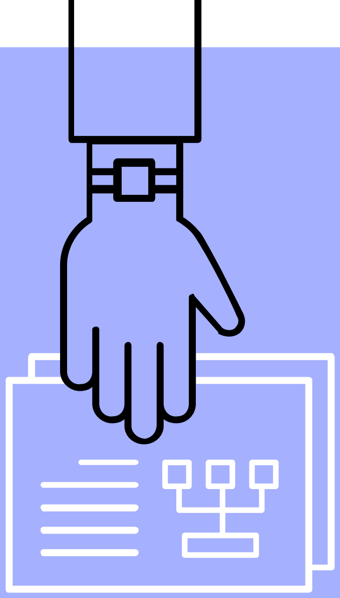
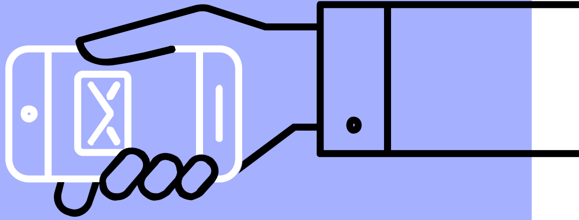
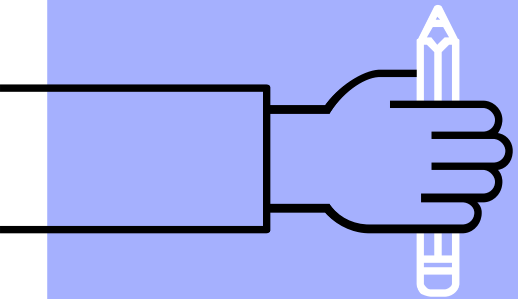
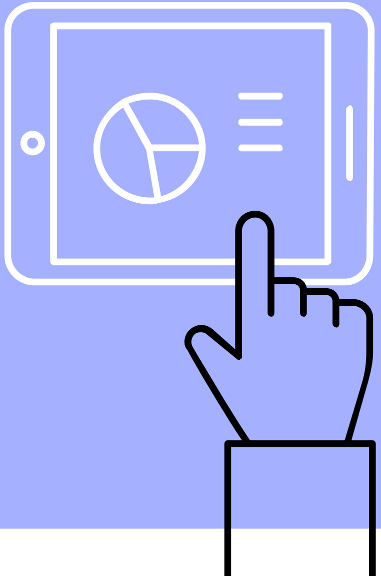


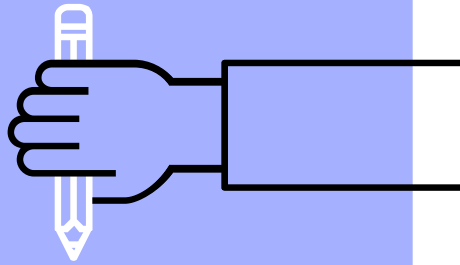


Advanced Artificial Intelligence

Lecture 1 (AI Basics)

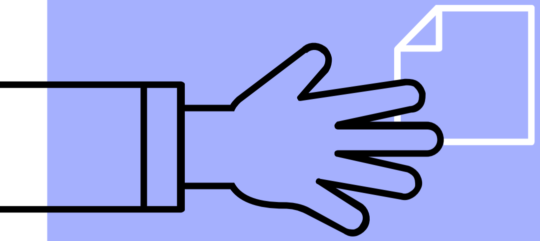




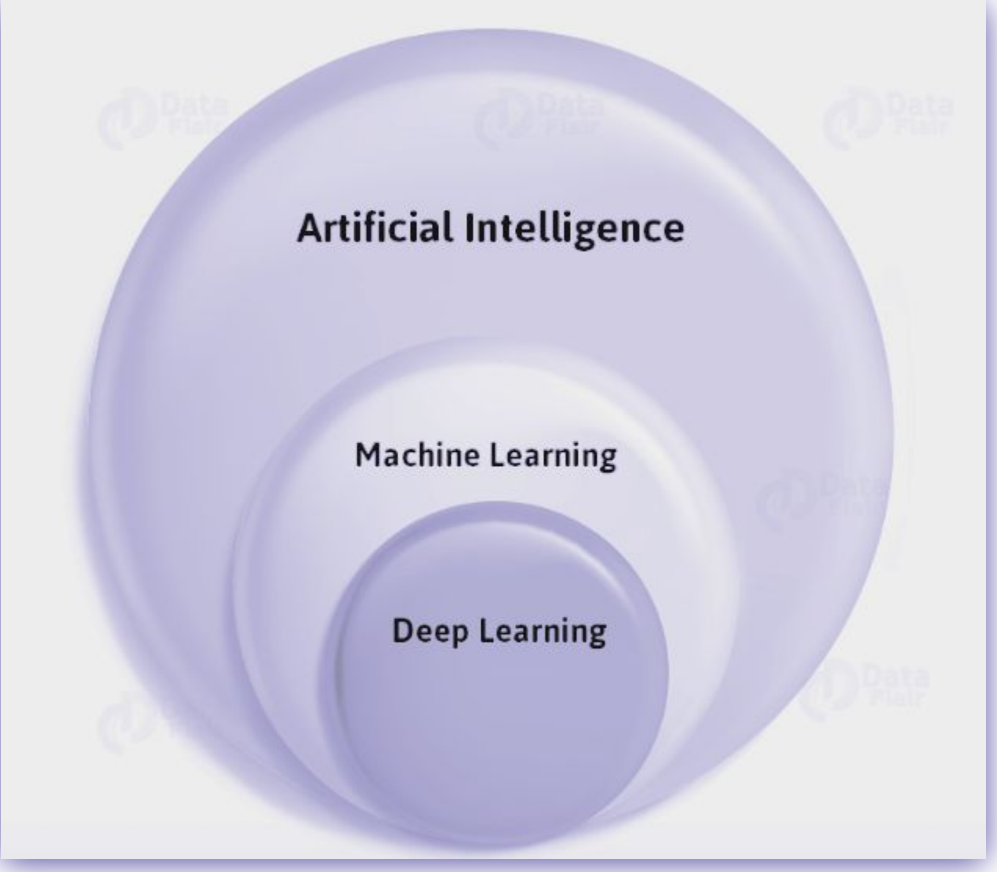


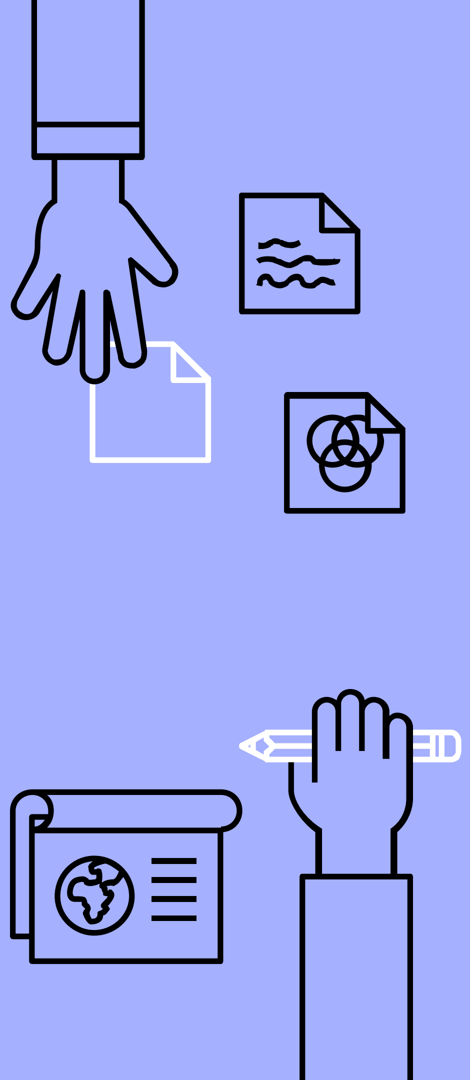
INTRODUCTION

What is Artificial Intelligence?



INTRODUCTION



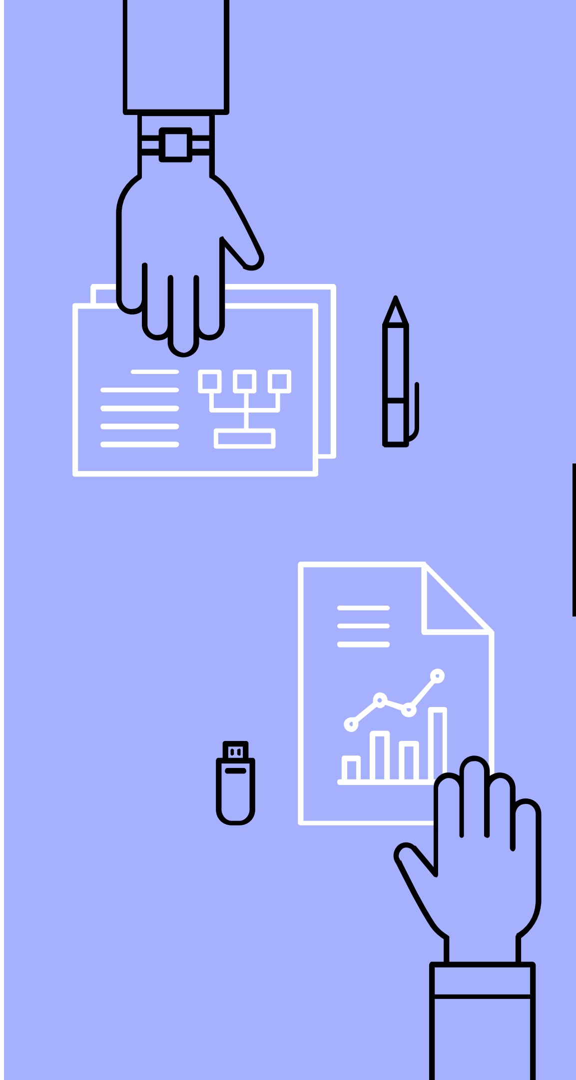


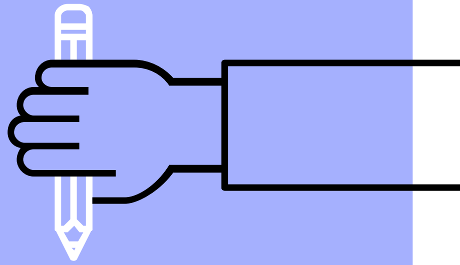
APPLICATIONS

[redacted]

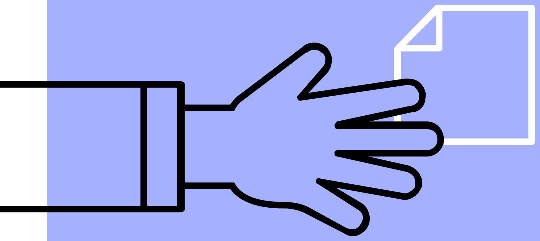
Applications

- Face Detection or Character Recognition
- Translation of spoken word to Text
- Intelligent Alarming in Healthcare
- Basket Analysis
- Loan Repayment Risk Classification
- Predicting Sales for Next New Year
- Extracting information from news articles

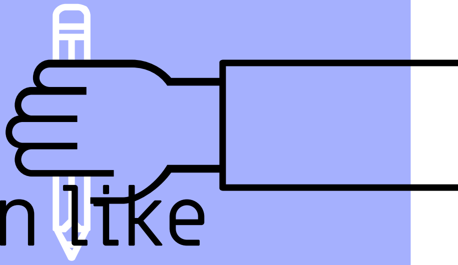




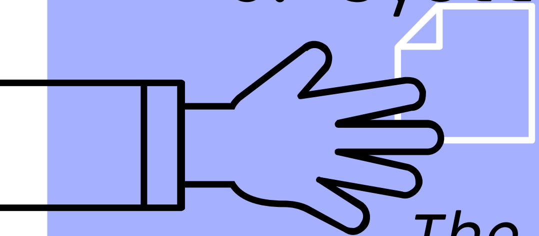
Introduction and Scope



Goals of An AI System



1. Systems that think and reason like humans
2. Systems that can think rationally.
3. Systems that act like humans



What is Rationality??

