Introduction to AI

- Artificial
  - Produced by human art or effort, rather than originating naturally.
- Intelligence
  - Quickness of understanding, wisdom.

(From the Concise Oxford Dictionary)

Definitions

"The exciting new effort to make computers think … machines with minds, in the full and literal sense" (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision making, problem solving, learning …" (Bellman, 1978)	"The study of mental faculties through the use of computational models" (Charniak and McDermott, 1985) "The study of computations that make it possible to perceive, reason, and act" (Winston, 1992)
"The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better" (Rich and Knight, 1991)	"A field of study that seeks to explain and emulate intelligent behaviour in terms of computational processes" (Schalkoff, 1990) "The branch of computer science that is concerned with the automation of intelligent behaviour" (Luger and Stubblefield, 1993)

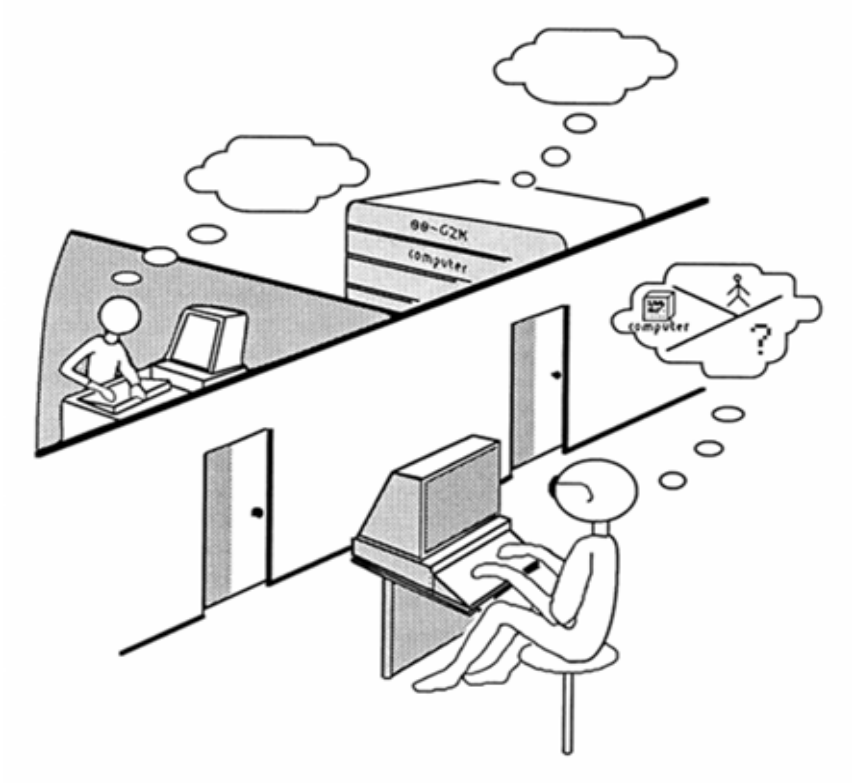
Definitions (con't)

- These definitions are organised into four categories:

Systems that think like humans	Systems that think rationally
Systems that act like humans	Systems that act rationally

Acting Humanly

- The *Turing Test* (Alan Turing 1950).
- To pass this test systems would need:
  - Natural Language Processing
  - Knowledge Representation
  - Automated Reasoning
  - Machine Learning



Thinking Humanly

- A precise theory of the mind can be expressed as a computer program.
- Newell & Simon's GPS (general problem solver) sought to solve problems *the way humans did*.
- Cognitive Science =
  - AI computer models + psychology techniques

