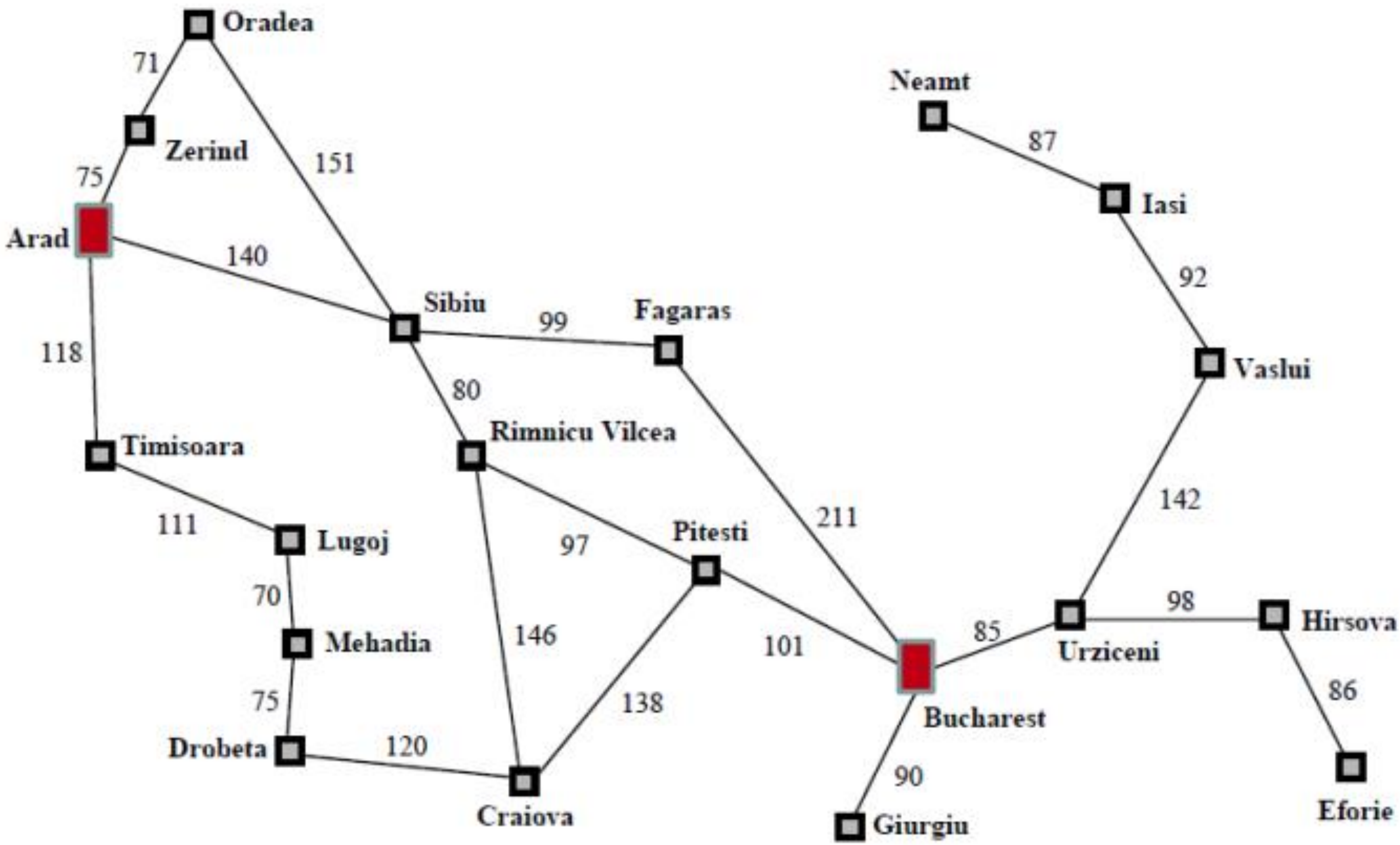
Introduction to Artificial Intelligence

Lecture: Problem Solving Using Searching

Outline

- Problem-Solving Agents
- Example Problems
- Searching for Solutions

Holiday in Romania

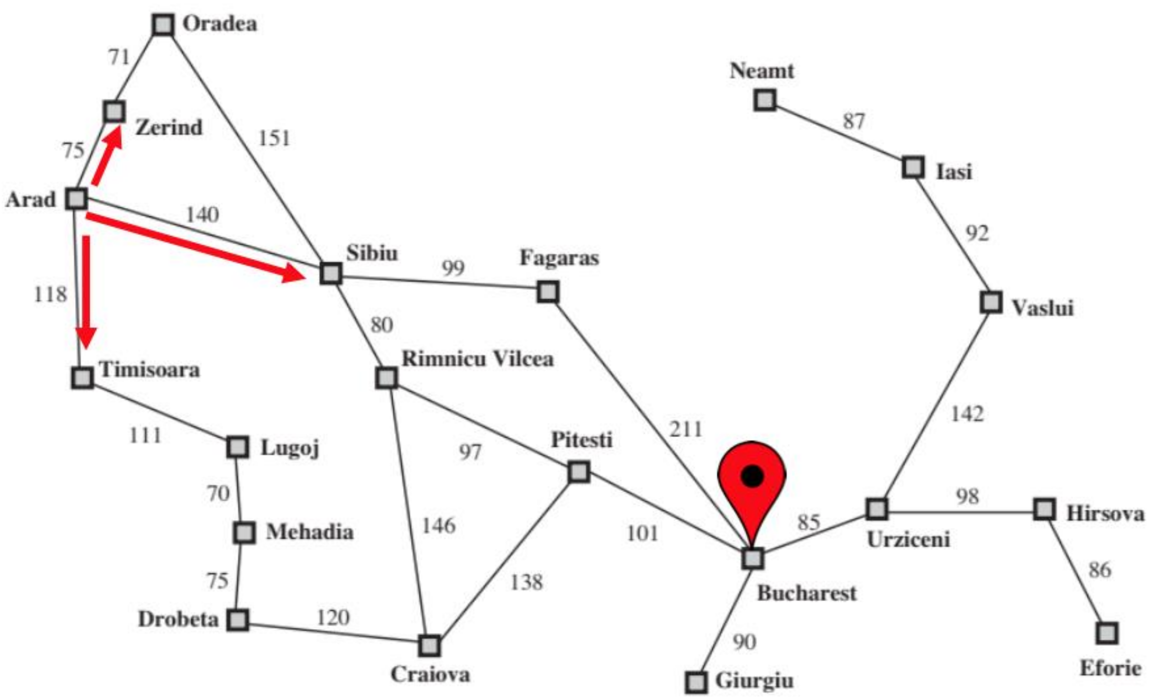


Goal-based Agents

- Intelligent agents maximize their performance measure.
- Goals help organize behavior by limiting the objectives that the agent is trying to achieve and the actions it considers.

Problem Formulation

- Consider a goal to be a set of world states in which the objective is satisfied.
- Problem formulation is the process of deciding what actions and states to consider, given a goal.



Properties of the Romania environment

- Observable
  - Each city has a sign indicating its presence for arriving drivers.
  - The agent always knows the current state.
- Discrete
  - Each city is connected to a small number of other cities.
  - There are only finitely many actions to choose from any given state.
- Known
  - The agent knows which states are reached by each action.
- Deterministic
  - Each action has exactly one outcome.

Solving problem by searching

- 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.
- Execution phase: once a solution is found, the recommended actions are carried out.

Solving problem by searching

```
function PROBLEM-SOLVING-AGENT(percept) returns an action
    persistent: seq, an action sequence, initially empty
