Alpha beta pruning (DFS)

The basic idea behind alpha-beta pruning is to reduce the number of nodes evaluated in the search tree by **eliminating branches that cannot possibly affect the final decision.**

- Alpha-beta pruning is an **optimization technique for the minimax algorithm**.
- It significantly reduces computation time by avoiding **unnecessary exploration of certain branches** in the game tree.

- The name **"alpha-beta"** comes from the **two extra parameters** passed during the minimax function: alpha and beta.

- Alpha: The **best value that the maximizer** can currently guarantee at or above the current level.(Max)

- Beta: The **best value that the minimizer** can currently guarantee at or below the current level.(Mini)

- Time complexity:**O(b^(d/2))**
- space complexity remains **O(bd)**

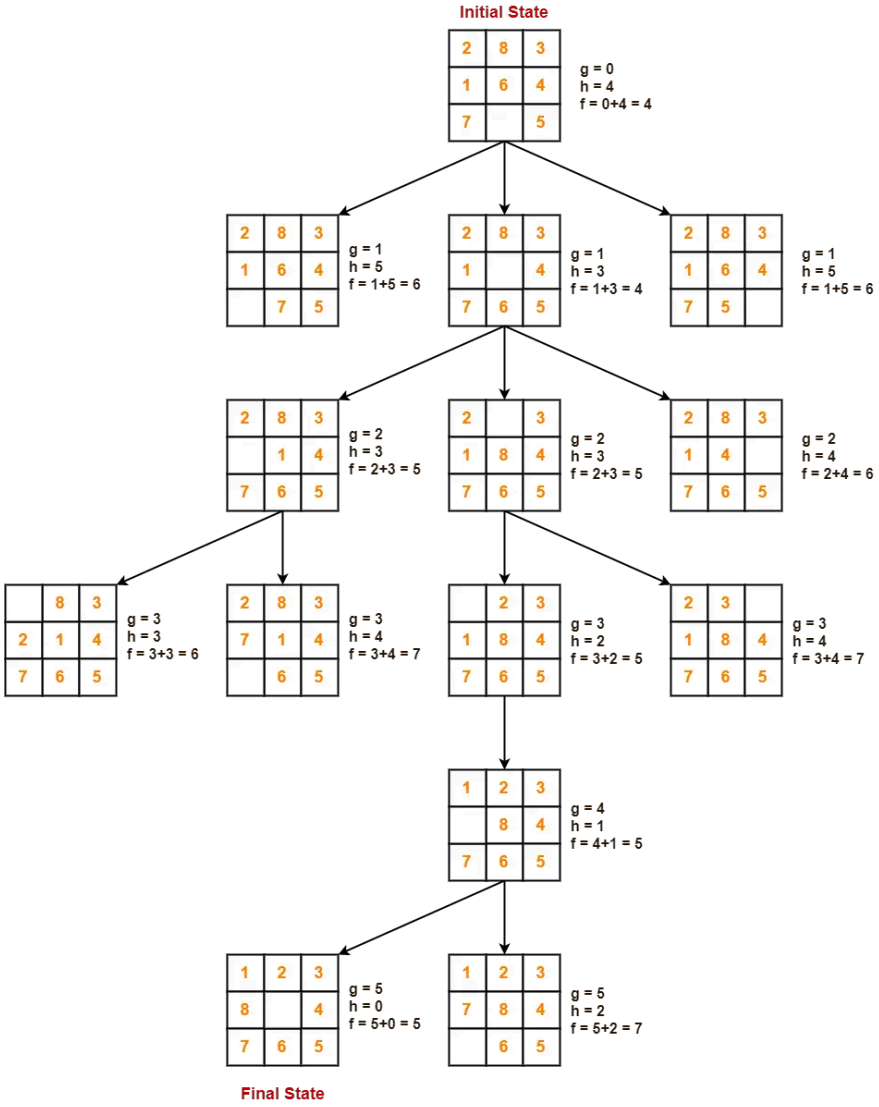
Example:

