

Problem Solving as Search & State Space Search

Dr. Mohamed Mostafa

AI 511 Foundations of Artificial Intelligence

Lecture 3



3.1 Solving problems by Searching

- Solving Problems by Searching
- Types of Agents (Reflex vs. Planning)
- State Space Search
- Example: Tic-Tac-Toe Game
- Example: 8-Queens Problem
- How human beings think?
- Heuristic Search

3.2 State Space Search Graph & Strategies

- Search Problem Components
- Example: Romania
- State Space & State Space Graph
- Example: Vacuum World
- Example: The 8-Puzzle
- Example: Robot Motion Planning

- Example: Tic-Tac-Toe
- Example: Traveling Salesperson
- State Space Search Strategies
- Selecting Search Strategy

3.3 Basic Idea of Search & the Backtracking Search Algorithm

- Search: Basic idea
- Search Tree
- Tree Search Algorithm Outline
- Tree Search Example
- Handling Repeated States
- Backtracking Search
- Backtracking Algorithm Data Structures
- Backtracking Algorithm

3.4 Practice Exercises (Solved Problems)

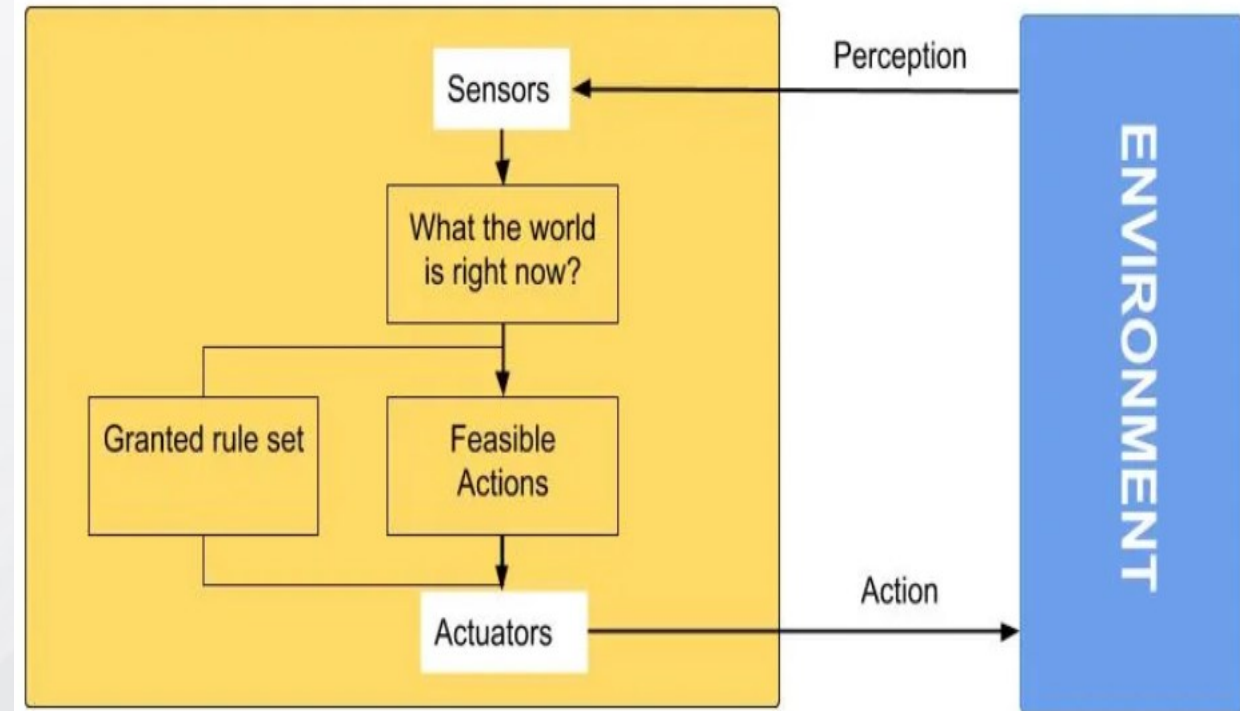
Resources for this lecture

- This lecture covers the following chapters:
 - **Chapter 3 (Solving Problems by Searching; only sections 3.1, 3.2, & 3.3)** *from* Stuart J. Russell and Peter Norvig, "Artificial Intelligence: A Modern Approach," Third Edition (2010), by Pearson Education Inc.
- .. AND ..
 - **Part II (pages 41 to 45) and Chapter 3 (Structures and Strategies for State Space Search; only sections 3.0, 3.1, and 3.2)** *from* George F. Luger, "Artificial Intelligence: Structures and strategies for complex problem solving," Fifth Edition (2005), Pearson Education Limited.
- .. AND ..
 - Introduction to Artificial Intelligence Course, Dr. Amr S. Ghoneim

Types of Agents

a Simple Reflex Agent :

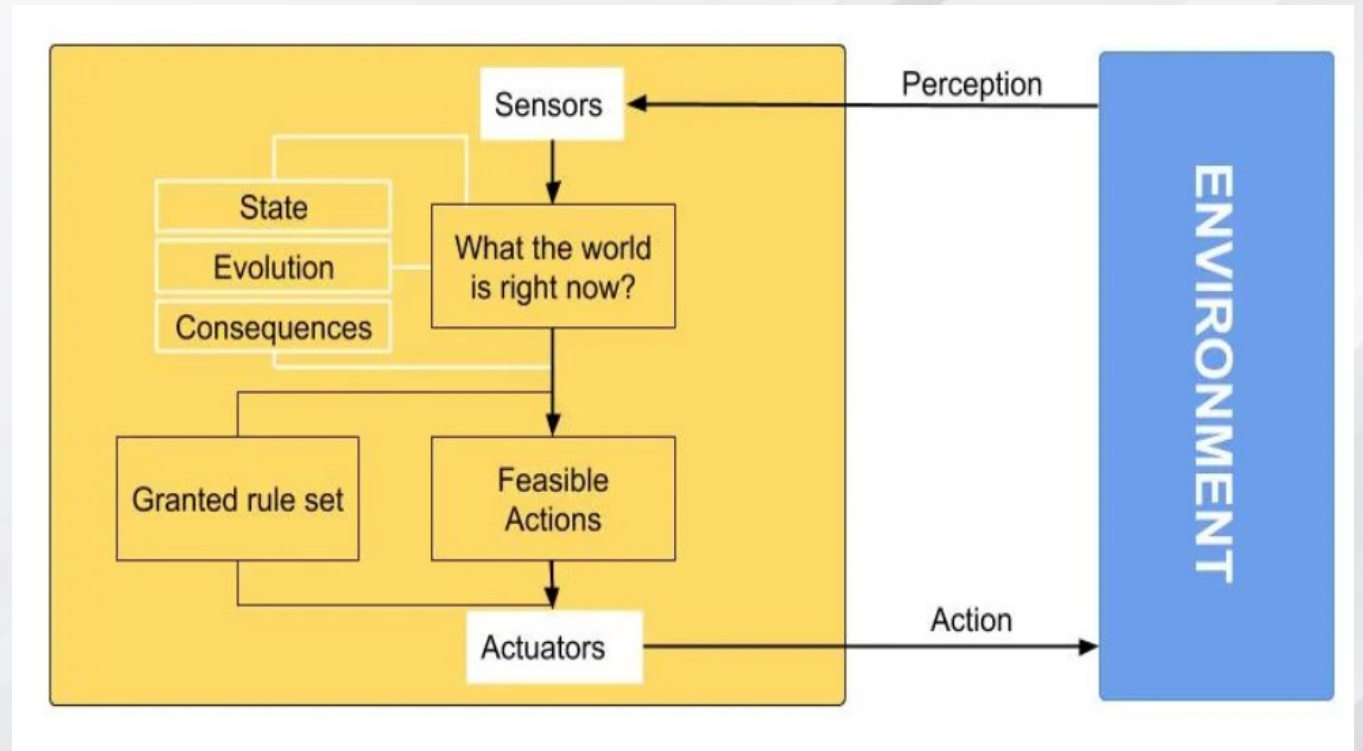
- **Considers how the world IS**
- Not adaptive to changes within the environment
- Choose action based on current percept.
- Do not consider the future consequences of actions.



