

# Problem-solving Agents:

## *2.1 Formulating Problems*



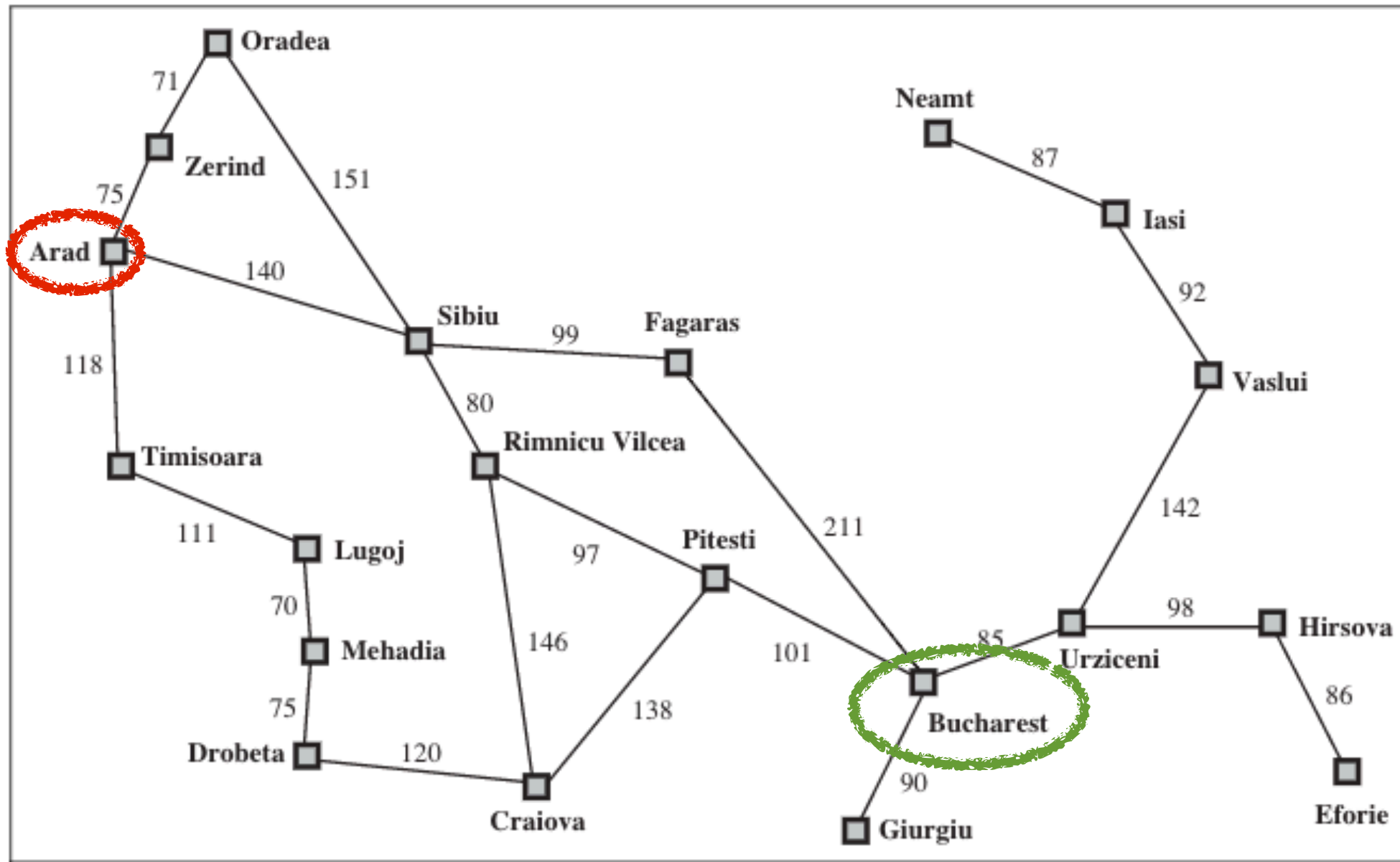
# Outline

- Goal and Problem formulation
- Search
- Well-defined Problems
- Example Problems