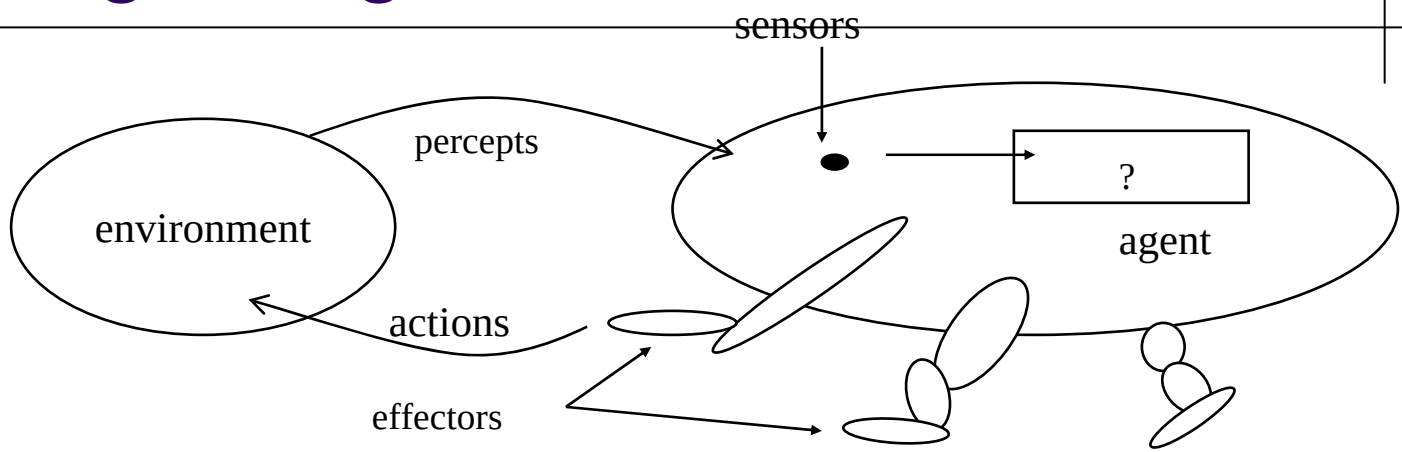
Thinking Rationally

- Logic
  - Stating knowledge in formal terms so that, given correct premises, correct conclusions can be drawn.

Acting Rationally

- Agent
  - something that perceives and acts.
- A rational agent acts to achieve goals, given beliefs.
- More general than thinking rationally, since a correct inference is not always possible, but an action is still needed.
- More agreeable to scientific development than approaches based on human thought or behaviour.

Intelligent Agents



- A rational agent is one that does the right thing.
- Right Action: one that causes the agent the most success.
  - Based on what has been perceived so far
  - Uses a performance measure to determine the degree of success
  - Based on the agent's knowledge of the environment
  - Limited by the set of actions the agent can perform.

- Ideal Rational Agent
  - does all the actions necessary to gain needed information for successful decision making.
    - E.g. looks up and down the street before crossing
- Agent Program
  - implemented action from percept to action
- Architecture
  - computer device that the program runs on

Agent = Program + Architecture

Two Approaches

- General Principles Approach
  - Agents have general intelligence, able to solve different problems in different environments.
  - E.g. Newell and Simon's General Problem Solver (GPS).
- Knowledge Based Approach
  - Agent has knowledge specific to the environment it was expected to solve problems in.
  - E.g. The expert systems MYCIN and DENDRAL

AI History

- ~~1943-1956 - Beginnings:~~
  - Early neural networks =
    - Physiology + Logic + Functions of neurons + Turing's Theory of Computation.
  - Newell & Simon's Logic Theorist
    - proved mathematical theorems.
- 1952-1969 - Lots of Enthusiasm
  - Newell & Simon's GPS
    - thought humanly
  - Arthur Samuel's program to play checkers
    - learned to improve it's game.
  - John McCarthy developed LISP.
  - John McCarthy described Advice Taker
    - had general knowledge and applied new axioms without reprogramming

AI History (con't)

- 1952-1969 (con't)
  - Minsky (and team) focused on microworlds e.g. blocks world (vision, learning,NLP,planning)
  - ELIZA
    - Seemed to engage in serious conversation.
- 1966-1974 - Reality:
  - Methods worked for simple examples but failed at more complex ones.
  - Early programs contained little knowledge of their subject matter e.g. Eliza.
  - More complex problems were too hard
    - NP complete

AI History (con't)

- 1969-1979: Knowledge-Based Systems
  - Knowledge representation schemes and languages became important e.g. Prolog.
  - E.g. Dendral, Mycin
- 1980-1988: Used in Industry
  - R1 - first commercial expert system.
    - Configured computer orders.
  - International competition increased research.
  - Software tools were developed to build expert systems.
  - Workstations optimised for LISP
  - Industrial robotic vision systems.
  - Resurrection of neural network research

AI History (con't)

- 1987-present
  - Building on existing theorems
  - Showing relevance to real world applications
  - base on solid experimental evidence and strong theories
  - Greater reliability and success
  - Improved capabilities

Problem Solving

- Many ways to solve problems:
  - Search: general problem solver
    - Uninformed
    - Informed
      - Some idea of where to find solution
  - Planning: sequence of actions
  - Deduction: logic
  - We will use concepts from analysis of algorithms
    - Review of these concepts are in Appendix A

