CS 520: Introduction to Artificial Intelligence

Prof. Louis Steinberg

Lecture 2:

state spaces

uninformed search

Review

What is AI?

- **A set of goals**
  - **build an artificial intelligence**
  - **useful subgoals**
- **A class of problems**
  - **characteristics common to these goals**
- **A set of methods**
  - **commonly useful in solving problems like these**
- **A set of people**

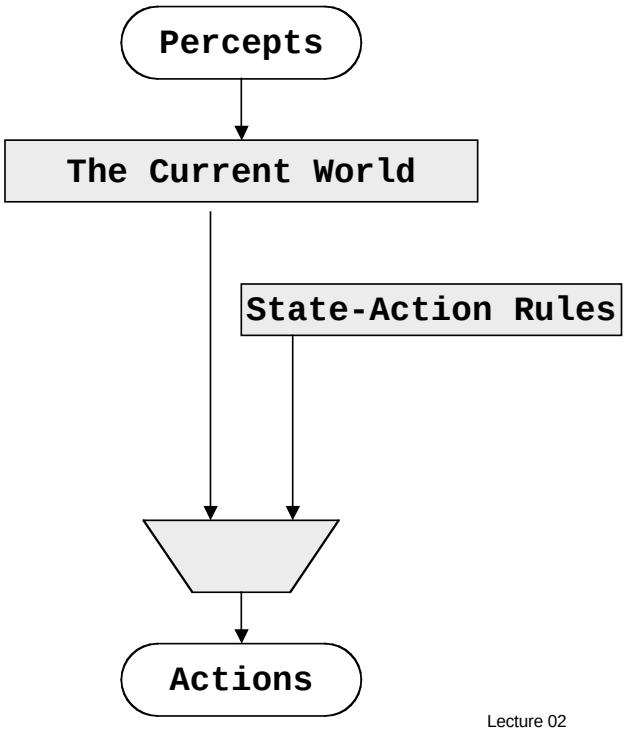
Review

Agents

- **Act in the world**
  - **Sensors, percepts**
  - **Effectors, actions**
  - **Goals**
  - **Environment**

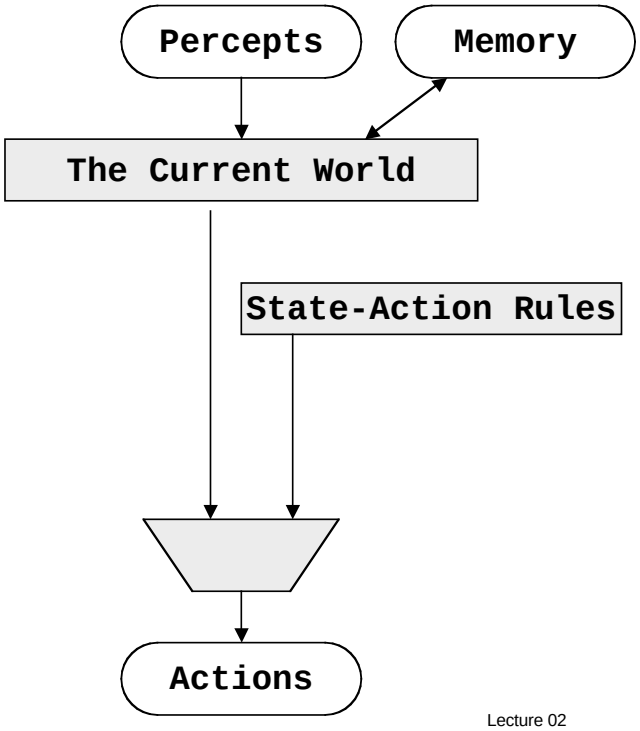
Classes of agent:

- **Reflex**
- **Reflex with State**
- **Planning with Goals**
- **Planning with Utility**



