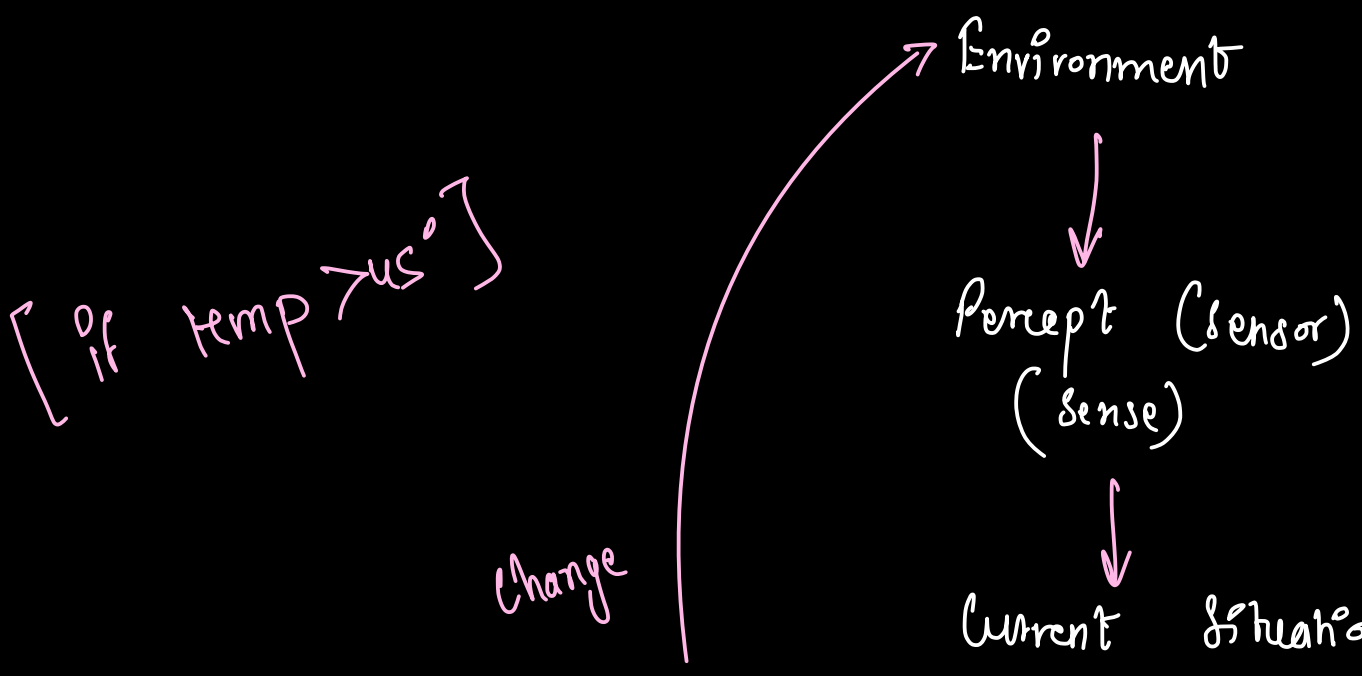
α → max
β → min.

- P → Performance
  E → Environment
  A → Actions
  S → Sensors.

Types of Agents:

1. Simple Reflex Agents.
2. Model Based ,,
3. Goal ,, ,,
4. Utility Based Agents.
5. Learning Agents.

Simple Reflex Agent:



(what the world is like now)

↓

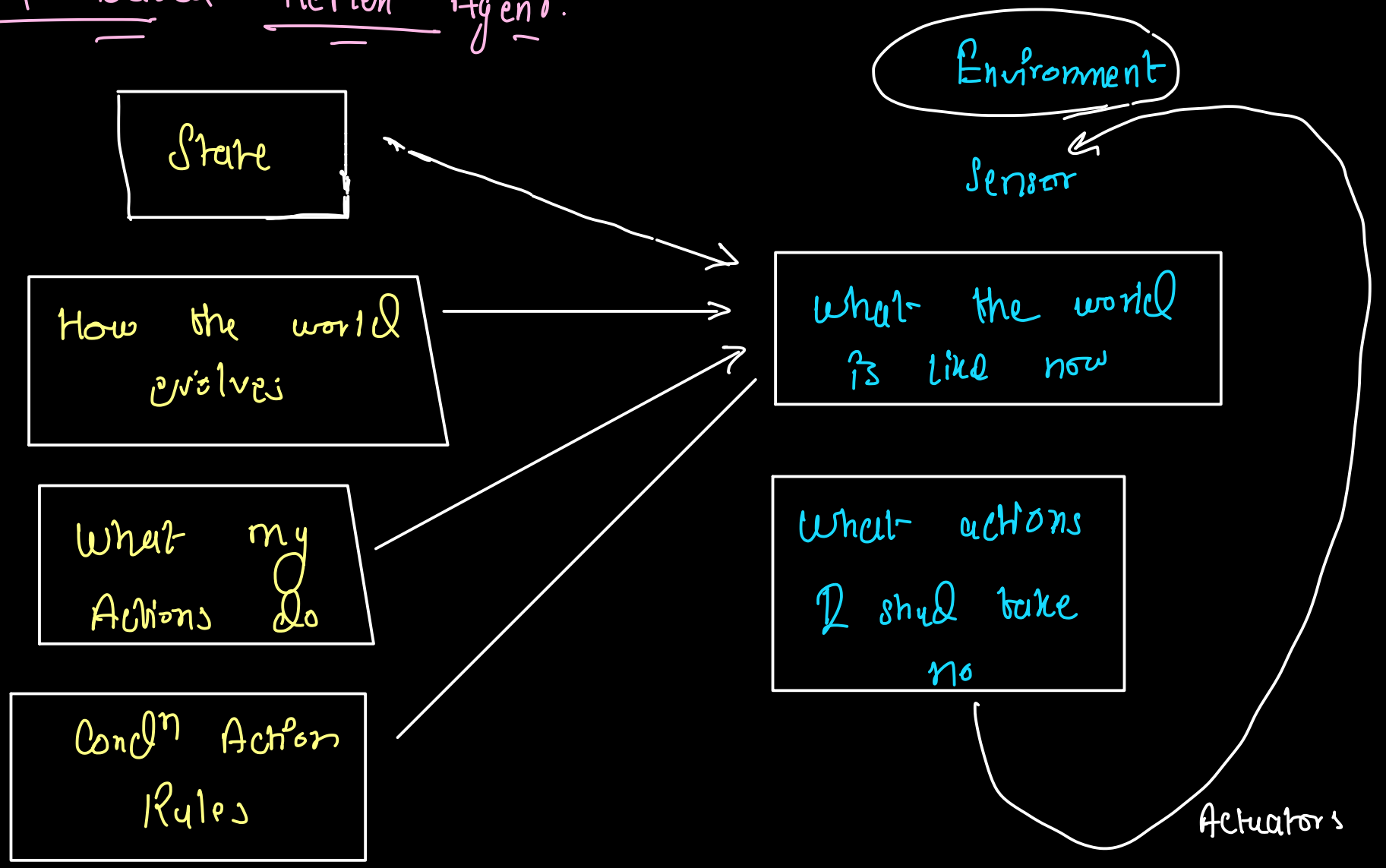
If-then (condn)

↓

Action

- Act on the basis of current perception.
- Ignores the rest of percept history.
- Based on if-then rules.
- Environment shud be fully observable.