Goal-based Agents

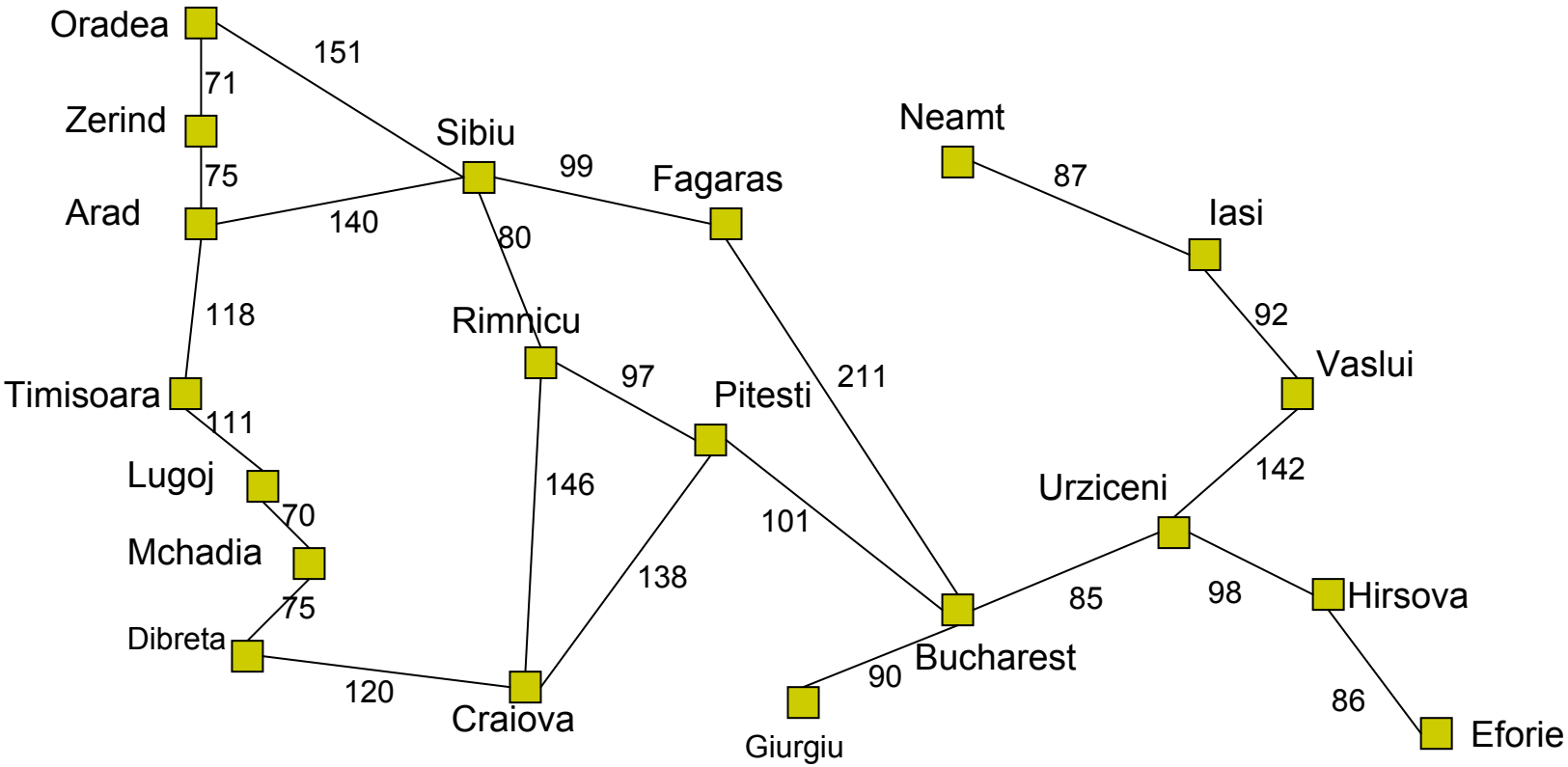
- Where do problems come from?
- Some agents have goals
- When a goal is selected, the agent now has a *problem*: how to achieve the goal?
- Therefore problems are usually designed by agents to achieve goals

Problems

- Math problems are easy!
  - 100^2 - 200
  - Just apply the function
  - Not what we would call “intelligent behavior”
- But there are many much more difficult problems!
  - Non linear
  - Symbolic
  - Interacting variables

Sample Problems

- Route planning
  - Find a path between pairs of cities



Sample Problems

- 8-puzzle:

7	2	4
5		6
8	3	1

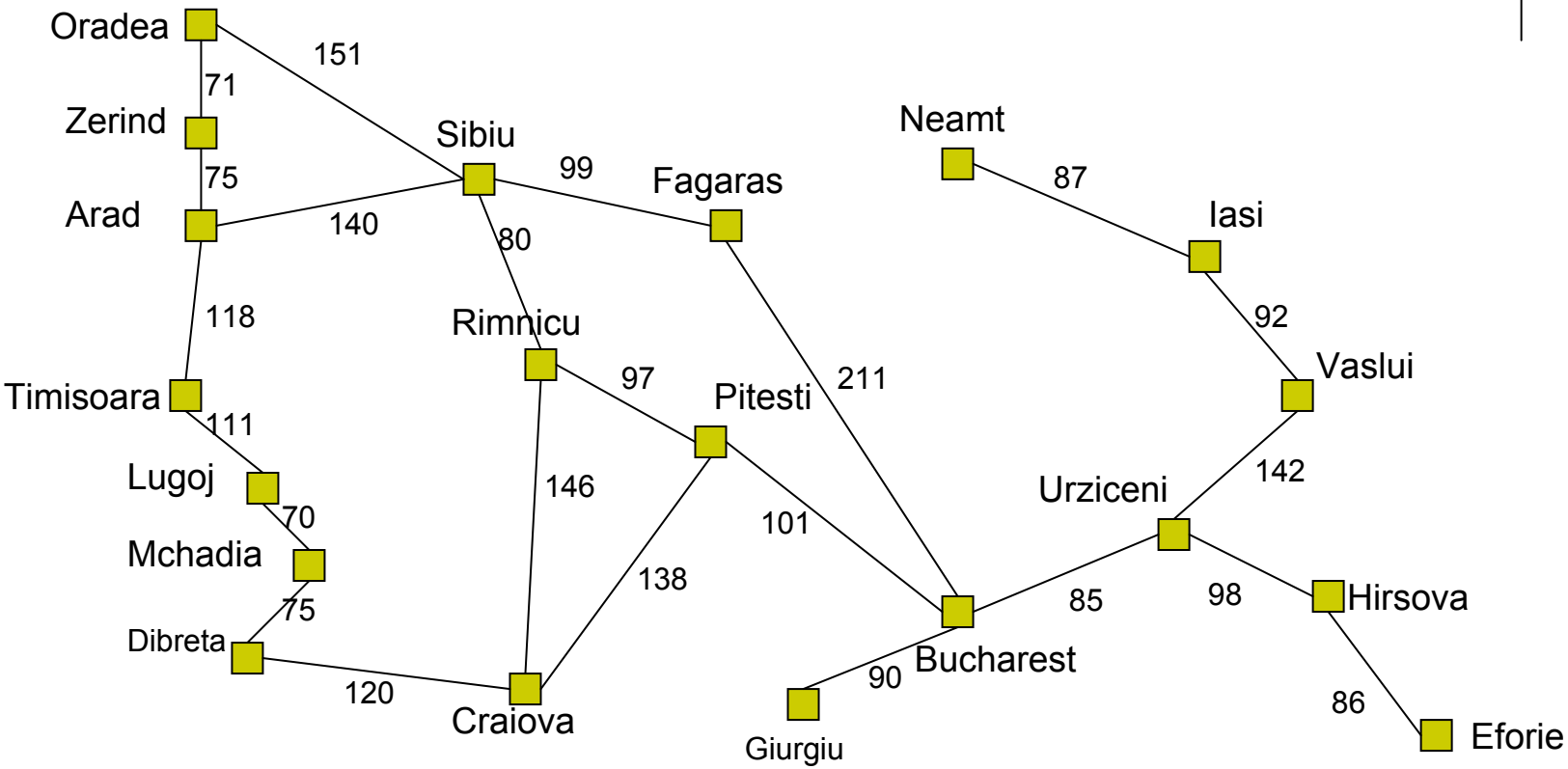
- 8-queens
  - Place eight queens on a chessboard such that no queen attacks any other.

