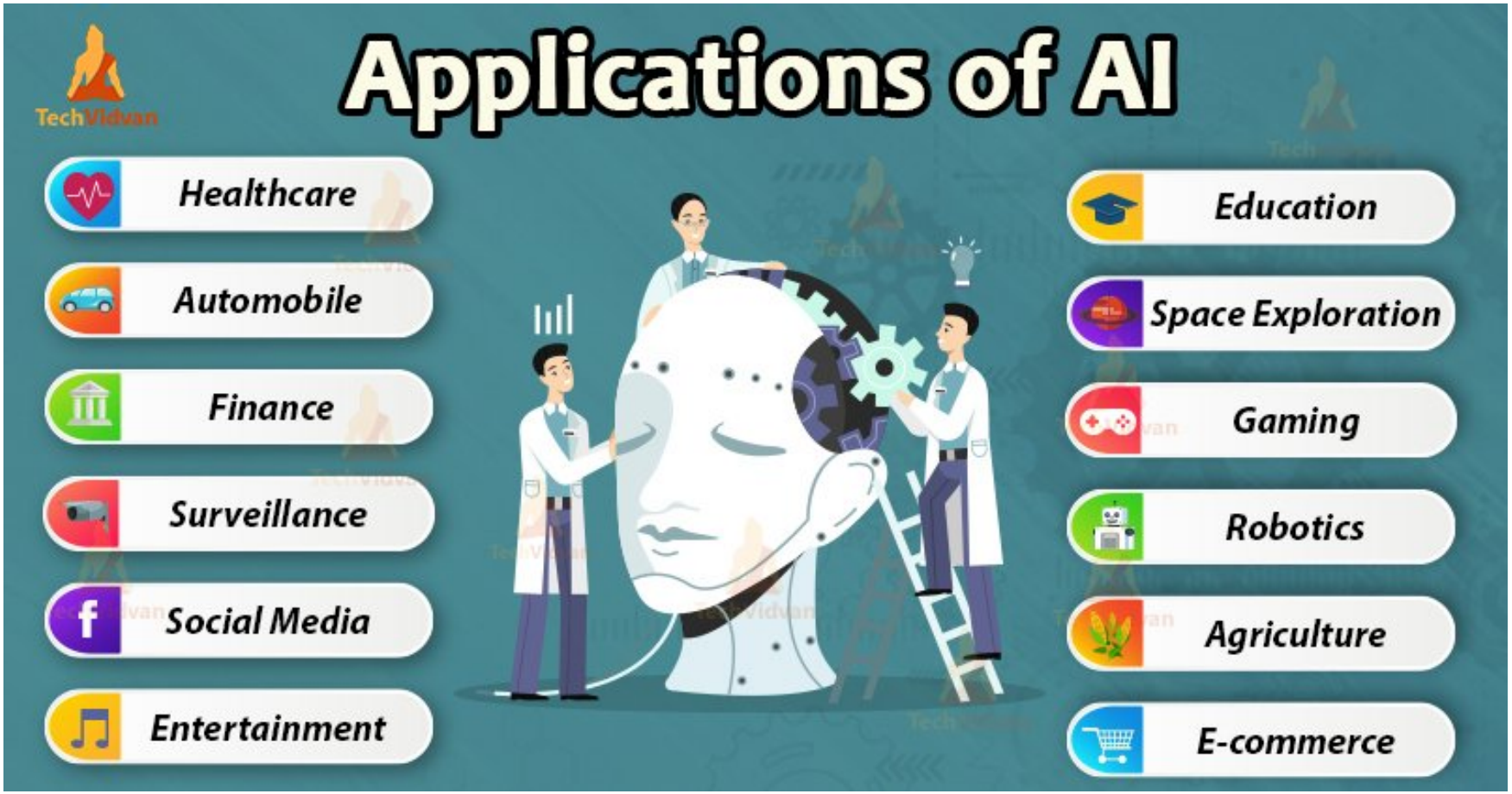
Unit-1

Introduction

What is Artificial Intelligence?

- "It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans, and able to make decisions."

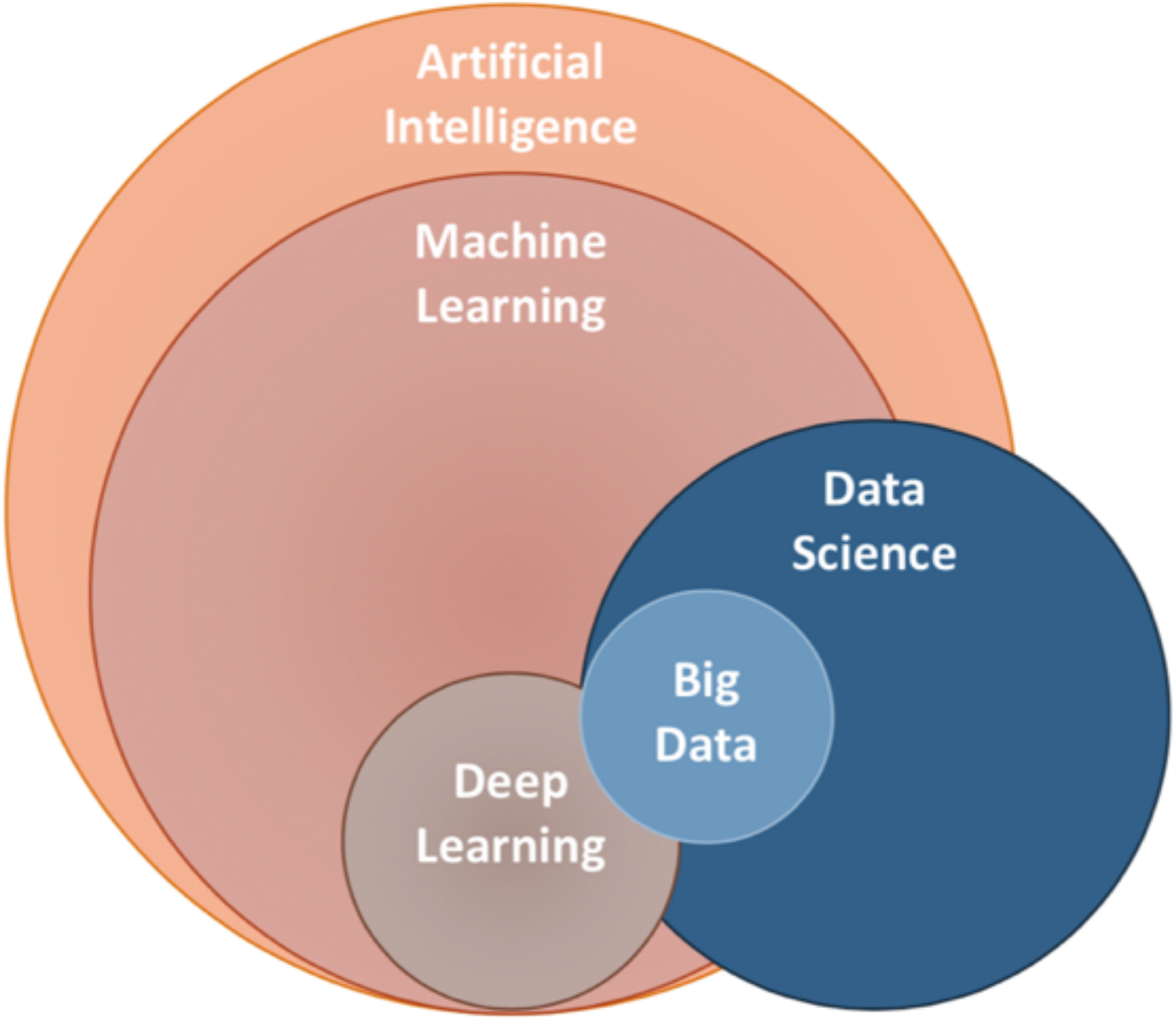
https://www.javatpoint.com/artificial-intelligence-tutorial

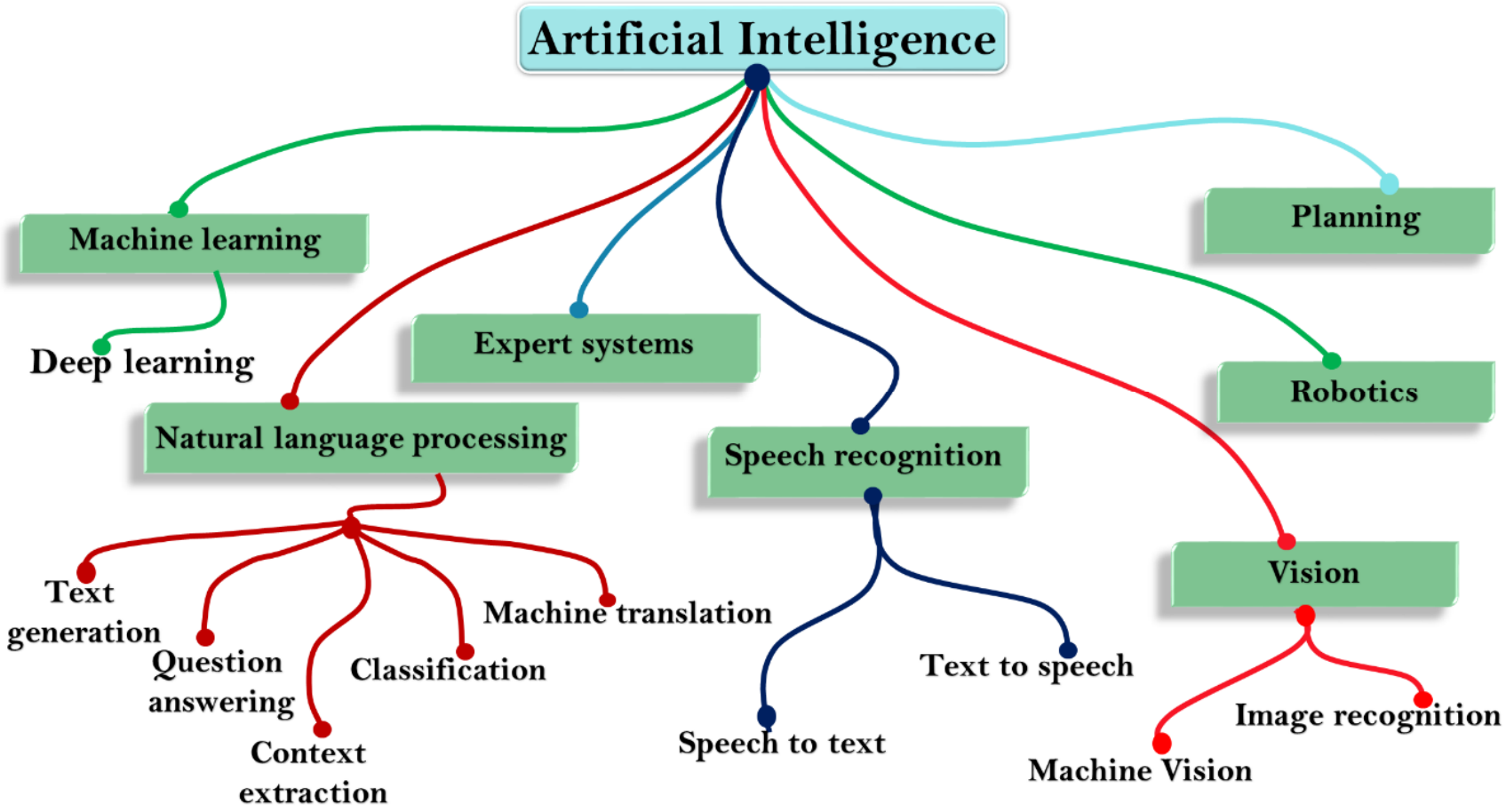


https://techvidvan.com/tutorials/artificial-intelligence-applications/

Why Artificial Intelligence?

- With the help of AI, you can create such software or devices which can solve real-world problems very easily and with accuracy such as health issues, marketing, traffic issues, etc.
- With the help of AI, you can create your personal virtual Assistant, such as Cortana, Google Assistant, Siri, etc.
- With the help of AI, you can build such Robots which can work in an environment where survival of humans can be at risk.
- AI opens a path for other new technologies, new devices, and new Opportunities.


Artificial Intelligence
Machine Learning
Data Science
Big Data
Deep Learning



https://www.javatpoint.com/subsets-of-ai

History of Artificial Intelligence

Year	Milestone / Innovation
1923	Karel Čapek plays named “Rossum’s Universal Robots, the first use of the word “robot” in English.
1943	Foundations for neural networks laid.
1945	Isaac Asimov, a Columbia University alumni, use the term Robotics.
1956	John McCarthy first used the term Artificial Intelligence. Demonstration of the first running AI program at Carnegie Mellon University.
1964	Danny Bobrow’s dissertation at MIT showed how computers could understand natural language.
1969	Scientists at Stanford Research Institute Developed Shakey. A robot equipped with locomotion and problem-solving.
1979	The world’s first computer-controlled autonomous vehicle, Stanford Cart, was built.
1990	Significant demonstrations in machine learning
1997	The Deep Blue Chess Program beat the then world chess champion, Garry Kasparov.
2000	Interactive robot pets have become commercially available. MIT displays Kismet, a robot with a face that expresses emotions.
2006	AI came into the Business world in the year 2006. Companies like Facebook, Netflix, Twitter started using AI.
2012	Google has launched an Android app feature called “Google now”, which provides the user with a prediction.
2018	The “Project Debater” from IBM debated complex topics with two master debaters and performed exceptionally well.

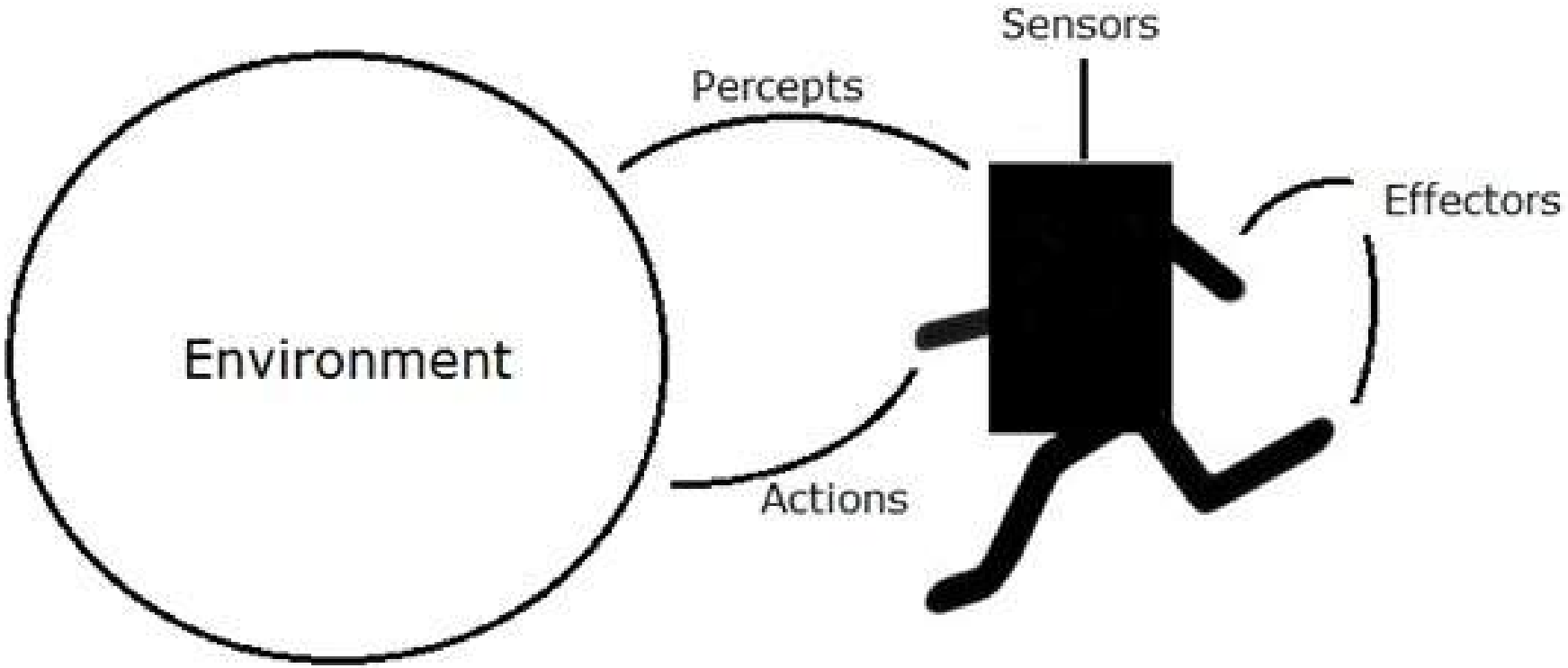
Disadvantages of Artificial Intelligence

- High Cost
- No Creativity
- Increase in unemployment
- Makes human lazy
- No Ethics

What are Agent and Environment?

- An **agent** is anything that can perceive its environment through **sensors** and acts upon that environment through **effectors.**
  - A **human agent** has sensory organs such as eyes, ears, nose, tongue and skin parallel to the sensors, and other organs such as hands, legs, mouth, for effectors.
  - A **robotic agent** replaces cameras and infrared range finders for the sensors, and various motors and actuators for effectors.
  - A **software agent** has encoded bit strings as its programs and actions.