The ability to differ right from wrong

Turing Test

A human judge engages in natural language with two other parties, one human and the other, a machine.

- If the judge cannot reasonably tell which is which, then the machine is said to pass the turing test.

- The Loebner Prize is the oldest Turing Test contest, started in 1991

Problem Solving

- Whenever an AI system is at some current state and does not know how to proceed in order to reach the desired goal.

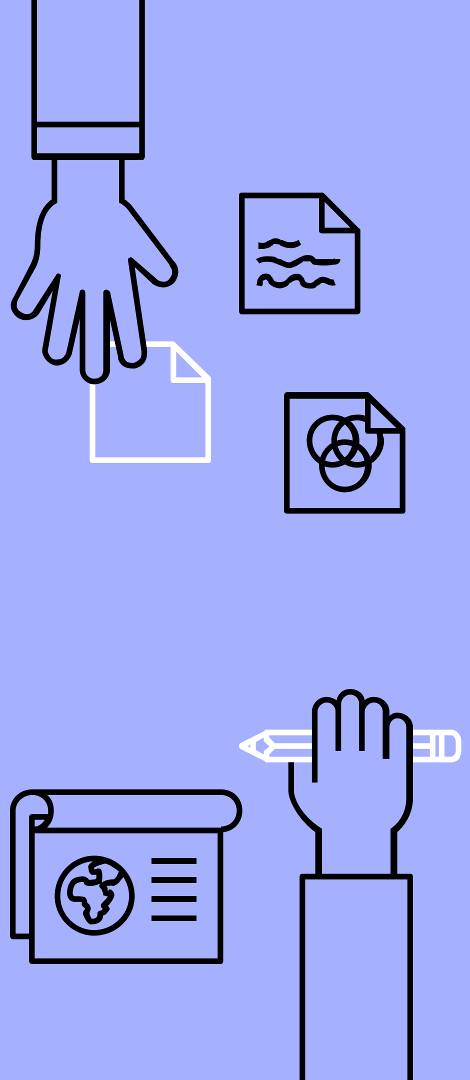
- This is considered to be a problem that can be solved by coming up with a series of actions from the state space* that lead to the goal state

*The **search space** is the set of all possible solutions.

[redacted]

Knowledge Representation

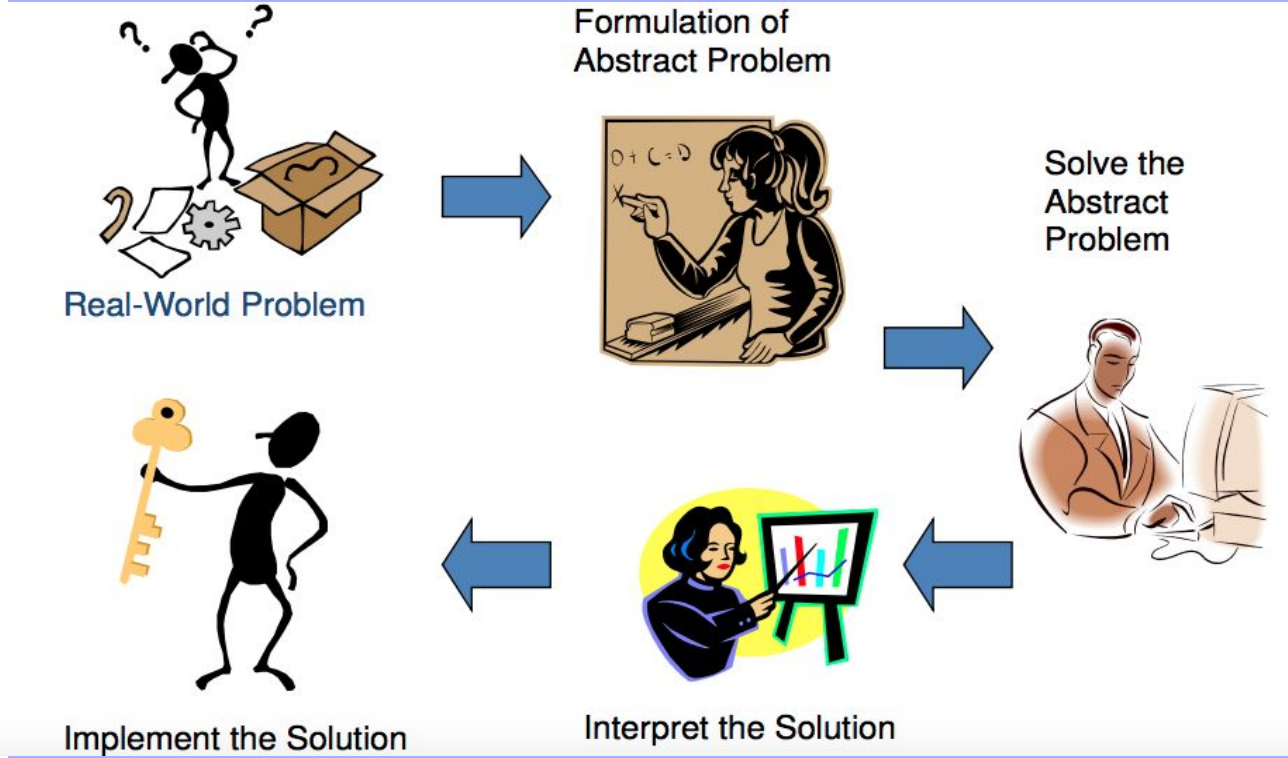
- Represent the problem in a language with which a computer can reason.
- Use the computer to compute an output, which is an answer presented to a user or a sequence of actions to be carried out in the environment.
- Interpret the output as a solution to the problem.



Problem Representation

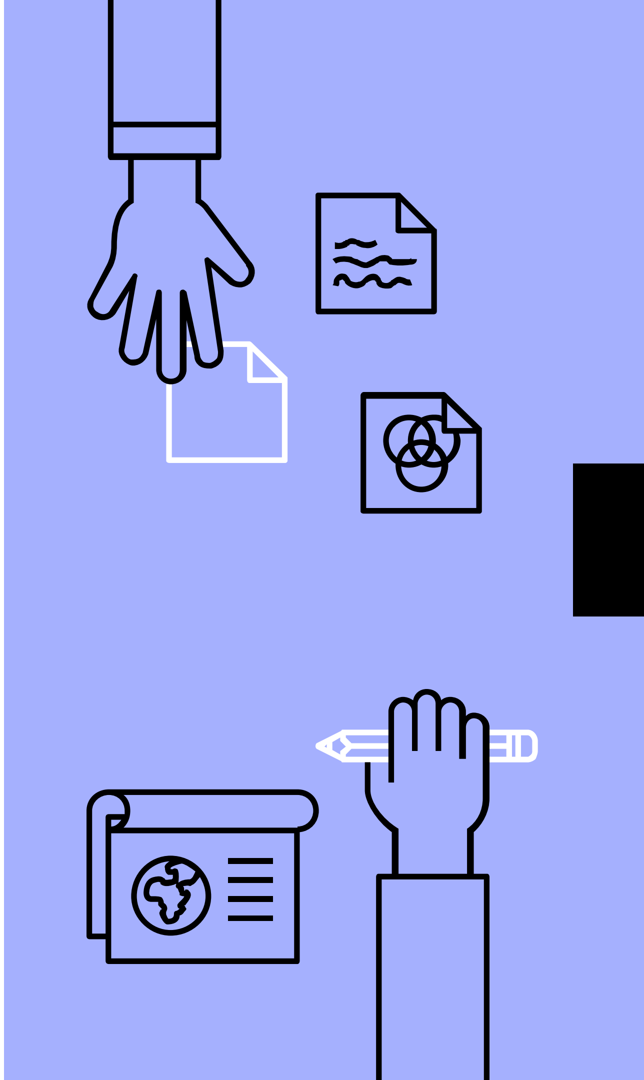
- Able to be acquired from people, data or past experience

- Able to be solved computationally



Search Techniques in AI

- ➔ Uninformed/Blind Search
- ➔ Informed Search
- ➔ Adversarial Search



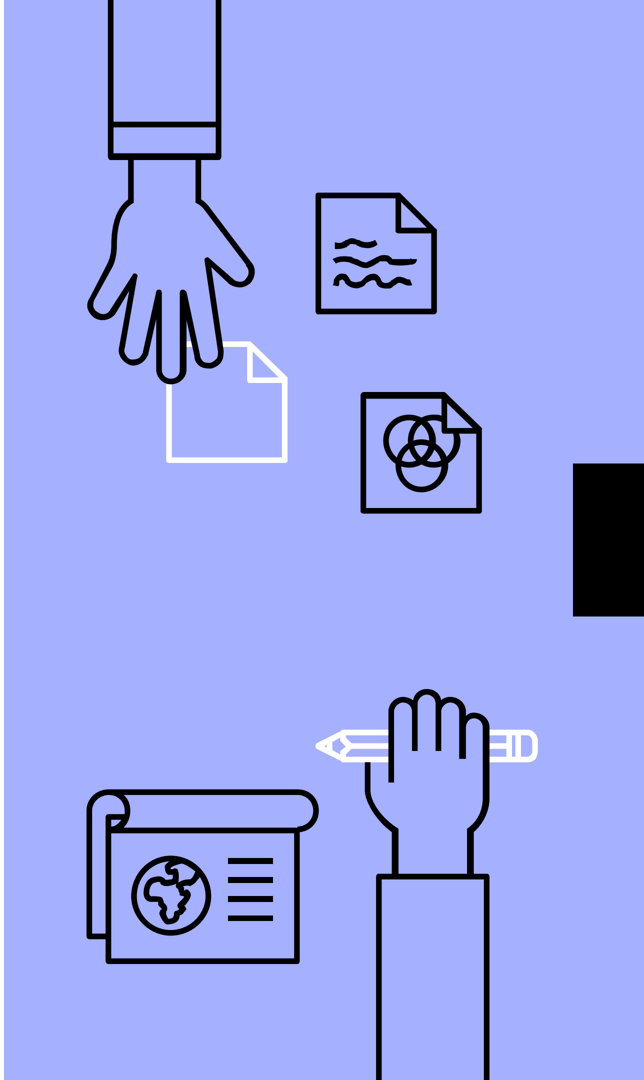
Uninformed Search Strategy

➔ Uninformed search has no information about the number of steps or the path costs from current state to goal.

