Discussion

CS 5/7320
Artificial Intelligence

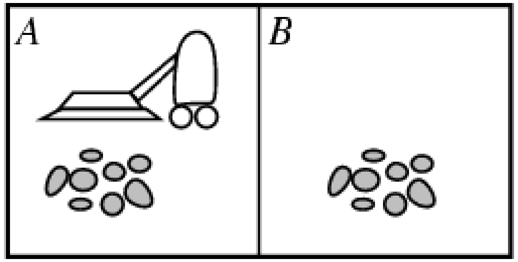
Search with Uncertainty

AIMA Chapters 4.3-4.5

Slides by Michael Hahsler
with figures from the AIMA textbook

Online Material

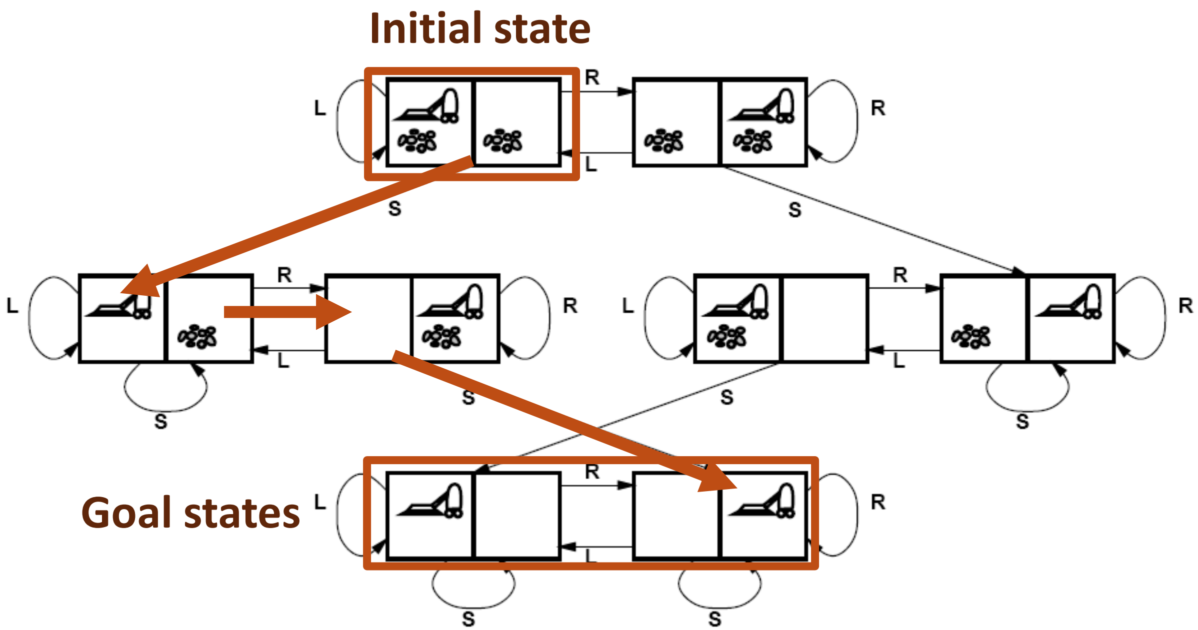
Recap: Solving Search Problems under Certainty



No Uncertainty

- **Full observability**: The agent always knows (=can observe) the state.
- **Deterministic environment** with a known transition model $Result(s, a) = s'$ The agent can predict the outcome of its actions.

State space: A state completely describes the condition of the environment and the agent.



Solution: Use tree search in the planning phase to create a **sequence of actions** also called a **plan.** Then blindly execute the plan: **[Suck, Right, Suck]**

Sources and Consequence of Uncertainty

Sources: The environment may be

- **Not fully observable**: The agent may be uncertain about its current state.
- **Stochastic (transition function)**: The agent may not be able to perfectly predict the outcome of its actions.

Consequences:

1. The agent needs to keep track of all the states it could be in. This set is called a ***belief state.***

2. A fixed precomputed plan (sequence of actions) does not work for stochastic transition functions, but a

   conditional plan (also called strategy or policy)

   that depends on percepts is needed.

Types of uncertainty in the environment*

Nondeterministic Actions:
Outcome of an action in a state is uncertain.

No observations:
Sensorless problems.

Partially observable environments:
The agent cannot directly observe the state of the environment.

