

# PROBLEM SOLVING AGENTS

**PROBLEM-SOLVING APPROACH IN ARTIFICIAL INTELLIGENCE PROBLEMS**

The **reflex agents** are known as the simplest agents because they directly map states into actions. Unfortunately, these agents fail to operate in an environment where the mapping is too large to store and learn. **Goal-based agent,** on the other hand, considers future actions and the desired outcomes.

Here, we will discuss one type of goal-based agent known as a **problem-solving agent**, which uses atomic representation with no internal states visible to the *problem-solving algorithms*.

# Problem-solving agent

The problem-solving agent perfoms precisely by defining problems and its several solutions.

* + - According to psychology, “*a problem-solving refers to a state where we wish to reach to a definite goal from a present state or condition.”*
    - According to computer science, *a problem-solving is a part of artificial intelligence which encompasses a number of techniques such as algorithms, heuristics to solve a problem.*

Therefore, a problem-solving agent is a **goal-driven agent** and focuses on satisfying the goal.

# PROBLEM DEFINITION

To build a system to solve a particular problem, we need to do four things:

1. **Define** the problem precisely. This definition must include specification of the initial situations and also final situations which constitute (i.e) acceptable solution to the problem.
2. **Analyze** the problem (i.e) important features have an immense (i.e) huge impact on the appropriateness of various techniques for solving the problems.
3. **Isolate and represent** the knowledge to solve the problem.
4. **Choose the best** problem – solving techniques and apply it to the particular problem.

**Steps performed by Problem-solving agent**

* + - * **Goal Formulation:** It is the first and simplest step in problem-solving. It organizes the steps/sequence required to formulate one goal out of multiple goals as well as actions to achieve that goal. Goal formulation is based on the current situation and the agent’s performance measure (discussed below).
      * **Problem Formulation:** It is the most important step of problem-solving which decides what actions should be taken to achieve the formulated goal. There are following five components involved in problem formulation:
      * **Initial State:** It is the starting state or initial step of the agent towards its goal.
      * **Actions:** It is the description of the possible actions available to the agent.
      * **Transition Model:** It describes what each action does.
      * **Goal Test:** It determines if the given state is a goal state.
      * **Path cost:** It assigns a numeric cost to each path that follows the goal. The problem- solving agent selects a cost function, which reflects its performance measure. Remember, **an optimal solution has the lowest path cost among all the solutions.**

**Note: Initial state, actions**, and **transition model** together define the **state-space** of the problem implicitly. State-space of a problem is a set of all states which can be reached from the initial state followed by any sequence of actions. The state-space forms a directed map or graph where nodes are the states, links between the nodes are actions, and the path is a sequence of states connected by the sequence of actions.

* **Search:** It identifies all the best possible sequence of actions to reach the goal state from the current state. It takes a problem as an input and returns solution as its output.
* **Solution:** It finds the best algorithm out of various algorithms, which may be proven as the best optimal solution.
* **Execution:** It executes the best optimal solution from the searching algorithms to reach the goal state from the current state.

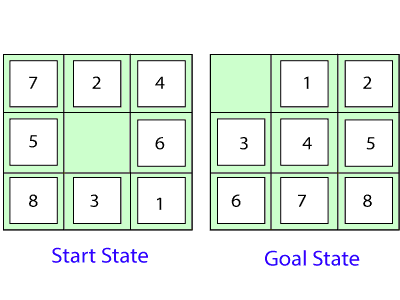
# Example Problems

Basically, there are two types of problem approaches:

* **Toy Problem:** It is a concise and exact description of the problem which is used by the researchers to compare the performance of algorithms.
* **Real-world Problem:** It is real-world based problems which require solutions. Unlike a toy problem, it does not depend on descriptions, but we can have a general formulation of the problem.

**Some Toy Problems**

* **8 Puzzle Problem:** Here, we have a 3×3 matrix with movable tiles numbered from 1 to 8 with a blank space. The tile adjacent to the blank space can slide into that space. The objective is to reach a specified goal state similar to the goal state, as shown in the below figure.
* In the figure, our task is to convert the current state into goal state by sliding digits into the blank space.