➔ These algorithms ignore where they are going until they find a goal and report success.

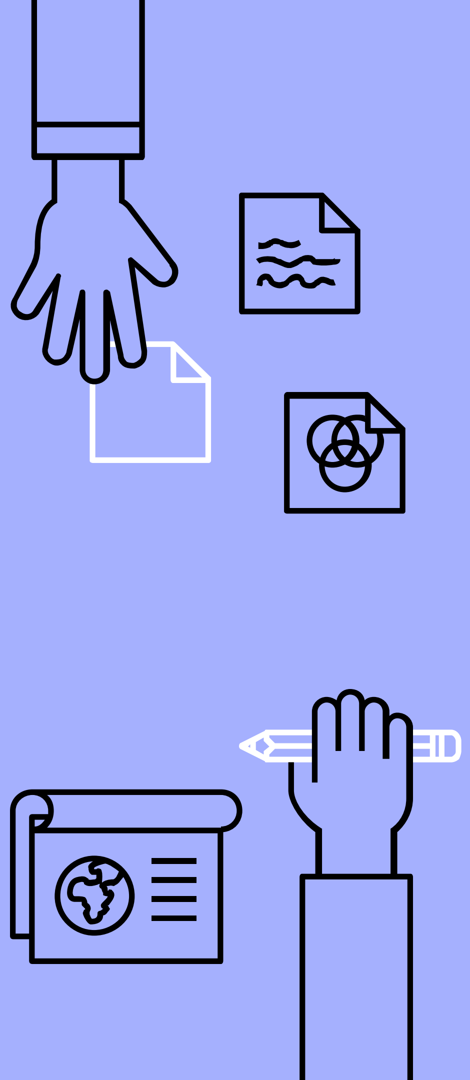
➔ They can only distinguish a goal state and a non goal state.

➔ No additional information about the state is present to prefer one child over other.



Uninformed Search Algorithms

- Breadth First Search (BFS)
- Depth First Search (DFS)
- Bidirectional Search
- Depth Limited Search (Iterative Deepening)

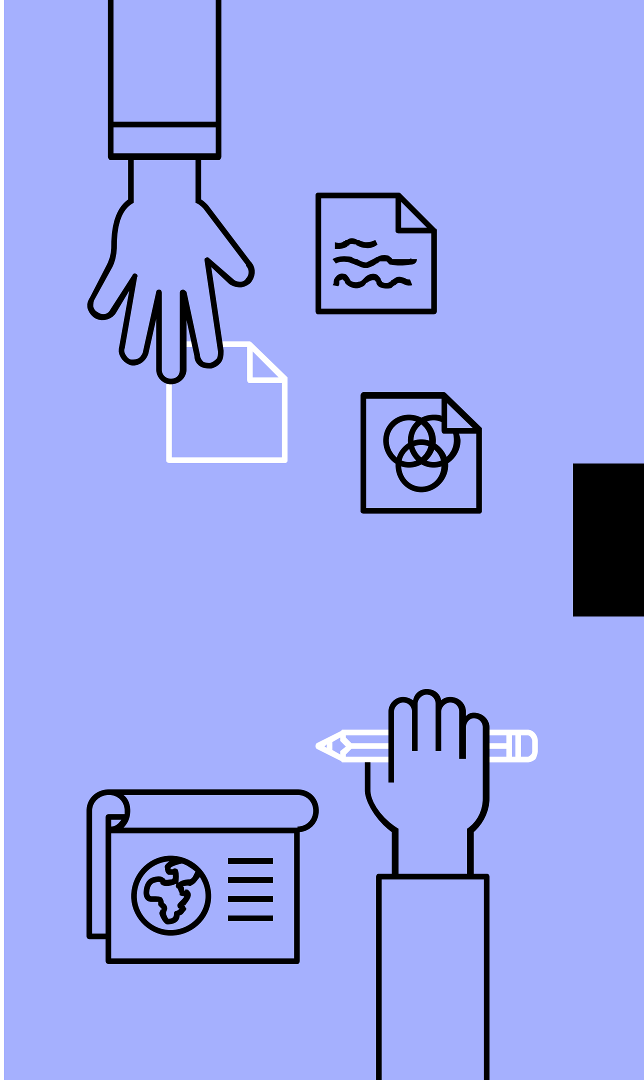


BFS

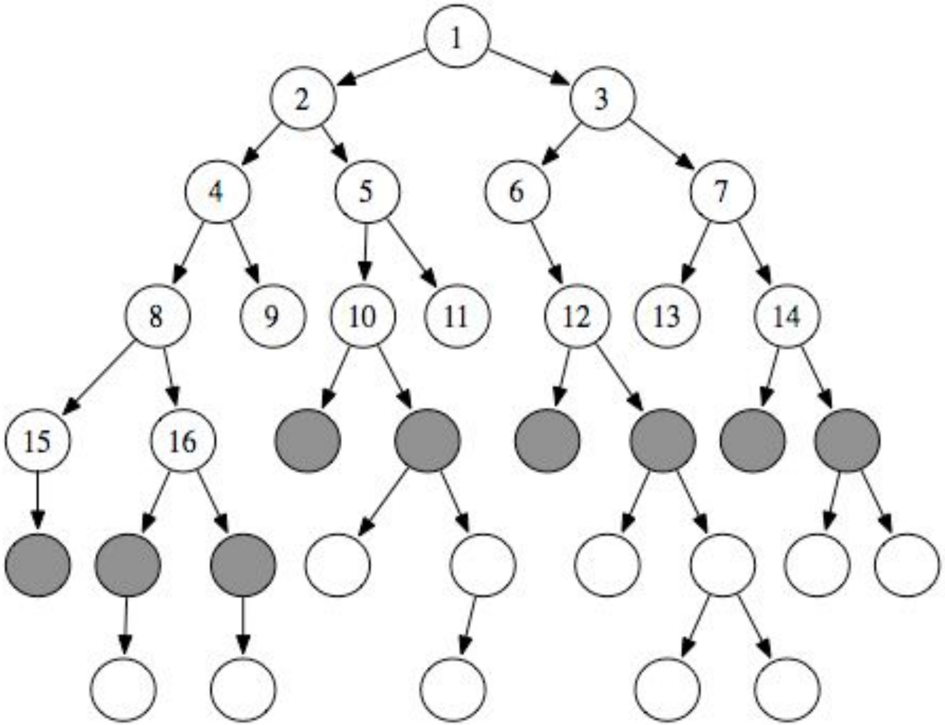
➔ Starts with the root node and explores all neighboring nodes and repeats, expanding the "depth" of the search tree by one in each iteration.

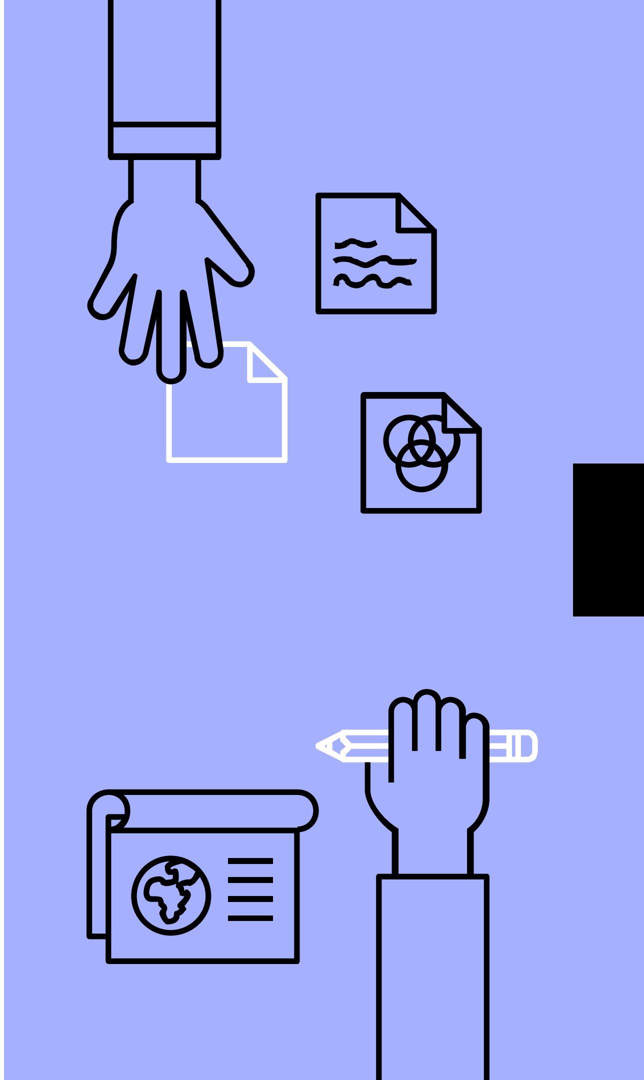
➔ Implemented in FIFO queue.

➔ BFS is optimal. If it finds the node it will be shallowest in the tree.



