Lecture 2: Search: Problem Formulation

Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

https://taohuang.info/cs3317

https://oc.sjtu.edu.cn/courses/89538

Why Problem Formulation Matters

- Same task → different formulations
- Difficulty can change **dramatically**

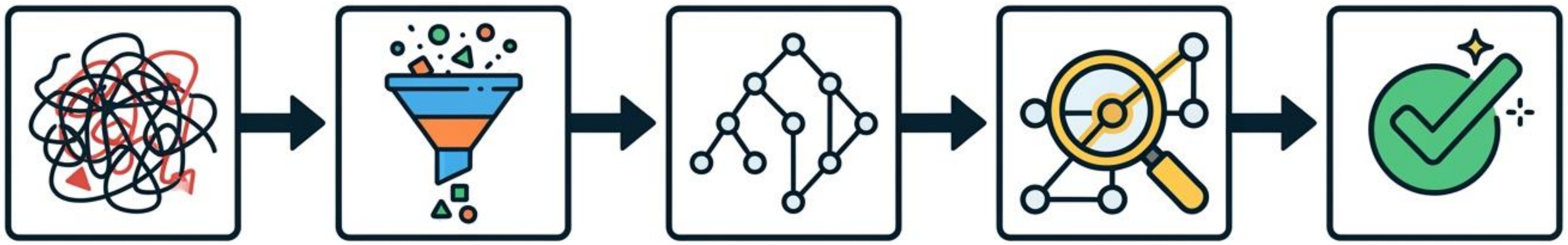
Examples:

Amap navigation   Chess planning Robot navigation

"If we want an AI to solve Sudoku（数独）, what information must the AI know?"

From Real Problem → Search Problem

- Real-world problems are messy
- AI simplifies them into: states, actions, goals
- Search finds: *sequence of actions* → goal

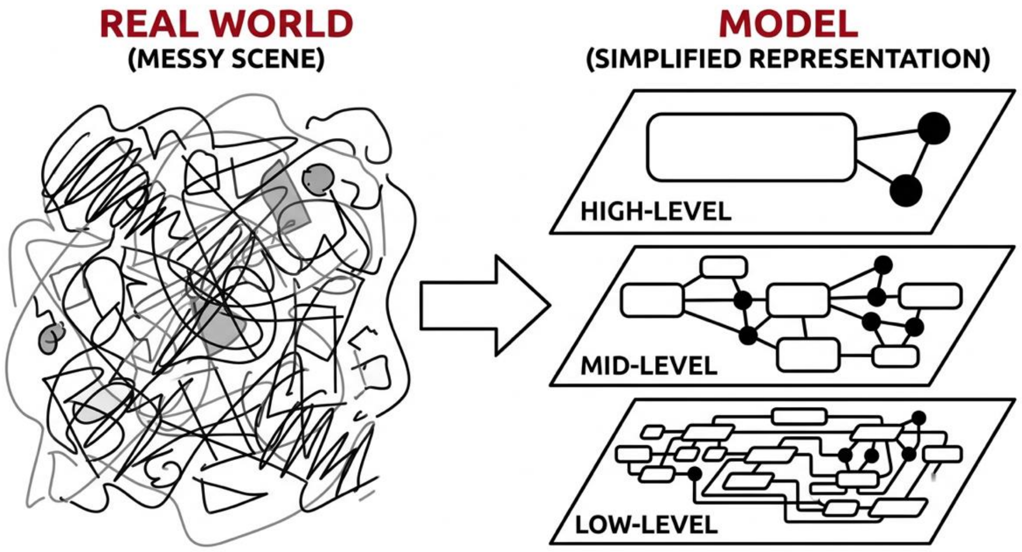


Levels of Abstraction

A **model** is an abstraction.

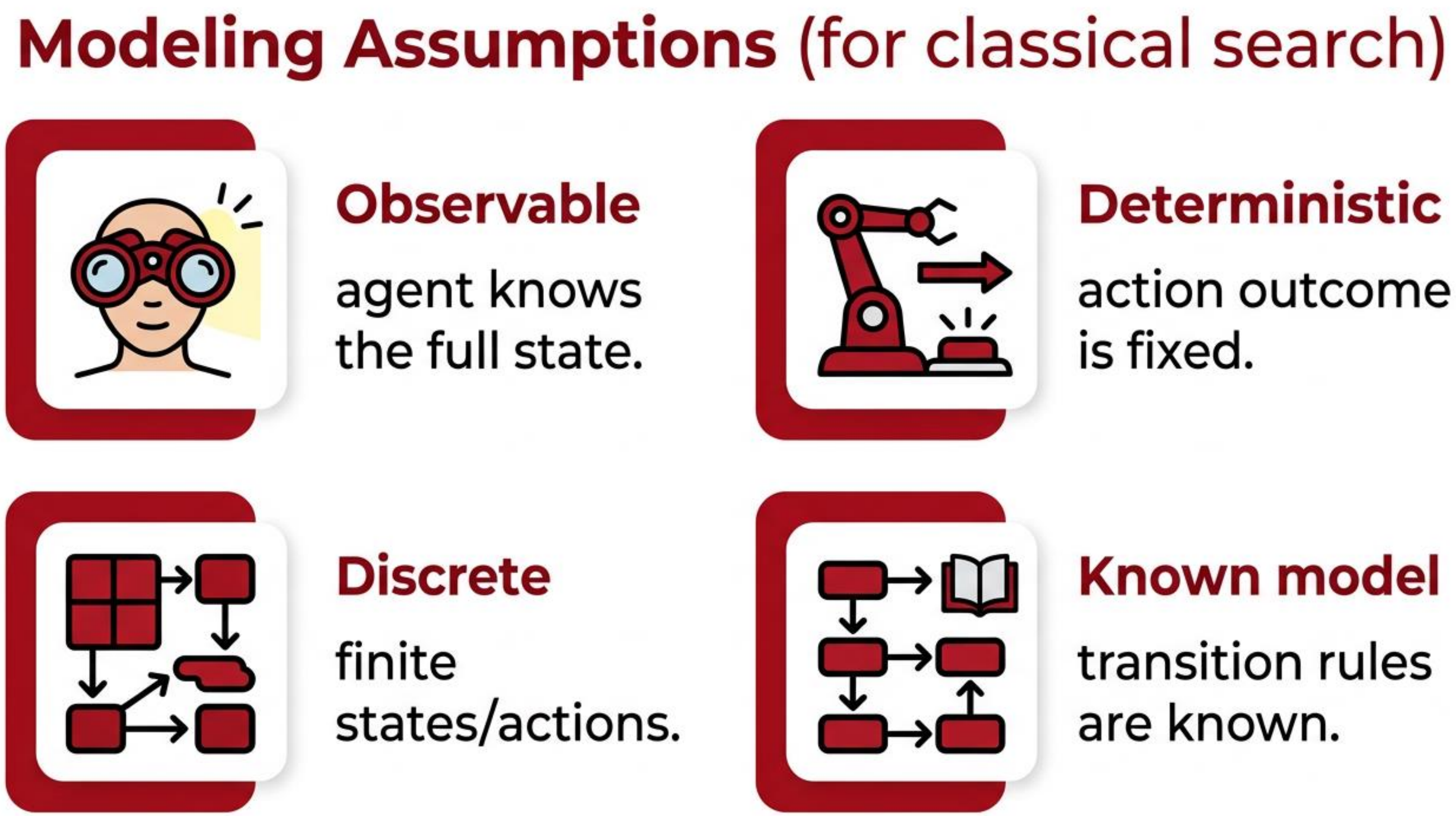
We strip away irrelevant details:

