State Space Search
Overview

- Problem-solving as search
- How to formulate an AI problem as search.
- Uninformed search methods
What is search?
Environmental factors needed

- **Static** — The world does not change on its own, and our actions don't change it.

- **Discrete** — A finite number of individual states exist rather than a continuous space of options.

- **Observable** — States can be determined by observations.

- **Deterministic** — Action have certain outcomes.
Terminology

• A **state** is a set of properties that define the current conditions of the world our agent is in.
  – The entire set of possible states is called the **state space**.
• The **initial state** is the state the agent begins in.
• A **goal state** is a state where the agent may end the search.
• An agent may take different **actions** that will lead the agent to new states.
Formulating problems as search

- *Canonical problem*: route-finding
- Sliding block puzzle
- 8 queens puzzle
- Roomba cleaning
- Automatic CS 172 proof completion
- Solitaire
- What else?
Formulating problems as search

• Define:
  – What do my states look like?
  – What is my initial state?
  – What are my goal state(s)?
  – What is my cost function?
    • How do I know how "good" a state or action is?
Formulating problems as search

• Solution:
  – A *path* between the initial state and a goal state.
  – *Quality* is measured by path cost.
  – *Optimal solutions* have the lowest cost of any possible path.
• State space search gives us graph searching algorithms.

• Are we searching a *tree* or a (true) *graph*?
Often-confusing point

- There are two simultaneous graph-ish structures used in search:
  - (1) Tree or graph of underlying state space.
  - (2) Tree maintaining the record of the current search in progress (the search tree).
Infrastructure needed

• A node $n$ of the search tree stores:
  – a state (of the state space)
  – a parent pointer to a node (usually)
  – the action that got you from the parent to this node (sometimes)
  – the path cost $g(n)$: cost of the path so far from the initial state to $n$.

• Frontier is often stored as a stack, queue, or priority queue.

• Explored set is often stored using a data structure that enables quick look-up for membership tests.
Uninformed search methods

• These methods have no information about which nodes are on promising paths to a solution.
• Also called: *blind search*
• Question — What would have to be true for our agent to need uninformed search?
  – No knowledge of goal location; or
  – No knowledge of current location or direction (e.g., no GPS, inertial navigation, or compass)
How do you evaluate a search strategy?

• **Completeness** — Does it always find a solution if one exists?
• **Optimality** — Does it find the best solution?
• **Time complexity**
• **Space complexity**
Search strategies

• Breadth-first search
  – Variant — Uniform-cost search
• Depth-first search
• Depth-limited search
• Iterative deepening depth-first search
  – Variant — iterative lengthening search
Breadth-first search

• Choose shallowest node for expansion.

• Data structure for frontier?
  – Queue (regular)

• Suppose we come upon the same state twice. Do we re-add to the frontier?
  – No.

• Complete? Optimal? Time? Space?