# Goals & Problem Formulation

- **State** – a representation of a physical configuration; a **snapshot** of the world at a given time step
- A state is a summary of all past actions sufficient to choose future actions optimally
- **Goal** – a set of world states, exactly in which the goal is satisfied
  - Be in Busan
- **Problem formulation** – process of deciding what actions and states to consider, given a goal
  - Action of driving from one city to another

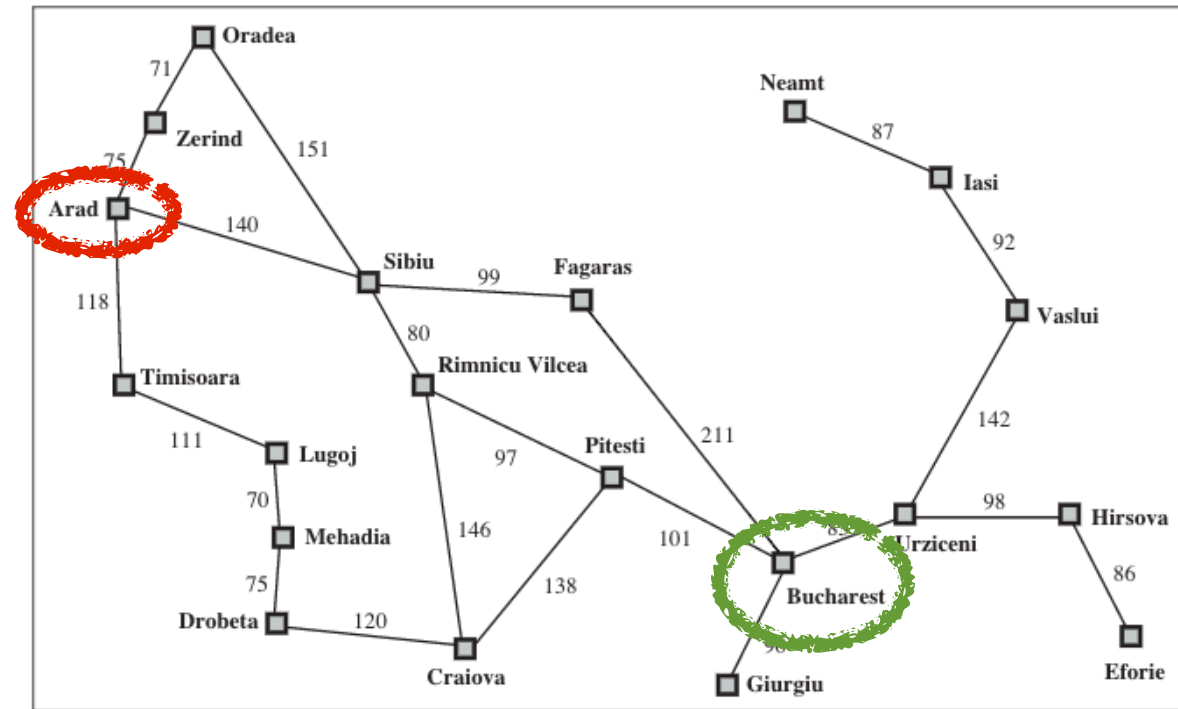
# Romania



**Initial state:** current location is Arad

# Problem-solving Sequence

- Initial state:
    - Arad
  - Formulate goal:
    - Be in Bucharest
  - Formulate problem:
    - states: cities
    - actions: drive between cities
  - Formulate solution:
    - Sequence of cities: Arad, Sibiu, Fagaras, Bucharest
- ➡ Solution is a **fixed sequence of actions** if environment is **observable, discrete, known(map), deterministic**



# Search

- The process of looking for a **sequence of actions** that reaches the **goal**
- A search algorithm takes a problem as input and returns a solution in the form of an action sequence
- Once a solution is found, the sequence of actions can be carried out or executed
- formulate → search → execute
- More details on search in the next few lectures