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_agents_and_environments.htm


Sensors
Percepts
Effectors
Environment
Actions

Agent Terminology

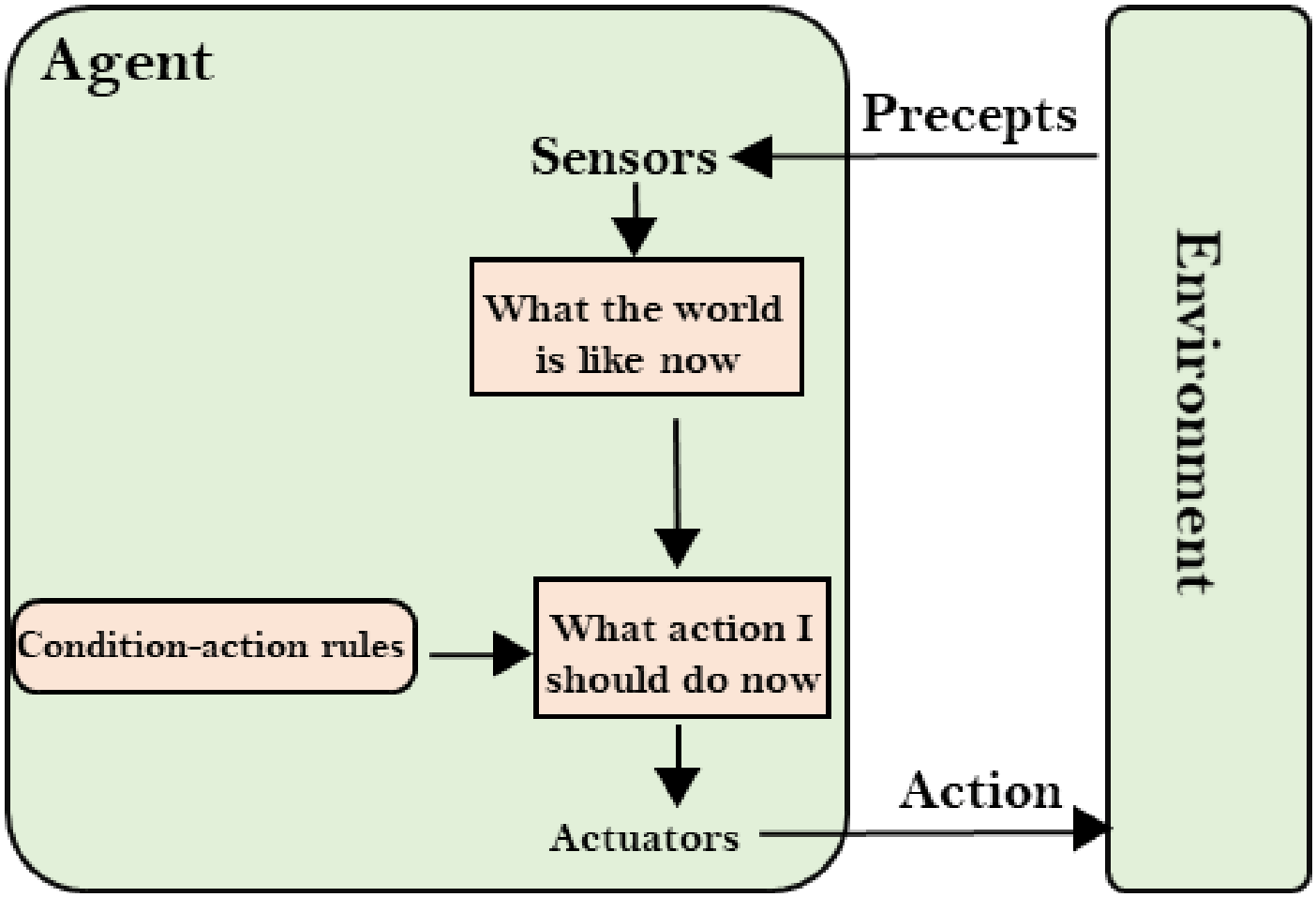
- **Performance Measure of Agent** – It is the criteria, which determines how successful an agent is.
- **Behavior of Agent** – It is the action that agent performs after any given sequence of percepts.
- **Percept** – It is agent's perceptual inputs at a given instance.
- **Percept Sequence** – It is the history of all that an agent has perceived till date.
- **Agent Function** – It is a map from the precept sequence to an action.

Types of AI Agents

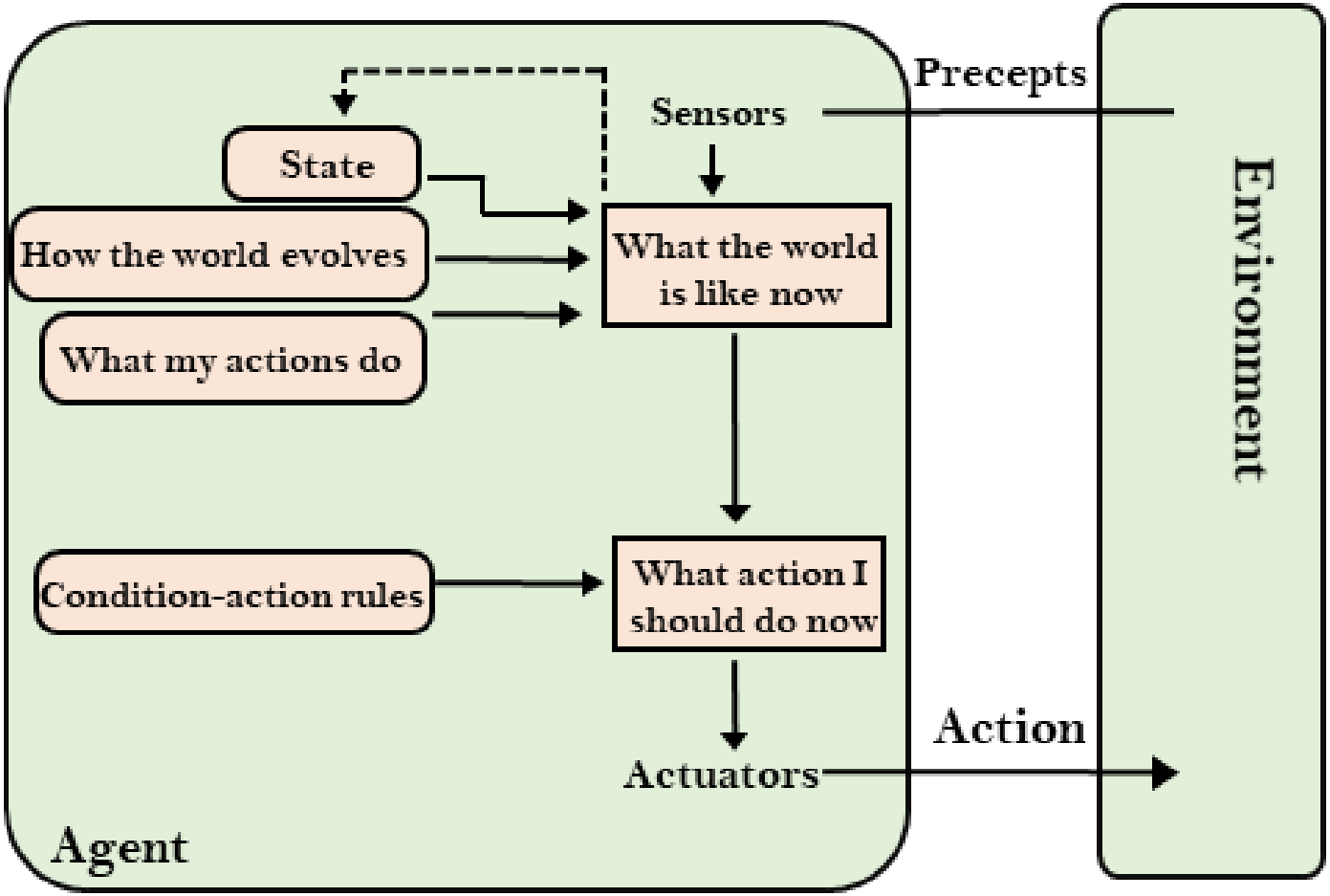
- Simple Reflex Agent
- Model-based reflex agent
- Goal-based agents
- Utility-based agent
- Learning agent

