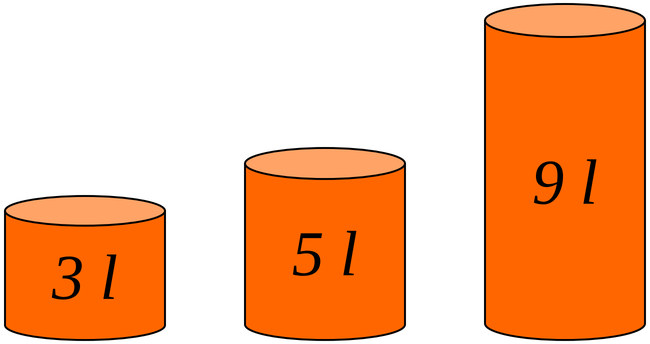
Introduction to AI

Lecture 4: Problem solving and search

Mona Taghavi

LaSalle College
Montréal

Example: Measuring problem!

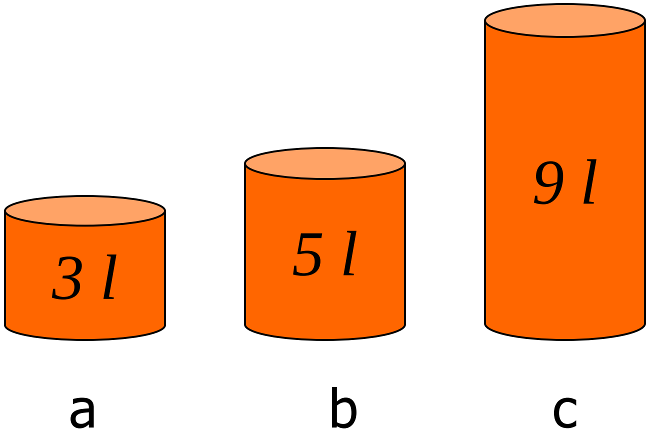


Problem: Using these three buckets, measure 7 liters of water.

Example: Measuring problem!

- **(one possible) Solution:**

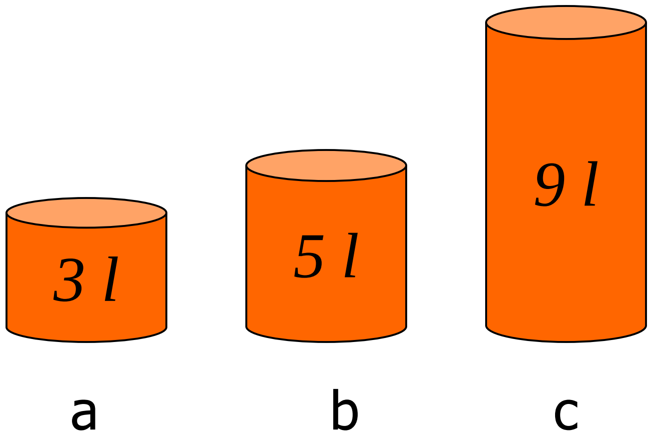
	a	b	c
start	0	0	0



Example: Measuring problem!

- **(one possible) Solution:**

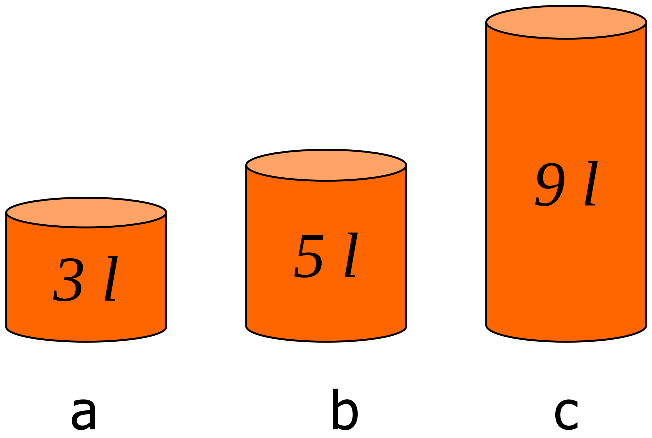
a	b	c	
0	0	0	start
3	0	0	



Example: Measuring problem!

- **(one possible) Solution:**

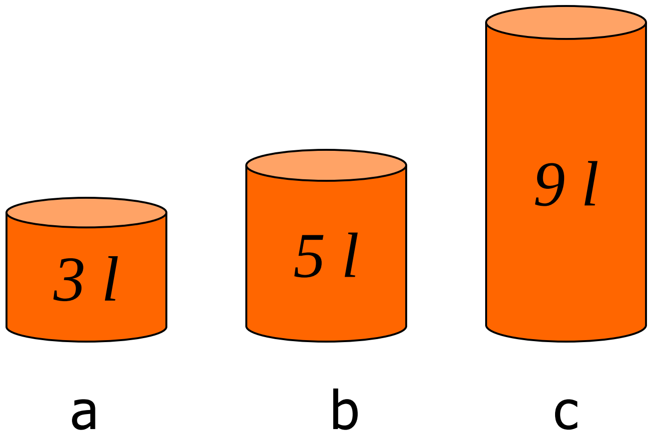
a	b	c	
0	0	0	start
3	0	0	
0	0	3	



Example: Measuring problem!

- **(one possible) Solution:**

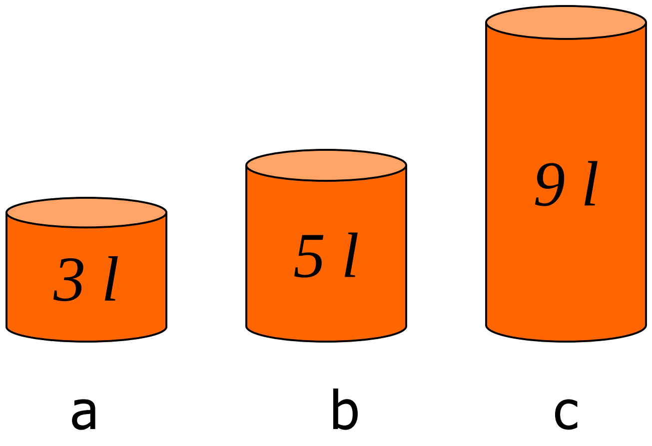
a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	



Example: Measuring problem!