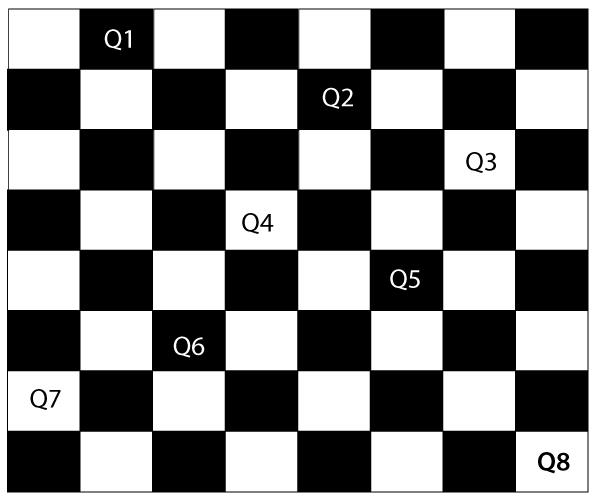


In the above figure, our task is to convert the current(Start) state into goal state by sliding digits into the blank space.

# The problem formulation is as follows:

* **States:** It describes the location of each numbered tiles and the blank tile.
* **Initial State:** We can start from any state as the initial state.
* **Actions:** Here, actions of the blank space is defined, i.e., either **left, right, up or down**
* **Transition Model:** It returns the resulting state as per the given state and actions.
* **Goal test:** It identifies whether we have reached the correct goal-state.
* **Path cost:** The path cost is the number of steps in the path where the cost of each step is 1. **Note:** The 8-puzzle problem is a type of [**sliding-block problem**](https://github.com/topics/sliding-puzzle-game?o=desc&s=updated) which is used for testing new search algorithms in [artificial intelligence.](https://www.tutorialandexample.com/artificial-intelligence-tutorial/)
* **8-queens problem:** The aim of this problem is to place eight queens on a chessboard in an order where no queen may attack another. A queen can attack other queens either **diagonally or in same row and column.**

From the following figure, we can understand the problem as well as its correct solution.



It is noticed from the above figure that each queen is set into the chessboard in a position where no other queen is placed diagonally, in same row or column. Therefore, it is one right approach to the 8-queens problem.

# For this problem, there are two main kinds of formulation:

1. **Incremental formulation:** It starts from an empty state where the operator augments a queen at each step.

# Following steps are involved in this formulation:

* **States:** Arrangement of any 0 to 8 queens on the chessboard.
* **Initial State:** An empty chessboard
* **Actions:** Add a queen to any empty box.
* **Transition model:** Returns the chessboard with the queen added in a box.
* **Goal test:** Checks whether 8-queens are placed on the chessboard without any attack.
* **Path cost:** There is no need for path cost because only final states are counted. In this formulation, there is approximately **1.8 x 1014** possible sequence to investigate.

1. **Complete-state formulation:** It starts with all the 8-queens on the chessboard and moves them around, saving from the attacks.

# Following steps are involved in this formulation

* **States:** Arrangement of all the 8 queens one per column with no queen attacking the other queen.
* **Actions:** Move the queen at the location where it is safe from the attacks.

This formulation is better than the incremental formulation as it reduces the state space from **1.8 x 1014** to **2057**, and it is easy to find the solutions.

# Some Real-world problems

* + **Traveling salesperson problem(TSP):** It is a **touring problem** where the salesman can visit each city only once. The objective is to find the shortest tour and sell-out the stuff in each city.
  + **VLSI Layout problem:** In this problem, millions of components and connections are positioned on a chip in order to minimize the area, circuit-delays, stray-capacitances, and maximizing the manufacturing yield.

# The layout problem is split into two parts:

* + **Cell layout:** Here, the primitive components of the circuit are grouped into cells, each performing its specific function. Each cell has a fixed shape and size. The task is to place the cells on the chip without overlapping each other.
  + **Channel routing:** It finds a specific route for each wire through the gaps between the cells.
  + **Protein Design:** The objective is to find a sequence of amino acids which will fold into 3D protein having a property to cure some disease.

# Searching for solutions

We have seen many problems. Now, there is a need to search for solutions to solve them.