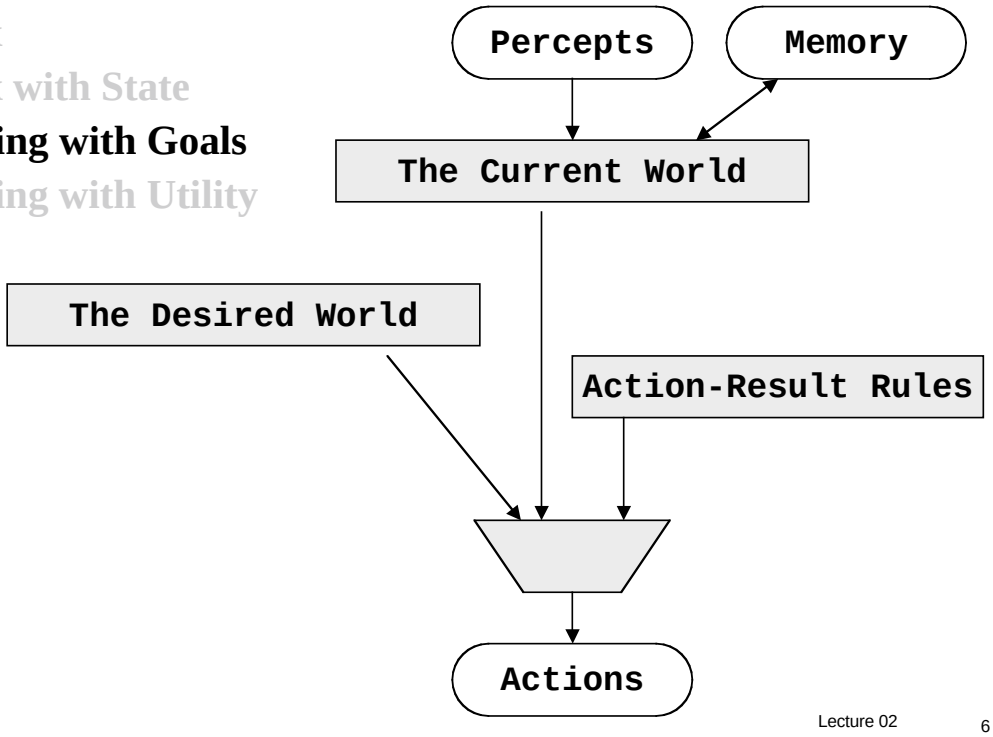
Classes of agent:

- Reflex
- **Reflex with State**
- Planning with Goals
- Planning with Utility



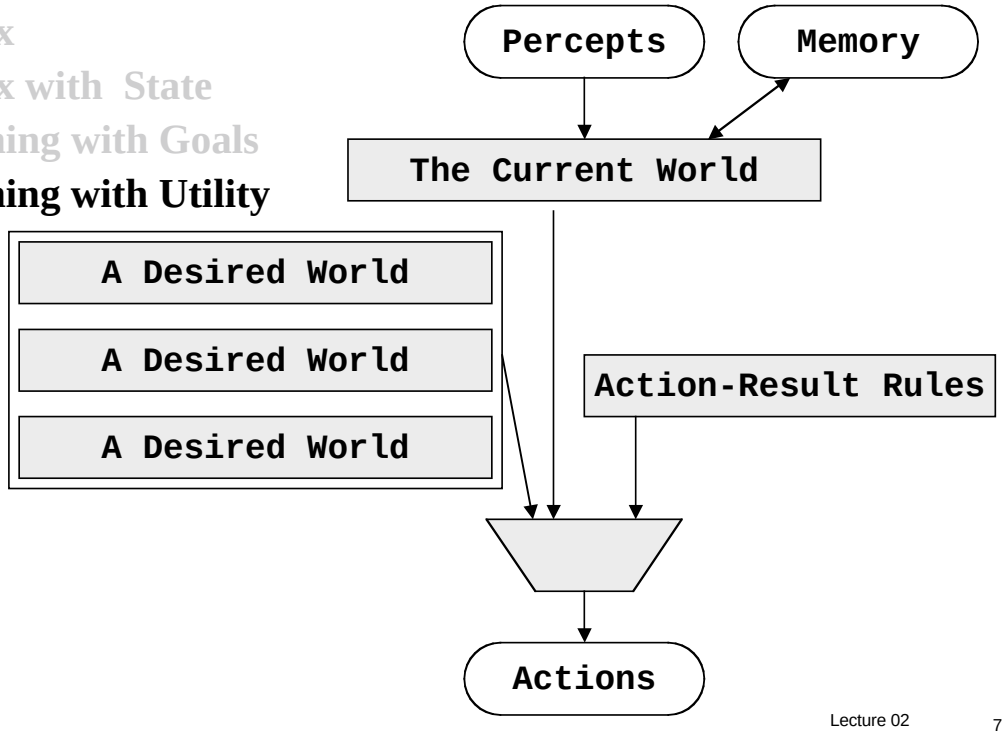
Classes of agent:

- Reflex
- Reflex with State
- **Planning with Goals**
- Planning with Utility



Classes of agent:

- Reflex
- Reflex with State
- Planning with Goals
- **Planning with Utility**



A State Space

- **Given:**
  - **A set of states**
  - **An initial state**
  - **A set of operators mapping states to states**
    - **Preconditions of each operator**
    - **Effects of operator**
    - **(Cost of each operator)**
  - **A set of goal states**

A State Space

- **Find: A (minimal cost) sequence of operators that transforms the initial state into a goal state.**
- **Kinds of cost**
  - **search cost: cost to find solution**
  - **path cost: cost to execute operators once the sequence is known**
  - **solution cost (or value): cost (value) of specific goal state**

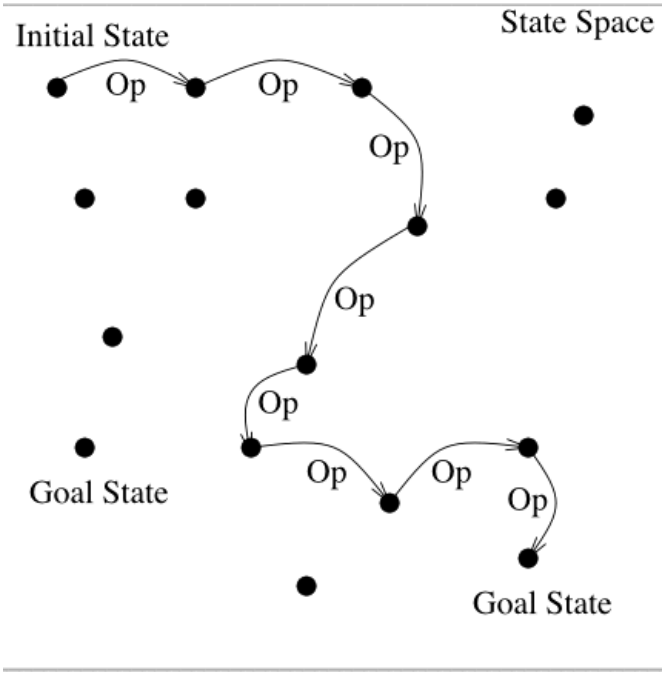
Lectures on Search

Formulation of search problems.

- **State Spaces**

- **Uninformed (blind) search algorithms.**
- **Informed (heuristic) search algorithms.**
- **Constraint Satisfaction Problems.**
- **Game Playing Problems.**

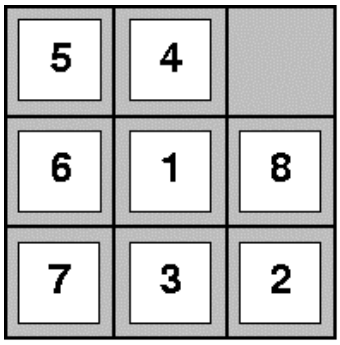
A State Space Illustration



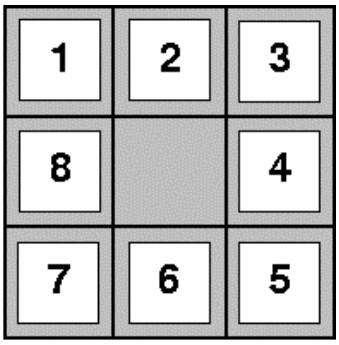
Two classes of problems

- **Space exists, search only:**
  - **A start state.**
  - **A graph G = (States,Edges).**
    - **Possibly represented by adjacency matrix.**
    - **Possibly represented by adjacency lists.**
  - **A list of goal states.**
- **Generate space while searching:**
  - **A start state.**
  - **A set of operators.**
  - **A goal test.**