- Weather in route planning
- Material of puzzle tiles
- Color of chess pieces



Modeling Assumptions

Classic AI search assumes a **perfect world** model:



Key Elements of Problem Formulation

A search problem defines:

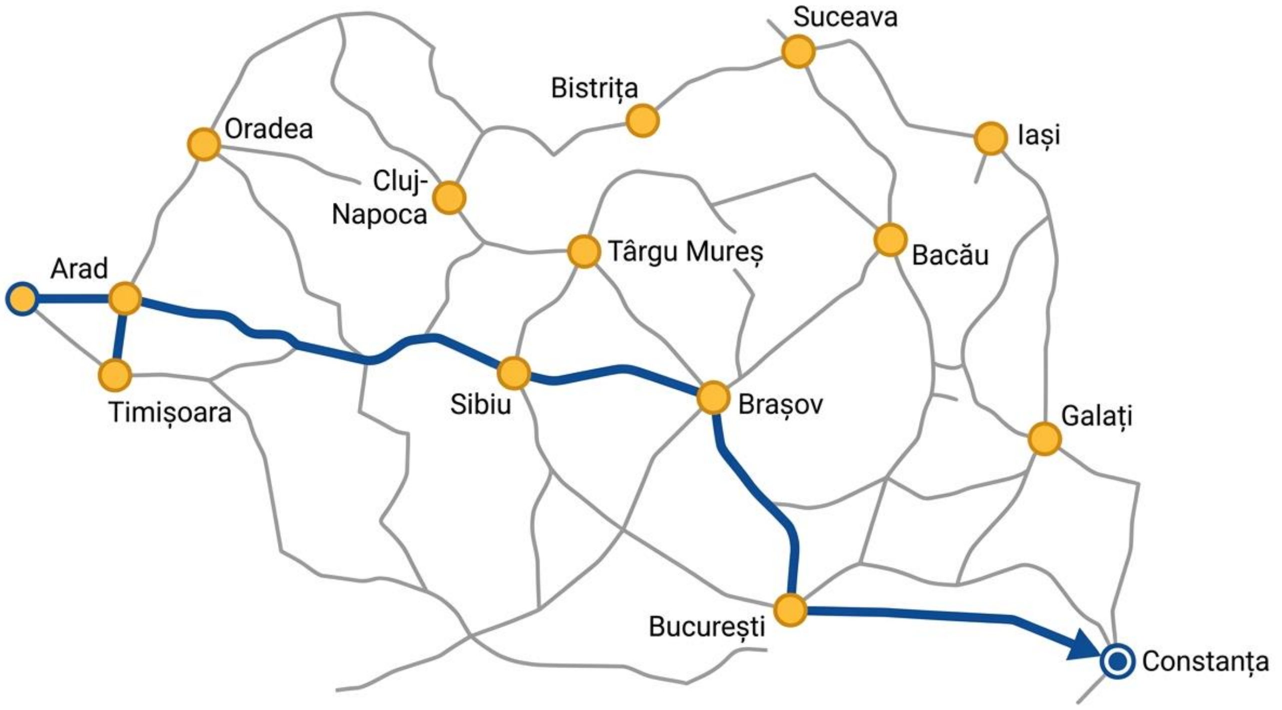
- **State space:** Set of all possible states
- **Initial state:** Where the agent begins
- **Actions:** Operators available at each state
- **Transition model:** Result of each action
- **Goal test:** Criteria for completion
- **Path cost:** Numerical value (e.g., distance)

Together, these define the search space.

Example — Route Planning

Find: Arad → Constanta

- **States:** Cities
- **Actions:** Drive to neighbc
- **Goal:** reach Constanta
- **Cost:** Road distance



Travelling in Romania

State Space

A **state** represents a configuration of the world.

Examples: Chess (Board), Route (City), Robot (Grid location).