Q							
				Q			
	Q						
					Q		
		Q					
						Q	
			Q				
							Q

Describing Problems

- States: Combination of features from *abstracted* world features
  - Abstraction:
    - removal of detail
- Goal test: Function that returns TRUE if a state is a *goal* state
  - Goal state: state we are trying to achieve

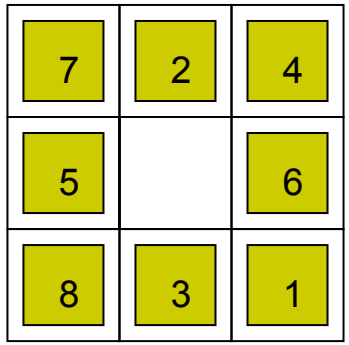
Rout Planning



- States: cities
- Goal test: city = desired city

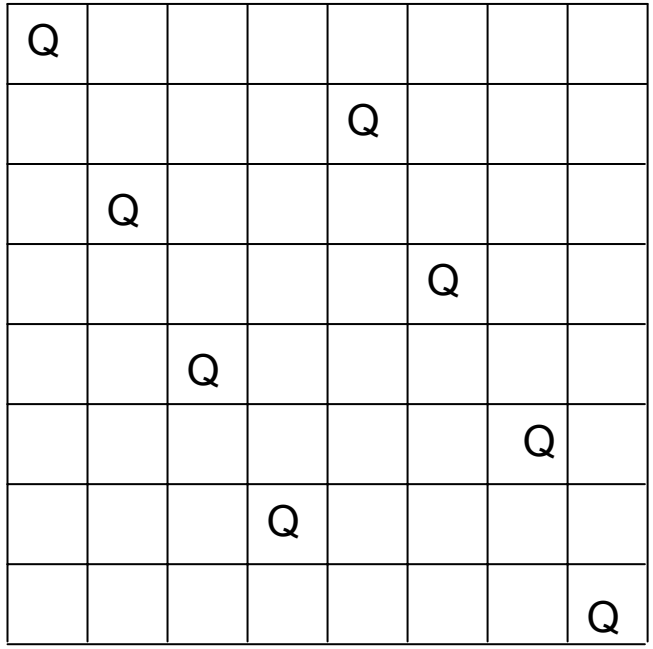
8-Puzzle

- States: puzzle configurations
- Goal-state: state = desired configuration



Sample Problems

- States: board configurations
- Goal state: no queen threatens another



Brute-force solution

- So, how do we solve problems:
- Brute force (dumb, but very simple):
  - *Generate & test:*
    - *Generate all possible states*
    - *Test each state until goal test returns true*
- Advantages:
  - Simple
- Disadvantages:
  - Slow, sometimes impossible