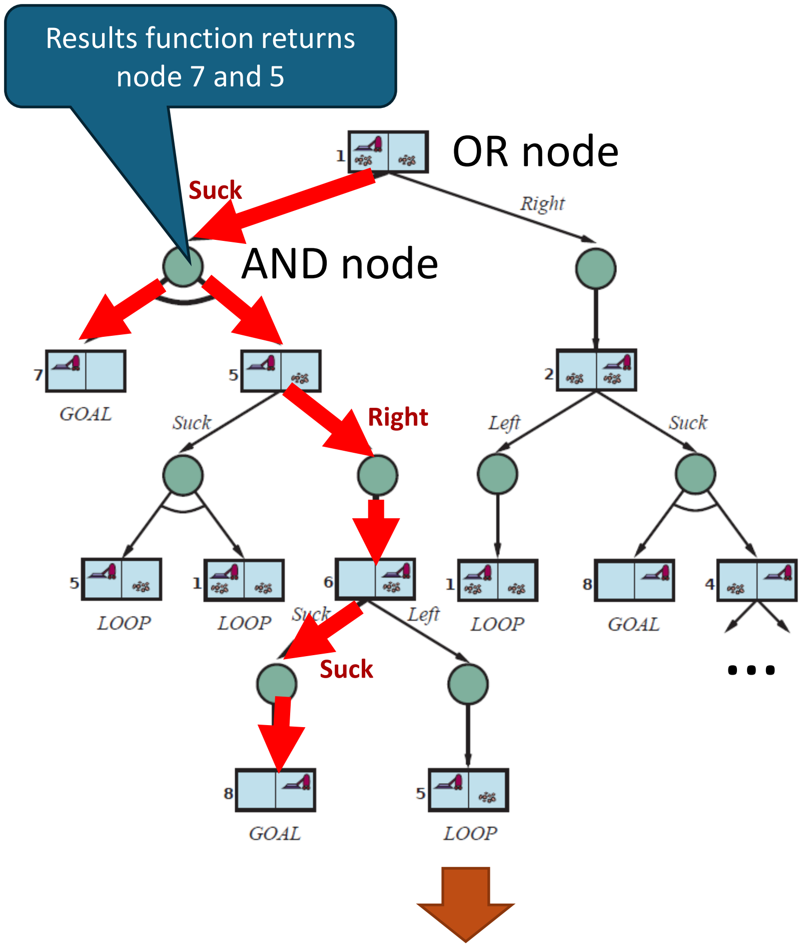
Exploration:
Unknown environments and online search.

* we will quantify uncertainty with probabilities later.

Nondeterministic Actions

Stochastic Environment (Stochastic Transition Model)

Search the AND-OR Tree



- **Goal**: Find a subtree with one action for each OR node and considering all outcomes of the AND nodes that has only goal leaf nodes.
- Descend the tree depth-first:
  - OR node: trying one action at a time.
  - AND node: consider all outcomes and check recursively.
  - Ignore cycles.
  - Abandon a subtree if not all leaf nodes are the desired goal nodes.
  - Stop when **the first complete subtree with only goal leaf nodes is found.**
- Construct the conditional plan that represents the subtree starting at the root node.

Conditional Plan:
[Suck, **if** State = 5 **then** [Right, Suck] **else** []]

Example

Playing Tic-Tac-Toe

Search With No Observations

Using actions to “coerce” the world into a smaller set of known states

Sensorless Problems

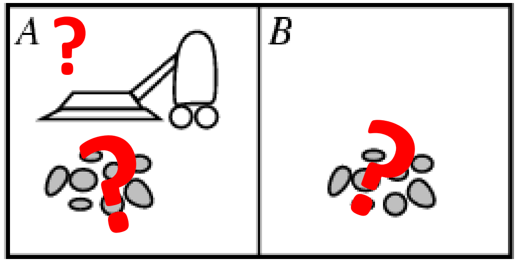
Conformant problem: The agent has no sensors, so the environment is not observable.