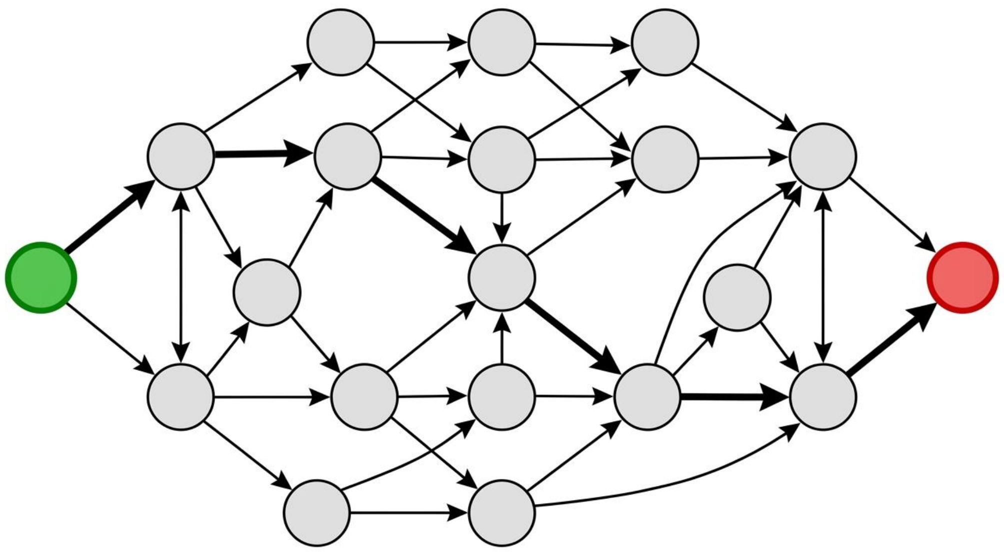
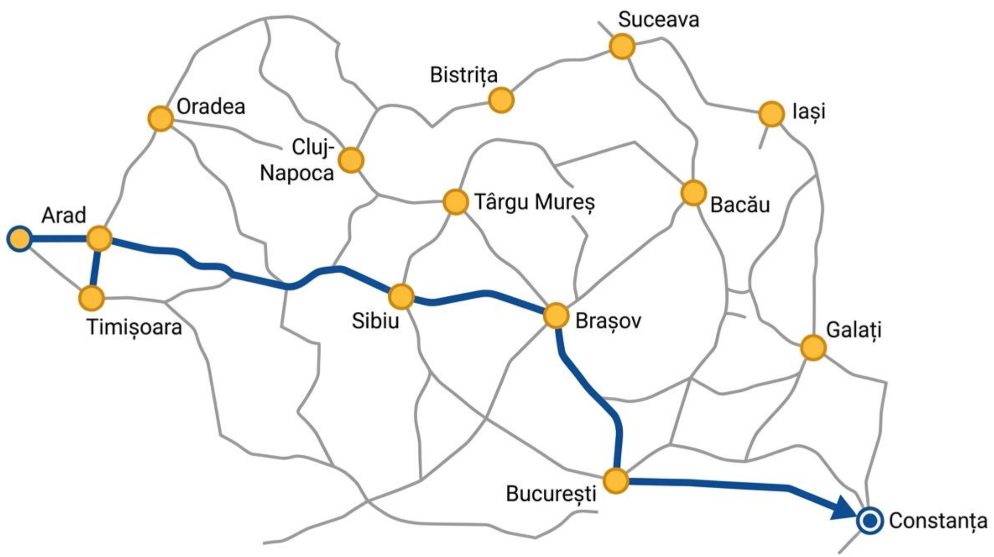


Illustration of a state space.

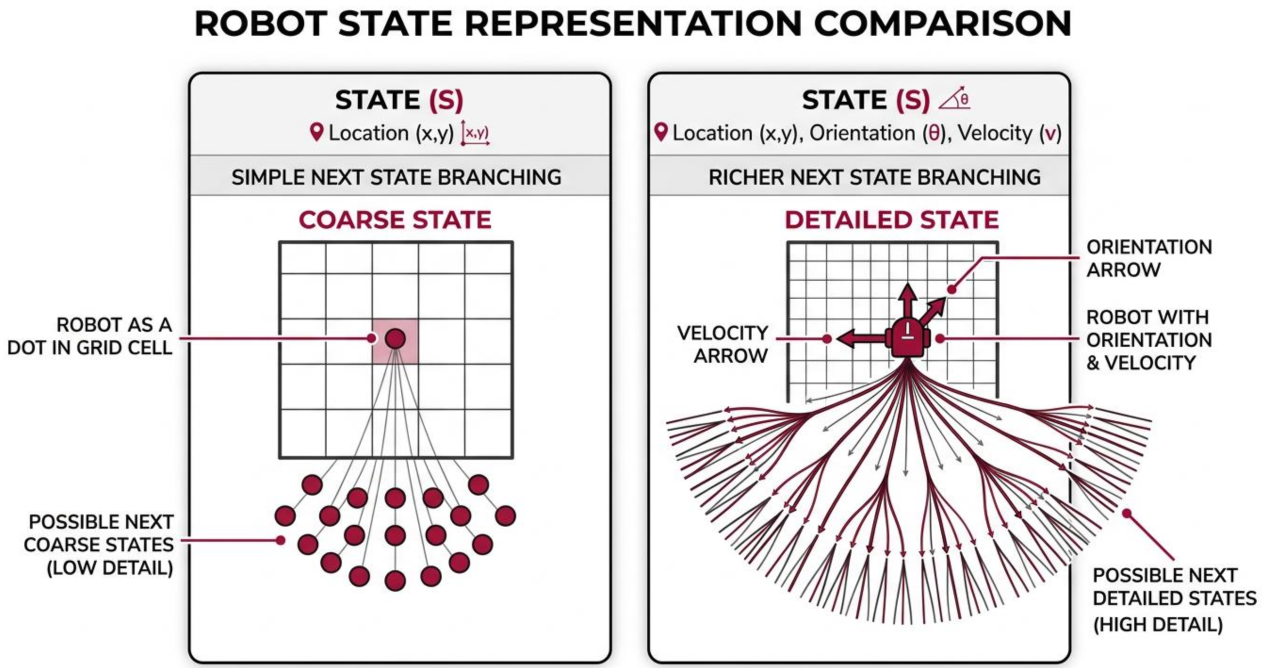


Granularity of States

How much detail is needed?

- **Coarse:** (x, y) coordinates
- **Fine:** (x, y, orientation, velocity)

Finer detail ➔ exponentially larger state space.