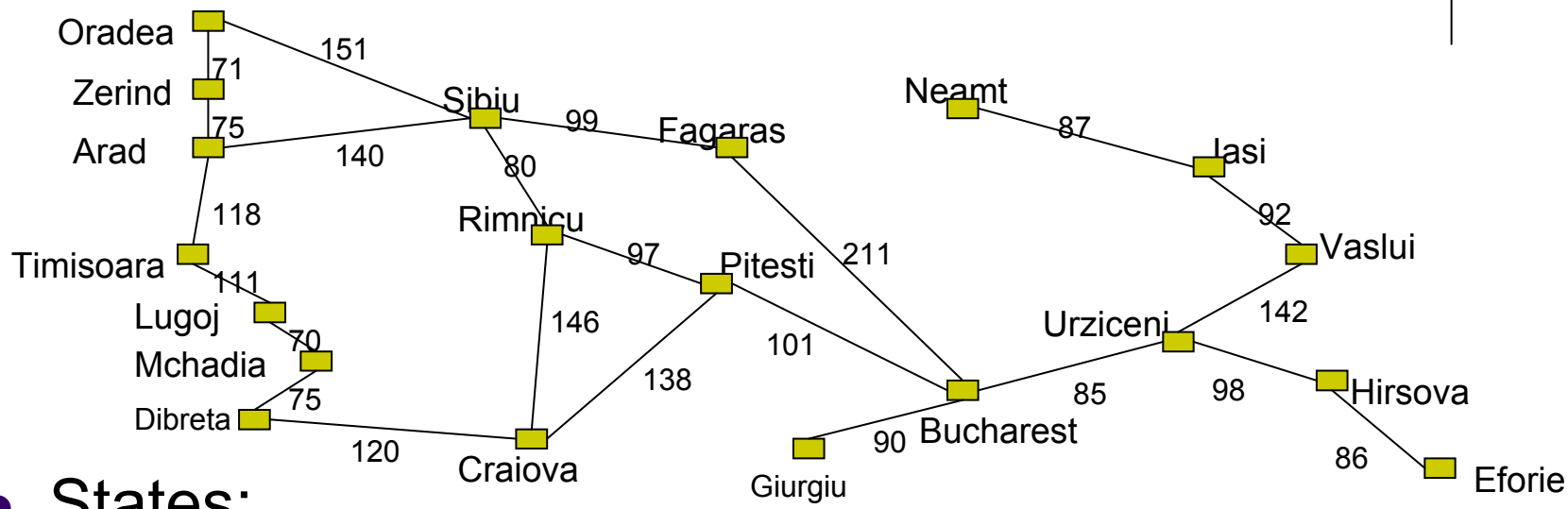
Better way to solve problems

- What are we trying to do?
- ***Find*** the solution in a space of possible answers
- Maybe we can ***search*** through the states in some kind of principled manner, beginning with an ***initial state*** (given or randomly chosen)
- Need a function to identify neighboring states: *successor function*—returns the next (successor) states to try
- Sometimes, as in route planning, we are interesting in ***how*** we get to the goal state from the start state
  - ***Path***
  - ***Path cost***

Search Problems

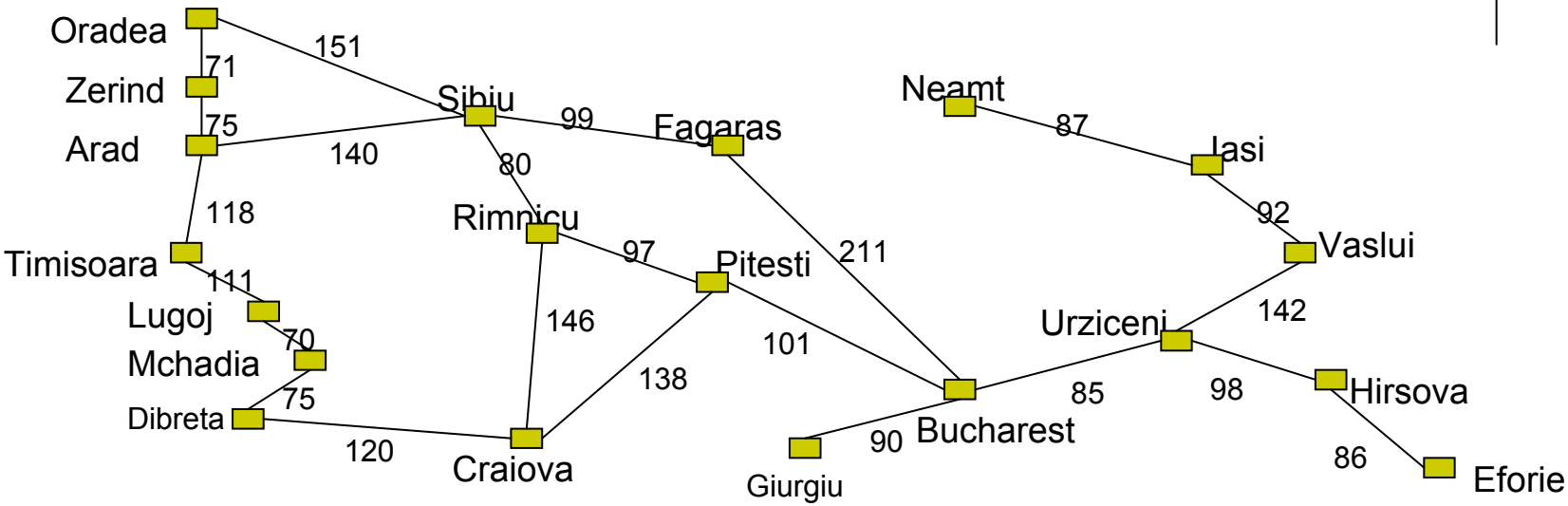
- ***States:*** configurations of features
- ***Initial state:*** initial configuration of features
- ***Successor function:*** function for finding states neighboring a given state
  - ***State space*** is defined by initial state and successor function
  - ***Path:*** Sequence of states
- ***Goal test:*** predicate for determining if a state is a goal state
- ***Path cost:*** a utility assigned to a path
  - ***Step cost:*** cost of a step in the path
- ***Solution:*** a path from the initial state to a goal state
  - ***Optimal solution:*** lowest path cost among solutions
- ***General search procedure:***
  - Begin with start state
    - Loop until an explored state is a goal state:
      - Explore neighbors of current state