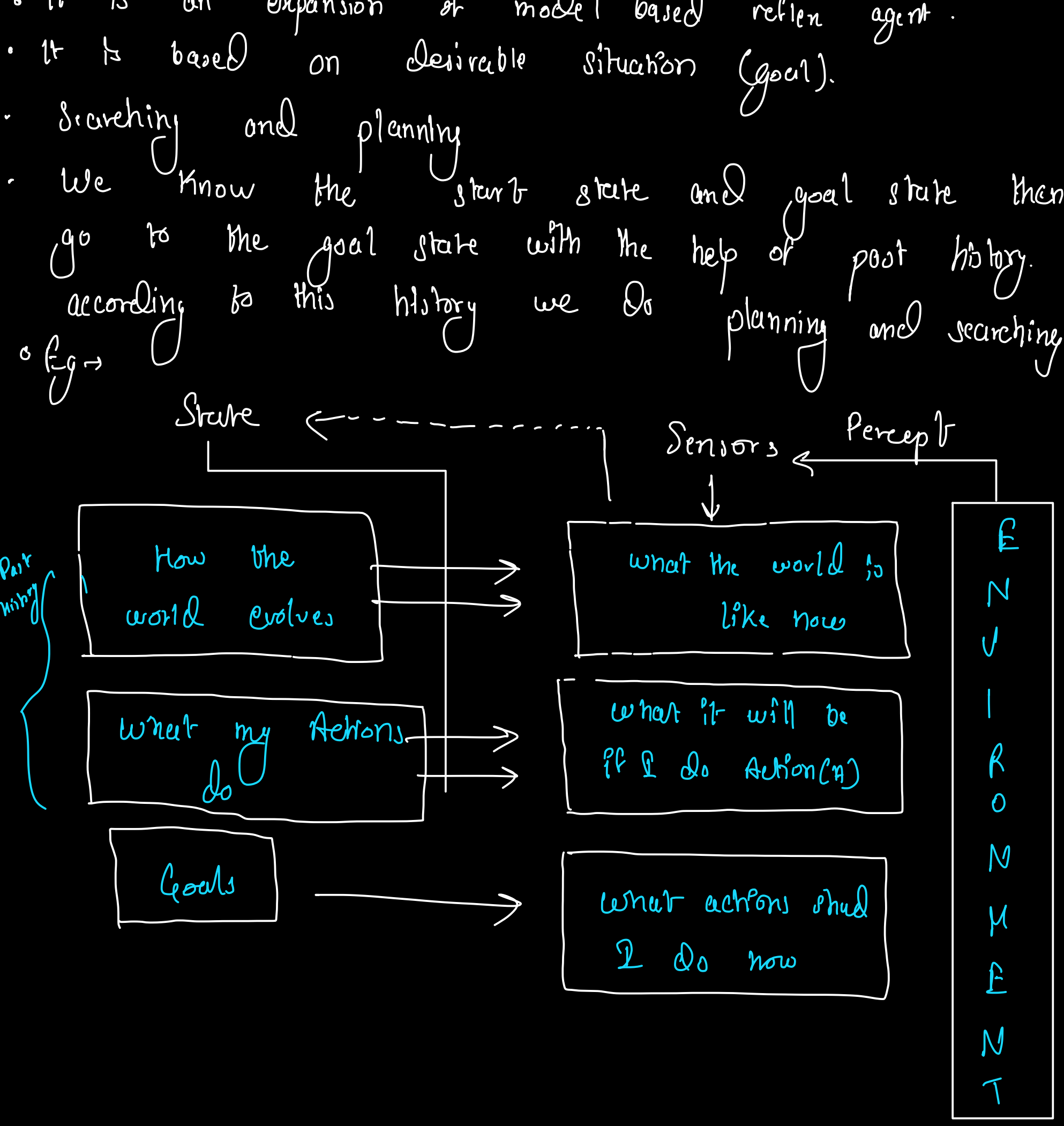
Model Based Reflex Agent:



- Model based means knowledge based made by historical or past condn then it makes the decision.
- The environment shud be partially observable environment.
- Stores percept history (internal model).
- It works on the modelling.

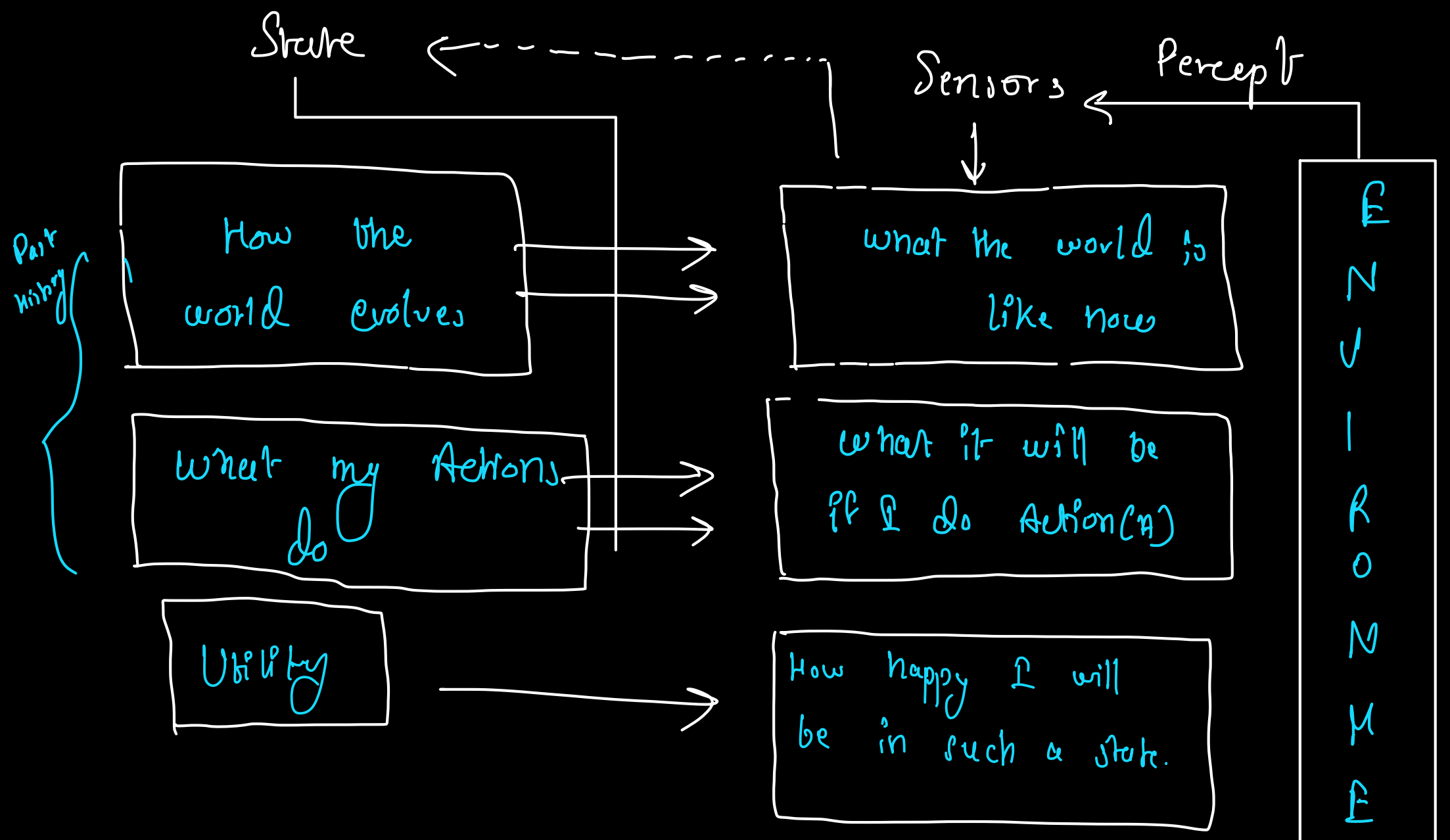
Goal Based Agent:

- It is an expansion of model based relex agent.
- It is based on desirable situation (goal).
- Searching and planning
- We know the start state and goal state then we go to the goal state with the help of past history. and according to this history we do planning and searching.
- Eg →