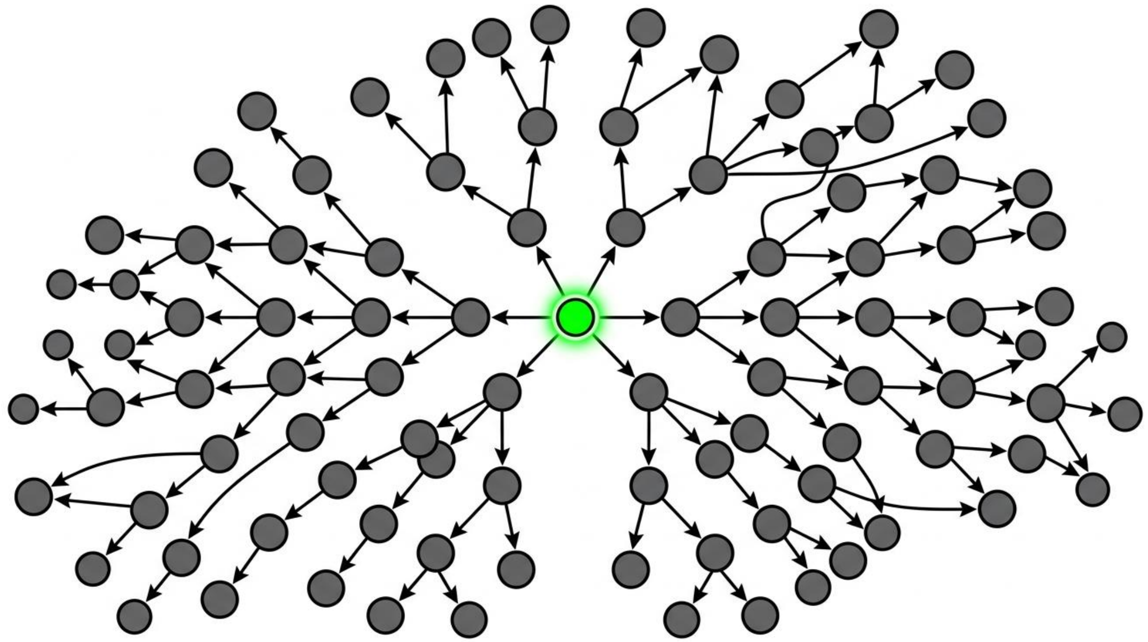


Initial State

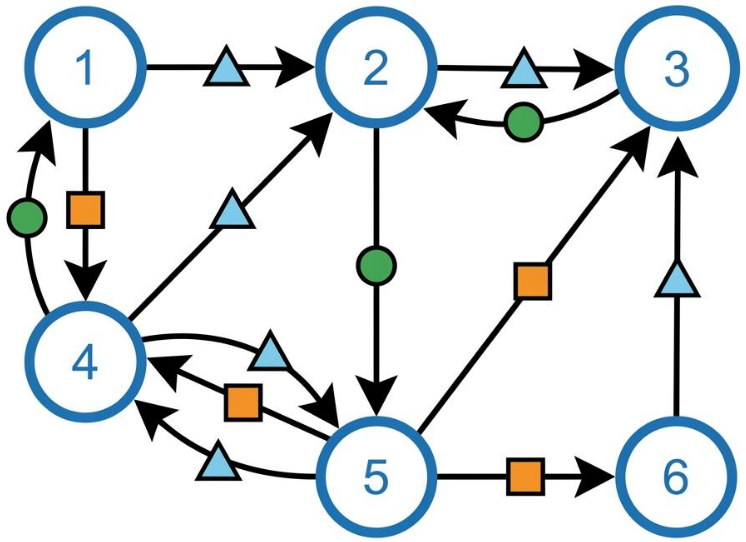
Starting configuration of the world.

- Route: Arad
- Chess: Start board
- Robot: Start point

Search begins here!



Actions



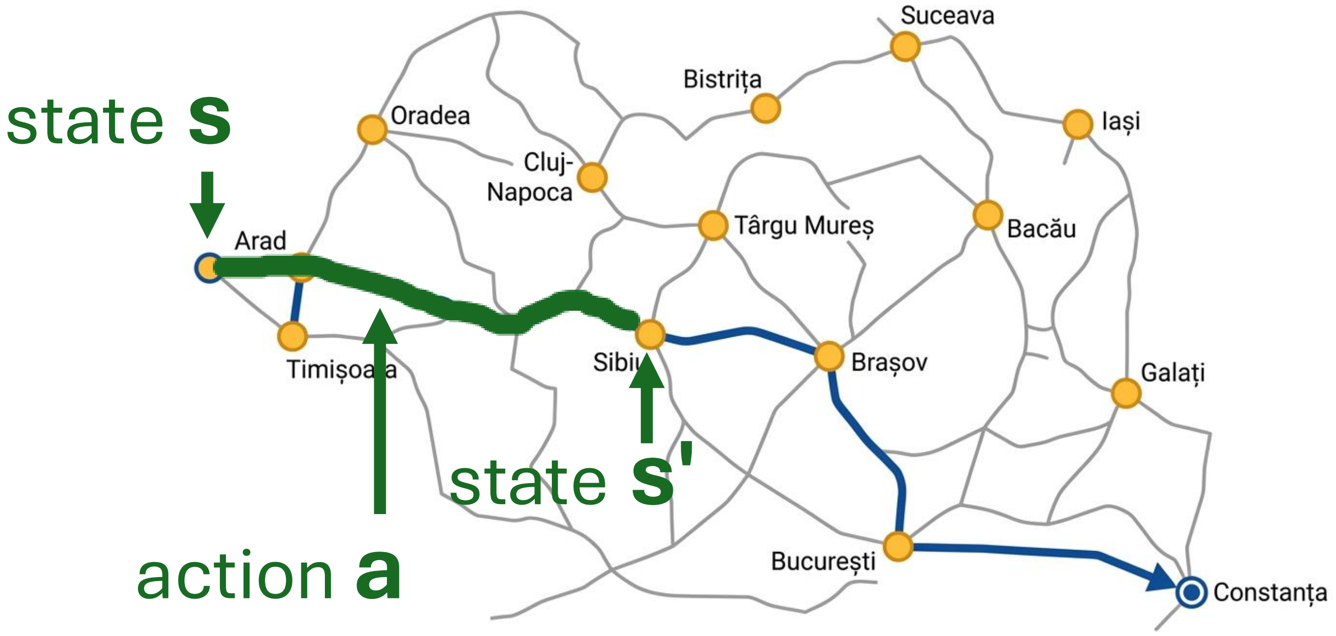
Operators that change states.

- Route: Drive (Sibiu)
- Chess: Move (Nf3)
- Puzzle: Slide (Tile 4)

Transition Model

Defines the result of each action:

Result(s, a) → s'



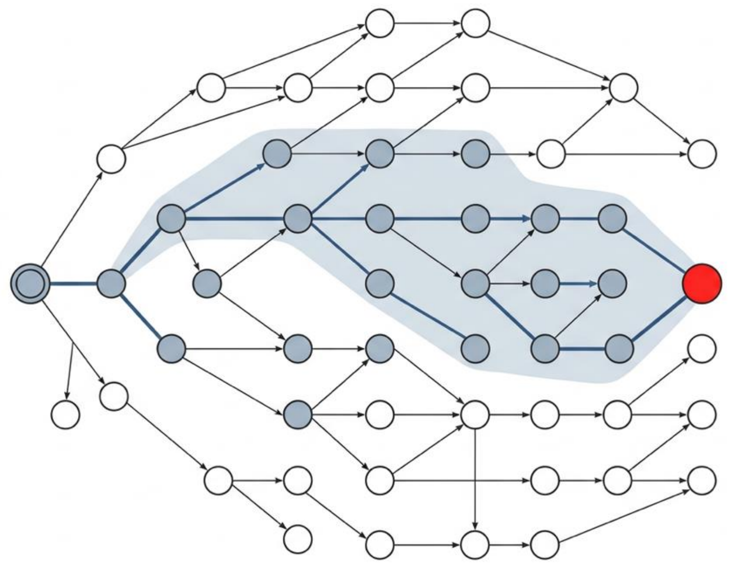
Example: Result(Arad, Drive_to_Sibiu) = Sibiu

Goal Test

Checks if goal is reached.

