Review

- Agent Definition
- PEAS description
  - (P)erformance Measure
  - (E)nvironment
  - (A)cutators
  - (S)ensors
- Environment Properties

Basic Types of Agent Programs

- Simple reflex agents
    - Condition-action rules on current percept
    - Environment must be fully observable

Basic Types of Agent Programs

- Simple reflex agents
    - Condition-action rules on current percept
    - Environment must be fully observable
- Model-based reflex agents
    - Maintain internal state about how world the world evolves and how actions effect the world

Basic Types of Agent Programs

- Simple reflex agents
    - Condition-action rules on current percept
    - Environment must be fully observable
- Model-based reflex agents
    - Maintain internal state about how world the world evolves and how actions effect the world
- Goal-based agents
    - Use goals and planning to help make decision

Basic Types of Agent Programs

- Simple reflex agents
    - Condition-action rules on current percept
    - Environment must be fully observable
- Model-based reflex agents
    - Maintain internal state about how world the world evolves and how actions effect the world
- Goal-based agents
    - Use goals and planning to help make decision
- Utility-based agents
    - What makes the agent “happiest”

Basic Types of Agent Programs

- Simple reflex agents
  - Condition-action rules on current percept
  - Environment must be fully observable
- Model-based reflex agents
  - Maintain internal state about how world the world evolves and how actions effect the world
- Goal-based agents
  - Use goals and planning to help make decision
- Utility-based agents
  - What makes the agent "happiest" Utility

Basic Types of Agent Programs

- Simple reflex agents
    - Condition-action rules on current percept
    - Environment must be fully observable
- Model-based reflex agents
    - Maintain internal state about how world the world evolves and how actions effect the world
- Goal-based agents
    - Use goals and planning to help make decision
- Utility-based agents
    - What makes the agent “happiest”
- Learning agents
    - Makes improvements

Problem solving and search

Problem Solving

- Goal-based agent
- Decides what to do by finding sequences of actions that lead to desirable states

Goal Based Agent in Romania

- An agent in Arad, Romania (on holiday)

Goal Based Agent in Romania

- An agent in Arad, Romania (on holiday)
- Performance measure could be
  - Sightsee,
  - eat good food,
  - enjoy nightlife, etc.

Goal Based Agent in Romania

- An agent in Arad, Romania (on holiday)
- Performance measure could be
  - Sightsee,
  - eat good food,
  - enjoy nightlife, etc.
- Suppose have nonrefundable ticket to fly out of Bucharest tomorrow
  - Agent now adopts goal of getting to Bucharest on time

Goal Based Agent in Romania

- An agent in Arad, Romania (on holiday)
- Performance measure could be
  - Sightsee,
  - eat good food,
  - enjoy nightlife, etc.
- Suppose have nonrefundable ticket to fly out of Bucharest tomorrow
  - Agent now adopts goal of getting to Bucharest on time
- Courses of action that do not reach Bucharest on time can be rejected
  - Simplifies decision making

Goal Based Agent in Romania

- An agent in Arad, Romania (on holiday)
- Performance measure could be
  - Sightsee,
  - eat good food,
  - enjoy nightlife, etc.
- Suppose have nonrefundable ticket to fly out of Bucharest tomorrow
  - Agent now adopts goal of getting to Bucharest on time
- Courses of action that do not reach Bucharest on time can be rejected
  - Simplifies decision making
- “Goals help organize behavior by limiting objective that agent is trying to achieve”

Goal Based Agent

- [Step 1] Formulate goal
  - Based on current situation

Goal Based Agent

- [Step 1] Formulate goal
  - Based on current situation
  - Goal: a set of states in which the goal is satisfied

Goal Based Agent

- [Step 1] Formulate goal
  - Based on current situation
  - Goal: a set of states in which the goal is satisfied
    - Actions cause transitions between world states
      - Need to find sequence of actions to get to goal

Goal Based Agent

- [Step 1] Formulate goal
  - Based on current situation
  - Goal: a set of states in which the goal is satisfied
    - Actions cause transitions between world states
      - Need to find sequence of actions to get to goal
- [Step 2] Formulate problem
  - Decide what actions and states to consider, given a goal
  - Decide how to quantify *best* solution
  - Need higher level of detail, else becomes intractable
  - Organize actions and states (graph structure, map)

Goal Based Agent

- [Step 1] Formulate goal
  - Based on current situation
  - Goal: a set of states in which the goal is satisfied
    - Actions cause transitions between world states
      - Need to find sequence of actions to get to goal
- [Step 2] Formulate problem
  - Decide what actions and states to consider, given a goal
  - Decide how to quantify *best* solution
  - Need higher level of detail, else becomes intractable
  - Organize actions and states (graph structure, map)

How find best sequence (path) of actions?

Goal Based Agent

- [Step 3] Search
  - Process of looking for best action sequence (to reach goal)
  - Input: formulated problem
  - Output: solution (as action sequence)

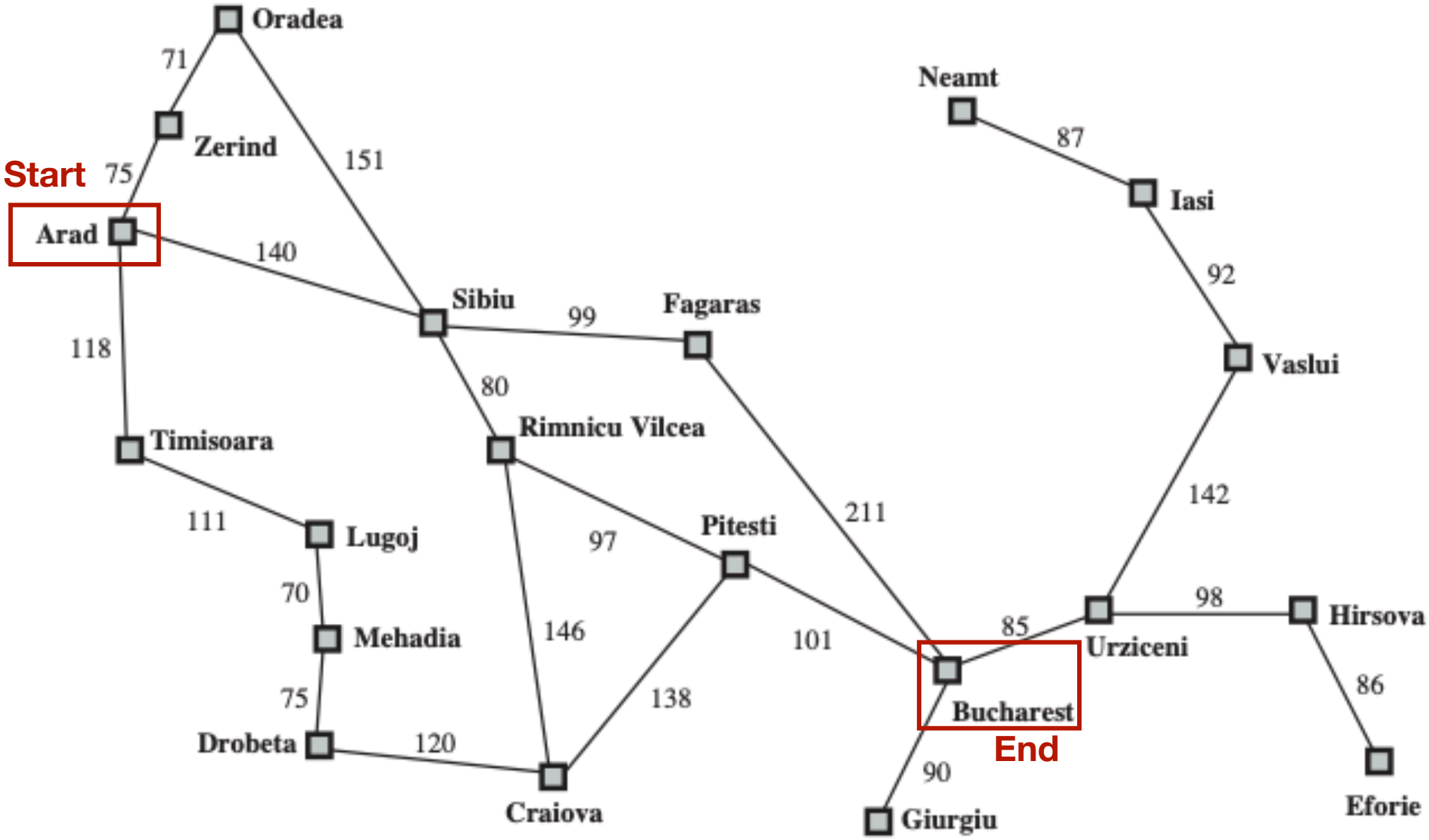
Goal Based Agent

- [Step 3] Search
  - Process of looking for best action sequence (to reach goal)
  - Input: formulated problem
  - Output: solution (as action sequence)
- [Step 4] Execution phase
  - Execute recommended actions

