Problem solving

Chapter 3, Sections 1–3

Problem types

Deterministic, fully observable $\Longrightarrow$ single-state problem

Agent knows exactly which state it will be in; solution is a sequence

Non-observable $\Longrightarrow$ conformant problem

Agent may have no idea where it is; solution (if any) is a sequence

Nondeterministic and/or partially observable $\Longrightarrow$ contingency problem

percepts provide **new** information about current state

solution is a contingent plan or a policy

often **interleave** search, execution

Unknown state space $\Longrightarrow$ exploration problem ("online")

Example: Romania

We are on holiday in Romania; currently in Arad
Our flight leaves tomorrow from Bucharest

Formulate goal:

be in Bucharest

Formulate problem:

states: various cities

actions: drive between cities

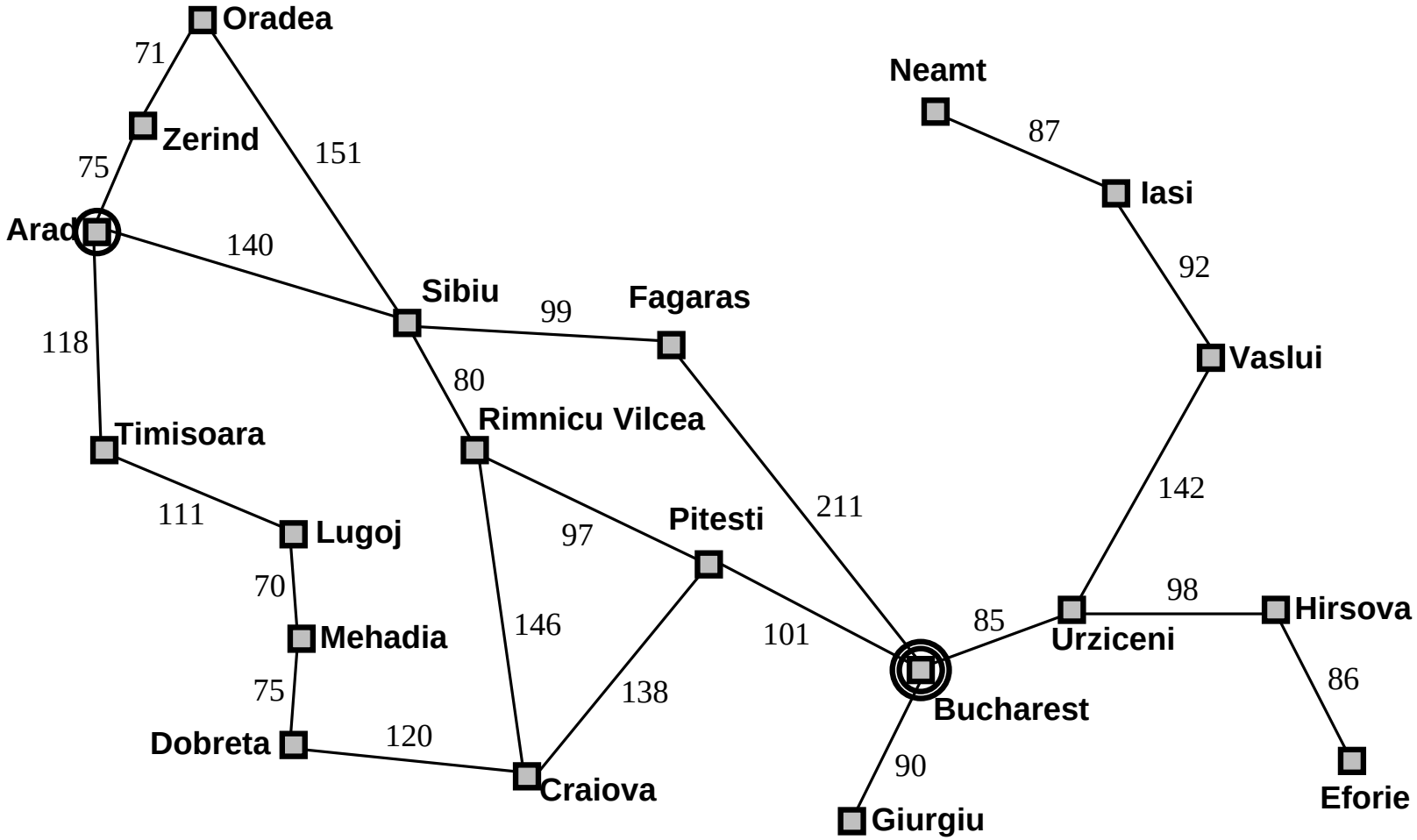
Find solution:

sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

Problem type:

deterministic, fully observable $\Longrightarrow$ single-state problem

Example: Romania



Single-state problem formulation

A problem is defined by five components:

initial state, e.g., $In(Arad)$

actions $\textsc{Actions}(s)$ = set of actions for state s
e.g., $\textsc{Actions}(In(Arad)) = \{Go(Sibiu), Go(Timisoara), Go(Zerind)\}$

transitions $\textsc{Result}(s, a)$ = the successor state
e.g., $\textsc{Result}(In(Arad), Go(Zerind)) = In(Zerind)$

goal test, can be an explicit set of states, e.g., $\{In(Bucharest)\}$
or an implicit property, e.g., **checkmate** in chess

path cost is the sum of the step costs $c(s, a, s')$
e.g., sum of distances, number of actions executed, etc.
in this chapter we assume c to be positive

A solution is a sequence of actions
leading from the initial state to a goal state

Selecting a state space

Real world is absurdly complex

$\Rightarrow$ state space must be **abstracted** for problem solving

(Abstract) state = set of real states

(Abstract) action = complex combination of real actions

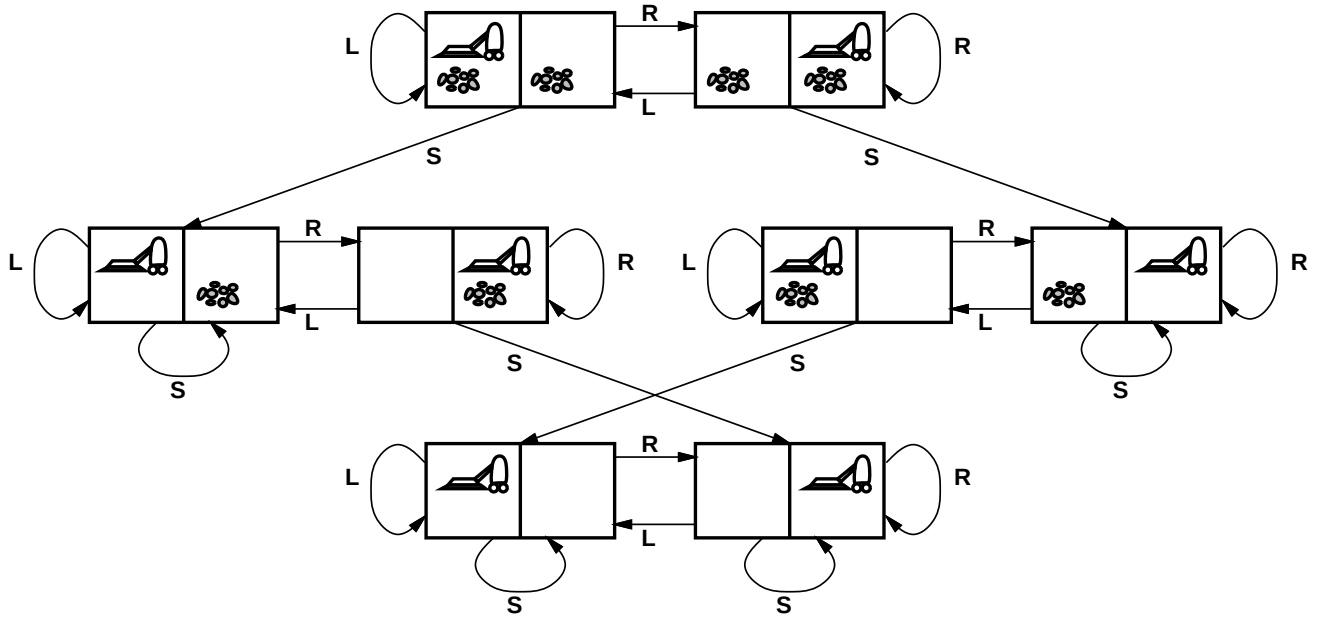
e.g., $\textsc{Result}(In(Arad), Go(Zerind))$ represents a complex set of possible routes, detours, rest stops, etc.