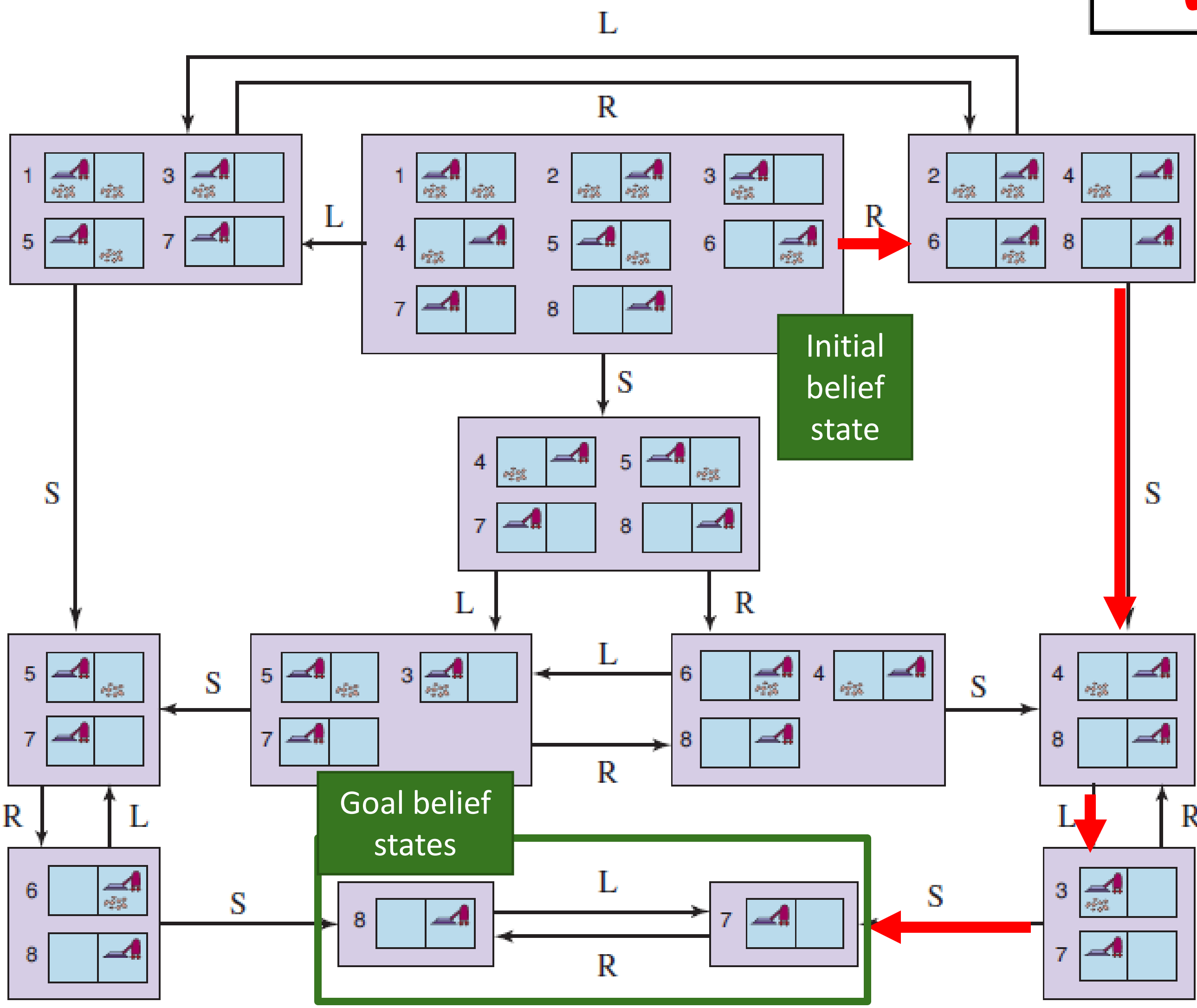
Why is this useful?

- **Example**: Doctor prescribes a broad-band antibiotic instead of performing time-consuming blood work to find a more targeted antibiotic. This saves time and money.

- **Basic idea**: Find a solution (a **plan**) that **works (reasonably well) from any state** and then just blindly execute it.