                state, some description of the current world state
                goal, a goal, initially null
                problem, a problem formulation
    state ← UPDATE-STATE(state, percept)
    if seq is empty then
        goal ← FORMULATE-GOAL(state)
        problem ← FORMULATE-PROBLEM(state, goal)
        seq ← SEARCH(problem)
        if seq = failure then return a null action
    action ← FIRST(seq)
    seq ← REST(seq)
    return action
```

Well-defined problems and solutions

- A problem can be defined formally by five components.
  - Initial state: in which the agent starts
    - E.g., the agent in Romania has its initial state described as $In(Arad)$
  - Actions: the possible actions available to the agent
    - E.g., $ACTION(Arad)$ = {

$$Go(Sibiu),\ Go(Timisoara),\ Go(Zerind)\}$$

  - Transition model: what each action does
    - E.g., $Result(In(Arad), Go(Zerind)) = In(Zerind)$
  - Successor: a state reachable from a given state by a single action

Well-defined problems and solutions

- Goal test: determine whether a given state is a goal state
  - The goal is specified by either an explicit set of possible goal states or an abstract property.
  - E.g., $In(Bucharest)$, checkmate
- Path cost: a function that sets a numeric cost to each path
  - Non-negative, reflecting the agent's performance measure
  - E.g., $c(In(Arad), Go(Zerind), In(Zerind)) = 75$

- An optimal solution has the lowest path cost.