Types of Agents

a Model based reflex agent:

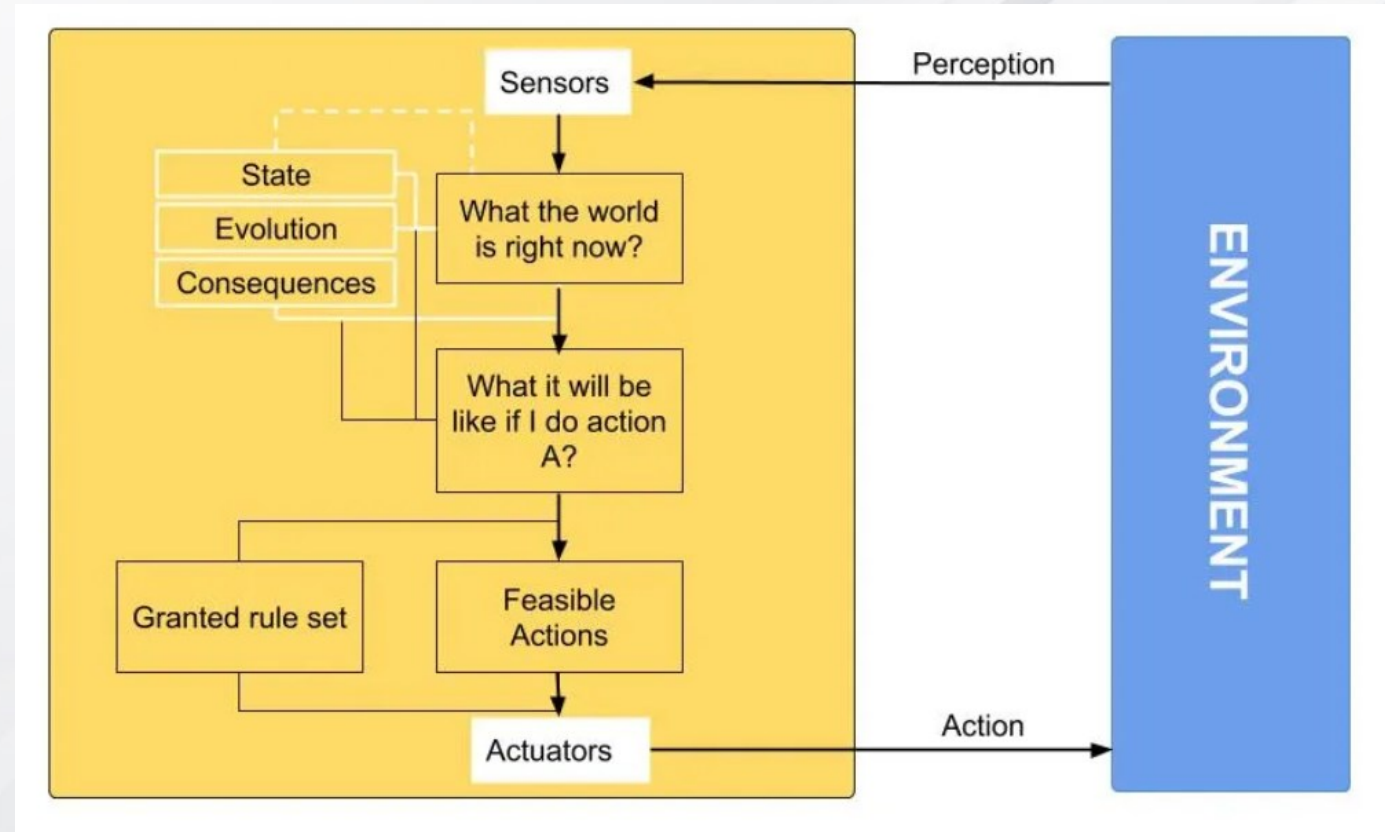
- Considers how the world WOULD BE
- Decisions based on (hypothesized) consequences of actions.
- Must have a model of how the world evolves in response to actions.
- Must formulate a goal.



Types of Agents

a Goal based agent:

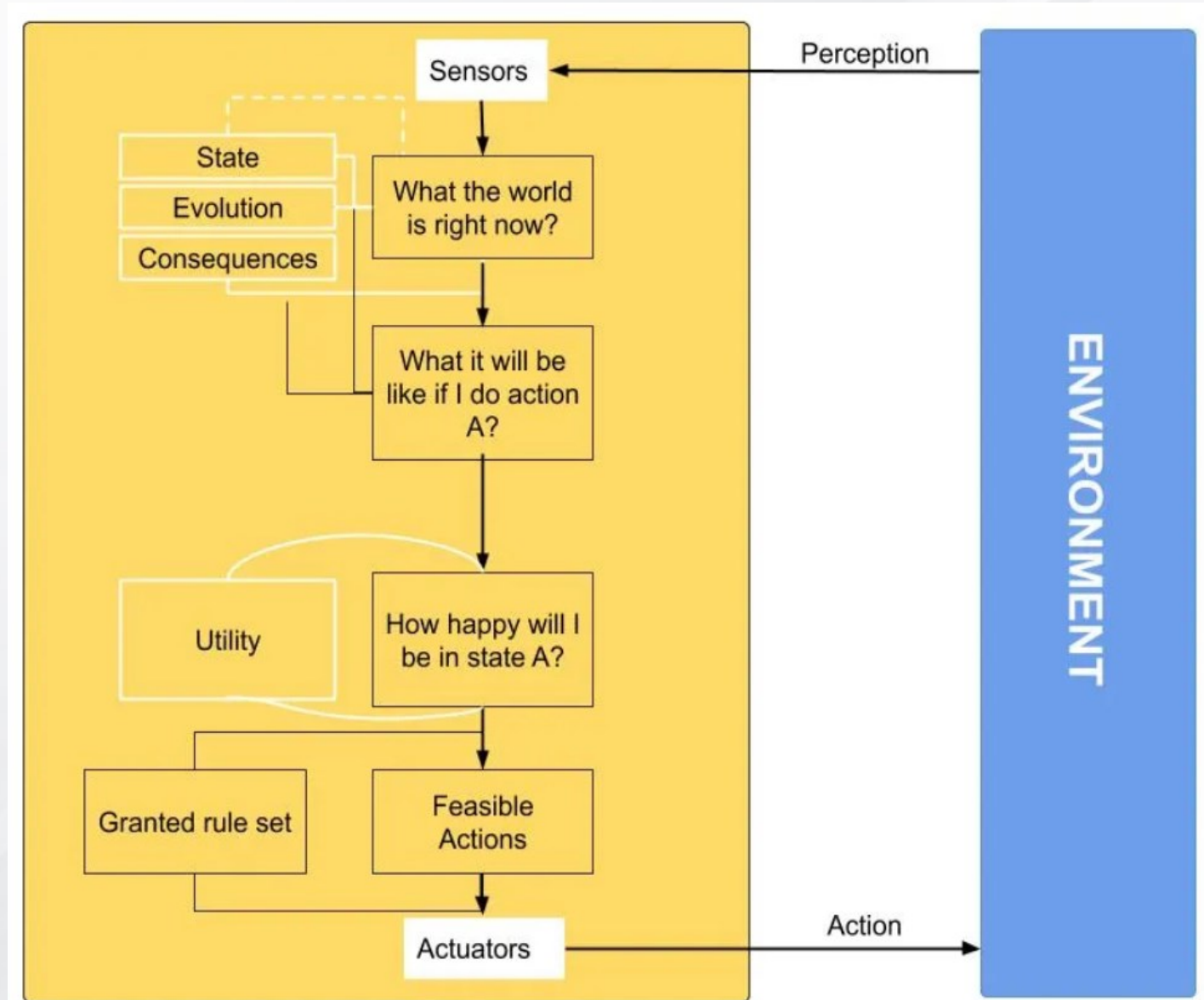
- The agent takes decisions based on the fact how far they are from the required goal.
- The agent chooses from multiple possibilities and selecting the one that finally lands it into the goal.
- they require searching and planning, so it can be explicitly be modified to make such agents more and more flexible.