# Well-defined Problems

- Goal test – check that a given (current) *state* is the goal state, e.g. *In(Bucharest)*
- Sometimes goal is an *abstract property*, e.g. “checkmate” in Chess
- **Transition model** is a description of what each action does – *Result(s,a)* returns the **successor state** from doing action *a* in state *s*  
$$\text{Result}(\text{In}(\text{Arad}), \text{Go}(\text{Zerind})) = \text{In}(\text{Zerind})$$
- **Path cost function** assigns *numeric* cost to each path, the agent chooses a cost function that reflects its own performance measure, e.g. in vacuum world: the number of steps in the path
- An **optimal solution** is a solution (action sequence) that has the lowest path cost among all solutions

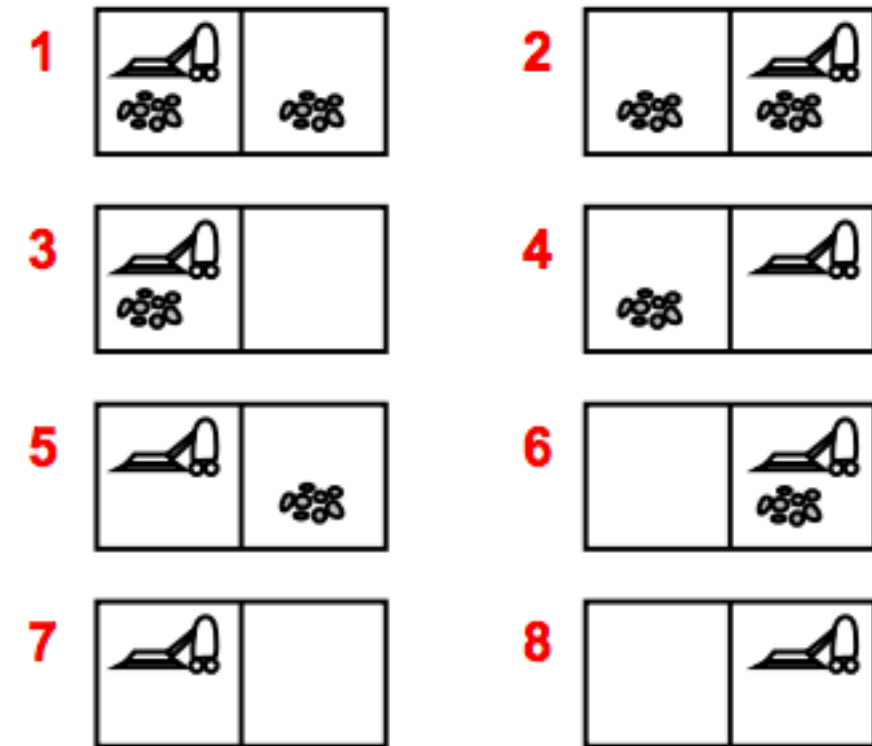
# Problem Types

- Deterministic, fully observable → **single-state** problem
  - agent knows exactly which state it will be in
  - solution is a sequence
- Non-observable → **conformant** problem
  - agent may have no idea where it is
  - solution (if any) is a sequence
- Nondeterministic and/or partially observable → **contingency** problem
  - percepts provide **new information** about current state
  - solution is contingent plan or policy, often interleave search and execution
- Unknown state space → exploration problem (“online”)



# Vacuum World Problem Types

- **Single-state**: start in #5.  
Solution?  
– *[Right, Suck]*
- **Conformant**: start anywhere  
{1, 2, 3, 4, 5,... , 8}.  
Solution?  
– *[Right, Suck, Left, Suck]*
- **Contingency**: start in #5,  
Suck can make a clean  
carpet dirty. Solution?  
– *[Right, if Dirty then Suck]*