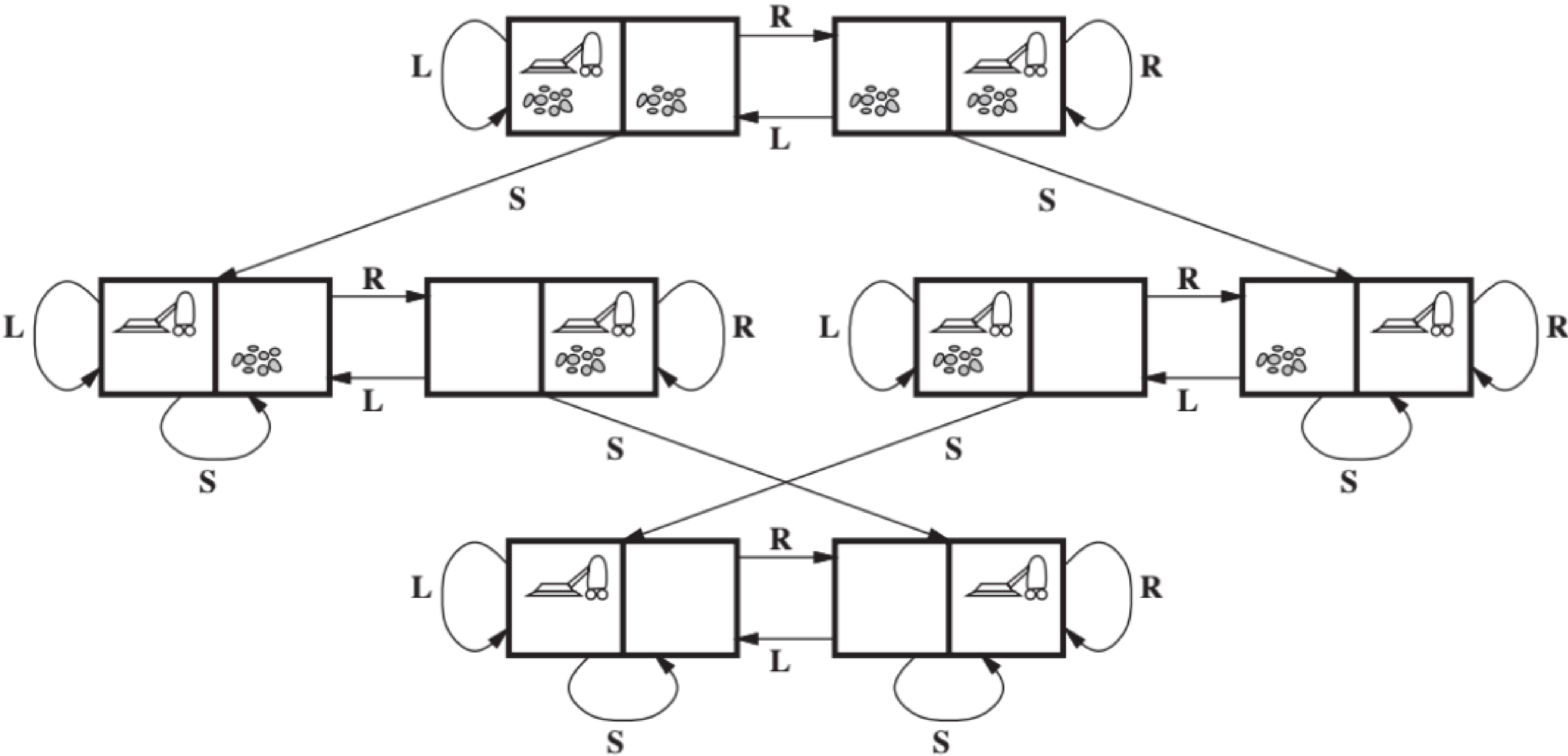
Formulating problems by abstraction

- Abstraction creates an approximate and simplified model of the real world, which is too detailed for computer.
- This is critical for automated problem solving.
- The choice of a good abstraction involves
  - Remove as much detail as possible while
  - Retain validity and ensure that the abstract actions are easy to be carried out.