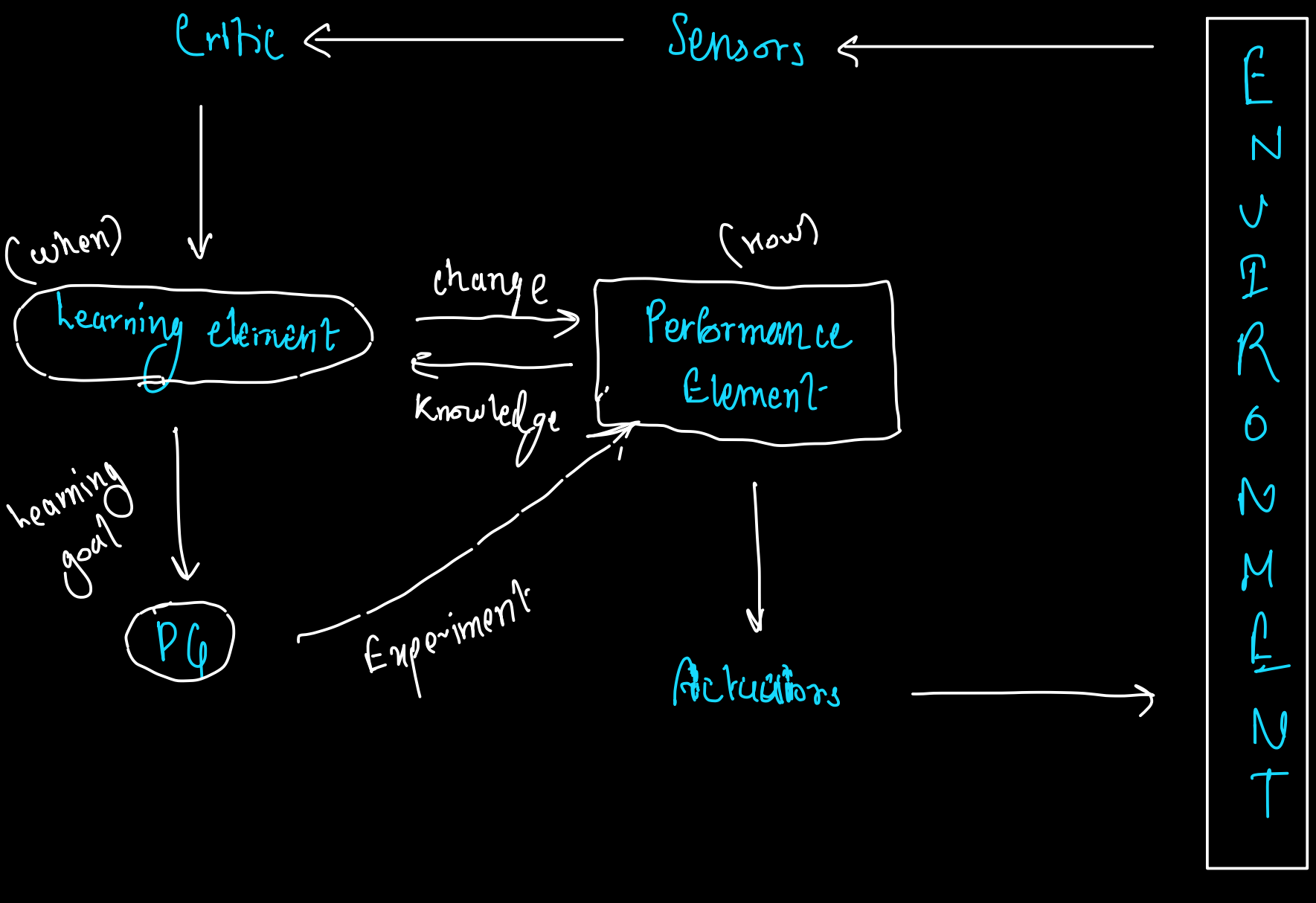


Utility Based Agent:

- Focus on utility not goal.
- Defined the utility funct.
- Deals with happy and unhappy state.



learning Agents → It learns overtime from experience.

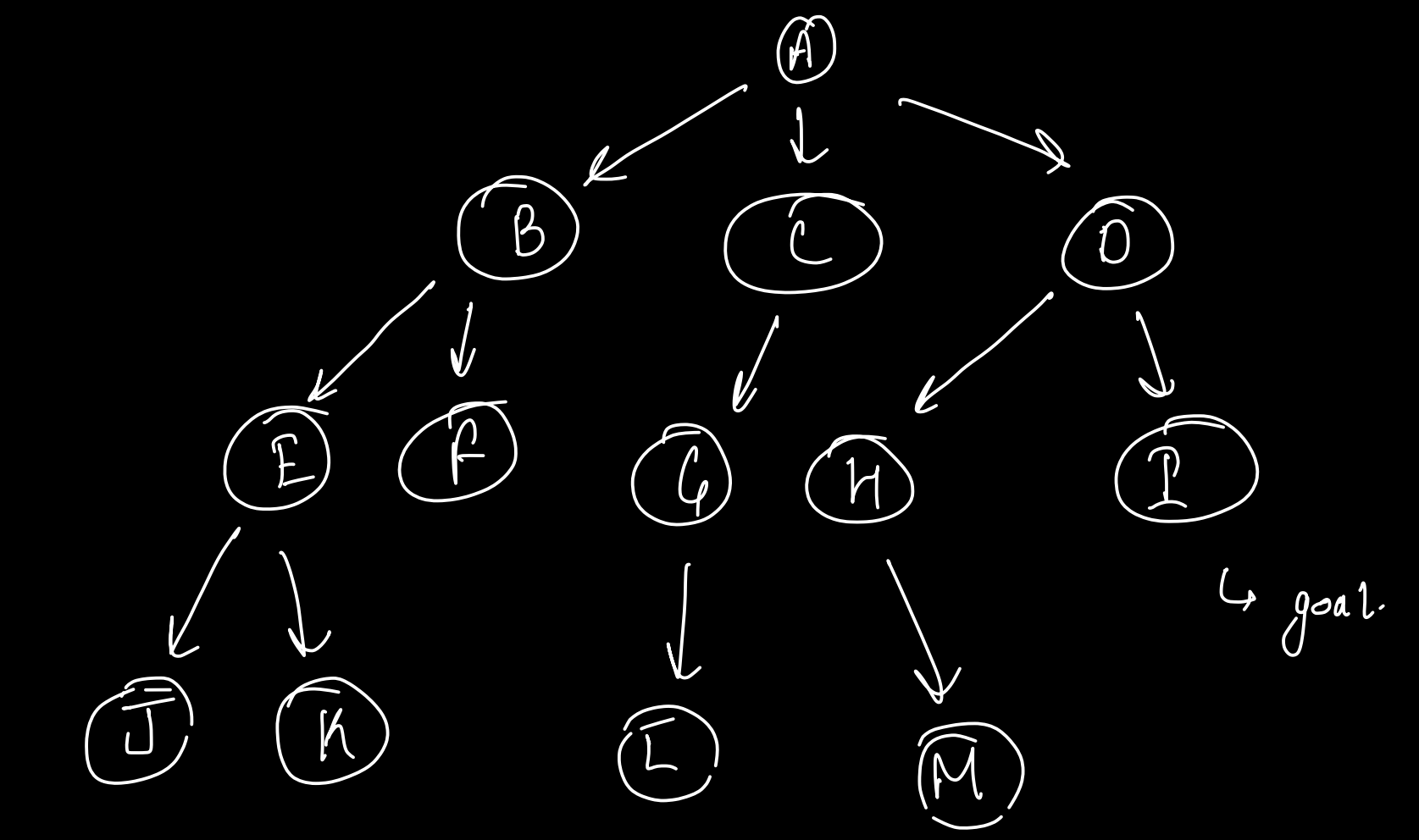


- Learning agent is a type of intelligent agent that learns over time from experience.
- Critic => It takes input from env and then pass the feedback to the learning element. It observes the environment.
- LE => Stands for learning element. It focuses on when to use.
- PE => " performance ". It focuses on how to do anything.
- PG => Problem Generator. It generates the problem. Ex: takes new route again and again.
- Actuators => Those who perform actions.

Iterative Deepening Search:

↳ Iterative Deepening Depth First Search (IDDFS)

- It combines both the benifits of DFS (Depth) & BFS (Breadth)
- It ensures completeness & optimality.
- Searches to a limited depth increasing the limit with each iteration.
- Ensures exploration of all nodes at each depth before going deeper
- Ex:

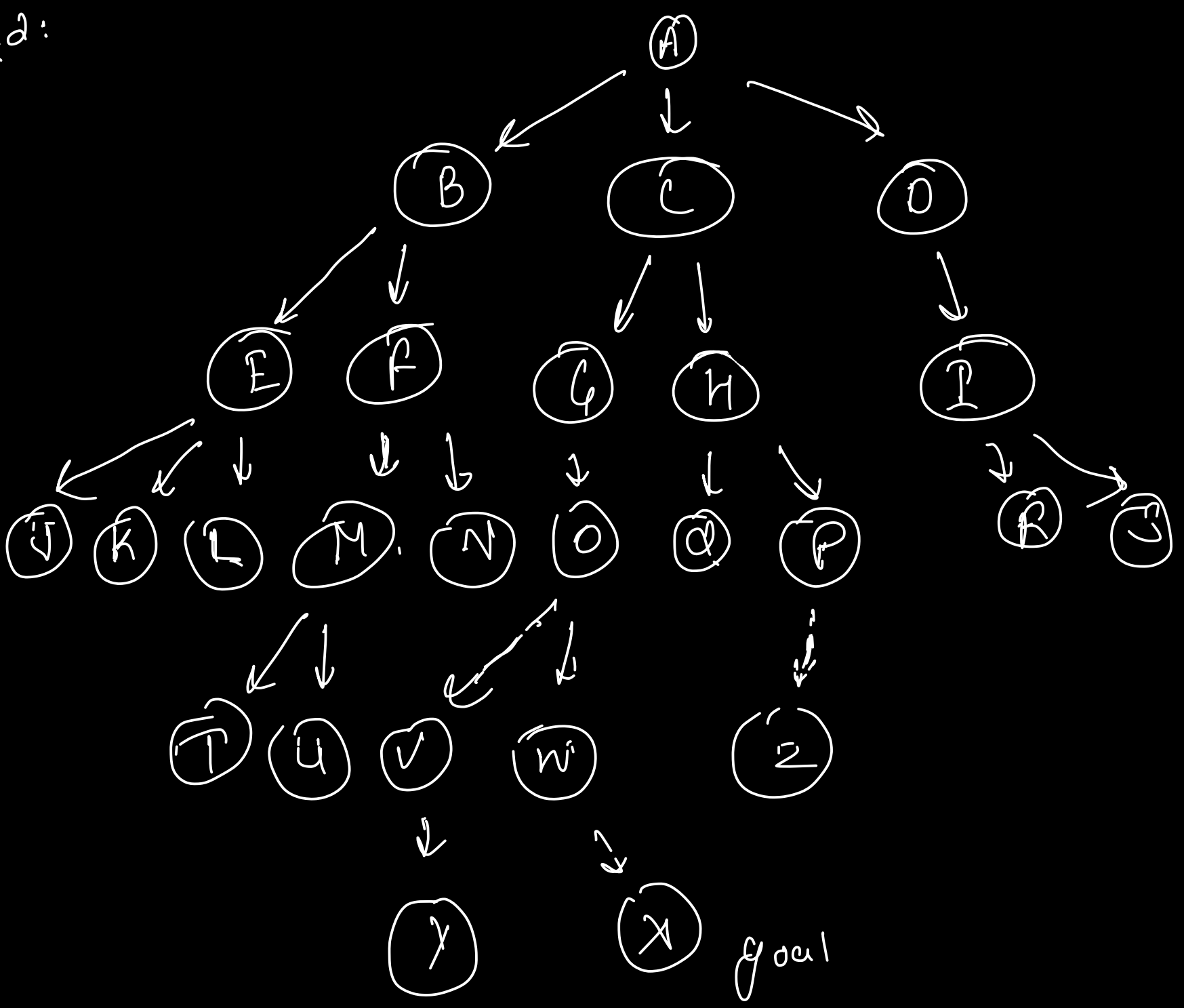


P_0 = A

P_1 = A → B → C → D

P_2 = A → B → E → F → C → G → D → H → I.

End:



$I_0 = A$

$I_1 = A \to B \to C \to D$

$I_2 = A \to B \to E \to F \to C \to G \to H \to D \to I$

$I_3 = A \to B \to E \to J \to K \to L \to F \to M \to N \to G \to O \to H \to Q \to P \to I$
$\to R \to J$

$I_4 = A \to B \to E \to J \to K \to L \to F \to M \to T \to U \to N \to C \to G \to O$
$\to V \to W \to H \to Q \to P \to Z \to D \to I \to R \to J$

$I_5 = A \to B \to E \to J \to K \to L \to F \to M \to T \to U \to N \to C \to G \to O \to V$
$\to Y \to W \to X$

↓
goal.

A* Algorithm in 8 puzzle Problem:

- The main aim is to find the most cost effective path to reach the goal state.

-

2	8	3
1	6	4
7		5

initial state

1	2	3
8		4
7	6	5

goal state

L R D

2	8	3
1	6	4
	7	5

2	8	3
1	6	4
7	5	

2	8	3
1		4
7	6	5

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

$= 5 + 1$

$= 6$

L R U D

2	8	3
	1	4
7	6	5

f(3)

2	8	3
1	4	
7	6	5

f(6)

2		3
1	8	4
7	6	5

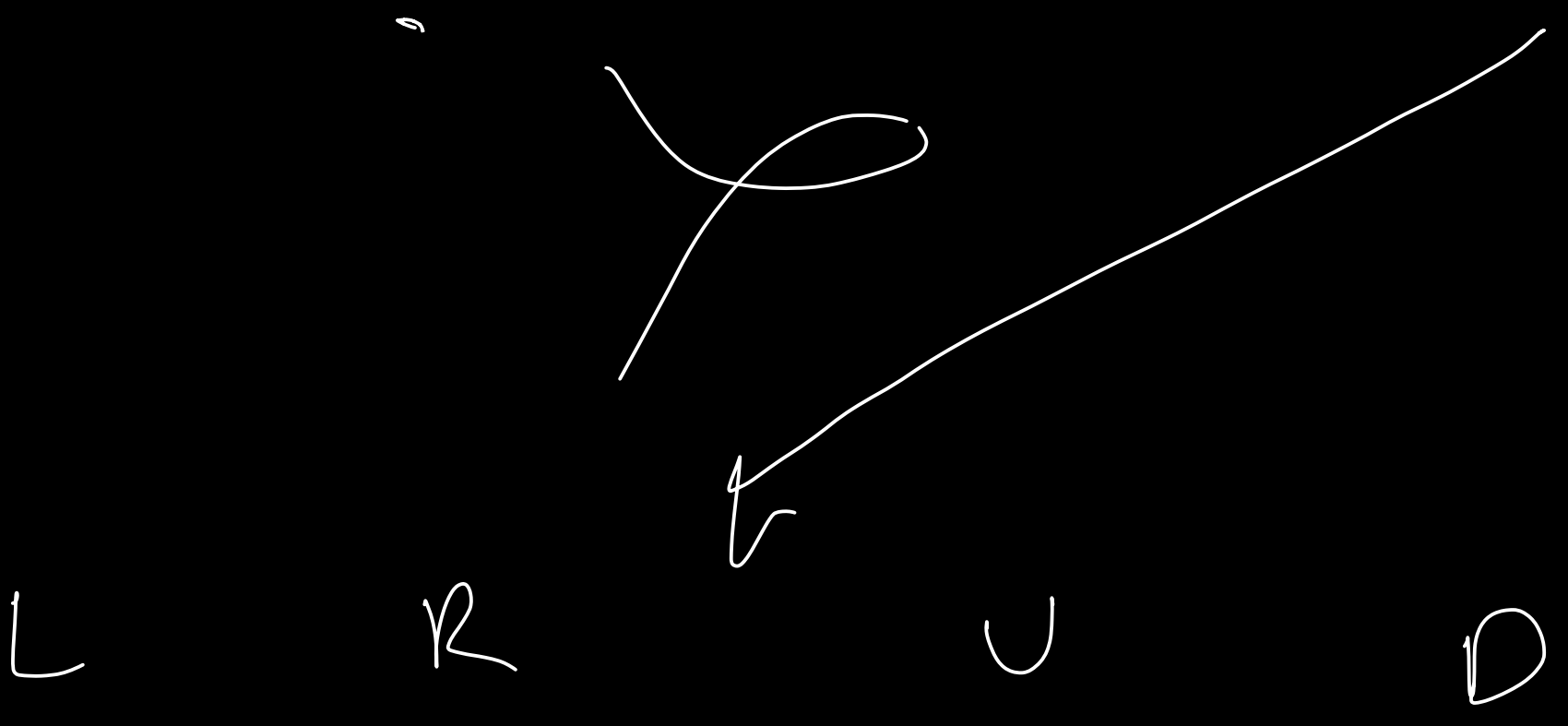
f(5)