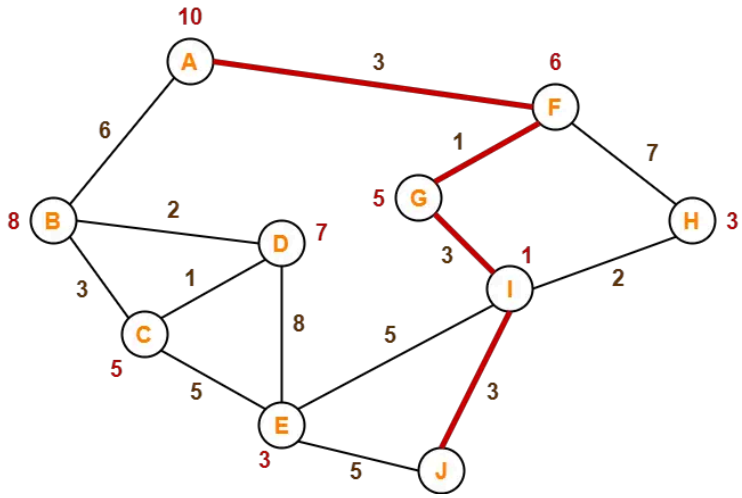
A star

A* is a widely used **informed search algorithm** that **combines** the advantages of both **breadth-first search** and the **greedy best-first search. f(N)=g(N)+h(N)**

- A* Search algorithm is a **path-finding and graph traversal** technique.
- It estimates the **shortest path** from a starting point to a goal point.
- Unlike other traversal methods, A* has "brains" – it's a **smart algorithm.**
- Many **games** and **web-based maps** use A* to find **efficient** (approximate) paths.
- A* guarantees the shortest path as long as the **heuristic function is admissible (never overestimates the true cost)**
- Time Complexity: **O(b^d)**
- Space Complexity: **O(b^d)**

Example: https://www.gatevidyalay.com/a-algorithm-a-algorithm-example-in-ai/





Crypt arithmetic

Cryptarithmetic, also known as verbal arithmetic or **word addition**, is a type of mathematical puzzle in which **letters or symbols represent digits**. The goal is to decipher the numerical values of these **symbols to make a valid arithmetic equation**. **Each letter** corresponds to a **unique digit**, and no two letters can represent the same digit.

Solving cryptarithmetic puzzles typically involves **trial and error**, **logical deduction**s, and sometimes the use of systematic strategies to **reduce possibilities**. It's common to start with the letters that have the fewest possible values based on the constraints and work from there.

FopA
https://www.javatpoint.com/first-order-logic-in-artificial-intelligence#:~:text=First%2Dorder%20logic%20is%20also,the%20relationship%20between%20those%20objects.

Universal Quantifier, (for all, everyone, everything)
Existential quantifier, (for some, at least one).

Example:

- **All man drink coffee.**

  ∀x man(x) → drink (x, coffee).
  It will be read as: There are all x where x is a man who drink coffee.

- **Some boys are intelligent.**

  ∃x: boys(x) ∧ intelligent(x)

  It will be read as: There are some x where x is a boy who is intelligent.

  - **The main connective for universal quantifier ∀ is implication →.**
  - **The main connective for existential quantifier ∃ is and ∧.**

1. All birds fly.
predicate is "fly(bird)."
And since there are all birds who fly so it will be represented as follows.

∀x bird(x) →fly(x).

2.Not all students like both Mathematics and Science.

In this question, the predicate is "like(x, y)," where x= student, and y= subject.
Since there are not all students, so we will use ∀ with negation, so following representation for this:

¬∀ (x) [student(x) → like(x, Mathematics) ∧ like(x, Science)].

3. Only one student failed in Mathematics.

In this question, the predicate is "failed(x, y)," where x= student, and y= subject.
Since there is only one student who failed in Mathematics, so we will use following representation for this:

∃ (x) [student(x) → failed (x, Mathematics) ∧∀ (y) [¬(x==y) ∧ student(y) → ¬failed (x, Mathematics)].