The 8-puzzle



Start State

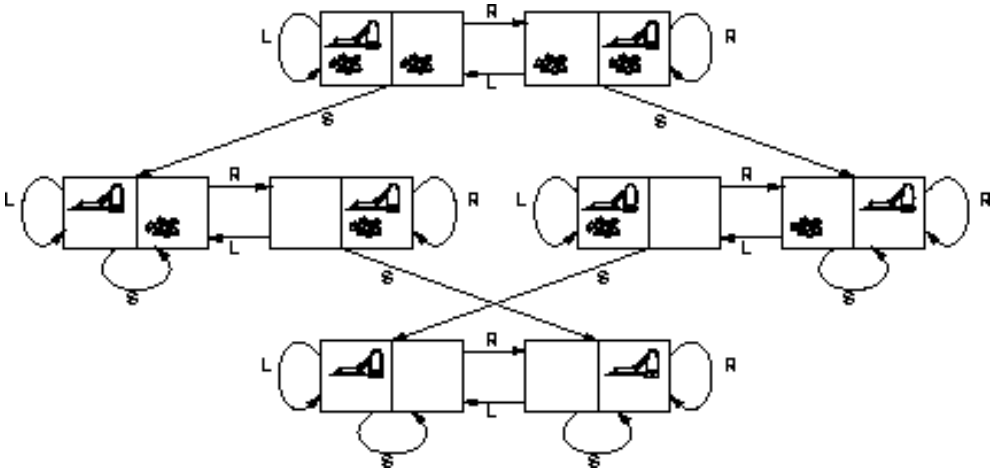


Goal State

The 8-puzzle

- **Problem formulation as search:**
  - **states: location of each tile as well as blank tile.**
  - **operators: blank moves left, right, up, or down.**
  - **goal test: state matches goal state shown.**
  - **path cost: each move costs 1; path cost=path length.**

The vacuum world

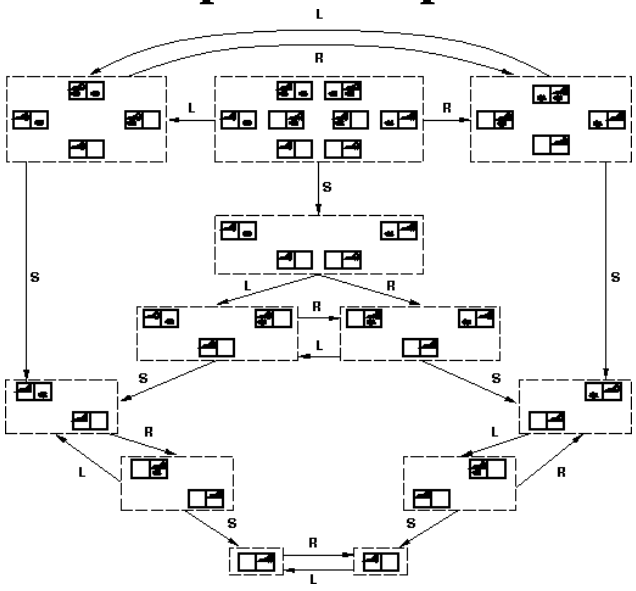


The vacuum world

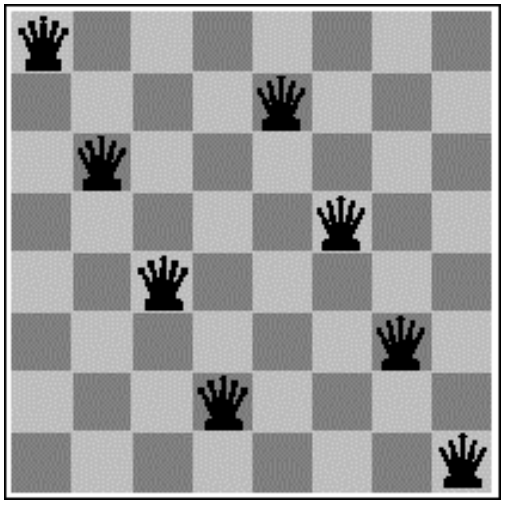
- **Problem formulation as search:**
  - **states: one of the 8 shown above.**
  - **operators: move left, move right, or suck.**
  - **goal test: no dirt left in any square.**
  - **path cost: each action costs 1; path cost=path length.**

The vacuum world (no sensors)