L R U D

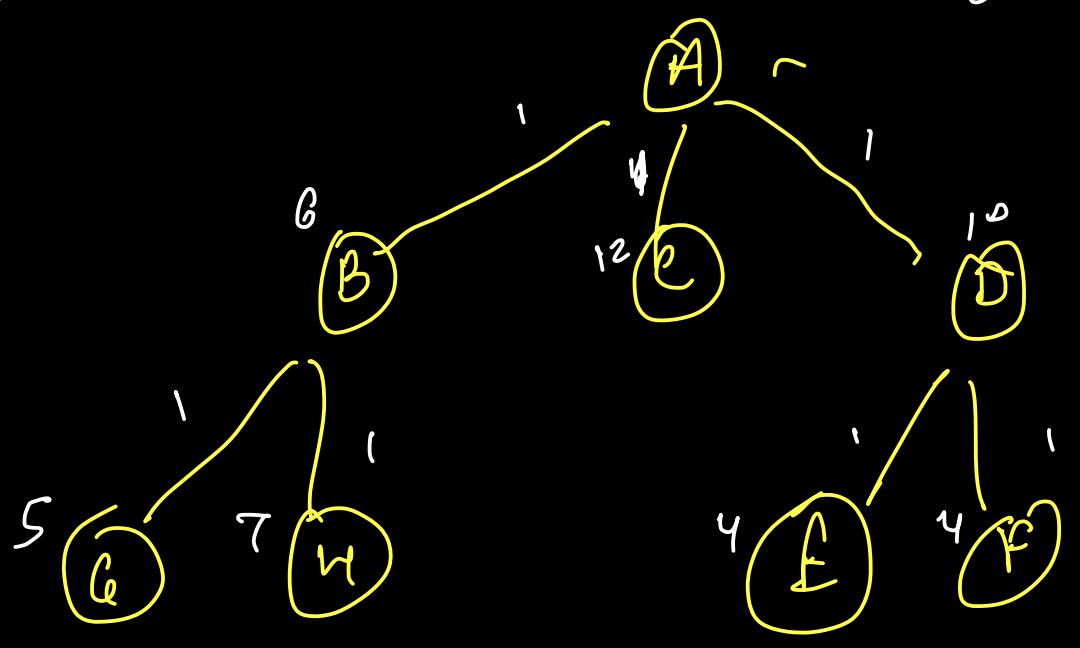
2	8	3
1		4
7	6	5

	8	3
2	1	4
7	6	5

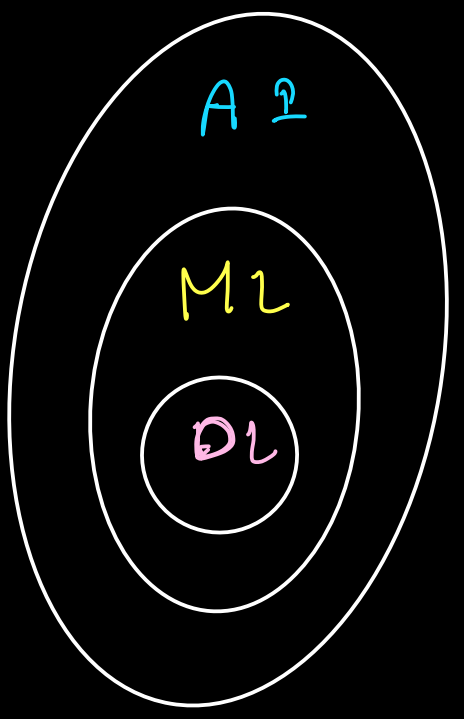
2	8	3
7	1	4
	6	5



* In this complex prob. is broken down into smaller pieces and the soln is found.
* A* and AO* both work on Best First search. Both are informed searching
- On the basis of heuristic value we find the soln
- A* gives optimal soln if we work on unestimate.
- AO* will always give optimal soln, there is no guarantee
- " explore all paths once it get a soln.

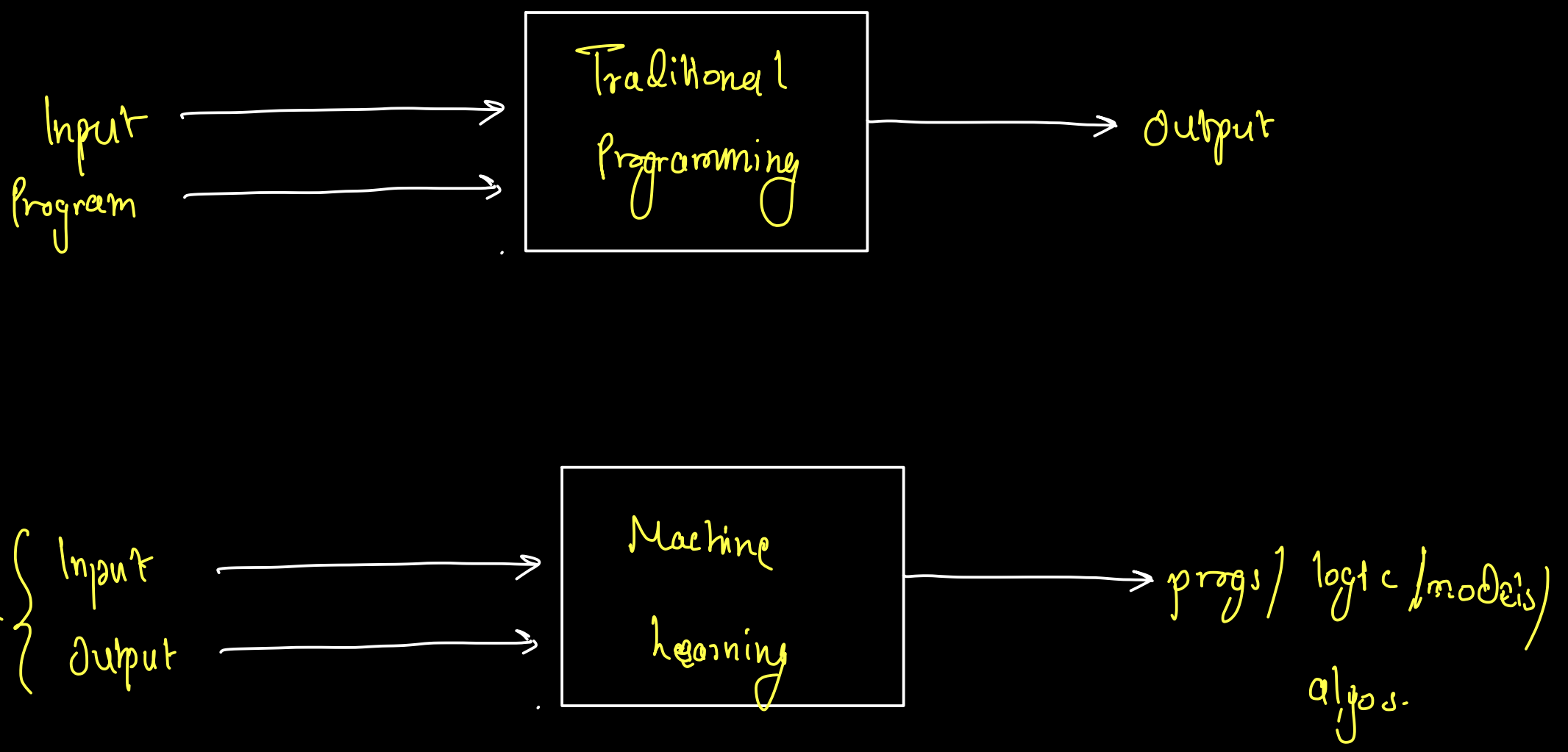


Machine learning



- ML is a branch of AI that focuses on developing models and algos that let computers learn from data without being explicitly programmed for every task.
- In simple words, ML teaches the systems to think and understand like humans while learning from the data.

How ML is diff from traditional programming:



- ML learns with the help of Data

How ML Works?

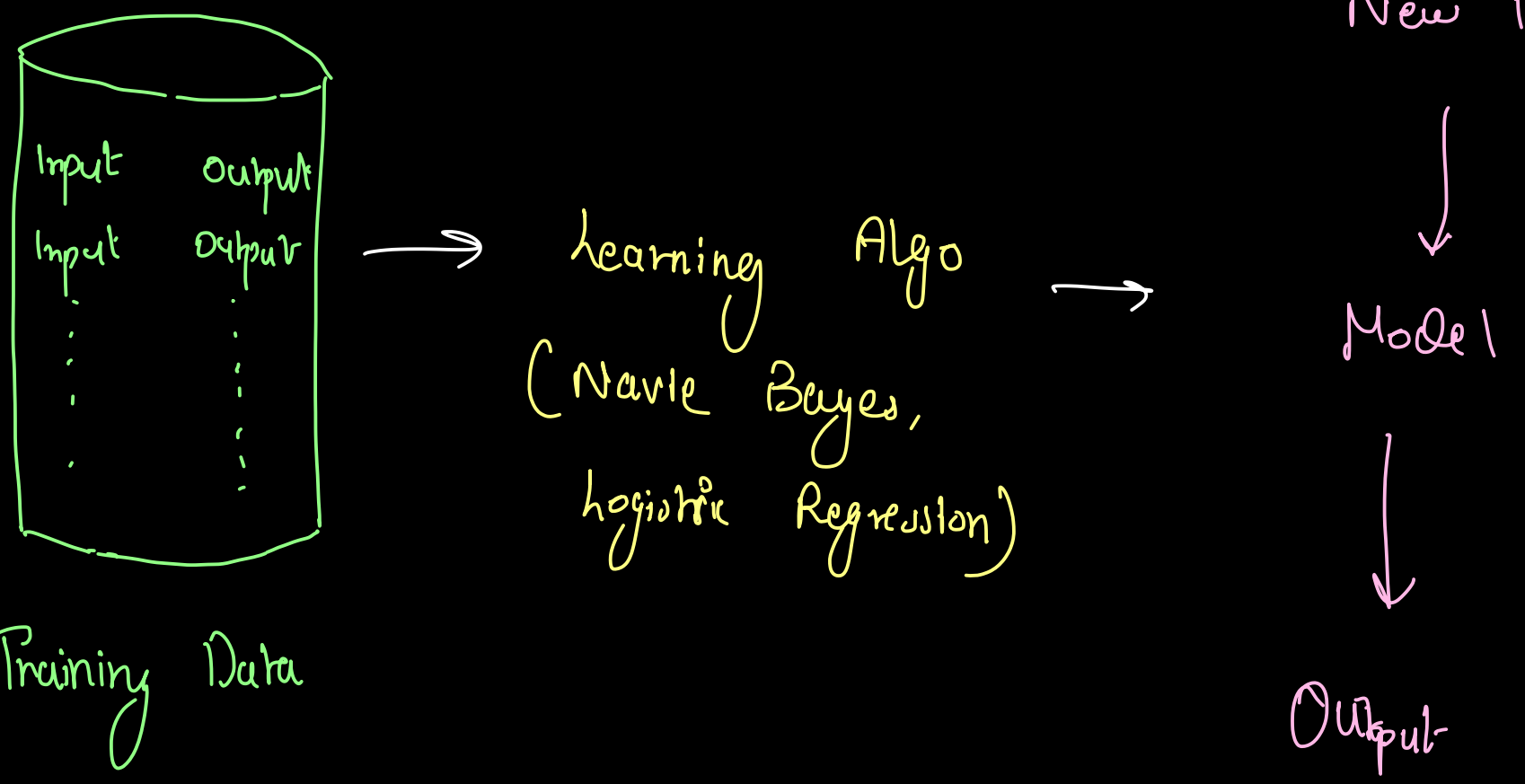
- Problems Statement
- Data Collection
- Data Cleaning
- " Analysis & Exploration
- " Modelling
- Optimisation and Deploymen

