Unit 2: Problem solving agents

Objectives

- To formulate problem from goal formulation
- To measure the problem solving performance

Outcomes

- Formulate problems from goal formulation
- Evaluate the algorithm’s performance

Outline

- Problem-solving agents
  - **A kind of goal-based agent**
- Problem types
  - **Single state (fully observable)**
  - **Search with partial information**
- Problem formulation
  - **Example problems**
- Basic search algorithms
  - **Uninformed**

Problem-solving agent

- Four general steps in problem solving:
  - **Goal formulation**
    - What are the successful world states
  - **Problem formulation**
    - What actions and states to consider give the goal
  - **Search**
    - Determine the possible sequence of actions that lead to the states of known values and then choosing the best sequence.
  - **Execute**
    - Give the solution perform the actions.

Problem Solving - Definition

- Problem solving is a process of generating solutions from observed data.
- Problem is characterized by a set of goals, set of objects, and set of operations.
- The method of solving problem through AI involves the process of
  - defining the search space,
  - deciding start and goal states
  - finding the path from start state to goal state through search space.

Problem Solving – Definition…

Agent world
-> environment

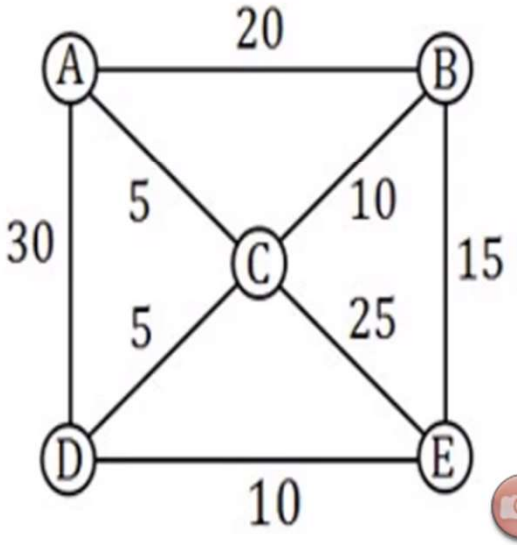
- Here search space or problem space is abstract space which has all valid states that can be generated by the application of any combination of operators or objects.
- A problem space can have one or more solutions.
- Solution is a combination of objects and operations that achieve the goals.
- Search refers a search of solution in problem space.

Goal Based Agent or Problem Solving Agent

- Goal-based agents are usually called planning agents.
- Problem solving begins with the definitions of problems and their solutions.
- We then describe several general-purpose search algorithms that can be used to solve these problems.
- We will see several uninformed search algorithms— algorithms that are given no information about the problem other than its definition.
- Informed search algorithms, on the other hand, can do quite well given some guidance on where to look for solutions

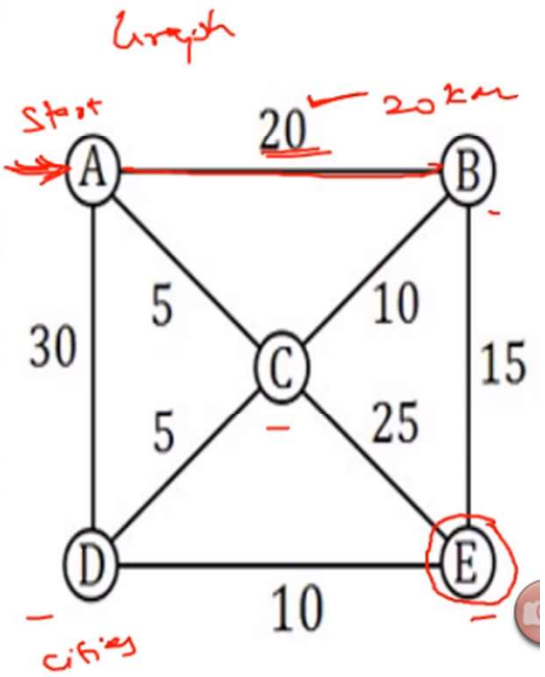
Problem Solving Agent

- Problem-solving agents: used to find sequence of actions that achieve goals.
- An example,
- Each node represents a city, and the cost to travel from a city to another is denoted by the number over the edge connecting the nodes of those 2 cities.
- In order for an agent to solve a problem it should pass by 2 phases of formulation:



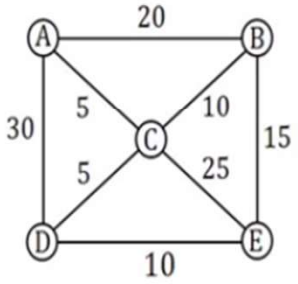
Problem Solving Agent

- Problem-solving agents: used to find sequence of actions that achieve goals.
- An example,
- Each node represents a city, and the cost to travel from a city to another is denoted by the number over the edge connecting the nodes of those 2 cities.
- In order for an agent to solve a problem it should pass by 2 phases of formulation:



Goal Formulation and Problem Formulation

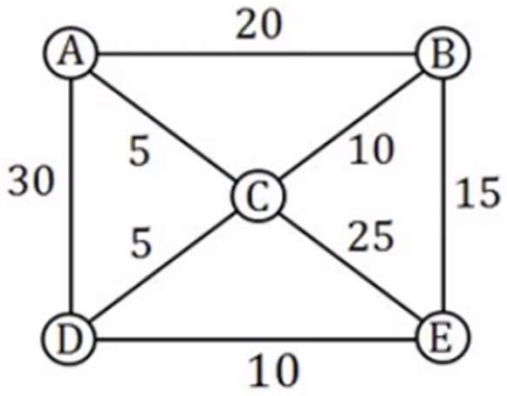
- Goal Formulation:
  - Problem solving is about having a goal we want to reach, (i.e. I want to travel from 'A' to 'E').
  - Goal is a set of environment states in which the goal is satisfied.



- Problem Formulation:
  - A problem formulation is about deciding what actions and states to consider,
  - describe states as "in (CITYNAME)" where CITYNAME is the name of the city in which we are currently in.

- The process of finding such sequence is called search, a search algorithm is like a black box which takes problem as input returns a solution, and once the solution is found the sequence of actions carried out, and this is called the execution phase.

Formulating Problems



- A problem can be defined formally by 4 components:
- **Initial state, successor function, goal test, and path cast.**

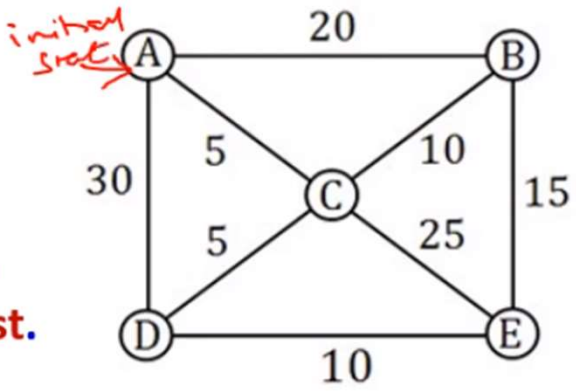
1. Initial State:

- Specify one or more states that describes the possible situations from which the problem solving process starts.

2. Successor Function :

- a description of the possible actions available to the agent, it is common to describe it by means of a successor function, given state x then SUCCESSOR-FN(x) returns a set of ordered pairs
  - **<action, successor>** where action is a legal action from state x and successor is the state in which we can be by applying action.
- The initial state and the successor function together defined what is called state space which is the set of all possible states reachable from the initial state {e.g: in(A), in(B), in(C), in(D), in(E)}.

Formulating Problems



- A problem can be defined formally by 4 components:
- **Initial state, successor function, goal test, and path cast.**

1. Initial State:

- Specify one or more states that describes the possible situations from which the problem solving process starts.

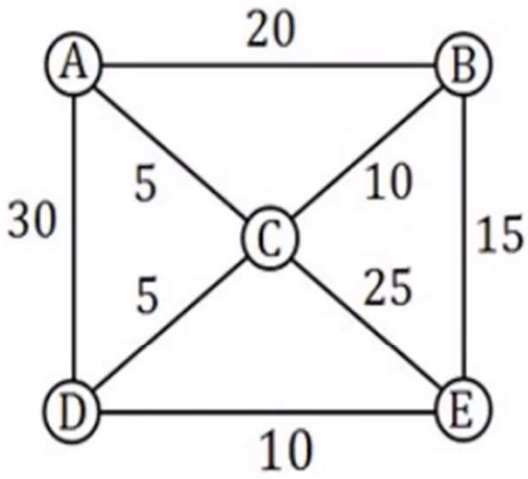
2. Successor Function :

– a description of the possible actions available to the agent, it is common to describe it by means of a successor function, given state x then SUCCESSOR-FN(x) returns a set of ordered pairs

- **<action, successor>** where action is a legal action from state x and successor is the state in which we can be by applying action.

– The initial state and the successor function together defined what is called state space which is the set of all possible states reachable from the initial state {e.g: in(A), in(B), in(C), in(D), in(E)}.

Formulating Problems...



3. Goal Test:

Knowledge about the goal

- Decide whether the current state is a goal state
- {i.e.: is the current state is in(E)?}.

4. Path cost:

- a function that assigns a numeric value to each path, C(A, cost, B)
- each step we take in solving the problem should be somehow weighted,
- If I travel from A to E our agent will pass by many cities, the cost to travel between two consecutive cities should have some cost measure, {e.g: Traveling from 'A' to 'B' costs 20 km or it can be typed as c(A, 20, B)}.

- A solution to a problem is path from the initial state to a goal state,
- and solution quality is measured by the path cost, and the optimal solution has the lowest path cost among all possible solutions.

minimum path cost

Simple Problem Solving Agent

function SIMPLE-PROBLEM-SOLVING-AGENT(p) returns an action

inputs: *p* - *a* percept

static: *s* - an action sequence, initially empty *state,* some description of the current world state

g - a goal, initially null

problem - a problem formulation

state ← UPDATE-STATE(state, *p)*

if *s* is empty then

g ← FORMULATE-GOAL(state)

Problem ← FORMULATE-PROBLEM(state, *g)*

s ← *SEARCH(problem)*

action ← RECOMMENDATION(s, *state)*

s ← REMAINDER(s, *state)*

return *action*

Example: Romania

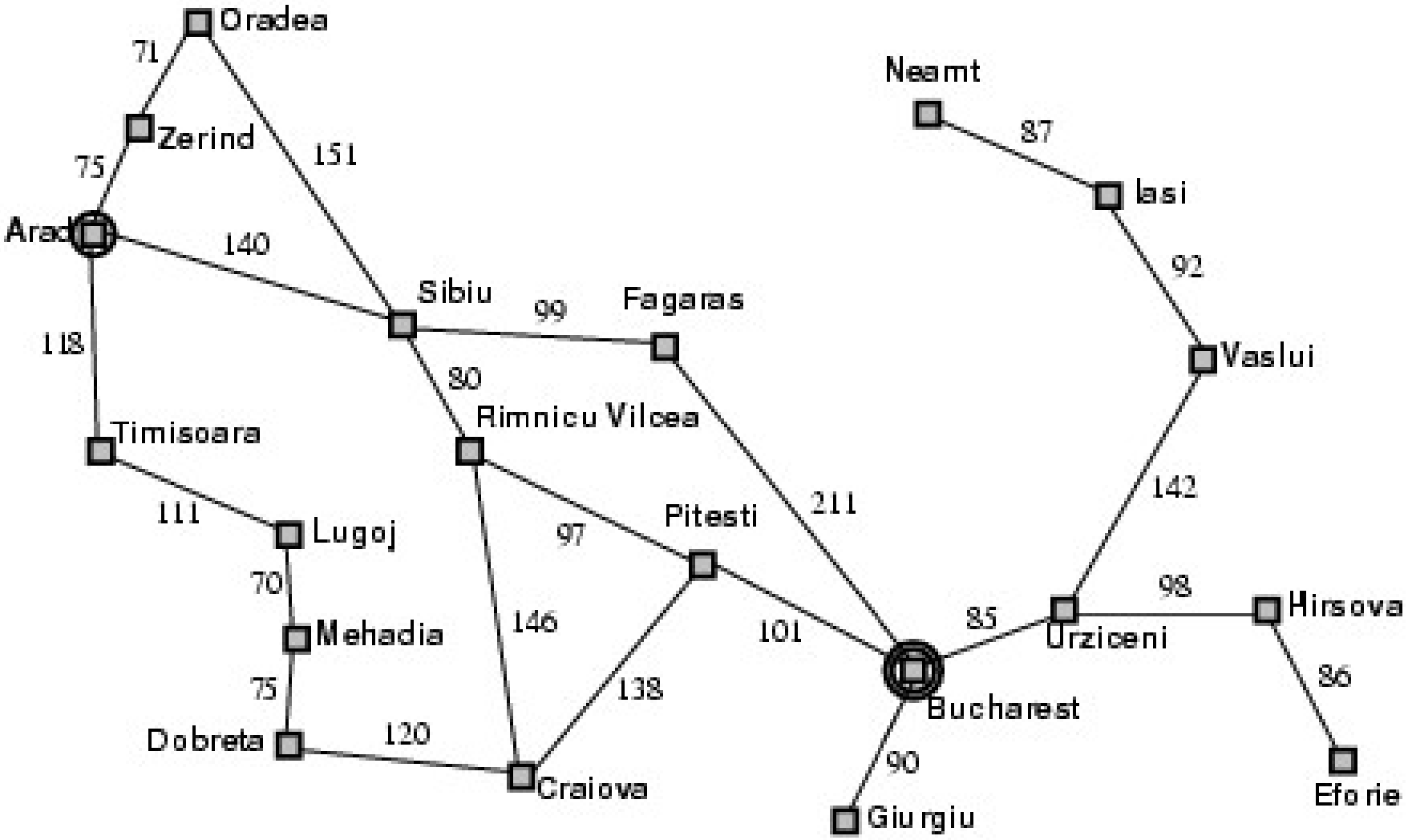
- On holiday in Romania; currently in Arad.
- Flight leaves tomorrow from Bucharest

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

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

- **Find solution:**
  - sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

Example: Romania



Single-state problem formulation

A problem is defined by four items:

1. initial state e.g., "at Arad“

2. actions or successor function *S(x)* = set of action–state pairs
   - e.g., *S(Arad) = {<Arad → Zerind, Zerind>, … }*
   -
3. goal test, can be
   - explicit, e.g., *x* = "at Bucharest"
   - implicit, e.g., *Checkmate(x)*

4. path cost (additive)
   - 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

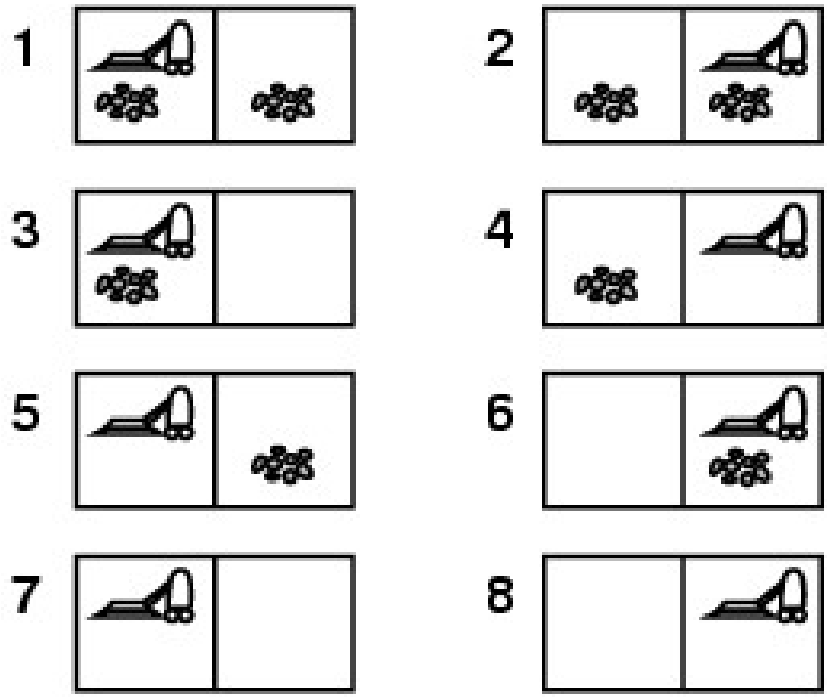
- Real world is absurdly complex
  → state space must be abstracted for problem solving
- (Abstract) state = set of real states
- (Abstract) action = complex combination of real actions
  - e.g., "Arad → Zerind" represents a complex set of possible routes, detours, rest stops, etc.
- For guaranteed reliability, 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

Problem types

- **Deterministic, fully observable → single-state problem**
  - Agent knows exactly which state it will be in; solution is a sequence
- **Non-observable → sensor less problem (conformant problem)**
  - Agent may have no idea where it is; solution is a sequence
- **Nondeterministic and/or partially observable → contingency problem**
  - percepts provide **new** information about current state
  - often **interleave** search, execution
- **Unknown state space → exploration problem**

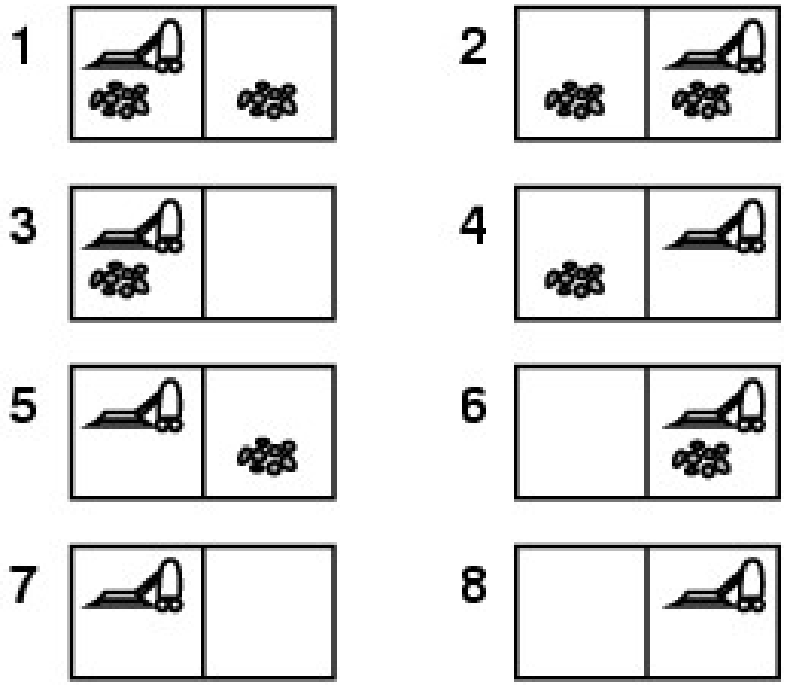
Example: vacuum world

- Single-state, start in #5.
  Solution?



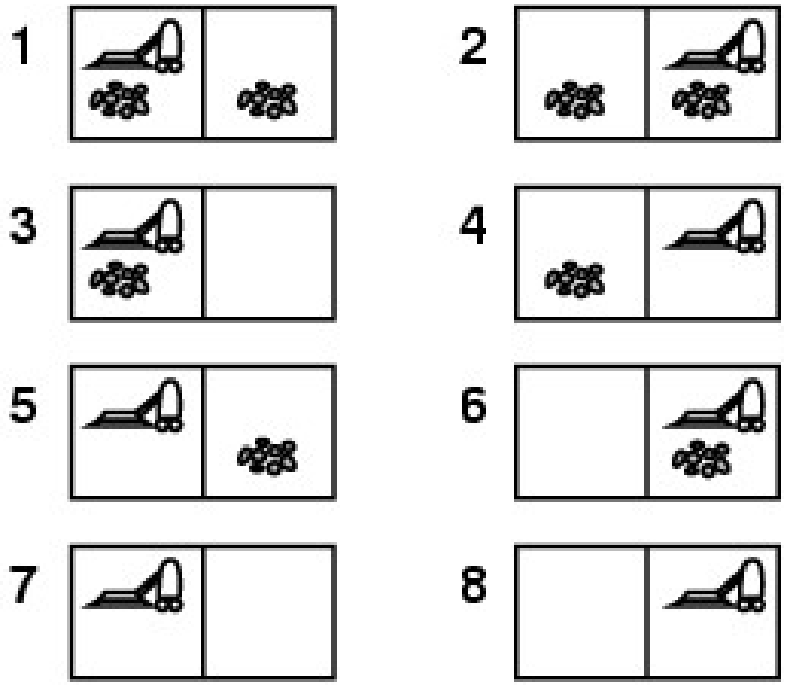
Example: vacuum world

- Single-state, start in #5. Solution? *[Right, Suck]*
- Sensorless, start in {*1,2,3,4,5,6,7,8*} e.g., *Right* goes to {*2,4,6,8*} Solution?



Example: vacuum world

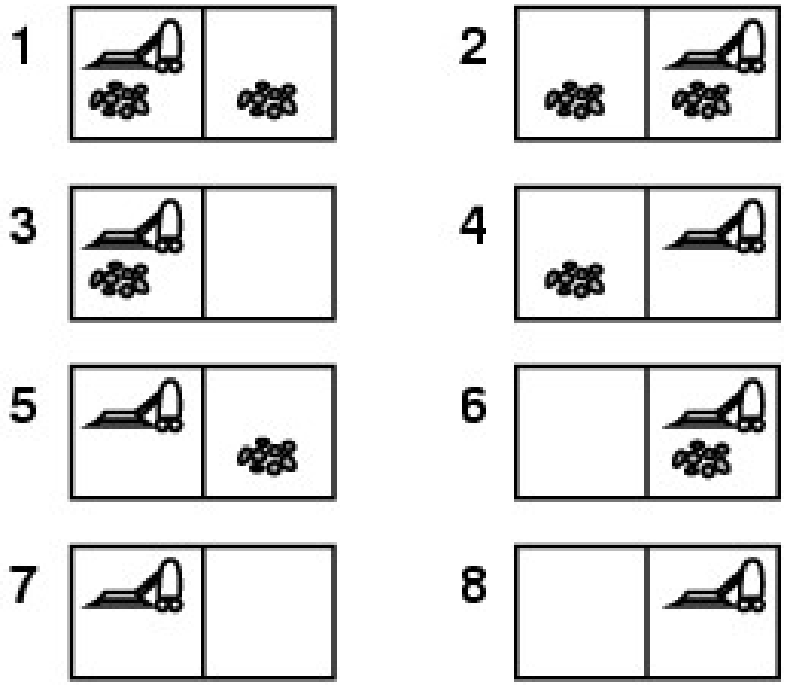
- Sensorless, start in {*1,2,3,4,5,6,7,8*} e.g., *Right* goes to {*2,4,6,8*} Solution? *[Right,Suck,Left,Suck]*



- Contingency
  - Nondeterministic: *Suck* may dirty a clean carpet
  - Partially observable: location, dirt
  - Percept: *[L, Clean],* i.e., start in #5 or #7 Solution?

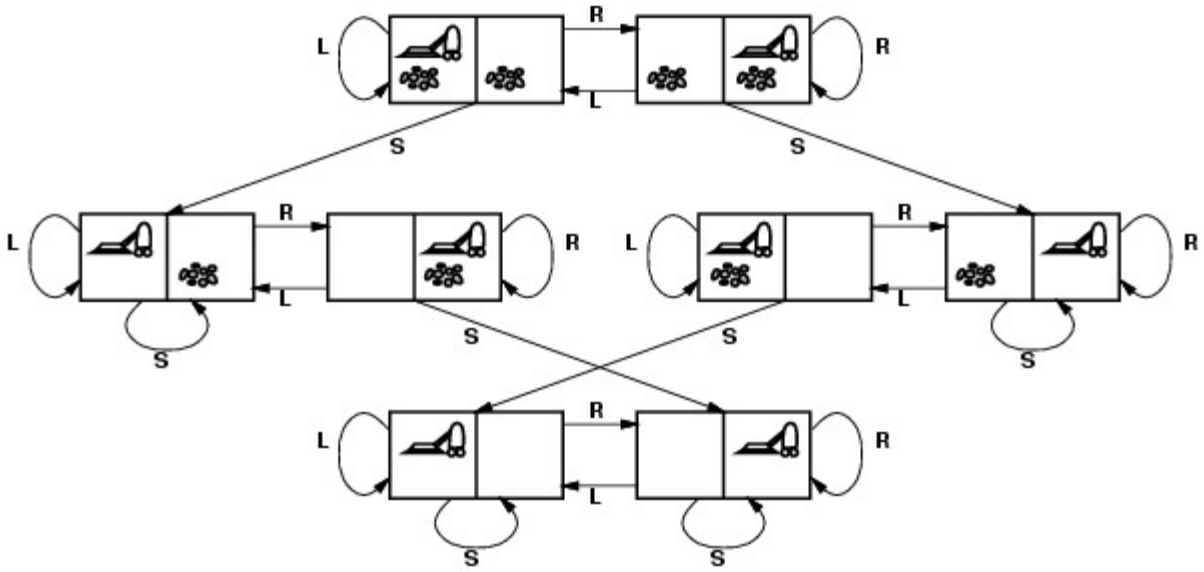
Example: vacuum world

- **Sensorless,** start in {*1,2,3,4,5,6,7,8*} e.g., *Right* goes to {*2,4,6,8*}
  Solution?
  [Right,Suck,Left,Suck]



- **Contingency**
  - Nondeterministic: *Suck* may dirty a clean carpet
  - Partially observable: location, dirt at current location.
  - Percept: *[L, Clean],* i.e., start in #5 or #7
    Solution? *[Right, **if** dirt **then** Suck]*

Vacuum world state space graph

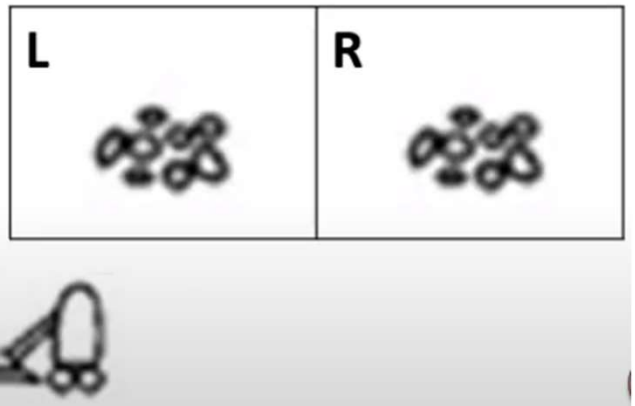


- states?
- actions?
- goal test?
- path cost?

Vacuum World Problem

- There are two dirty square blocks and we need to clean these dirty square blocks by using vacuum cleaner.
- Percepts : Location and content e.g. (L,Dirty)
- Possible actions for Vacuum cleaner

1. Move Left (L)
2. Move Right (R)
3. Suck (S)
4. No Operation (NoOp)



Vacuum World Problem…

- Initial State
- Vacuum Agent can be in any states (Left or Right) shown in the picture
- Successor function
- Successor function generates legal states resulting from applying the three actions {Left, Right, and Suck}
- The states space is shown in the picture, there are 8 world states
- Goal test
- Checks whether all squares are clean
- Path cost
- Each step costs 1, so the path cost is the sum of steps in the path from Initial state to goal state.

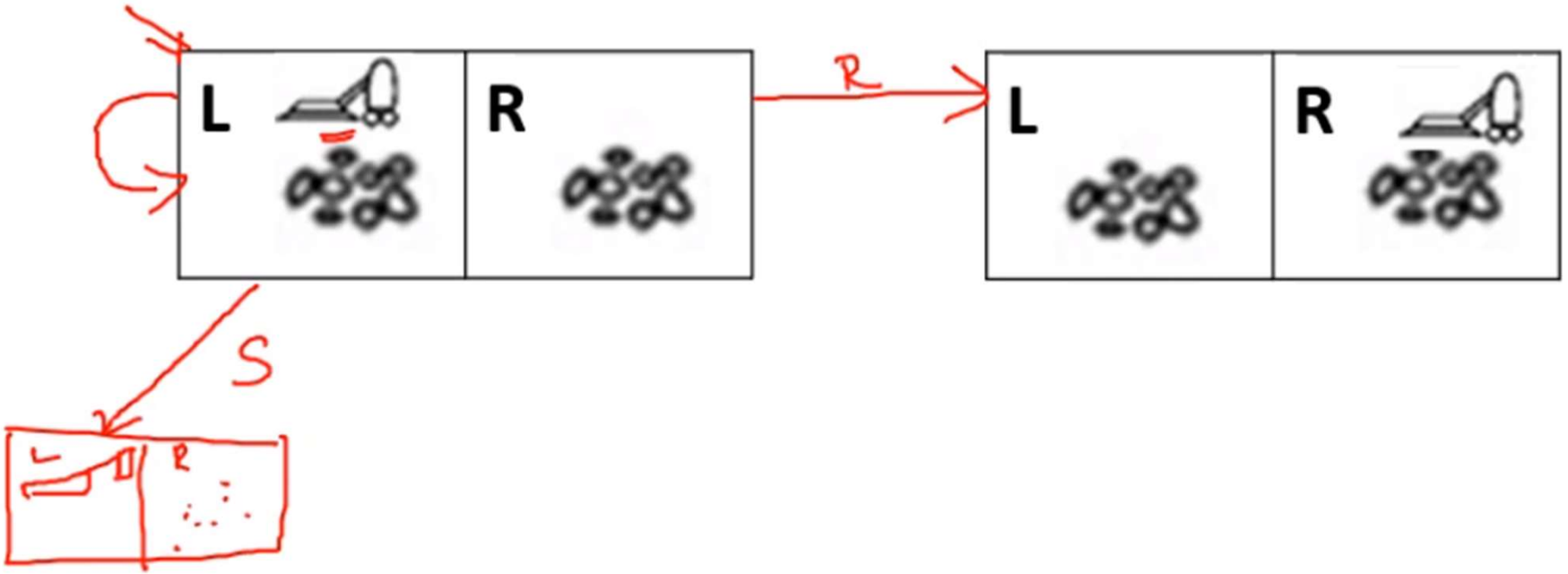
Vacuum World – Transition diagram

2 ← block

3 ← action

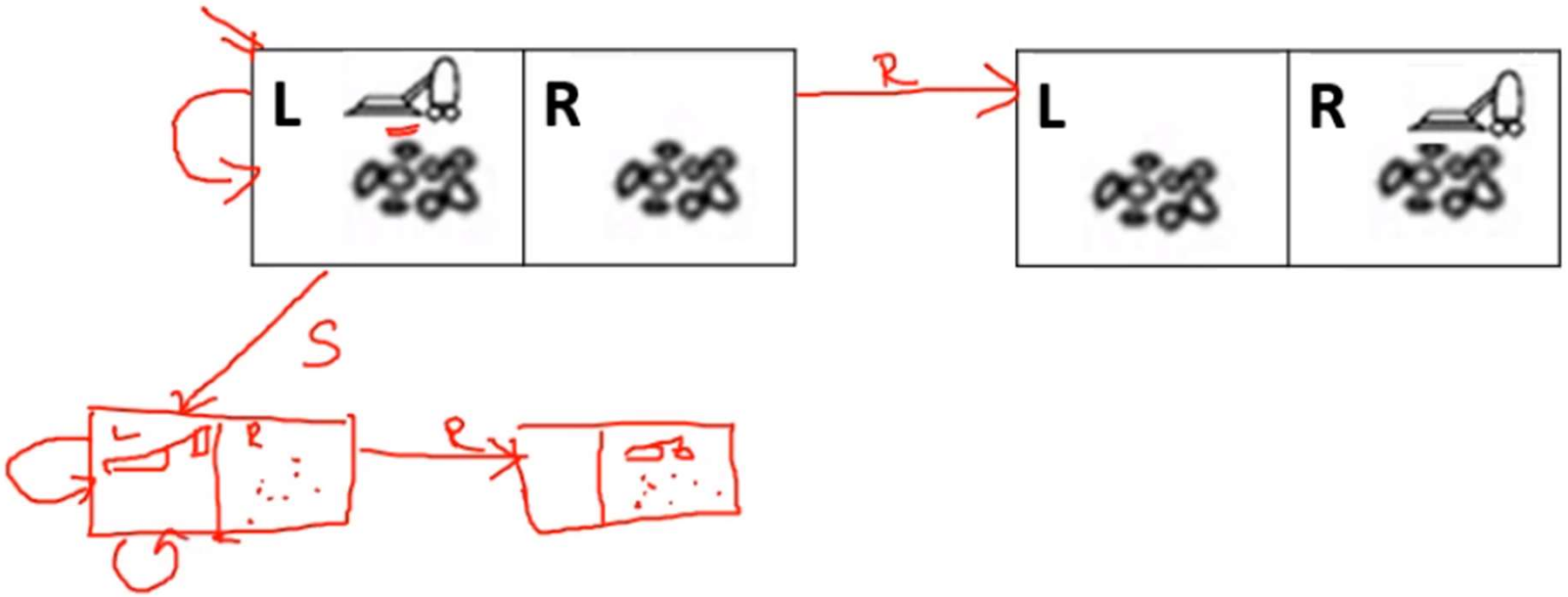
2^3 ← 8 states

Vacuum World – Transition diagram



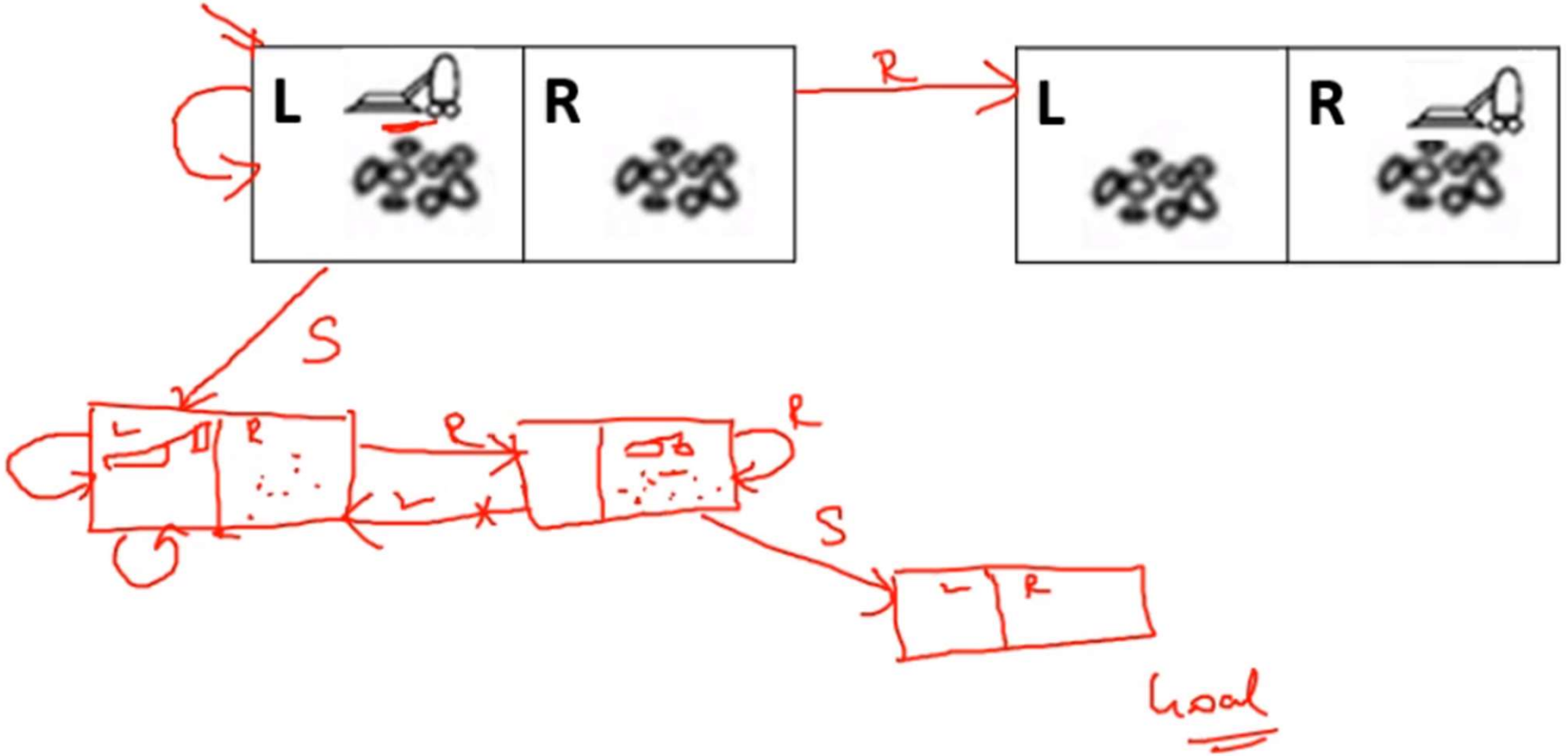
Possible Actions: 1. May move Right -> New state
2. May move Left -> No Operation (Already cleaner in Left side only)
3. S -> Suck Operation. So Right side is dirty and left side is Cleaned.

Vacuum World – Transition diagram



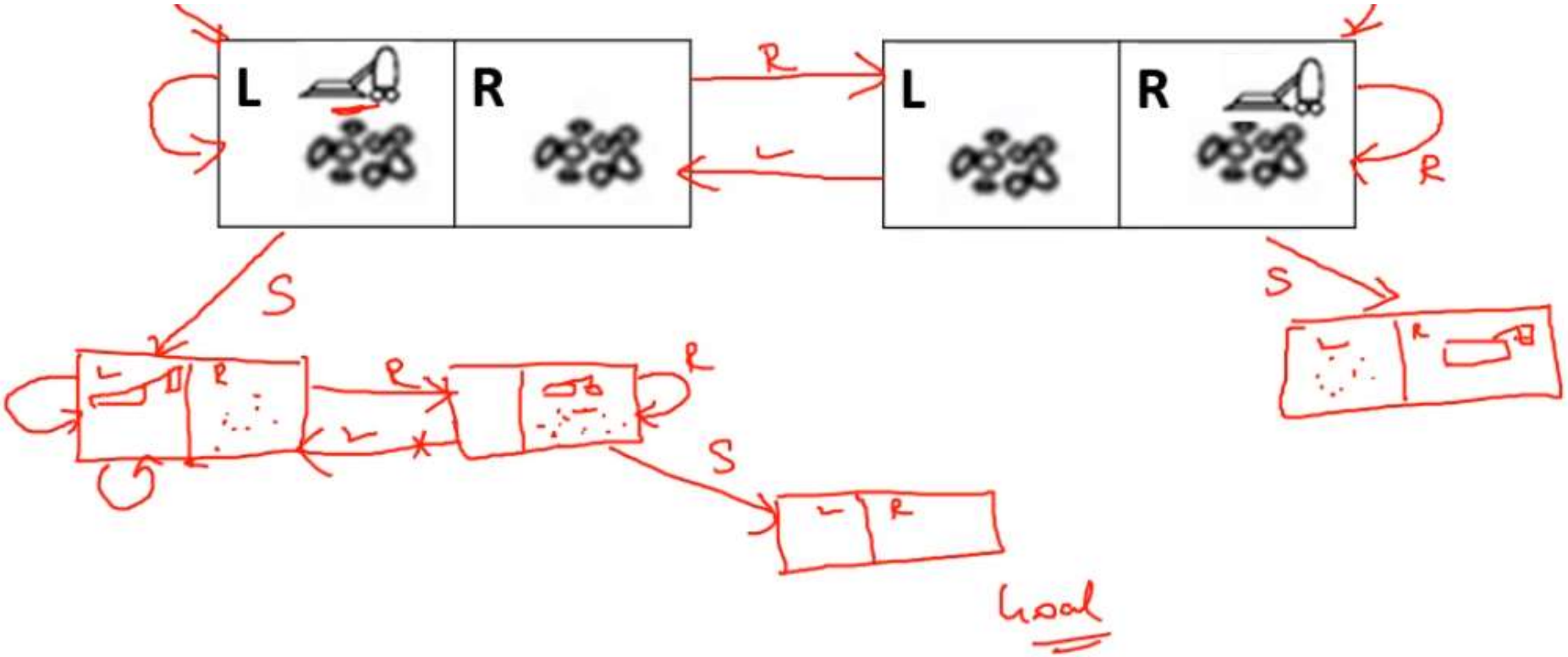
Possible Actions: 1. May move Right (Only Possible move)
2. May move Left -> No Operation (Already cleaner in Left side only)
3. S -> No Operation. Already cleaned.

Vacuum World – Transition diagram



Possible Actions: 1. May move Right -> No Operation
2. May move Left -> Not required here(Already cleaned only)
3. S -> Suck -> Only Possible move => Reached **GOAL**

Vacuum World – Transition diagram

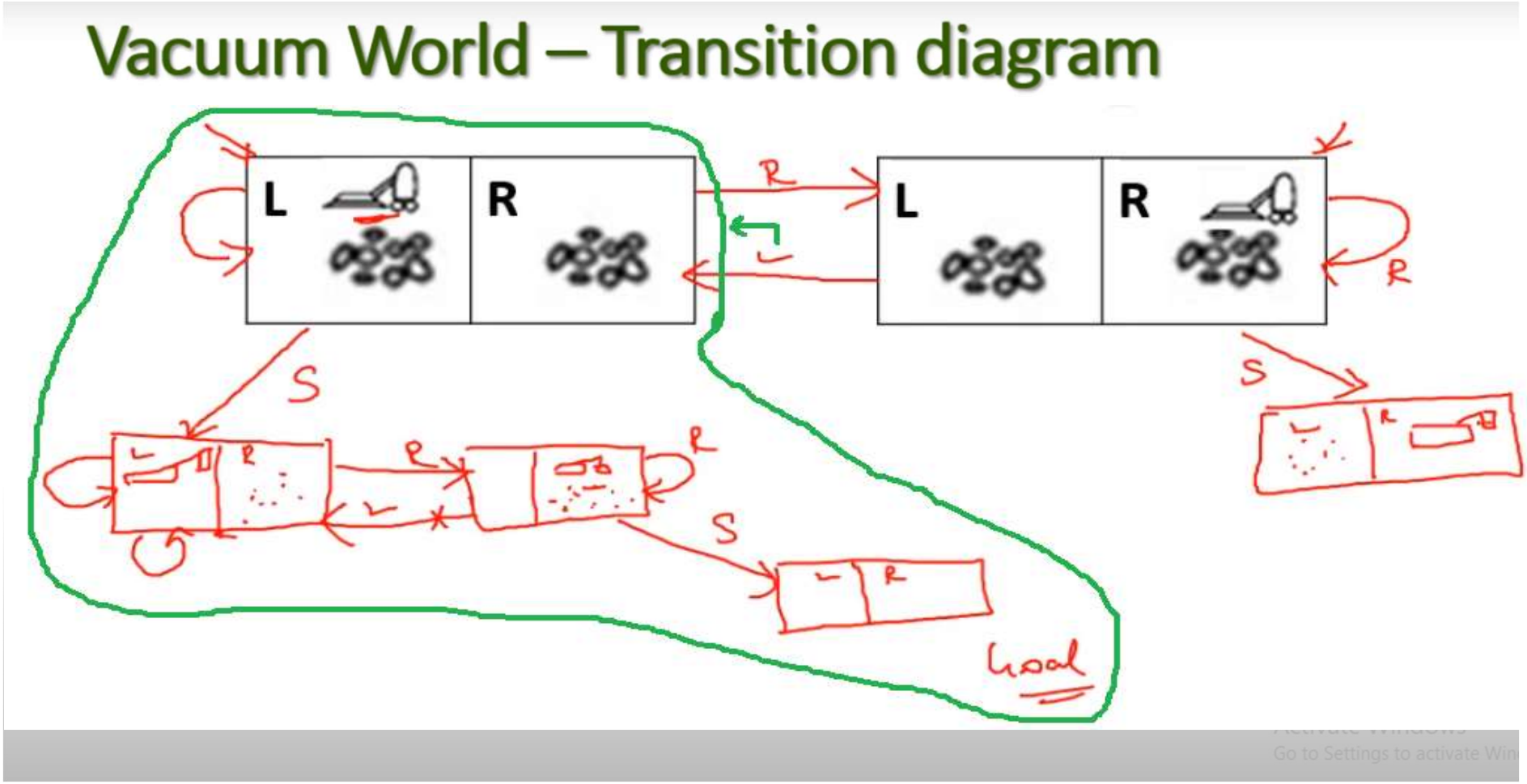


Possible Actions: 1. May move Right -> No Operation

2. **May move Left** -> Not required here(Already cleaned only)

3. S -> Suck -> Only Possible move

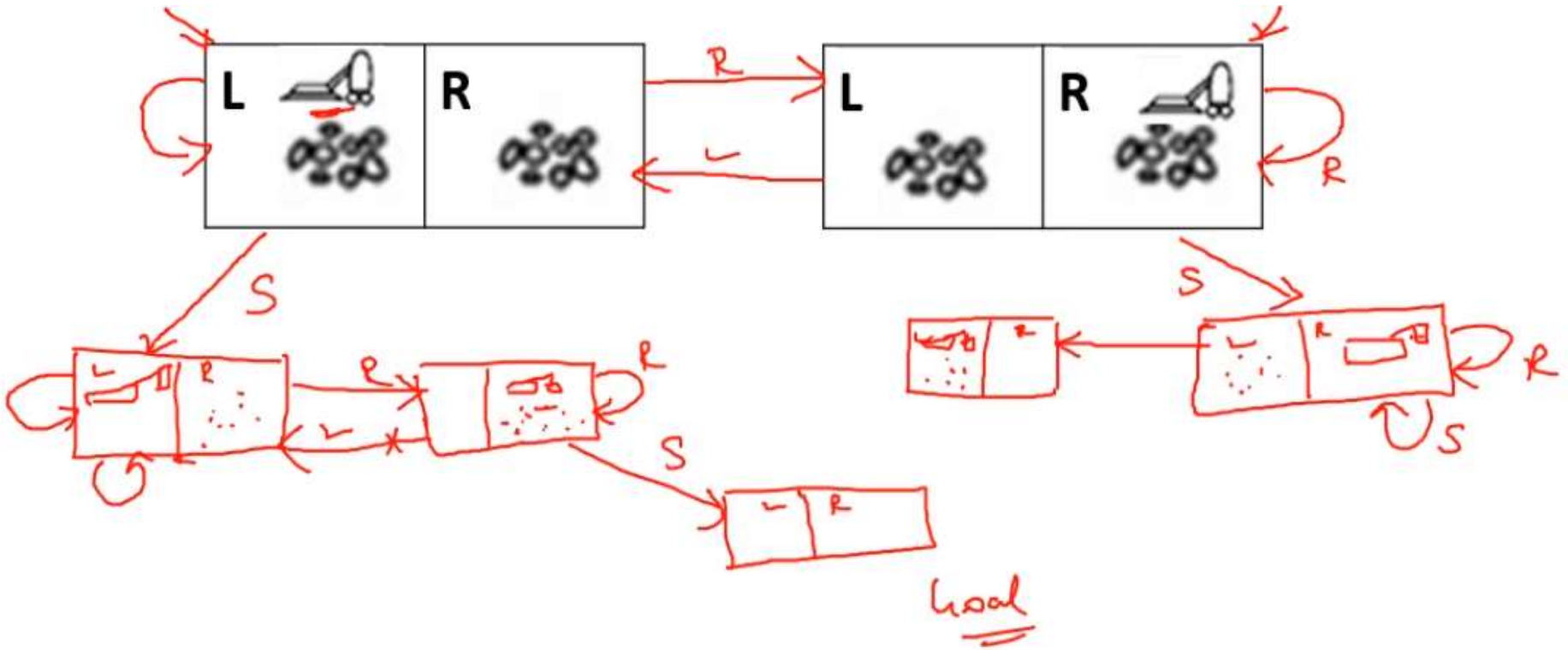
Vacuum World – Transition diagram



Possible Actions: 1. May move Right -> No Operation

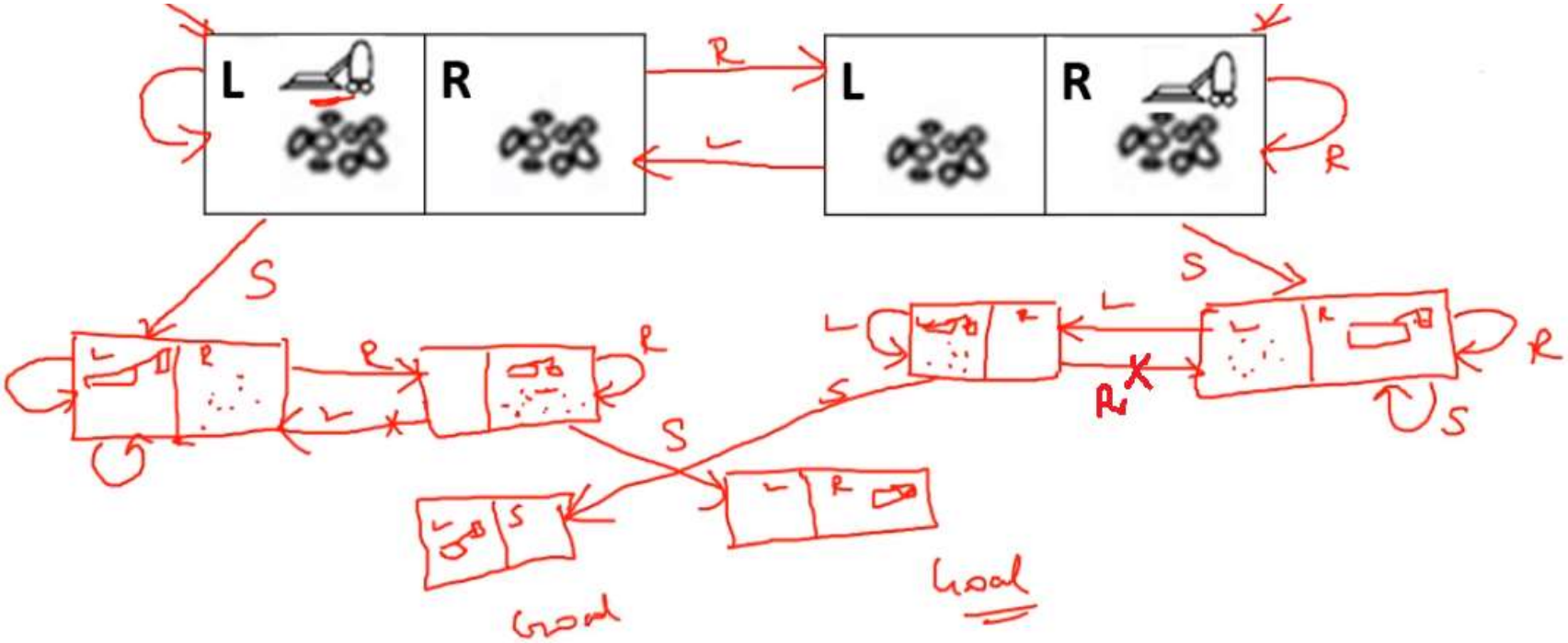
2. **May move Left** -> Not required here(Already cleaned only)
3. S -> Suck -> Only Possible move

Vacuum World – Transition diagram



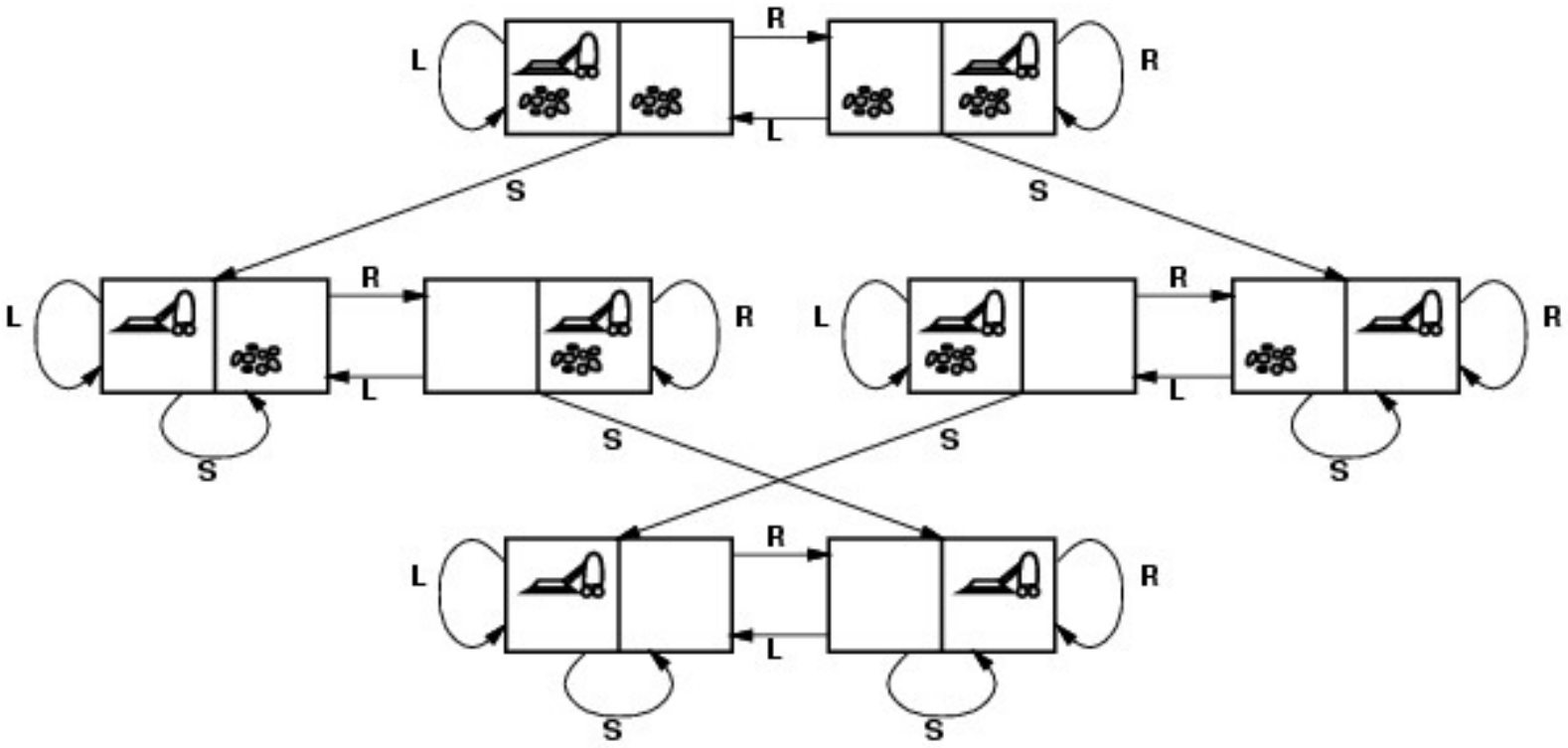
Possible Actions: 1. May move Right -> No Operation (Agent is in same location)
2. **May move Left** -> Only Possible move
3. S -> Suck -> No Operation (Already cleaned)

Vacuum World – Transition diagram



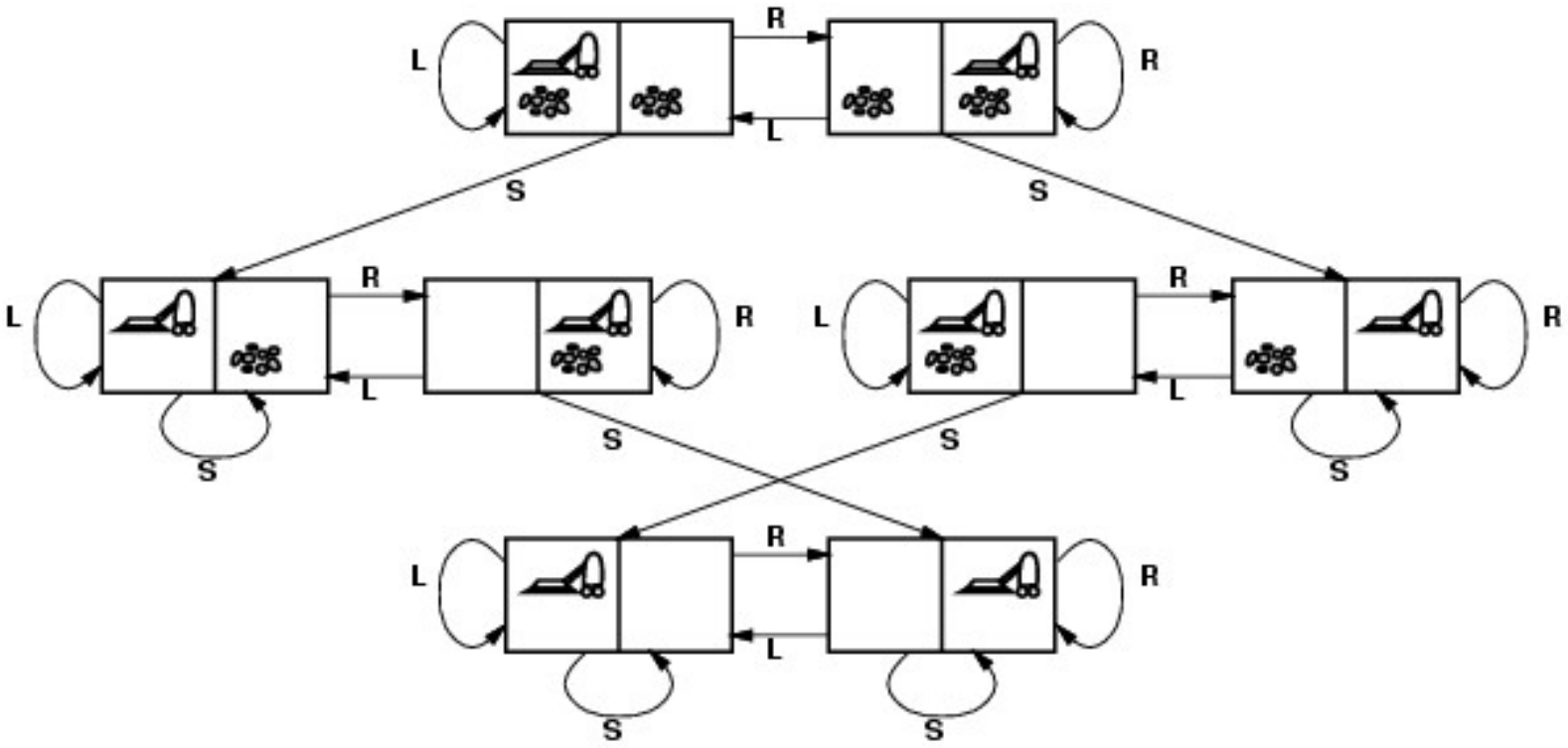
Possible Actions:
1. May move Right -> Not required (Already cleaned)
2. **May move Left** -> No Operation
3. S -> Suck -> Suck Operation

Vacuum world state space graph



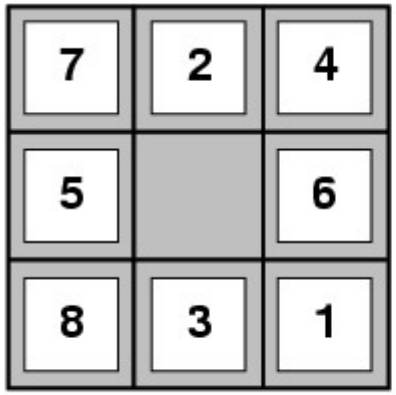
- states?
- Initial State?
- actions?
- goal test?
- path cost?

Vacuum world state space graph

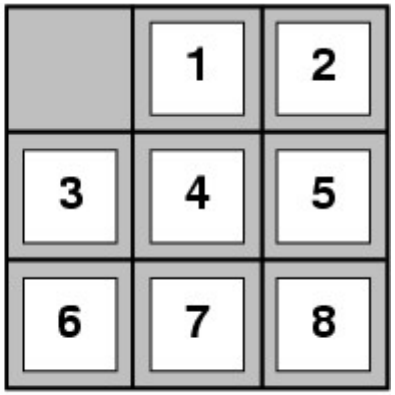


- states? 2 locations with or without dirt and robot location
- Initial State ? Any state can be initial
- actions? *Left*, *Right*, *Suck*
- goal test? no dirt at all squares (locations)
- path cost? No. of actions to reach the goal (1 per action)

Example 1: The 8-puzzle



Start State



Goal State

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

Toy Problem – 8 Puzzle Problem

- It consists of 3 x 3 board with 8 numbered tiles and a blank space
- A tile adjacent to the blank space can slide into the blank space.
- The objective is to reach the specified goal state.

2	1	6
4		8
7	5	3

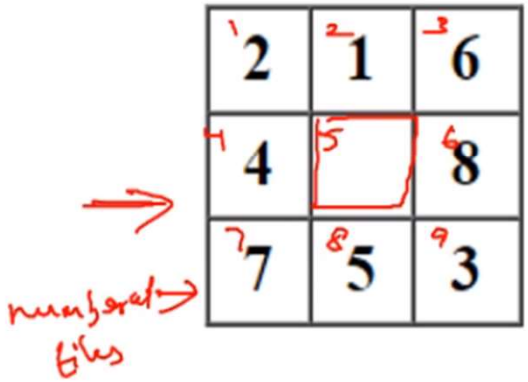
Initial State

1	2	3
8		4
7	6	5

Goal State

Toy Problem – 8 Puzzle Problem

- It consists of 3 x 3 board with 8 numbered tiles and a blank space
- A tile adjacent to the blank space can slide into the blank space.
- The objective is to reach the specified goal state.



Initial State

1	2	3
8		4
7	6	5

Goal State

8 Puzzle Problem…

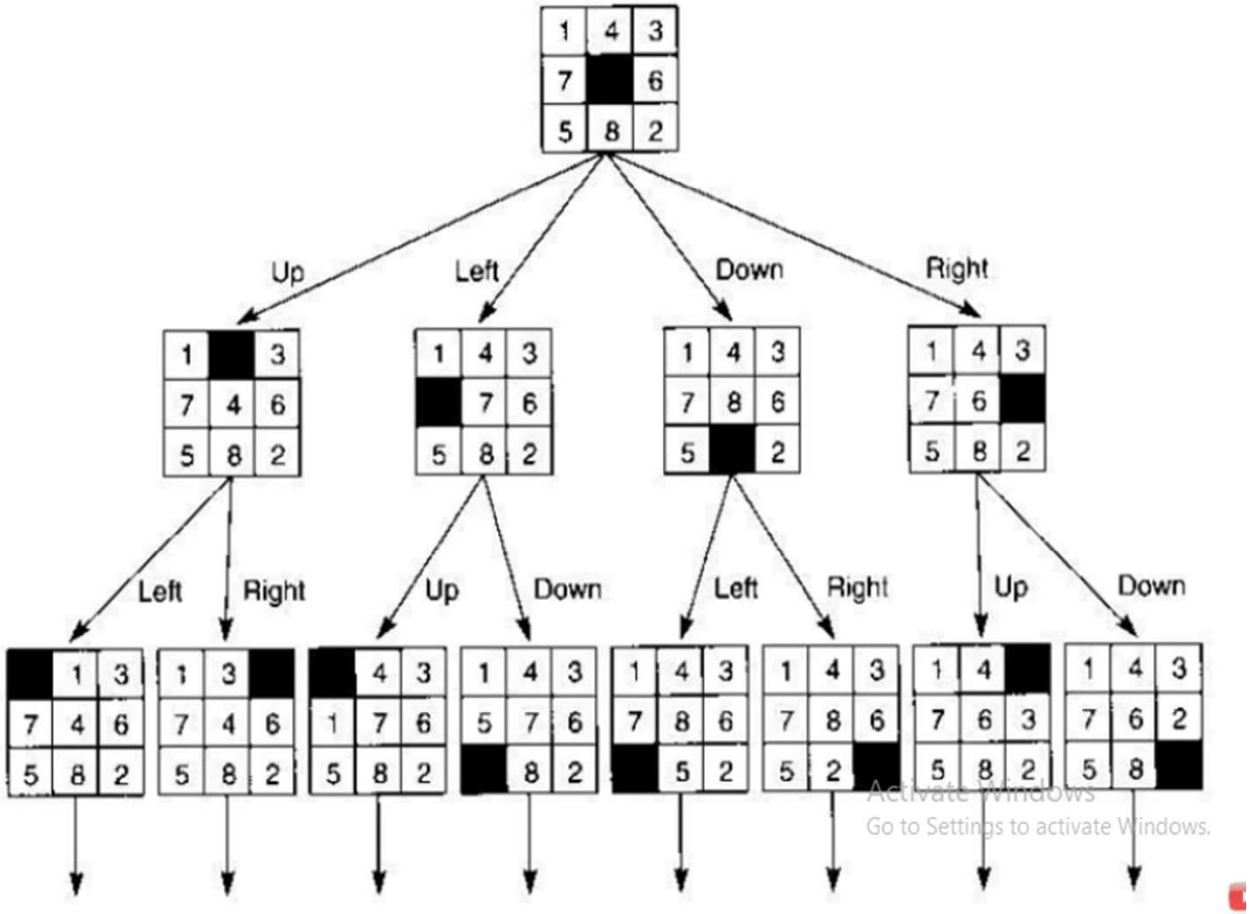
- States :
  - Integer location of each tile.
- Initial State:
  - Any state can be initial, (no fixed state for initial state)
- Successor Function:
  - Generates legal states that result from trying the four actions to move the blank
  - Left, Right, Top, Bottom
- Goal Test:
  - This checks whether the state matches the goal configuration
- Path Cost:
  - Each step costs 1, so the path cost is the number of steps in the path.

8 Puzzle Problem...

- States :
  - Integer location of each tile.
- Initial State:
  - Any state can be initial, (no fixed state for initial state)
- Successor Function:
  - Generates legal states that result from trying the four actions to move the blank
  - Left, Right, Top, Bottom
- Goal Test:
  - This checks whether the state matches the goal configuration
- Path Cost:
  - Each step costs 1, so the path cost is the number of steps in the path.

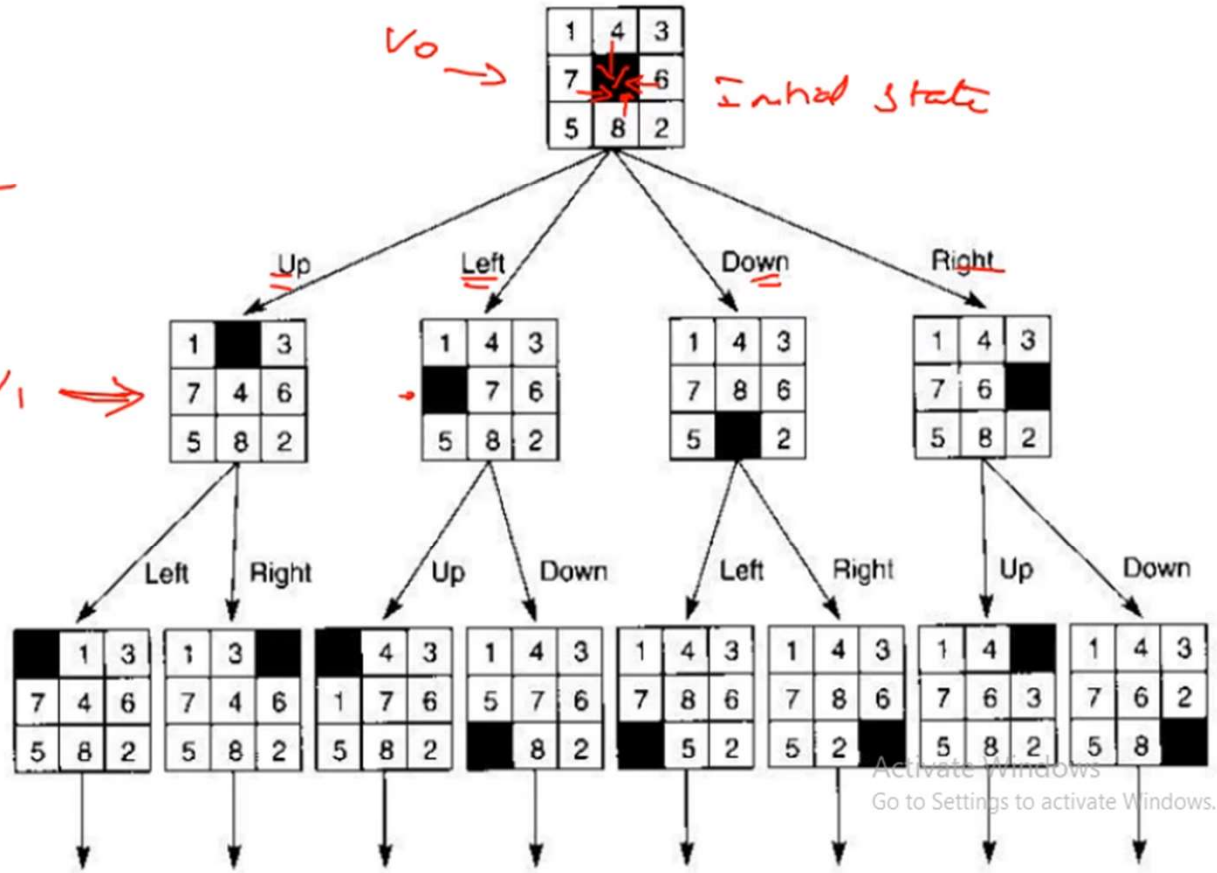
State space for 8 Puzzle Problem

- 9!/2=1,81,440 possible states and easily solved.
- All state space search problems required to find minimum cost between start state to goal state.

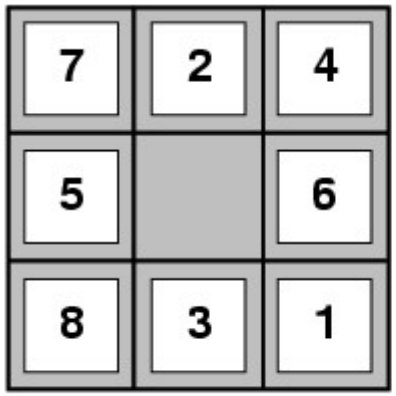


State space for 8 Puzzle Problem

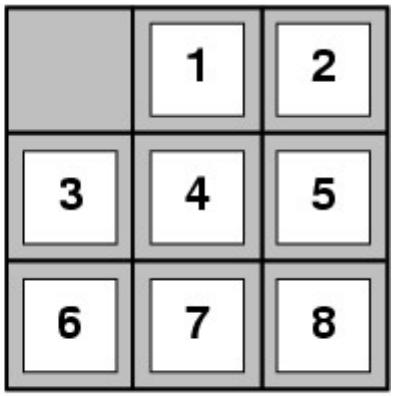
- 9!/2=1,81,440 possible states and easily solved.
- All state space search problems required to find minimum cost between start state to goal state.



Example 1: The 8-puzzle..contd



Start State

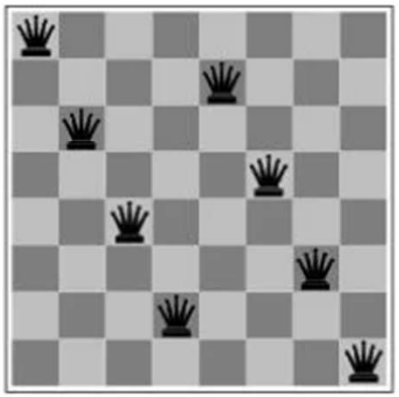


Goal State

- states? Integer locations of tiles
- Initial State? Any state can be initial
- actions? move blank left, right, up, down
- goal test? = goal state (given) – Goal configuration is reached
- path cost? No. of actions to reach the goal (1 per action)

[Note: optimal solution of *n*-Puzzle family is NP-hard]

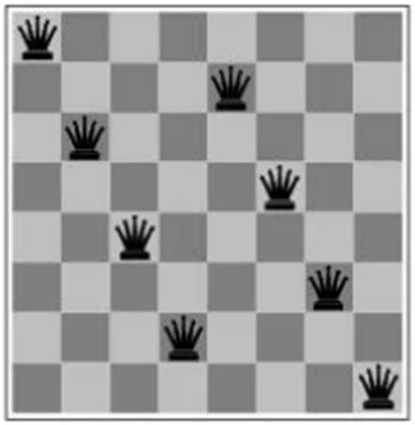
Example: 8-queens problem



Incremental formulation vs. complete-state formulation

- States??
- Initial state??
- Actions??
- Goal test??
- Path cost??

Example: 8-queens problem

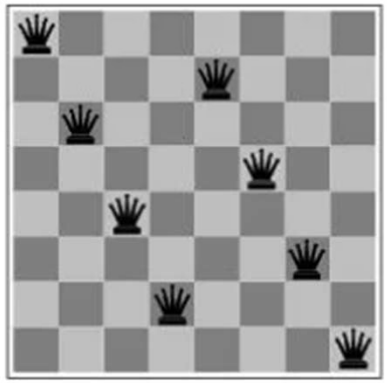


Incremental formulation

- States?? Any arrangement of 0 to 8 queens on the board
- Initial state?? No queens
- Actions?? Add queen in empty square
- Goal test?? 8 queens on board and none attacked
- Path cost?? None

  3×10^{14} possible sequences to investigate

Example: 8-queens problem

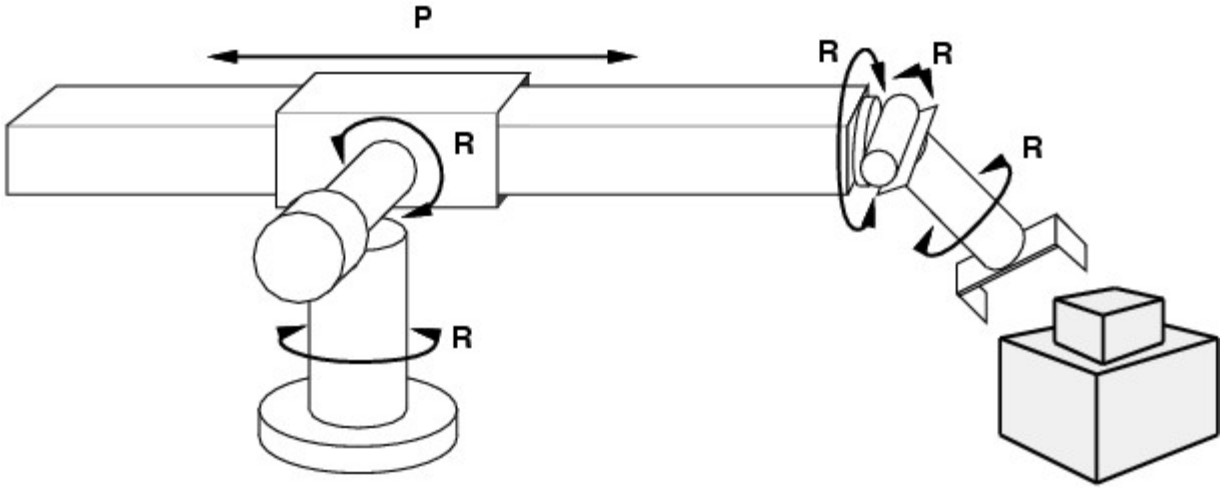


Incremental formulation (alternative)

- States?? n ($0 \leq n \leq 8$) queens on the board, one per column in the n leftmost columns with no queen attacking another.
- Actions?? Add queen in leftmost empty column such that is not attacking other queens

2057 possible sequences to investigate; Yet makes no difference when n=100

Example 2: robotic assembly



- states?: real-valued coordinates of robot joint angles parts of the object to be assembled
- Initial State?: Any arm position and object configuration
- actions?: continuous motions of robot joints
- goal test?: complete assembly
- path cost?: time to execute

Basic search algorithms

- How do we find the solutions of previous problems?
  - **Search the state space (remember complexity of space depends on state representation)**
  - **Here: search through *explicit tree generation***
    - ROOT= initial state.
    - Nodes and leafs generated through successor function.
  - **In general search generates a graph (same state through multiple paths)**

Possible Assessment Questions

Define state
Define state space
Define search tree
Define search node
Define goal
Define action
Define successor function
Define branching factor
Give the initial state, goal test, successor function and cost function for the following problem You have to color a planar map using only four colors in such a way that no two adjacent regions have the same color
Give the initial state, goal test, successor function and cost function for the following problem You have 3 jugs measuring 12 gallons, 8 gallons and 3 gallons and a water faucet. You can fill the jugs up or empty them out from one to another or onto the ground. You need to measure but exactly on gallon