The Vacuum-cleaner world

- States: determined by both the agent location and the dirt locations
  - $2 \times 2^2 = 8$ possible world states ($n \times 2^n$ in general)
- Initial state: Any state can be designated as the initial state.
- Actions: Left, Right, and Suck
- Transition model: The actions have their expected effects.
- Goal test: whether all the squares are clean
- Path cost: each step costs 1

The Vacuum-cleaner world



The 8-puzzle

- States: the location of each of the eight tiles and the blank
- Initial state: any state can be designated as the initial state
- Actions: movements of the blank space
  - Left, Right, Up, or Down.
  - Different subsets of these are possible depending on where the blank is
- Transition model: return a resulting state given a state and an action
- Goal test: whether the state matches the goal configuration
- Path cost: each step costs 1

The 8-puzzle

1	8	2
	4	3
7	6	5

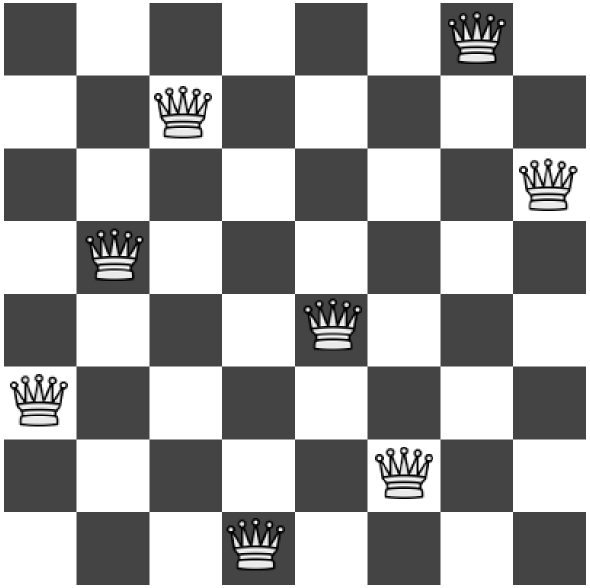
Given State

1	2	3
4	5	6
7	8	

Goal State

The 8-queens

- Incremental formulation: add a queen step-by-step to the empty initial state
- Complete-state formulation: start with all 8 queens on the board and move them around
- The path cost is trivial because only the final state counts