- **(one possible) Solution:**

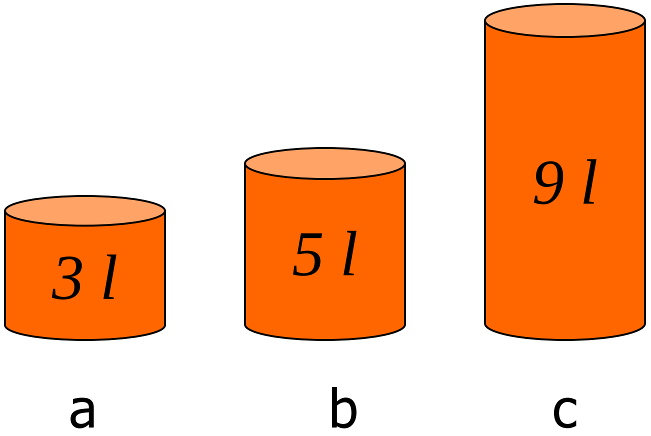
a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	



Example: Measuring problem!

- **(one possible) Solution:**

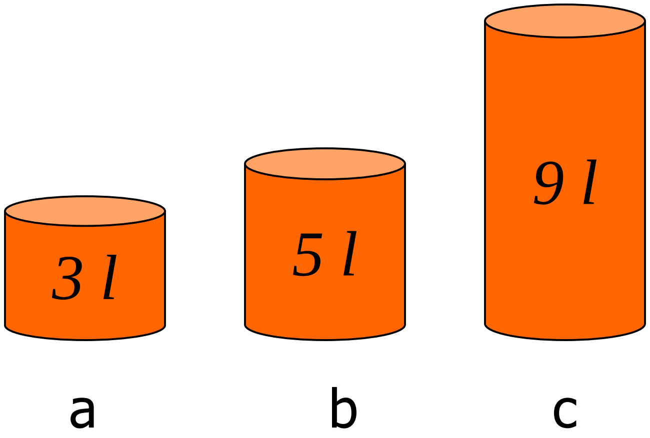
a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	



Example: Measuring problem!

- **(one possible) Solution:**

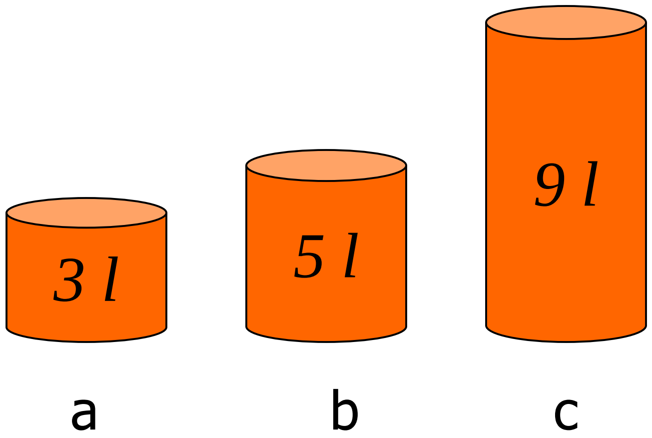
a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	



Example: Measuring problem!

- **(one possible) Solution:**

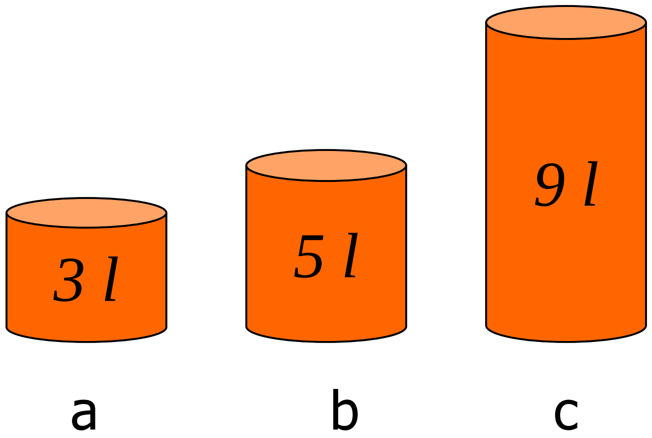
a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	



Example: Measuring problem!

- **(one possible) Solution:**

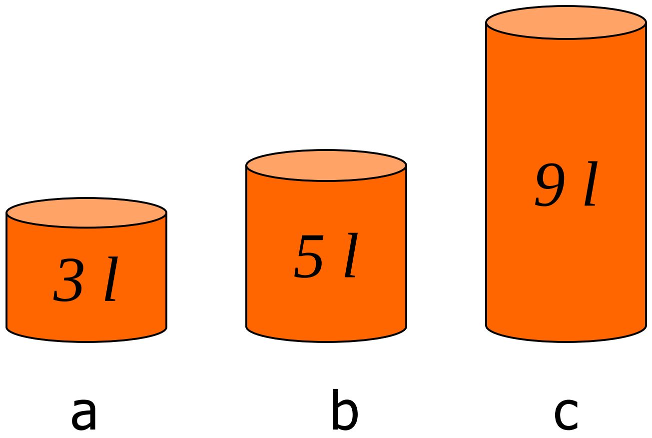
a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	
1	5	6	



Example: Measuring problem!

- **(one possible) Solution:**

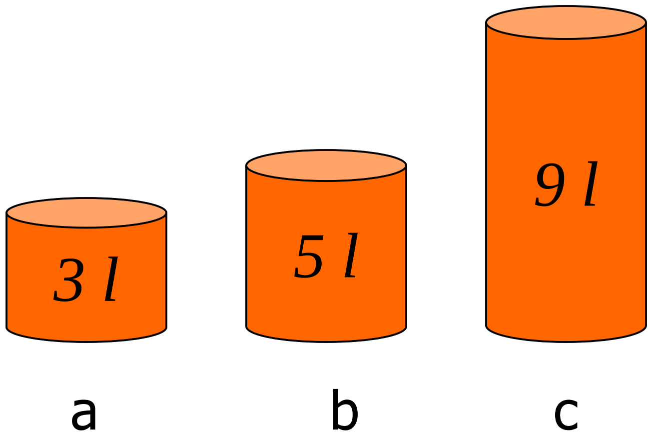
a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	
1	5	6	
0	5	**7**	**goal**



Example: Measuring problem!