Types of Agents

a Utility based agent:

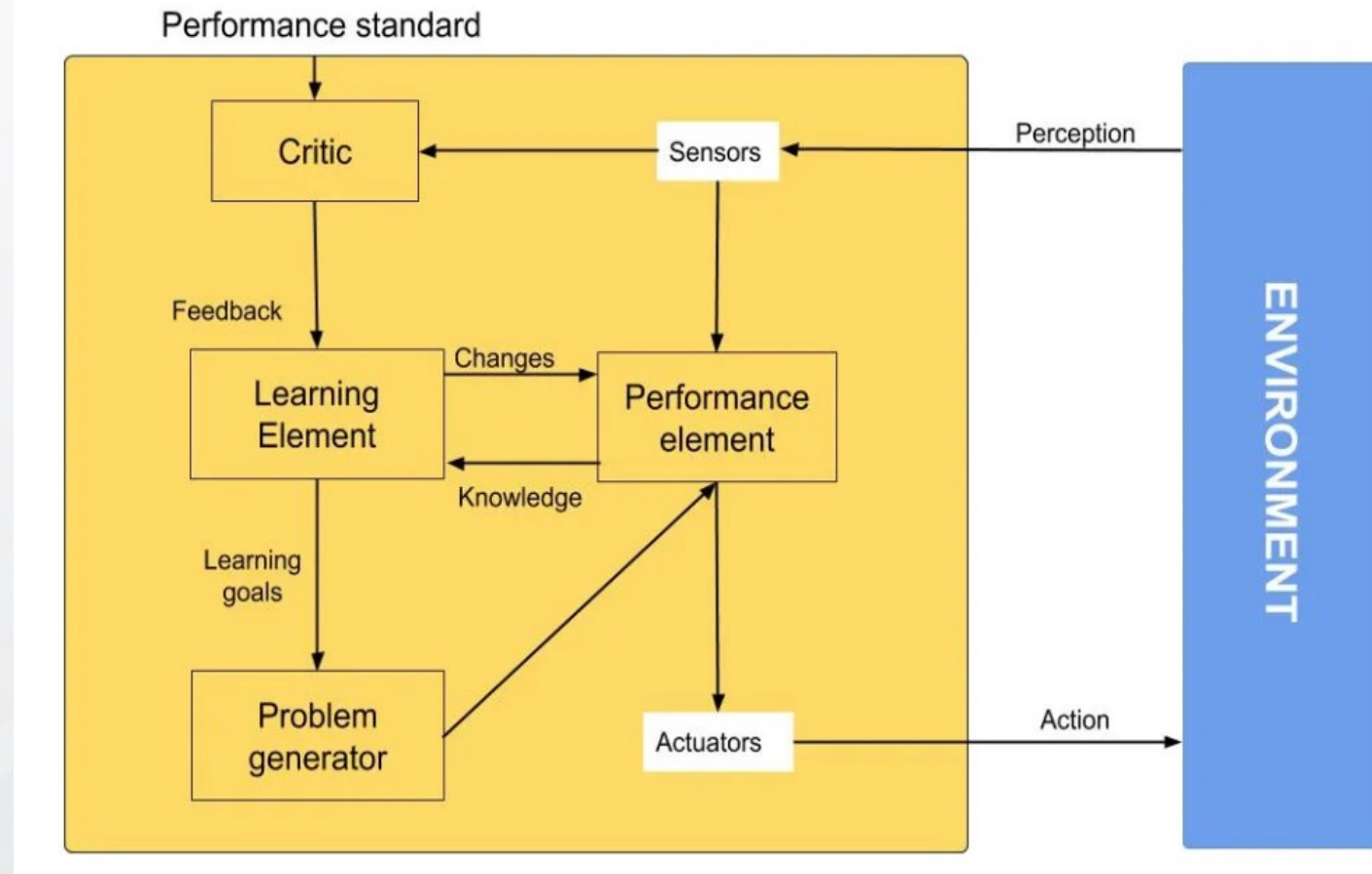
- When there are more than one possibilities, then to decided the best among them, utility agents are useful. The action they choose are based on the utility of each state.



Types of Agents

a Learning agent:

- Different from all the agents mentioned above, a learning agent is the one that takes consideration of the past experiences and taking them into account, learns from them.
- Due to its capabilities of learning from the past experiences, it can be ported to adapt to newer goals by generating newer and newer experiences. It start to act from the basic knowledge and builds upon the same through automated learning.

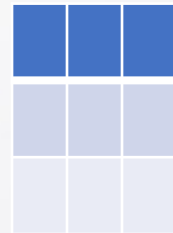


State Space Search

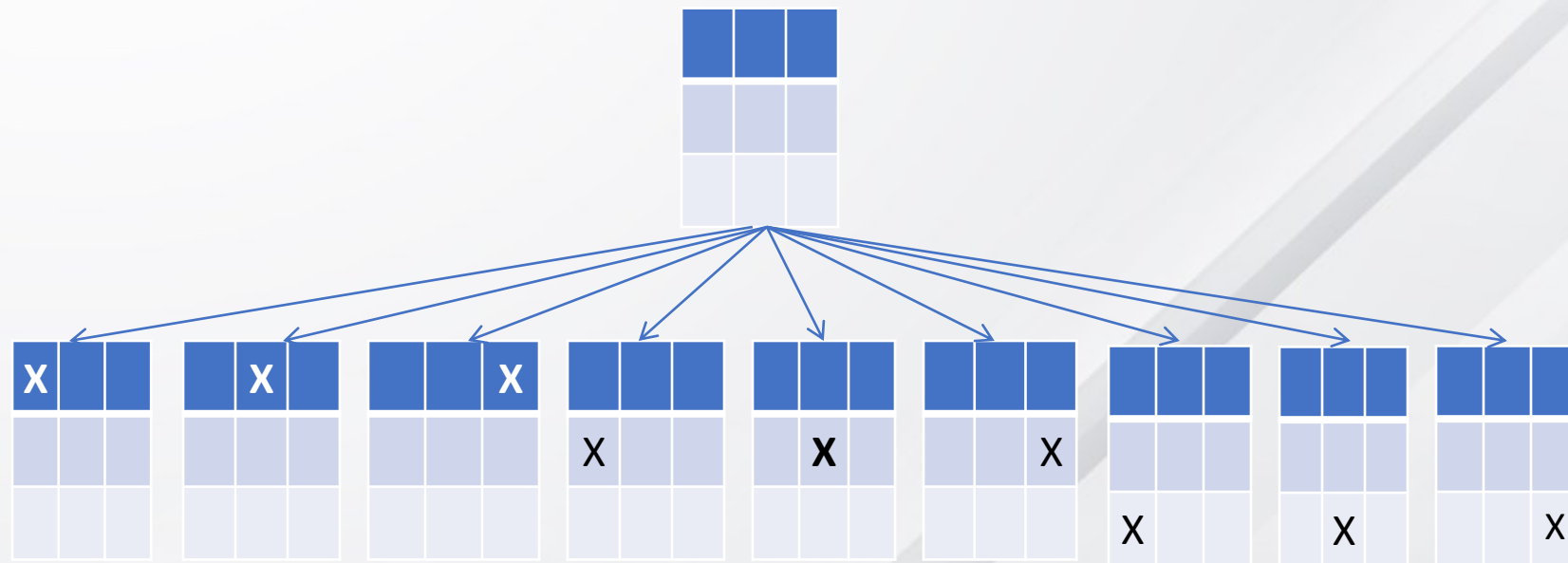
Problems are solved by searching among alternative choices.

- Humans consider several alternative strategies on their way to solving a problem.
 - *A Chess player* considers a few alternative moves.
 - *Supply chain managers* analyze multiple suppliers, transportation routes, and inventory management strategies to optimize the efficiency, cost-effectiveness, and reliability of the supply chain operations.
 - *Software engineers* explore alternative algorithms, programming languages, or architectural designs to develop robust and efficient software solutions that meet user requirements and performance criteria.