In this section, we will understand how searching can be used by the agent to solve a problem.

For solving different kinds of problem, an agent makes use of different strategies to reach the goal by searching the best possible algorithms. This process of searching is known as **search strategy.**

**State Space Search in Artificial Intelligence**

**Overview**

**State space search** is a problem-solving technique used in Artificial Intelligence (AI) to find the solution path from the initial state to the goal state by exploring the various states. The state space search approach searches through all possible states of a problem to find a solution. It is an essential part of Artificial Intelligence and is used in various applications, from game-playing algorithms to natural language processing.

Introduction

A **state space** is a way to mathematically represent a problem by defining all the possible states in which the problem can be. This is used in search algorithms to represent the initial state, goal state, and current state of the problem. Each state in the state space is represented using a set of variables.

The **efficiency** of the search algorithm greatly depends on the size of the state space, and it is important to choose an appropriate representation and search strategy to search the state space efficiently.

One of the most well-known **state space search algorithms** is the A algorithm. Other commonly used state space search algorithms include **breadth-first search (BFS)**, **depth-first search (DFS)**, **hill climbing**, **simulated annealing**, and **genetic algorithms**.

## Features of State Space Search

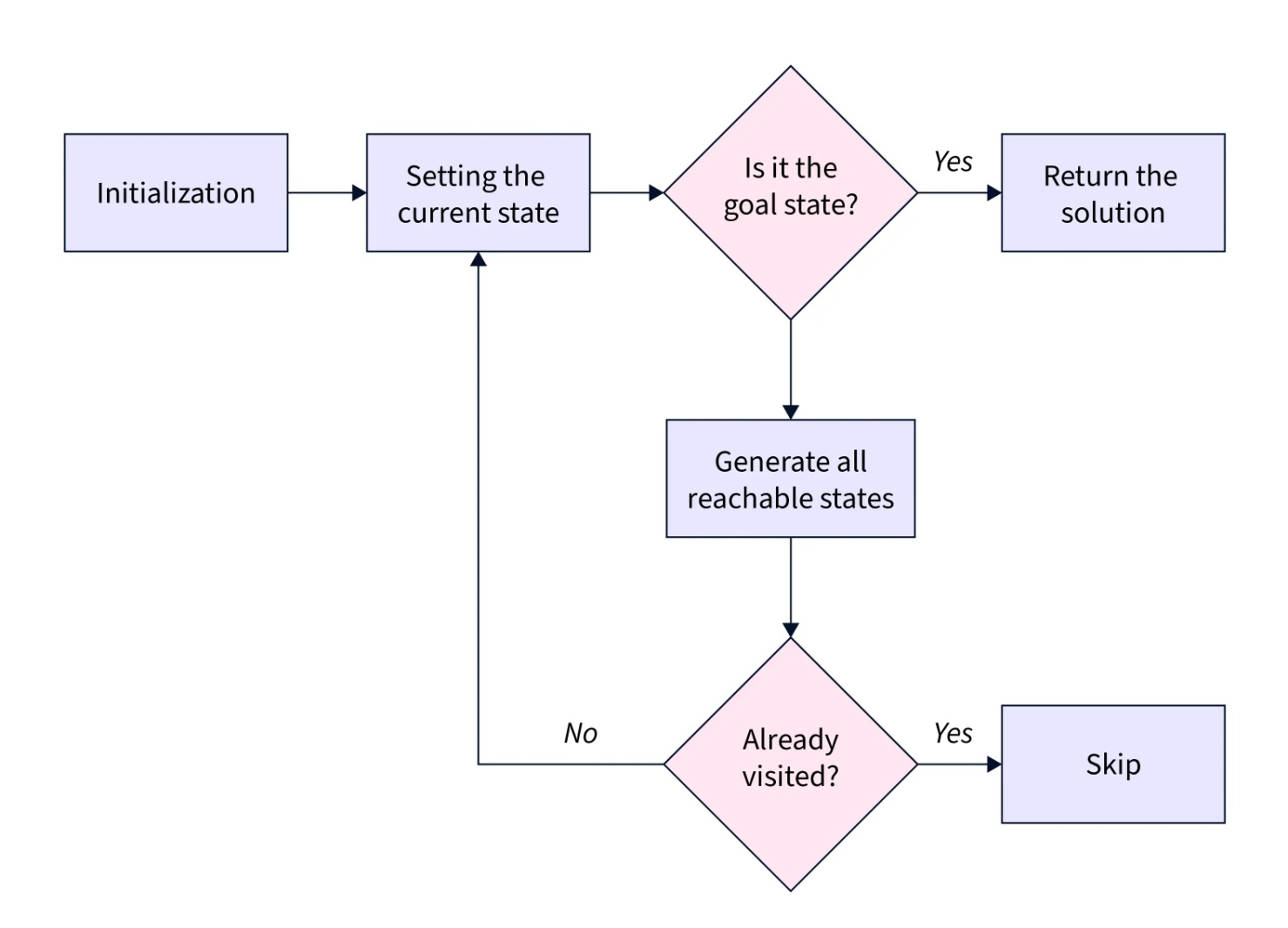
**State space search** has several features that make it an effective problem-solving technique in Artificial Intelligence. These features include:

* **Exhaustiveness:**  
  State space search explores all possible states of a problem to find a solution.
* **Completeness:**  
  If a solution exists, state space search will find it.
* **Optimality:**  
  Searching through a state space results in an optimal solution.
* **Uninformed and Informed Search:**  
  State space search in artificial intelligence can be classified as uninformed if it provides additional information about the problem.

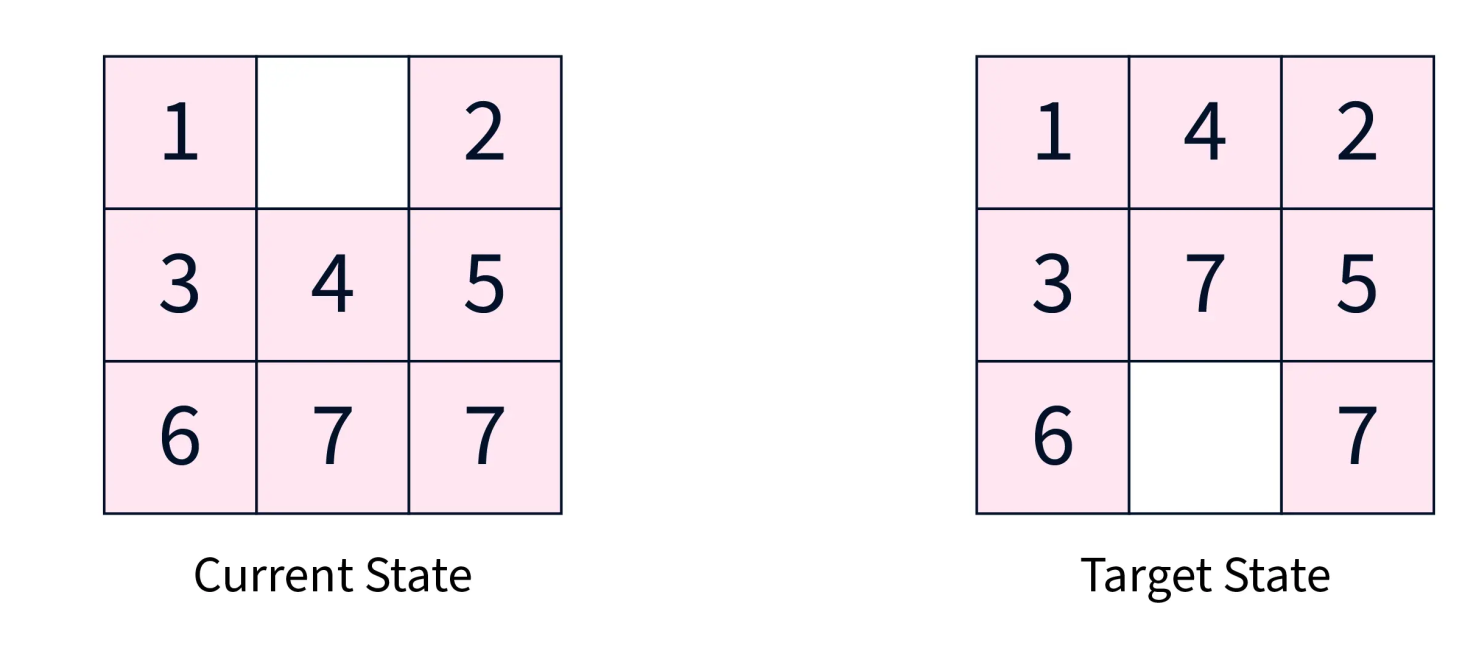
In contrast, informed search uses additional information, such as heuristics, to guide the search process.