https://www.javatpoint.com/types-of-ai-agents

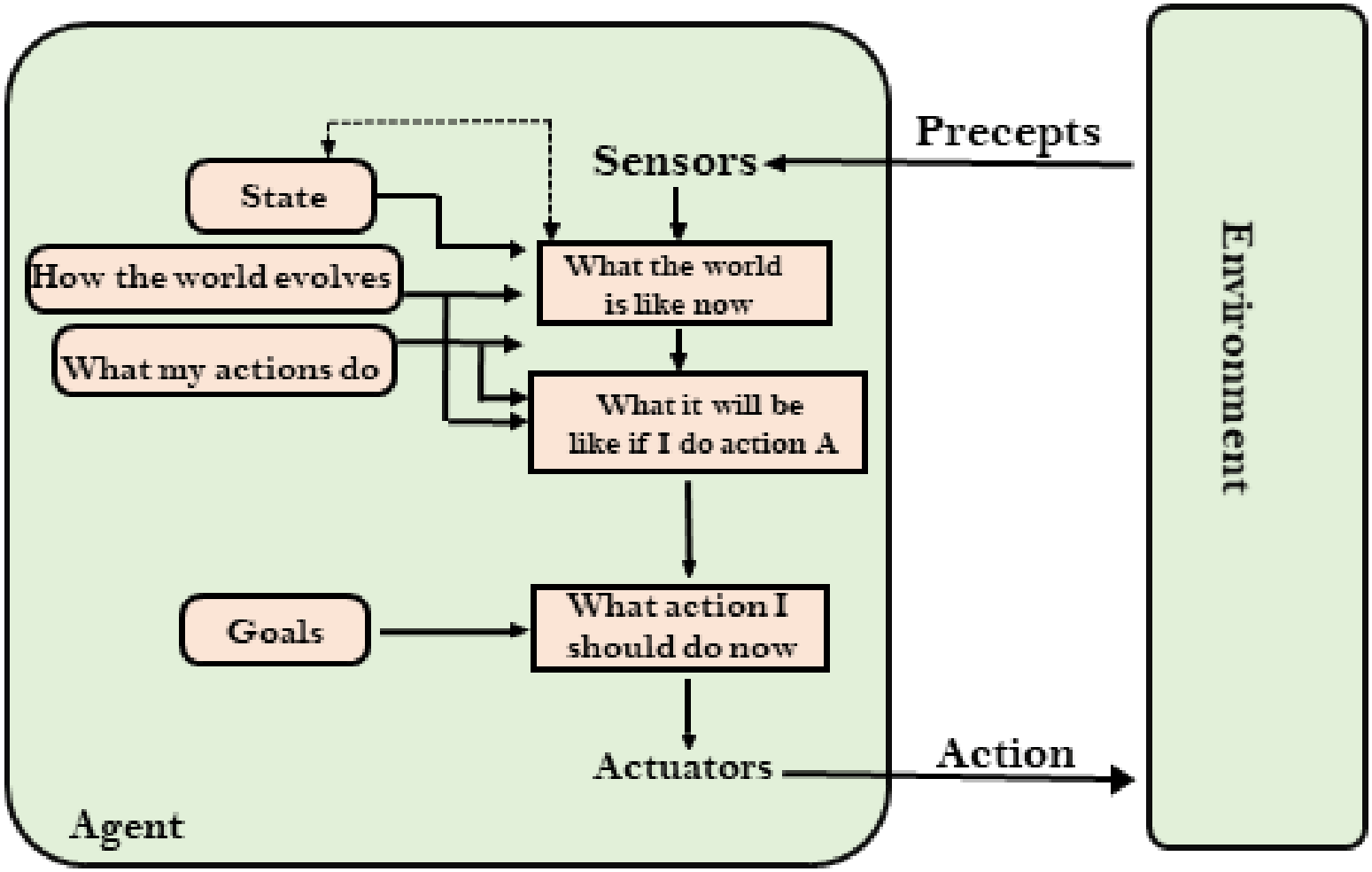
Simple Reflex Agent



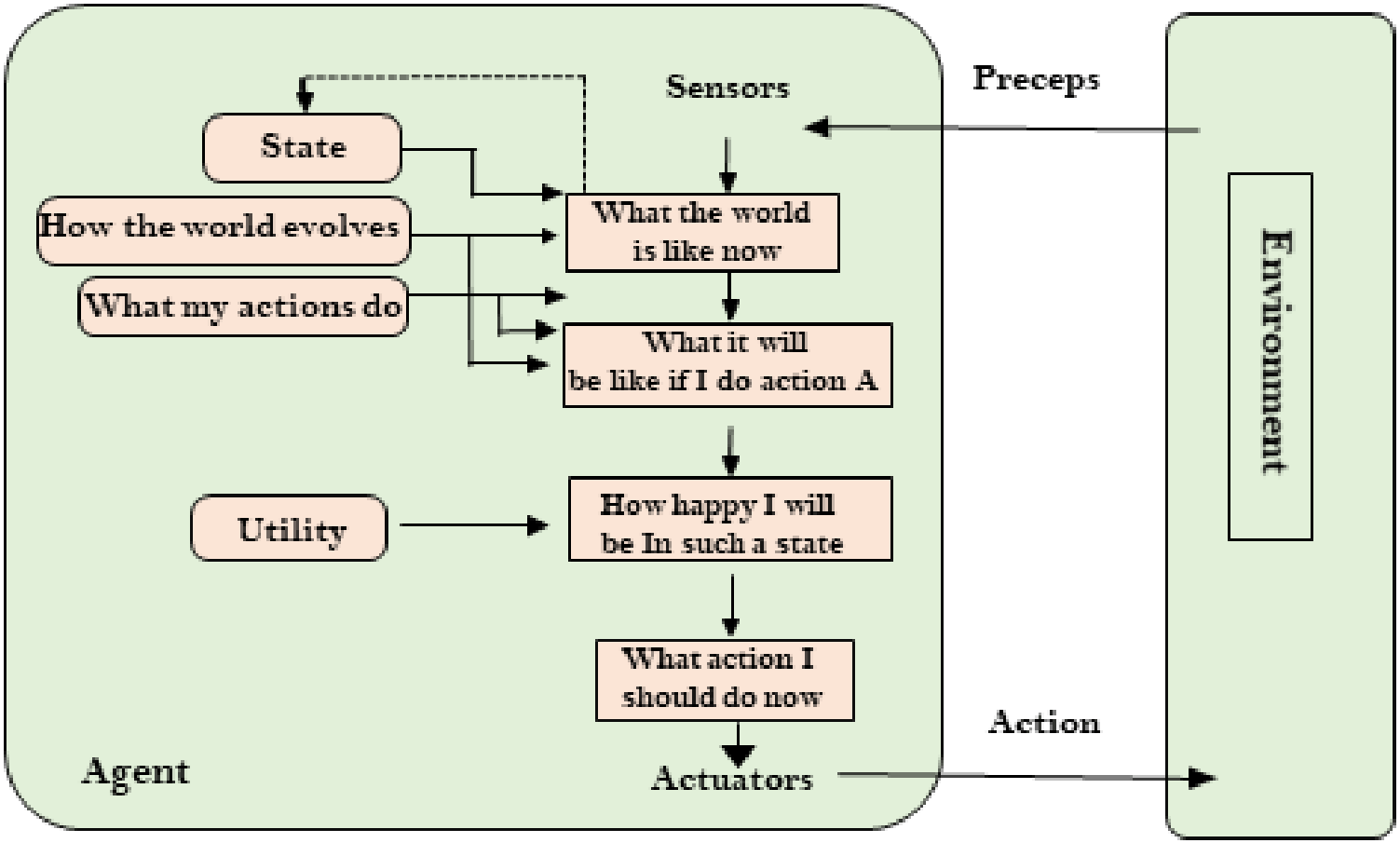
Model-based reflex agent



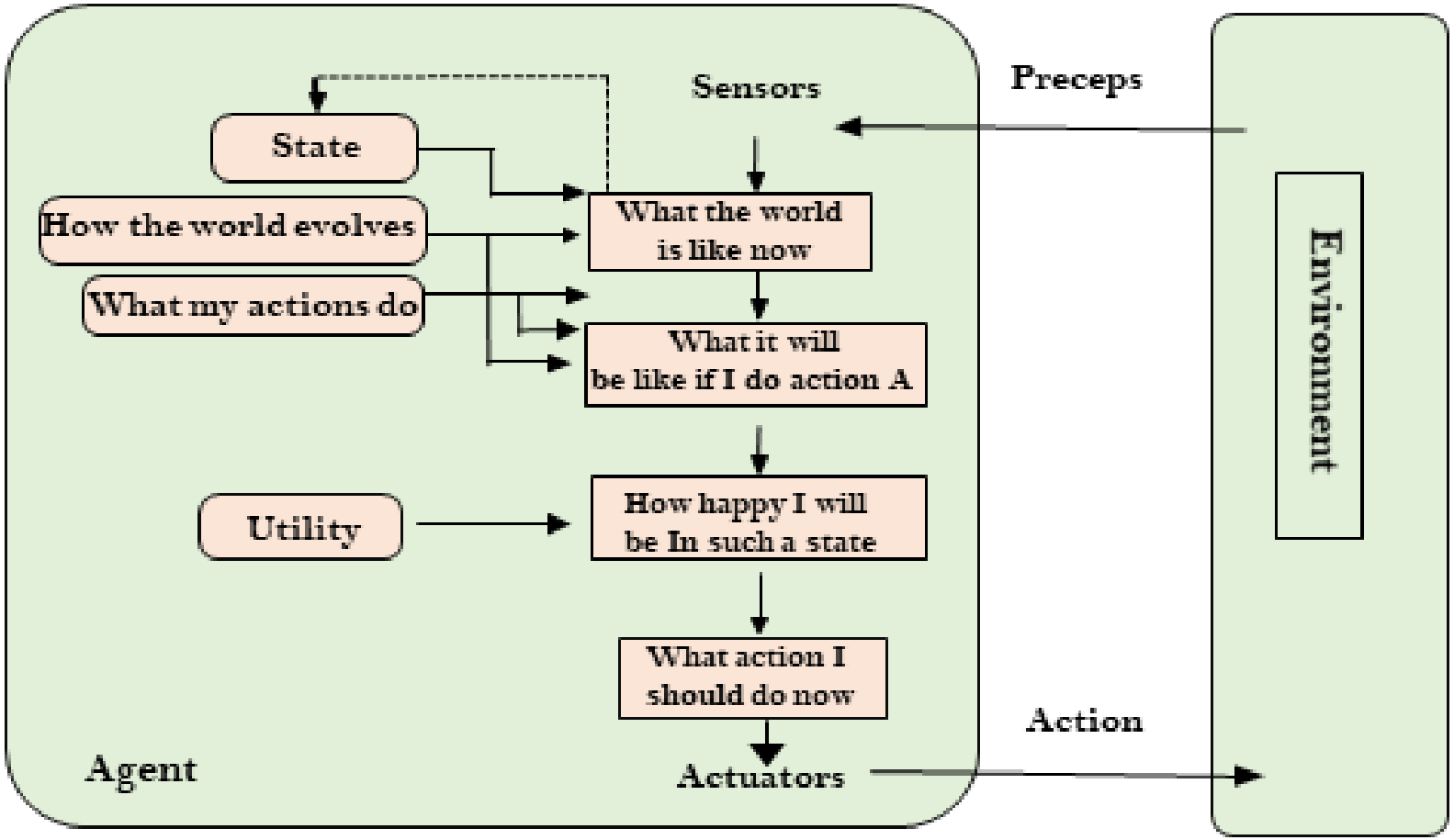
Goal-based agents



Utility-based agents



Learning Agents



Types of Environments in AI

- Fully Observable vs Partially Observable
- Deterministic vs Stochastic
- Competitive vs Collaborative
- Single-agent vs Multi-agent
- Static vs Dynamic
- Discrete vs Continuous
- Episodic vs Sequential

https://www.geeksforgeeks.org/types-of-environments-in-ai/

Fully Observable vs Partially Observable

- When an agent sensor is capable to sense or access the complete state of an agent at each point in time, it is said to be a fully observable environment else it is partially observable.
- Maintaining a fully observable environment is easy as there is no need to keep track of the history of the surrounding.
- An environment is called **unobservable** when the agent has no sensors in all environments.
- **Examples:**
  - **Chess –** the board is fully observable, so are the opponent's moves.
  - **Driving –** the environment is partially observable because what's around the corner is not known.

https://www.geeksforgeeks.org/types-of-environments-in-ai/

Deterministic vs Stochastic

- When a uniqueness in the agent's current state completely determines the next state of the agent, the environment is said to be deterministic.
- The stochastic environment is random in nature which is not unique and cannot be completely determined by the agent.
- **Examples:**
  - **Chess –** there would be only a few possible moves for a coin at the current state and these moves can be determined.
  - **Self Driving Cars –** the actions of a self-driving car are not unique, it varies time to time.

https://www.geeksforgeeks.org/types-of-environments-in-ai/

Competitive vs Collaborative

- An agent is said to be in a competitive environment when it competes against another agent to optimize the output.
- The game of chess is competitive as the agents compete with each other to win the game which is the output.
- An agent is said to be in a collaborative environment when multiple agents cooperate to produce the desired output.
- When multiple self-driving cars are found on the roads, they cooperate with each other to avoid collisions and reach their destination which is the output desired.

https://www.geeksforgeeks.org/types-of-environments-in-ai/

Single-agent vs Multi-agent

- An environment consisting of only one agent is said to be a single-agent environment.
- A person left alone in a maze is an example of the single-agent system.
- An environment involving more than one agent is a multi-agent environment.
- The game of football is multi-agent as it involves 11 players in each team.

https://www.geeksforgeeks.org/types-of-environments-in-ai/

Dynamic vs Static

- An environment that keeps constantly changing itself when the agent is up with some action is said to be dynamic.
- A roller coaster ride is dynamic as it is set in motion and the environment keeps changing every instant.
- An idle environment with no change in its state is called a static environment.
- An empty house is static as there’s no change in the surroundings when an agent enters.

https://www.geeksforgeeks.org/types-of-environments-in-ai/

Discrete vs Continuous

- If an environment consists of a finite number of actions that can be deliberated in the environment to obtain the output, it is said to be a discrete environment.
- The game of chess is discrete as it has only a finite number of moves. The number of moves might vary with every game, but still, it's finite.
- The environment in which the actions performed cannot be numbered i.e.. is not discrete, is said to be continuous.
- Self-driving cars are an example of continuous environments as their actions are driving, parking, etc. which cannot be numbered.

https://www.geeksforgeeks.org/types-of-environments-in-ai/

Episodic vs Sequential

- In **Episodic task environment**, each of the agents action is divided into an atomic incidents or episodes. There is no dependency between current and previous incident. In each incident agent receives input from environment and then performs corresponding action.
- **Example:** Consider an example of **Pick and Place robot**, which is used to detect defective parts from conveyer belt. Here, every time robot(agent) will make decision on current part i.e. there is no dependency between current and previous decision.
- In **Sequential environment**, previous decision can affect all future decisions. The next action of agent depends on what action he has taken previously and what action he is supposed to take in future.
- **Example:**
  - **Checkers-** Where previous move can affect all the following moves.

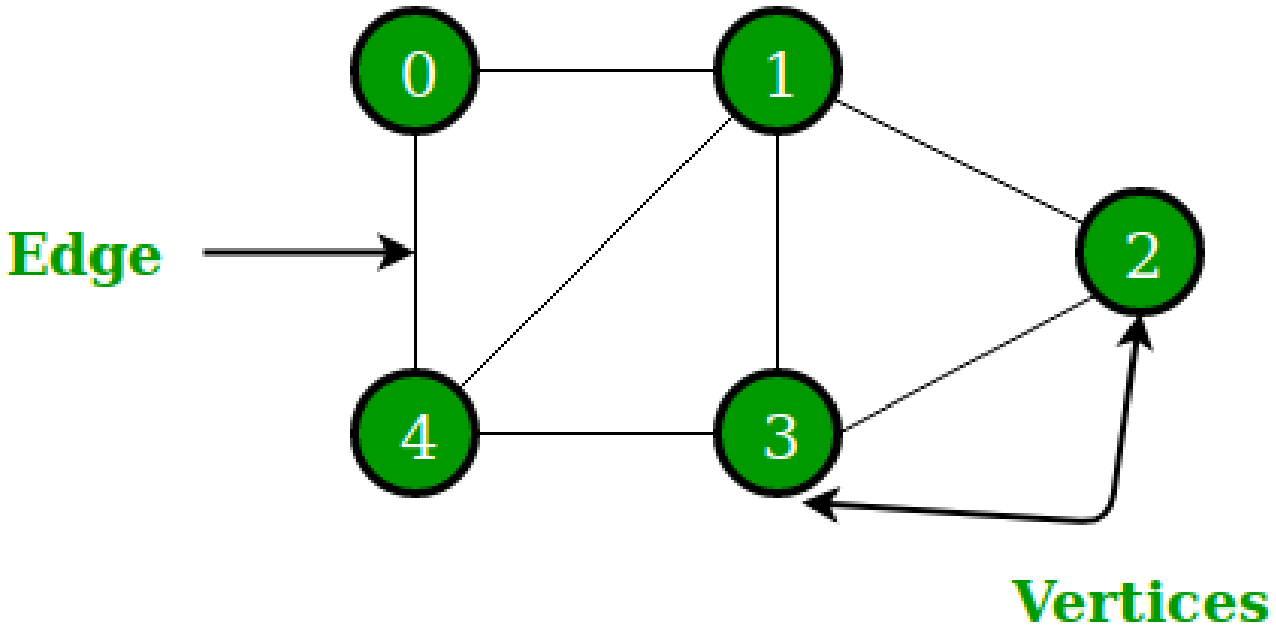
https://www.geeksforgeeks.org/types-of-environments-in-ai/

Review of tree and Graph structures

Graph

- A graph is collection of two sets V and E where V is a finite non-empty set of vertices and E is a finite non-empty set of edges.
  - Vertices are nothing but the nodes in the graph.
  - Two adjacent vertices are joined by edges.
  - Any graph is denoted as G = {V, E}.

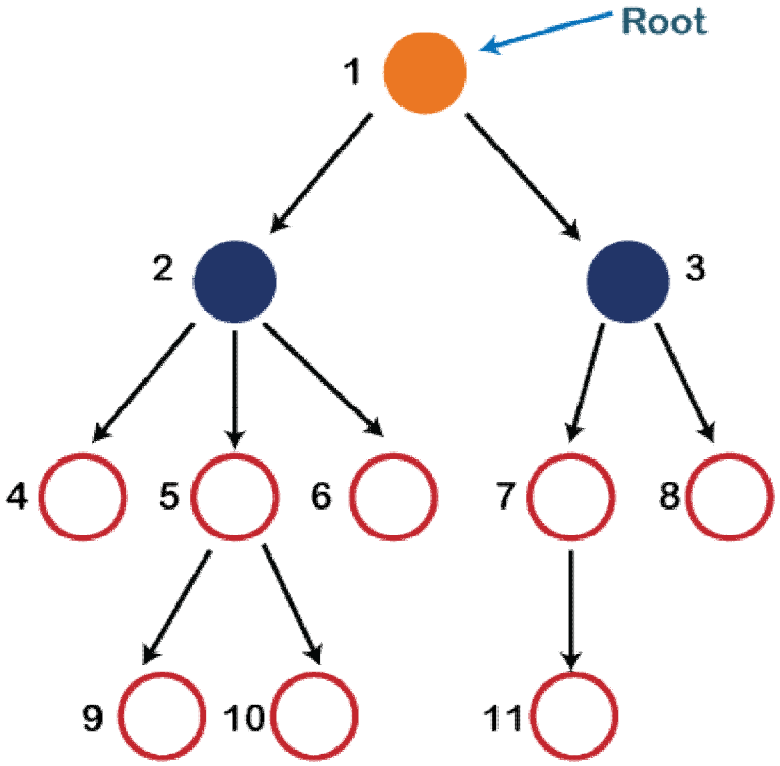
https://www.geeksforgeeks.org/difference-between-graph-and-tree/


0
1
2
3
4
Edge
Vertices

Tree

- A tree data structure is a non-linear data structure because it does not store in a sequential manner. It is a hierarchical structure as elements in a Tree are arranged in multiple levels.
- In the Tree data structure, the topmost node is known as a root node. Each node contains some data, and data can be of any type.

Introduction to Trees



https://www.javatpoint.com/tree

Eight Puzzle Problem

Initial state

1	2	3
	4	6
7	5	8

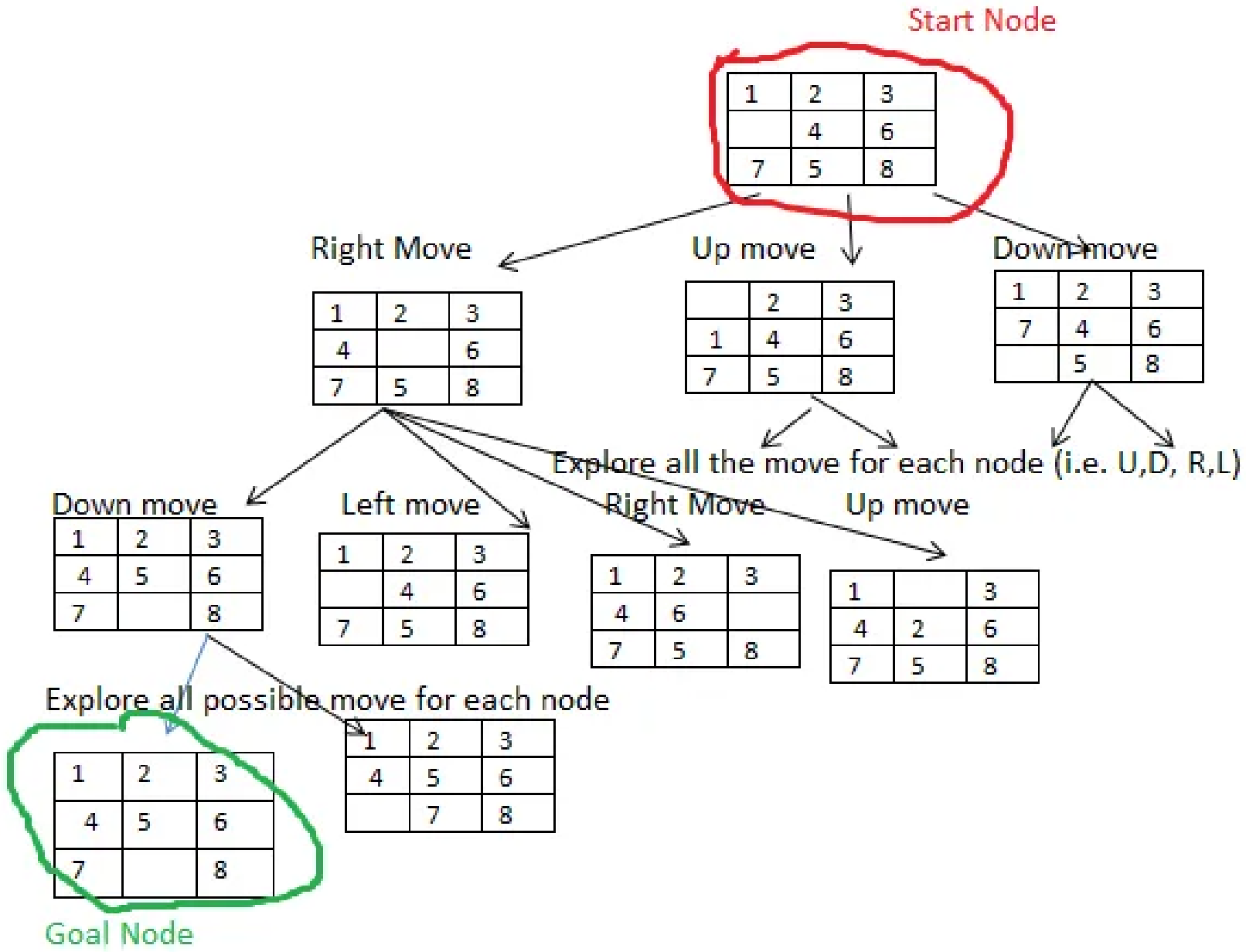
Goal state

1	2	3
4	5	6
7	8	

The empty space can only **move in four directions** (Movement of empty space)

- Up
- Down
- Right or
- Left

https://www.goeduhub.com/8062/8-puzzle-problem-in-ai-artificial-intelligence


Start Node
1 2 3
4 6
7 5 8
Right Move
1 2 3
4 6
7 5 8
Up move
2 3
1 4 6
7 5 8
Down move
1 2 3
7 4 6
5 8
Explore all the move for each node (i.e. U,D, R,L)
Down move
1 2 3
4 5 6
7 8
Left move
1 2 3
4 6
7 5 8
Right Move
1 2 3
4 6
7 5 8
Up move
1 3
4 2 6
7 5 8
Explore all possible move for each node
1 2 3
4 5 6
7 8
1 2 3
4 5 6
7 8
Goal Node

Time complexity: In worst case time complexity in **BFS is O(b^d) know as order of b raise to power d.** In this **particular case it is (3^20).**