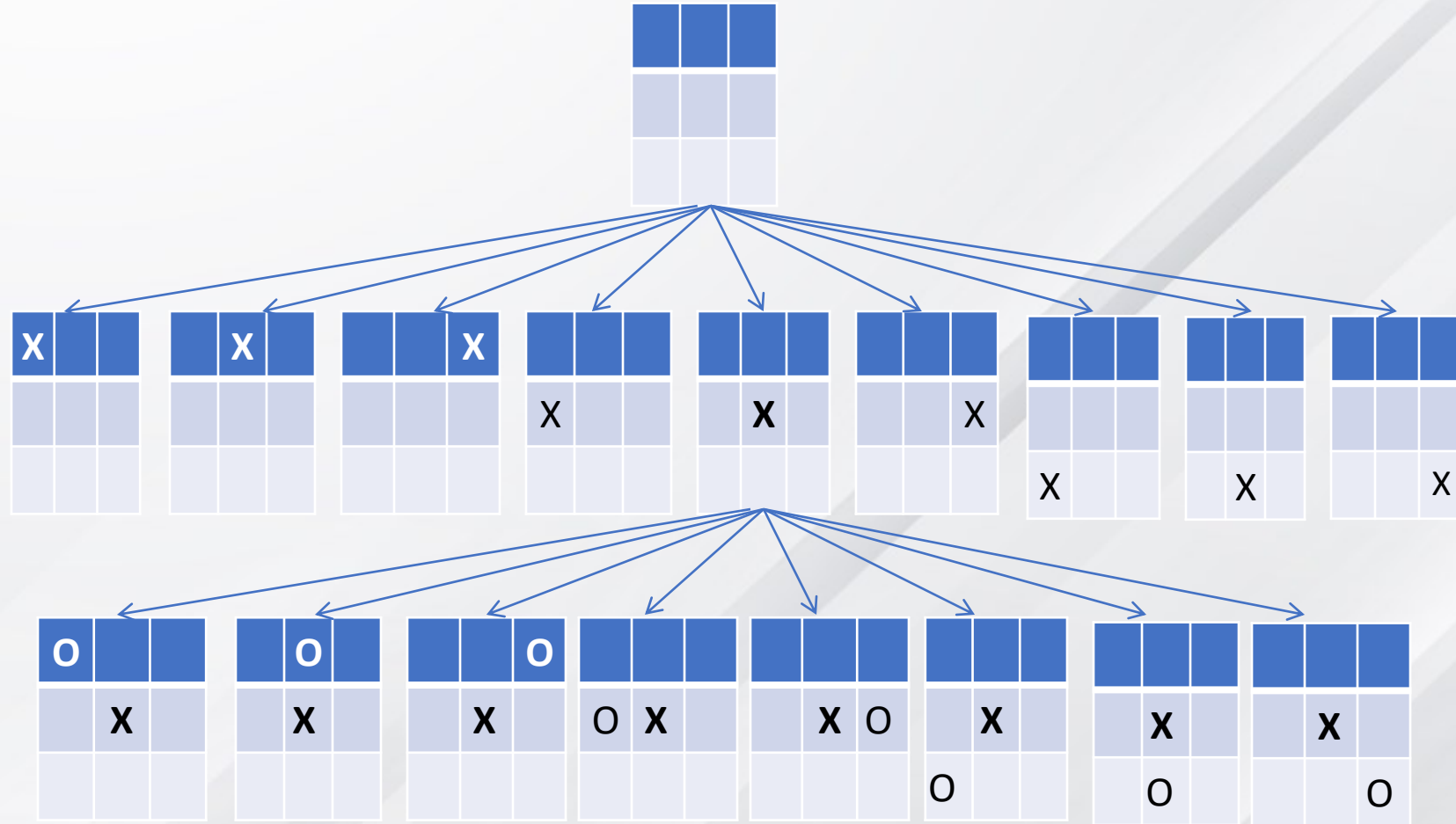
Example: Tic-Tac-Toe Game



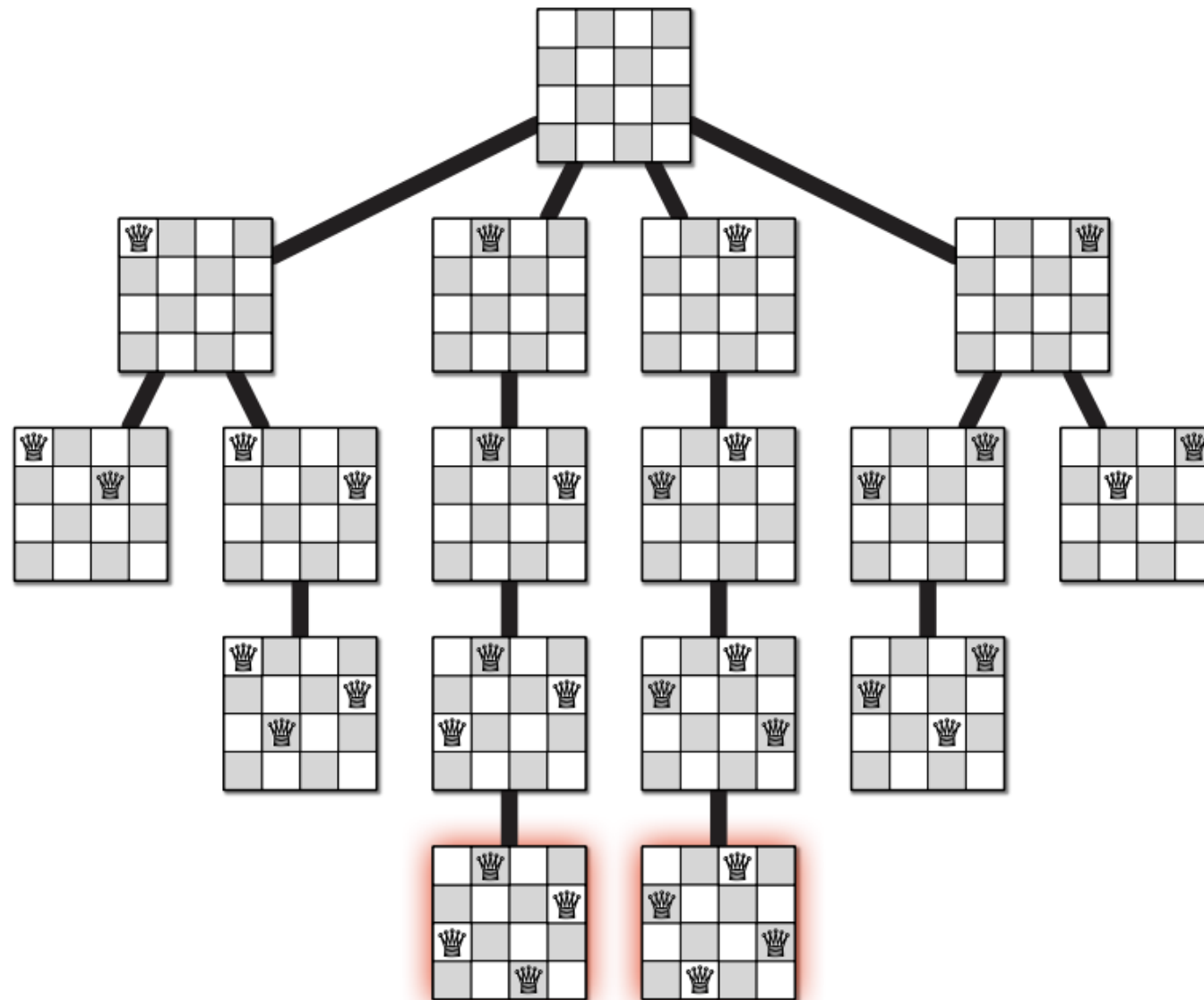
Example: Tic-Tac-Toe Game



Example: Tic-Tac-Toe Game



Example: 8-Queens Problem



How Human Beings Think..?

- Human beings do not engage in search of the entire space. Instead, they focus on exploring alternatives have proven to be effective through experience. Human problem-solving relies on judgmental rules that restrict the exploration of the search space to areas of the space that appear promising. These guiding rules are commonly referred to as "heuristics."



Heuristic Search

- A heuristic is a method employed to expedite problem-solving compared to traditional approaches.
- These methods are utilized to approximate solutions when conventional methods are not effective.
- Heuristics are strategies that are derived from past experience with similar problems.
- They employ practical techniques and shortcuts to generate solutions, which may not always be optimal but are adequate within a specified timeframe.

A Heuristic search vs an Exhaustive search?

An Exhaustive Search

- Generate all possible solutions
- Evaluates all generated solutions
- Select the best solution
- Guarantee a solution
- Examples of exhaustive search algorithms include depth-first search (DFS), breadth-first search (BFS), and backtracking.

A Heuristic Search

- Generate a subset of solutions (random)
- Evaluates the generated solutions
- Select the best solutions
- Generate new subset of solution from the selected
- No guarantee that the solution is found

A Heuristic search vs an Exhaustive search?

An Exhaustive Search

- Small search space
- Known search space
- Need the best solution
- Deterministic

A Heuristic Search

- Huge search space
- Unknown search space
- A suboptimal solution is valid
- Not deterministic
- Heuristic search algorithms are often more efficient than exhaustive search methods, particularly for large or complex problem spaces.
- Examples of heuristic search algorithms include A* search, greedy best-first search, and hill climbing.

Advantages and Disadvantages of Heuristic search algorithms

- **Advantages of heuristic search algorithms:**

- 1. **Efficiency:** Heuristic search algorithms can be much faster than exhaustive search methods, especially in cases where there are a large number of possible states or options to consider.
- 2. **Memory Efficiency:** They typically require less memory because they do not need to store the entire search space.
- 3. **Suitability for Complex Problems:** Heuristic algorithms are well-suited for solving complex problems where an exhaustive search would be impractical due to the sheer number of possible solutions.

- **Disadvantages of heuristic search algorithms:**

- 1. **Approximate Solutions:** They may not always find the optimal solution. The solution they find may be close, but not guaranteed to be the best.
- 2. **Dependence on Heuristic Quality:** The effectiveness of a heuristic search algorithm heavily relies on the quality of the heuristic function. A poorly chosen heuristic may lead to suboptimal results.
- 3. **Difficulty in Designing Heuristics:** Crafting an effective heuristic can be challenging, especially for complex problems where finding a good heuristic is non-trivial.