- **Another Solution:**

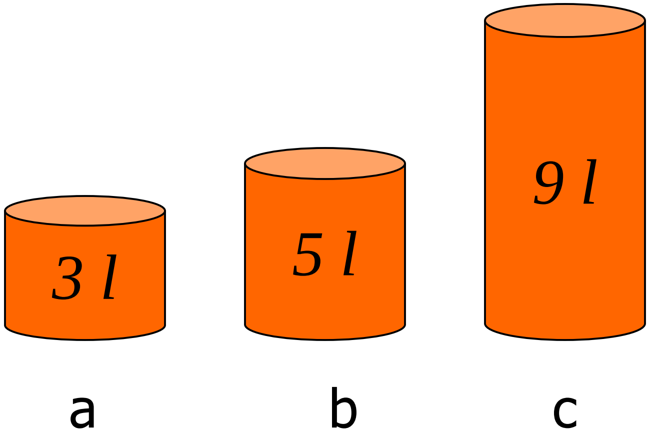
a	b	c	
0	0	0	start
0	5	0	



Example: Measuring problem!

- **Another Solution:**

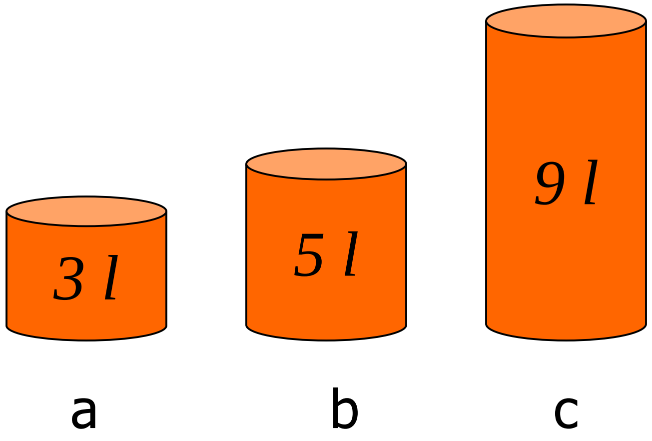
a	b	c	
0	0	0	start
0	5	0	
3	2	0	



Example: Measuring problem!

- **Another Solution:**

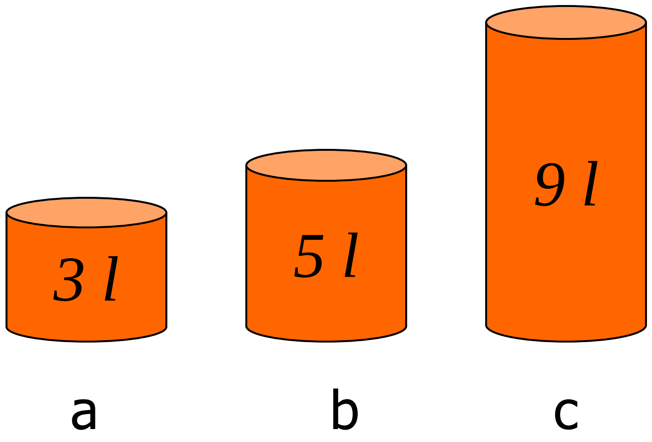
a	b	c	
0	0	0	start
0	5	0	
3	2	0	
3	0	2	



Example: Measuring problem!

- **Another Solution:**

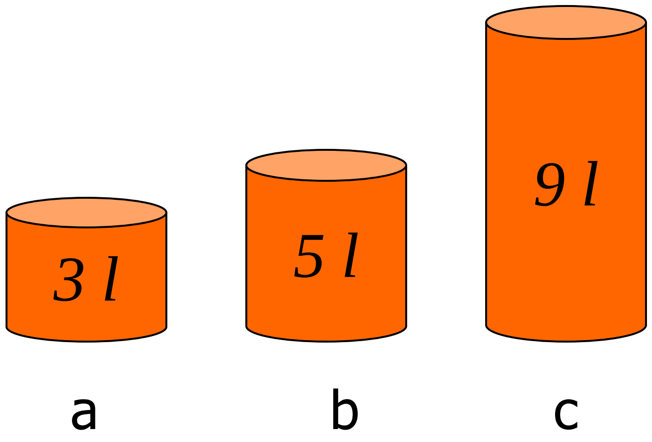
a	b	c	
0	0	0	start
0	5	0	
3	2	0	
3	0	2	
3	5	2	



Example: Measuring problem!

- **Another Solution:**

a	b	c	
0	0	0	start
0	5	0	
3	2	0	
3	0	2	
3	5	2	
3	**0**	**7**	**goal**



Which solution do we prefer?

- **Solution 1:**

a	b	c	
0	0	0	start
3	0	0	
0	0	3	
3	0	3	
0	0	6	
3	0	6	
0	3	6	
3	3	6	
1	5	6	
0	5	**7**	**goal**

- **Solution 2:**

a	b	c	
0	0	0	start
0	5	0	
3	2	0	
3	0	2	
3	5	2	
3	**0**	**7**	**goal**

Problem Solving Agent

Measure 7 liters of water using a 3-liter, a 5-liter, and a 9-liter buckets.

- **Formulate goal:** Have 7 liters of water in 9-liter bucket

- **Formulate problem:**
  - States: amount of water in the buckets
  - Operators: Fill bucket from source, empty bucket

- **Find solution:** sequence of operators that bring you from current state to the goal state

Remember (lecture 3): Environment types

Environment	Accessible	Deterministic	Episodic	Static	Discrete
Operating System	Yes	Yes	No	No	Yes
Virtual Reality	Yes	Yes	Yes/No	No	Yes/No
Office Environment	No	No	No	No	No
Mars	No	Semi	No	Semi	No

The environment types largely determine the agent design.

Problem types

- **Single-state problem:** deterministic, accessible

  Agent knows everything about world, thus can calculate optimal action sequence to reach goal state.

- **Multiple-state problem:** deterministic, inaccessible

  Agent must reason about sequences of actions and states assumed while working towards goal state.

- **Contingency problem:** nondeterministic, inaccessible
  - *Must use sensors during execution*
  - *Solution is a tree or policy*
  - *Often interleave search and execution*

- **Exploration problem:** unknown state space

  Discover and learn about environment while taking actions.

Problem types

- **Single-state problem:** deterministic, accessible
  - Agent knows everything about world (the exact state),
  - Can calculate optimal action sequence to reach goal state.
  - E.g., playing chess. Any action will result in an exact state

Problem types

- **Multiple-state problem:** deterministic, inaccessible
  - Agent does not know the exact state (could be in any of the possible states)
    - May not have sensor at all
  - Assume states while working towards goal state.
  - E.g., walking in a dark room
    - If you are at the door, going straight will lead you to the kitchen
    - If you are at the kitchen, turning left leads you to the bedroom
    - …

Problem types

- **Contingency problem:** nondeterministic, inaccessible
  - Must use sensors during execution
  - Solution is a tree or policy
  - Often interleave search and execution
  - E.g., a new skater in an arena
    - Sliding problem.
    - Many skaters around

Problem types

- **Exploration problem:** unknown state space

Discover and learn about environment while taking actions.