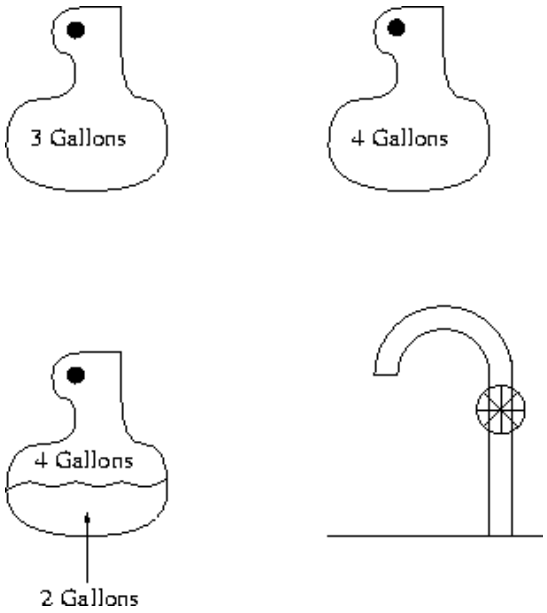
- **This is a multiple state problem.**



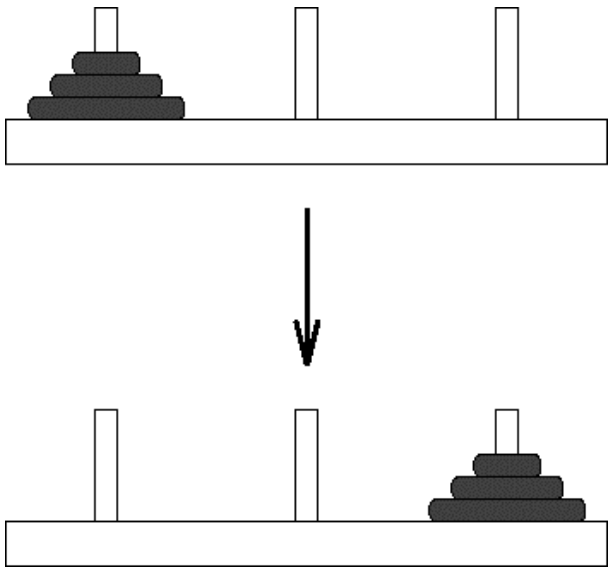
The 8-queens problem



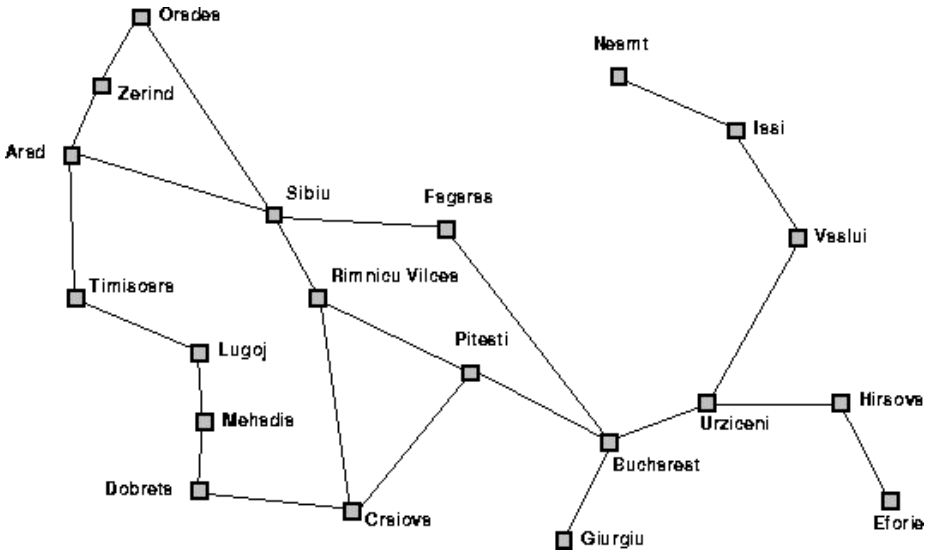
Water Jug Problem



Towers of Hanoi Problem



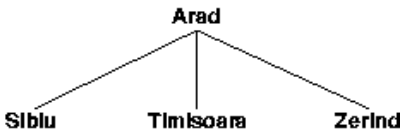
Route (Path) finding



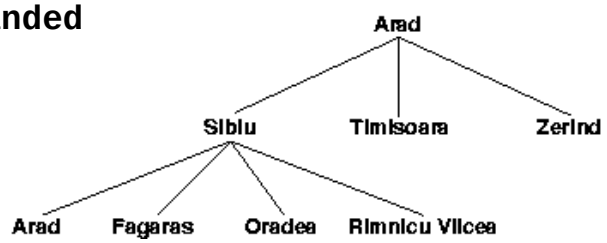
Solving the Route (Path) finding problem by search

Initial state

Arad expanded



Sibu expanded

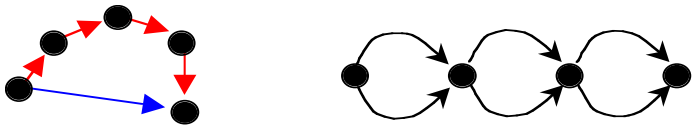


Lectures on Search

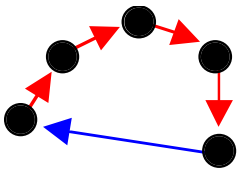
- **Formulation of search problems.**
- **Uninformed (blind) search algorithms.**
- **Informed (heuristic) search algorithms.**
- **Constraint Satisfaction Problems.**
- **Game Playing Problems.**

Complications

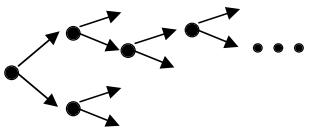
- **Multiple paths**



- **Cycles**



- **Infinite paths**



Search Method Evaluation Criteria

- **Correctness: Are all ``solutions'' found by the algorithm correct?**
- **Completeness: Does the algorithm find a solution, whenever one exists?**
- **Termination: Is the algorithm guaranteed to terminate on all problems?**
- **Solution Optimality: Is the algorithm guaranteed to find optimal solutions?**
- **Complexity: How much time and space does the algorithm use?**
- **Algorithm Optimality: Does the algorithm use as little time or space as possible?**

Costs

- **Depth: d**
- **Branching factor: b**

	Time	Space
Depth First	b^d	d
Breadth First	b^d	$b^{(d-1)}$

Other Pros & Cons

	Cycles & Infinite Paths	Shortest Path first	Recursive
Depth First	Incomplete	No	Yes
Breadth First	Complete	Yes	No

Limited Depth Depth-first Search

```
Lddfs(node, limit)
  If solution(node) return node;
  If (limit == 0)
        then return false
        else loop for child in expand(node)
              {result = Ldffs(child, limit-1);
               if result then return result}
```

- Completeness of breadth-first … up to depth limit

Iterative Deepening

- **Low space cost, recursive like depth first**
- **Complete, shortest first like breadth first**
- **Small constant-factor extra time cost**
- **Algorithm:**
  - **Loop for d from 1 to infinity**
    - **Do a limited-depth depth-first to depth d**
    - **If solution found, return it**
- **Cost: $d*b^1 + (d-1)b^2 + b^3 + \ldots + b^d$**

  which is $O(b^d)$, and in fact not much more than b^d