- The main of ML is to reduce the error margin

E.M. = Desired (Expected) output − Actual output

Types of ML:

- Supervised learning



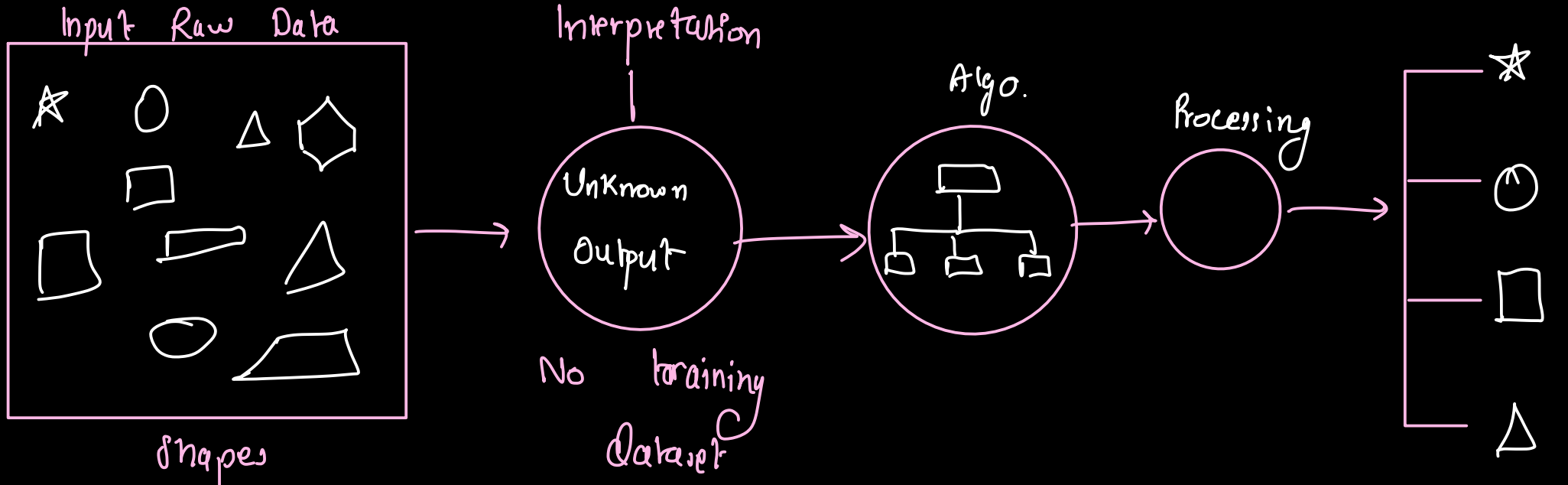
- Training Data
- Both input and Output

- It performs classification.
- In Supervised learning, models are trained on labelled data to predict or classify new, unseen data.
- A supervisor (teacher) provides both input and output, helping the machine learn from this data.

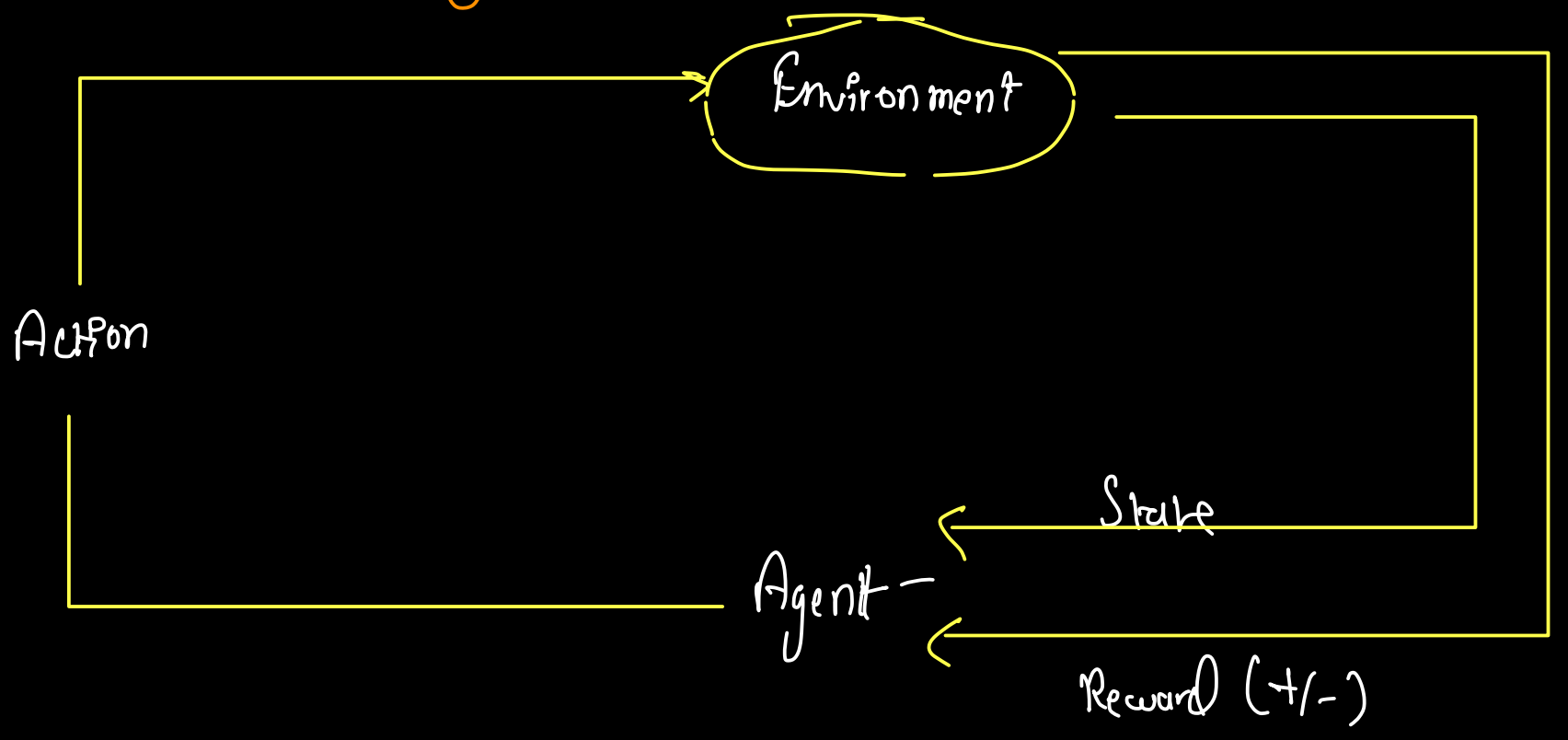
Unsupervised learning:

- learning based only on an input data without labelled outputs.
- The machine has no prior knowledge only input and it groups data into clusters.
- Algos: K Mean Clustering
- In unsupervised learning no supervisor is present

#



Reinforcement learning:



- It is a feedback based ML approach. Here an agent learns to which action to perform by looking at the env. and the results of actions.
- It is related to semi-supervised learning.
- For each current action, the agent gets +ve feedback and for incorrect action the agent gets -ve feedback or penalty.
- The agent interacts with the env and identifies the possible action it can perform.
- The primary goal of an agent is to perform actions by looking at the env and get the max. +ve rewards.
- In RL, the agent learns automatically using feedbacks without any labelled data unlike supervised learning
- RL is used to solve specific type of problems where decision making is sequential and the goal is long term.

Types of RL: → +ve feedback
→ -ve feedback.

Q) Where do we use RL?

↳ Its used in designing robots or in games.

- It is not suitable for env where complete info is present.

For eg: The problems like obj. detection, brand detection can be better solved using a classifier than by RL.

Steps of Feature Engineering:

1. EDA { Exploring Data Analysis }

↳ ① Analysis → ① How many numerical features may be there.

② How many categorical features may be there:

③ Missing values

④ Outliners.

Step 2 → Handling the missing values → Mean, Median, Mode.

Step 3 → Handling Imbalanced Dataset.

↳ Imbalanced Dataset always gives poor accuracy so its essential to convert imbalanced Dataset into balanced Dataset to improve the accuracy.

Step 4 → Treating the outliners

Step 5 → Scaling the data.

(a) Standardization

Normalization.

Converting the categorical features into numerical features.