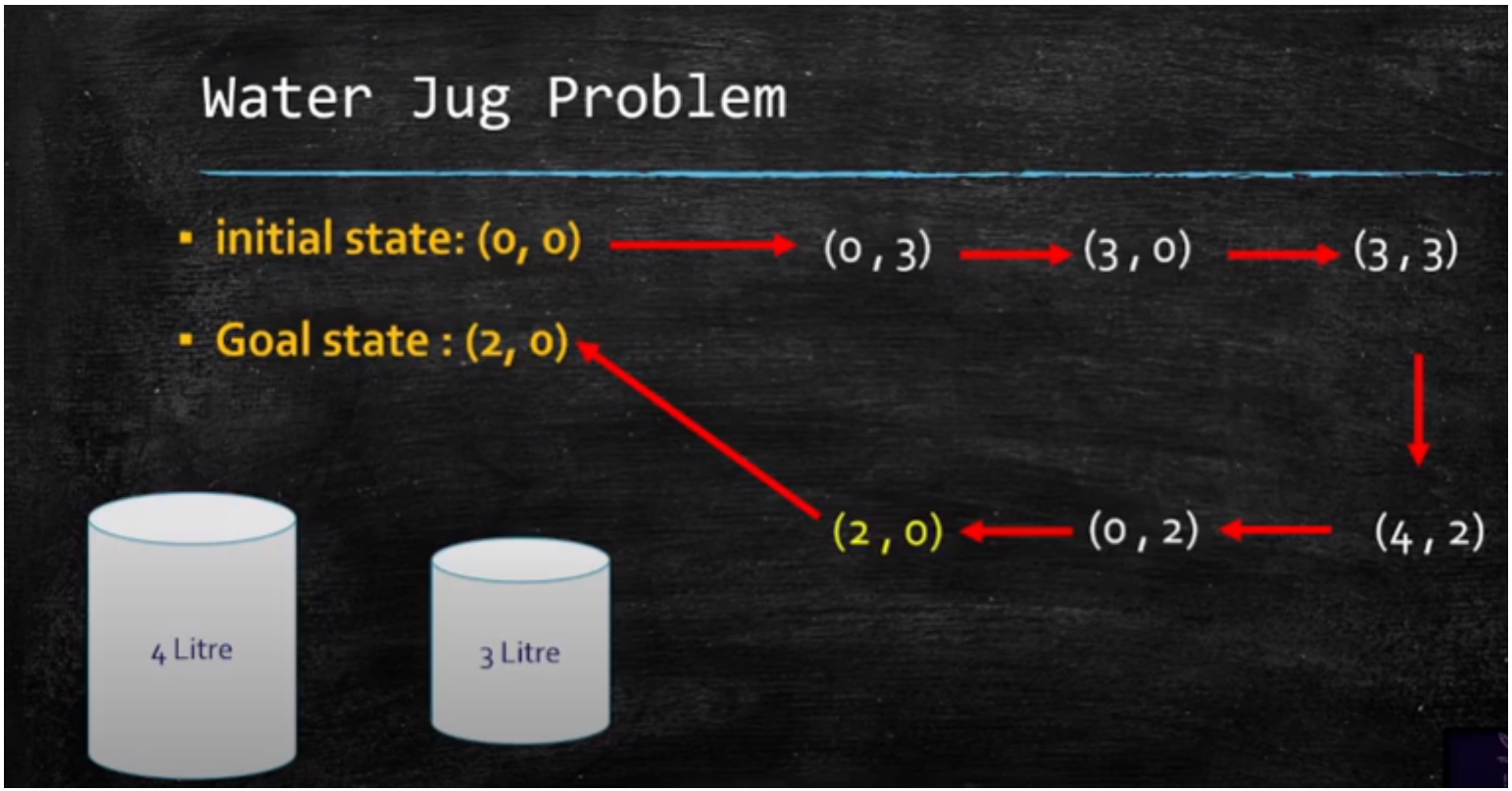
b-branch factor

d-depth factor

Water-Jug Problem

- **Problem:** You are given two jugs, a 4-Litter one and a 3-Litter one. Neither has any measuring mark on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 Litter of water into the 4-gallon jug.

http://aimaterials.blogspot.com/p/blog-page_18.html



https://www.youtube.com/watch?v=f4rD5vmtsp0

Watch Video

State space search

- **State space search** is a process used in the field of computer science, including artificial intelligence (AI), in which successive configurations or *states* of an instance are considered, with the intention of finding a *goal state* with the desired property.

State Space Representation

$$S : \langle S, A, Action(s), Result(s, a), Cost(s, a) \rangle,$$

- S is the set of all possible states;
- A is the set of possible actions, not related to a particular state but regarding all the state space;
- $Action(s)$ is the function that establish which action is possible to perform in a certain state;
- $Result(s, a)$ is the function that returns the state reached performing action a in state s
- $Cost(s, a)$ is the cost of performing an action a in state s. In many state spaces is a constant, but this is not true in general.

https://en.wikipedia.org/wiki/State_space_search

Problem Solving through AI

- Let us first discuss some terms related with AI problem solution methodology:
- — **Problem**: It is the question which is to be solved. For solving a problem it needs to be precisely defined. The definition means, defining the start state, goal state, other valid states and transitions.
- — **Search Space**: It is the complete set of states including start and goal states, where the answer of the problem is to be searched.
- — **Search**: It is the process of finding the solution in search space. The input to search space algorithm is problem and output is solution in form of action sequence.
- — **Well defined problem**: A problem description has three major components. Initial state, final state, space including transition function or path function. A path cost function assigns some numeric value to each path that indicates the goodness of that path.

Problem Solving through AI

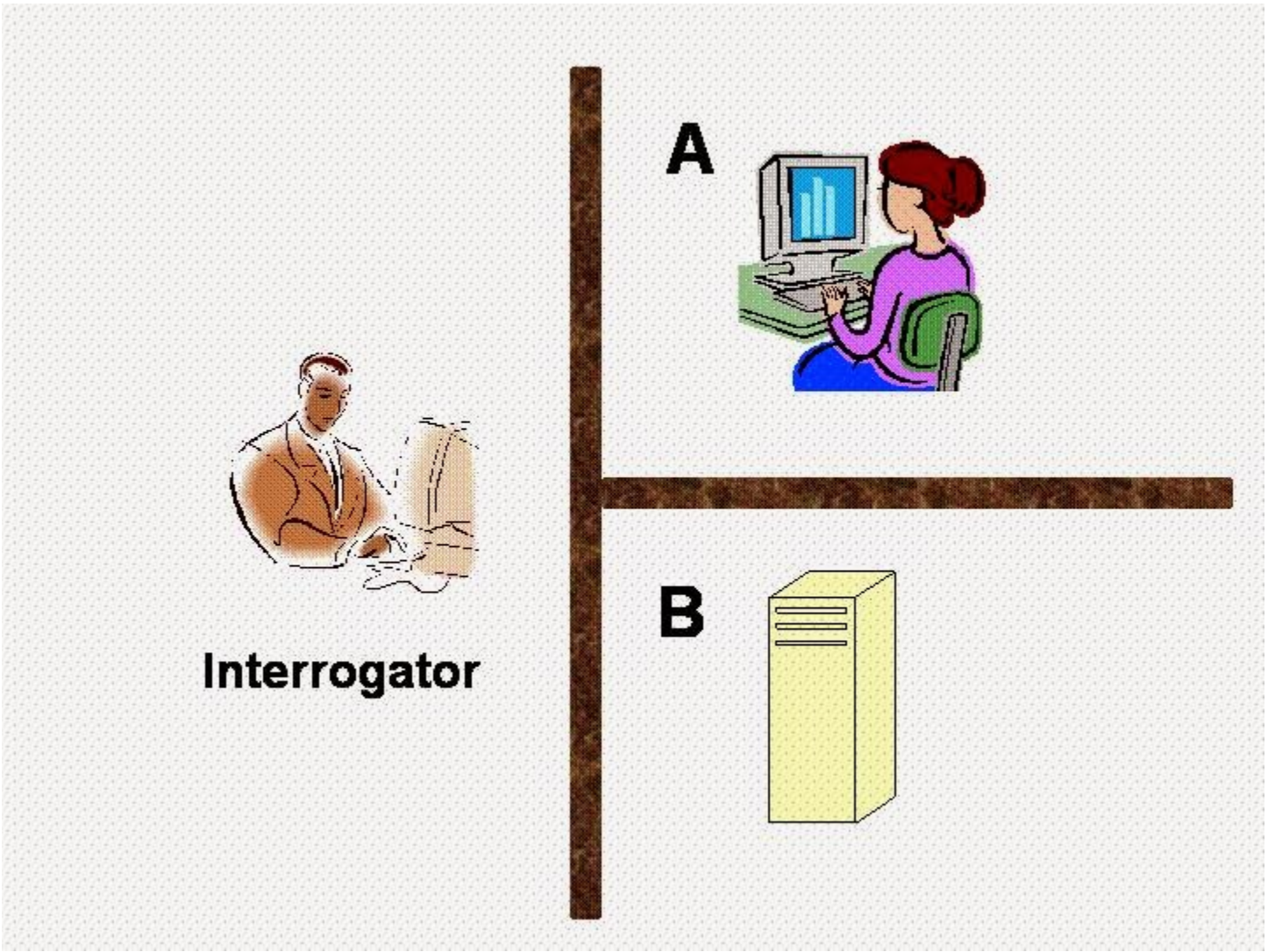
- — **Solution of the problem**: A solution of the problem is a path from initial state to goal state. The movement from start states to goal states is guided by transition rules. Among all the solutions, whichever solution has least path cost is called optimal solution.
- — Hence, it is evident that the method of solving problem through AI techniques involves the **process of defining the search space, deciding about start and goal state and then finding a path from start state to goal state through search space.** The movement from start state to goal state is guided by transition rules or production rules.

Search Trees

- **Search algorithms differ by the order in which they visit (reach) the states in the state graph following the edges between them.** For some algorithms, that order creates a tree superimposed over the state graph and whose root is the start state. We call that tree a search tree and will consider only the algorithms that grow it.

Turing Test

- — Turing in 1950 published an article in the Mind magazine, which triggered a controversial topic “Can a Machine Think?”
- — Turing proposed an ‘imitation game’ which was later modified to “Turing Test”. In the imitation game, the players are three humans- a male, a female and an interrogator. The interrogator who is shielded from the other two; asks questions to both of them and based on their typewritten answers determines who is male and who is female.
- — The aim of the male is to imitate the female and deceive the interrogator and the role of female is to provide replies that would inform the interrogator about her true gender.
- — The “Turing Test” is a slight variation of this imitation game wherein the human interrogator now communicates with the players via a teletype – one human and the other, a machine. The diagrammatic representation on next slide:


A
B
Interrogator

- Turing proposed that if the human interrogator in Room C is not able to identify who is in room A or in room B, then the machine possesses intelligence. Turing considered this as a sufficient test for attributing thinking capacity to a machine.
- — However, the test is not that easy as proposed. Humans are far more superior in creativity, common sense and reasoning. If any question is put on these topics, humans are sure to excel. On the numerical computation part, machines are accurate and faster.
- — Even though a delay loop can be introduced to provide answers after some time, machines can never be wrong in their computations whereas there exist a probability that humans may give incorrect answers after taking sufficiently long time.
- — As of today, Turing test is the ultimate test a machine must pass in order to be called as intelligent.

Unit-2

Search Algorithms

Search Algorithms in AI

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

https://www.geeksforgeeks.org/search-algorithms-in-ai/

Search Algorithms

- Search algorithms are algorithms that help in solving search problems. A search problem consists of a search space, start state, and goal state.
- These algorithms are important because they help in solving AI problems and support other systems such as neural networks and production systems.
- The main properties of search algorithms include optimality, completeness, time complexity, and space complexity.
- Search algorithms work by defining the problem (initial state, goal state, state space, space cost, etc) and conducting search operations to establish the best solution to the given problem.

Search Algorithm Terminologies

- **Search:** Searching is a step by step procedure to solve a search-problem in a given search space. A search problem can have three main factors:
  - **Search Space:** Search space represents a set of possible solutions, which a system may have.
  - **Start State:** It is a state from where agent begins **the search**.
  - **Goal test:** It is a function which observe the current state and returns whether the goal state is achieved or not.
- **Search tree:** A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.
- **Actions:** It gives the description of all the available actions to the agent.
- **Transition model:** A description of what each action do, can be represented as a transition model.
- **Path Cost:** It is a function which assigns a numeric cost to each path.
- **Solution:** It is an action sequence which leads from the start node to the goal node.
- **Optimal Solution:** If a solution has the lowest cost among all solutions.

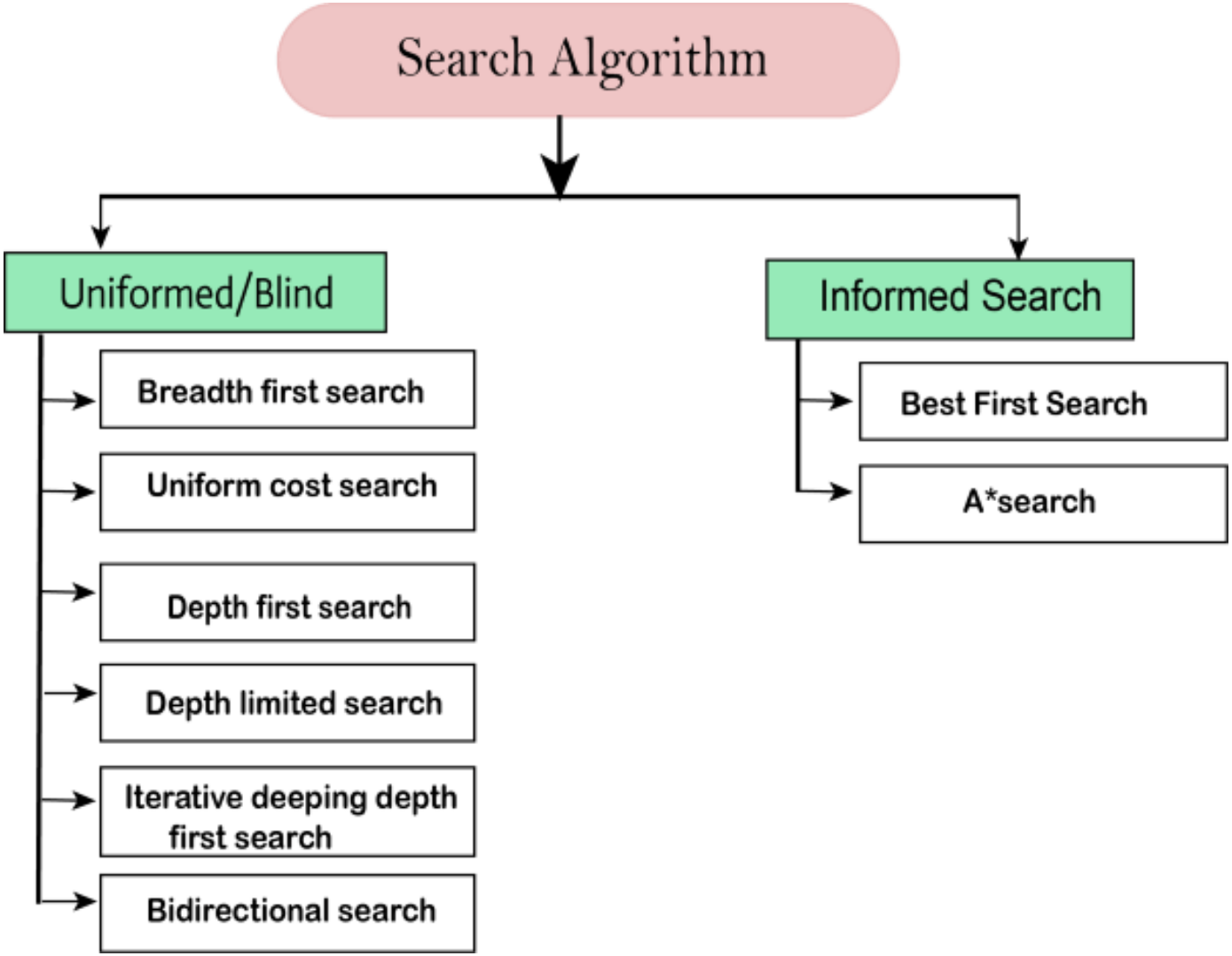
https://www.javatpoint.com/search-algorithms-in-ai

Properties of Search Algorithms

- **Completeness:** A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.
- **Optimality:** If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.
- **Time Complexity:** Time complexity is a measure of time for an algorithm to complete its task.
- **Space Complexity:** It is the maximum storage space required at any point during the search, as the complexity of the problem.

https://www.javatpoint.com/search-algorithms-in-ai

Types of search algorithms



https://www.javatpoint.com/search-algorithms-in-ai

Informed Search	Uninformed Search
It uses knowledge for the searching process.	It doesn’t use knowledge for searching process.
It finds solution more quickly.	It finds solution slow as compared to informed search.
It may or may not be complete.	It is always complete.
Cost is low.	Cost is high.
It consumes less time.	It consumes moderate time.
It provides the direction regarding the solution.	No suggestion is given regarding the solution in it.
It is less lengthy while implementation.	It is more lengthy while implementation.
Greedy Search, A* Search, Graph Search	Depth First Search, Breadth First Search

https://www.geeksforgeeks.org/difference-between-informed-and-uninformed-search-in-ai/

Random Search

- Random search is a technique where random combinations of the hyper parameters are used to find the best solution for the built model. It is similar to grid search, and yet it has proven to yield better results comparatively. The drawback of random search is that it yields high variance during computing. Since the selection of parameters is completely random; and since no intelligence is used to sample these combinations, luck plays its part.