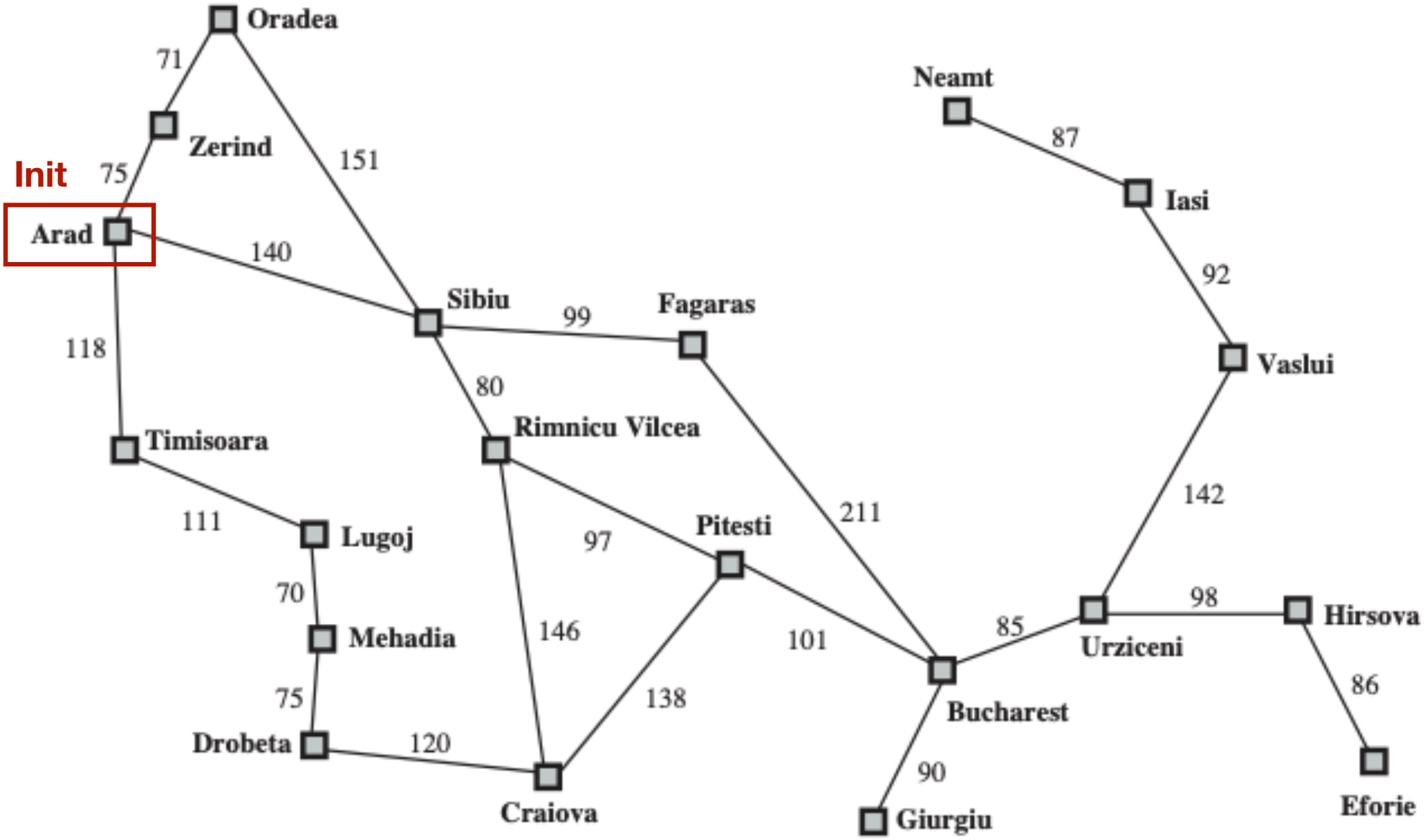
Formulating problems

- [Step 3] Search
  - Process of looking for best action sequence (to reach goal)
  - Input: formulated problem
  - Output: solution (as action sequence)
- [Step 4] Execution phase
  - Execute recommended actions
- [Step 5] Find a new goal (repeat from step 1)