Find a Path in the Reachable Belief State Space



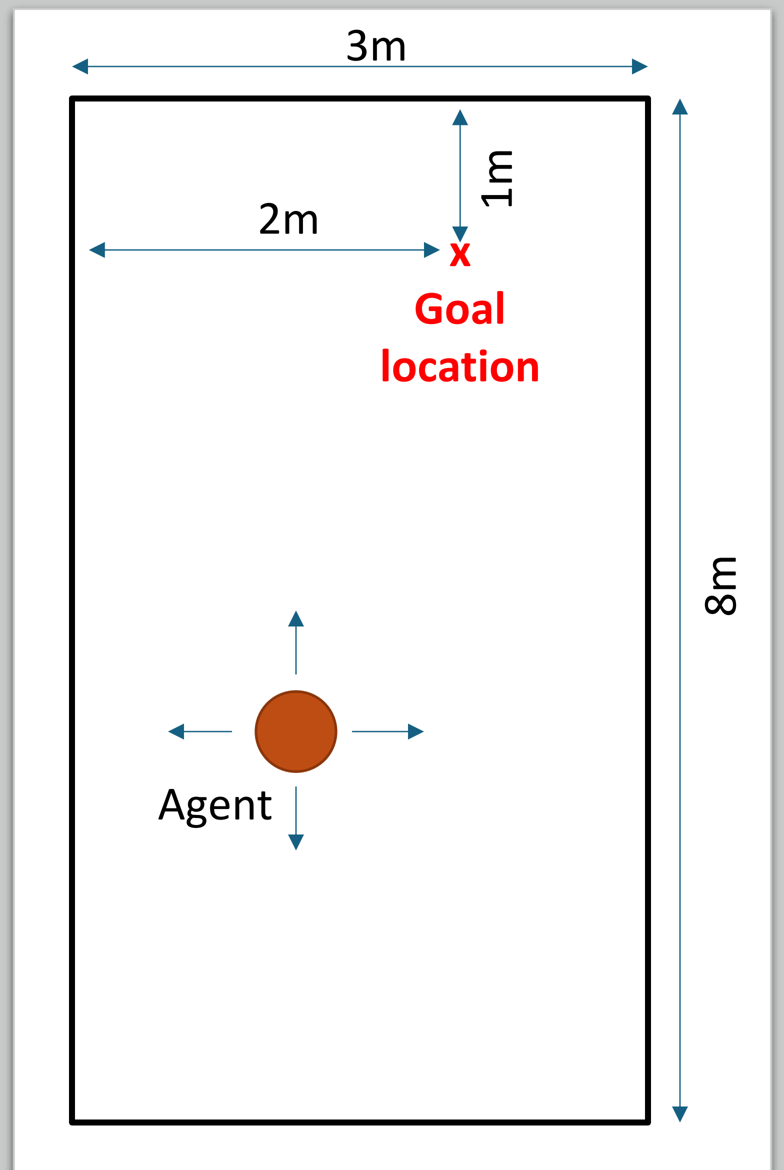


The size of the belief state space is the powerset of the original N states:

$$\mathcal{P}_s = 2^N = 2^8 = 256$$

Only a small fraction (12 belief states) are reachable by actions.

No observations, so we get a solution sequence from an initial belief state:
[Right, Suck, Left, Suck]



Case Study

The agent can move up, down right, and left.

The agent has **no sensors** and does not know its current location.

1. Can you navigate to the goal location? How?

2. What would you need to know about the environment?

3. What type of agent can do this?

Partially Observable Environments

Using Observations to Learn About the State

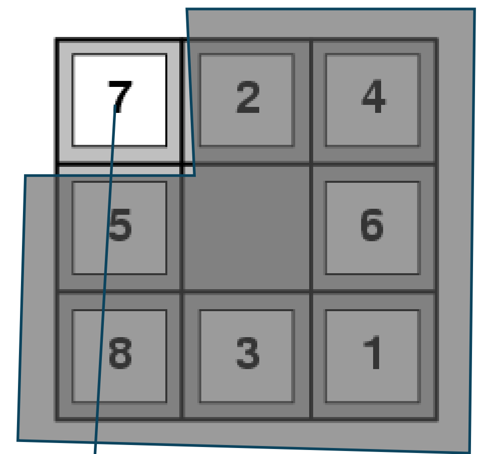
Percepts and Observability

- Many problems cannot be solved efficiently without sensing (e.g., 8-puzzle).
- We need to see at least one square.