All the possible states for the  
vacuum world problem with  
2 squares

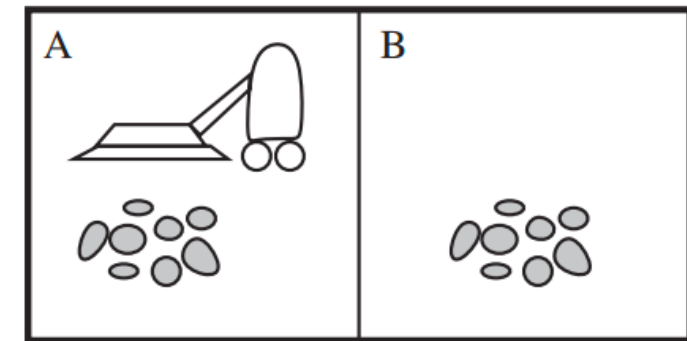
# Single-state Problem Formulation

- Problem definition
  1. **Initial State**, e.g. In Arad, In state #5
  2. **Successor function**,  $S(x)$  = set of action-state pairs, e.g.  $S(Arad) = \{ \langle Arad \rightarrow Zerind, Zerind \rangle, \dots \}$
  3. **Goal test**, can be explicit e.g. “In Bucharest” or implicit e.g.  $NoDirt(x)$
  4. **Path cost** (additive), e.g. sum of distances, number of actions executed, etc
- **Solution** is a sequence of actions leading from the initial state to the goal state

# Example Problems

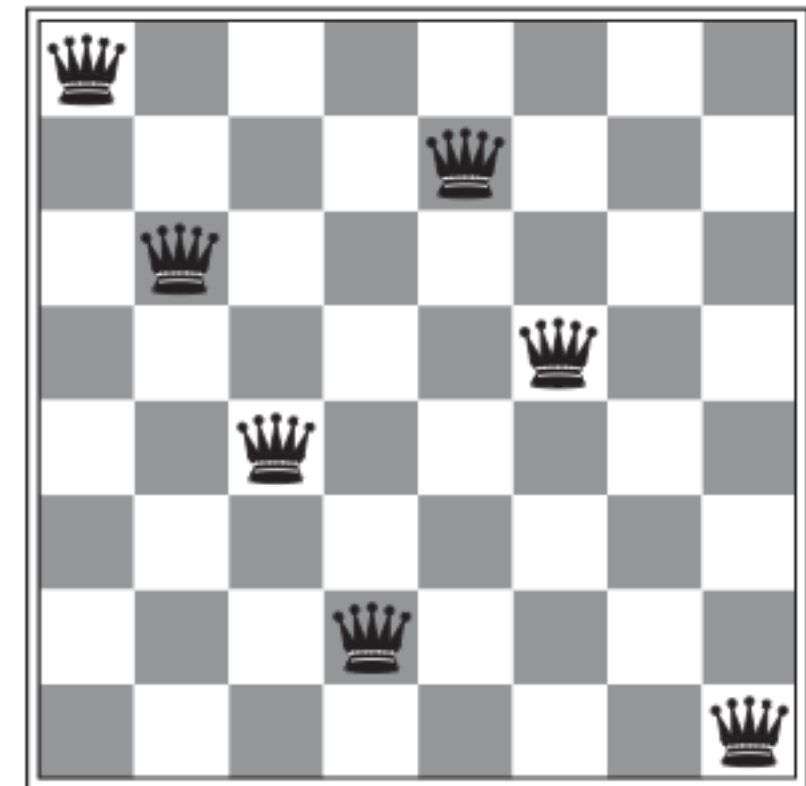
# Toy Problem – Vacuum World

- **States** agent location, dirt locations. Agent can be in one of 2 locations and each location may contain dirt.  $n \times 2^n = 2 \times 2^n = 8$  possible states
- **Initial state** Any one of the states
- **Actions** *Left, Right, Suck, NoOp*
- **Transition model** Actions should have their expected effects, except for moving left from the leftmost square, and right from the rightmost square. Sucking in a clean square has no effect.
- **Goal test** Check that all squares are clean
- **Path cost** Each step taken costs 1, so path cost is the number of steps in the path OR 1 per action (0 for *NoOp*)