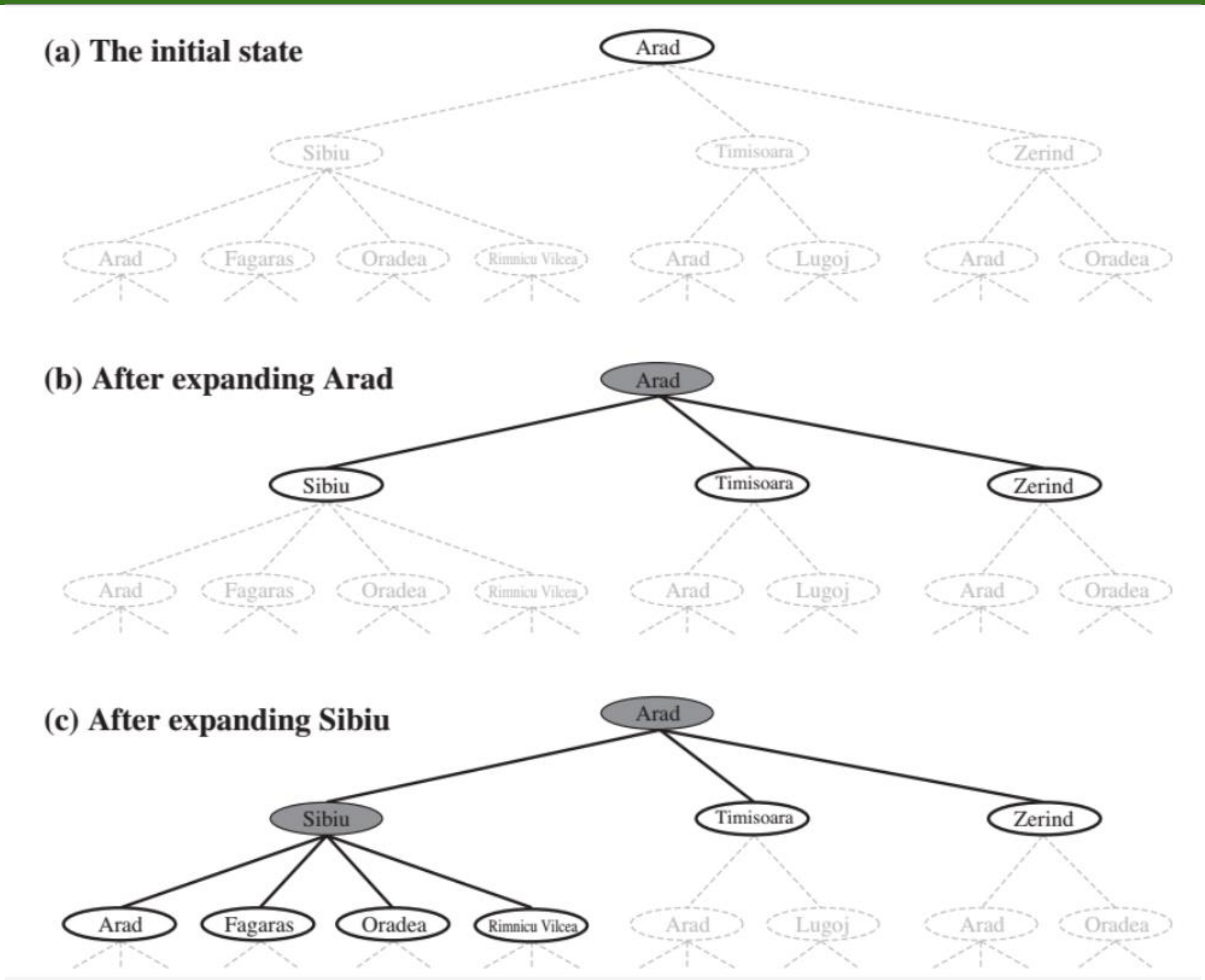
The 8-queens

- States: any arrangement of 0 to 8 queens on the board
- Initial state: no queens on the board
- Actions: add a queen to any empty square
- Transition model: returns the board with a queen added to the specified square
- Goal test: 8 queens are on the board, none attacked
- $64 \cdot 63 \cdots 57 \approx 1.8 \times 10^{14}$ possible sequences to investigate

Search Tree

- Search algorithms consider many possible action sequences to find the solution sequence.
- Search tree: the possible action sequences starting at the initial state (root)
    - Branches are actions and nodes are states in the problem's state space
- Frontier: the set of all leaf nodes available for expansion at any given point
- Search algorithms all share the basic structure while vary according to how they choose which state to expand next -- called **search strategy**.

Search Tree


(a) The initial state
Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea
(b) After expanding Arad
Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea

Search Tree

function TREE-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

loop do

if the frontier is empty **then** return failure

choose a leaf node and remove it from the frontier

if the node contains a goal state **then** **return** the corresponding solution

expand the chosen node, adding the resulting nodes to the frontier

Redundant paths

- Redundant paths are unavoidable.
- Following redundant paths may cause a tractable problem to become intractable.
- This is true even for algorithms that know how to avoid infinite loops.

Graph Search

function GRAPH-SEARCH(problem) **returns** a solution, or failure

initialize the frontier using the initial state of problem

initialize the explored set to be empty

loop do

if the frontier is empty **then** return failure

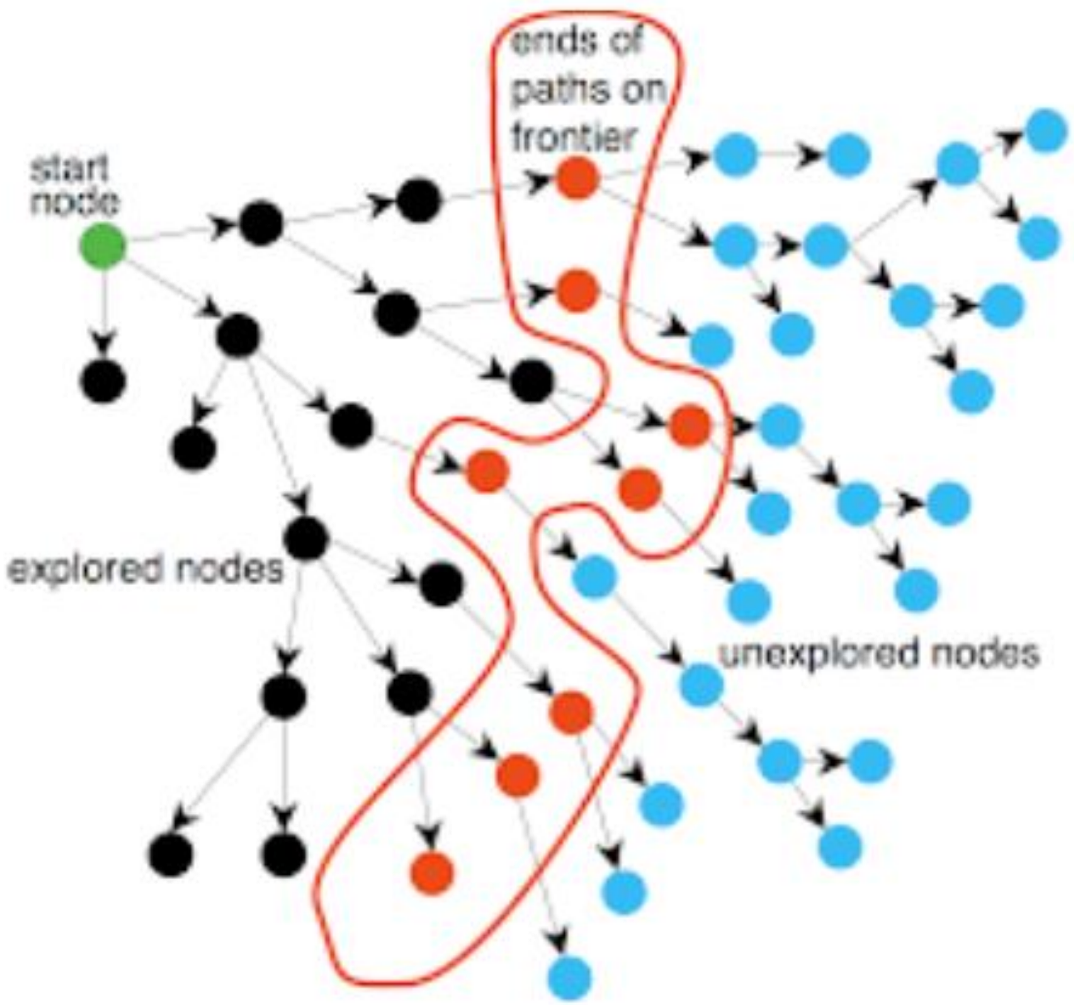
choose a leaf node and remove it from the frontier