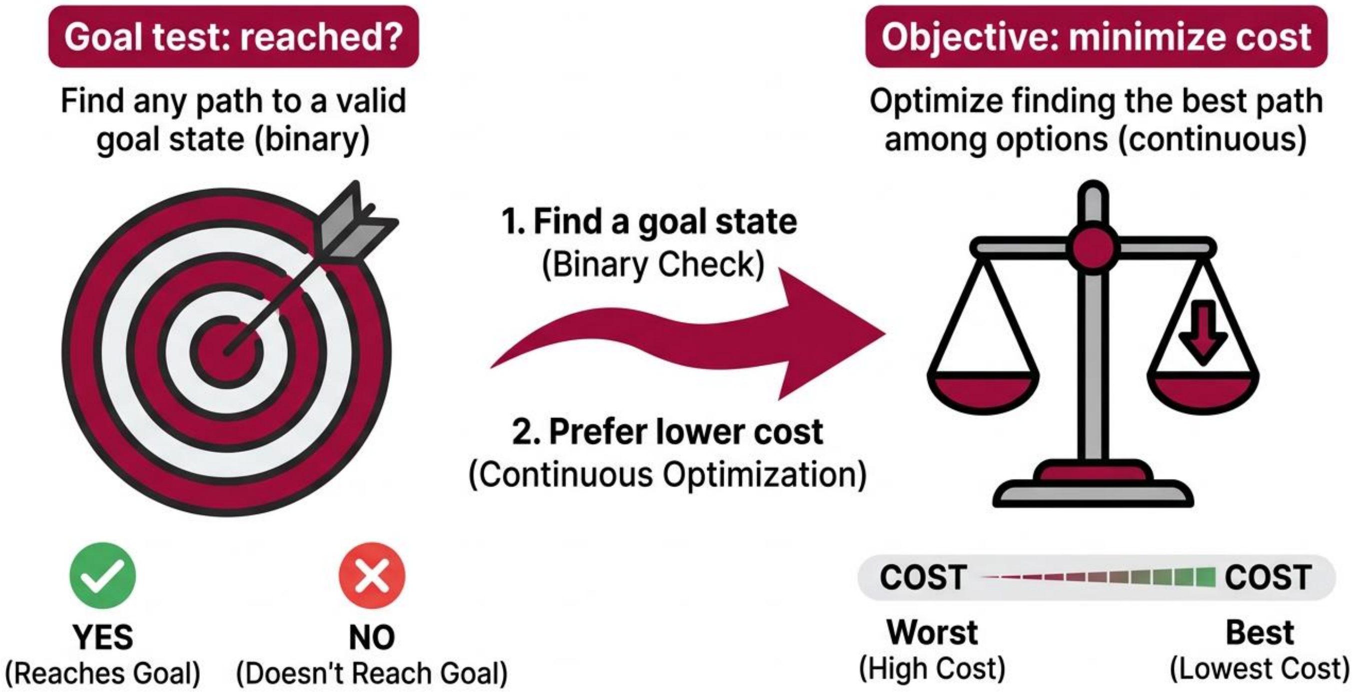
- Route: city == Constanta?
- Chess: checkmate?
- Puzzle: ordered grid?

Search stops at SUCCESS.



Goal Test vs. Objective

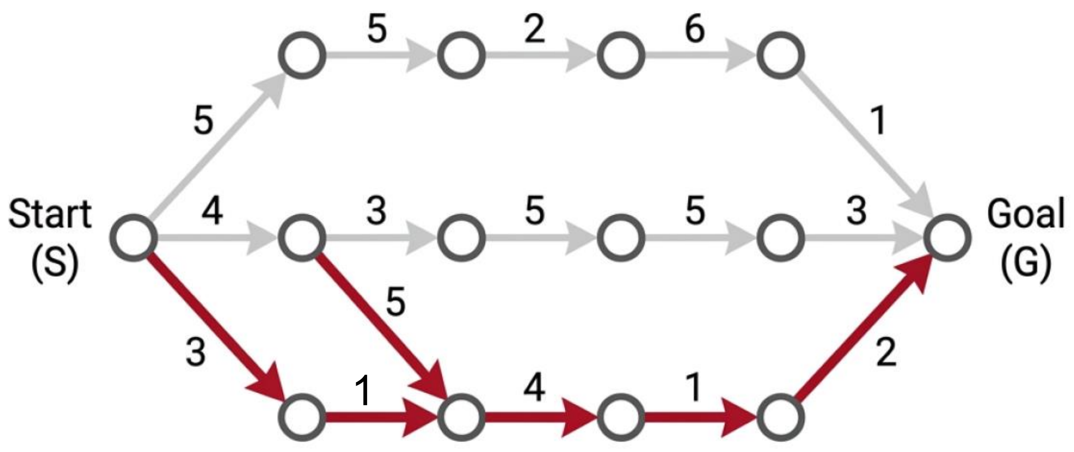
- **Goal Test:** "Are we there yet?" (Binary: Yes/No)
- **Objective:** "How good is this path?" (Optimization: Minimize cost)



Path Cost

Actions are rarely free. We assign a **cost**.

- **Route:** distance / time
- **Robot:** energy consumption
- **Game:** number of moves



Objective: Minimize Total Cost

Different paths yield different costs.