Lecture 3: Solving Problem by Searching, & .. State Space Search Strategies and Structures

3.1 Solving problems by Searching

- Solving Problems by Searching
- Types of Agents (Reflex vs. Planning)
- State Space Search
- Example: Tic-Tac-Toe Game
- Example: 8-Queens Problem
- How Human Beings Think?
- Heuristic Search

3.2 State Space Search Graph & Strategies

- Search Problem Components
- Example: Romania
- State Space & State Space Graph
- Example: Vacuum World
- Example: The 8-Puzzle
- Example: Robot Motion Planning

- Example: Tic-Tac-Toe
- Example: Traveling Salesperson
- State Space Search Strategies
- Selecting Search Strategy

3.3 Basic Idea of Search & the Backtracking Search Algorithm

- Search: Basic idea
- Search Tree
- Tree Search Algorithm Outline
- Tree Search Example
- Handling Repeated States
- Backtracking Search
- Backtracking Algorithm Data Structures
- Backtracking Algorithm

3.4 Practice Exercises (Solved Problems)

Problem as a State Space Search

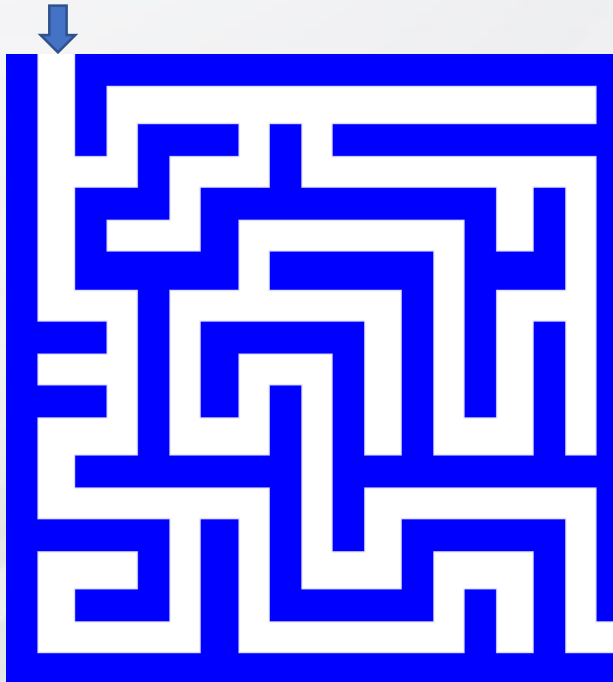
To build as system to solve a particular problem, we need:

- **Define the problem:** must include precise specifications
→ initial solution & final solution.
- **Analyze the problem:** select the most important features that can have an immense impact.
- **Isolate and represent:** convert these important features into knowledge representation.
- **Problem solving technique(s):** choose the best technique and apply it to particular problem.

Search

- We will consider the problem of designing **goal-based agents** in **fully observable, deterministic, discrete, known** environments.

Start State



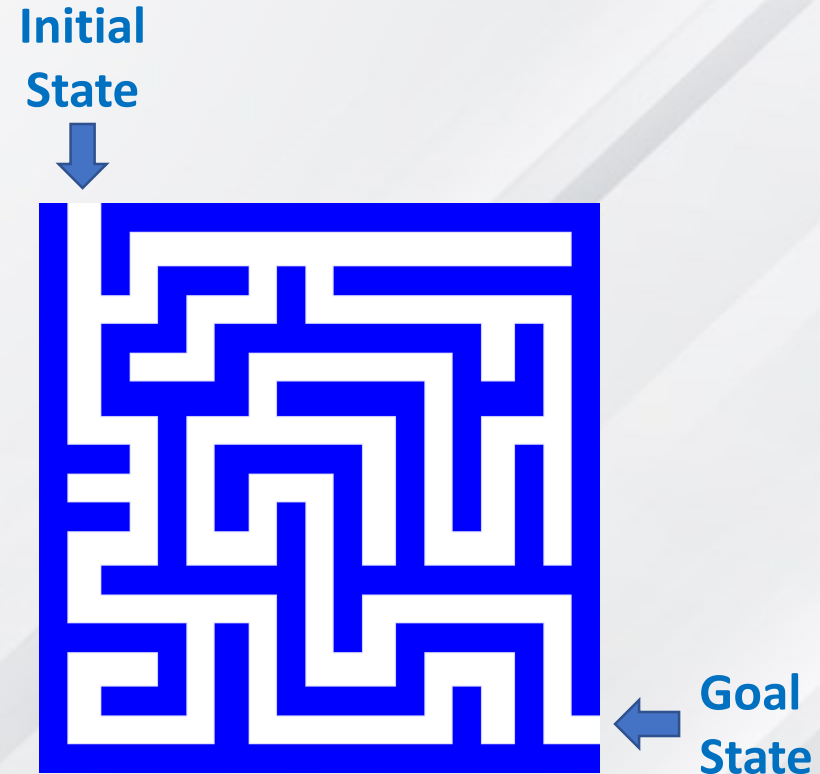
← Goal State

Search

- We will consider the problem of designing **goal-based agents** in **fully observable, deterministic, discrete, known** environments.
- The agent must find a *sequence of actions* that reaches the goal. The **performance measure** is defined by:
 - (a) reaching the goal, *and ..*
 - (b) how “expensive” the path to the goal is.

Search Problem Components

- **Initial state**
 - **Actions**
 - **Transition model**
 - What state results from performing a given action in each state?
 - **Goal state**
 - **Solution Path**
 - **Path cost**
 - Assume that it is a sum of nonnegative *step costs*
- The **optimal solution** is the sequence of actions that gives the lowest path cost for reaching the goal.



Example: Romania

- On vacation in Romania; currently in Arad.
- Flight leaves tomorrow from Bucharest.



Initial state

- Arad

Actions

- Go from one city to another

Transition Model

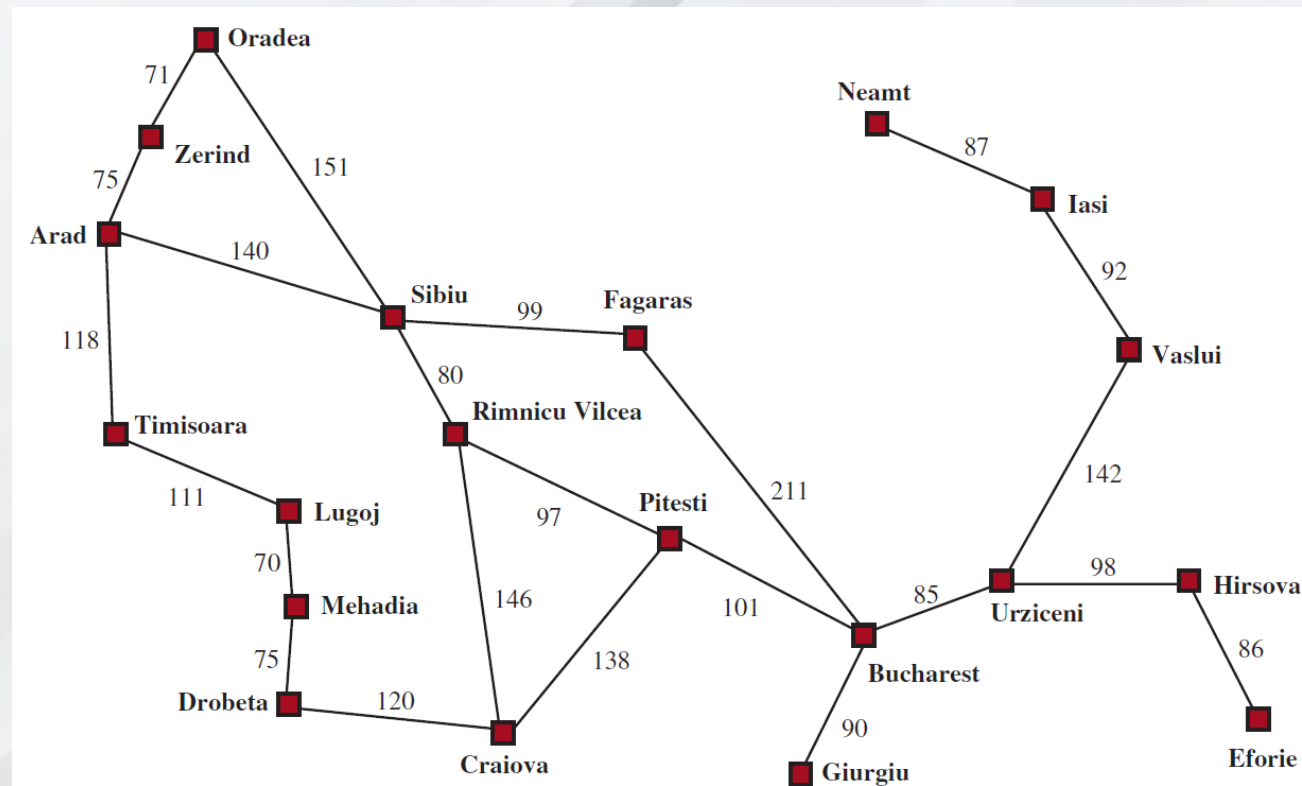
- If you go from city A to city B, you end up in city B