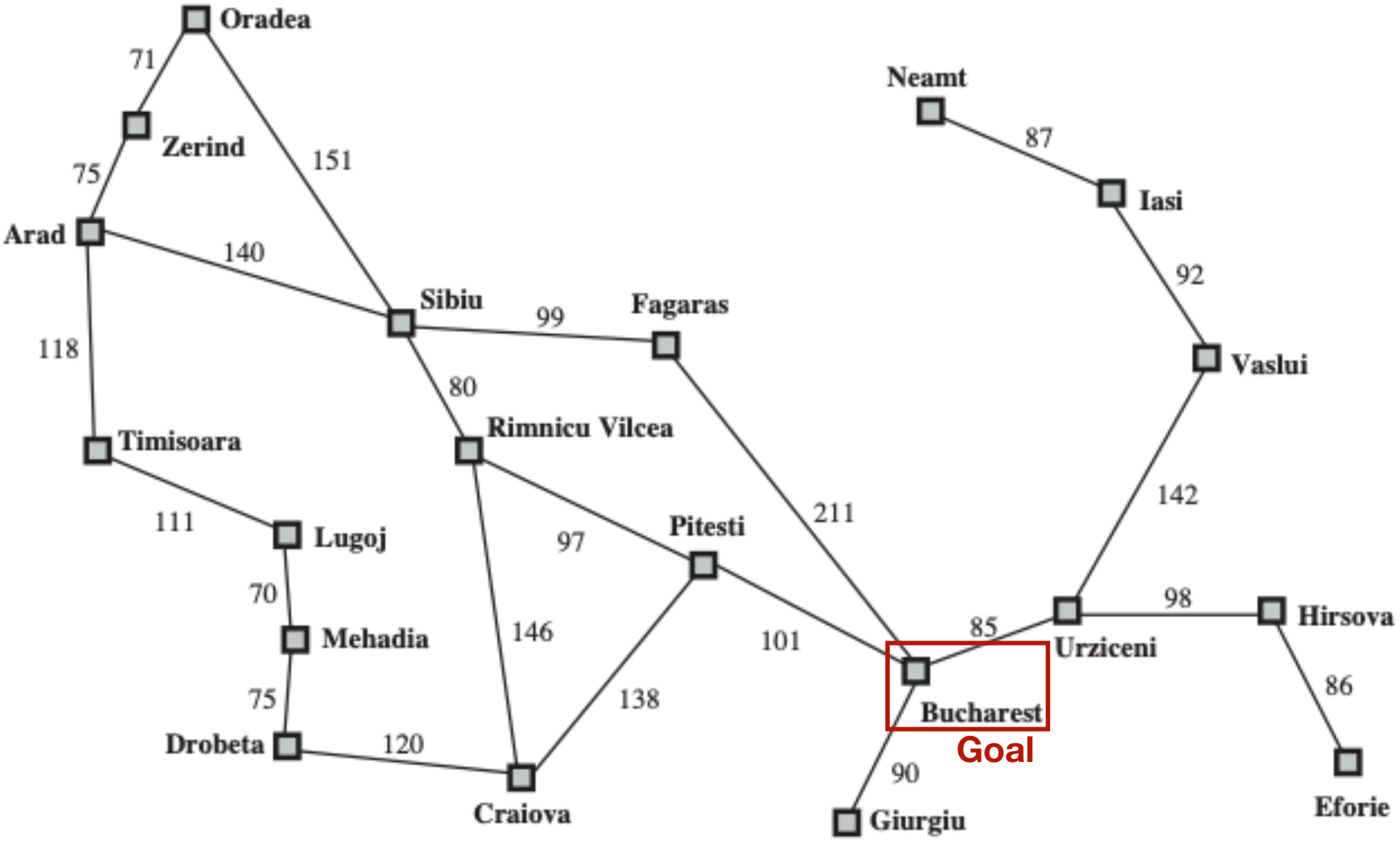
Formulating Romania Visit



Formulating Romania Visit



Formulating Romania Visit



Goal Based Agent Environment

- Static
  - Formulating and solving problem in a fixed environment
- Fully observable
  - All states knowable
- Discrete
  - Cities are nodes, actions are links
- Deterministic
  - No randomness assumed
- Agent
  - Single

Well-Defined Problems and Solutions

A problem is defined by four items

1. Initial state
2. Actions/Operators
3. Goal test
4. Path cost

Well-Defined Problems and Solutions

A problem is defined by four items

1. **Initial state**
    1. *In*(*Arad*)

Well-Defined Problems and Solutions

A problem is defined by four items

1. **Initial state**
    1. *In*(*Arad*)
2. **Actions/Operators**
    1. Actions available to agent
    2. SUCCESSOR-FN(*In*(*Arad*)) = {< *Go*(*Sibiu*), *In*(*Sibiu*) >, < *Go*(*Zerind*), *In*(*Zerind*)>, < Go(Timisoara), In(Timisoara)>}

Well-Defined Problems and Solutions

A problem is defined by four items

1. **Initial state**
    1. *In*(*Arad*)
2. **Actions/Operators**
    1. Actions available to agent
    2. SUCCESSOR-FN(*In*(*Arad*)) = {< *Go*(*Sibiu*), *In*(*Sibiu*) >, < *Go*(*Zerind*), *In*(*Zerind*)>, < Go(Timisoara), In(Timisoara)>}
3. **Goal test**
    1. Determines whether a given state is a goal state
    2. *In*(*Bucharest*)

Well-Defined Problems and Solutions

A problem is defined by four items

1. **Initial state**
    1. *In*(*Arad*)
2. **Actions/Operators**
    1. Actions available to agent
    2. SUCCESSOR-FN(*In*(*Arad*)) = {< *Go*(*Sibiu*), *In*(*Sibiu*) >, < *Go*(*Zerind*), *In*(*Zerind*)>, < Go(Timisoara), In(Timisoara)>}
3. **Goal test**
    1. Determines whether a given state is a goal state
    2. *In*(*Bucharest*)
4. **Path cost**
    1. A path is sequence of states connected by actions
    2. Used to check if one solution path is better than another
    3. Assigns numeric cost to each path
    4. Should reflect performance measure
    5. Sum of distances