Rout Planning: Arad to Bucharest



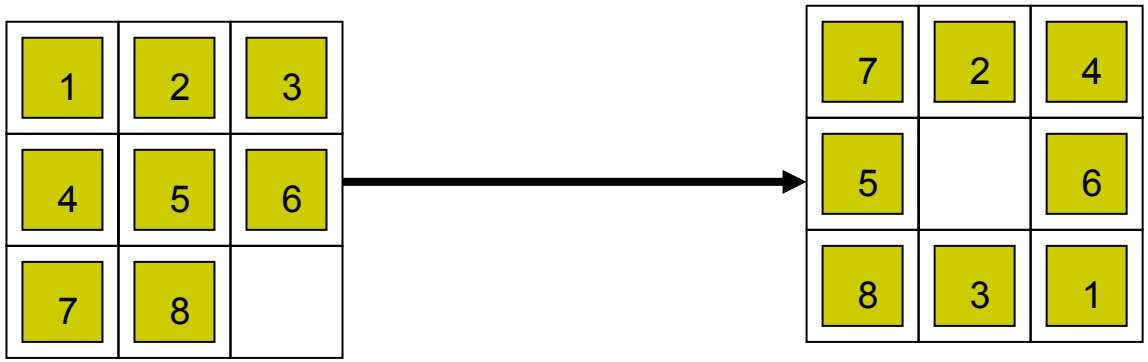
- States:
- Initial state:
- Successor function:
- Goal test:
- Path cost:

Rout Planning: Arad to Bucharest



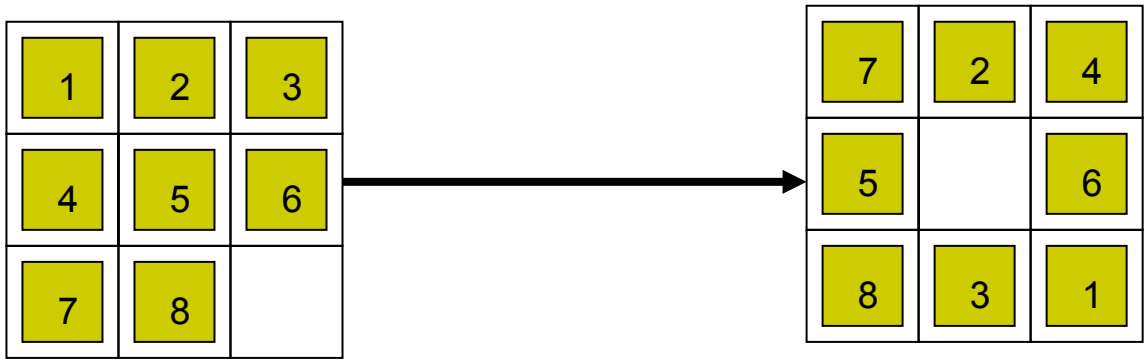
- States: cities
- Initial state: Arad
- Successor function: neighboring cities
- Goal test: city = Bucharest
- Path cost: total mileage in path

8-Puzzle



- States:
- Initial state:
- Successor function:
- Goal test:
- Path cost:

8-Puzzle



- States: puzzle configurations
- Initial state: see above
- Successor function: move blank up, down, left, right
- Goal test: see above
- Path cost: 1 per move

8-Queens

- States:
- Initial state:
- Successor function:
- Goal test:
- Path cost:

Q							
				Q			
	Q						
					Q		
		Q					
						Q	
			Q				
							Q

8-Queens

- States: board configurations
- Initial state: empty board
- Successor function:
  - Version 1: add a queen to any empty square
  - Version 2: add a queen to any square in the leftmost empty column such that it is not attacked by any other queen
- Goal test: 8 queens are on the board, none are attacked
- Path cost: 0; path does not matter