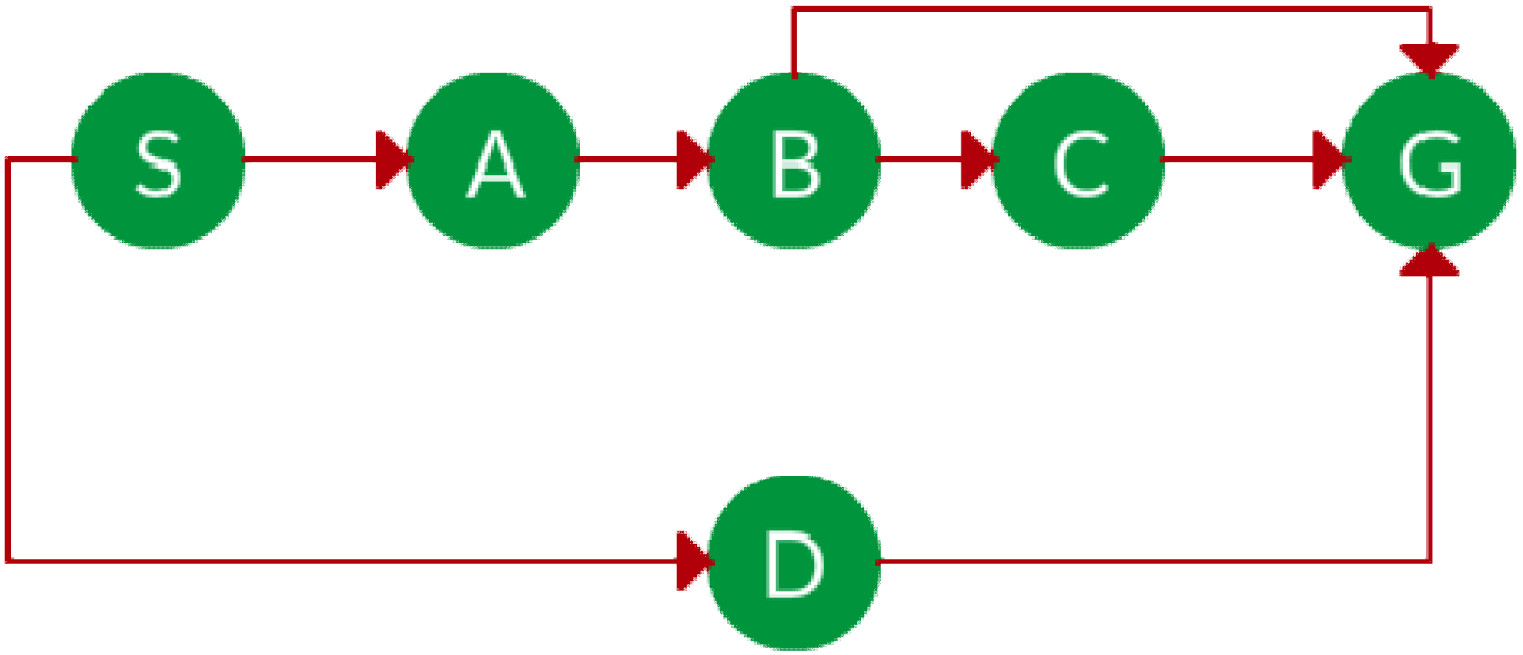
https://analyticsindiamag.com/why-is-random-search-better-than-grid-search-for-machine-learning/

Depth First Search (DFS)

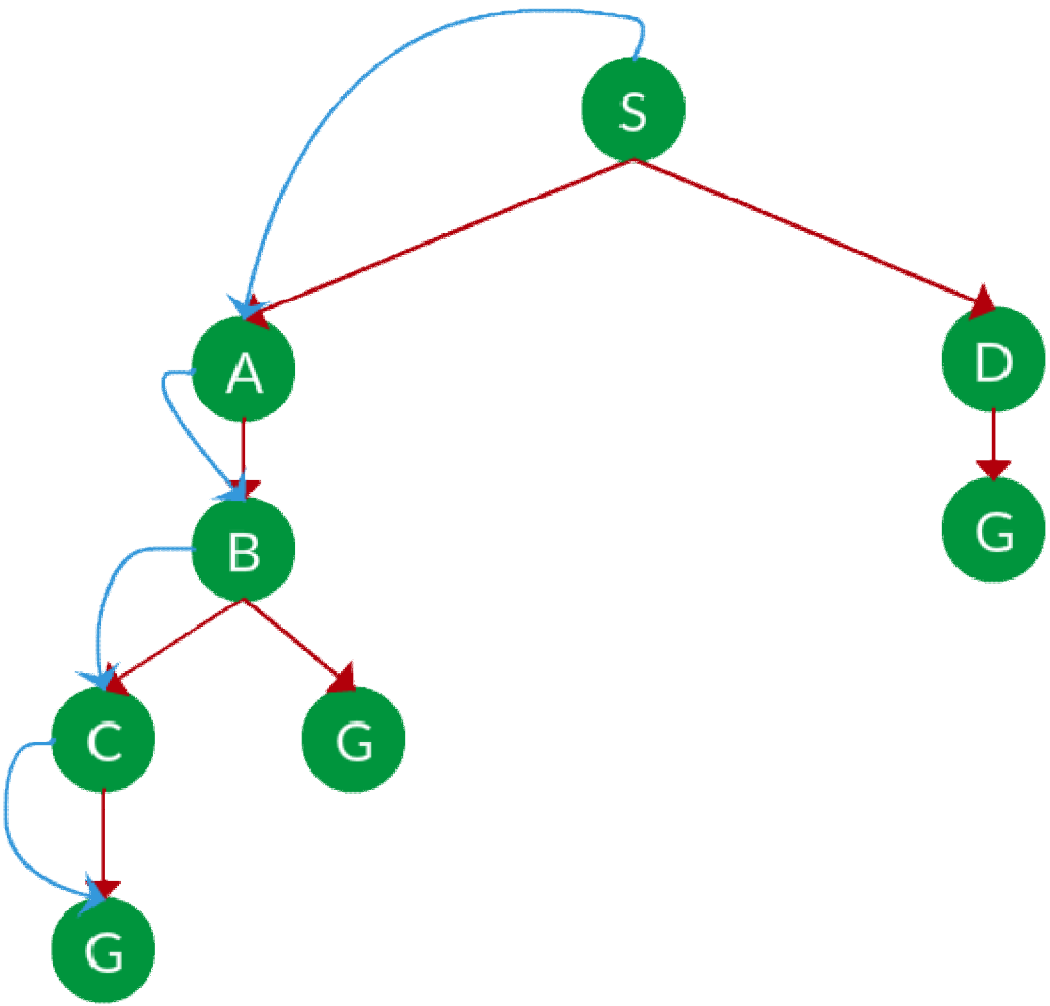
- Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

https://www.geeksforgeeks.org/search-algorithms-in-ai

Which solution would DFS find to move from node S to node G if run on the graph below?



https://www.geeksforgeeks.org/search-algorithms-in-ai


S
A
D
B
G
C
G
G

DFS Properties

- ***Time complexity:*** *Equivalent to the number of nodes traversed in DFS* $O(b^d)$. *b=branch factor*
- ***Space complexity:*** *Equivalent to how large can the fringe* get.*
- ***Completeness:*** *DFS may be complete if the search tree is finite, meaning for a given finite search tree, DFS will come up with a solution if it exists.*
- ***Optimality:*** *DFS is may not be optimal.*

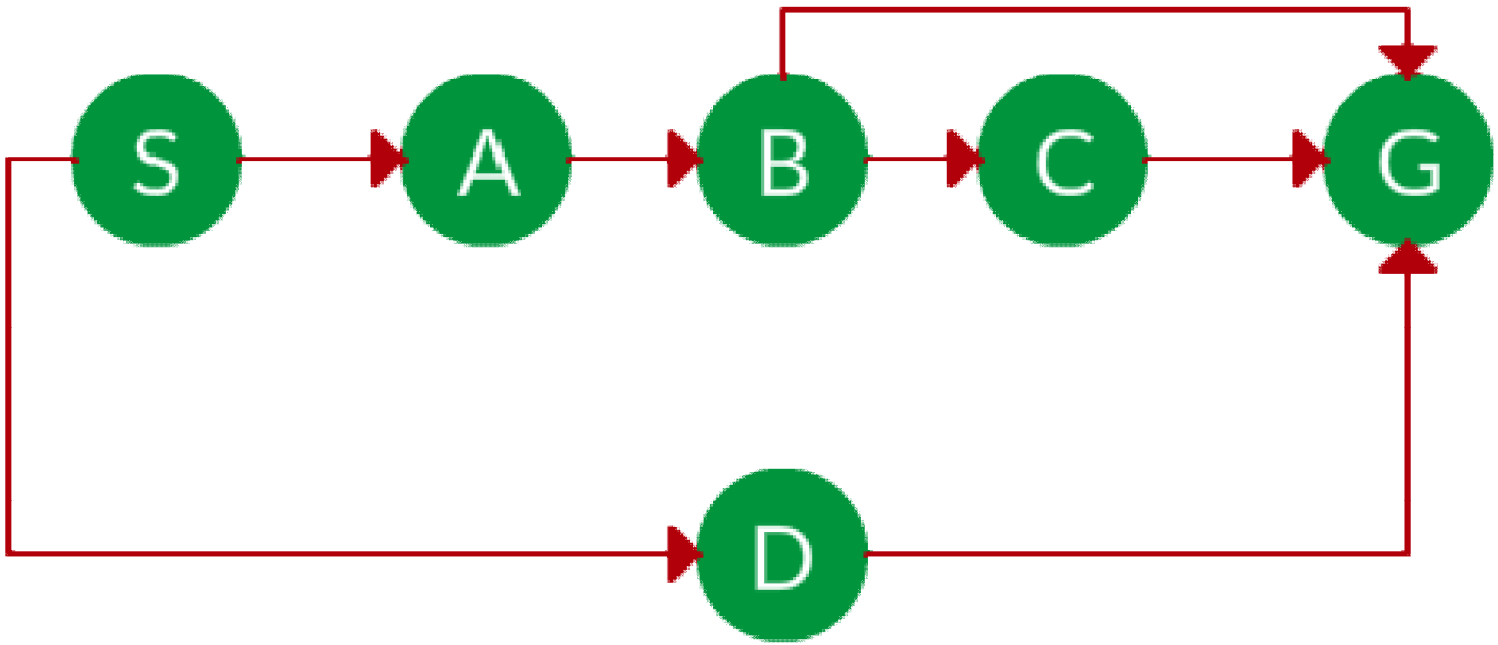
*A **fringe**, which is **a data structure used to store all the possible states (nodes) that you can go from the current states**

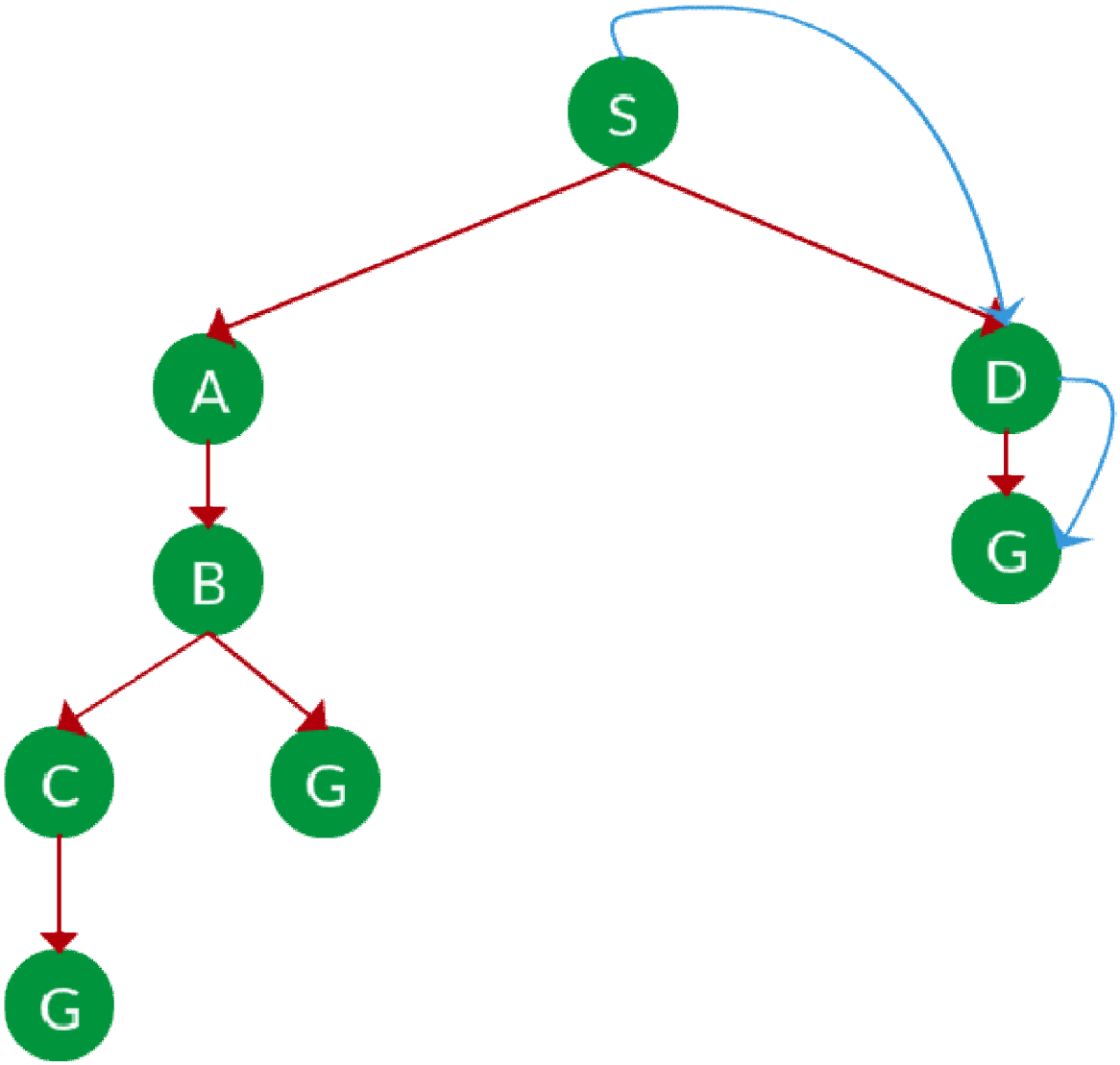
Breadth First Search

- It is another search algorithm in AI which traverses breadth wise to search the goal in a tree. It begins searching from the root node and expands the successor node before going expanding it further expands along breadth wise and traverses those nodes

https://www.analyticsvidhya.com/blog/2021/02/uninformed-search-algorithms-in-ai/

Which solution would BFS find to move from node S to node G if run on the graph below?




S
A
D
B
G
C
G
G

BFS Properties

- ***Completeness:*** *BFS is complete, meaning for a given search tree, BFS will come up with a solution if it exists.*
- ***Optimality:*** *BFS is optimal as long as the costs of all edges are equal.*
- ***Time complexity:*** *Equivalent to the number of nodes traversed in DFS* $O(b^d)$*. b=branch factor*
- ***Space complexity:*** *Equivalent to how large can the fringe* get.*

Informed Search

Heuristics function

- A heuristic is a *function* that estimates how close a state is to the goal state. For example Manhattan distance, Euclidean distance, etc. (Lesser the distance, closer the goal.)
- Different heuristics are used in different informed algorithms
- The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time

Heuristics function

- **Admissible Heuristics.**
  - An admissible heuristic never overestimates the cost of reaching the goal. Using an admissible heuristic will always result in an optimal solution.
- **Non-Admissible Heuristics.**
  - A non-admissible heuristic may overestimate the cost of reaching the goal. It may or may not result in an optimal solution.