Percept function: $Percept(s)$

…s is the state

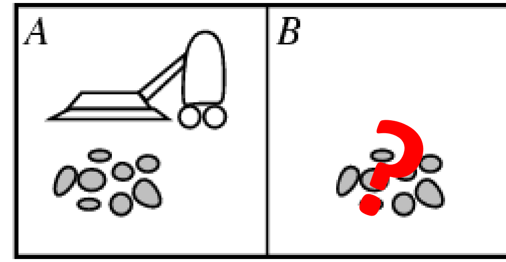
- **Fully observable**: $Percept(s) = s$
- **Sensorless**: $Percept(s) = None$
- **Partially observable**: $Percept(s) = o$
  o is called an observation and tells us something about s



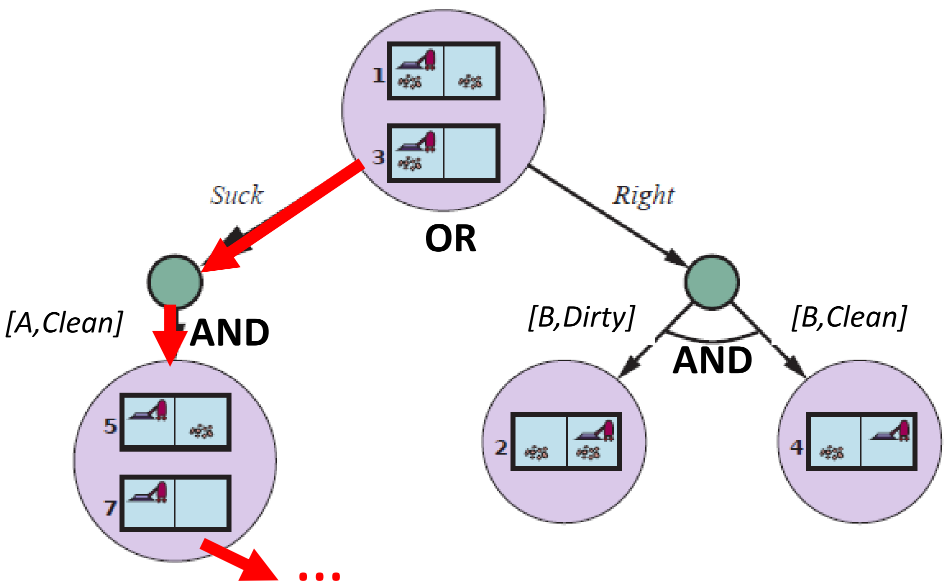
$Percept(s) = Tile7$

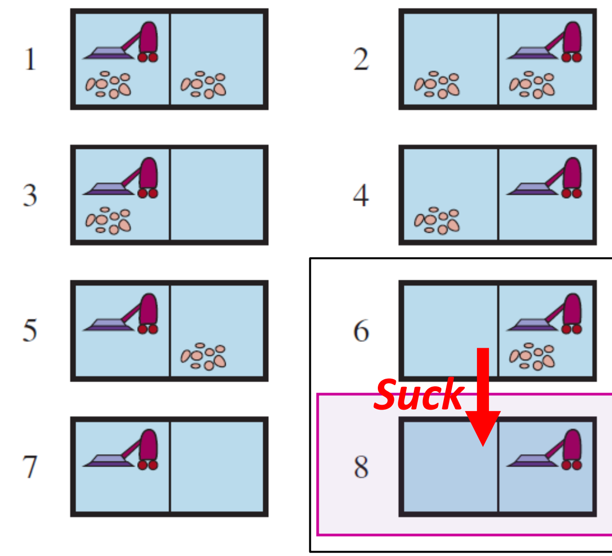
Problem: Many states (different order of the hidden tiles) can produce the same observation!

Solving Partially Observable Problems 4



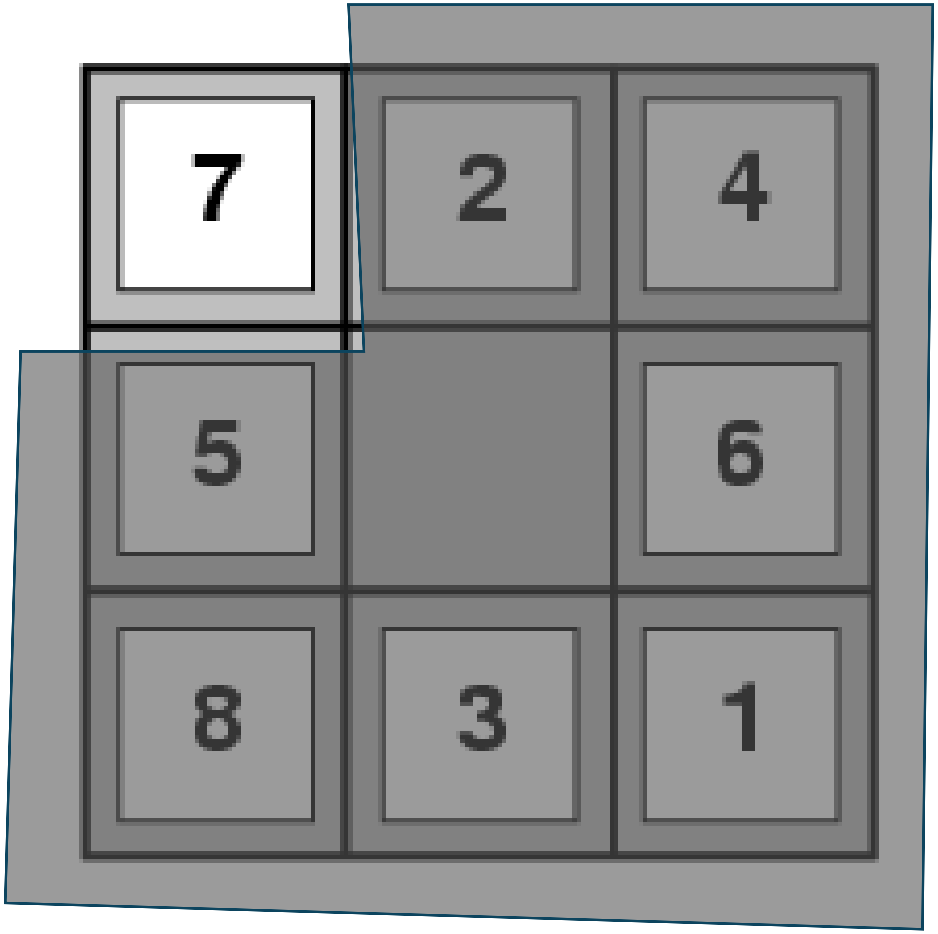
Use an AND-OR tree on **belief states** to create a conditional plan





Plan: *[Suck, Right,* **if** *b = {6}* **then** *Suck* **else** *[]]*

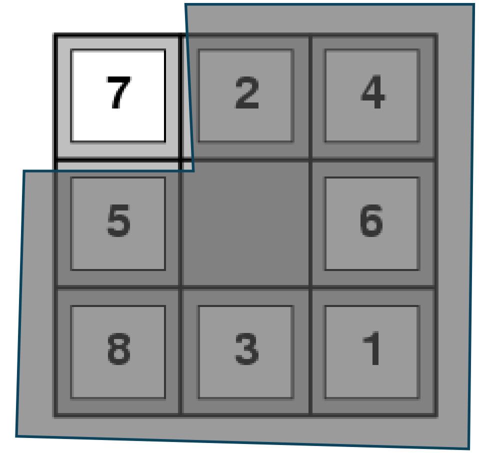
b = {6} is the result of the update with *o = [B, Dirty]*



Case Study:

Partially Observable 8-Puzzle

Partially Observable 8-Puzzle



Give a problem description for this problem.

- States:
- Initial state:
- Actions:
- Transition model:
- Goal test:
- Percept function:

This problem can be solved using an AND-OR Tree, but is there an easier solution?

a. What type of agents would we use?
b. What algorithms can be used?

Exploration

Unknown Environments and Online Search

Recap: Offline Search

- **Offline search aka planning**: Create a plan using the state space and the transition model before taking any action.
- The **plan** can be
  - **a sequence of actions,** or
  - **a conditional plan** that uses observations to account for uncertainty or imperfect observability.
- The agent plans using search with the known transition function to predict the consequence of actions.

- **Issue**: In an **unknown environment,** we do not know the transition function.
- We cannot predict outcomes of actions; therefore, we cannot plan using offline search!