if the node contains a goal state **then** **return** the corresponding solution

add the node to the explored set

expand the chosen node, adding the resulting nodes to the frontier ***only if not in the frontier nor explored set***

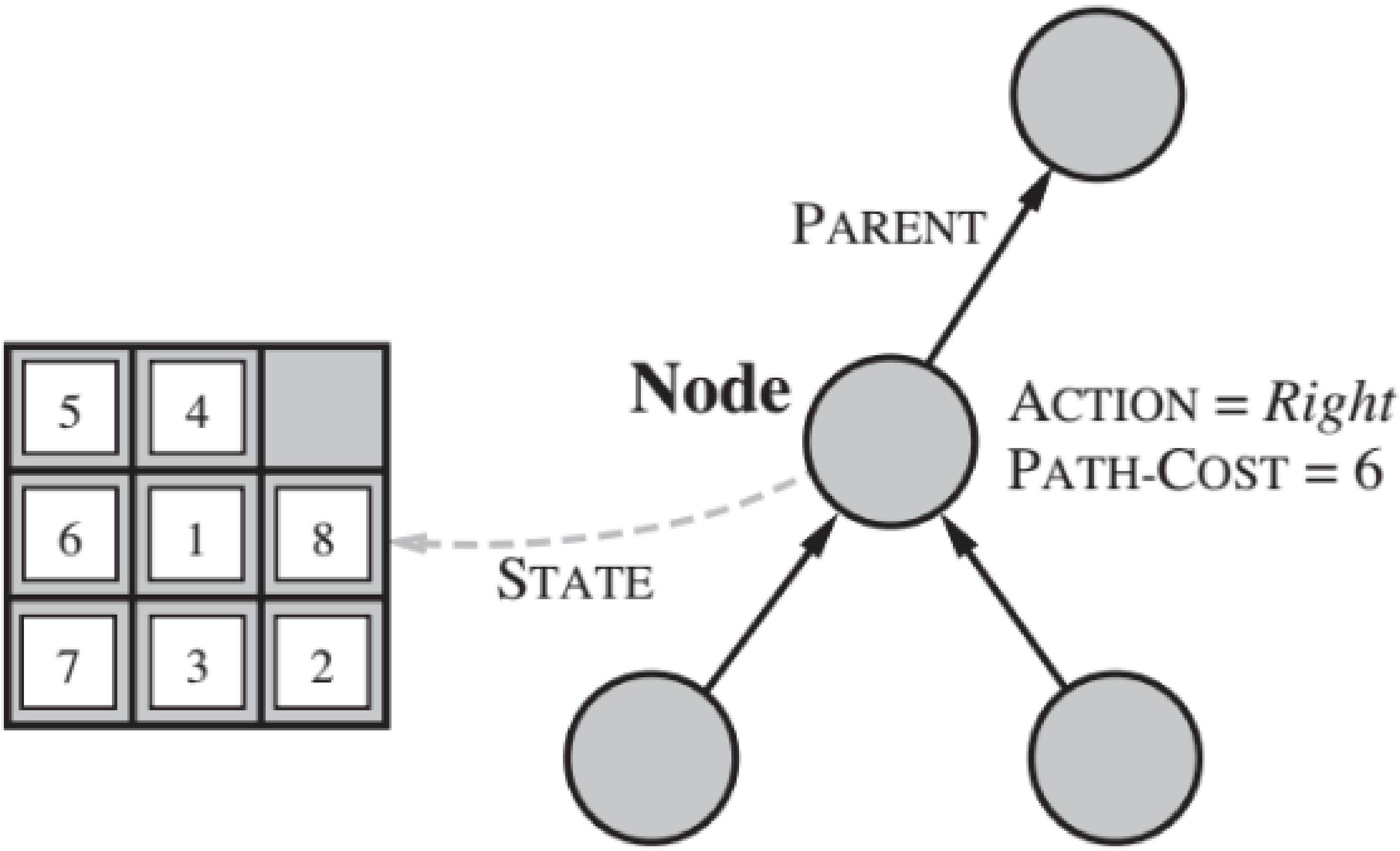
Graph Search



Infrastructure for search algorithms

- Each node n is structuralized by four components.
  - $\boldsymbol{n}.\ \mathbf{STATE}$: the state in the state space to which the node corresponds
  - $\boldsymbol{n}.\ \mathbf{PARENT}$: the node in the search tree that generated the node n
  - $\boldsymbol{n}.\ \mathbf{ACTION}$: the action applied to the parent to generate n
  - $\boldsymbol{n}.\ \mathbf{PATH-COST}$: the cost, denoted by $\boldsymbol{g(n)}$, of the path from the initial state to the node, as indicated by the parent pointer
- Frontier can be implemented with a (priority) queue or stack.
- Explored set can be a hash table that allows for efficient checking of repeated states.