  - *E.g., Maze*

Example: Romania

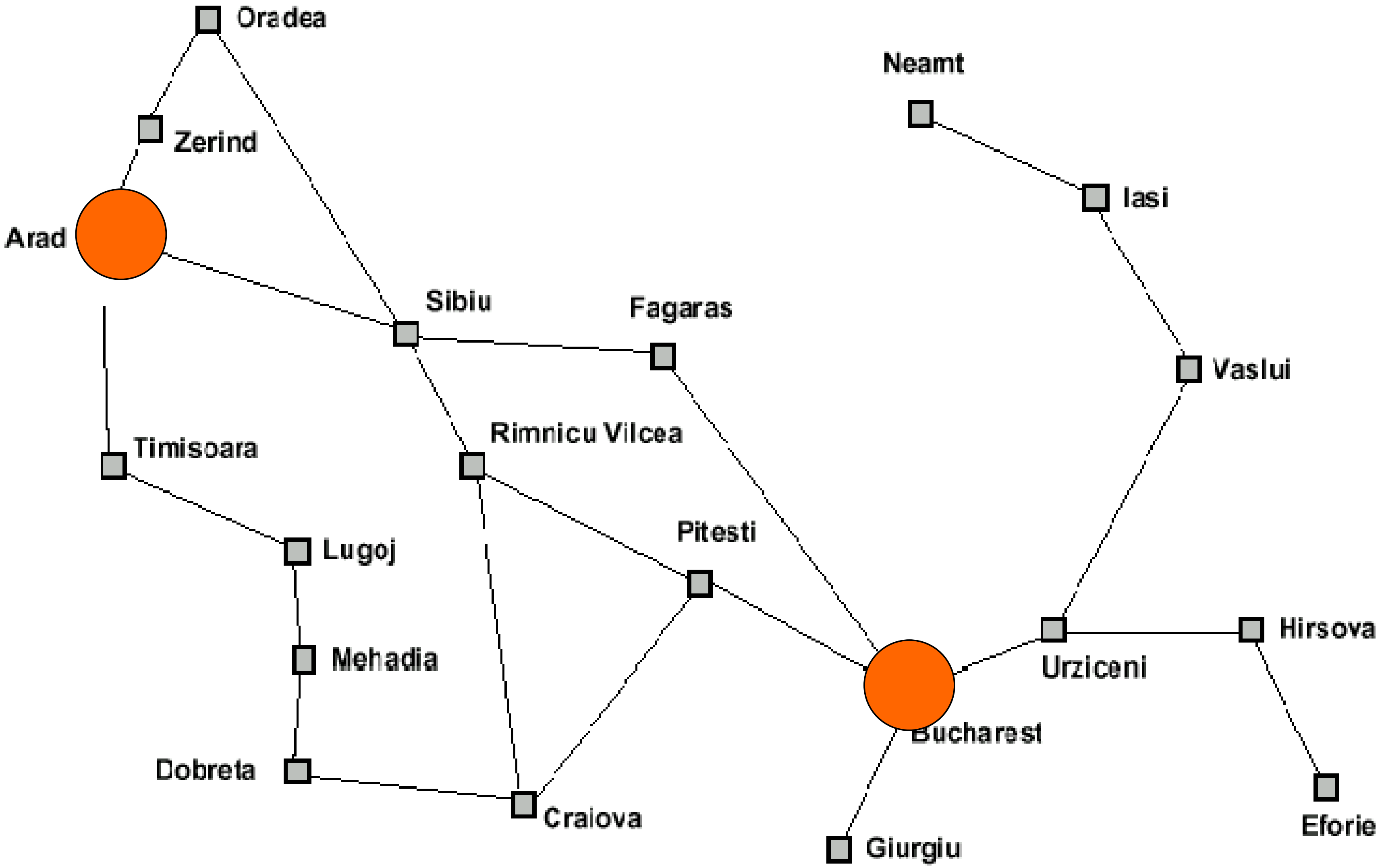
- In Romania, on vacation. Currently in Arad.
- Flight leaves tomorrow from Bucharest.

- **Formulate goal:**
  - be in Bucharest

- **Formulate problem:**
  - states: various cities
  - operators: drive between cities

- **Find solution:**
  - sequence of cities, such that total driving distance is minimized.

Example: Traveling from Arad To Bucharest



Problem formulation

A *problem* is defined by four items:

<u>*initial state*</u> e.g., "at Arad"

<u>*operators*</u> (or *successor function* $S(x)$)
e.g., Arad $\rightarrow$ Zerind Arad $\rightarrow$ Sibiu etc.

<u>*goal test*</u>, can be
explicit, e.g., $x =$ "at Bucharest"
implicit, e.g., $NoDirt(x)$

<u>*path cost*</u> (additive)
e.g., sum of distances, number of operators executed, etc.

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

Selecting a state space

- Real world is absurdly complex; some abstraction is necessary to allow us to reason on it…

- Selecting the correct abstraction and resulting state space is a difficult problem!