# Toy problem – 8-queens

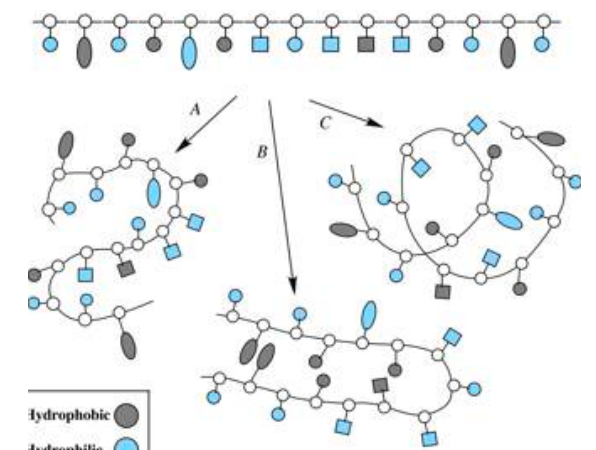
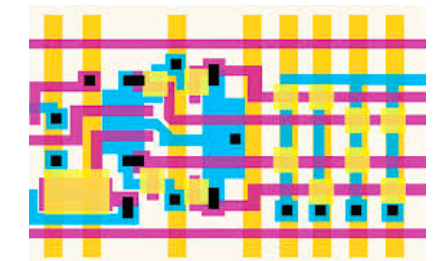
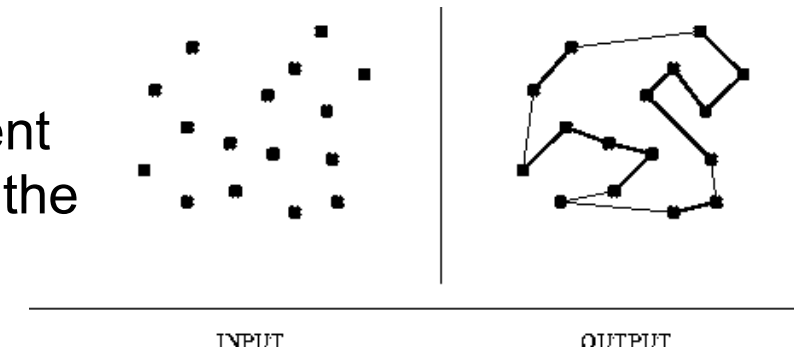
- **States** any arrangement of 0 to 8 queens on the board
  - **Initial state** no queens on the board
  - **Actions** add a queen to an empty square
  - **Transition model** returns the board with a queen added to the specified square
  - **Goal test** 8 queens on the board, none attacked
  - Improved formulation:
    - ◉ **States** all possible arrangements of  $n$  queens ( $0 \leq n \leq 8$ ), one per column in the leftmost  $n$  columns, with no queen attacking another
    - ◉ **Actions** add a queen to any square in the leftmost empty column such that it is not attacked by any other queen
- ➔ Reduces state space from  $1.8 \times 10^{14}$  to just 2,057



**Goal:** to place 8 queens on a chessboard such that no queen attacks any other. (A queen attacks any piece in the same row, column or diagonal)

# Real World Problems

- **Route-finding problem** – airline travel-planning
- **Touring problems** – closely related to route-finding with different state space; each state must include the current location and the set of cities the agent has visited.
  - Travelling salesman problem (TSP) visit each city exactly once
- **VLSI** (Very large-scale integration) layout – positioning millions of components and connections on a chip to minimise area, circuit delays, minimise stray capacitances and maximise manufacturing yields. Split into cell layout and channel routing
- **Robot navigation** – generalisation of route-finding problem; robot can move in a continuous space with an infinite set of possible actions and states
- **Automatic assembly sequencing & protein design** – aims are to find an order in which to assemble the parts of some object & to find a sequence of amino acids that will fold into a three-dimensional protein with the right properties to cure some disease



# Exercise

- \* Define the 8-puzzle game in terms of states, actions, goal test and path cost

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

- \* states??
- \* actions??
- \* goal test??
- \* path cost??

# Summary

- Looked at problem-solving agent's goals, states and solution
- Defined the problem better with **initial states**, **transition models** and **path cost** for optimal solutions
- Introduced the notion of **search** as the process of get from an initial state to the goal state
- Different types of problems
  - Toy and real world examples



# References

- Russel and Norvig, Chapter 3.1–3.2