Goal State

- Bucharest

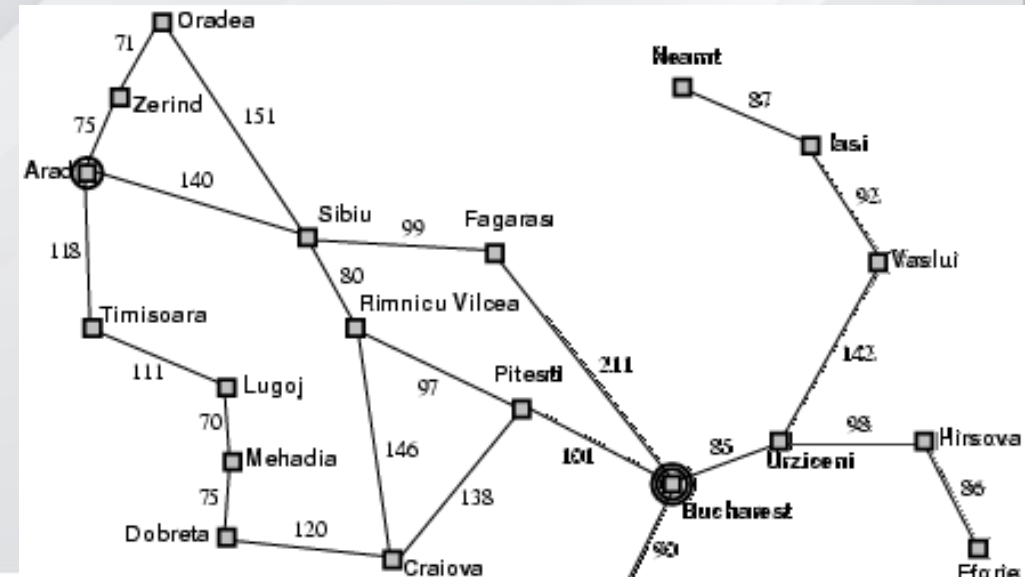
Path Cost

- Sum of edge costs (*total distance traveled*)

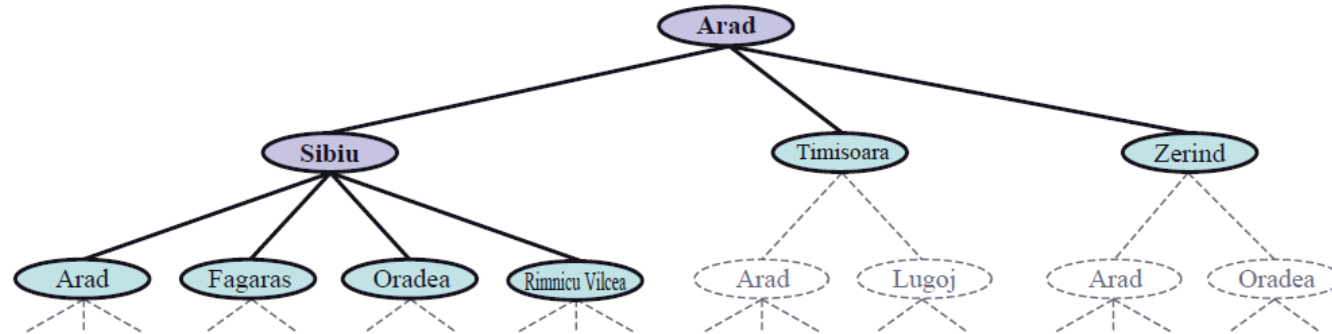
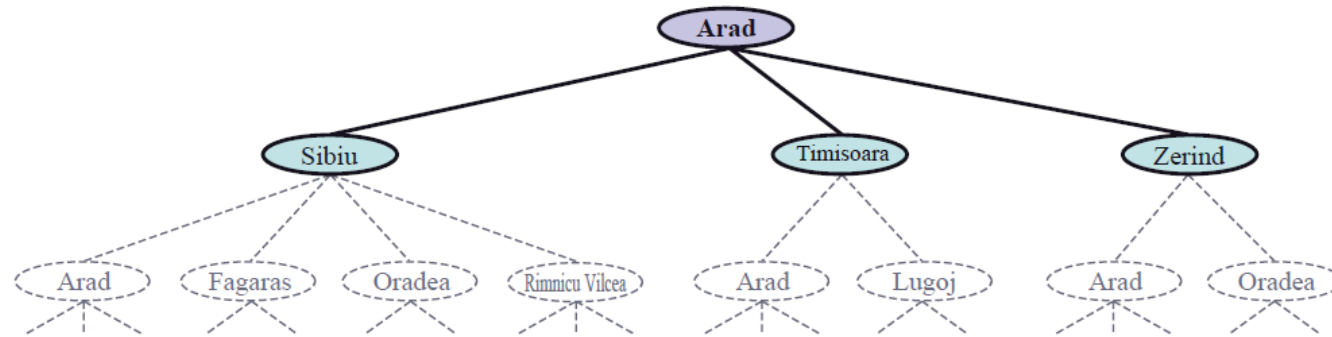
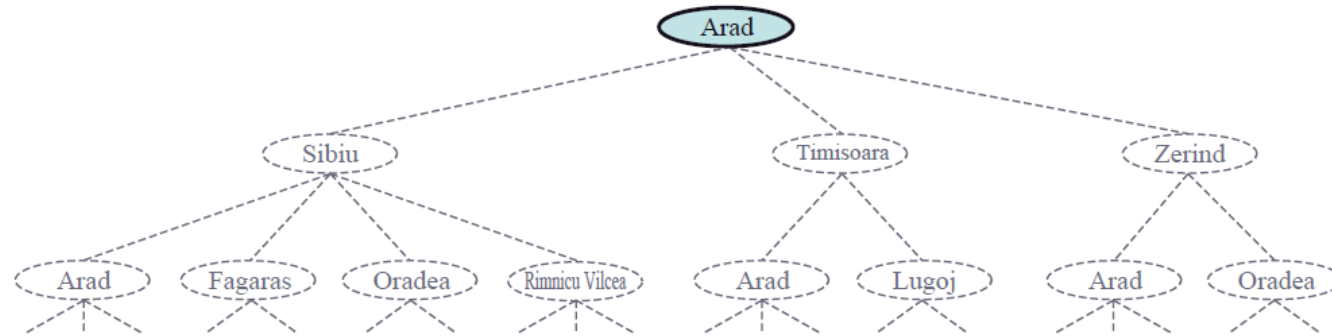


State Space

- The initial state, actions, and transition model define the **state space** of the problem;
 - The set of all states reachable from initial state by any sequence of actions.
 - Can be represented as a **directed graph** where the nodes are states and links between nodes are actions.
- What is the state space for the Romania problem?



Example: Romania



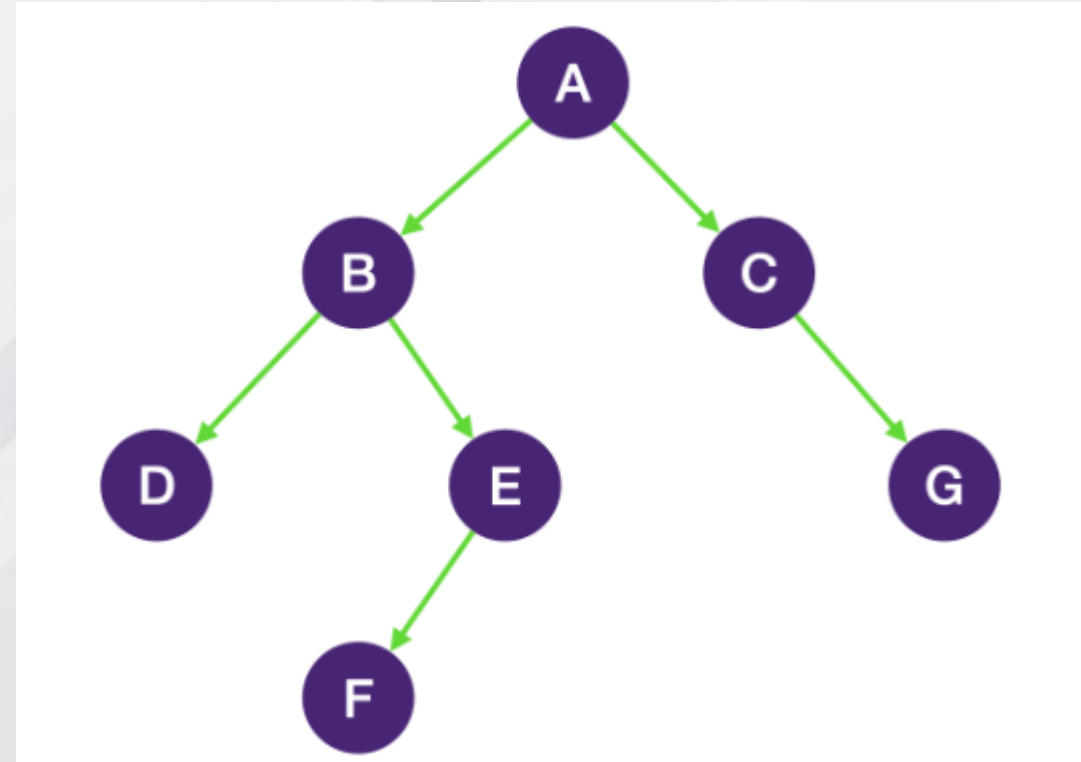
State Space

An AI problem can be represented as a *state space graph*.