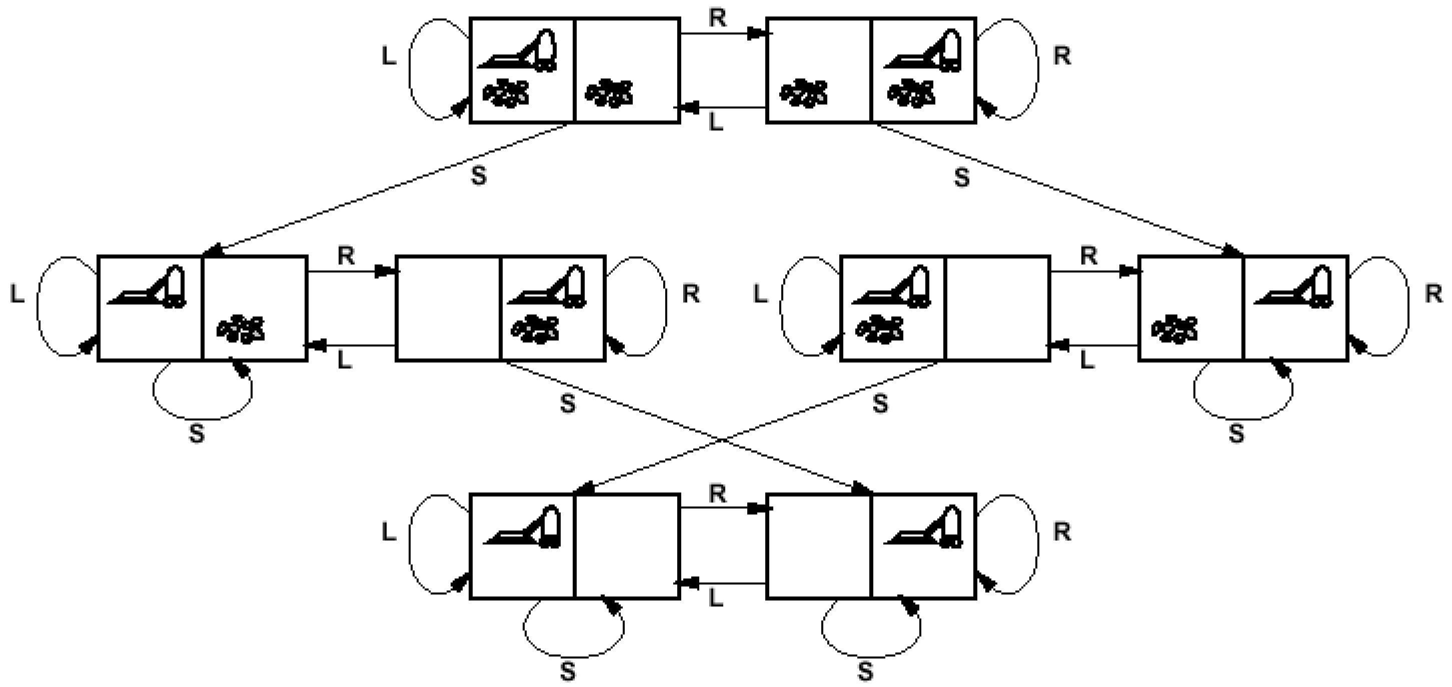
- Abstract states ⇔ real-world states

- Abstract operators ⇔ sequences or real-world actions

  (e.g., going from city i to city j costs Lij ⇔ actually drive from city i to j)

- Abstract solution ⇔ set of real actions to take in the real world such as to solve problem

Vacuum World



states??

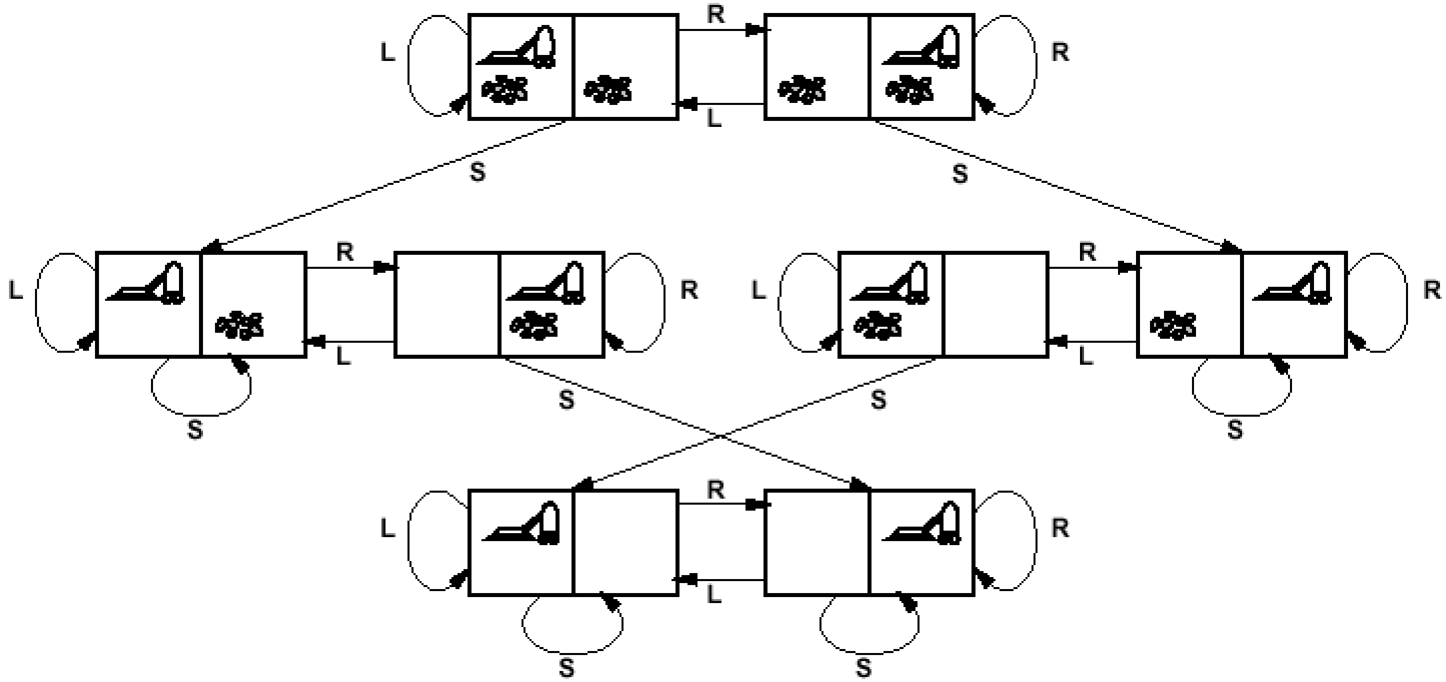
operators??

goal test??

path cost??

Simplified world: 2 locations, each may or not contain dirt, each may or not contain vacuuming agent.