Types of Agents: SMGUL

Simple reflex agent

Simple reflex agent is said to be the simplest kind of agent.

- These agents select an action based on the current percept ignoring the rest of the percept history.
- Intelligence level in these agents is very limited.
- It works only in a fully observable environment.
- It does not hold any knowledge or information of nonperceptual parts of state.

Model based-Decision-making is based not only on the current percept but also on the history of percepts.

Goal based-

- Goal-based agents have specific goals or objectives they aim to achieve.
- They take actions that move them closer to their goals, considering the current state and future consequences.

Utility-to pick 'best' sequence of actions

Learning Agent-By actively exploring and experimenting with their environment, the most powerful agents are able to learn.

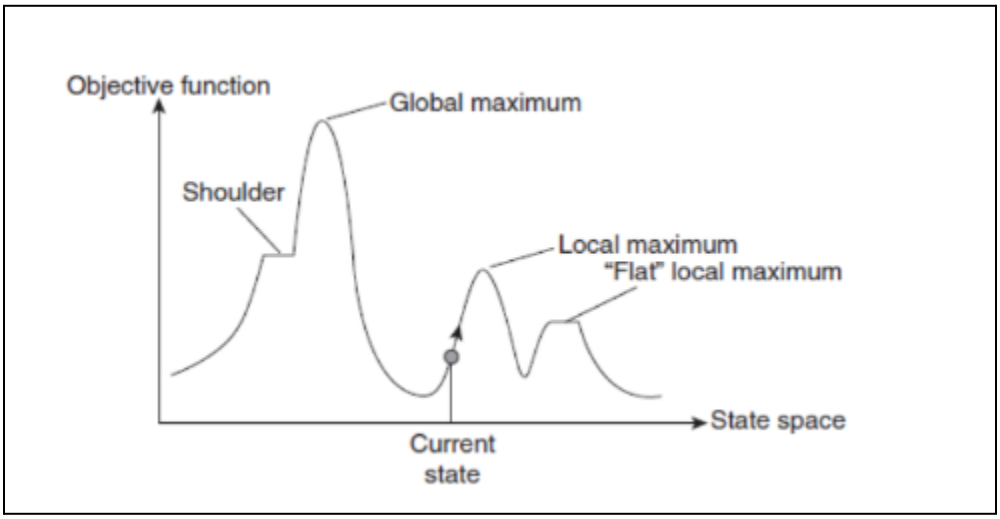
Difference: FDPARLEg

Aspect	Goal-Based Agents	Knowledge-Based Agents
Focus	Achieving specific objectives or goals.	Leveraging a repository of information and knowledge.
Decision Process	Deciding actions based on the pursuit of goals.	Deciding actions based on reasoning and information in a knowledge base.
Perception and Action	Perceive the environment and take actions to move closer to desired goals.	Use logical reasoning and inference to process information and determine actions.
Adaptability	Can adapt to changes in the environment or in the pursuit of goals.	Flexibility may vary; adaptation through goal adjustments.
Reasoning	Emphasis on evaluating the current state and selecting actions based on goal achievement.	Emphasis on logical reasoning, inference, and leveraging explicit knowledge.
Learning	May incorporate learning mechanisms to improve goal achievement.	May update the knowledge base through learning processes.
Example	A robot navigating toward a specific location.	An expert system in medicine diagnosing illnesses using a database of medical knowledge.

Aspect	Utility-Based Agents	Learning Agents
Objective	Maximize expected utility or value of outcomes.	Adapt and improve performance through learning from experience.
Decision Process	Decision-making based on maximizing utility.	Decision-making involves learning from data or experience.
Perception and Action	Perceive the environment, evaluate utility, and choose actions to maximize it.	Adapt behavior based on experiences and feedback to improve over time.
Adaptability	Can adapt based on utility values and changes in the environment.	Adaptability is a primary focus; the agent evolves over time through learning.
Reasoning	Evaluate actions based on utility functions.	Learn from data, adjust behavior, and improve decision-making over time.
Learning Mechanism	Focus on utility functions and optimization.	Incorporate learning mechanisms such as machine learning to improve performance.
Example	A decision-making system in finance maximizing profit based on risk and return.	An autonomous vehicle learning to navigate by adjusting its actions based on past experiences.