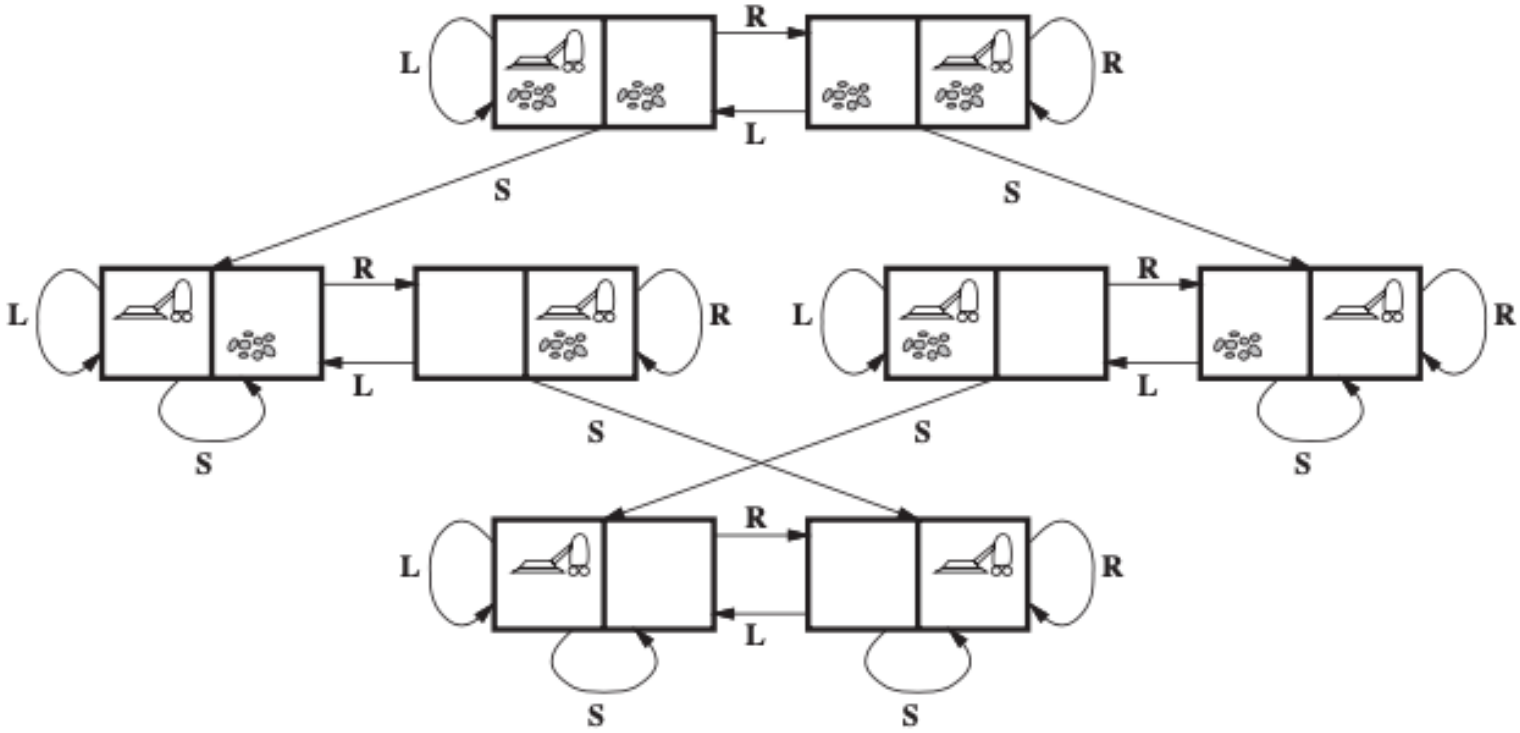
Well-Defined Problems and Solutions

A problem is defined by four items

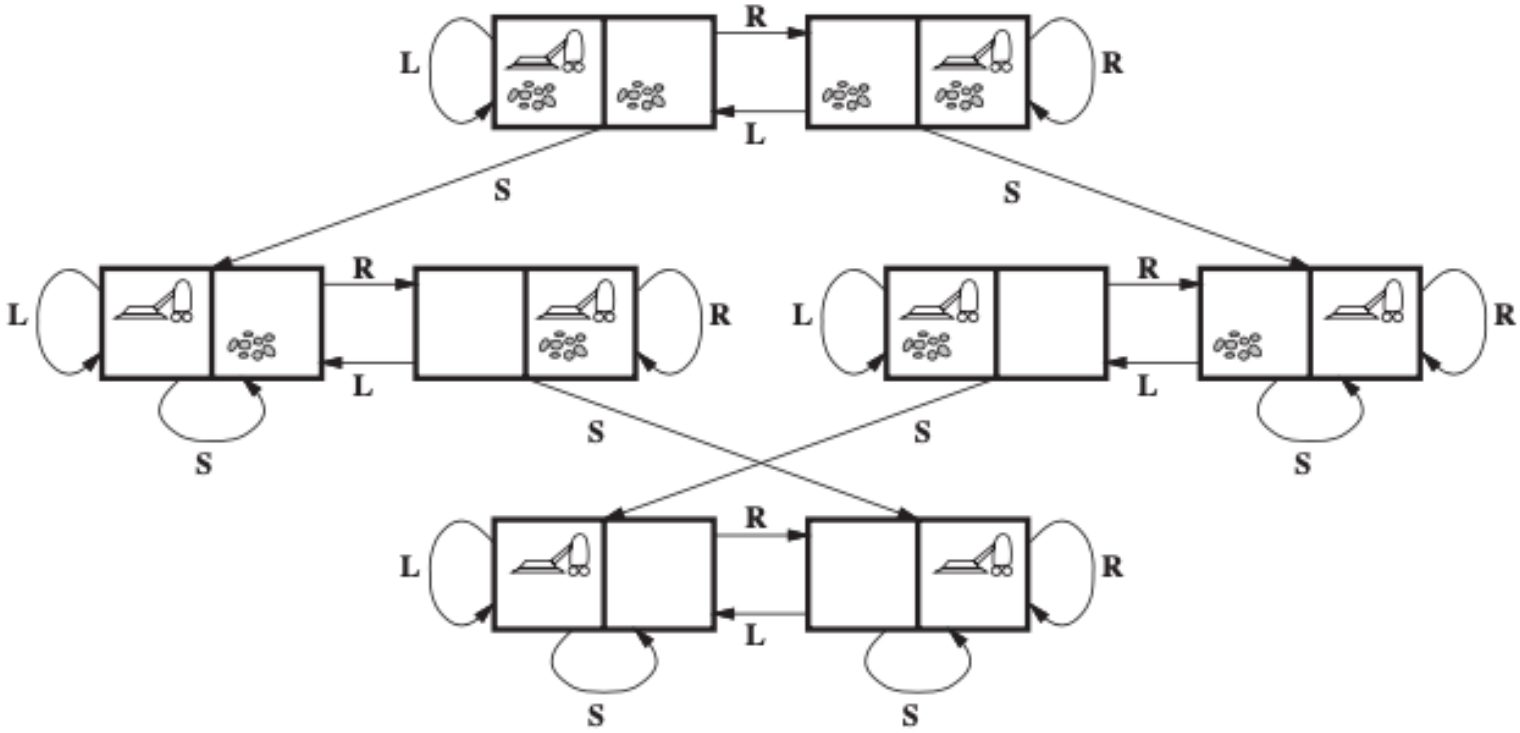
1. **Initial state**
    1. *In*(*Arad*)
2. **Actions/Operators**
    1. Actions available to agent
    2. SUCCESSOR-FN(*In*(*Arad*)) = {< *Go*(*Sibiu*), *In*(*Sibiu*) >, < *Go*(*Zerind*), *In*(*Zerind*)>, < Go(Timisoara), In(Timisoara)>}
3. **Goal test**
    1. Determines whether a given state is a goal state
    2. *In*(*Bucharest*)
4. **Path cost**
    1. A path is sequence of states connected by actions
    2. Used to check if one solution path is better than another
    3. Assigns numeric cost to each path
    4. Should reflect performance measure
    5. Sum of distances

A solution is a sequence of operators leading from the initial state to a goal state & Optimal solution has lowest path cost