Vacuum World



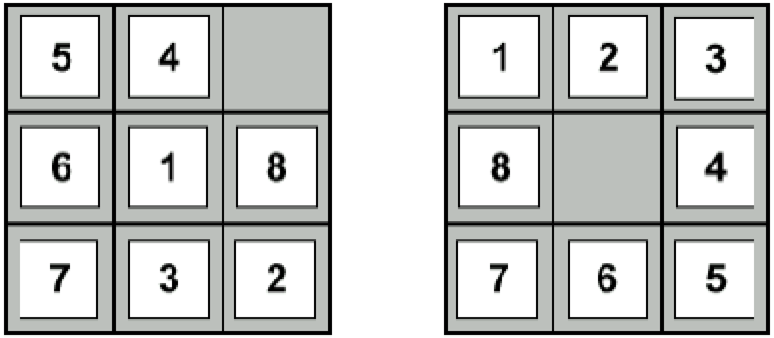
states??: integer dirt and robot locations (ignore dirt *amounts*)

operators??: *Left*, *Right*, *Suck*

goal test??: no dirt

path cost??: 1 per operator

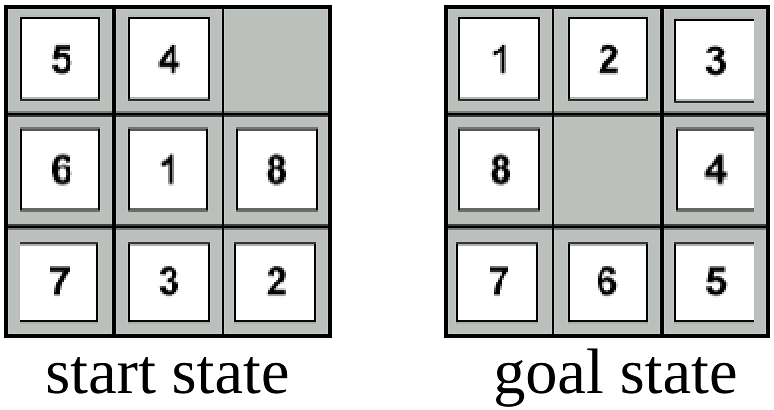
Example: 8-puzzle



start state goal state

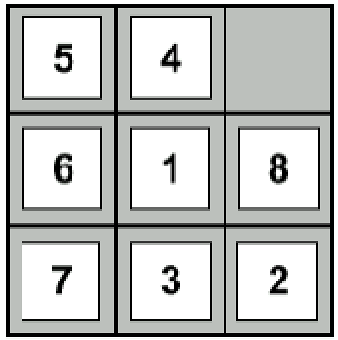
- State:
- Operators:
- Goal test:
- Path cost:

Example: 8-puzzle

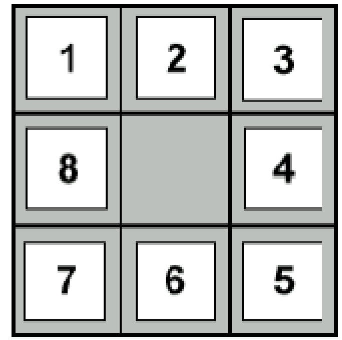


- State: integer location of tiles (ignore intermediate locations)
- Operators: moving blank left, right, up, down (ignore jamming)
- Goal test: does state match goal state?
- Path cost: 1 per move

Example: 8-puzzle



start state



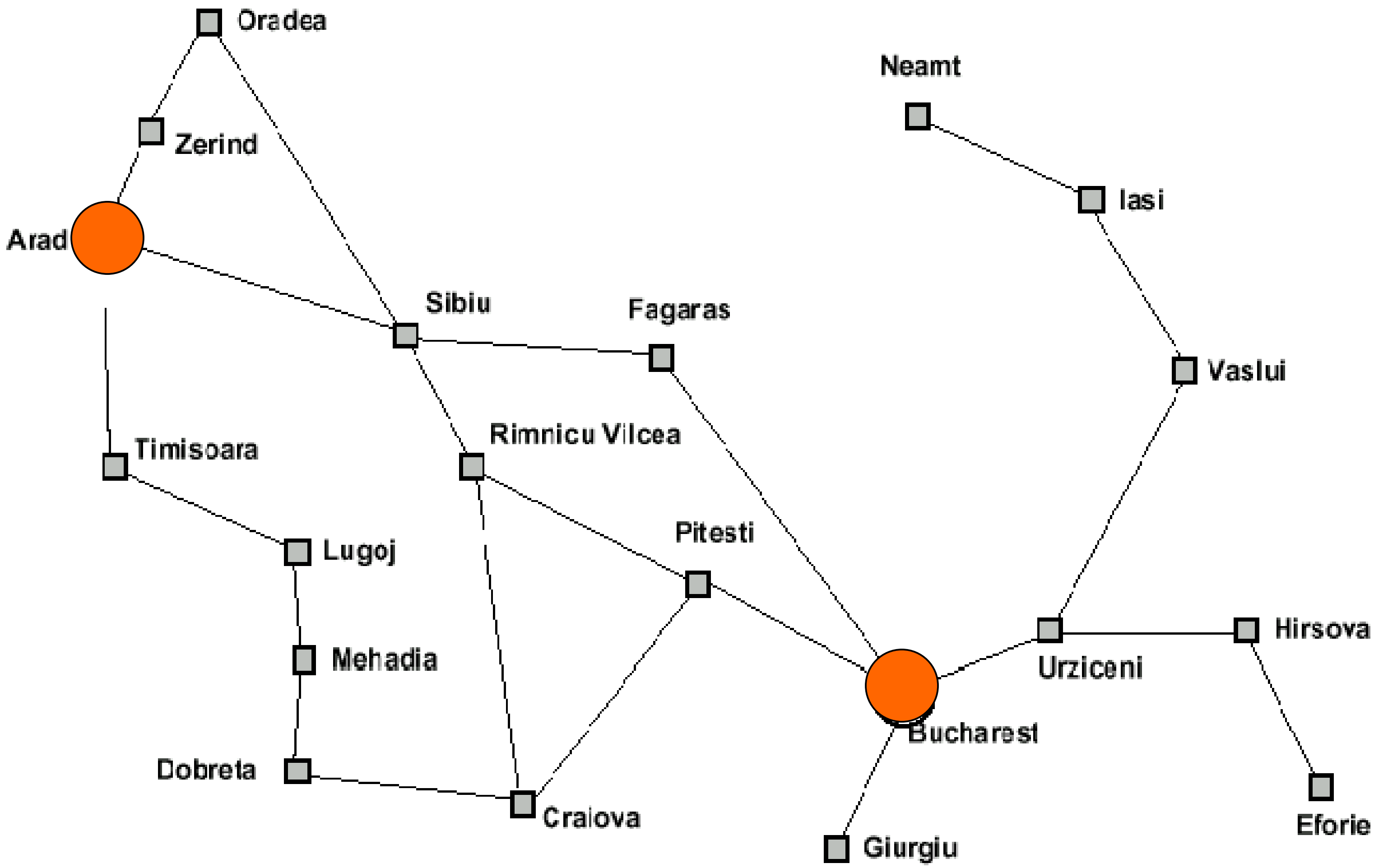
goal state

Why search algorithms?