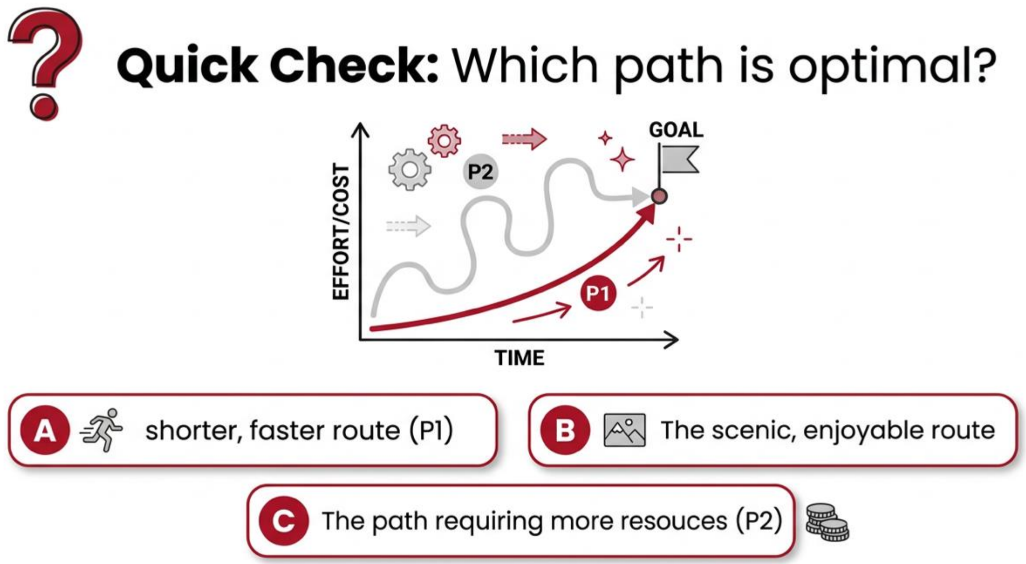
Path A cost = 5 + 5 + 2 + 6 + 1 =19

Path B cost = 4 + 3 + 5 + 5 + 3 = 20

Path C cost = 4 + 5 + 4 + 1 + 2 =16

Path D cost = 3 + 1 + 4 + 1 + 2 =11

Quick Check: Optimal Path



If Path A cost = 19, Path B cost = 20, Path C cost = 16...

Which one does a search algorithm prefer?

Think about 'Optimal Solution' definition.

Search Problem Definition

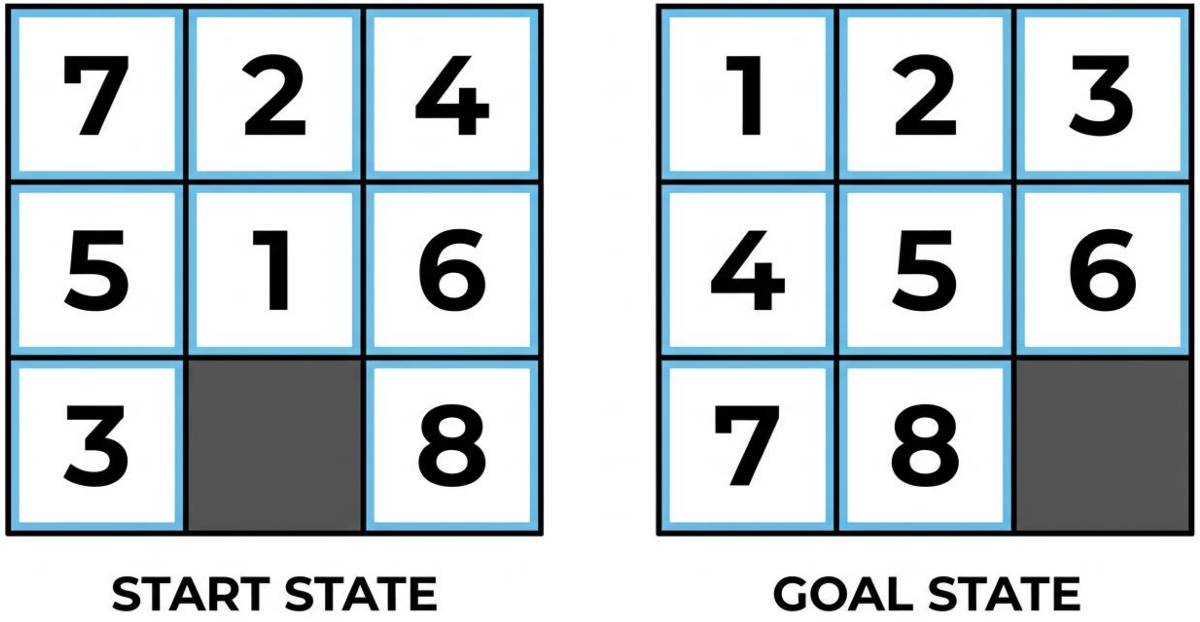
A search problem is a 6-tuple:

(S, S_0, A, T, G, C)

These components define the entire state-space graph.