## Steps in State Space Search

The steps involved in state space search are as follows:

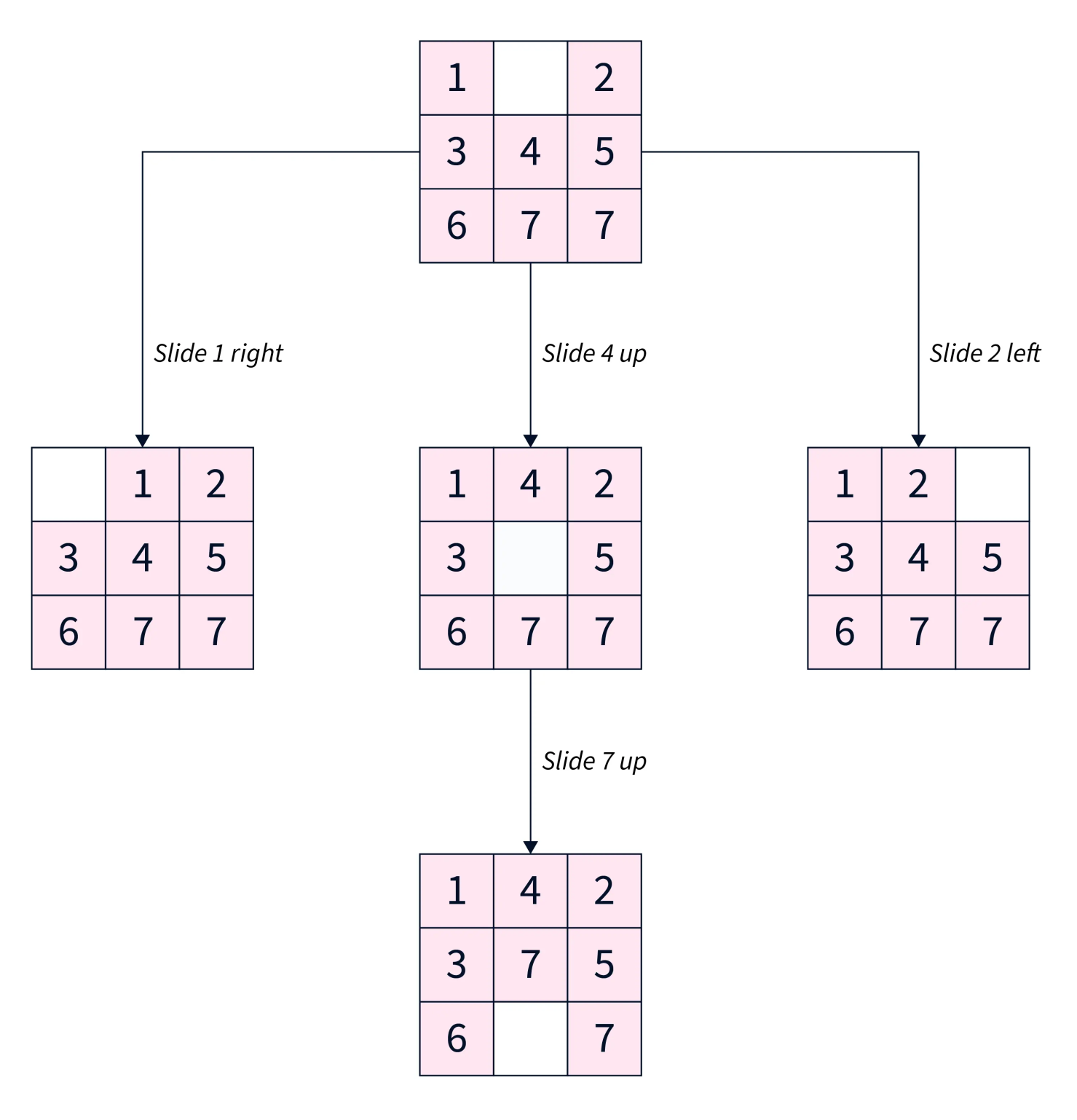


* To begin the search process, we set the current state to the initial state.
* We then check if the current state is the goal state. If it is, we terminate the algorithm and return the result.
* If the current state is not the goal state, we generate the set of possible successor states that can be reached from the current state.
* For each successor state, we check if it has already been visited. If it has, we skip it, else we add it to the queue of states to be visited.
* Next, we set the next state in the queue as the current state and check if it's the goal state. If it is, we return the result. If not, we repeat the previous step until we find the goal state or explore all the states.
* If all possible states have been explored and the goal state still needs to be found, we return with no solution.
* State Space Representation
* **State space Representation** involves defining an INITIAL STATE and a GOAL STATE and then determining a sequence of actions, called states, to follow.
* **State:**  
  A state can be an Initial State, a Goal State, or any other possible state that can be generated by applying rules between them.
* **Space:**  
  In an AI problem, space refers to the exhaustive collection of all conceivable states.
* **Search:**  
  This technique moves from the beginning state to the desired state by applying good rules while traversing the space of all possible states.
* **Search Tree:**  
  To visualize the search issue, a search tree is used, which is a tree-like structure that represents the problem. The initial state is represented by the root node of the search tree, which is the starting point of the tree.
* **Transition Model:**  
  This describes what each action does, while Path Cost assigns a cost value to each path, an activity sequence that connects the beginning node to the end node. The optimal option has the lowest cost among all alternatives.
* Example of State Space Search
* The **8-puzzle** problem is a commonly used example of a state space search. It is a sliding puzzle game consisting of 8 numbered tiles arranged in a 3x3 grid and one blank space. The game aims to rearrange the tiles from their initial state to a final goal state by sliding them into the blank space.
* To represent the state space in this problem, we use the nine tiles in the puzzle and their respective positions in the grid. Each state in the state space is represented by a 3x3 array with values ranging from 1 to 8, and the blank space is represented as an empty tile.