Toy Problem: Vacuum World

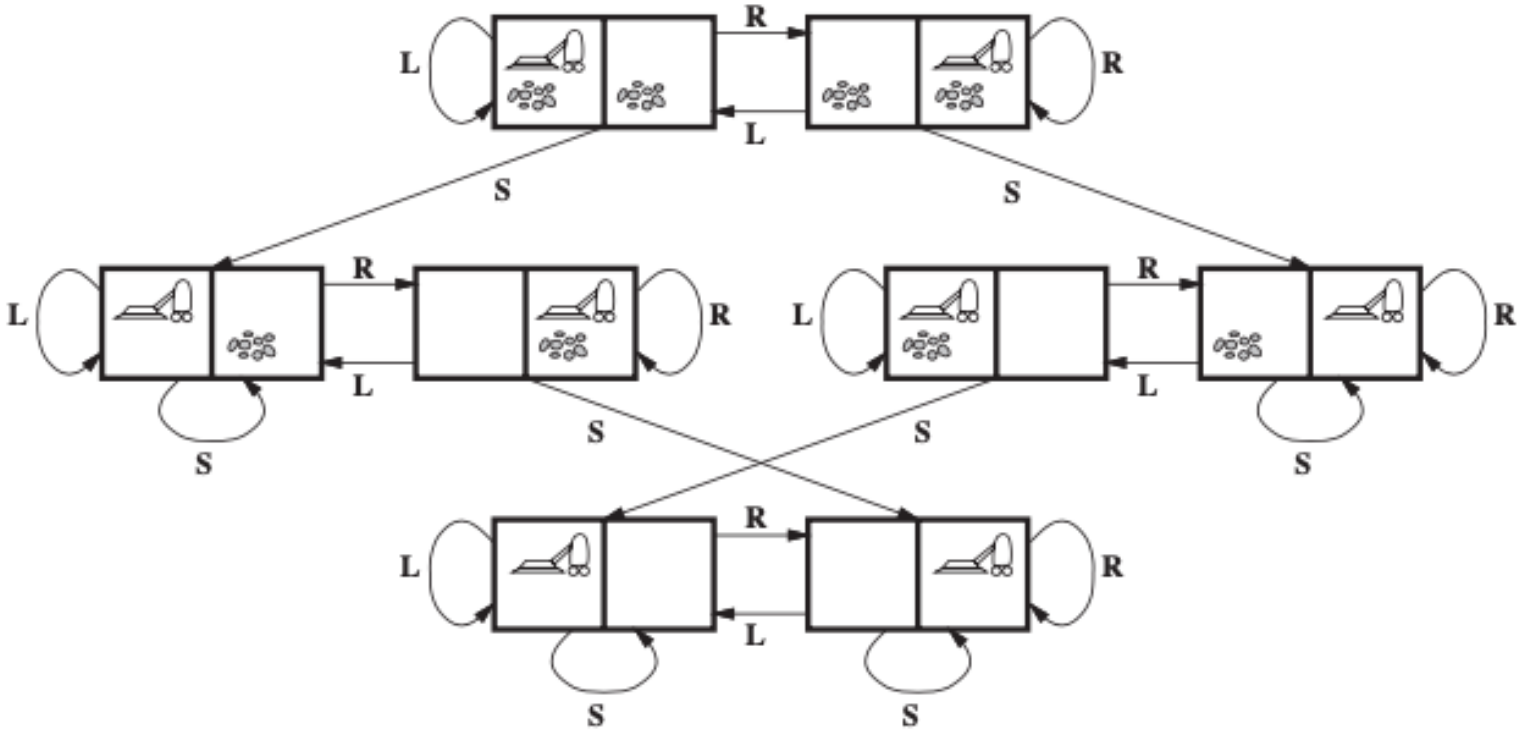


Toy Problem: Vacuum World



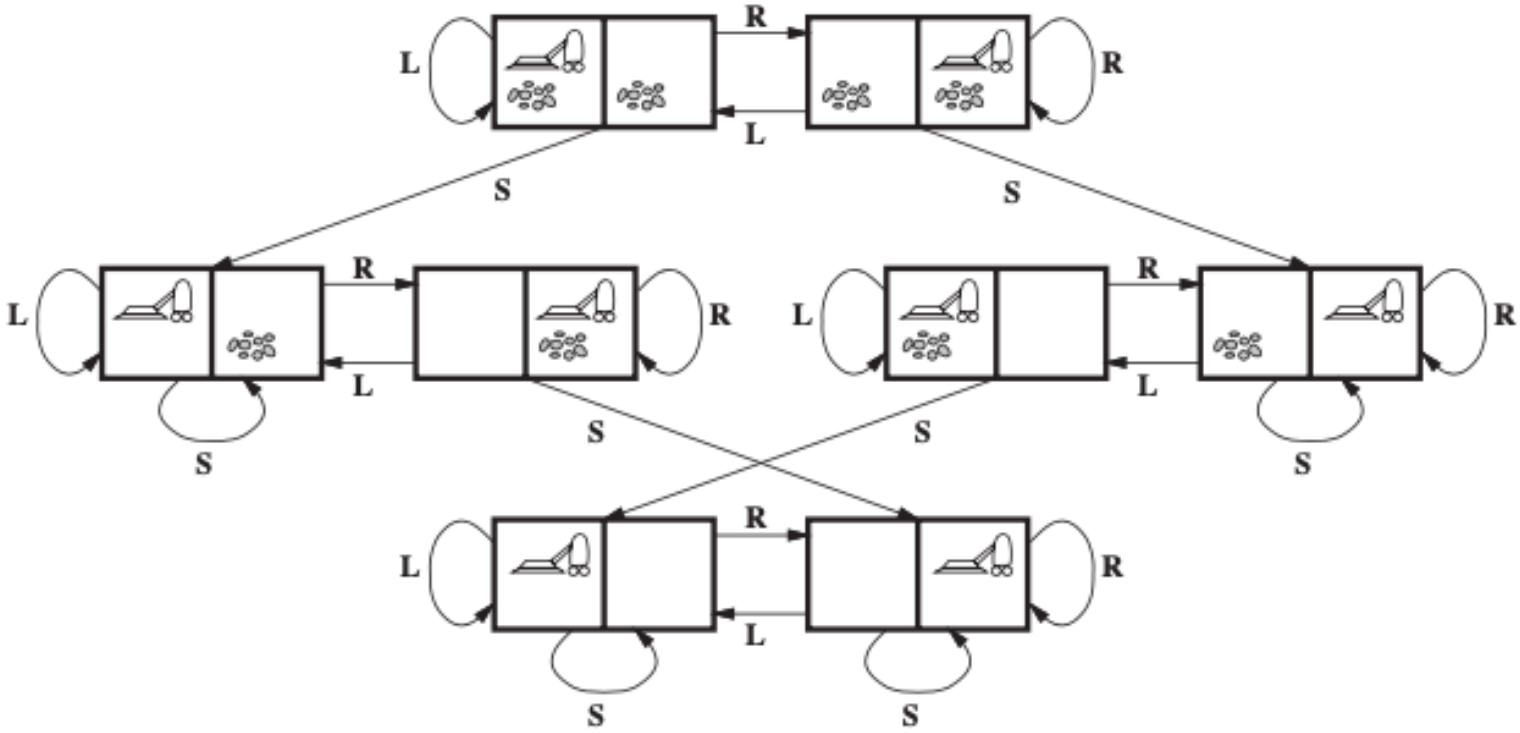
- **States:** Integer (discrete) dirt and robot locations

Toy Problem: Vacuum World



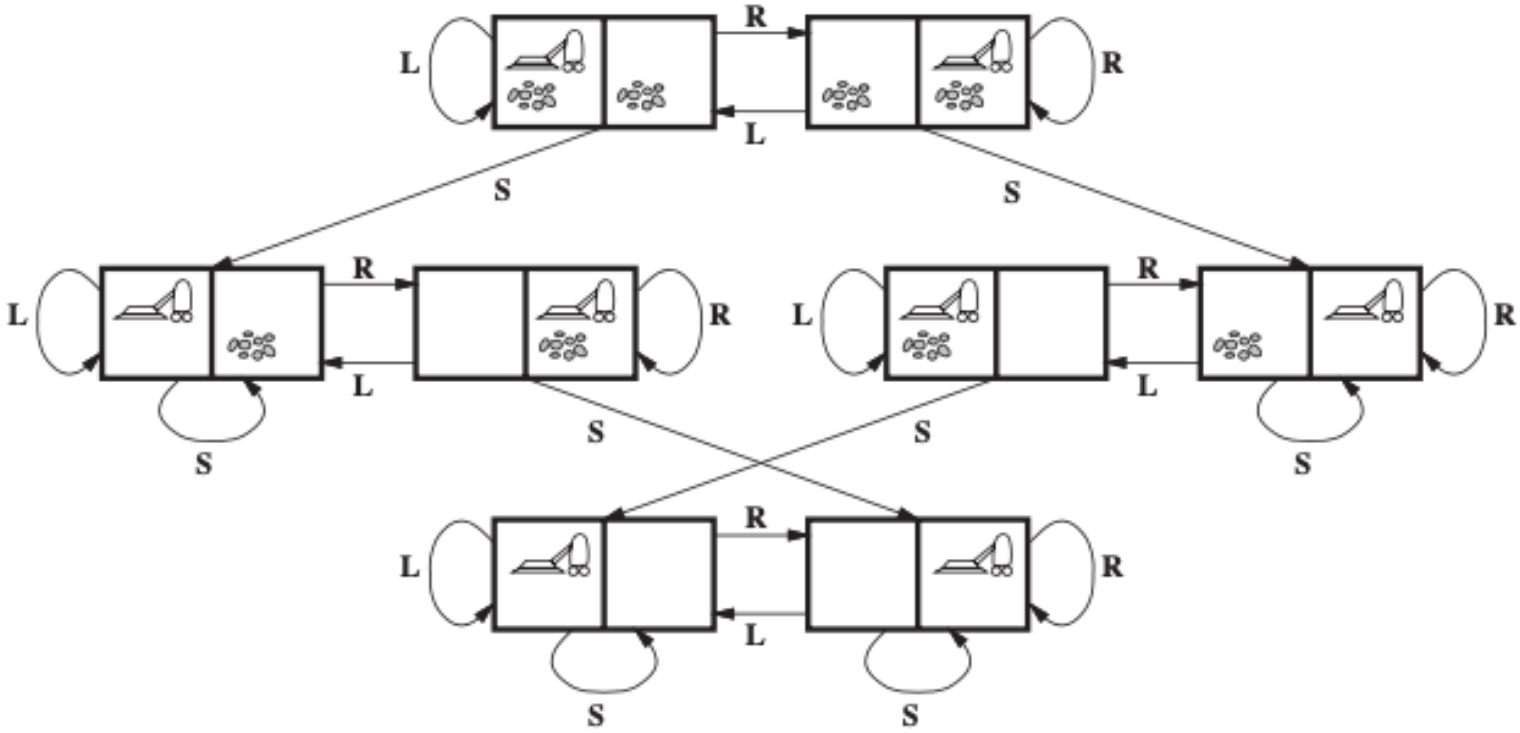
- **States:** Integer (discrete) dirt and robot locations
- **Initial State:** Any state