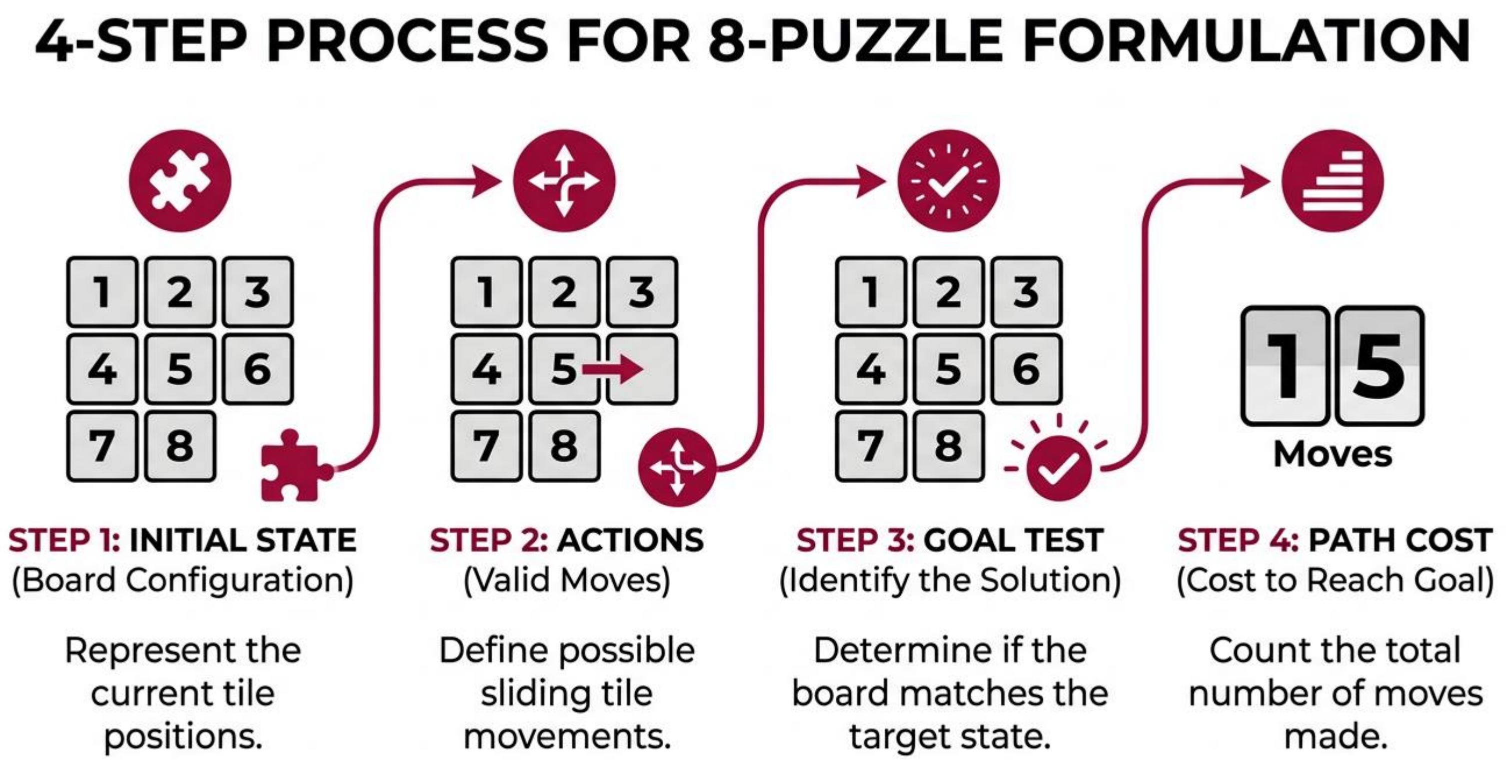
Example — 8 Puzzle

- **Goal:** Arrange numbers (1-8) in correct order
- **Action:** Slide tile into the adjacent empty space



Formulating the 8-Puzzle

1. **States:** Board configs · 2. **Actions:** Slide tile · 3. **Goal:** Order grid



State Space of 8-Puzzle

State = Any unique board configuration

Total possible states:

9! = 362,880

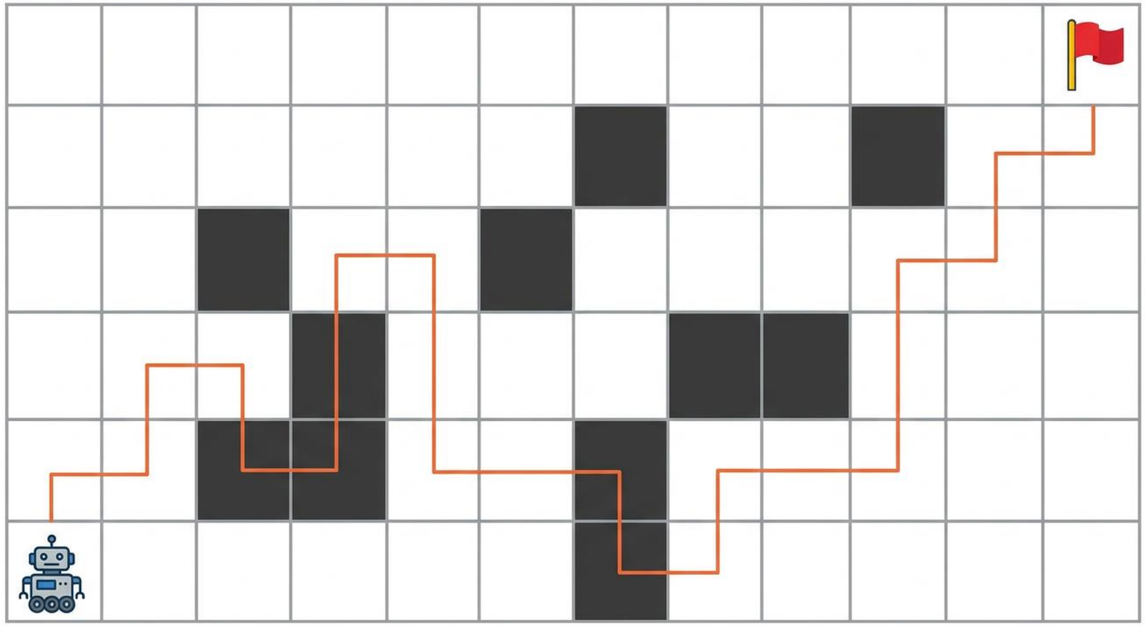
Note: Only half of these are reachable from any given state.

Robot Navigation

Robot moving in a grid world:

- **Actions:** Up, Down, Left, Right
- **Goal:** reach target cell

Obstacles block movement.