- 8-puzzle has 362,800 states
- 15-puzzle has 10^12 states
- 24-puzzle has 10^25 states

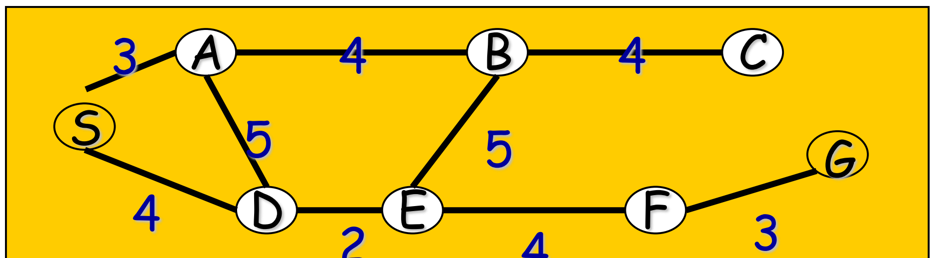
So, we need a principled way to look for a solution in these huge search spaces…

Example: Traveling from Arad To Bucharest

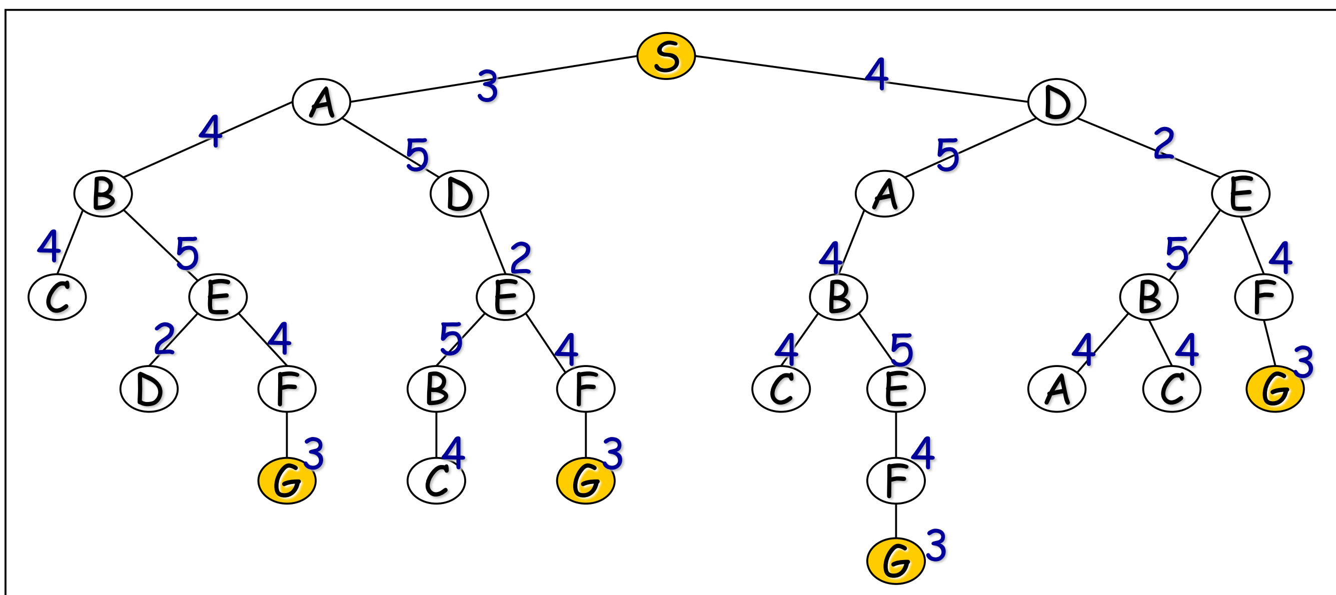


From problem space to search tree