Toy Problem: Vacuum World



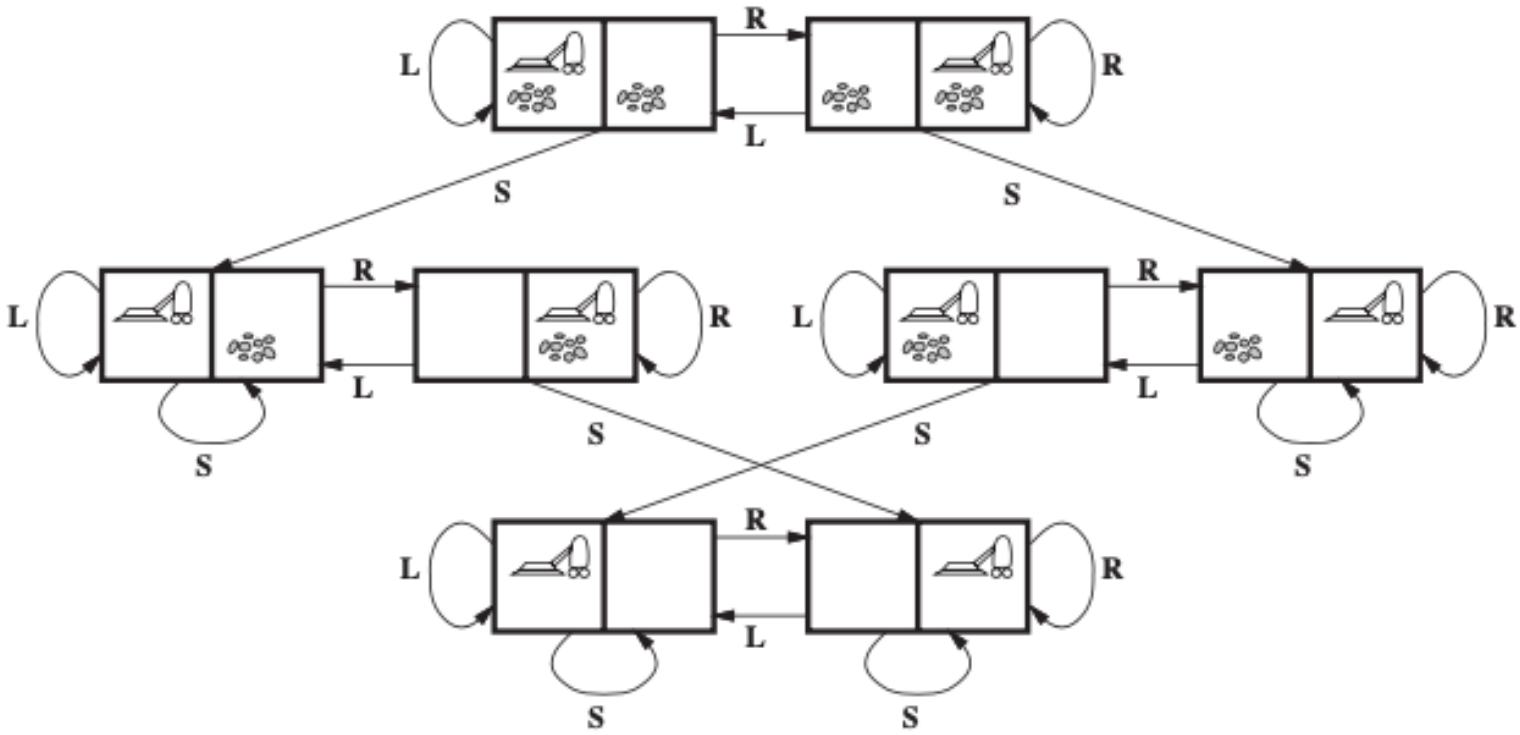
- **States:** Integer (discrete) dirt and robot locations
- **Initial State:** Any state
- **Actions:** Left, Right, Vacuum

Toy Problem: Vacuum World



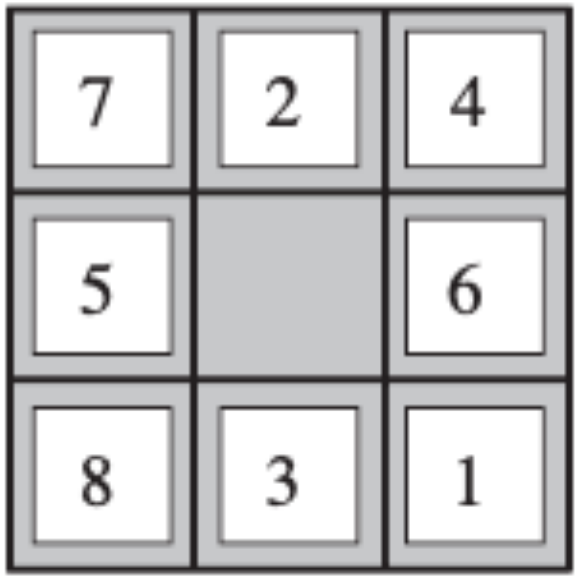
- **States:** Integer (discrete) dirt and robot locations
- **Initial State:** Any state
- **Actions:** Left, Right, Vacuum
- **Goal Test:** Al squares clean

Toy Problem: Vacuum World

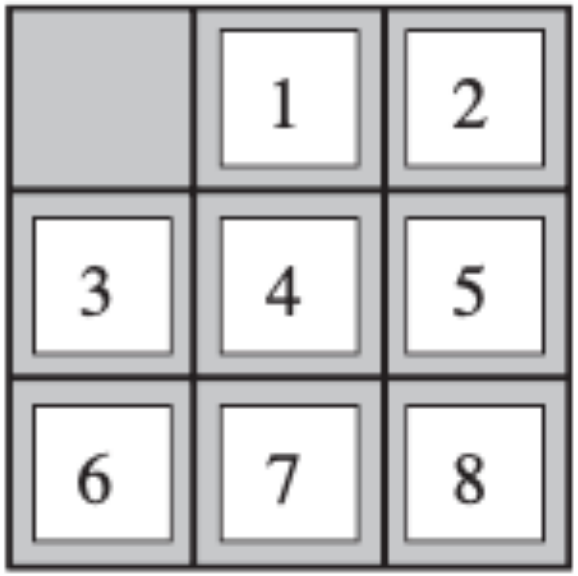


- **States:** Integer (discrete) dirt and robot locations
- **Initial State:** Any state
- **Actions:** Left, Right, Vacuum
- **Goal Test:** Al squares clean
- **Path Cost:** Additive, each step costs 1

Toy Problem: 8 Puzzle

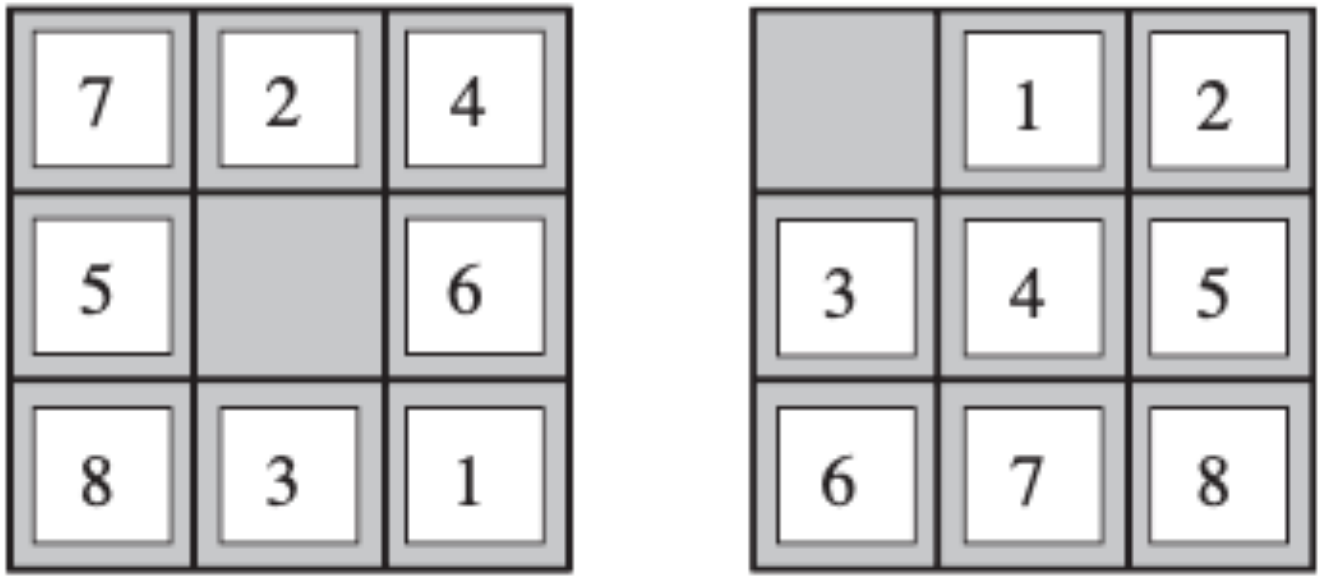


Start State



Goal State

Toy Problem: 8 Puzzle



Start State

Goal State

- **States:**Integer (discrete) locations of each tile and blank
- **Initial State:** Any state
- **Actions:** (Left, Right, Up, Down)
- **Goal Test:** Matches goal configuration
- **Path Cost:** Additive, each step costs 1