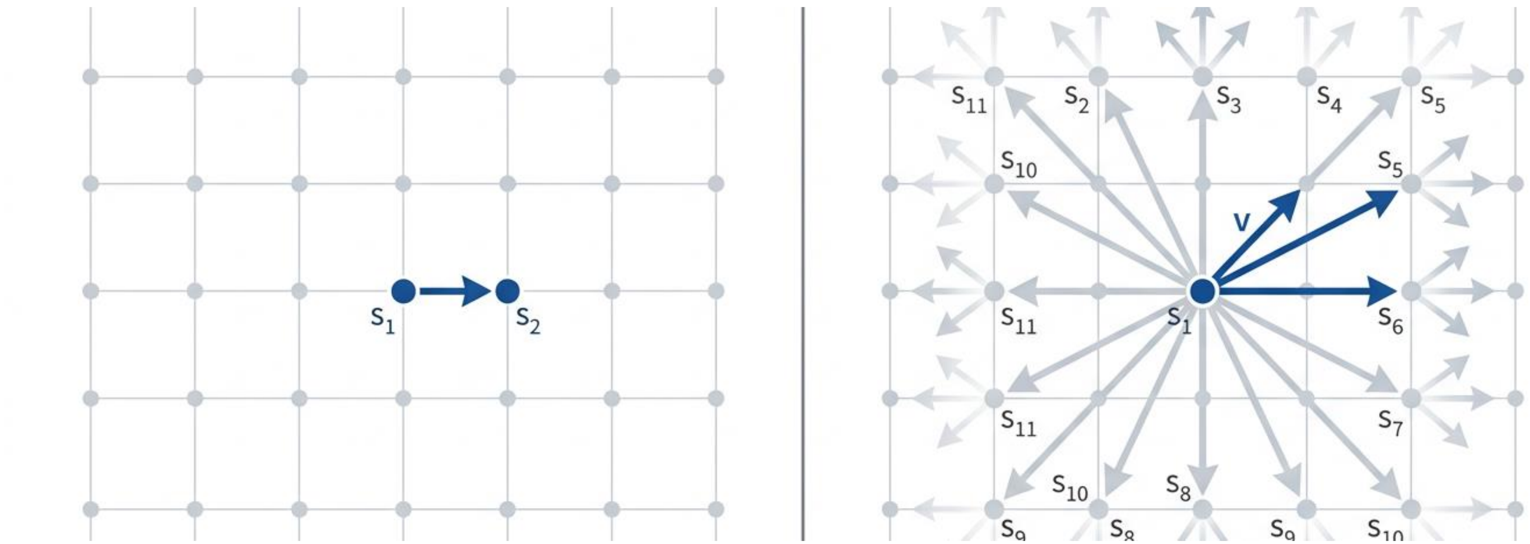


Formulation Matters

Example: Robot navigation

Option 1: State = (x, y) → *Compact*

Option 2: State = (x, y, dx, dy) → *Much larger space*

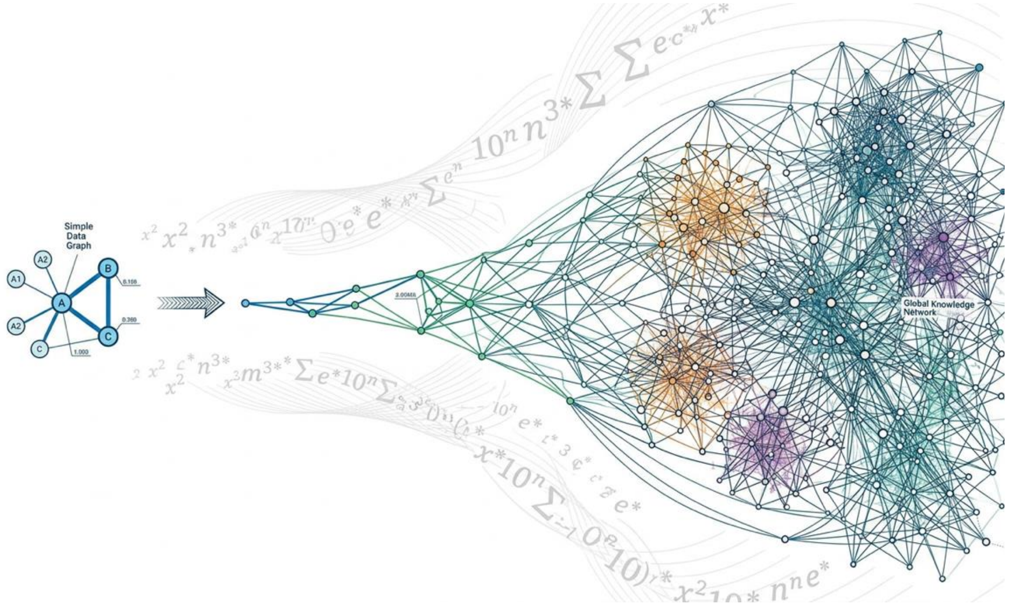


State Space Explosion

Complex problems have massive spaces:

- **Chess:** ~ 10^{120} possible games
- **Go:** ~ 10^{170} states

We need efficient search & heuristics!



Exercise: Sudoku（数独） Formulation

Formulate **Sudoku** as a search problem:

1. What is the state?
2. What are the actions?
3. What is the goal?

Take 2–3 minutes to discuss with your neighbor.

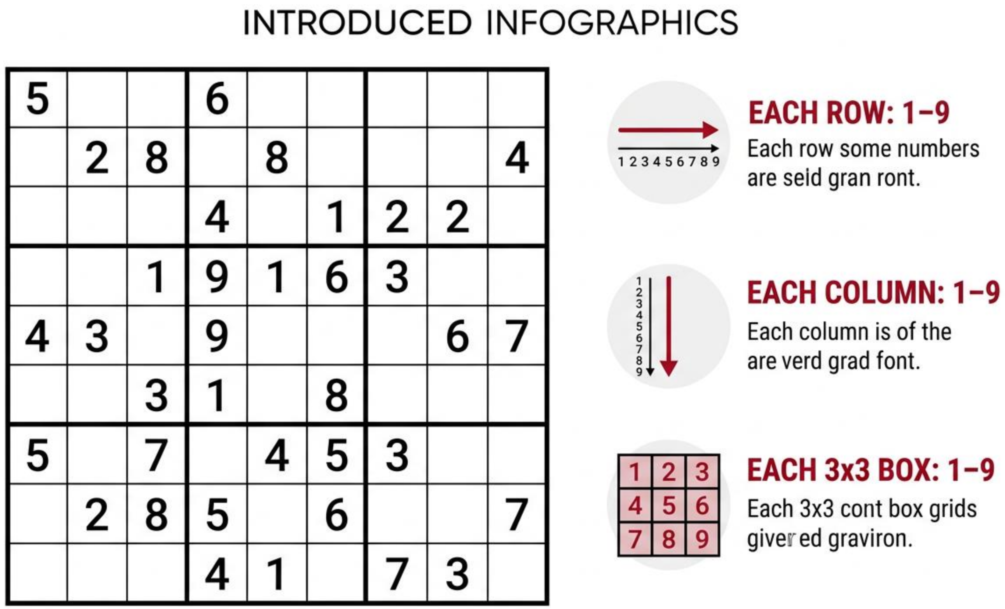
What is Sudoku?

The Puzzle:

A 9x9 grid divided into nine 3x3 subgrids.

The Rules:

- Fill cells with digits **1 to 9**
- **No repetition** in any row
- **No repetition** in any column
- **No repetition** in any 3x3 box



One Possible Formulation

- **State:** Partially filled 9x9 grid
- **Action:** Fill a valid number in an empty cell
- **Goal:** Valid completed grid

			5	3	4			
6	7	2			7			
			6			1	9	5
	9	8					6	
8				6				3
4			8		3		1	
7				2				6
	6					2	8	

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	6	7	8	6	4	7	3
7	9	8	4	2	7	5	3	1
9	2	3	3	5	1	7	4	6
8	5	7	7	4	9	6	8	3
4	6	9	8	7	3	4	1	2
7	3	4	9	2		9	5	6
5	6	7	4	1	9	2	8	9

Key Takeaways

- Problem formulation **defines the search graph**
- Choice of states & actions affects **complexity**
- A good formulation makes the problem **tractable**

Next Lecture

Uninformed Search Algorithms

BFS · DFS · Uniform Cost Search