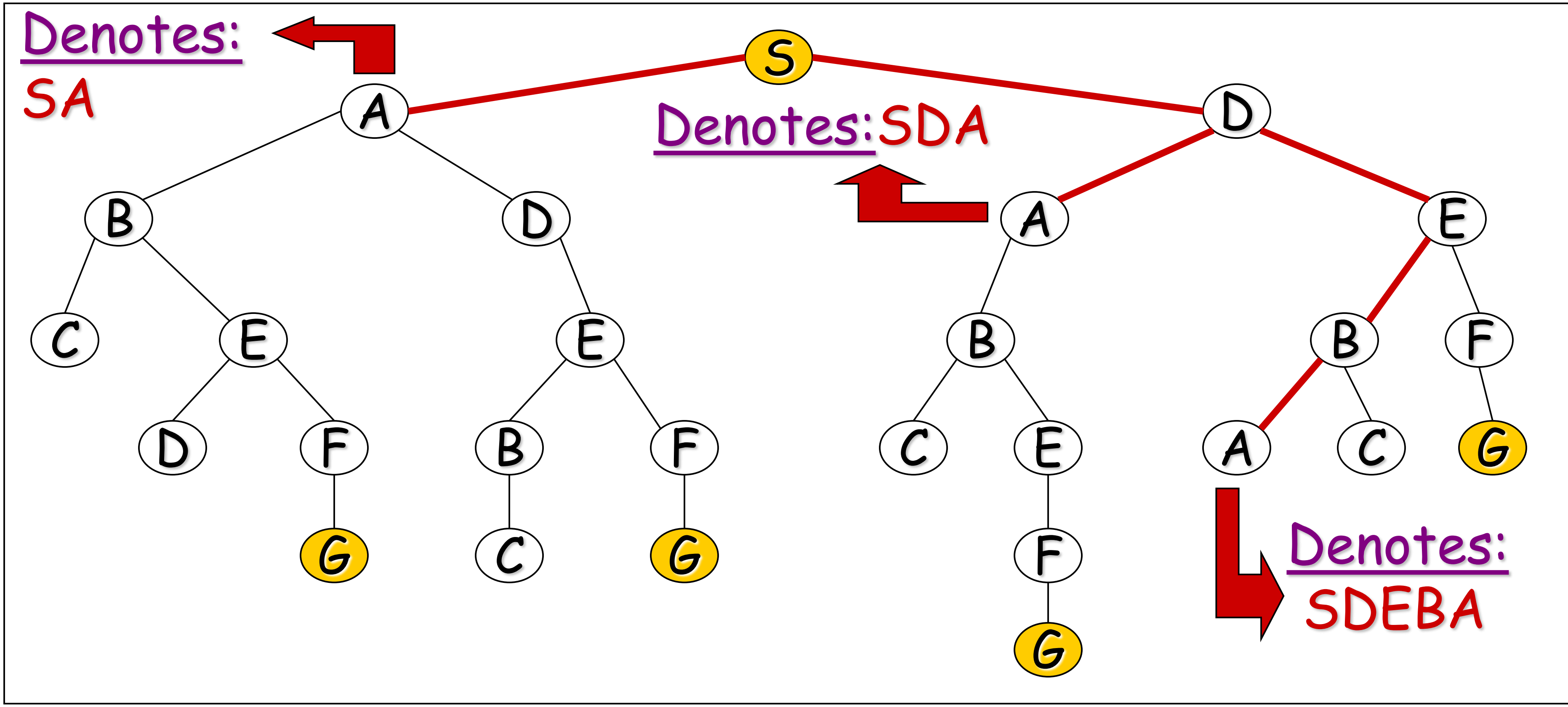
Problem space



Associated loop-free search tree

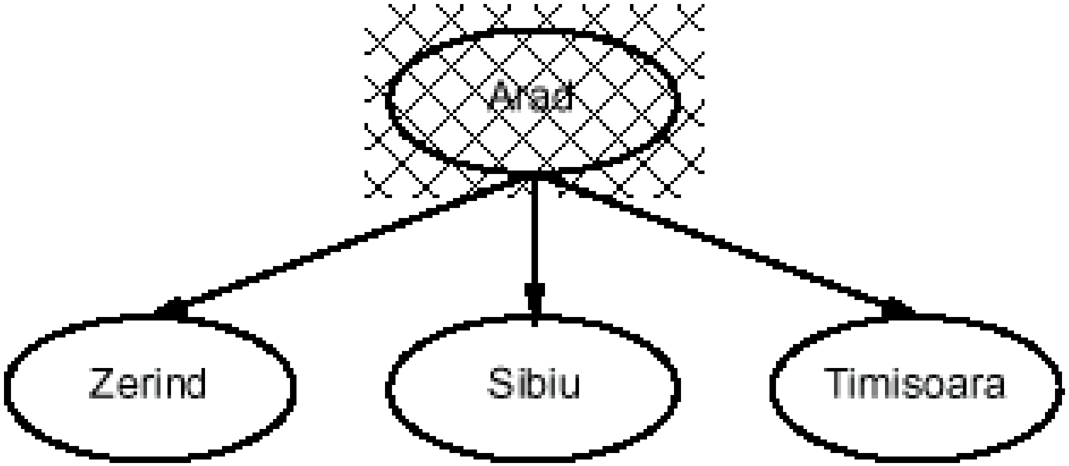


Paths in search trees

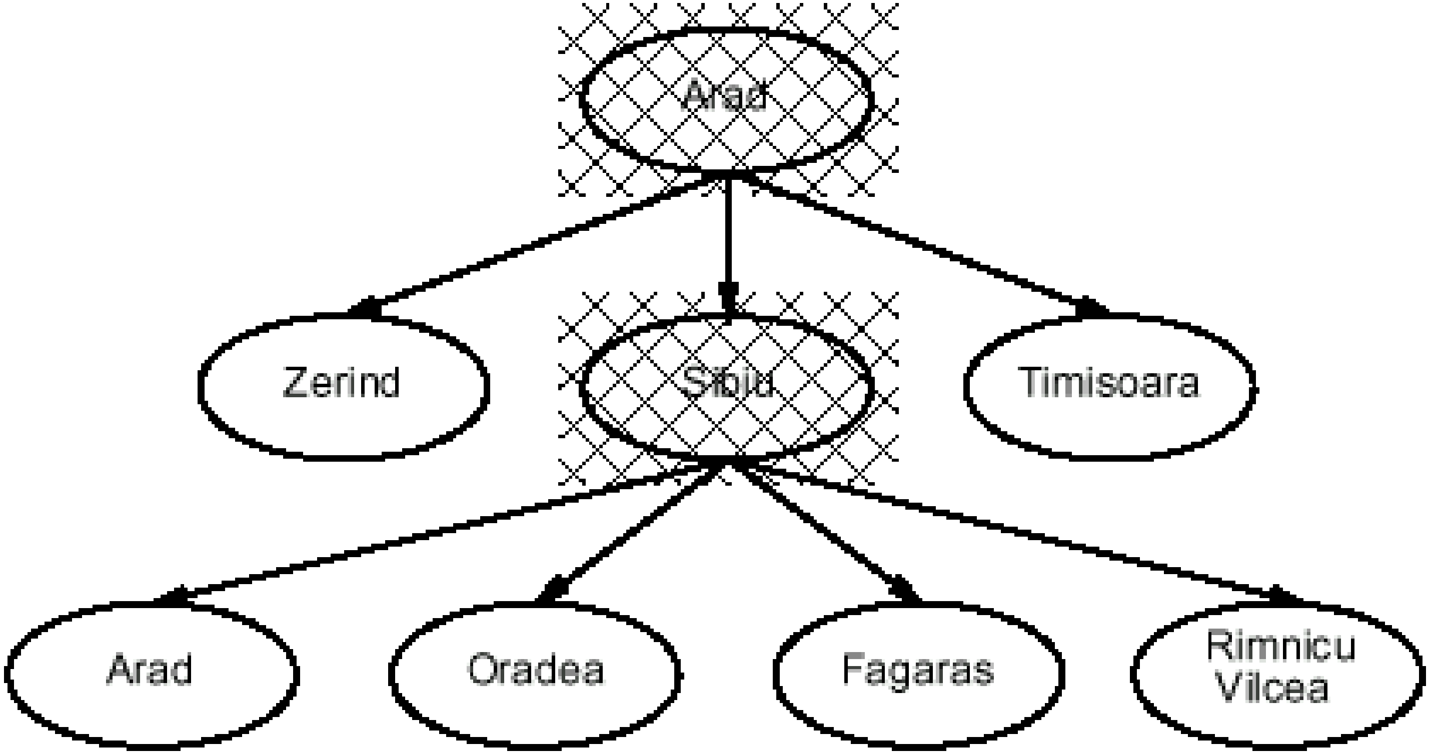


General search example

General search example



General search example



General search example