Searching For Solutions

- Have formulated problem, now need to solve it

Searching For Solutions

- Have formulated problem, now need to solve it
- Search through state space
  - Using search tree
- Search node
  - Initial state
  - **Test if already goal state!!!**

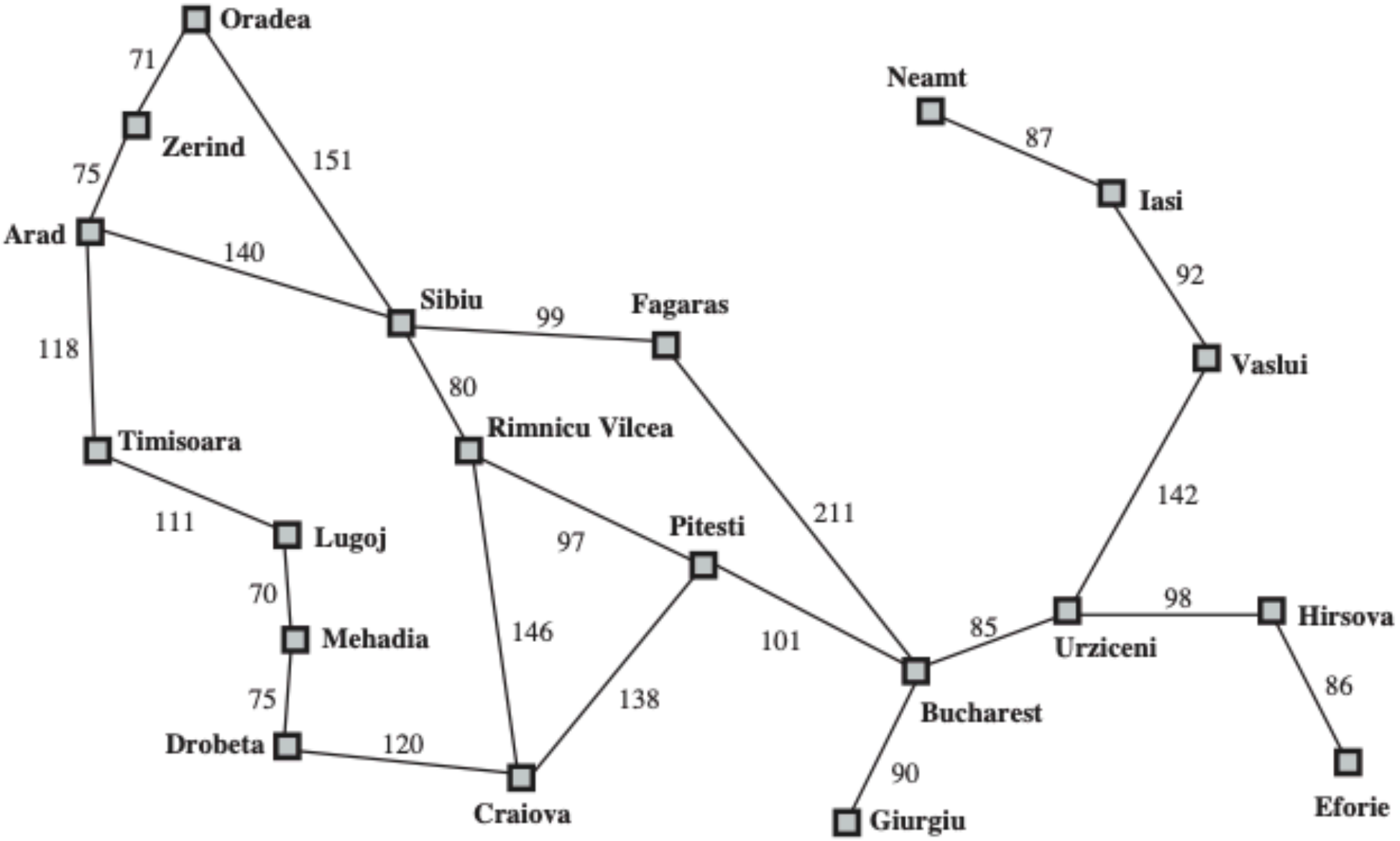
Searching For Solutions

- Have formulated problem, now need to solve it
- Search through state space
  - Using search tree
- Search node
  - Initial state
  - **Test if already goal state!!!**
- Expand current state
  - Apply successor function to generate new states

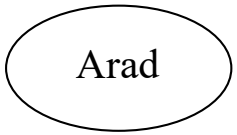
Searching For Solutions

- Have formulated problem, now need to solve it
- Search through state space
  - Using search tree
- Search node
  - Initial state
  - **Test if already goal state!!!**
- Expand current state
  - Apply successor function to generate new states
  - Which state to examine (and expand) next?
    - Use search strategy