8 Puzzle Problem with Heuristic Search

- Number of misplaced tiles.
- Sum of Eucledian distances of the tiles from their goal positions
- Sum of Manhattan distances of the tiles from their goal positions

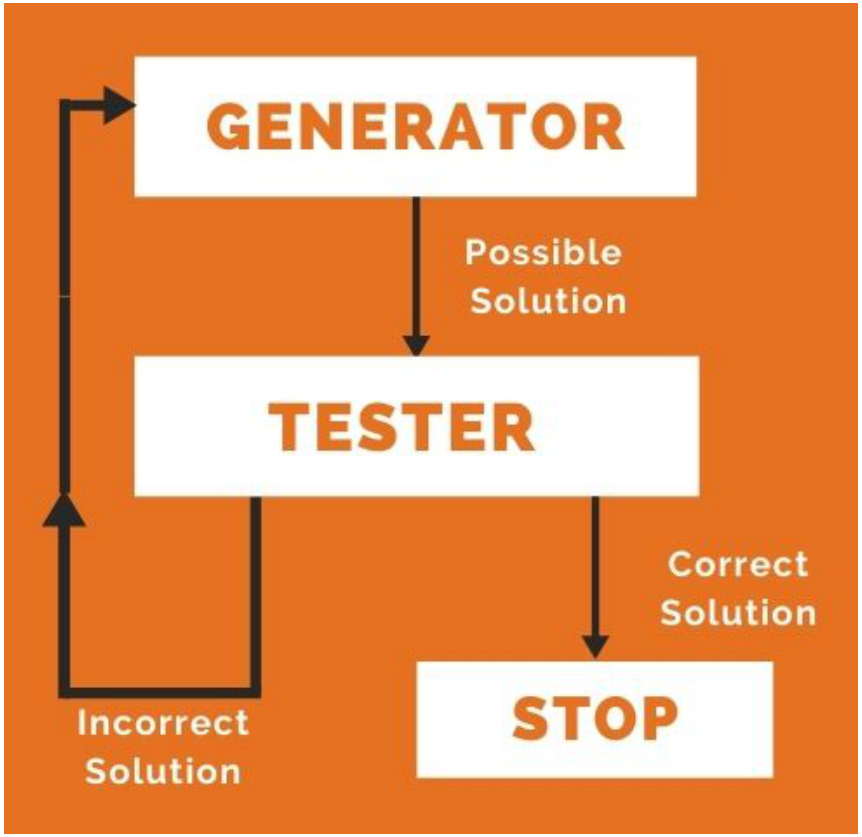
Generate and Test Search

- Generate and Test Search is a heuristic search technique based on Depth First Search with Backtracking which guarantees to find a solution if done systematically and there exists a solution.
- The evaluation is carried out by the heuristic function as all the solutions are generated systematically in generate and test algorithm

https://www.geeksforgeeks.org/generate-and-test-search/

Generate and Test Algorithm

1. Generate a possible solution.
2. Test to see if this is the expected solution.
3. If the solution has been found quit else go to step 1.



https://www.geeksforgeeks.org/generate-and-test-search/

Properties of Good Generators

- **Complete:** Good Generators need to be complete i.e. they should generate all the possible solutions and cover all the possible states. In this way, we can guaranty our algorithm to converge to the correct solution at some point in time.
- **Non Redundant:** Good Generators should not yield a duplicate solution at any point of time as it reduces the efficiency of algorithm thereby increasing the time of search and making the time complexity exponential.
- **Informed:** Good Generators have the knowledge about the search space which they maintain in the form of an array of knowledge. This can be used to search how far the agent is from the goal, calculate the path cost and even find a way to reach the goal.

https://www.geeksforgeeks.org/generate-and-test-search/

Best First Search Algorithm

- In BFS and DFS, when we are at a node, we can consider any of the adjacent as next node. So both BFS and DFS blindly explore paths without considering any cost function.
- The idea of Best First Search is to use an evaluation function to decide which adjacent is most promising and then explore. Best First Search falls under the category of Heuristic Search or Informed Search.
- Greedy Approach.

https://www.geeksforgeeks.org/best-first-search-informed-search/

BFS Algorithm

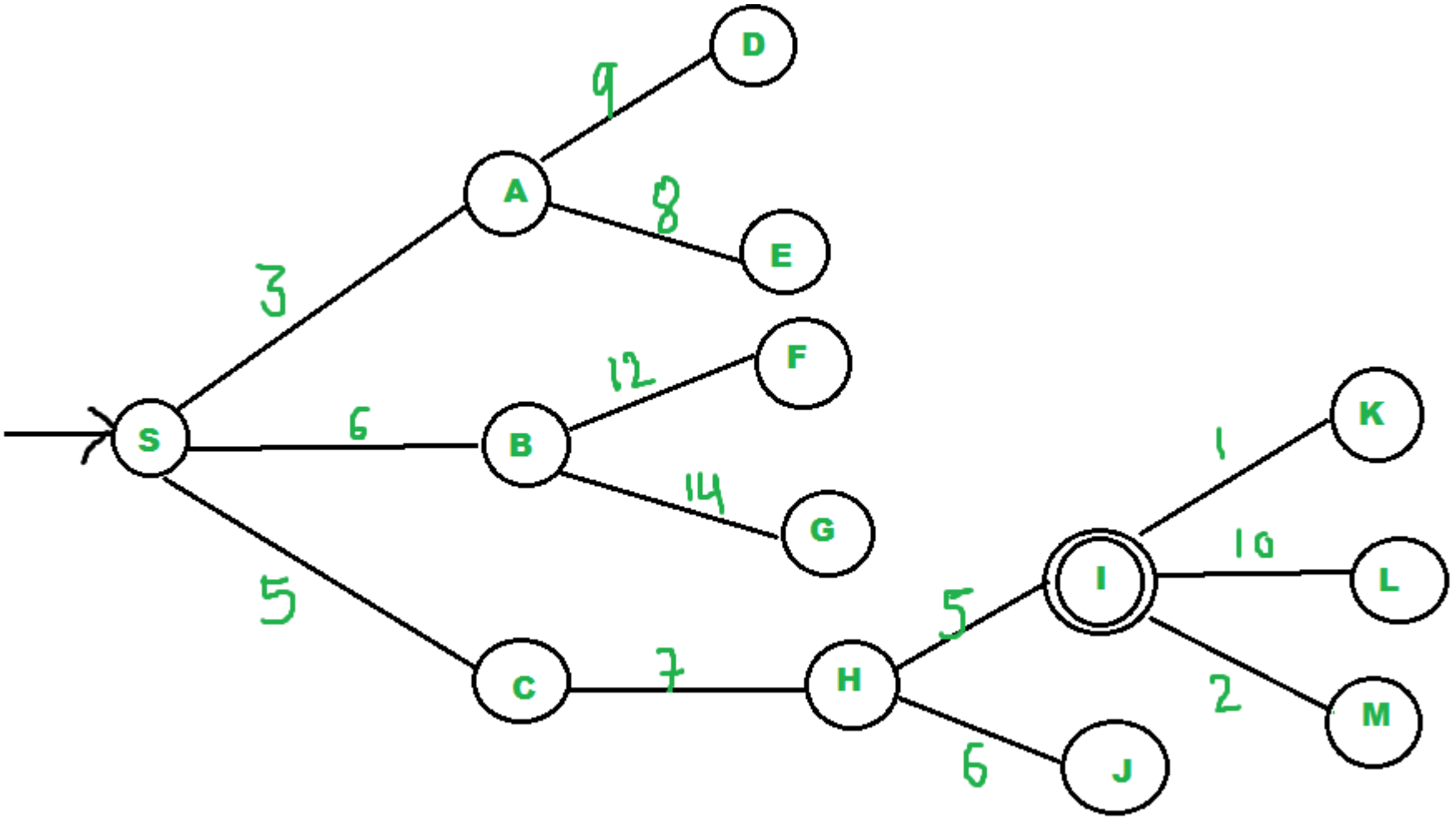
```
Best-First-Search(Graph g, Node start)
    1) Create an empty PriorityQueue
       PriorityQueue pq;
    2) Insert "start" in pq.
       pq.insert(start)
    3) Until PriorityQueue is empty
          u = PriorityQueue.DeleteMin
          If u is the goal
             Exit
          Else
             Foreach neighbor v of u
                If v "Unvisited"
                    Mark v "Visited"
                    pq.insert(v)
             Mark u "Examined"
End procedure
```

https://www.geeksforgeeks.org/best-first-search-informed-search/

Open & Close List

- An ‘Open’ list which keeps track of the current ‘immediate’ nodes available for traversal and ‘CLOSED’ list that keeps track of the nodes already traversed.

Apply BFS, Start = S Goal = I



https://www.geeksforgeeks.org/best-first-search-informed-search/

BFS Analysis

- worst case time complexity for Best First Search is O(n * Log n) where n is number of nodes. In worst case, we may have to visit all nodes before we reach goal. Note that priority queue is implemented using Min(or Max) Heap, and insert and remove operations take O(log n) time.
- Performance of the algorithm depends on how well the cost or evaluation function is designed.

Unit-3

Probabilistic Reasoning

Probabilistic Reasoning

- Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the **uncertainty** in knowledge.
- In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

https://www.javatpoint.com/probabilistic-reasoning-in-artifical-intelligence

causes of uncertainty to occur in the real world.

- Information occurred from unreliable sources.
- Experimental Errors
- Equipment fault
- Temperature variation
- Climate change.

Probability

- Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

$$\textbf{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

https://www.javatpoint.com/probabilistic-reasoning-in-artifical-intelligence

Conditional Probability

- Conditional probability is a probability of occurring an event when another event has already happened.
- Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

Conditional Probability Example

- In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?

$$P(A|B) = \frac{P(A \wedge B)}{P(B)} = \frac{0.4}{0.7} = 57\%$$

Joint probability distribution

- If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.
- **$P[x_1, x_2, x_3,....., x_n]$**, it can be written as the following way in terms of the joint probability distribution.
- **$= P[x_1 | x_2, x_3,....., x_n]P[x_2, x_3,....., x_n]$**
- **$= P[x_1 | x_2, x_3,....., x_n]P[x_2 | x_3,....., x_n]....P[x_{n-1} | x_n]P[x_n]$.**
- In general for each variable Xi, we can write the equation as:
- **$P(X_i | X_{i-1},....., X_1) = P(X_i | Parents(X_i))$**

Bayes‘ Theorem

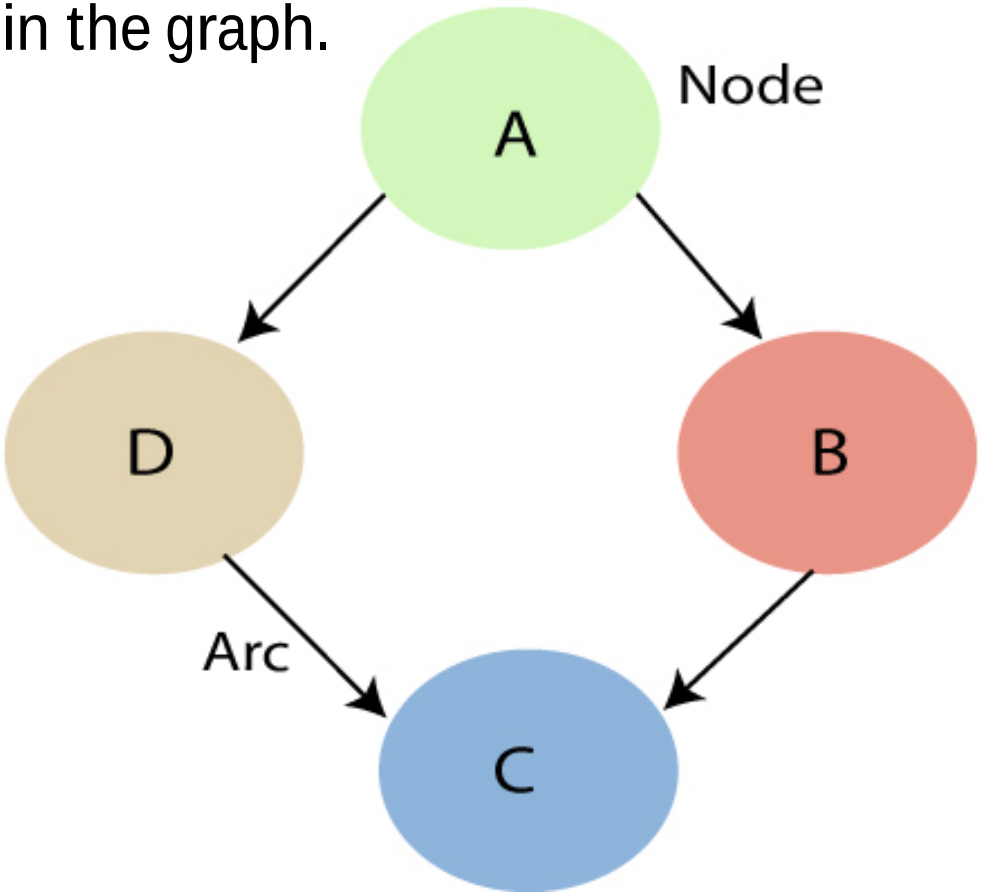
- Bayes' theorem is also known as **Bayes' rule, Bayes' law**, or **Bayesian reasoning**, which determines the probability of an event with uncertain knowledge.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayesian Belief Networks

- "A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph.“
- It can also be used in various tasks including prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction, and decision making under uncertainty.

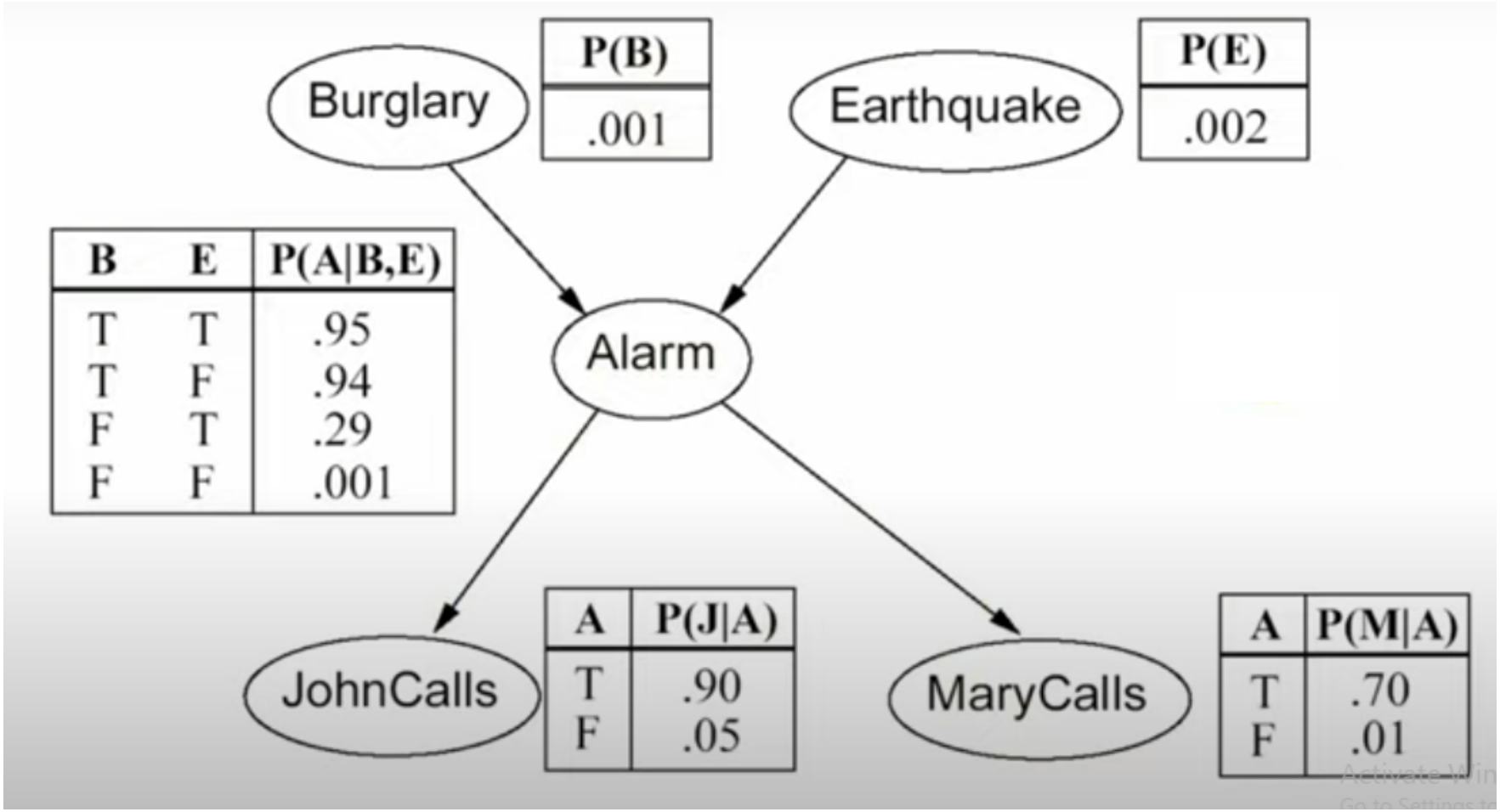
A Bayesian network graph is made up of nodes and Arcs (directed links), where Each **node** corresponds to the random variables, and a variable can be **continuous** or **discrete**. **Arc or directed arrows** represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.



https://www.javatpoint.com/bayesian-belief-network-in-artificial-intelligence

Example: Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbors David and Mary, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he got confused with the phone ringing and calls at that time too. On the other hand, Mary likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.

Problem: Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Mary both called the Harry.


Burglary
P(B)
.001
Earthquake
P(E)
.002
B E P(A|B,E)
T T .95
T F .94
F T .29
F F .001
Alarm
JohnCalls
A P(J|A)
T .90
F .05
MaryCalls
A P(M|A)
T .70
F .01

Solution:

$$P(j \wedge m \wedge a \wedge \neg b \wedge \neg e) = P(j \mid a)\, P(m \mid a)\, P(a \mid \neg b, \neg e)\, P(\neg b)\, P(\neg e)$$

$$= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998$$

$$= 0.00062$$

Video Lectures

- Example-1
  - https://www.youtube.com/watch?v=hEZjPZ-Ze0A
- Example-2
  - https://www.youtube.com/watch?v=iz7Kl2gcmlk

Must watch these video lectures for numerical examples as discussed in the class.

