BFS/DFS:
BFS and DFS are using different data structure to achieve difference result. BFS using first in first out to browse the same level node first then to the next level. DFS using last in first out queue to browse one branch as deep as possible then browse next branch.

A* search:
A star search can be used to find the shortest path on a graph or map. It can use exact heuristic or approximate heuristic. Manhattan distance can be an approximate heuristic.

Uniform cost search:
Uniform cost search can give the minimum cost to the destination. It solves like BFS/DFS but with a cost argument. Every time pick next node, always pick the lowest available cost node.

Suboptimal Search:
Suboptimal Search be used when the optimal heuristic is too expensive to do.

Minimax:
Minimax can use on decision making. Assume your opponents move always the optimal move for them, pre-calculate the result base on their move then make the optimal move on that scenario.

Expectimax:
May be your opponent not always make the best move. Base on the reward of different move and combine their possibility to happen then pick the highest expected value move.

Markov decision process:
Markov decision process are used to solving decision making problem. The history of how to get to the current state doesn't matter in MDP, only the future matters. It will return a policy base on the possibility and reward of all possible result on that state.

Q-Learning:
Q learning is a reinforcement learning method to make the optimal policy. Keep training the agent and keep update the q value. Agent will learn to make better policy as the agent play more episode.

Epsilon Greedy:
The goal of epsilon greedy algorithm is to pick the best expected option. Keep tracking each option then update the average reward of each one. Use a random number to determine if pick the current highest expected value option or to explore the others.

Approximate Q-Learning:
Instead of update the action with best reward, update the difference between current policy and the better policy.

$$Q(s, a) \mathrel{+}= \alpha[R(s) + \gamma Q(s', a') - Q(s, a)]$$

Linear Random search:
Use to solve problem not to complicate. Keep guessing until find the right move and update that episode. Would be faster to solve simple problems but not complicate big size problem.

Cross-entropy method:
Cross-entropy method can be applying to continuous and combinatorial problems. Keep training the agent and update the good run throw out the bad run. Minimize the difference between optimal reward and current reward.