Problem Solving by Searching

305450 Lecture 2

Problem Solving by Searching

- **Problem Solving:** การแก้ปัญหาประกอบด้วยกระบวนการดังต่อไปนี้
  - **Problem formulation:** การแทนค่าขององค์ประกอบที่สำคัญในโดเมนของปัญหาด้วยสัญลักษณ์ต่าง ๆ
  - **Possible Solutions:** คำตอบที่เป็นไปได้ทั้งหมดของปัญหาที่กำลังพิจารณา
  - **Searching:** การค้นหาคำตอบที่เป็นไปได้ทั้งหมดของปัญหาโดยการประมวลรูปแบบของสัญลักษณ์ต่าง ๆข้างต้น

State Space Search

- เป็นวิธีหนึ่งที่ใช้แก้ปัญหาโดย
  - มีการกำหนดให้ปัญหามีสถานะเริ่มต้น (Initial State) และมีสถานะเป้าหมาย (Goal State)
  - มีข้อกำหนดของการเปลี่ยนสถานะ
  - การแก้ปัญหาคือการหาทาง (path) ที่เปลี่ยนสถานะจาก initial state ไปยัง goal state โดยผ่าน states ต่างๆ
  - วิธีการที่ใช้ในการเปลี่ยนสถานะอาจทำให้ได้คำตอบที่แตกต่างกัน

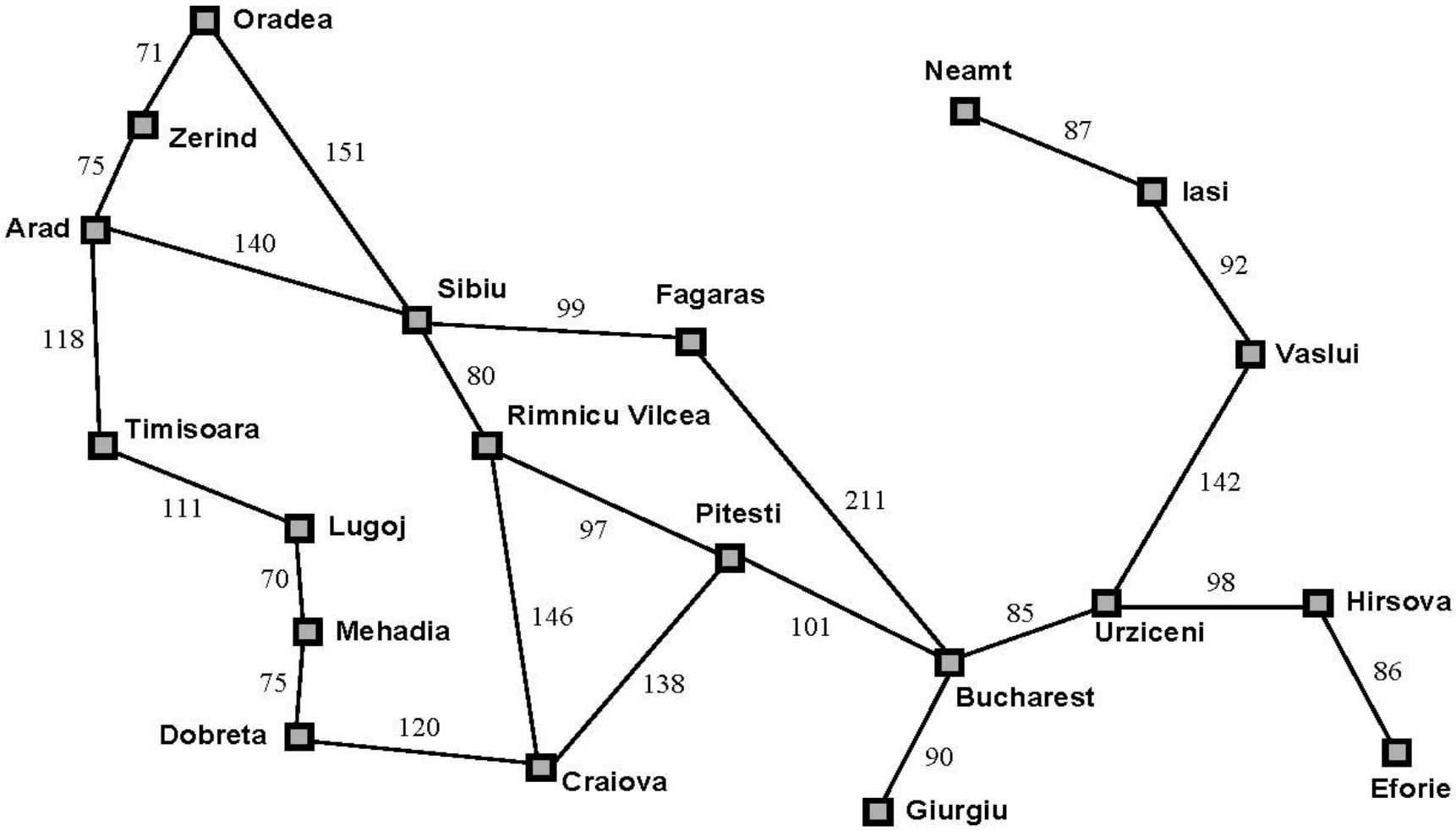
Problem-solving

```
function Simple-Problem-Solving( percept) returns an action
    static: 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)
    action ← Recommendation(seq, state)
    seq ← Remainder(seq, state)
return action
```

Example: Romania

- On holiday in Romania: currently in Arad
- Flight leaves tomorrow from Bucharest
- Formulate goal:
  - Be in Bucharest
- Formulate problem:
  - States: various cities
  - Actions: drive between cities
- Find solutions:
  - Sequence of cities e.g., Arad, Fagaras, Bucharest

Example: Romania



Example: Romania

- On holiday in Romania: currently in Arad
- Flight leaves tomorrow from Bucharest
- Formulate goal:
  - Be in Bucharest
- Formulate problem:
  - States: various cities
  - Actions: drive between cities
- Find solutions:
  - Sequence of cities e.g., Arad, Fagaras, Bucharest

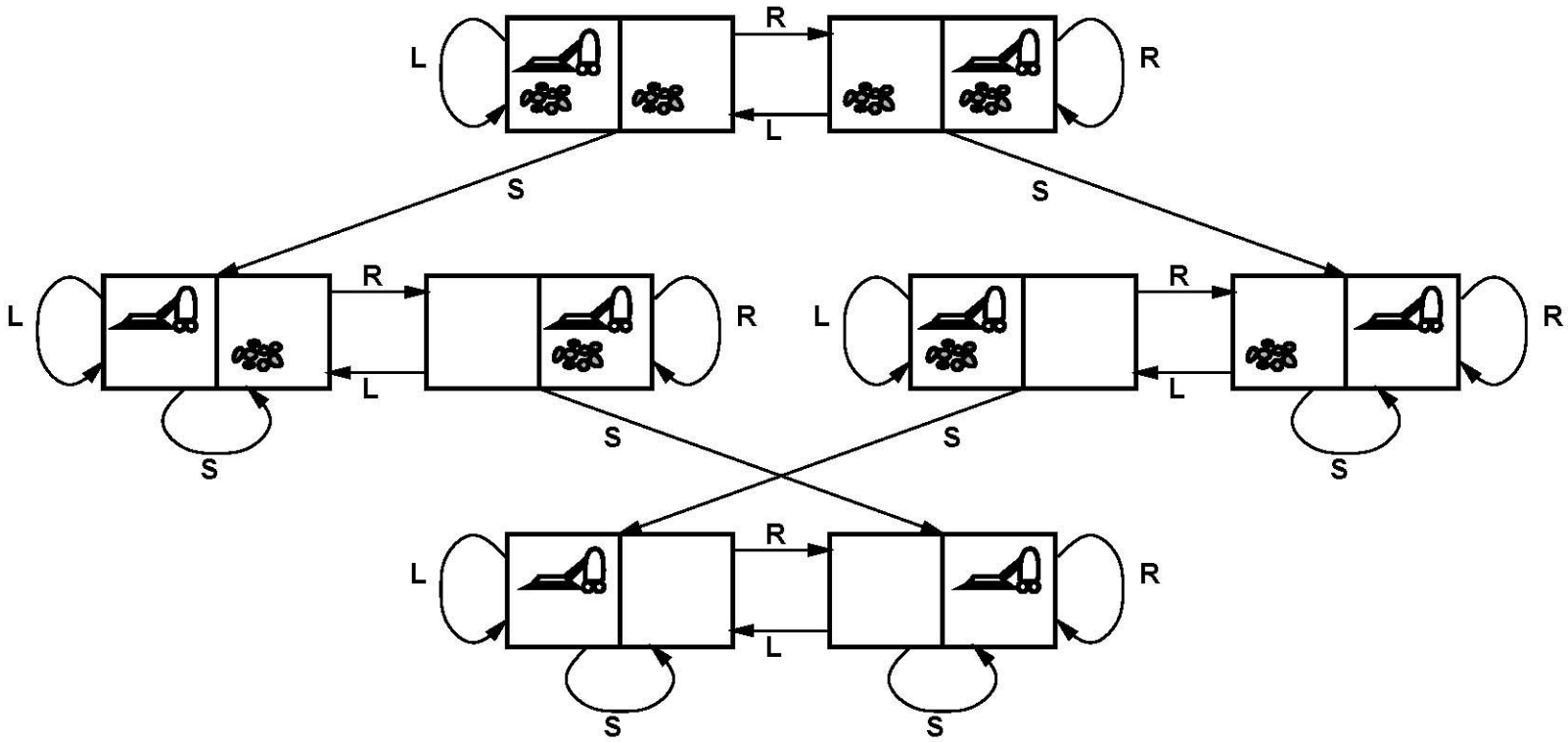
Problem Formulation

- A problem is defined by four items
- Initial state e.g., “at Arad”
- Successor function S(*x*) = set of action-state pairs e.g., S(Arad) = {(Arad → Zerind, Sibiu), …}
- Goal test, can be
  - Explicit, e.g., *x* = “at Bucharest”
  - Implicit, e.g., NoDirt(*x*)
- Path cost, e.g., sum of distances, number of actions executed, etc.
  - *C(x,a,y)* is the step cost, assumed to be ≥ 0
- A solution is a sequence of actions leading from the initial state to a goal state

Selecting a state space

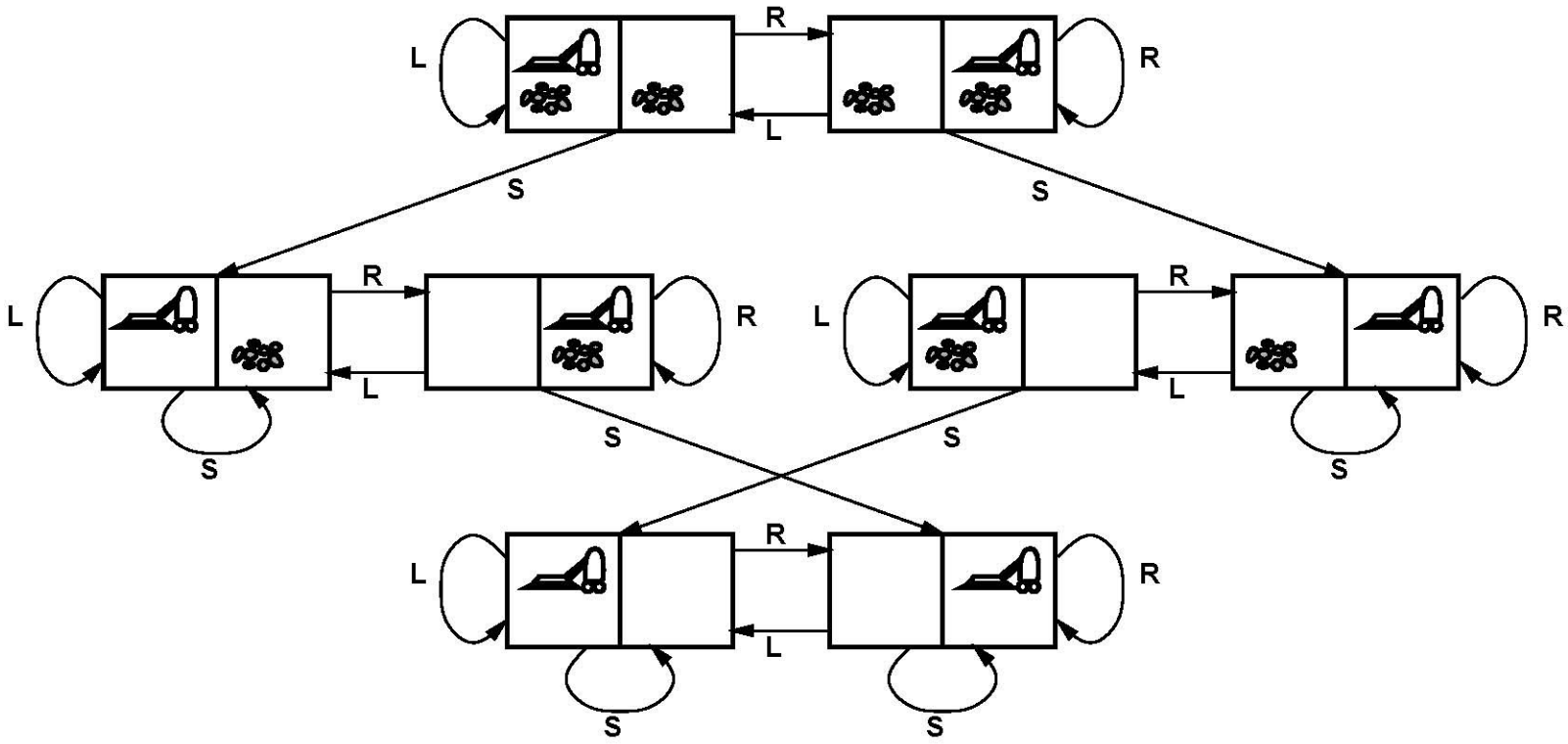
- Real world is complex
  - State space must be abstracted for problem solving
- (Abstract) state = set of real states
  - E.g. “in Arad” represents a complex set of possible rest stops, travel companions, condition of the road, weather, etc.
- (Abstract) action = complex combination of real actions
  - E.g., “Arad → Zerind” ignores details, e.g., turn tseering wheel to the left by three degree, etc.
- For guaranteed realizability, any real state “in Arad” must get to some real state “in Zerind”
- (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



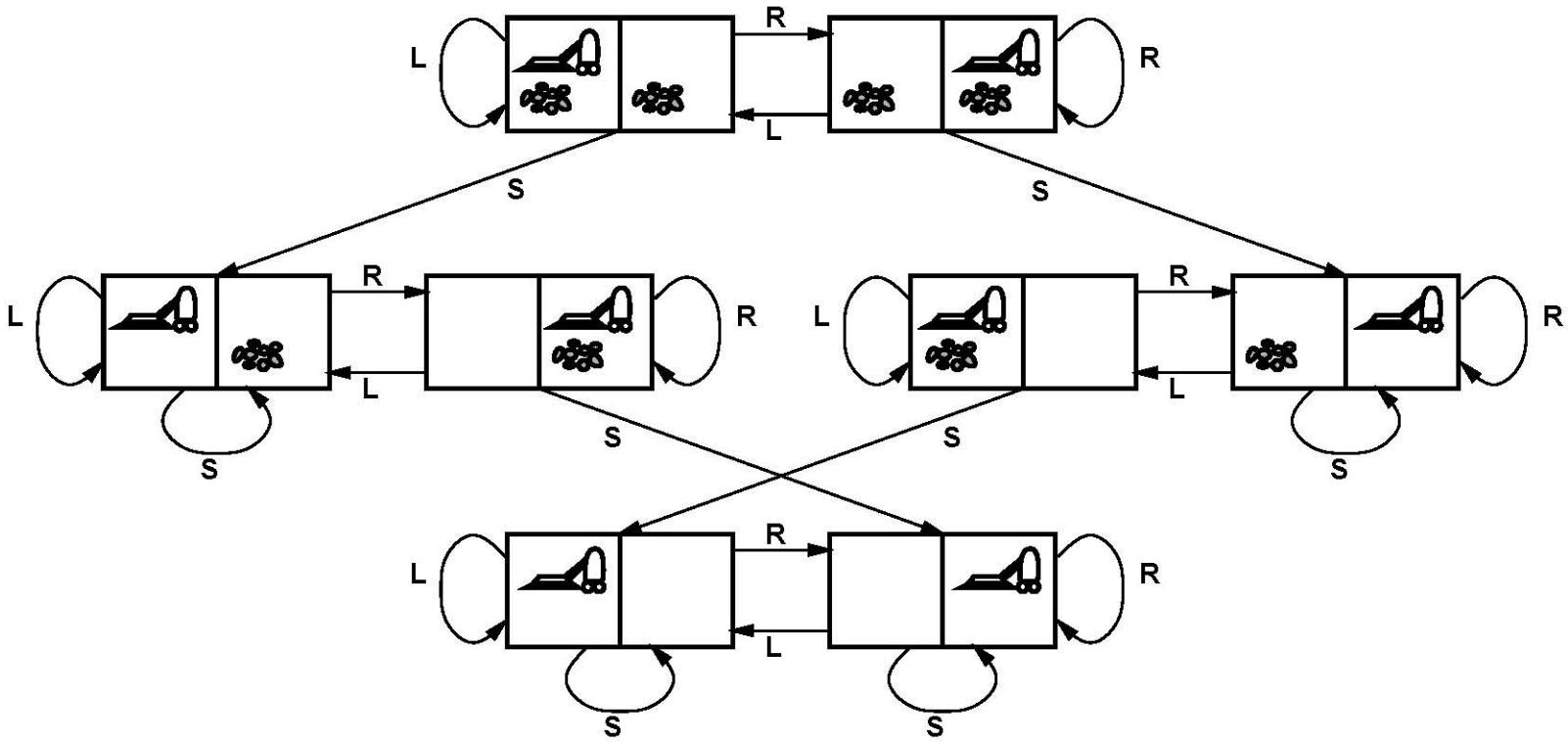
- states??
- actions??
- goal test??
- path cost??

Example: vacuum world state space graph



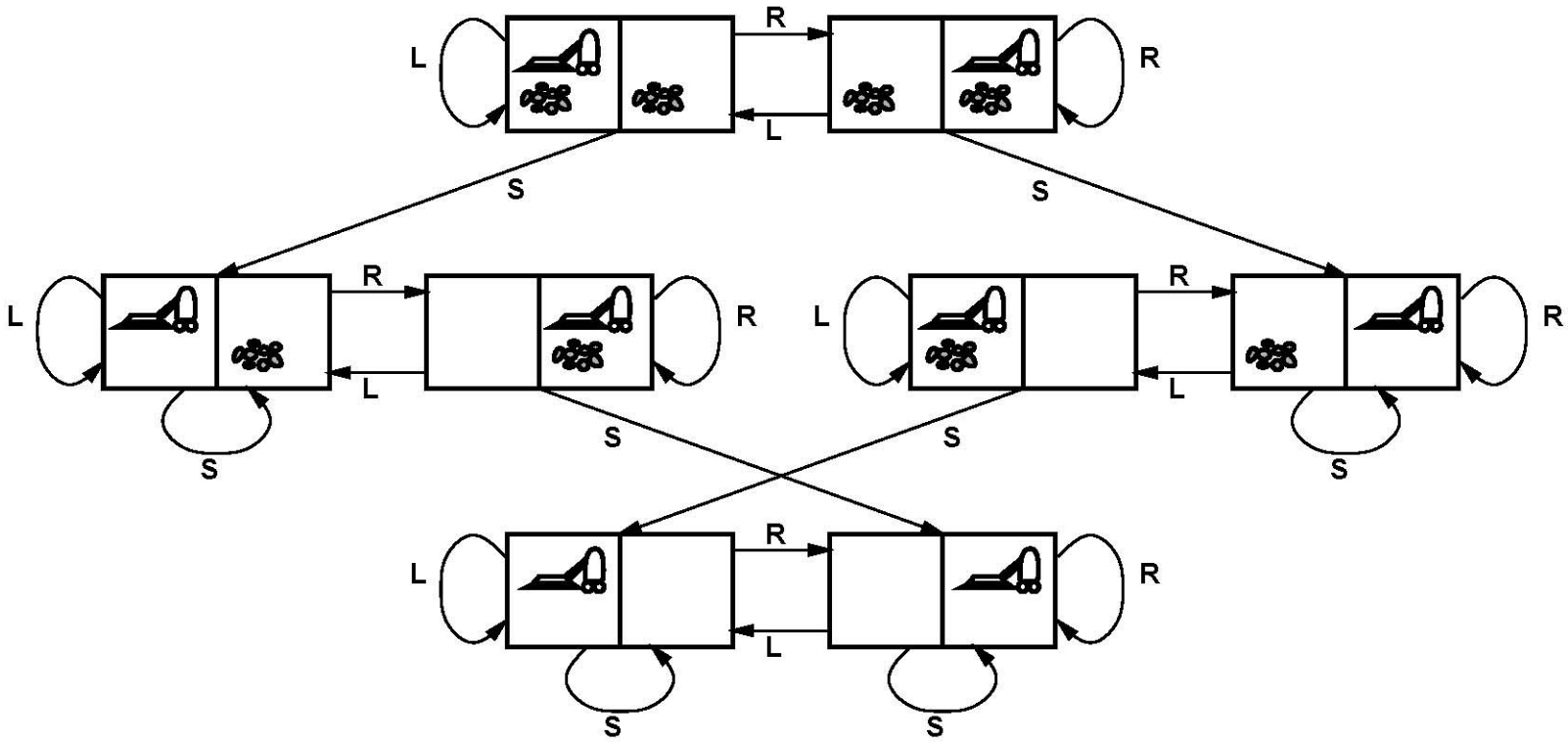
- states: dirt and robot locations (ignore dirt amount, etc)
- actions:
- goal test:
- path cost:

Example: vacuum world state space graph



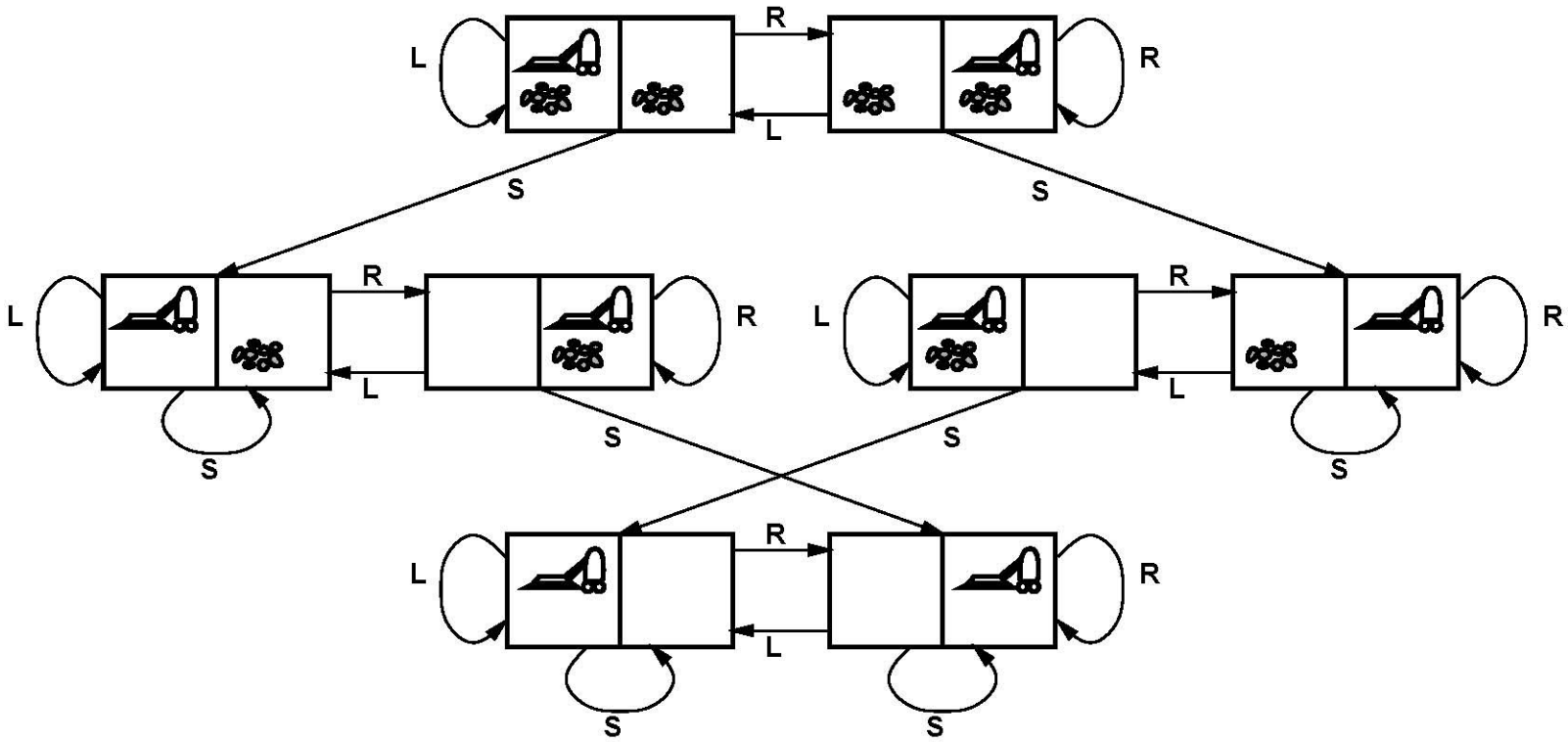
- states: dirt and robot locations (ignore dirt amount, etc)
- actions: *Left*, *Right*, *Suck*, *NoOp*
- goal test:
- path cost:

Example: vacuum world state space graph



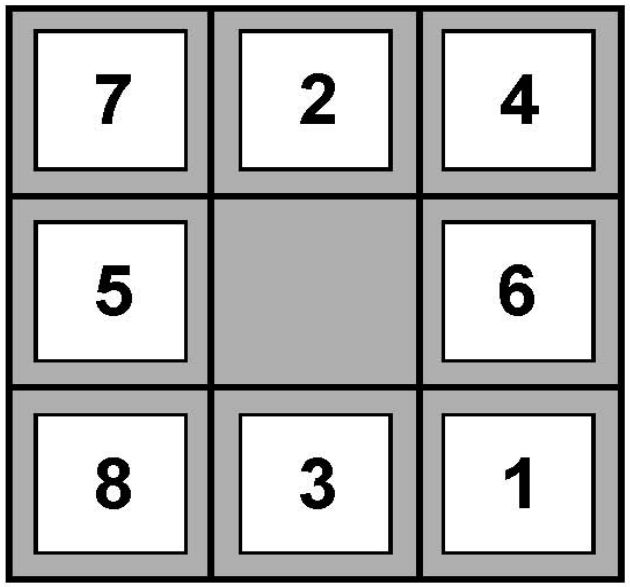
- states: dirt and robot locations (ignore dirt amount, etc)
- actions: *Left*, *Right*, *Suck*, *NoOp*
- goal test: no dirt
- path cost:

Example: vacuum world state space graph

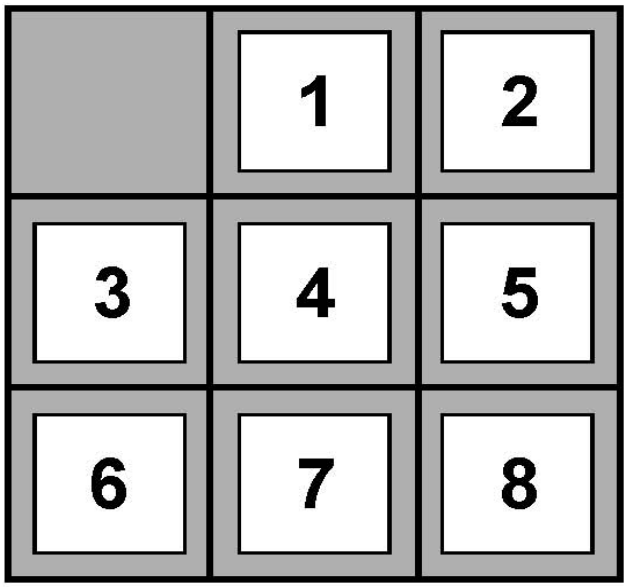


- states: dirt and robot locations (ignore dirt amount, etc)
- actions: *Left*, *Right*, *Suck*, *NoOp*
- goal test: no dirt
- path cost: 1 per action (0 for NoOp)

Example: The 8-puzzle



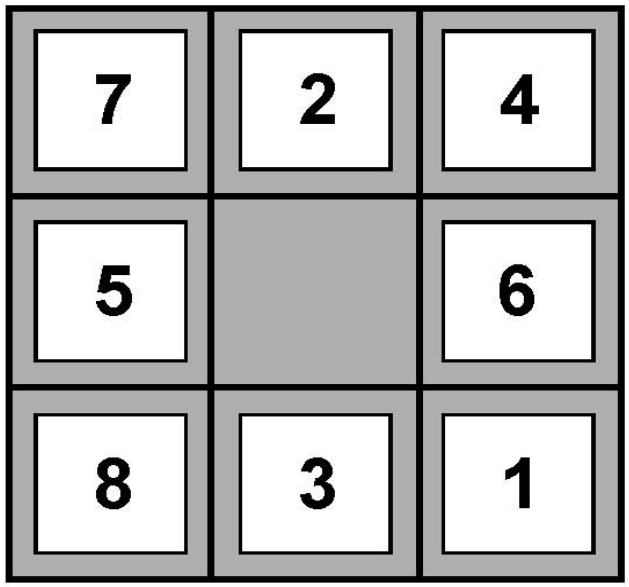
Start State



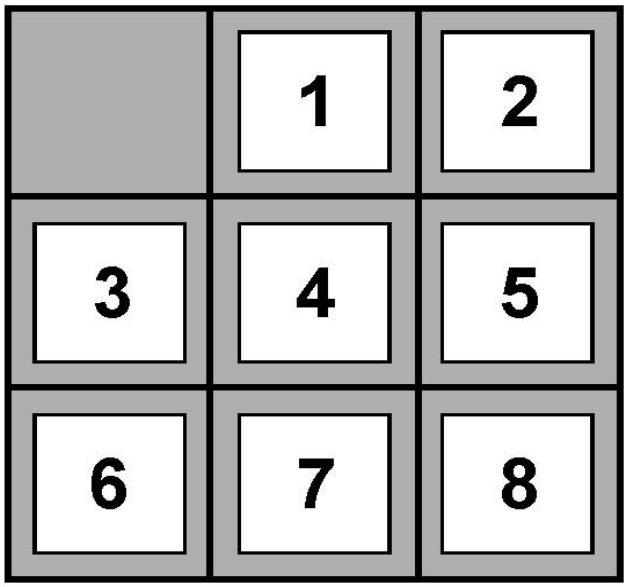
Goal State

- states??
- actions??
- goal test??
- path cost??

Example: The 8-puzzle



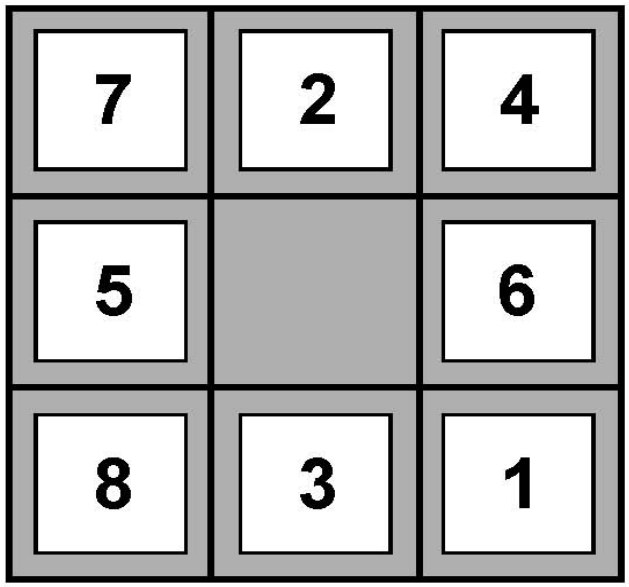
Start State



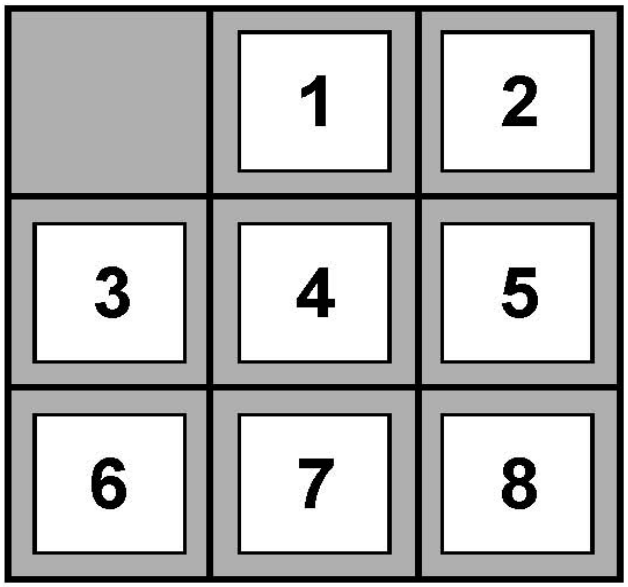
Goal State

- states: locations of tiles
- actions:
- goal test:
- path cost:

Example: The 8-puzzle



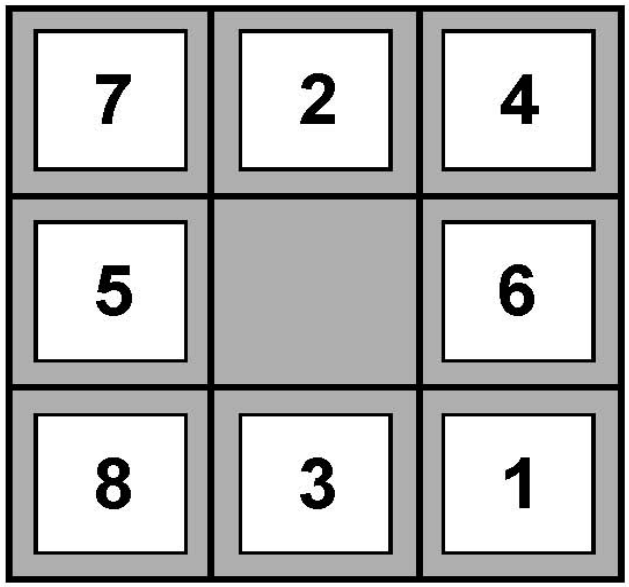
Start State



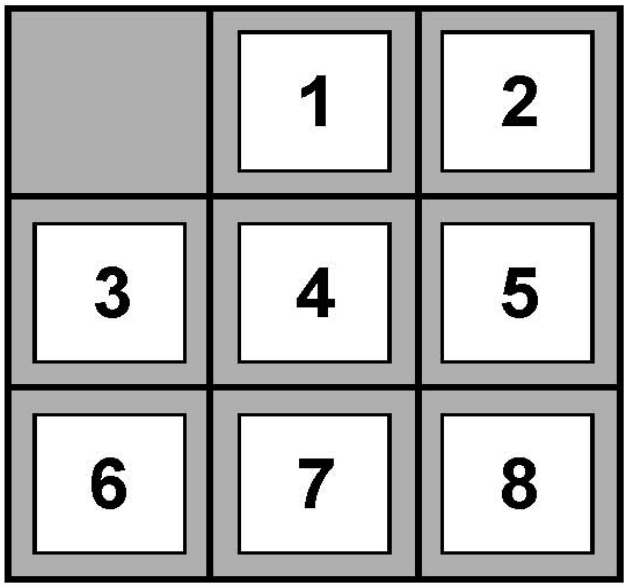
Goal State

- states: locations of tiles
- actions: move blank left, right, up, down
- goal test:
- path cost:

Example: The 8-puzzle



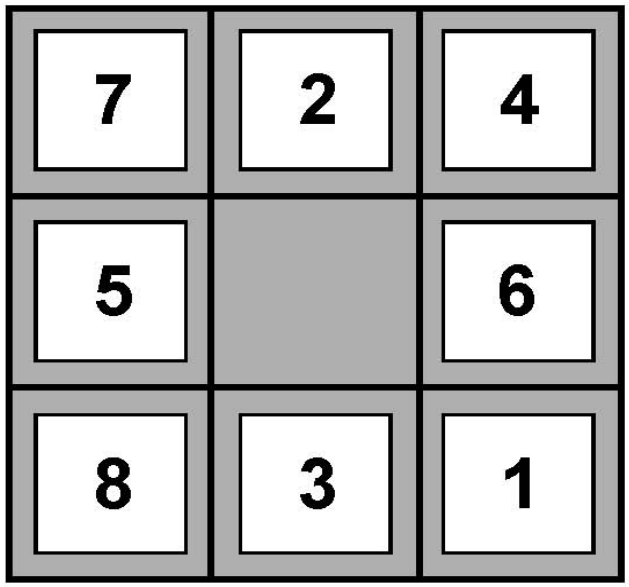
Start State



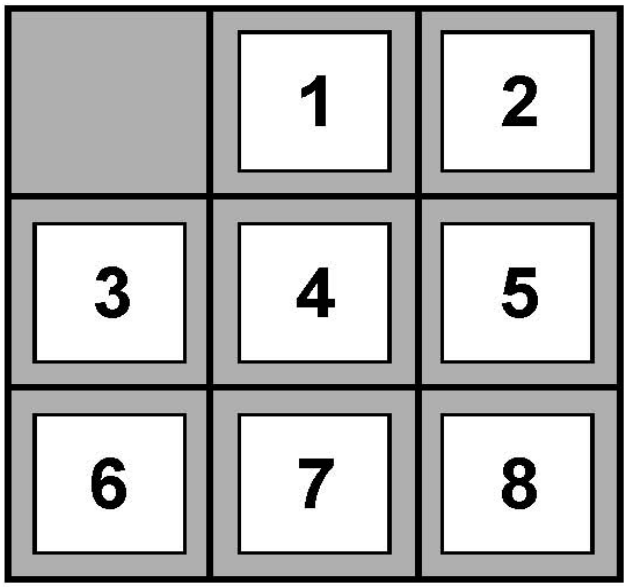
Goal State

- states: locations of tiles
- actions: move blank left, right, up, down
- goal test: = goal state (given)
- path cost:

Example: The 8-puzzle



Start State



Goal State

- states: locations of tiles
- actions: move blank left, right, up, down
- goal test: = goal state (given)
- path cost: 1 per move