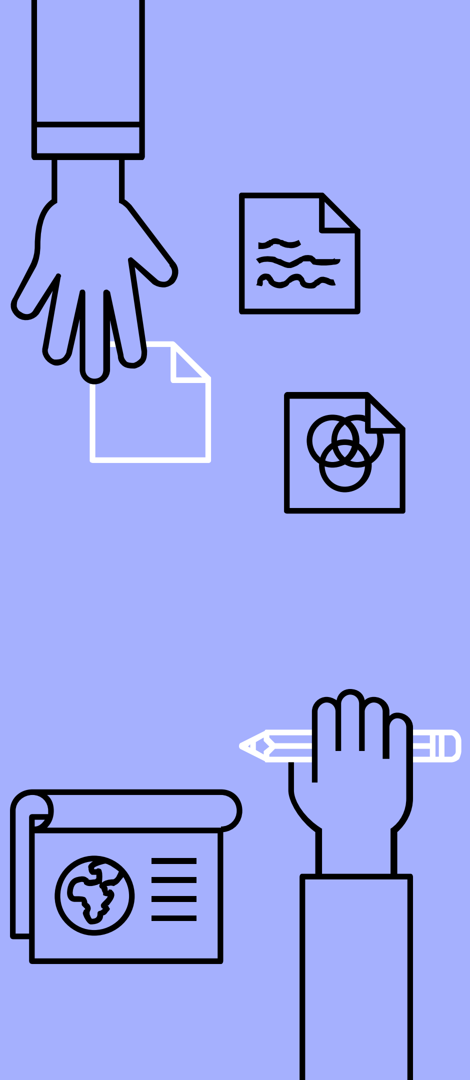
BFS Order of Traversal





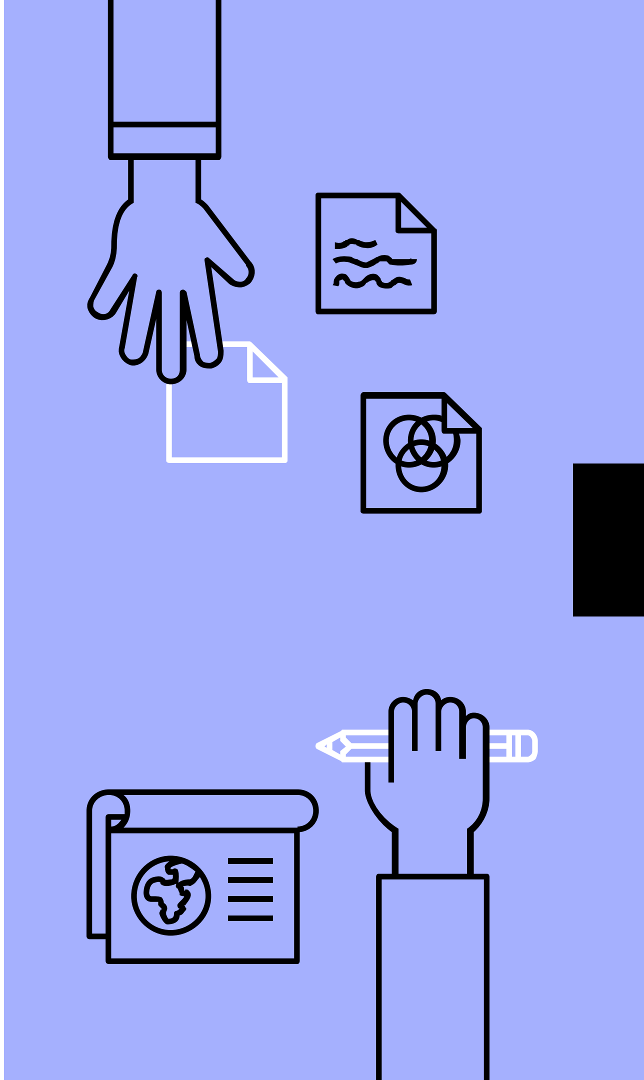
Advantages

- ➔ If any solution exists, BFS guarantees to find it.
- ➔ If there are many solutions, BFS will always find the shortest path solution.



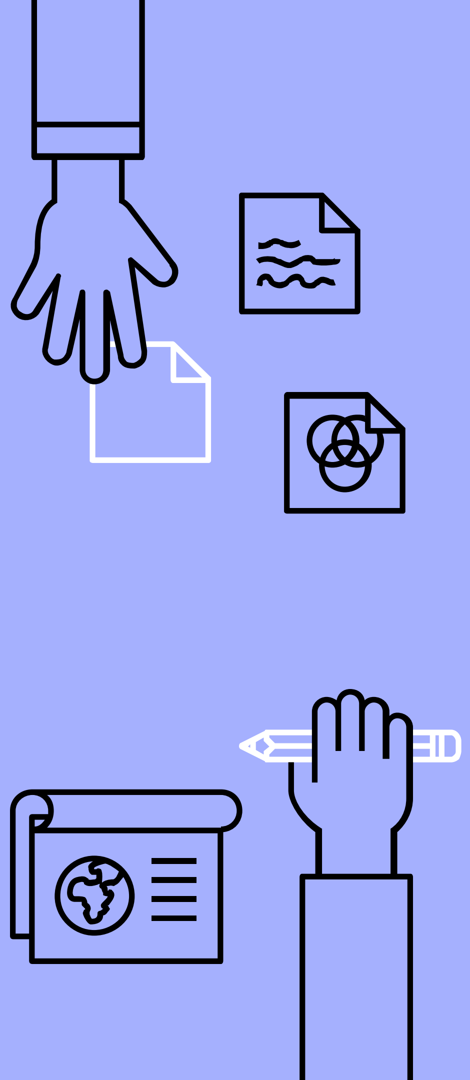
Disadvantages

➔ Time and Space Complexity is exponential.

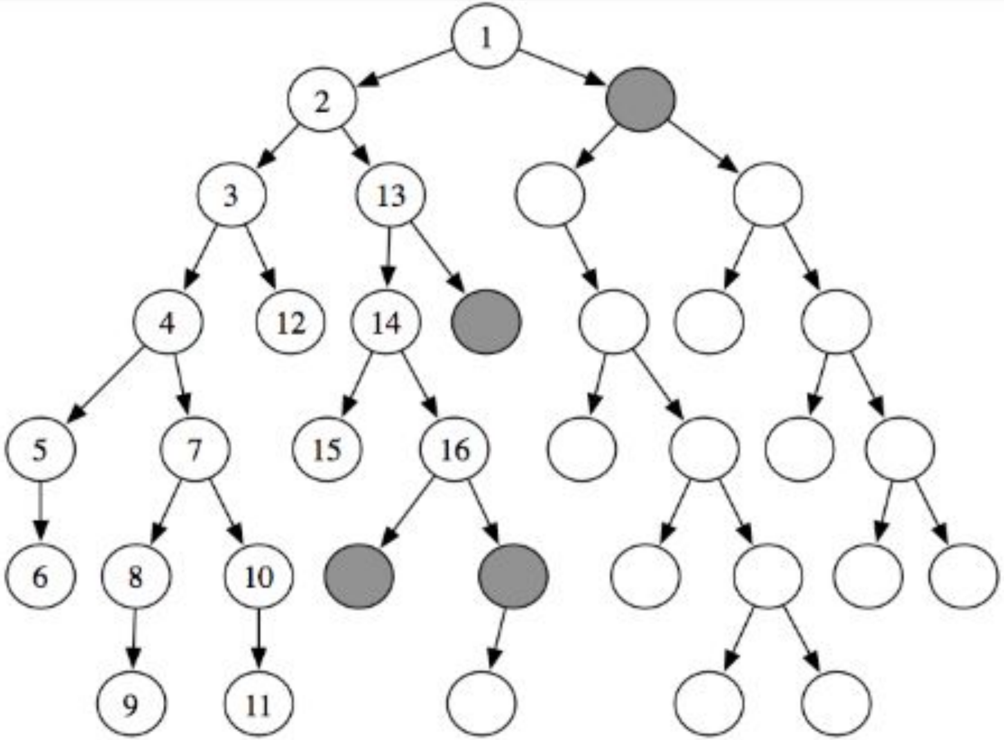


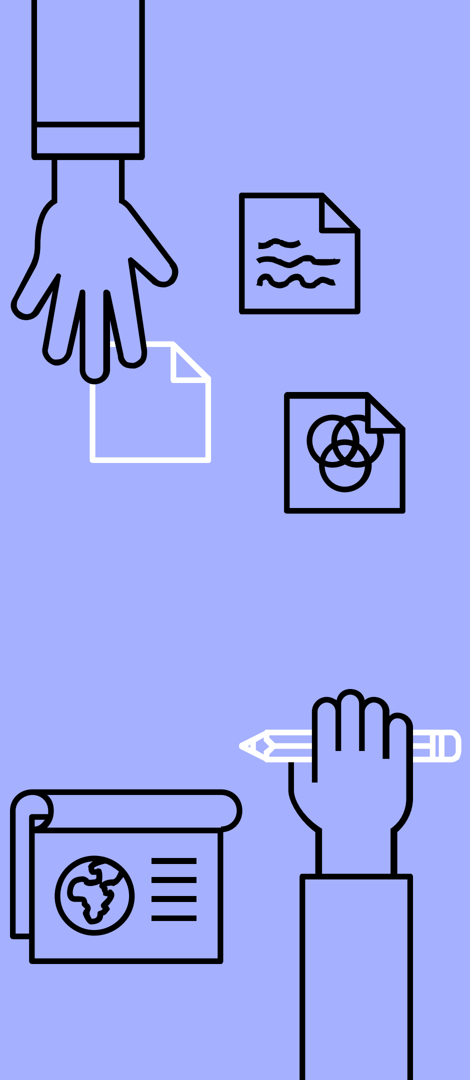
DFS

- ➔ Explores one path to the deepest level and then backtracks until it finds a goal state.
- ➔ Depth First Search is not optimal.
- ➔ It stops at the first goal state it finds, no matter if there is another goal state that is shallower than that.