* The initial state of the puzzle represents the starting configuration of the tiles, while the goal state represents the desired configuration. **Search algorithms** utilize the state space to find a sequence of moves that will transform the initial state into the goal state.



This algorithm guarantees a solution but can become very slow for larger state spaces. Alternatively, other algorithms, such as **A search**, use heuristics to guide the search more efficiently.

Our objective is to move from the current state to the target state by sliding the numbered tiles through the blank space. Let's look closer at reaching the target state from the current state.



To summarize, our approach involved exhaustively exploring all reachable states from the current state and checking if any of these states matched the target state.

## Applications of State Space Search

* State space search algorithms are used in various fields, such as robotics, game playing, computer networks, operations research, bioinformatics, cryptography, and supply chain management. In artificial intelligence, state space search algorithms can solve problems like **pathfinding**, **planning**, and **scheduling**.
* They are also useful in planning robot motion and finding the best sequence of actions to achieve a goal. In games, state space search algorithms can help determine the best move for a player given a particular game state.
* **State space search algorithms** can optimize routing and resource allocation in computer networks and operations research.
* In **Bioinformatics**, state space search algorithms can help find patterns in biological data and predict protein structures.
* In **Cryptography**, state space search algorithms are used to break codes and find cryptographic keys.

## Conclusion

* **State Space Search** is a problem-solving technique used in AI to find a solution to a problem by exploring all possible states of the problem.
* It is an exhaustive, complete, and optimal search process that can be classified as uninformed or informed.
* **State Space Representation** is a crucial step in the state space search process as it determines the efficiency of the search algorithm.
* State space search in artificial intelligence has several applications, including game-playing algorithms, natural language processing, robotics, planning, scheduling, and computer vision.