Infrastructure for search algorithms



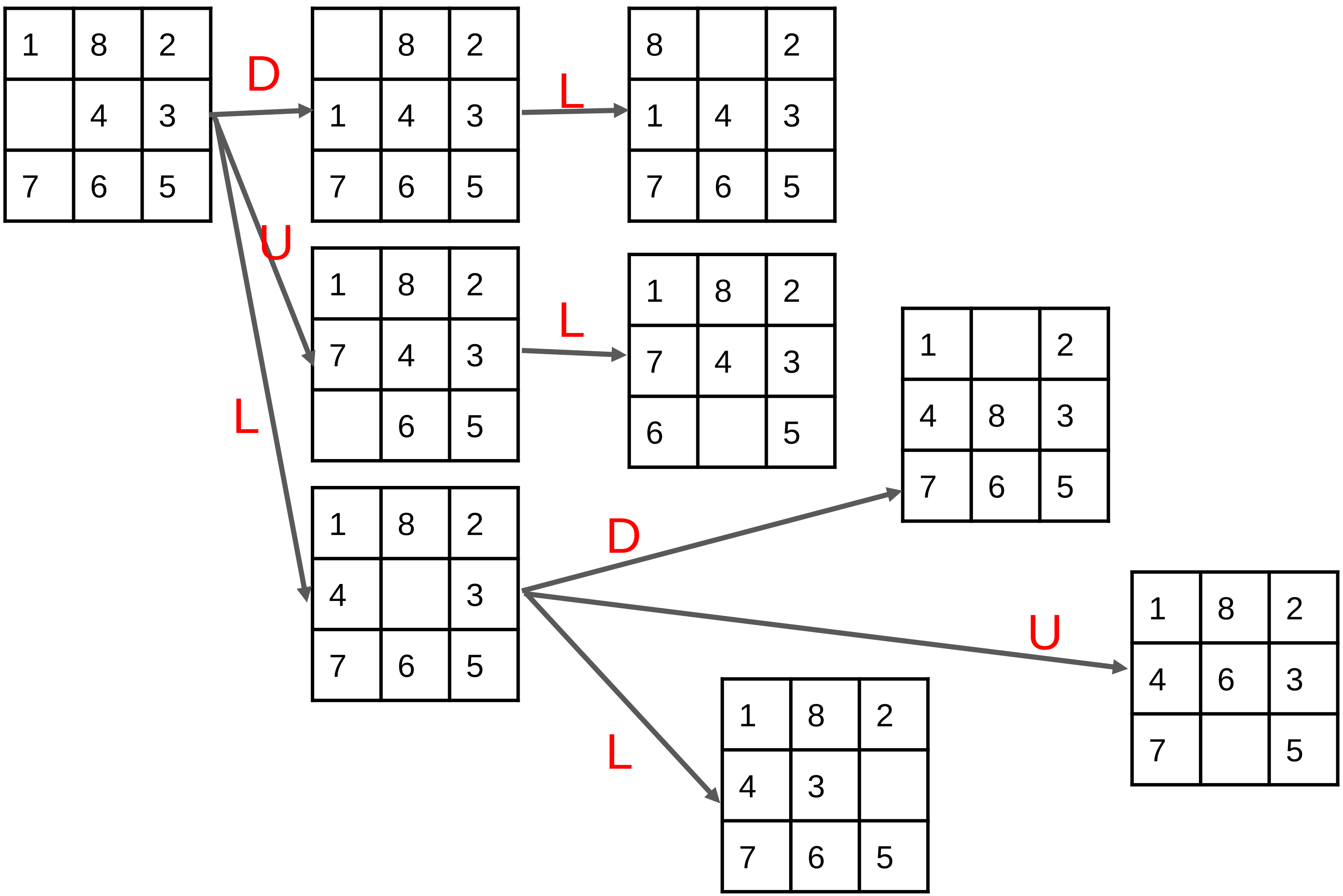
Problem-solving performance

- **Completeness**: does it always find a solution if one exists?
- **Time complexity**: how long does it take to find a solution?
- **Space complexity**: how much memory is needed to perform the search?
- **Optimality**: does it always find a least-cost solution?

Homework

- Task 1:
  - Given the start state and the goal one of the 8-puzzle in page 15.
  - Students draw a search tree by the expanding nodes.
  - The expansion stops when the goal state is expanded.
- Task 2:
  - Program Task 1 in Python using Google Colab
  - Use graph search instead of tree search

Homework



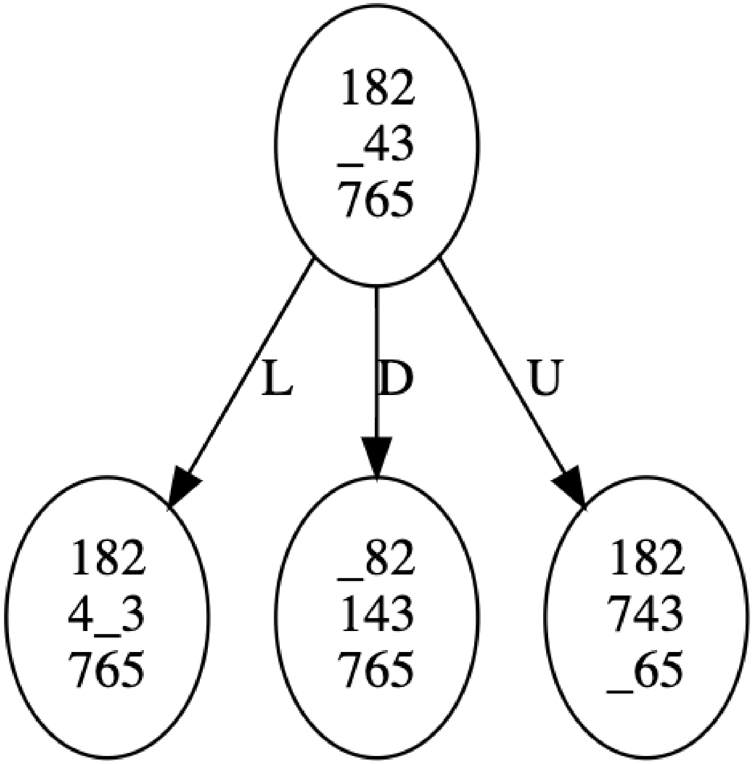
1 8 2
4 3
7 6 5
D
8 2
1 4 3
7 6 5
L
8 2
1 4 3
7 6 5
U
1 8 2
7 4 3
6 5
L
1 8 2
7 4 3
6 5
L
1 8 2
4 3
7 6 5
D
1 2
4 8 3
7 6 5
U
1 8 2
4 6 3
7 5
L
1 8 2
4 3
7 6 5

Homework

```
from graphviz import Digraph

dot = Digraph()
dot.node('0', '182\n_43\n765')
dot.node('1', '182\n4_3\n765')
dot.node('2', '_82\n143\n765')
dot.node('3', '182\n743\n_65')
dot.edge('0', '1', 'L')
dot.edge('0', '2', 'D')
dot.edge('0', '3', 'U')

dot
```



Homework

- Conduct homework in the given notebook

References

- Stuart Russell and Peter Norvig. 2009. Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.
- Lê Hoài Bắc, Tô Hoài Việt. 2014. Giáo trình Cơ sở Trí tuệ nhân tạo. Khoa Công nghệ Thông tin. Trường ĐH Khoa học Tự nhiên, ĐHQG-HCM.
- Nguyễn Ngọc Thảo, Nguyễn Hải Minh. 2020. Bài giảng Cơ sở Trí tuệ Nhân tạo. Khoa Công nghệ Thông tin. Trường ĐH Khoa học Tự nhiên, ĐHQG-HCM.