Feature Selection:

- We select only those features that are important.

1. Co-relation
2. K- Neighbours
3. Chi- Squared Test.
4. Genetic Algorithm
5. Feature Importance {Extra tree classifier}.

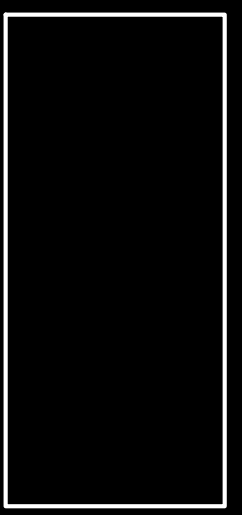
Handling Missing Values:

1. Delete the records which has missing values.
2. Create a separate model to handle missing values.
3. Statistical methods → mean, median point.

Deleting records which has missing values:

f_1	f_2	f_3	f_4
1	200	100	70
2	300	–	80
–	400	–	–
4	600	20	10
5	800	10	15

O/P



↳ In this most of the features have empty.

- When we have a huge dataset like millions of records then only its applicable. But if we have a smaller dataset, we never follow this step.

Create a separate Model:

- It takes much more time for large dataset to get it is only applicable for small dataset.

	f_1	f_2	f_3	f_4	O/P
test dataset	—	20	30	40	100
Training dataset	100	150	60	40	200
	50	80	90	100	300
	70	20	10	40	100

f_2, f_3, f_4

↳ model (training model) → O/P.

↳ Whenever we have a missing value in a feature, then we have to consider our test dataset and remaining all the datasets are training dataset

Statistical Technique (Best technique):

Eg.

10
20
30
40
30

- Mean
- Median

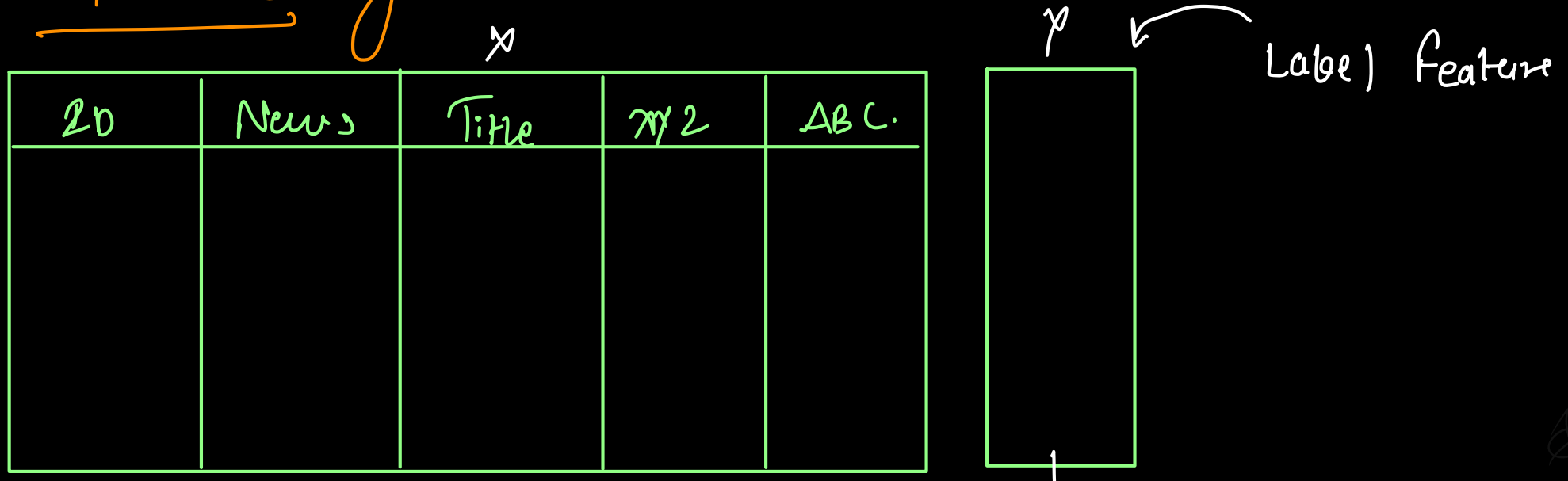
Dealing with Categorical Feature:

1) Nominal feature (No rank) [eg, States]
2) Ordinal " (Rank given) [eg→ Grades]

Techniques for Handling the Categorical Feature:

1. Label Encoding
2. Count or Frequency Encoding.
3. Ordinal Encoding { Ordinal Dataset }.
4. One Hot Encoding { Nominal Dataset }.

Label Encoding:



↳ Output

- Output is having labels, it means labels are available in classification problem then we have to apply label encoding.

Ordinal Encoding:

- If we have ordinal dataset means the dataset having any order the O.E. is applied.

* PG > UG > 12 > 10

HS → 0
UG → 1
PG → 2
PG → 2
UG → 1
HS → 0
UG → 1

HS → 0
UG → 1
PG → 2

*

Age	Gender	Review	Education	Purchase
10	Female	Good	UG → 0	Yes → 1
24	"	Avg	PG → 1	Yes → 1
26	"	Poor	PG → 1	No → 0
27	"	Poor	PG → 1	No → 0
44	"	Avg	UG → 0	No → 0

Good → 2
Avg → 1
Poor → 0

One Hot Encoding:

↳ It is applied only for nominal data.

↳ No rank in this

↳ Ex => Gender → Male
→ Female

Gender	Male	female
Male	1	0
female	0	1
Male	1	0
Male	1	0

Freq. Encoding:

- If any features having more categories, it means bicardinal features.
- Replace each category with count of how often it appears in data.

Target Encoding:

- It replaces each category with mean of target variable (only for classification)

	Categorical Feature	label	OHE	LE	FE	Target Enc (label/freq)
			A B C			
A:4	A	No	1 0 0	0	4	0.5
	A	Yes	1 0 0	1	4	"
	A	No	1 0 0	0	4	"
	B	Yes	0 1 0	1	2	1

B=2	A	Yes	1 0 0	1	4	0.5
C=2	C	No	0 0 1	0	2	1
	B	Yes	0 1 0	1	2	"
	C	Yes	0 0 1	1	2	"

fn → 2

Categorical Feature	Label	OHE M	OHE F	FE	TE	LE
Male	N	1	0	4	2/4	0
female	Y	0	1	5	2/3	1
"					"	
Male	Y	1	0	4	2/4	1
Femule	N	0	1	5	2/5	0
"	N	0	1	5	"	0
"	Y	0	1	5	"	1
Male	Y	1	0	4	2/4	1
"	N	1	0	4	"	0

M: 4
F: 5

Feature Scaling:

Male Height in feet (f1)	Female Height in cm (f2)	Life Span in Year (f3)
8.0	150	30
8.5	165	40
7.9	170	36
8.2	140	41

- Its a method to scale numeric features in the same scale or range {-1 to 1, 0 to 1, 0 to 10 and so on...}.
  It is also called as data normalization.
- Its involved in last step of data pre-processing and before ML Model training. We apply feature scaling on independent variables. We fit feature scaling with trained data and transform on train and test data.

Q.) Why do we use feature scaling?

- The scale of row feature is different acc. to its units. ML algos can't understand feature's units, it understands only nos.
- If height 140cm and 8.2feet. ML algos understand only nos then it assumes that 140 > 8.2

Q.) Which ML algo requires feature scaling?

- K-Nearest Neighbours (KNN).
- K-Means
- Support Vector Machine
- PCA (Principle Component Analysis).
- LDA (Linear Discriminant Analysis).
- Linear Regression.
- Logistic Regression

} linear Based Algo

- Neural Networks

* Tree based algos don't require feature scaling

Types of Feature Scaling:

- Min - Max Scaler
- Std Scaler
- Max Abs Scaler
- Robust Scaler
- Quantile Transformer Scaler
- Power Transformer Scaler
- U

PCA:

* Calculate Eigen values & Eigen Vectors.

$$\begin{bmatrix} 12.67 & 16.33 \\ 16.33 & 22 \end{bmatrix}$$

* $|S - \lambda I| = 0$

$$\Rightarrow \begin{bmatrix} 12.67 - \lambda & 16.33 \\ 16.33 & 22 - \lambda \end{bmatrix} = 0$$

$$\Rightarrow (12.67 - \lambda)(22 - \lambda) - (16.33)^2 = 0$$

$$\Rightarrow 278.74 - 12.67\lambda - 22\lambda + \lambda^2 - (16.33)^2 = 0$$

$$\Rightarrow \lambda^2 - 34.67\lambda + 12.072 = 0$$

$$\Rightarrow \lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\simeq 34.31 \quad , \quad 0.35. \qquad \{\text{Picking the larger value}\}.$$

Step 4:

$$\begin{bmatrix} 12.67 - \lambda & 16.33 \\ 16.33 & 22 - \lambda \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -21.64 & 16.33 \\ 16.33 & -12.31 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow -21.64u_1 + 16.33u_2 = 0$$

$$16.33u_1 - 12.31u_2 = 0$$

$$\rightsquigarrow \frac{u_2}{u_1} = \frac{21.64}{16.33} = 1.325.$$

$$\rightsquigarrow \frac{u_1}{u_2} = \frac{16.33}{21.64} = 0.754$$

Steps: Project Data onto principle component.