DFS Order of Traversal

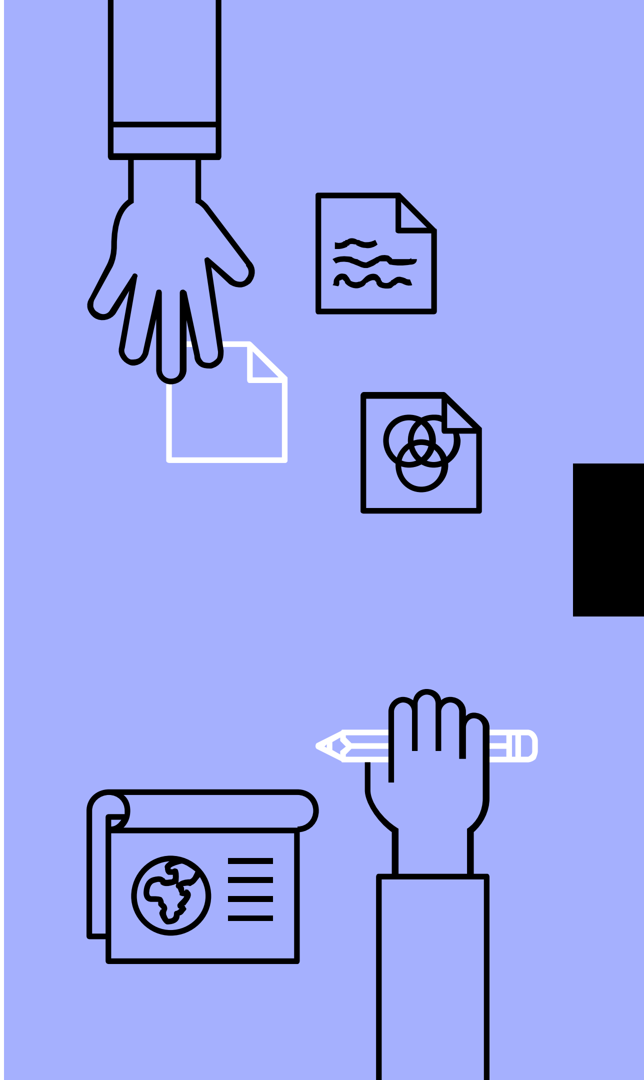




Advantages

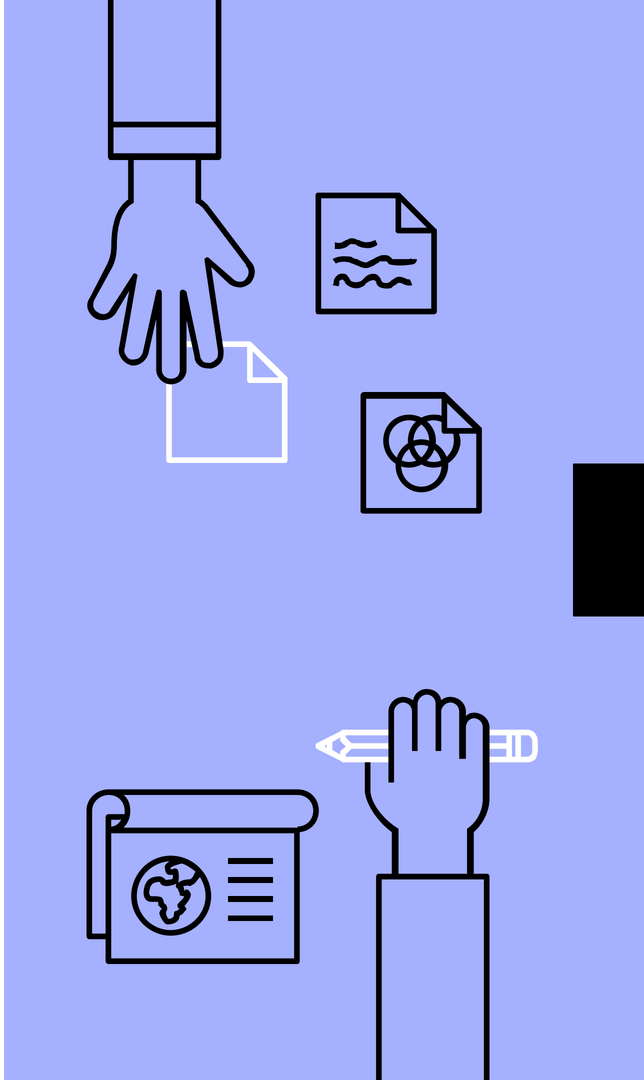
- ➔ Memory requirements in DFS are less compared to BFS as only nodes on the current path is stored instead of the entire level.

- ➔ DFS can find a solution without much of the search space.



Disadvantages

- Prone to blind alley
    - The search can go deeper and deeper into the search space and thus can get lost.

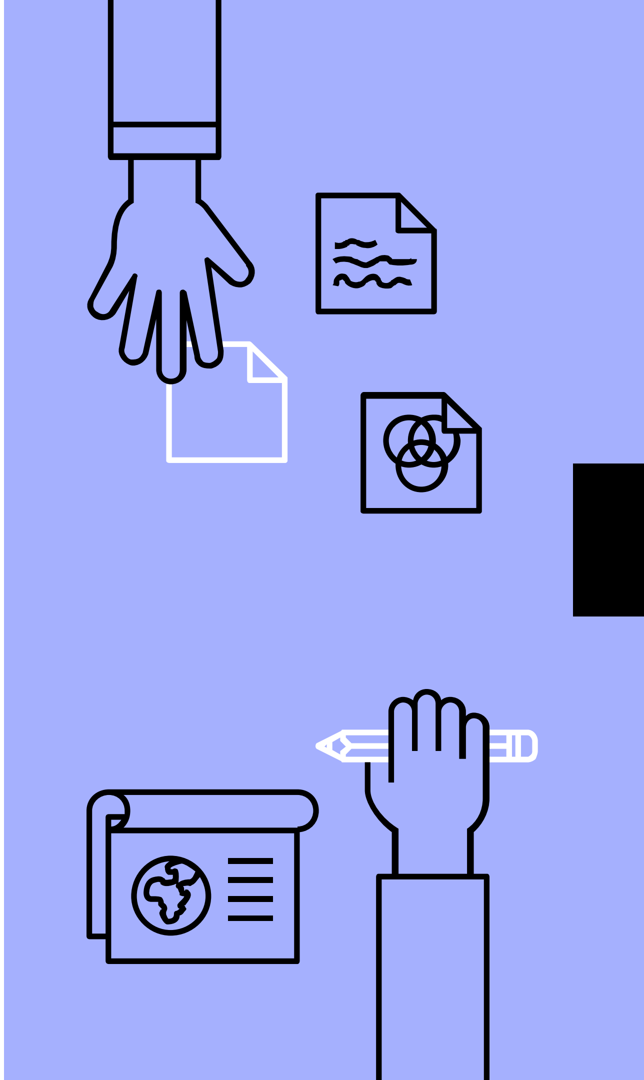


Bidirectional Search

➔ Search begins from initial state in forward direction and backwards from goal state.

➔ Till it meets in the middle to identify a common state.

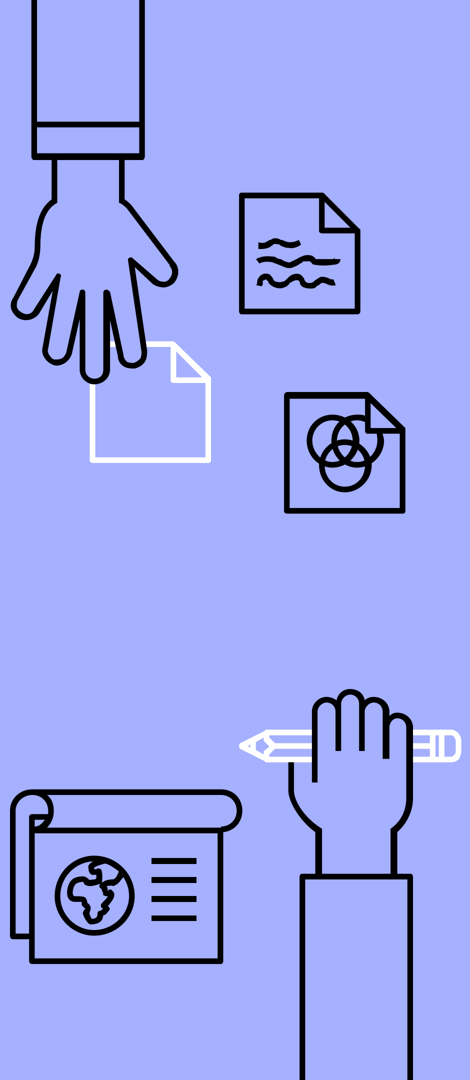
➔ It is optimal



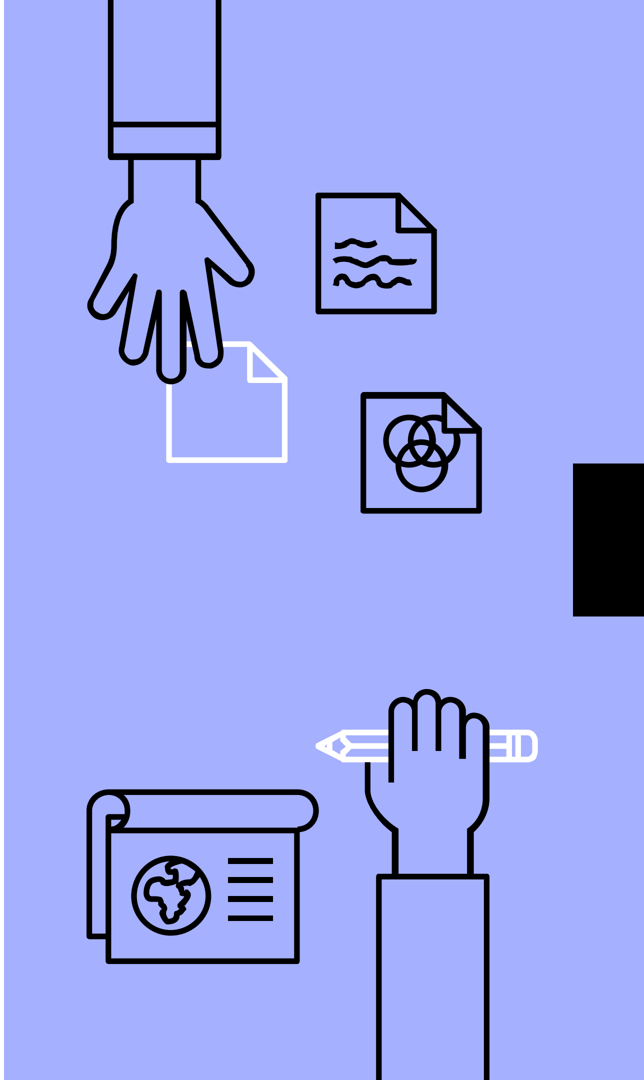
Iterative Deepening

➔ The search depth for DFS is either limited to a constant value or increased iteratively over time.

➔ It is optimal



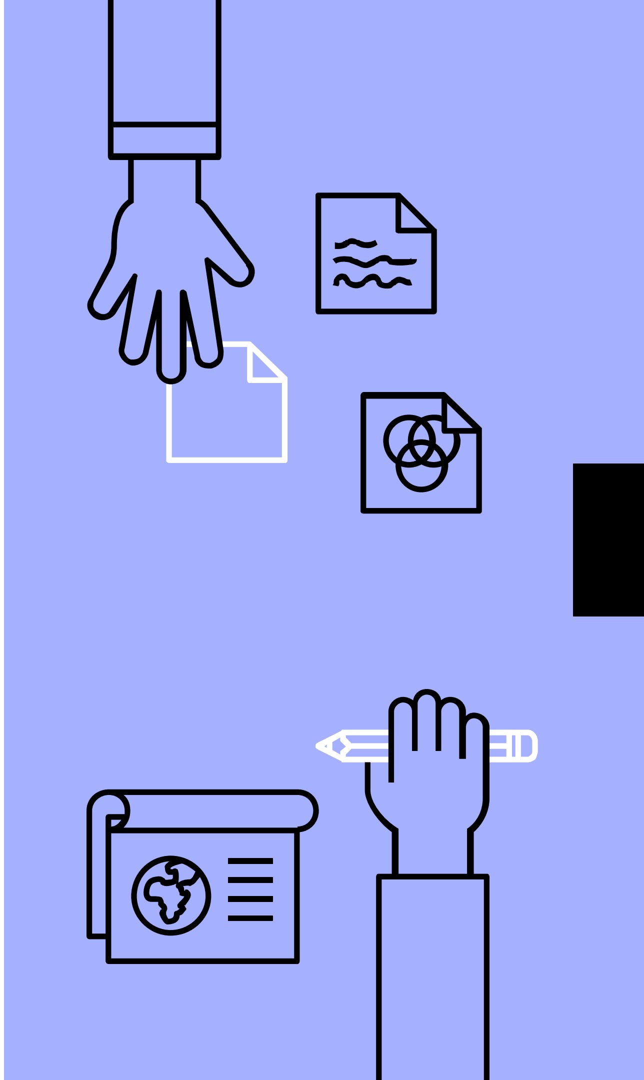
Informed Heuristic Search Strategies



- Instead of exploring the search tree blindly, one node at a time, the nodes that we could go to are ordered according to some evaluation function.