Aspect	Model-Based Agents	Goal-Based Agents
Focus	Use internal models of the world to make decisions.	Driven by specific objectives or goals.
Representation	Maintain an internal model or representation of the world.	Centered around achieving specific goals.
Decision Process	Decision-making involves consulting internal models to predict outcomes.	Decision-making based on actions that lead to the fulfillment of goals.
Perception and Action	Perceive the environment, update the internal model, and choose actions based on predictions.	Perceive the environment and take actions to achieve desired goals.
Adaptability	Can adapt by updating internal models to changes in the environment.	Adaptability may depend on the flexibility of the goal structure.
Reasoning	Reasoning involves using internal models to simulate and predict outcomes.	Reasoning involves evaluating the current state and selecting actions based on goal achievement.
Learning	May involve learning the model from experience to improve predictions.	Learning may be incorporated to adjust actions and goals based on feedback.
Example	A robot with a map of its environment predicting the outcomes of different movements.	A robot with the goal of reaching a specific location and adapting its actions to achieve the goal.

Hill climbing limitations



Local Optima:

- Hill climbing algorithms are prone to getting stuck in local optima, which are points in the search space where the algorithm finds a solution that is better than its neighbors but not the global optimum.
  To overcome local maximum problem Solution to the problem are:

  One possible solution is **backtracking.**
  We can backtrack to some earlier node and try to go in a different direction to attain the global peak. We can maintain a list of paths almost taken and go back to one of them if the path that was taken leads to a dead end.

Plateaus:

- In regions where the objective function is relatively flat, hill climbing may progress very slowly or become stuck altogether. This is known as a plateau problem, and it can hinder the algorithm's ability to make progress.
  To overcome plateaus:

  (a) A **big jump in some direction** can be done in order to get to a new section of search space. This method is recommended as in a plateau all neighbouring points
  have the same value.

  (b) Another solution is to **apply small steps several times in the same direction.**

Ridges and Valleys:

- Hill climbing may struggle in regions with ridges (where neighboring solutions have similar values) or valleys (where there is a steep drop in the objective function). It may oscillate
- Solution

Simulated Annealing:
Simulated annealing is a probabilistic optimization algorithm that allows the algorithm to occasionally accept moves that worsen the objective function. This stochastic behavior helps the algorithm escape from local optima and explore the solution space more extensively, including regions along ridges.
Random Restarts:
Random restarts involve running the hill climbing algorithm multiple times with different initial solutions. This approach increases the likelihood of finding a better solution by exploring different regions of the search space. It helps counteract the impact of getting stuck in a local optimum along a ridge.

No Memory of Past Moves:

- Hill climbing does not keep a memory of past moves, which means **it might revisit the same states repeatedly**. This lack of memory prevents the algorithm from considering the history of its search and learning from previous iterations.

Limited Exploration:

- Hill climbing explores the solution space based on local information, which may result in a limited exploration of the overall search space

Water Jug Problem:

Scenario:

You have two water jugs, **Jug A and Jug B**, each with known capacities.
Jug A has a capacity of X liters, and Jug B has a capacity of Y liters (where X and Y are positive integers).
There is an infinite supply of water available.
The objective is to measure out a specific target volume of water, denoted as Z liters (where Z is a positive integer).

Rules:

You can perform the following actions:

Fill: Fill a jug completely.
Empty: Empty a jug completely.
Pour: Pour water from one jug into another until the receiving jug is full or the source jug is empty.

Objective:

Find a **sequence of these actions to measure exactly Z liters of water** using the given jugs.