X_1	X_2	PCA
5	10	16.625
3	7	10.975
10	15	28.256
2	4	

* $\begin{bmatrix} 5 & 10 \end{bmatrix} \begin{bmatrix} 1.3256 \\ 1 \end{bmatrix}$ { $\begin{bmatrix} 2 & 4 \end{bmatrix} \begin{bmatrix} 1.3256 \\ 1 \end{bmatrix}$

↳ $5 \times 1.3256 + 10 \times 1 = 16.628$

Feature Selection:

* Need for Feature Selection:

- Student Dataset:

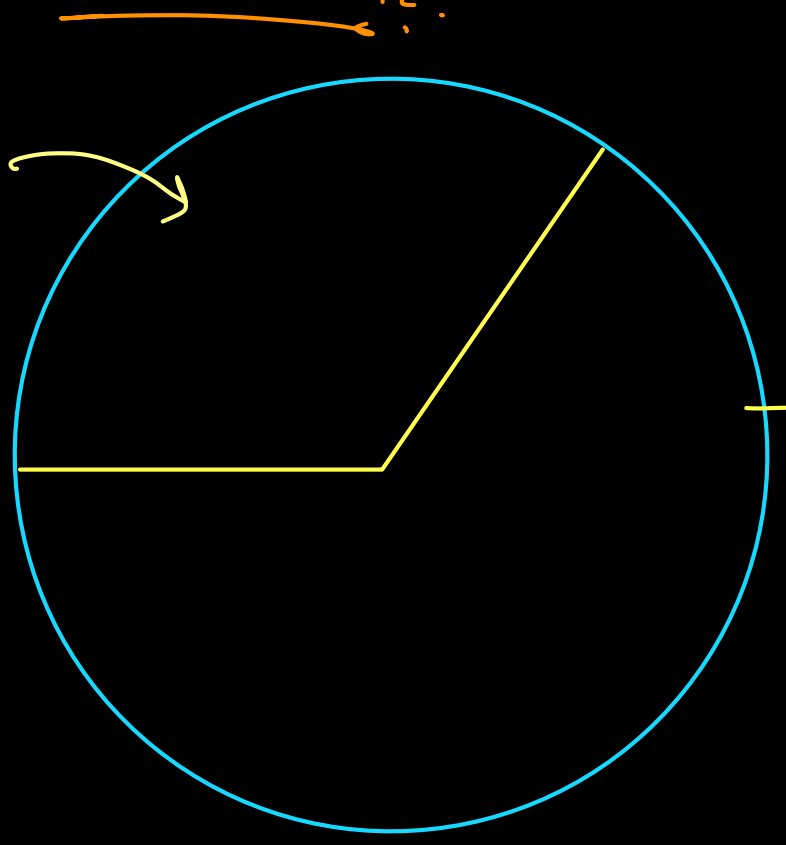
Name | Marks | Address | Race | Religion | Attendance

(Race, Religion) → ✗

- All the features the model does not give goal accuracy. It gives 60% accuracy.
- With relevant and best features, the model performs well. It gives above 60% accuracy.

What is feature Selection?:

Selecting optimal & best features for M.L. Models



→ Removing irrelevant or partially relevant features from the Data.

Importance of Feature Selection:

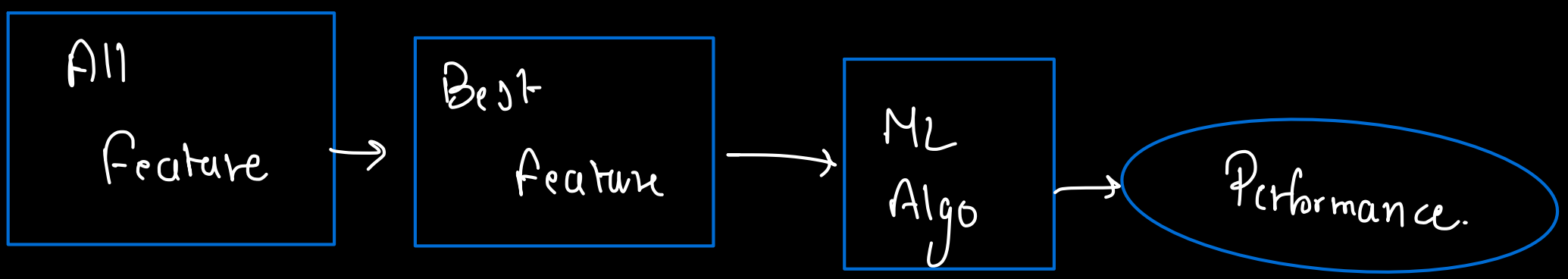
- Reduces the curse of dimensionality.
- It works on the relevance and importance of feature.
- It minimizes the cost of computation.
- It helps in learning the model by providing the best features for training the ML Model.
- It helps in achieving very good accuracy.

Feature Selection Techniques:

- Supervised Feature selection
  (we require the target or output variable)
  - Filter Method
  - Wrapper Method
  - Embedded Method.
- Unsupervised Feature Selection
  (We do not require the target or output variable)

1. Filter Method →

- Information gain → Determines Reduction in Entropy
- Chi squared Test → Determines the rs b/w categorical variable
- Fishers Score → Returns the rank of the feature in descending order.
- Missing value Ratio → Evaluate the features at against threshold value.



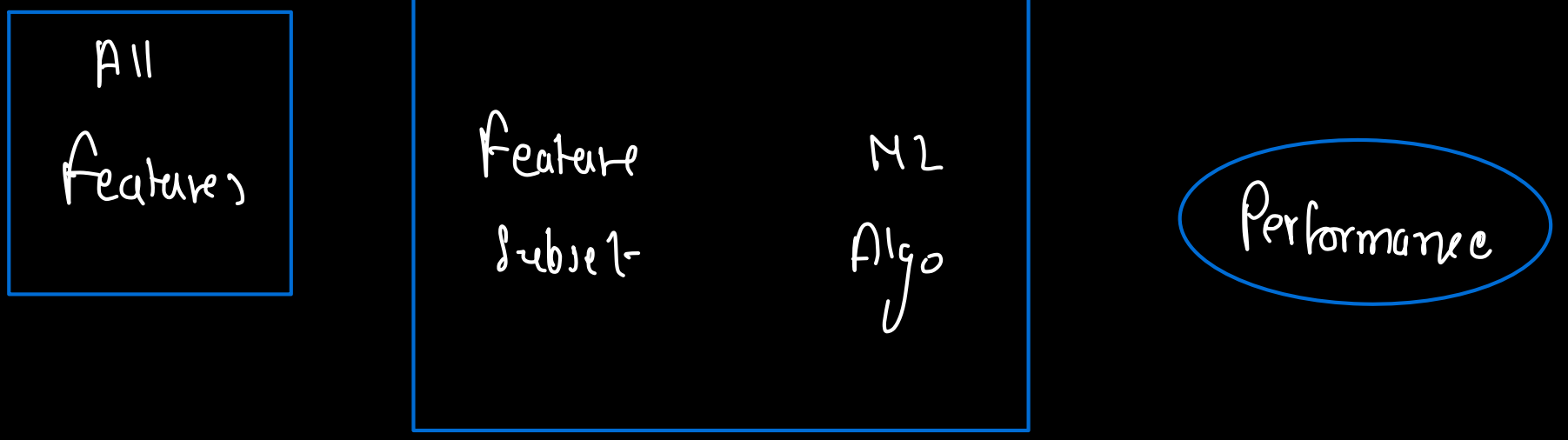
Advantages of Filter Method:

- Better computational Complexity.
- Fast and Scalable
- Independent of Classifier

Disadvantages of Filter Method:

- Ignores Interaction with Classifier
- " Feature Dependencies.

Wrapper Method:



- Forward Selection: Begins with empty set of features, with each iteration keeps adding on features and evaluates.
- Backward Elimination: Begins with all the features, with each iteration removes the least significant feature.
- Exhaustive Feature Selection: Evaluation is based on Brute-Force Method.
- Recursive Feature Elimination: It is recursive, greedy optimisation approach.

Advantages:

- Interacts with classifier
- Minimises computational cost
- Model is feature dependent

Disadvantages:

- High Risk of Overfitting.
- Computationally intensive.
- Lower Discriminative power

4. Embedded Method:

- Regularization techniques like L1, L2 and Elastic Nets.
- Random Forest, Decision Tree, Trees Based Methods for Feature Selection.