(Abstract) solution =

set of real paths that are solutions in the real world

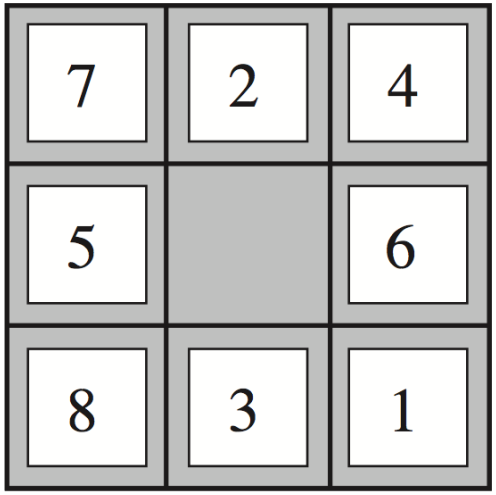
Each abstract action should be "easier" than the original problem!

Example: vacuum world state space graph

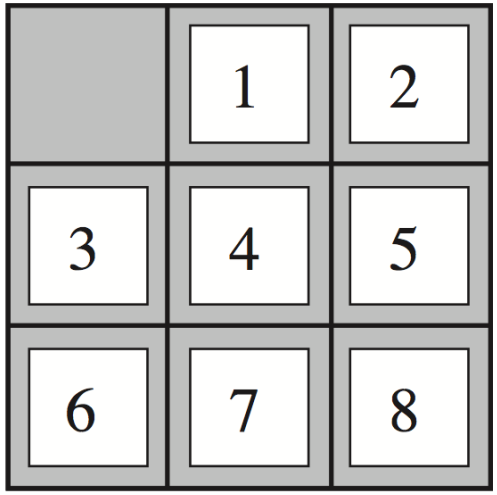


states??: integer dirt and robot locations (ignore dirt amounts etc.)
initial state??: any state
actions??: *Left*, *Right*, *Suck*, *NoOp*
goal test??: no dirt in any location
path cost??: 1 per action (0 for *NoOp*)

Example: The 8-puzzle



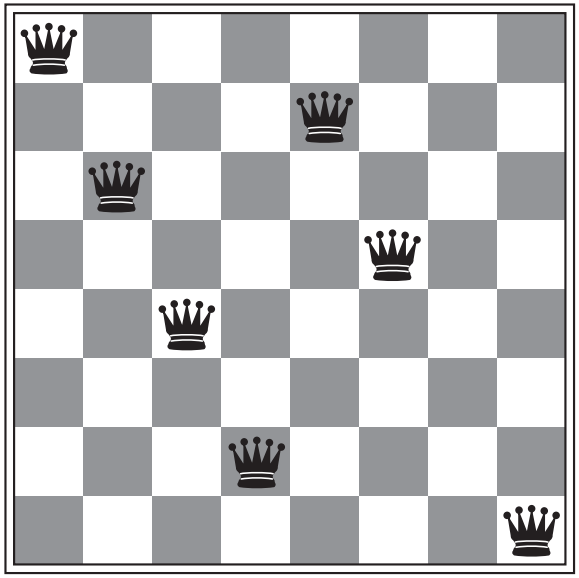
Start State



Goal State

states??: a 3×3 matrix of integers $0 \leq n \leq 8$
initial state??: any state
actions??: move the blank space: left, right, up, down
goal test??: equal to goal state (given above)
path cost??: 1 per move