- This function uses a heuristic as a guide to decide which node is best to go next and will lead to the best overall performance.

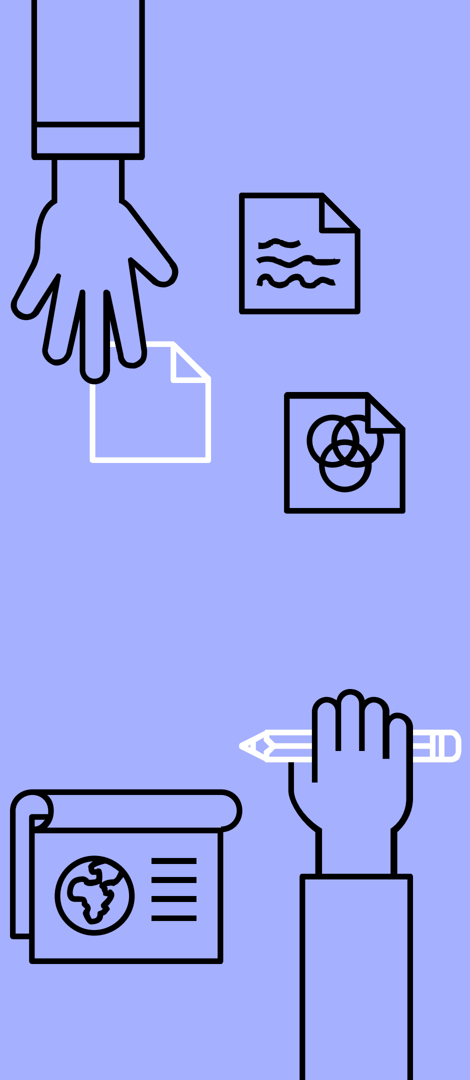
Informed Search Algorithms



- ➔ Greedy Best First Search
- ➔ A* Search

Greedy Best First Search

- Minimizes the estimated cost to reach the goal.
- The node that is closest to the goal is always expanded first.
- Every node has a heuristic function attached to it.
- Decision of which node to be expanded depends upon the value of evaluation function.



A* Search

➔ Evaluation function is combination greedy search and uniform cost function.

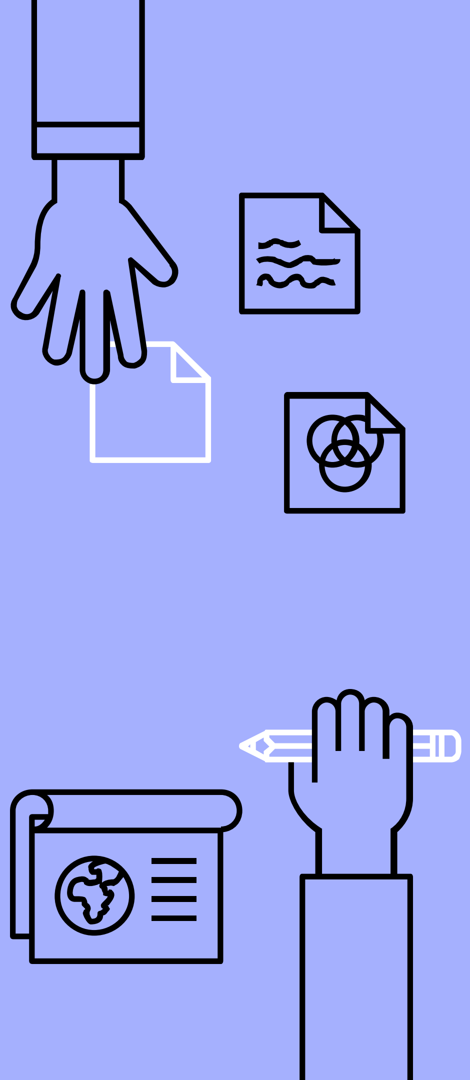
➔ A* takes the evaluation function as

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

Where,

$g(n)$ = cost(distance) of the current node from start state.

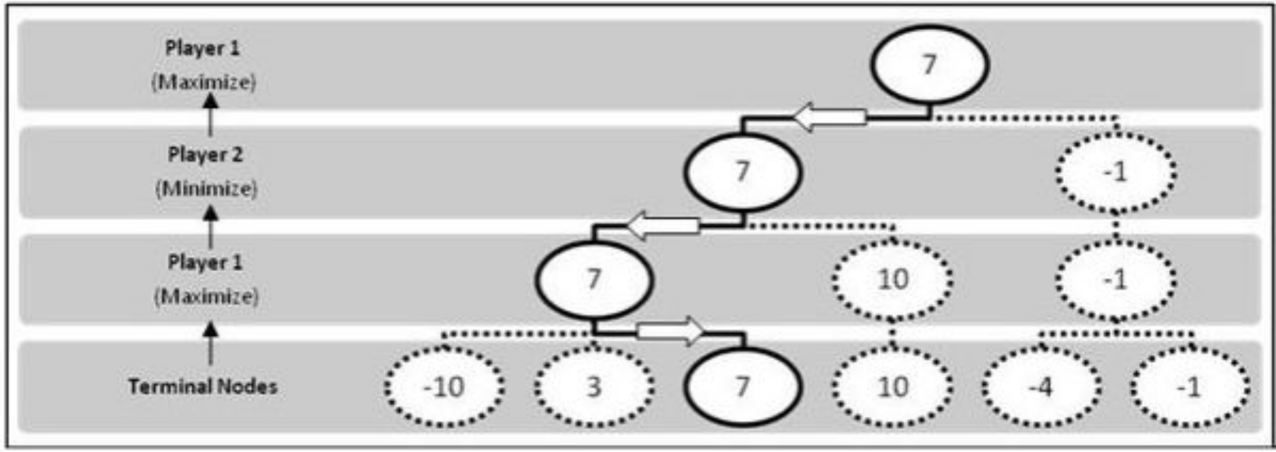
$h(n)$ = estimated cost of current node from goal node.



Adversarial Search

- You change state but do not control next state (Chess)

- Opponent will change state unpredictably (MinMax)



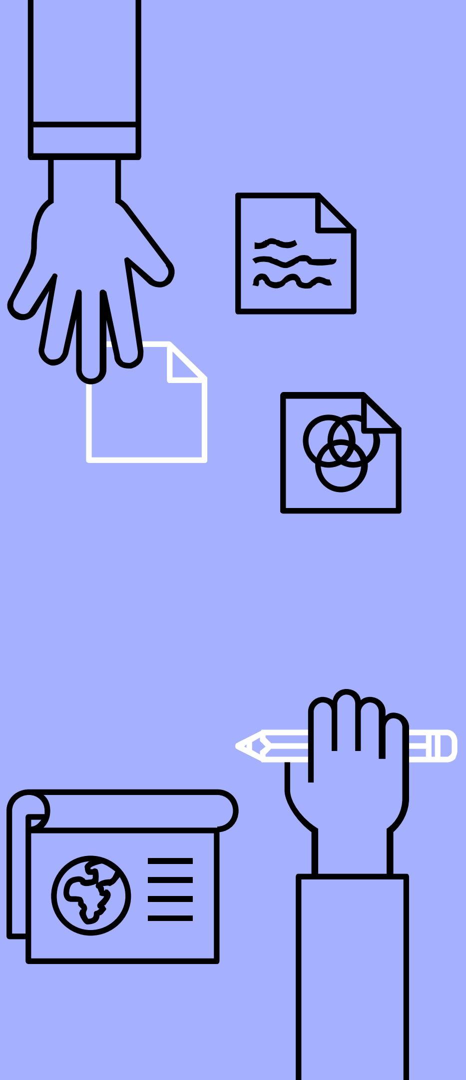
Minimax : 2 Player Games

1. Max tries to maximize its score

2. Min tries to minimize Max's score

3. *Goal :*
   a. Max to Move to position with highest minimax value

   b. Identify best achievable payoff against best play.