A **graph** is a set of *nodes* and *links* that connect them.

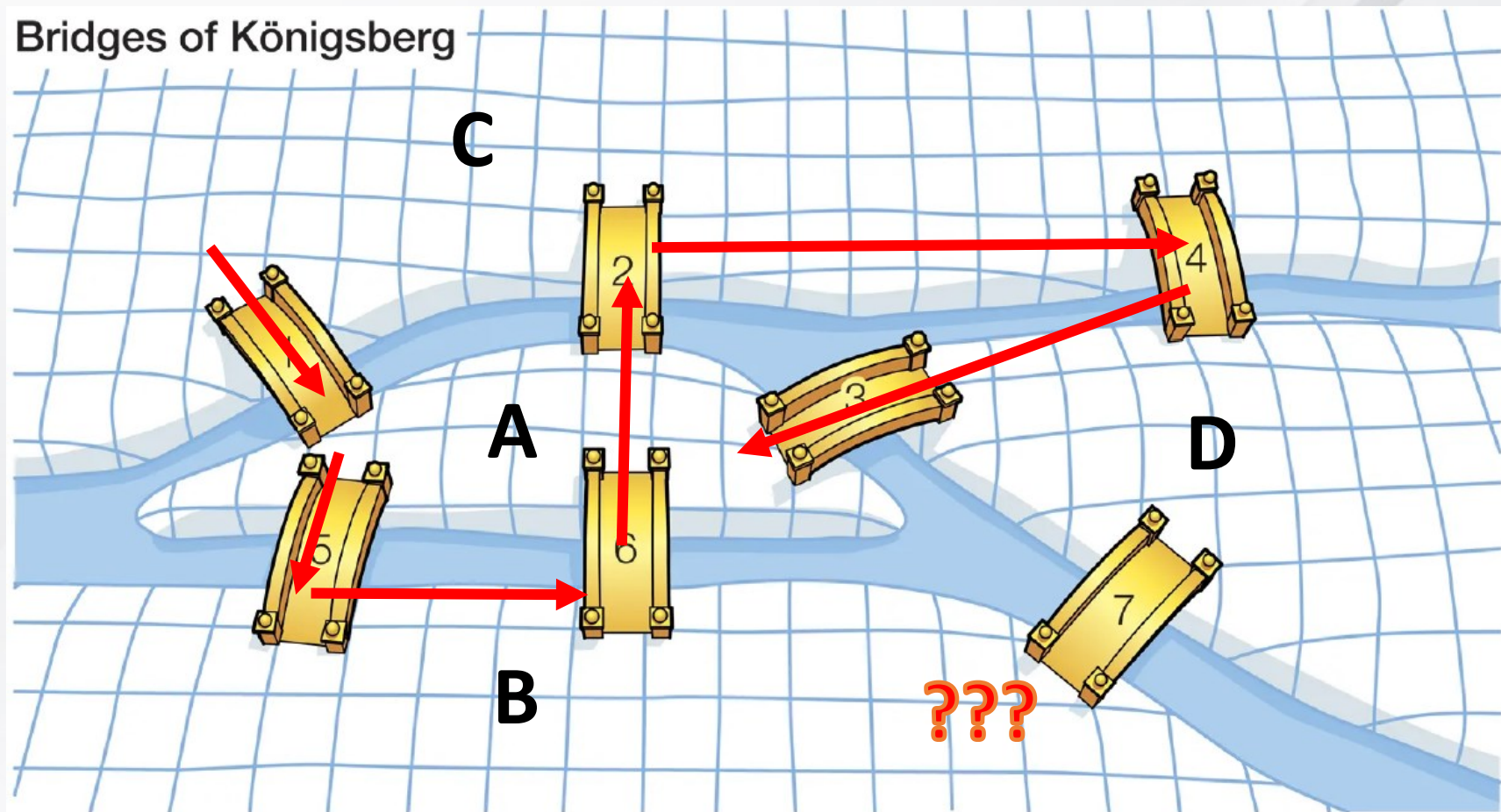
Graph theory:

- Labeled graph.
- Directed graph.
- Path.
- Rooted graph.
- Tree.
- Parent.
- Child.
- Sibling.
- Ancestor.
- Descendant.



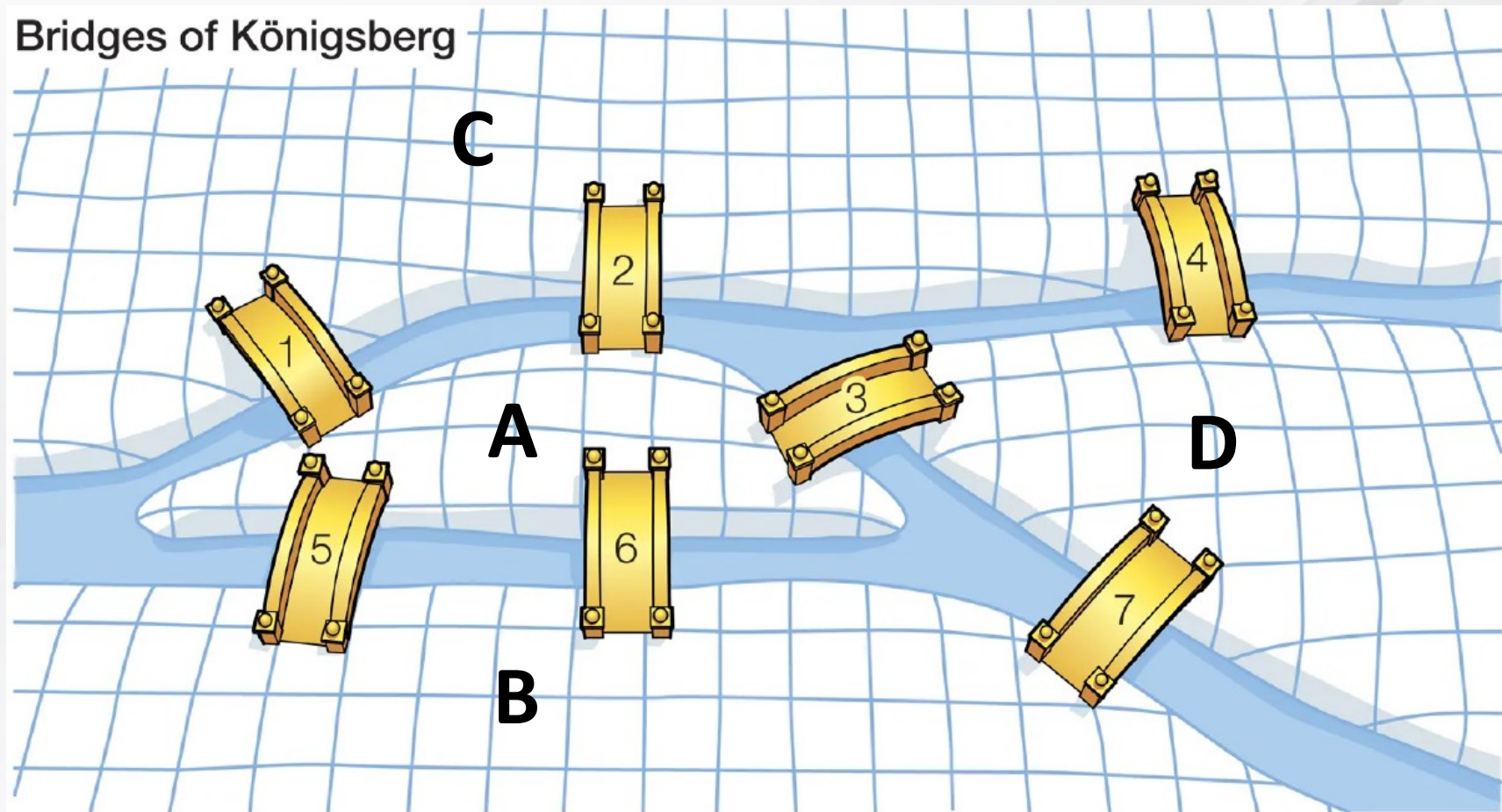
State Space

Euler Paths & the 7 Bridges of Königsberg, *Graph Theory*



State Space

Euler Paths & the 7 Bridges of Königsberg, *Graph Theory*



State Space

A state space is represented by four-tuple $[N, A, S, GD]$.