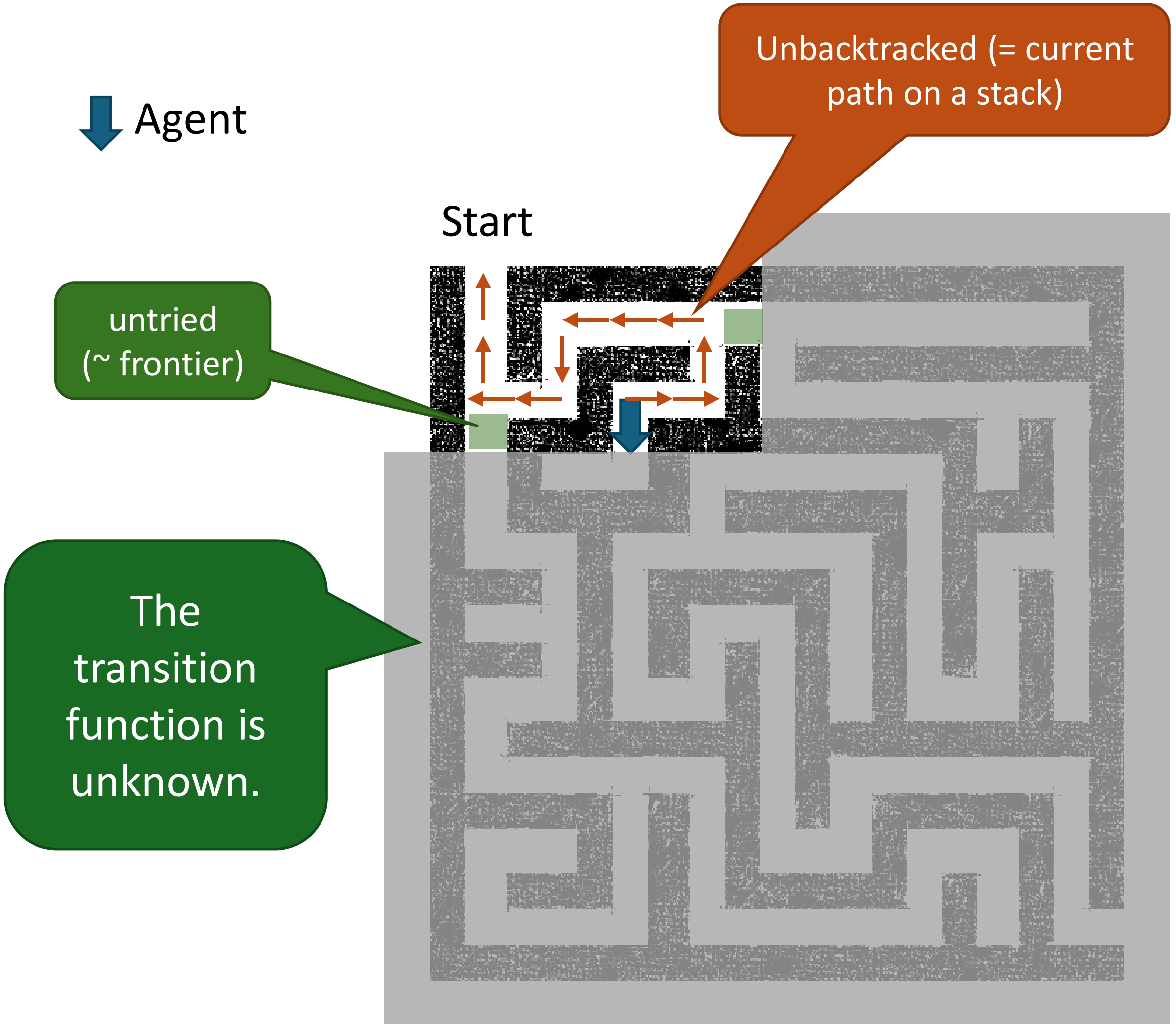
Online Search

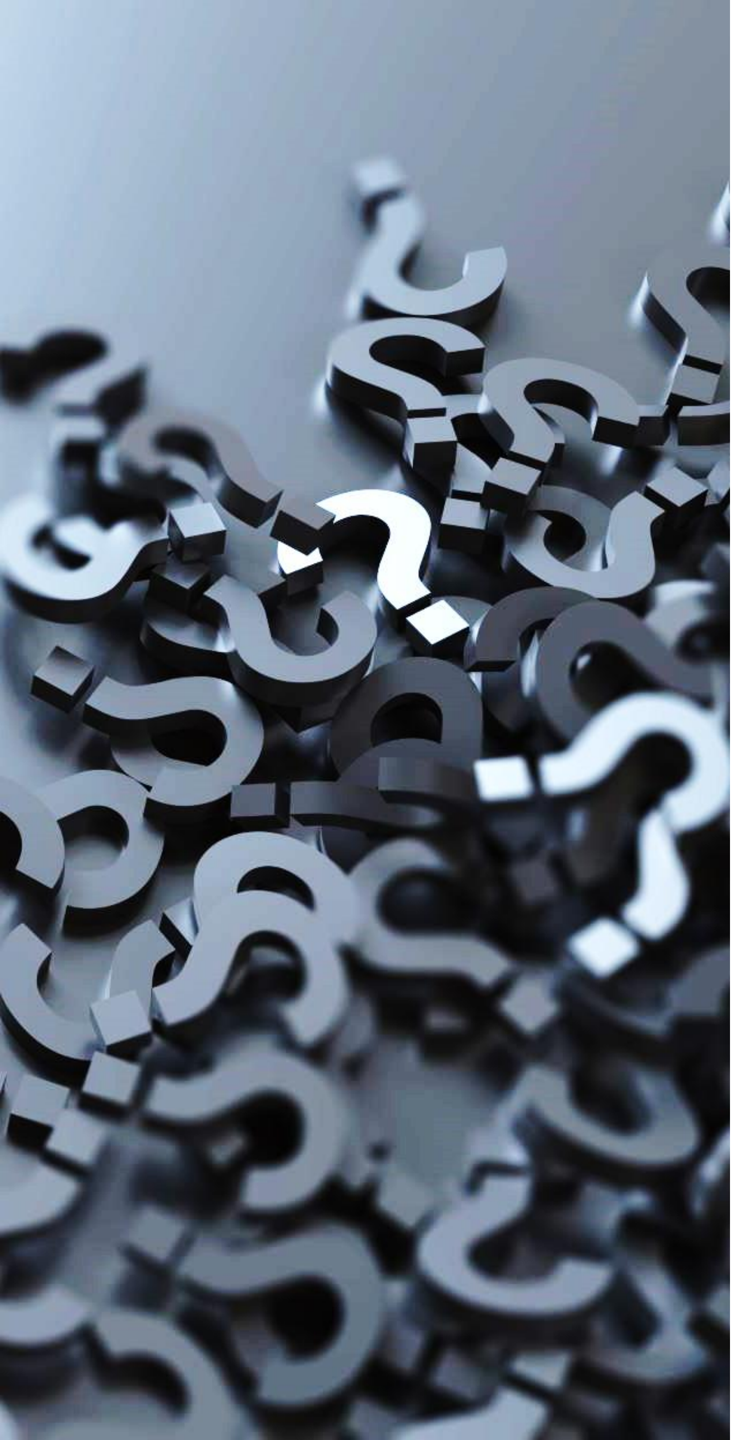
- **Online search** does not use planning! It explores the real world one action at a time. Offline prediction and update are replaced by "act" and "observe."

- Useful for
  - **Unknown environment**: The agent has no complete model of how the environment works. It needs to explore an unknown state space and/or what actions do. I.e., it needs to **learn the transition function**
    $$f : S \times A \rightarrow S$$
  - **Real-time problems**: When offline computation takes too long, and there is a penalty for sitting around and thinking.
  - **Nondeterministic domain**: Conditional plans become very large. Only focus on what happens instead of planning for everything!

Case Study: DFS with Backtracking for an unknown Maze

- We don’t have a map of the maze. We can only see adjacent squares.
- We cannot plan so we must explore by walking around!
- A simple method is to store the path for backtracking to get back to untied actions when we run into a dead end (think leaving breadcrumbs or a string).
- This is an iterative implementation of DFS without a reached data structure. Unbacktaced represents the currently explored path, and untried represents the frontier. DFS memory management applies.





Important concepts that you should be able to explain and use now…

- Difference between solution types:
  a. a fixed action sequence (a plan),
  b. a **conditional plan** (also called a strategy or policy), and
  c. **exploration**.
- What are **belief states**?
- How actions can be used to coerce the world into known states.
- How actions and observations (from **percept functions**) can be used to learn about the state: State estimation with repeated predict and update steps.
- The use of AND-OR trees to solve small problems.
- Large problems are hard!