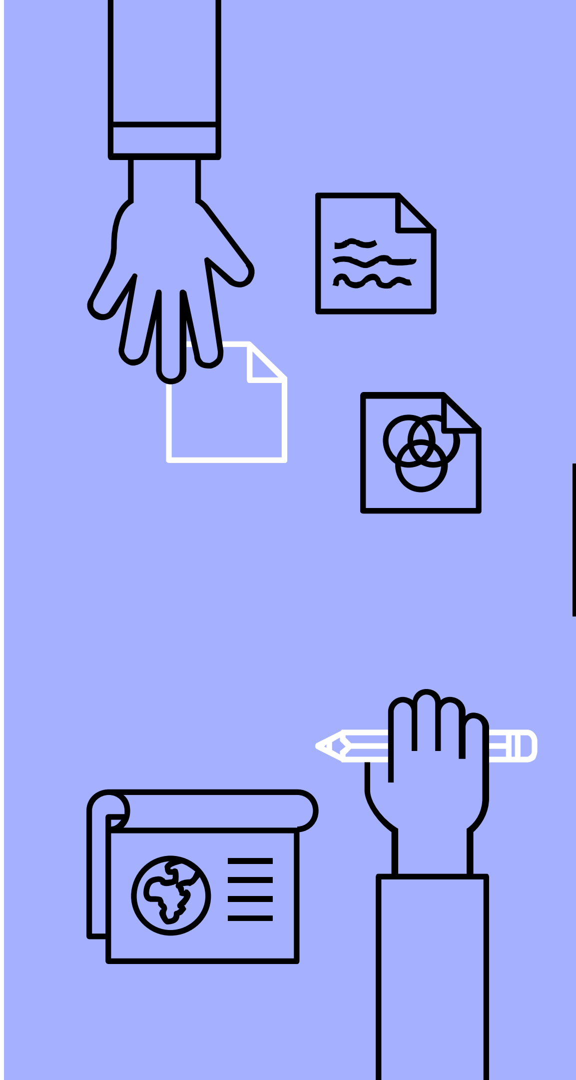


Minimax Algorithm

1. Two player Zero Sum Game.

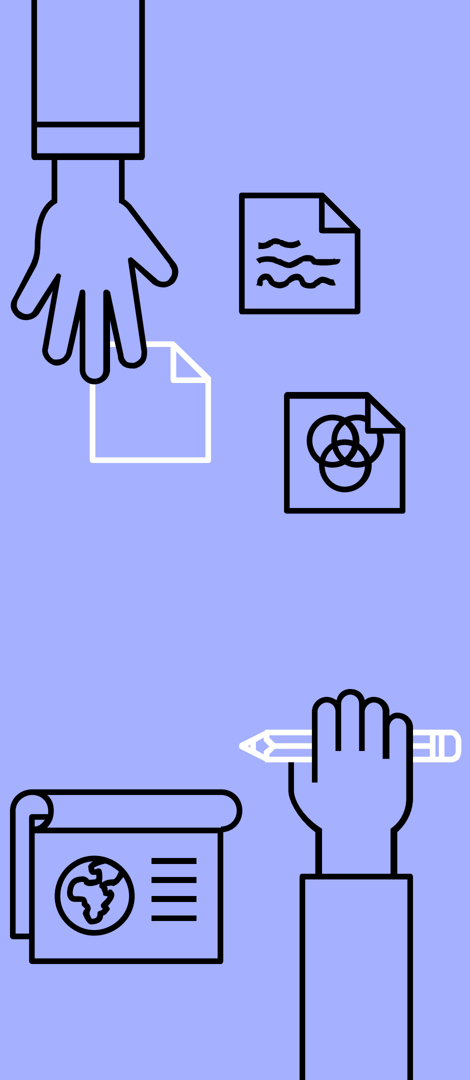
2. Both players make a move alternatively.

3. It is defined by an initial state
   i. Board positions
   ii. A set of legal operations
   iii. A terminal test to decide the end of game.
   iv. A utility function that determines the outcome of game e.g., win(1)/loss(-1)



Why is Game Playing A Challenge for AI

1. Competent Game Playing is a mark of some aspects of intelligence.
2. Requires planning, reasoning and learning.
3. Proxy for Real World Decision Making Problems.
4. Easy to Represent States and Define Rules.
5. Obtaining good performance is hard.



Traditional Board Games

Finite

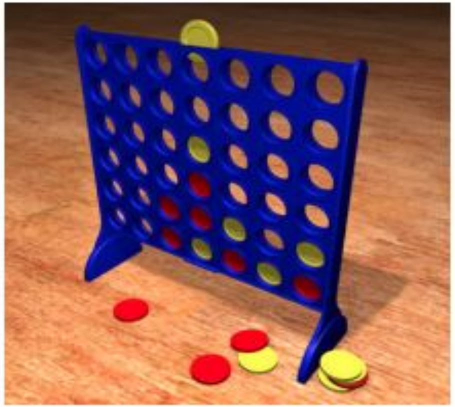
Two-player

Zero-sum

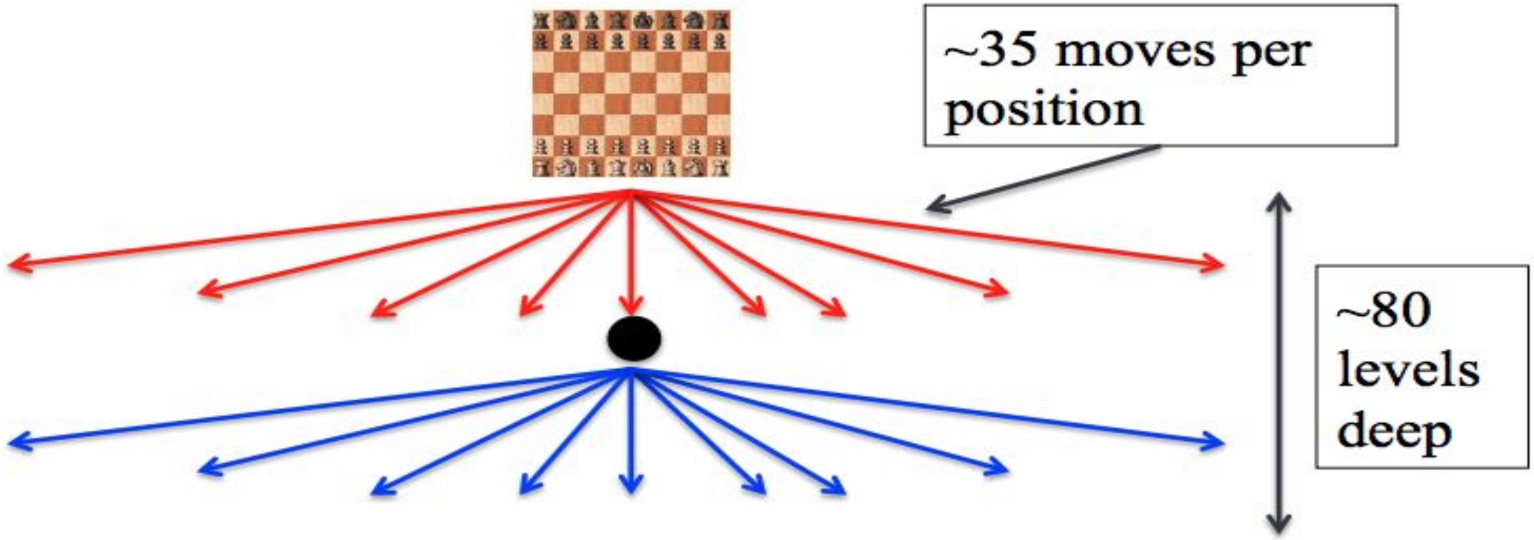
Deterministic

Perfect Information

Sequential



How big is this tree?



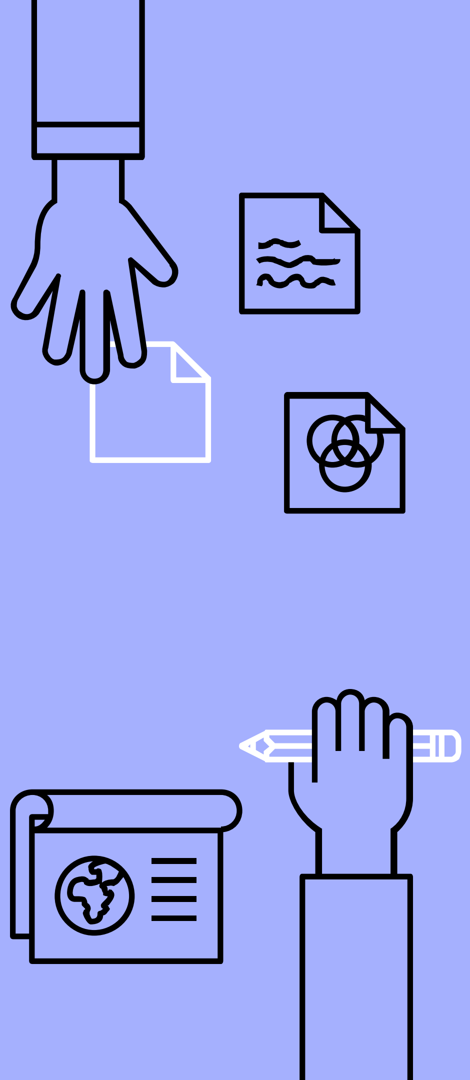
Approx. 10^120 > Number of atoms in the observable universe (10^80)

We can really only search a **tiny, miniscule faction** of this tree!

Around 60 x 10^9 nodes for 5 minute move. **Approx. 1 / 10^70 fraction.**

Formal Definition of A Game

1. **Initial State**
2. **Successor Function**
    a. *Returns List of (move,state) pairs*
3. **Terminal Test :** *Determines when Game is Over*
4. **Terminal States :** *States where game Ends*
5. **Utility Function (objective function or payoff function) :** *Gives numerical value to terminal States*

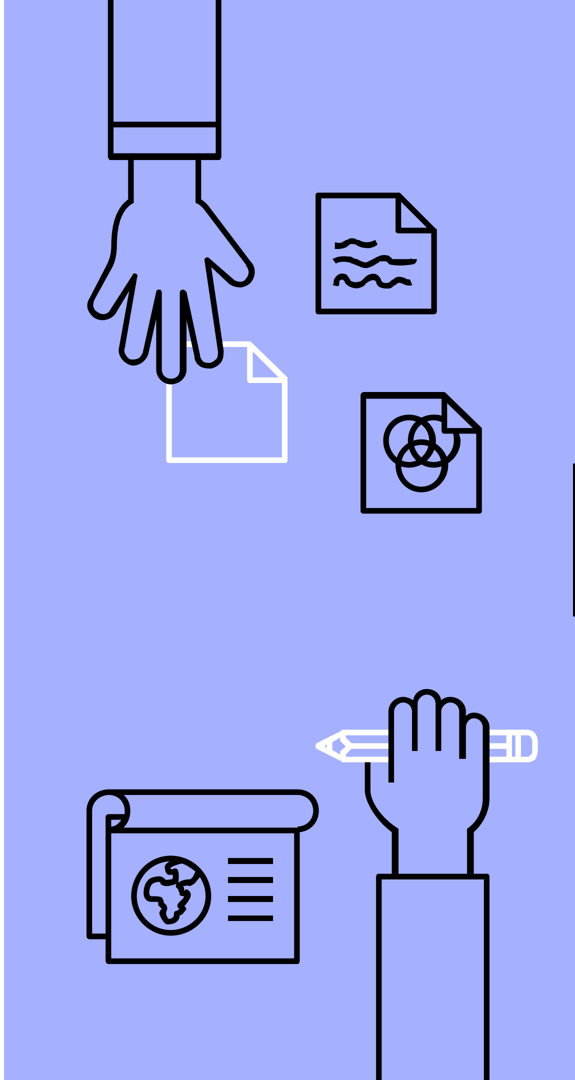


Minimax Algorithm : Steps

1. The opponent Min tries to minimize Max's outcome.

2. The minimax algorithm generates the whole game tree and applies the utility function to each terminal state.

3. For games that are too complex, to compute the whole game tree, the game tree is cut off at some point and the utility value is estimated by a heuristic function.