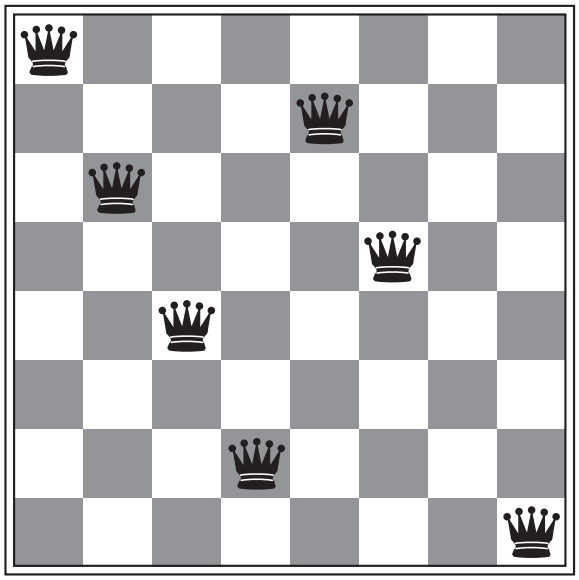
Example: The 8-queens problem



states??: any arrangement of 0 to 8 queens on the board
initial state??: no queens on the board
actions??: add a queen to any empty square
goal test??: 8 queens on the board, none attacked
path cost??: 1 per move

Using this formulation, there are $64 \cdot 63 \cdot \cdots \cdot 57 \approx 1.8 \times 10^{14}$ possible sequences to explore!

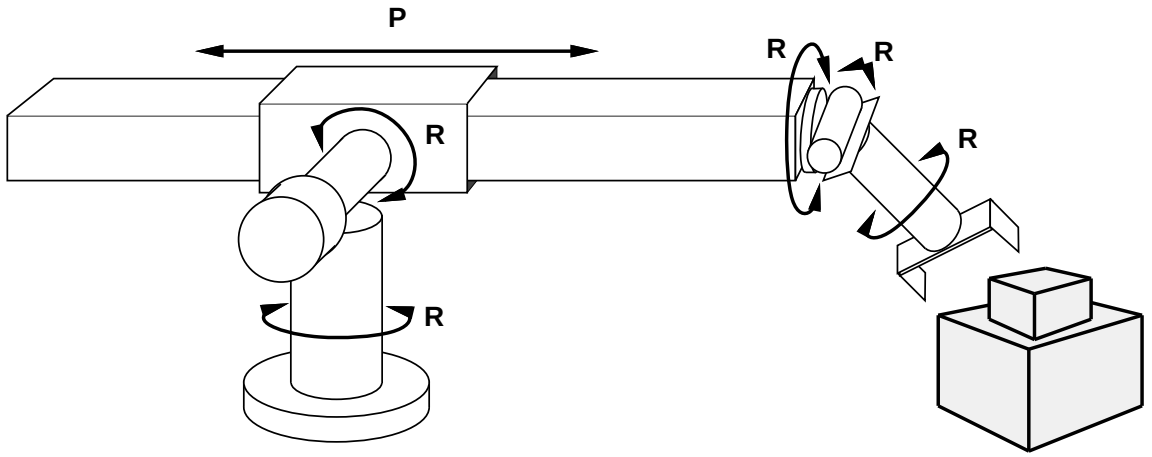
Example: The 8-queens problem (alternative)



states??: one queen per column in the leftmost columns, none attacked
initial state??: no queens on the board
actions??: add a queen to any square in the leftmost empty column,
making sure that no queen is attacked
goal test??: 8 queens on the board, none attacked
path cost??: 1 per move

Using this formulation, we have only 2,057 sequences!

Example: robotic assembly



states??: real-valued coordinates of robot joint angles
parts of the object to be assembled

actions??: continuous motions of robot joints

goal test??: complete assembly of the object

path cost??: time to execute