Unit-4

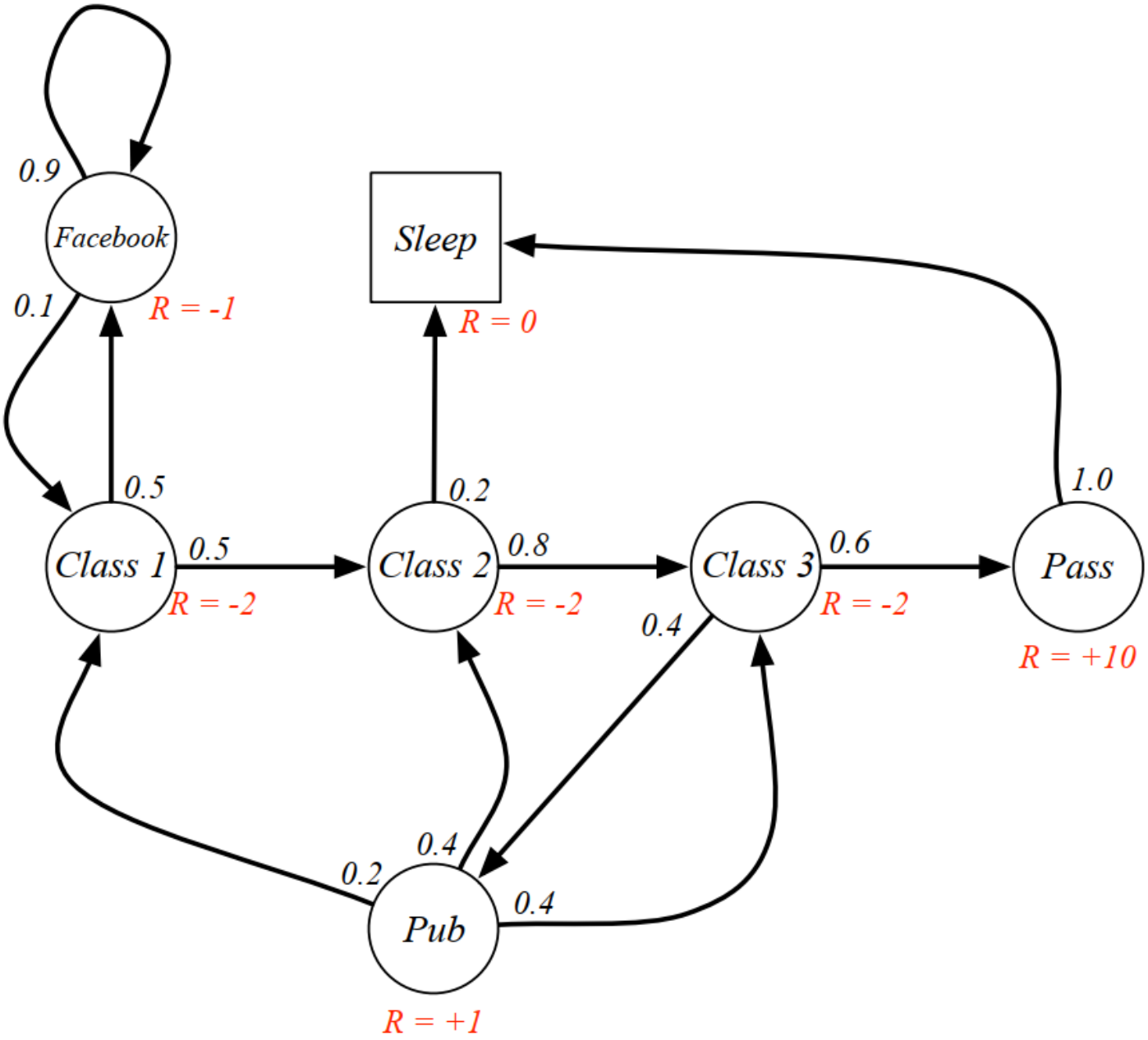
Markov Decision Process

Markov Property

- *"If the agent is present in the current state S1, performs an action a1 and move to the state s2, then the state transition from s1 to s2 only depends on the current state and future action and states do not depend on past actions, rewards, or states.“*
- Chess game, the players only focus on the current state and do not need to remember past actions or states.

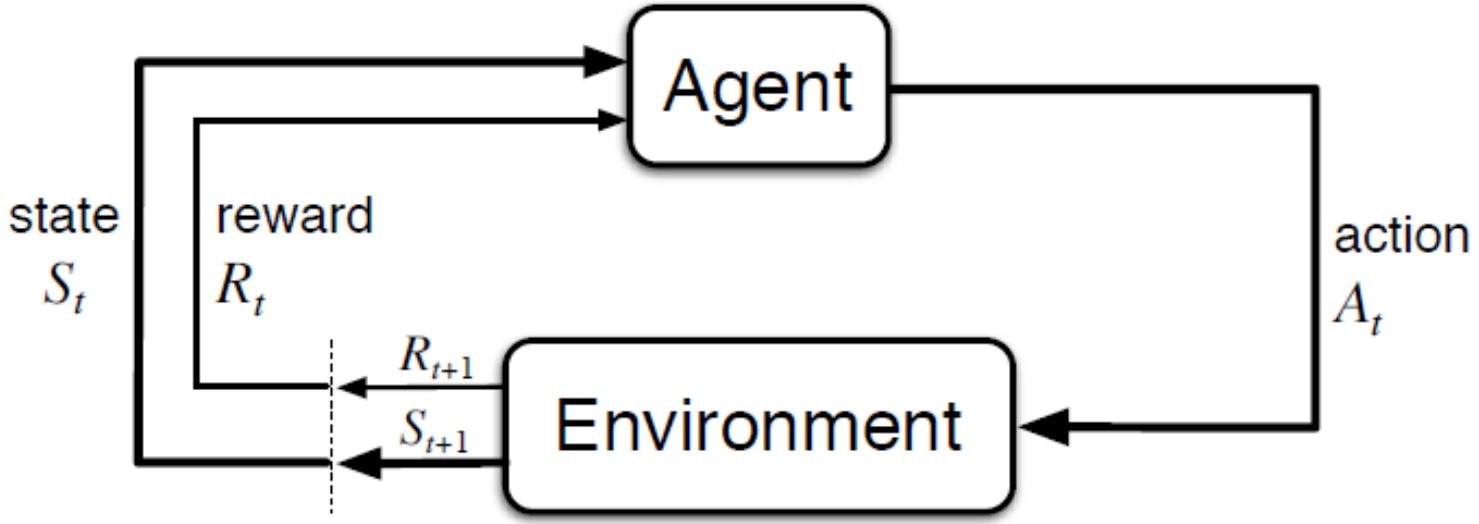
Introduction to Markov Decision Process

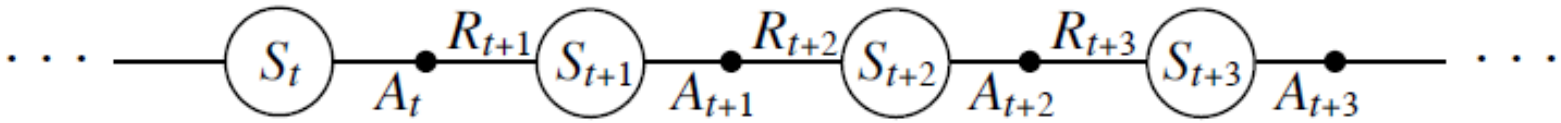
- A Markov Decision Process (MDP) model contains:
  - A set of possible world states **S**
  - A set of possible actions **A**
  - A real valued reward function **R(s,a)**
  - A description **T** of each action's effects in each state


0.9
Facebook
0.1
R = -1
Sleep
R = 0
0.5
0.2
1.0
Class 1
0.5
Class 2
0.8
Class 3
0.6
Pass
R = -2
R = -2
R = -2
R = +10
0.4
0.4
0.2
0.4
Pub
R = +1

MDP

- Any random process in which the probability of being in a given state depends only on the previous state, is a **markov process**.
- In other words, in the markov decision process setup, the environment's response at time t+1 depends only on the state and action representations at time t, and is independent of whatever happened in the past.





St: State of the agent at time t

At: Action taken by agent at time t

Rt: Reward obtained at time t

The above diagram clearly illustrates the iteration at each time step wherein the agent receives a reward R_{t+1} and ends up in state S_{t+1} based on its action A_t at a particular state S_t. The overall goal for the agent is to maximize the cumulative reward it receives in the long run. Total reward at any time instant t is given by:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

A **State** is a set of tokens that represent every state that the agent can be in.

A **Model** (sometimes called Transition Model) gives an action's effect in a state. In particular, T(S, a, S') defines a transition T where being in state S and taking an action 'a' takes us to state S' (S and S' may be same). For stochastic actions (noisy, non-deterministic) we also define a probability P(S'|S,a) which represents the probability of reaching a state S' if action 'a' is taken in state S. Note Markov property states that the effects of an action taken in a state depend only on that state and not on the prior history.
What is Actions?

An **Action** A is set of all possible actions. A(s) defines the set of actions that can be taken being in state S.

A **Policy** is a solution to the Markov Decision Process. A policy is a mapping from S to a. It indicates the action 'a' to be taken while in state S.

What is Hidden Markov Model?

- A Hidden Markov Model is a statistical Markov Model (chain) in which the system being modeled is assumed to be a Markov Process with hidden states (or unobserved) states.
- Considering the problem statement of our example is about predicting the sequence of seasons, then it is a Markov Model. Besides, our requirement is to predict the outfits that depend on the seasons. In case of initial requirement, we don’t possess any hidden states, the observable states are seasons while in the other, we have both the states, hidden(season) and observable(Outfits) making it a Hidden Markov Model.

Utility Theory

- The agent‘s preferences are captured by a utility function, U(s), which assigns a single number to express desirability of a state.
- Utility Theory is the discipline that lays out the foundation to create and evaluate Utility Functions. Typically, Utility Theory uses the notion of Expected Utility (EU) as a value that represents the average utility of all possible outcomes of a state, weighted by the probability that the outcome occurs. The other key concept of Utility Theory is known as the Principle of Maximum Utility(MEU) which states that any rational agent should choose to maximize the agent’s EU.

Utility Function

- Utility functions are a product of Utility Theory which is one of the disciplines that helps to address the challenges of building knowledge under uncertainty. Utility theory is often combined with probabilistic theory to create what we know as decision-theoretic agents. Conceptually, a decision-theoretic agent is an AI program that can make rational decisions based on what it believes and what it wants.

Utility Function

- In many AI scenarios, agents don’t have the luxury of operating in an environment in which they know the final outcome of every possible state. Those agents operate under certain degree of uncertainly and need to rely on probabilities to quantify the outcome of possible states. That probabilistic function is what we call Utility Functions.

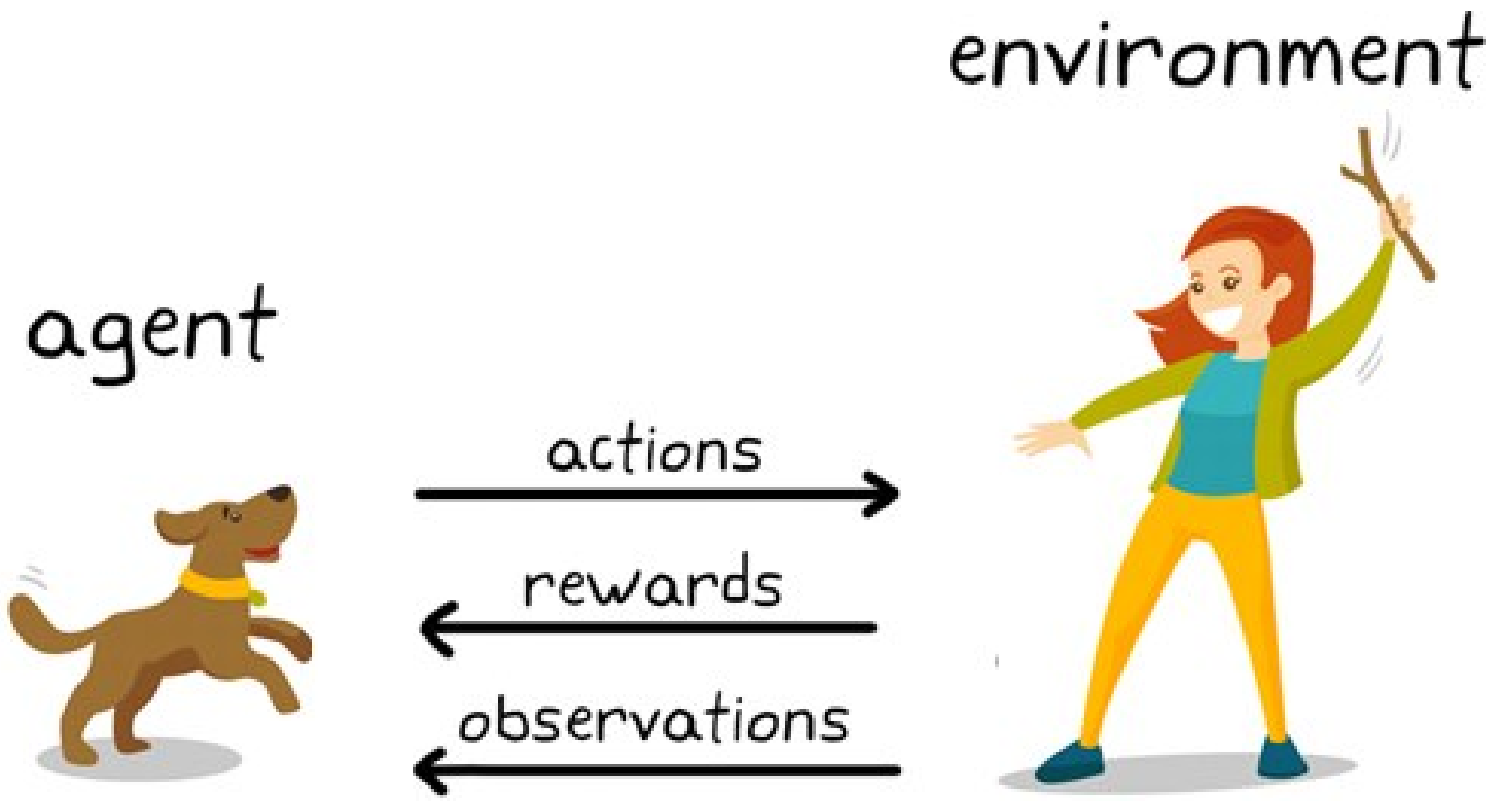
Partially Observable MDP

- **Fully observable vs. partially observable**: *If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is* ***fully observable***. A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action; relevance, in turn, depends on the performance measure. Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world. An environment might be **partially observable** because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data—for example, a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and an automated taxi cannot see what other drivers are thinking. If the agent has no sensors at all then the environment is unobservable.

Unit-5

Reinforcement Learning

What is Reinforcement Learning?



What is Reinforcement Learning?

- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.

https://www.javatpoint.com/reinforcement-learning

- Since there is no labeled data, so the agent is bound to learn by its experience only.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself. The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.

Elements of Reinforcement Learning

1. Policy
2. Reward Signal
3. Value Function
4. Model of the environment

Policy

- A policy can be defined as a way how an agent behaves at a given time. It maps the perceived states of the environment to the actions taken on those states. A policy is the core element of the RL as it alone can define the behavior of the agent. In some cases, it may be a simple function or a lookup table, whereas, for other cases, it may involve general computation as a search process. It could be deterministic or a stochastic policy:

Reward Signal

- **Reward Signal:** The goal of reinforcement learning is defined by the reward signal. At each state, the environment sends an immediate signal to the learning agent, and this signal is known as a **reward signal**. These rewards are given according to the good and bad actions taken by the agent. The agent's main objective is to maximize the total number of rewards for good actions.

Value Function

- **Value Function:** The value function gives information about how good the situation and action are and how much reward an agent can expect. A reward indicates the **immediate signal for each good and bad action**, whereas a value function specifies **the good state and action for the future**.

Model

- **Model:** The last element of reinforcement learning is the model, which mimics the behavior of the environment. With the help of the model, one can make inferences about how the environment will behave. Such as, if a state and an action are given, then a model can predict the next state and reward.