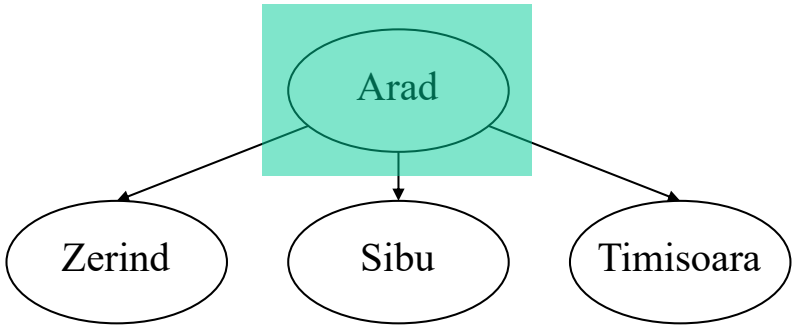
Romania Visit: Searching For Solution

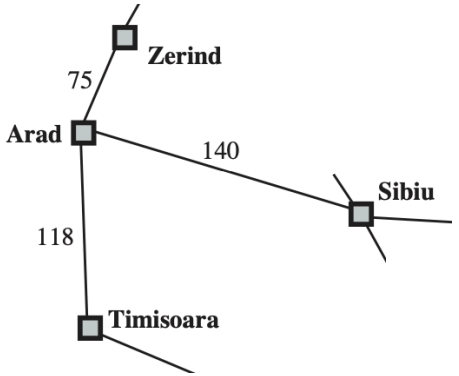


Romania Visit: Searching For Solution

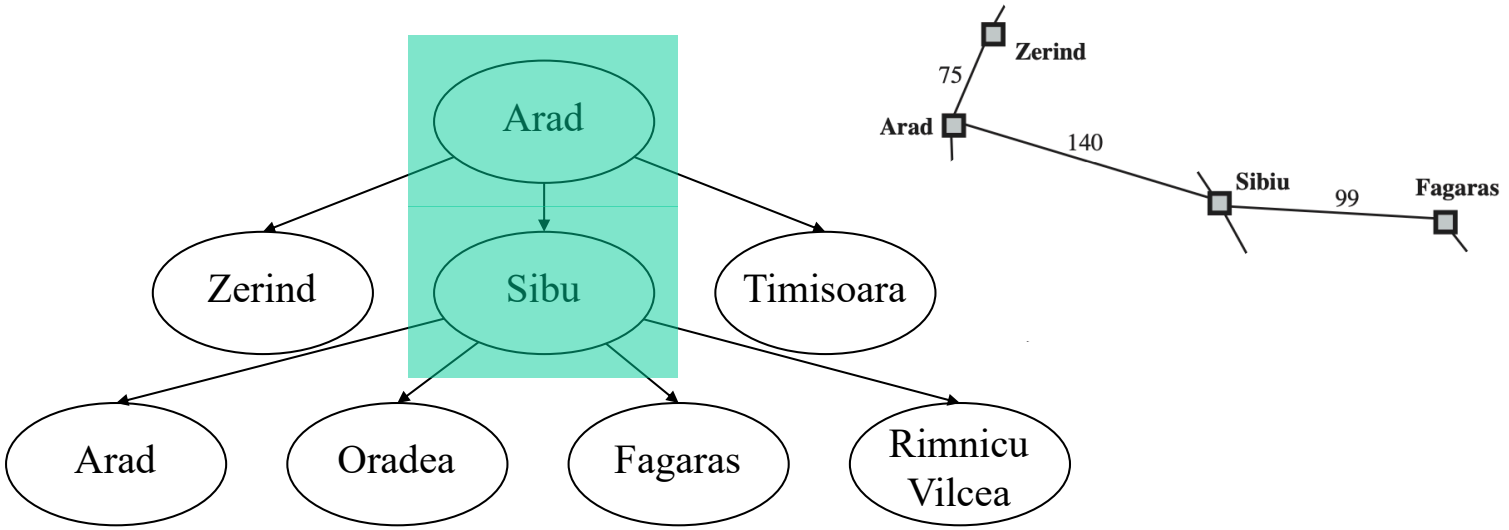


Romania Visit: Searching For Solution

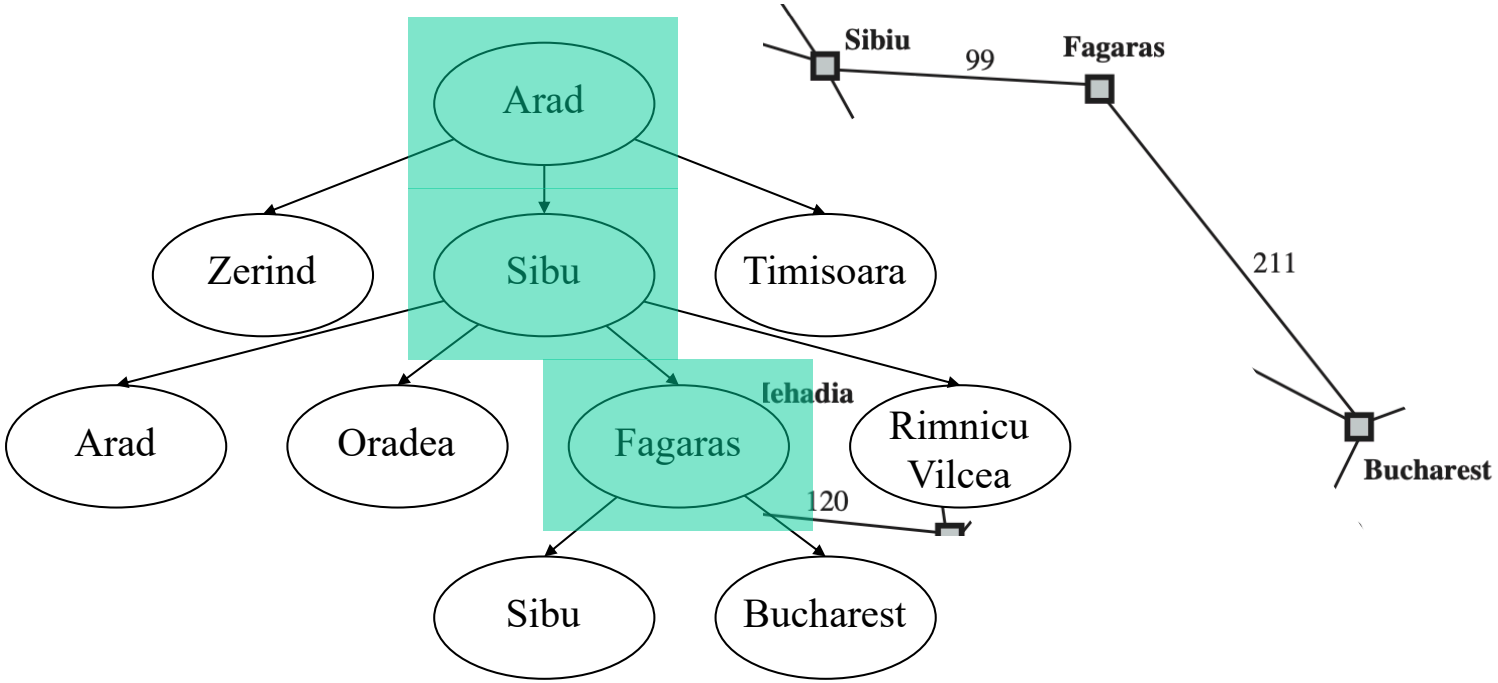




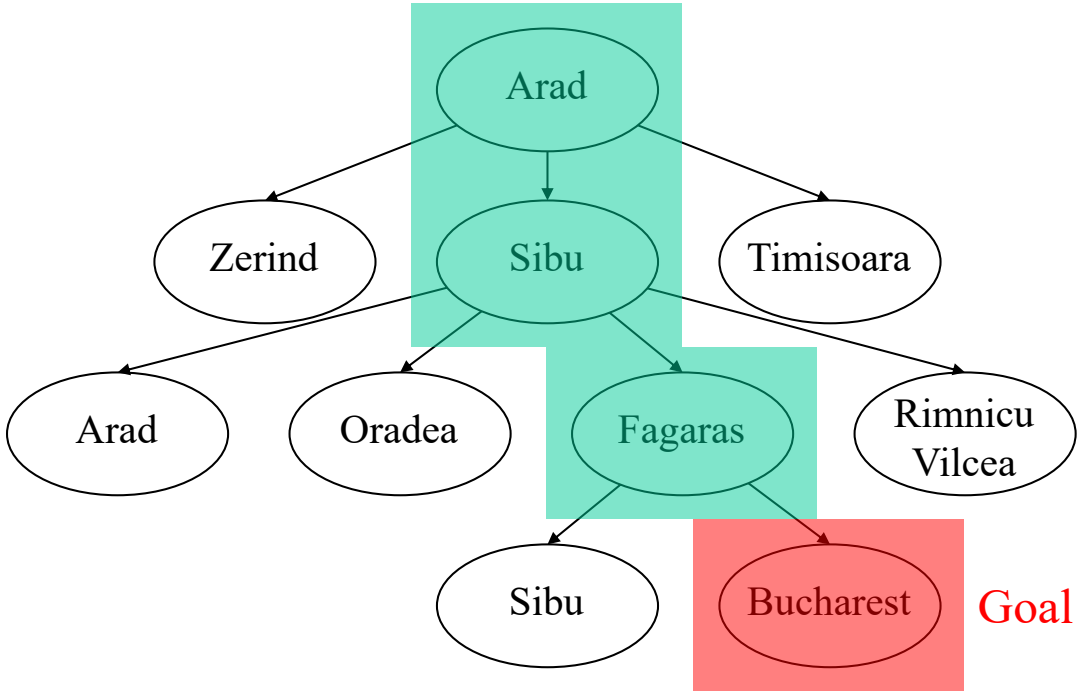
Romania Visit: Searching For Solution



Romania Visit: Searching For Solution



Romania Visit: Searching For Solution



Measuring Performance

- Completeness
  - Does it find a solution when one exists?

Measuring Performance

- Completeness
  - Does it find a solution when one exists?
- Optimality
  - Is it the solution with lowest path cost?

Measuring Performance

- Completeness
  - Does it find a solution when one exists?
- Optimality
  - Is it the solution with lowest path cost?
- Time complexity
  - How long does it take to find a solution?

Measuring Performance

- Completeness
  - Does it find a solution when one exists?
- Optimality
  - Is it the solution with lowest path cost?
- Time complexity
  - How long does it take to find a solution?
- Space complexity
  - How much memory is needed to perform the search?

QuestionS