Example:

Suppose Jug A has a capacity of 3 liters, Jug B has a capacity of 5 liters, and the target volume is 4 liters. A solution might be:

Fill Jug B: Jug B now has 5 liters.
Pour from B to A: Jug A has 3 liters, and Jug B has 2 liters.
Empty Jug A: Jug A is now empty, and Jug B has 2 liters.
Pour from B to A: Jug A has 2 liters, and Jug B has 0 liters.
Fill Jug B: Jug B is now full (5 liters).
Pour from B to A: Jug A has 4 liters, and Jug B has 1 liter.

Representation:(FPEPFP)

Fill(B): Jug B now has 5 liters.
Pour(B, A): Jug A has 3 liters, and Jug B has 2 liters.
Empty(A): Jug A is now empty, and Jug B has 2 liters.
Pour(B, A): Jug A has 2 liters, and Jug B has 0 liters.
Fill(B): Jug B is now full (5 liters).
Pour(B, A): Jug A has 4 liters, and Jug B has 1 liter.

Inference Rules

Modus Ponens:

If P→Q is true, and P is true, then Q is true.

Example:

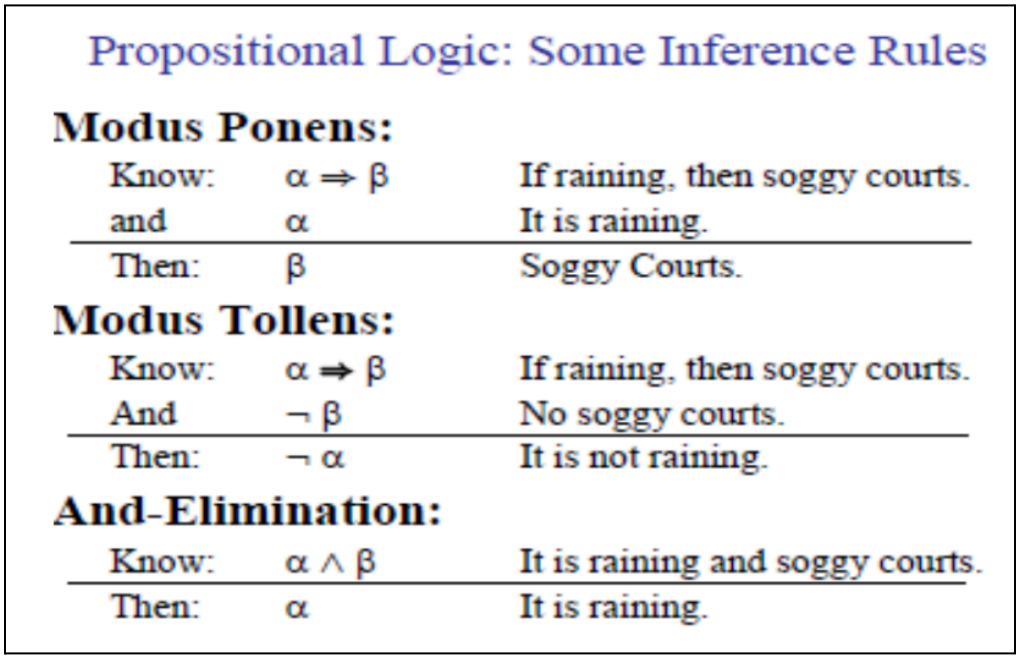
> If it is raining (P→Q) and you observe that the ground is wet (P), then you can infer that it is wet outside (Q).

Modus Tollens:

If P→Q is true, and Q is false, then P is false.

Example:

> If it is not wet outside (¬Q) and it is supposed to be wet if it is raining (P→Q), then you can infer that it is not raining (¬P).



Conjunction:

If **P is true** and **Q is true**, then **P∧Q is true.**

Example:

> If it is sunny (P) and it is warm (Q), then you can infer that it is sunny and warm (P∧Q).

Resolution:

If **P∨Q is true** and **¬P∨R is true**, then **Q∨R is true.**

Example:

> Either it is sunny or raining (P∨Q), and it is not sunny but windy (¬P∨R), then you can infer that it is raining or windy (Q∨R).