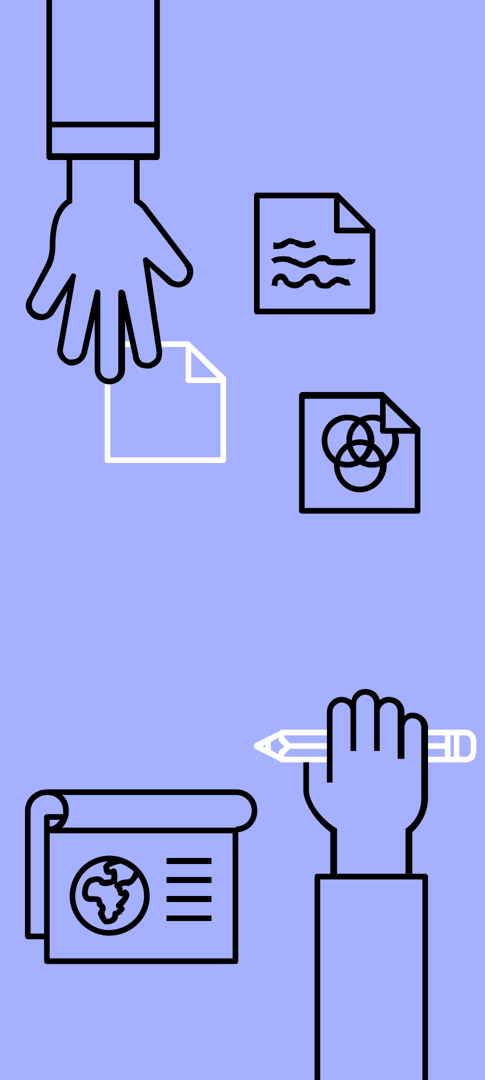
Steps

1. The opponent Min tries to minimize Max's outcome.

2. Min is assumed to always choose the option that is worst for Max (minimum utility value).

3. If one has three terminal states in one branch with 1,2, and 3 as their utility values, then min would choose 1.

4. Minimax Decision = Maximizing Utility under the assumption that the opponent will play perfectly to minimize it.



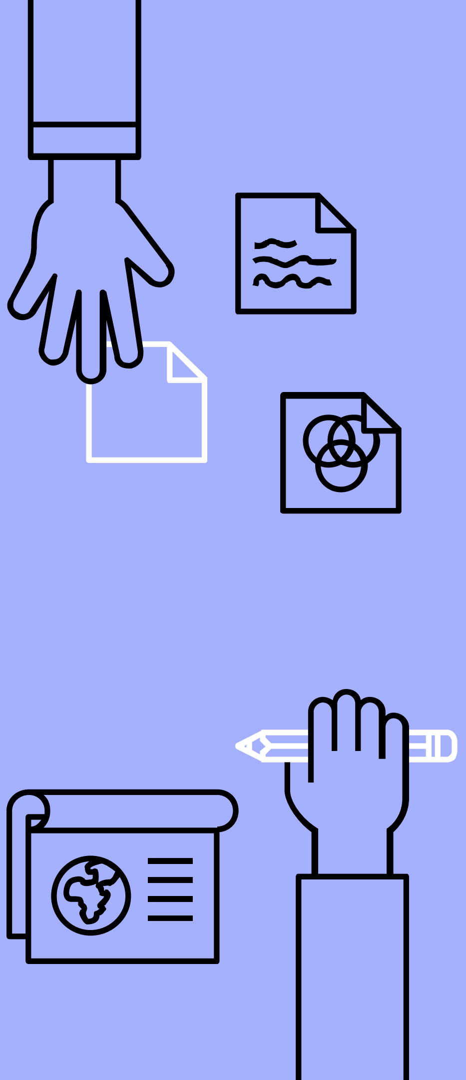
Hill Climbing Search

- An iterative algorithm that starts with an arbitrary solution to a problem and attempts to find a better solution by changing a single element of the solution incrementally.

- If the change produces a better solution, an incremental change is taken as a new solution.

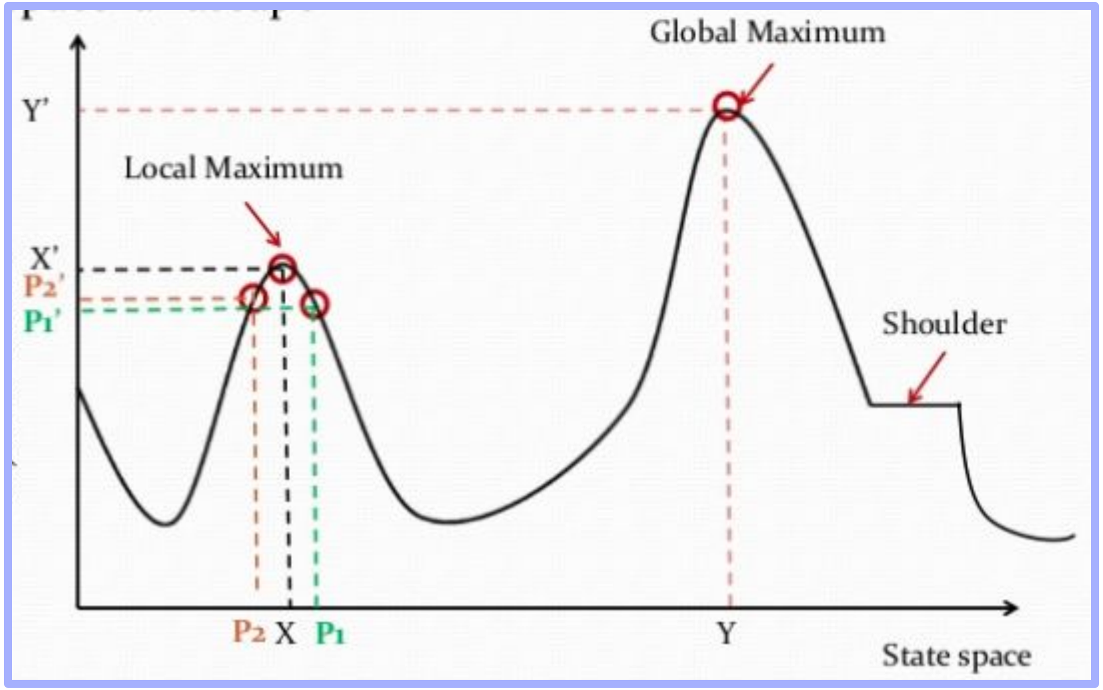
- This process is repeated until there are no further improvements.

- Hill Climbing approach returns local maxima.



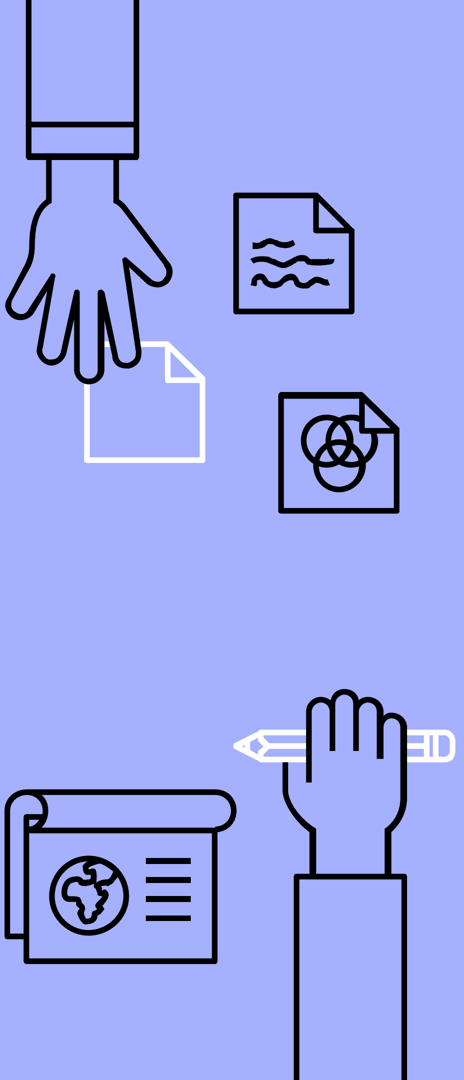
Hill Climbing Search

Maximization Function ⇒ Objective Function
Objective Function ⇒ Profit, Success



Hill Climbing Search

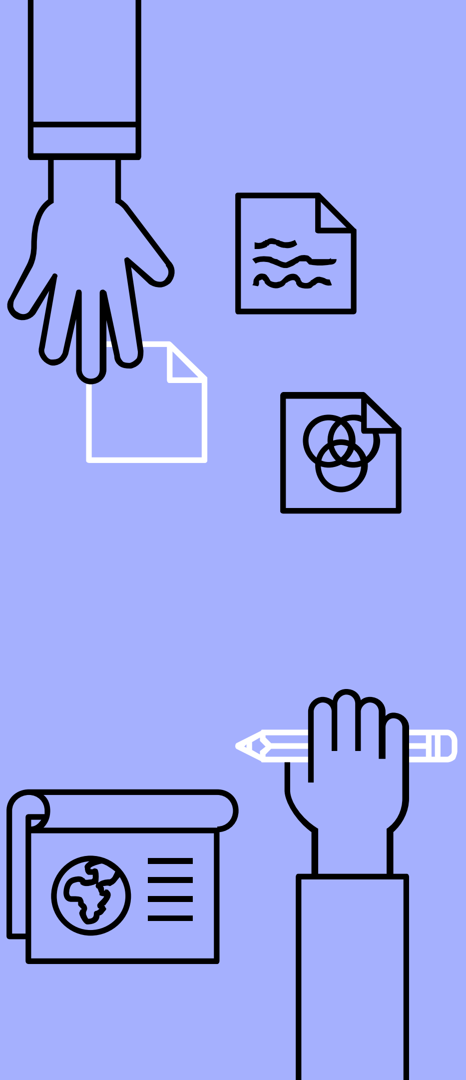
Minimization Function ⇒ Heuristic Cost Function
Heuristic Cost ⇒Distance, Time, Money Spent



Algorithm for Hill Climbing

- ➔ Evaluate the initial state. If it is the goal state then return and quit. Otherwise continue with initial state as current state.

- ➔ Loop until a solution is found or until there are no new operators left to be applied to the current state.
  - ◆ Select operator that has not been applied to the current state and apply it to produce the new state.
    - Evaluate the new state if --
      - If it is the goal state then return and quit
      - If it is not a goal state but better than the current state then make it current state.
      - If it is not better than the current state, then continue in the loop.



Probabilistic Reasoning

A probabilistic model describes the world in terms of a set S of possible states - the sample space.

Since, we don't know the true state of the world, we come up with a probability of any state being the true one.

Product rule:

$$
\begin{aligned}
P(A, B|C) &= P(A|B,C)P(B|C) \\
&= P(B|A,C)P(A|C)
\end{aligned}
$$

Bayes' rule:

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

Used in Bayesian statistics :

$$P(Model|Data) = \frac{P(Model)P(Data|Model)}{P(Data)}$$

THANKS!

Any questions?

You can find me at:

ankita90@linkedin.com