Q							
				Q			
	Q						
					Q		
		Q					
						Q	
			Q				
							Q

8-Queens

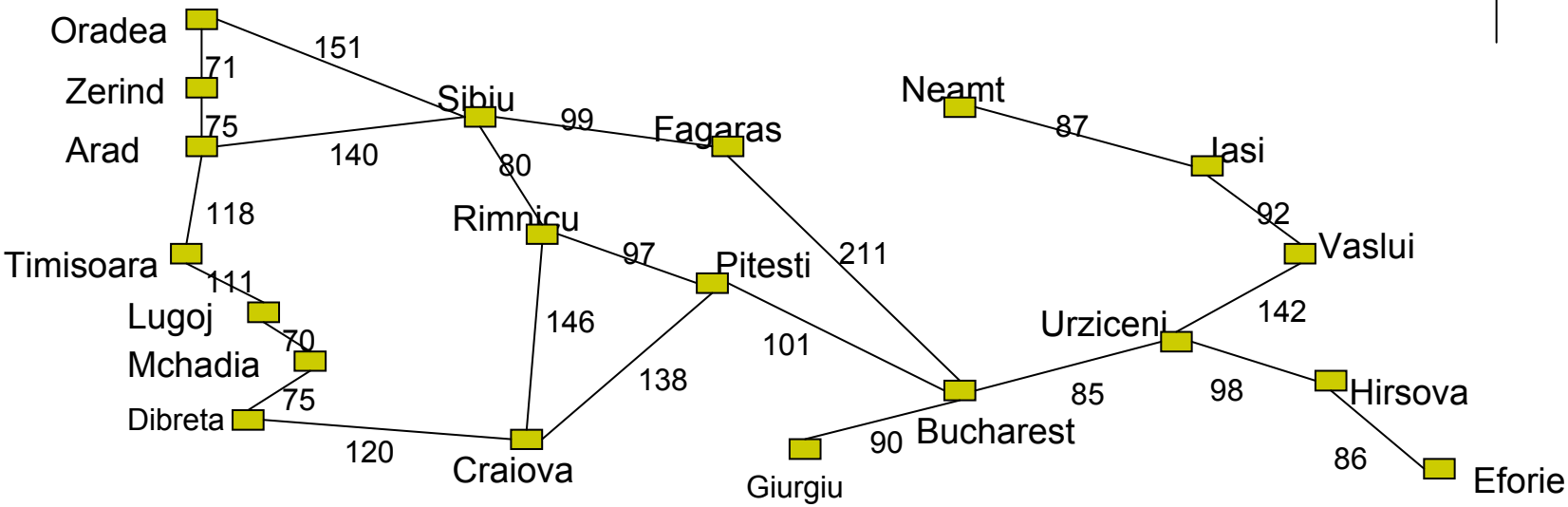
- States: board configurations
- Initial state: empty board
- Successor function:
  - Version 1: add a queen to any empty square
    - ≈ 3 X 10^{14} configurations
  - Version 2: add a queen to any square in the leftmost empty column such that it is not attacked by any other queen
    - 2,057 configurations!
- Goal test: 8 queens are on the board, none are attacked
- Path cost: 0; path does not matter

Q							
				Q			
	Q						
					Q		
		Q					
						Q	
			Q				
							Q

Successor functions

- Morals:
  - How you abstract the world can make a big difference
  - Successor function can make a big difference
    - Design successor functions wisely!

Example: Rout Planning--Arad to Bucharest



- States: cities
- Initial state: Arad
- Successor function: neighboring cities
- Goal test: city = Bucharest
- Path cost: total mileage in path

Conclusions

- Well-defined search problem:
  - States
    - abstractions of world
  - Initial state
  - Successor function
    - How to get to neighboring states
  - Goal test
  - Path cost
- Definition of states, successor function can make a big difference in how hard the problem is

Real problems

- Route-finding
  - Touring problems
    - Shortest trip to visit all destinations
    - Traveling salesperson problem
    - Robot navigation
- VLSI layout
  - Cell layout
  - Channel routing
- Automatic assembly sequencing
  - Protein design
- Internet searching

Search Problems

- Two types of search problems
- Configuration search
  - Solution is a state (configuration) satisfying the goal test
- Path search
  - Solution is a *path* to a goal state

Search Strategies

- General search:
  - Most search strategies are combinations of these
  - Some of the features interact when combined
- Basic strategy
  - breadth-first vs. depth-first vs. depth-limited depth-first vs. iterative deepening
- Direction
  - Forward vs. backward vs. bi-directional
- Repeated states
  - Don't check vs. check parents vs. check all old states
- Paths
  - None vs. path vs. paths with costs
- Heuristics
  - None vs. heuristics
- How many unexplored nodes to save
  - All
  - One (iterative improvement)
  - Some (beam search)
- Specialized searches
  - Often use the general search strategies as components.
  - And/Or e.g,. menu planning
  - Constraint satisfaction e.g., map coloring
  - Game playing e.g, Chess

Search

- Searching is often viewed as a tree
  - Initial state is the root
- How to find new states to try:
  - Apply successor to current state
  - Expanding current state
- Search node
  - A step in the search
  - Components:*
    - State: current state
    - Parent node
    - Path
    - Path-cost: cost of path from initial state to state
    - Depth: number of steps from initial state to state
    - Expanding a node ➔ expanding the node's state
      - A node will be generated for each new state
- Node ≠ State!
  - A node refers to one state
  - Many nodes may refer to the same state
  - Node space is different from state space