Double Negation Rule:

If **¬(¬P) is true**, then **P is also true.**

Example:P:It is sunny

Time and Space Complexity of some algorithm

#Un Informed(Blind search)

- **BFS O(b^d+1)**
- **DFS O(b^m)**
- **UCS O(b^C*/E)**
- **DLS O(b^l)**
- **IDS O(b^d)**

#Informed Search(Heuristic search algorithms)

- **Best FS O(b^d)**
- **A* O(b^d)**
- **Alpha Beta Pruning O(b^d/2)**

Criterion	*BFS*	*UCS*	*DFS*	*DLS*	*Iterative deepening*	*Bidirectional*
Completeness	Complete	Complete	Incomplete	Incomplete	Complete	Complete
Time Complexity (running time)	$O(b^{d+1})$	$O(b^{(C*/\epsilon)})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Space Complexity (memory)	$O(b^{d+1})$	$O(b^{(C*/\epsilon)})$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$
Optimality	Optimal	Optimal	Non-optimal	Non-optimal	Optimal	Optimal

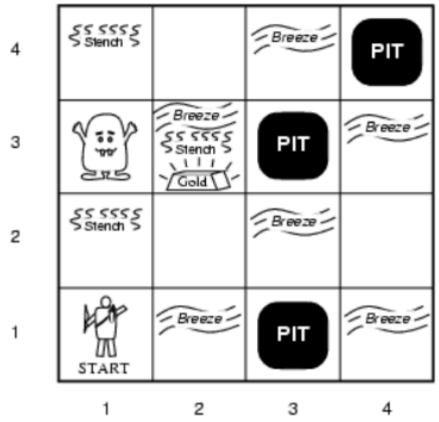
PEAS Description of soccer problem

- **Performance Measure:** The soccer-playing agent aims to maximize the number of goals scored, minimize the number of goals conceded, and adhere to the team strategy to achieve success in the game.
- **Environment**: The soccer field is the playing environment, containing the ball, other players, goalposts, and other relevant elements. The environment is dynamic, with changing positions of players and the ball.
- **Actuators**: The agent's actuators include its legs for kicking the ball, body for movement and balance, and other relevant body parts for executing soccer-related actions.
- **Sensors**: Sensors provide information about the state of the environment. Visual sensors (eyes) observe the positions of players and the ball, auditory sensors (ears) receive communication from teammates, and proprioceptive sensors provide feedback on the agent's body position.

BASIS FOR COMPARISON	BFS	DFS
Basic	Vertex-based algorithm	Edge-based algorithm
Data structure used to store the nodes	Queue	Stack
Memory consumption	Inefficient	Efficient
Structure of the constructed tree	Wide and short	Narrow and long
Traversing fashion	Oldest unvisited vertices are explored at first.	Vertices along the edge are explored in the beginning.
Optimality	Optimal for finding the shortest distance, not in cost.	Not optimal
Application	Examines bipartite graph, connected component and shortest path present in a graph.	Examines two-edge connected graph, strongly connected graph, acyclic graph and topological order.

Wumpus World PEAS description

- Performance measure
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot



PEAS for English Speaking Tutor:

- Performance Measure:
  - Language Proficiency Improvement
  - Engagement
  - Completion of Exercises
  - Feedback Quality
- Environment:
  - Virtual Classroom
  - Learning Materials
  - Communication Tools
- Actuators:
  - Speech Synthesis
  - Text Analysis
  - Lesson Planning
  - Feedback Generation
- Sensors:
  - Speech Recognition
  - Text Recognition
  - Engagement Monitoring
  - Performance Analytics

John has at least two friends.

$\exists x \exists y(x/=y \wedge F(John,x) \wedge F(John,y))$

If two people are friends, then they are not enemies.

$\forall x \forall y((F(x,y) \wedge F(y,x)) \rightarrow \neg E(x,y))$

F:friend

E:enemy

all happy people smile

$\forall x(H(x) \rightarrow S(x))$

all people who are graduating are happy

$\forall x(G(x) \rightarrow H(x))$