Passive and Active Reinforcement learning

- Both active and passive reinforcement learning are types of RL. In case of passive RL, the agent's policy is fixed which means that it is ***told what to do***. In contrast to this, in active RL, an agent ***needs to decide what to do*** as there's no fixed policy that it can act on. Therefore, the goal of a passive RL agent is to execute a fixed policy (sequence of actions) and evaluate it while that of an active RL agent is to act and learn an optimal policy.

https://towardsdatascience.com/explaining-reinforcement-learning-active-vs-passive-a389f41e7195

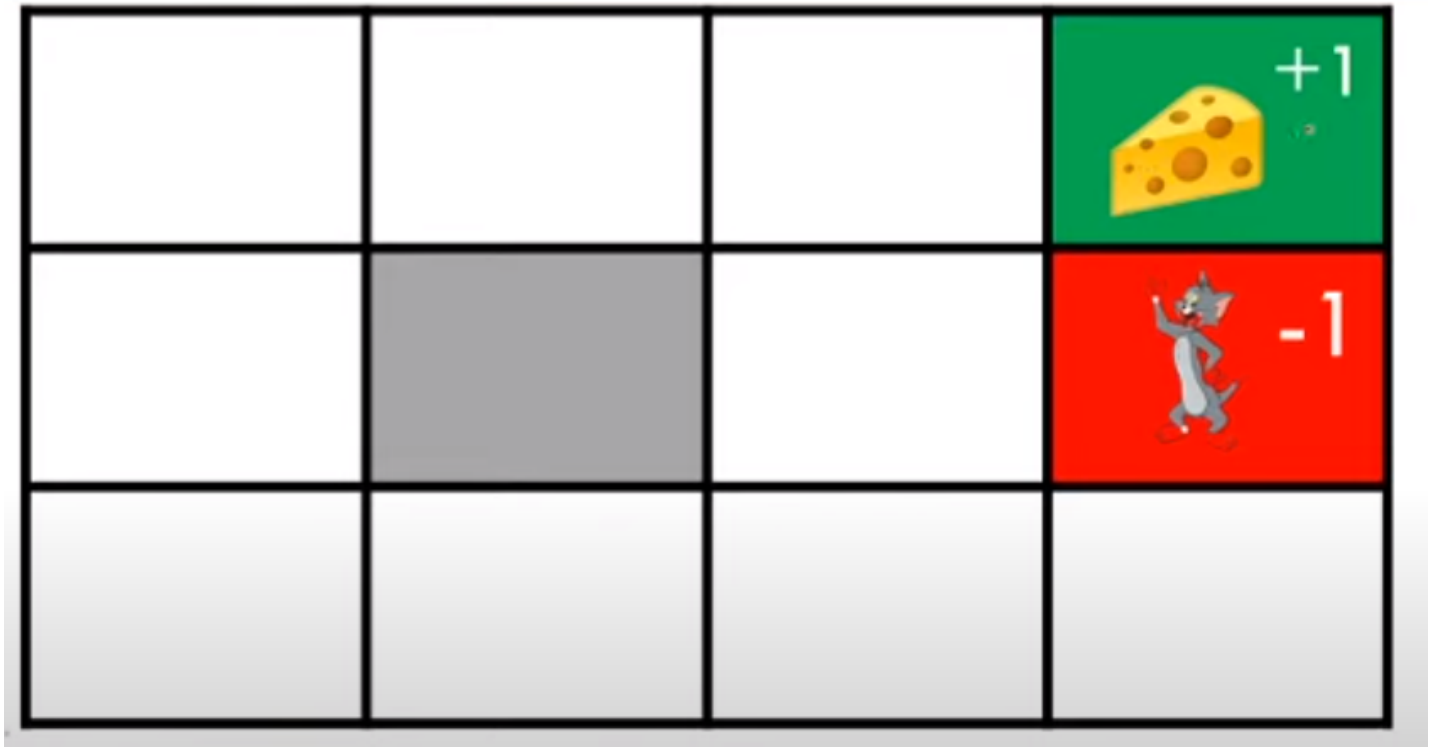
- Passive learning
  - The agent acts based on a fixed policy π and tries to learn how good the policy is by observing the world go by.
  - Analogous to policy evaluation in policy iteration
- Active learning
  - The agent attempts to f d in an optimal (or at least good) policy by exploring different actions in the world
  - Analogous to solving the underlying MDP

Passive Reinforcement Learning

- As the goal of the agent is to evaluate how good an optimal policy is, the agent needs to learn the expected utility $U\pi(s)$ for each state s. This can be done in three ways.
  1. ***Direct Utility Estimation***
  2. ***Adaptive Dynamic Programming(ADP)***
  3. ***Temporal Difference Learning (TD)***

Direct Utility Estimation

- In this method, the agent executes a **sequence of trials or runs** (sequences of states-actions transitions that continue until the agent reaches the terminal state). Each trial gives a sample value and the agent estimates the utility based on the samples values. Can be calculated as **running averages of sample values**.



(1,1)→(1,2)→(1,3)→(1,2)→(1,3)→(2,3)→(3,3) →(3,4) +1
(1,1)→(1,2)→(1,3)→(2,3)→(3,3)→(3,2)→(3,3)→(3,4) +1
(1,1)→(2,1)→(3,1)→(3,2)→(4,2) -1

Direct Utility Estimation

Suppose we observe the following trial:

$$(1,1)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$$

The total reward starting at (1,1) is 0.72. We call this a sample of the observed-reward-to-go for (1,1).

For (1,2) there are two samples for the observed-reward-to-go (assuming γ=1):

1. $(1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$ [Total: 0.76]
2. $(1,2)_{-0.04} \rightarrow (1,3)_{-0.04} \rightarrow (2,3)_{-0.04} \rightarrow (3,3)_{-0.04} \rightarrow (4,3)_{+1}$ [Total: 0.84]

Direct Utility Estimation

- Direct Utility Estimation keeps a running average of the observed reward-to-go for each state
- Eg. For state (1,2), it stores (0.76+0.84)/2 = 0.8
- As the number of trials goes to infinity, the sample average converges to the true utility


0.812
0.868
0.918
+ 1
0.762
0.660
−1
0.705
0.655
0.611
0.388

Adaptive Dynamic Programming(ADP)

- ADP is a smarter method than Direct Utility Estimation **as it runs trials to learn the model of the environment** by estimating the utility of a state as a sum of reward for being in that state and the expected discounted reward of being in the next state.

Temporal Difference Learning (TD)

- TD learning does not require the agent to learn the transition model. The update occurs between successive states and agent only updates states that are directly affected.

Active Reinforcement Learning

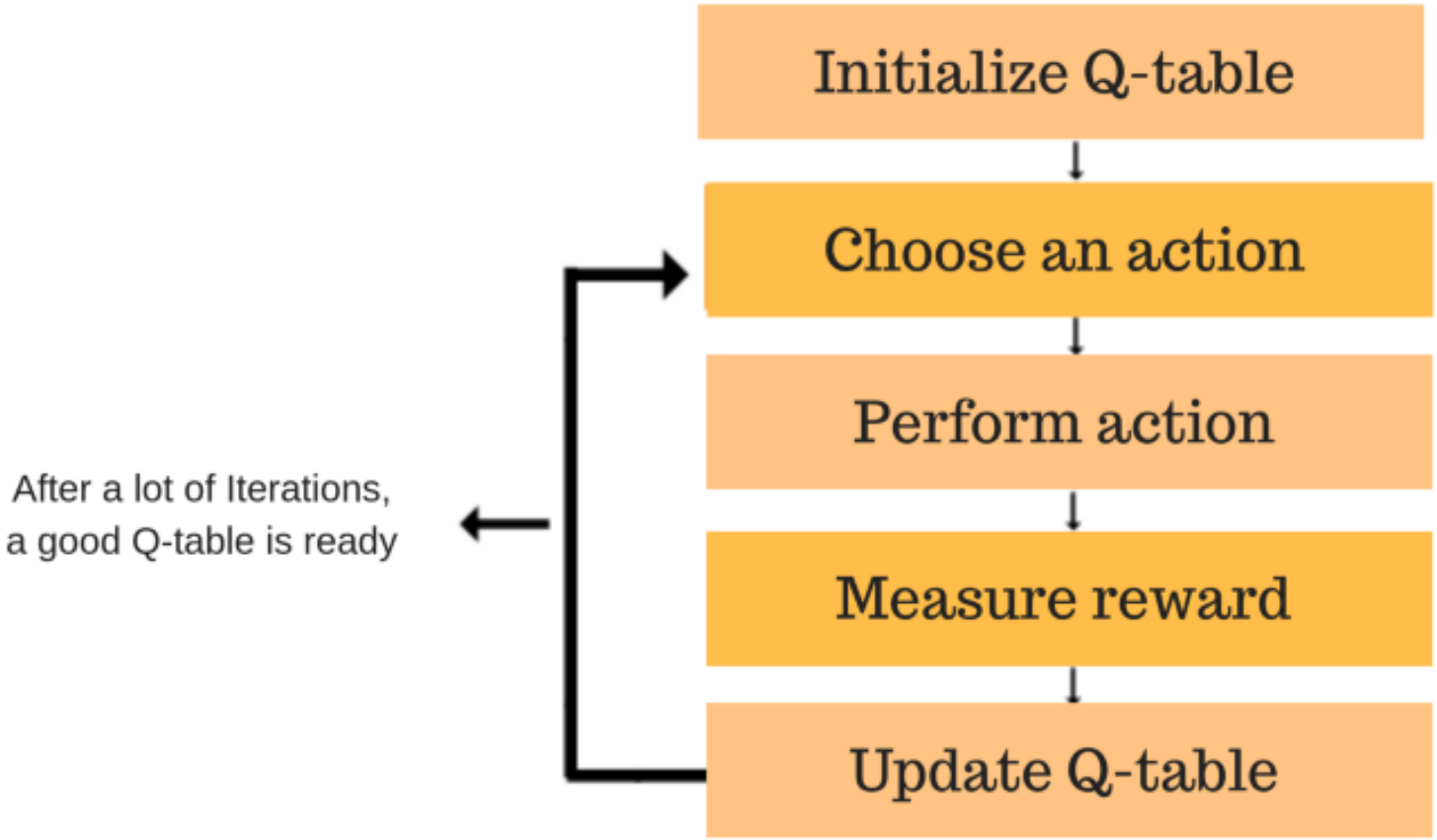
- **Active RL**, an agent ***needs to decide what to do*** as there's no fixed policy that it can act on. Therefore, the goal of a passive RL agent is to execute a fixed policy (sequence of actions) and evaluate it while that of an active RL agent is to act and learn an optimal policy.

Q-Learning

- Q-learning is a popular model-free reinforcement learning algorithm based on the Bellman equation.
- **The main objective of Q-learning is to learn the policy which can inform the agent that what actions should be taken for maximizing the reward under what circumstances.**
- It is an **off-policy RL** that attempts to find the best action to take at a current state.
- The goal of the agent in Q-learning is to maximize the value of Q.

Q-table

- A Q-table or matrix is created while performing the Q-learning. The table follows the state and action pair, i.e., [s, a], and initializes the values to zero. After each action, the table is updated, and the q-values are stored within the table.


Initialize Q-table
Choose an action
Perform action
Measure reward
Update Q-table
After a lot of Iterations,
a good Q-table is ready

Q learning algorithm

For each s, a initialize the table entry $\hat{Q}(s, a)$ to zero

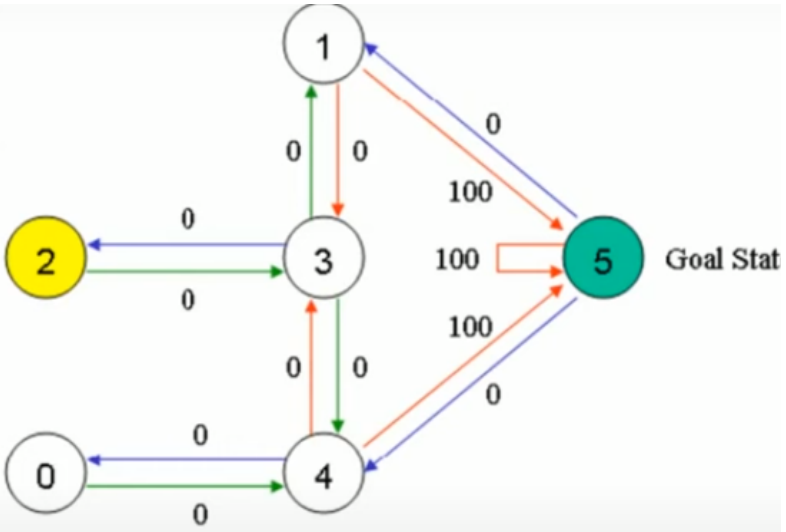
Observe the current state s

Do forever:

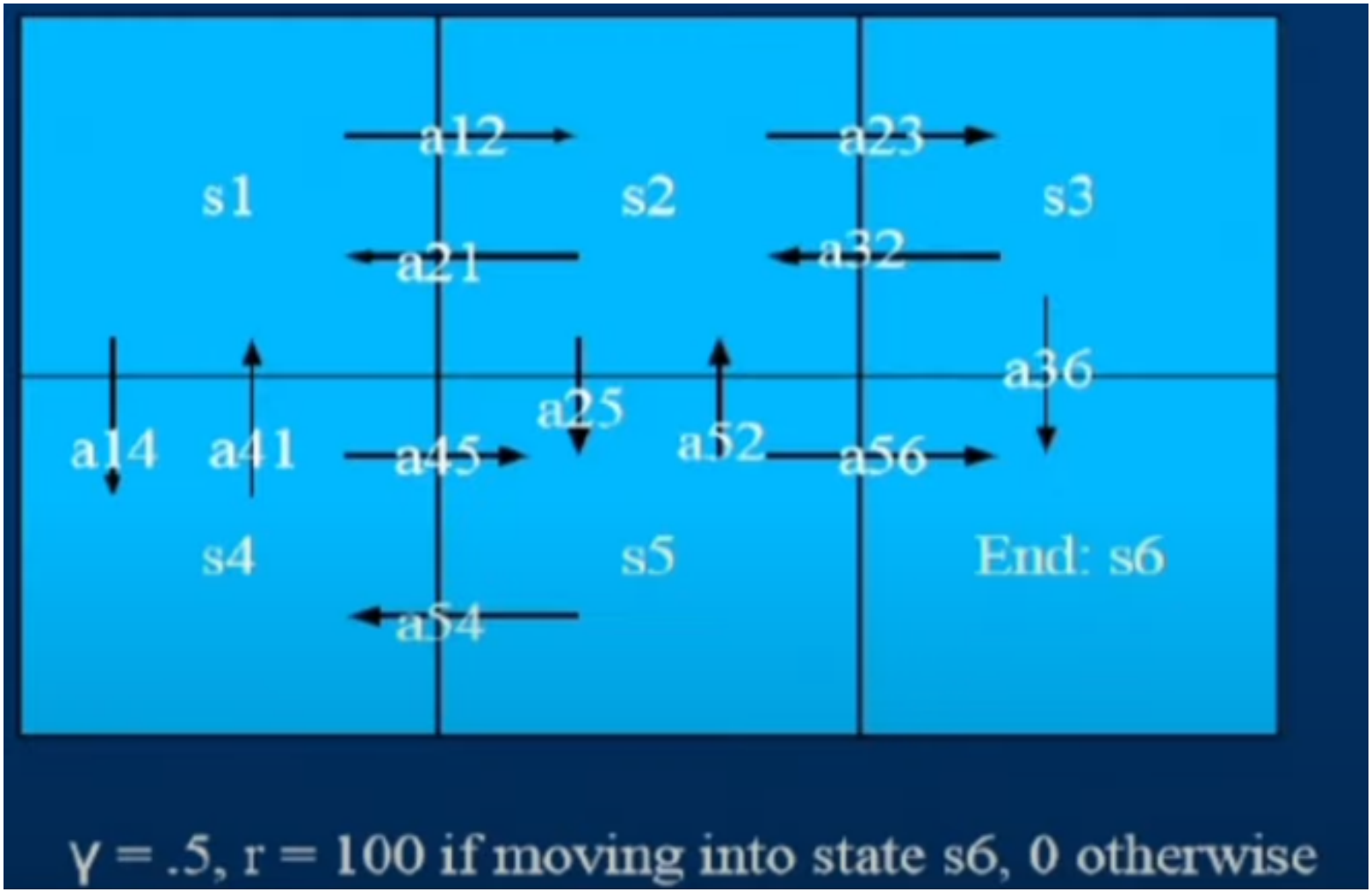
- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$



https://www.youtube.com/watch?v=_O9a5xbBe-s&list=PL4gu8xQu0_5JBO1FKRO5p20wc8DprlOgn&index=56



https://www.youtube.com/watch?v=bMQZjyP08Pc

s1, a12	0
s1, a14	0
s2, a21	0
s2, a23	0
s2, a25	0
s3, a32	0
s3, a36	0
s4, a41	0
s4, a45	0
s5, a54	0
s5, a52	0
s5, a56	0

s1, a12	25
s1, a14	25
s2, a21	12.5
s2, a23	50
s2, a25	25
s3, a32	25
s3, a36	100
s4, a41	12.5
s4, a45	50
s5, a54	25
s5, a52	25
s5, a56	100

Reinforcement Learning	Supervised Learning
RL works by interacting with the environment.	Supervised learning works on the existing dataset.
The RL algorithm works like the human brain works when making some decisions.	Supervised Learning works as when a human learns things in the supervision of a guide.
There is no labeled dataset is present	The labeled dataset is present.
No previous training is provided to the learning agent.	Training is provided to the algorithm so that it can predict the output.
RL helps to take decisions sequentially.	In Supervised learning, decisions are made when input is given.

Parameters	Reinforcement Learning	Supervised Learning
Decision style	reinforcement learning helps you to take your decisions sequentially.	In this method, a decision is made on the input given at the beginning.
Works on	Works on interacting with the environment.	Works on examples or given sample data.
Dependency on decision	In RL method learning decision is dependent. Therefore, you should give labels to all the dependent decisions.	Supervised learning the decisions which are independent of each other, so labels are given for every decision.
Best suited	Supports and work better in AI, where human interaction is prevalent.	It is mostly operated with an interactive software system or applications.
Example	Chess game	Object recognition