- **N**, is the set of nodes or states of the graph. These correspond to the states in the problem-solving process.
- **A**, is the set of arcs between nodes. These correspond to the steps in a problem-solving process.
- **S**, a nonempty subset of N , contains the start-state (s) of the problem.
- **GD**, a nonempty subset of N , contains the goal-state(s) of the problem. The states in GD are described using either:
 - A measurable property of the states encountered in the search.
 - A property of the solution path developed in the search (*a **solution path** is a path through this graph from a node in S to a node in GD .*)

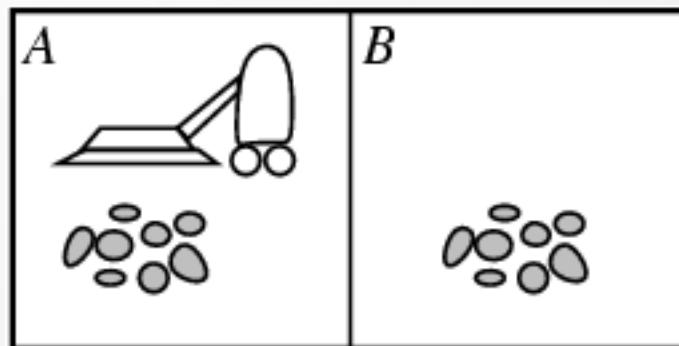
Example: Tic-Tac-Toe

- **Nodes(N)**: all the different configuration of Xs and Os that the game can have.
- **Arcs (A)**: generated by legal moves by placing an X or an O in unused location.
- **Start state (S)**: an empty board.
- **Goal states (GD)**: a board state having three Xs in a row, column, or diagonal.
- The arcs are directed, then no cycles in the state space, directed acyclic graph (DAG).
- **Complexity**: 9! Different paths can be generated.

Example: Traveling Salesperson

- A salesperson has five cities to visit and then must return home.
- Nodes(N): represent 5 cities.
- Arcs(A): labeled with weight indicating the cost of traveling between connected cities.
- Start state(S): a home city.
- Goal states(GD): an entire path contains a complete circuit with minimum cost.
- Complexity: $(n-1)!$ Different cost-weighted paths can be generated.

Example: Vacuum World



- **States:**

- Agent location and dirt location
- How many possible states?
- What if there are n possible locations?
 - The size of the state space grows exponentially with the “size” of the world!

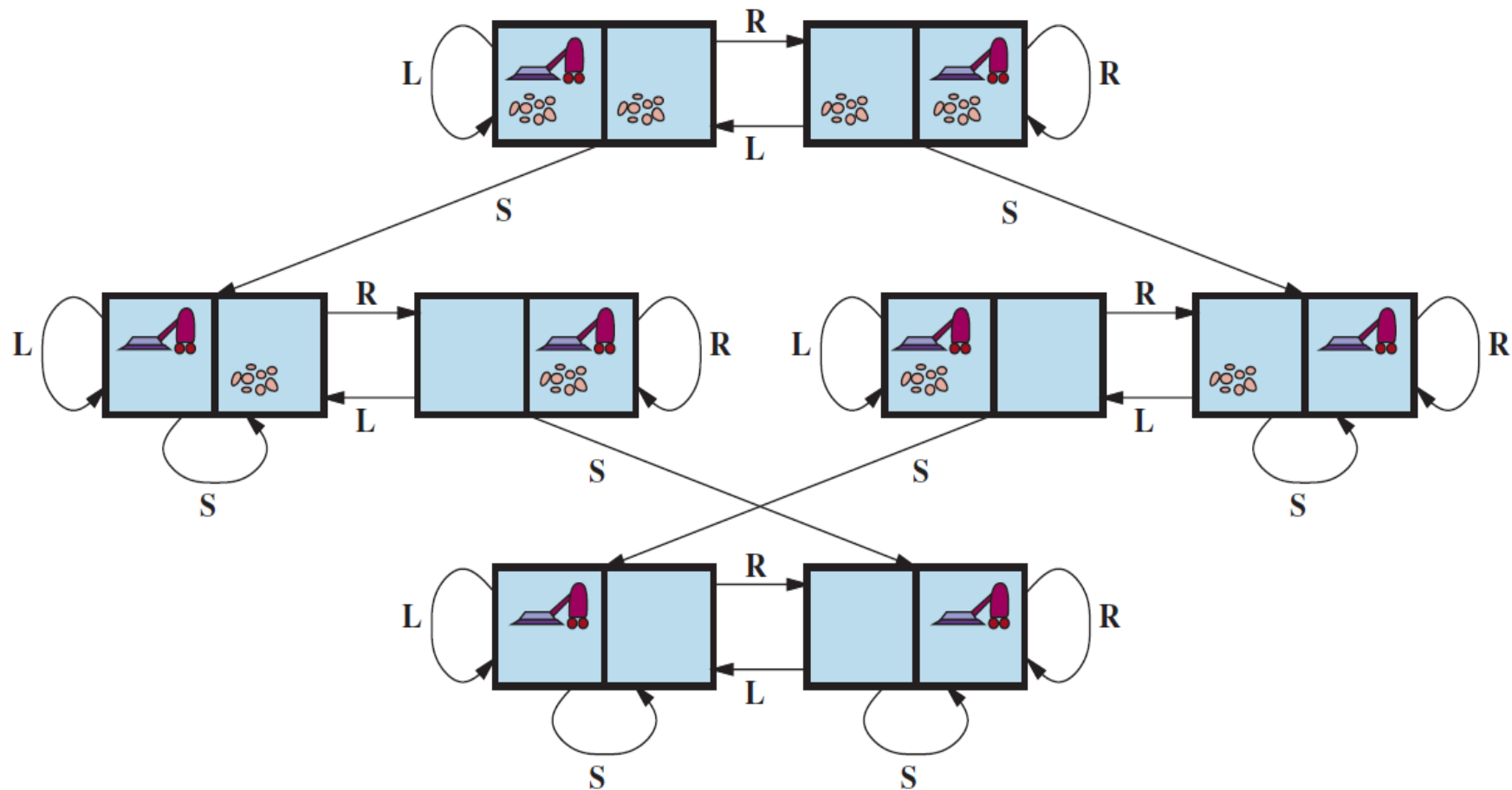
- **Actions:**

- Left, right, suck.

- **Transition Model .. ?**

Example: Vacuum World State Space Graph

○ Transition Model:



Example: the 8-Puzzle

○ States

○ Locations of tiles

- 8-puzzle: 181,440 states ($9!/2$)
- 15-puzzle: ~10 trillion states
- 24-puzzle: $\sim 10^{25}$ states

○ Actions

- Move blank left, right, up, down

○ Path Cost

- 1 per move

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

State Space Search Strategies

- There are two distinct ways for searching a state space graph:
 - **Data-Driven Search: (Forward chaining)**
 - Start searching from the given data of a problem instance toward a goal.
 - **Goal-Driven Search: (Backward chaining)**
 - Start searching from a goal state to facts or data of the given problem.

State Space Search **Strategies** ..

Selecting Search Strategy

- **Data-Driven Search** is suggested if:
 - The data are given in the initial problem statement.
 - There are few ways to use the given facts.
 - There are large number of potential goals.
 - It is difficult to form a goal or hypothesis.
- **Goal- Driven Search** is appropriate if:
 - A goal is given in the problem statement or can easily formulated.
 - There are large number of rules to produce a new facts.
 - Problem data are not given but acquired by the problem solver.

Additional Resources

- Euler Paths & the 7 Bridges of Konigsberg | Graph Theory

<https://www.youtube.com/watch?v=dSK5jTEe-AM&t=307s>

- Introduction to Traveling Sales Man Problem (TSP) n why it is NP Hard

<https://www.youtube.com/watch?v=IBGiOsF7WWY>

Exercises

- **Search Space Exercise:** Consider a 3x3 grid representing a puzzle where each cell can be either empty or occupied. Define the search space for finding a valid configuration of the puzzle.
- **Exhaustive Search Exercise:** Consider a set of numbers {2, 4, 6, 8, 10}. Using exhaustive search, find all possible subsets of numbers that sum up to a target value of 12. Iterate through all possible combinations of numbers and check if their sum equals the target value. List all subsets that satisfy this condition.



مجموعة جامعة
الجميع الذكية
للتعليم القابضة



Thank You

