List of algorithms

* **Depth First Search**

- 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.

- Unlike BFS, a DFS algorithm traverses a tree or graph from the parent vertex down to its children and grandchildren vertices in a single path until it reaches a dead end.

- When there are no more vertices to navigate in a path, the DFS algorithm will back-trace a point where it can choose another path to take. It repeats the process until all vertices are visited

> **Breadth First Search**

Breadth-first search is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root, and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level.

- Traversal happens through one entire level of children nodes first, before moving on to traverse through the grandchildren nodes. And then traversal occurs through an entire level of grandchildren nodes before going on to traverse through great-grandchildren nodes.

> **Dijkstra's Algorithm**

- Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks.. Like Prim’s MST (minimum spanning tree), we generate a SPT (shortest path tree) with given source as root.

- Two sets are considered, one set contains vertices included in shortest path tree, other set includes vertices not yet included in shortest path tree.

- At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.

* **Uniform Cost Search**

Uniform Cost Search is one of the best algorithm for a searchproblem, which does not involve the use of heuristics. It can solve any general graph for optimal cost. Uniform Cost Search as it sounds searches in branches which are more or less the same in cost.

- Uniform cost search is a tree search algorithm related to breadth-first search. Whereas breadth-first search determines a path to the goal state that has the least number of edges, uniform cost search determines a path to the goal state that has the lowest weight.

- It's worth observing that uniform cost search assumes that no edges have negative weight. If any edges have negative weight, then it is possible that a path p begins with a vertex whose edge to its parent has a high positive weight, which will exclude it from consideration by the search.

**> Minimum-Spanning-Tree**

A minimum spanning tree or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight

- A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

1. Kruskal's Algorithm

- Kruskal's algorithm is a minimum-spanning-tree algorithm which finds an edge of the least possible weight that connects any two trees in the forest. It is a greedy algorithm in graph theory as it finds a minimum spanning tree for a connected weighted graph adding increasing cost arcs at each step.

2. Prim's Algorithm

- Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.

**> Priority Queues**

- Priority Queue is similar to queue where we insert an element from the back and remove an element from front, but with a one difference that the logical order of elements in the priority queue depends on the priority of the elements.

The binary heap is a data structure that can efficiently support the basic **priority**-**queue**operations. In a binary heap, the items are stored in an array such that each key is guaranteed to be larger than (or equal to) the keys at two other specific positions.

- The element with highest priority will be moved to the front of the queue and one with lowest priority will move to the back of the queue.

**> A\* Search**

- A\* is a computer algorithm that is widely used in pathfinding and graph traversal. The algorithm efficiently plots a walkable path between multiple nodes, or points, on the graph.

 A\* is a computer algorithm that is widely used in pathfinding and graph traversal, which is the process of finding a path between multiple points, called "nodes". It enjoys widespread use due to its performance and accuracy.

- On a map with many obstacles, pathfinding from points A to B can be difficult. A robot, for instance, without getting much other direction, will continue until it encounters an obstacle, as in the path-finding example to the left below.

- However, the A\* algorithm introduces a heuristic into a regular graph-searching algorithm, essentially planning ahead at each step so a more optimal decision is made.

**> Minimax**

- Minimax is a decision-making algorithm, typically used in a turn-based, two player games. The goal of the algorithm is to find the optimal next move.

Minimax is a decision rule used in artificial intelligence, decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case scenario. When dealing with gains, it is referred to as "maximin"—to maximize the minimum gain.

- In the algorithm, one player is called the maximizer, and the other player is a minimizer. If we assign an evaluation score to the game board, one player tries to choose a game state with the maximum score, while the other chooses a state with the minimum score.

- In other words, the maximizer works to get the highest score, while the minimizer tries get the lowest score by trying to counter moves.

**> Expectimax Search**

- The expectiminimax algorithm is a variation of the minimax algorithm, for use in artificial intelligence systems that play two-player zero-sum games, such as backgammon, in which the outcome depends on a combination of the player's skill and chance elements such as dice rolls.

**Expectimax search** is a **search**/decision-making **algorithm** that maximizes the average (expected) reward. It is typically applied to trees that have stochastic models

- In addition to "min" and "max" nodes of the traditional minimax tree, this variant has "chance" nodes, which take the expected value of a random event occurring. In game theory terms, an expectiminimax tree is the game tree of an extensive-form game of perfect, but incomplete information.

**> Alpha-Beta Pruning**

Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It is an adversarial search algorithm used commonly for machine playing of two-player games.

- Alpha-Beta pruning is not actually a new algorithm, rather an optimization technique for minimax algorithm.

- It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available.

- It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely alpha and beta.

**> Q-Learning**

- Q-learning is a reinforcement learning technique used in machine learning. The goal of Q-learning is to learn a policy, which tells an agent what action to take under what circumstances.

Q-learning is a values-based learning algorithm in reinforcement learning

- It does not require a model of the environment and can handle problems with stochastic transitions and rewards, without requiring adaptations.

**> Iterative Deepening**

- Iterative deepening depth first search is a hybrid of BFS and DFS. In IDDFS, we perform DFS up to a certain “limited depth,” and keep increasing this “limited depth” after every iteration.

IDDFS (iterative-deepening depth-first search) is simply a depth-first search performed multiple times, deepening the level of nodes searched at each iteration. Therefore, the memory requirements are the same as depth-first search because the maximum depth iteration is just the full depth-first search.

- It is a state space/graph search strategy in which a depth-limited version of depth-first search is run repeatedly with increasing depth limits until the goal is found.

- IDDFS is equivalent to breadth-first search, but uses much less memory; at each iteration, it visits the nodes in the search tree in the same order as depth-first search, but the cumulative order in which nodes are first visited is effectively breadth-first.

**> State Space Graphs**

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

For every search problem, thereʼs a corresponding state space graph ! The successor function is represented by arcs

- Problems are often modelled as a state space, a set of states that a problem can be in. The set of states forms a graph where two states are connected if there is an operation that can be performed to transform the first state into the second.

* **Greedy Search**

- A greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the intent of finding a global optimum.

It **always makes the choice that seems to be the best at that moment**. This means that it makes a locally-optimal choice in the hope that this choice will lead to a globally-optimal solution.

- In many problems, a greedy strategy does not usually produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time.

\* **Markov Decision Processes**

- MDPs are meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The agent and the environment interact continually, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent.

- Formally, an MDP is used to describe an environment for reinforcement learning, where the environment is fully observable. Almost all RL problems can be formalized as MDPs.

\* **Reinforcement Learning**

- Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

- The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms.