Agents

Reflex Agent

- Has an idea of the world now
- Makes a decision
- Doesn't consider consequences of decisions

Planning Agents

- Asks 'what if'
- Decisions based on what it thinks consequences of actions may be
- Must have model of how the world evolves
- Consider how the world would be

Search Problems

- Consists of four elements
  1. A state space $\triangleq$ all of the possible configurations of the world
  2. A successor function (with actions, cost) $\triangleq$ function that takes in one state and decides the possible futures
  3. The start state
  4. A goal test $\triangleq$ function $\Longrightarrow$ whether output is a goal or not
- Solution $\triangleq$ sequence of actions that transform the start state to the goal state
- They are models

State Space

- World state $\triangleq$ every single detail of the environment
- Search state $\triangleq$ only the details needed for planning
- Contains above elements as well

Sizes

- How many possible configurations of the world are there
- Naively can be done by determining the permutations

State Space Graph

- A mathematical representation of a search problem
- Node $\triangleq$ state / world configurations
  - Only one node per state
- Directed edges $\triangleq$ evaluation of successor function
- Rarely can be built in memory
  - Generally use search trees because of this

Search Trees

- Evaluate successor states from a specific point in time
- Can have duplicate nodes / world states
  - Can be infinite if connected
- 'what if' tree of plans and their outcomes
- Root $\triangleq$ start state
- Child $\triangleq$ possible successor states
- Generally never build the entire tree

- Maintain a fringe of nodes that have not yet been evaluated
  - Starts with the start space
  - Expand elements to the fringe if not at goal space from the element popped off the fringe
  - Remove element from fringe if evaluated
- Branching factor b
- Maximum depth m
- Solutions may exist on various depths

Properties

- Complete $\triangleq$ guaranteed to find a solution if one exists
- Optimal $\triangleq$ guaranteed to find the least cost path
- Time complexity
- Space complexity

DFS

- Always takes the deepest thing off the fringe (future with the most actions applied to it)
  - Fringe $\triangleq$ LIFO stack
- Expands left prefixes on tree
- $O(b^m)$ time
- $O(bm)$ space (you leave b unexplored items at every level)
- Complete $\triangleq$ m could be infinite, so only if we prevent cycles
- Optimal $\triangleq$ No; find the 'leftmost' solution regardless of depth or cost

BFS

- Always take the shallowest thing on the fringe
  - Fringe $\triangleq$ FIFO queue
- $s \triangleq$ depth of solution
- $O(b^s)$ time
- $O(b^s)$ space (worse than DFS)
- Complete $\triangleq$ yes if $s \in$ finite
- Optimal $\triangleq$ iff costs are 1

Iterative Deepening

- Idea: get DFS's space advantages with BFS's time / shallow-solution advantages
- Run DFS with depth of i where i advances by 1 per solution
- out of scope

Uniform Cost Search

- Variation of BFS
- Always expand the cheapest future
  - Also have to keep track of how costly the state is on the fringe
- $C^* \triangleq$ cost of solution
- $\varepsilon \triangleq$ minimal cost of traversal
- Effective depth $\widehat{=} \frac{C^*}{\varepsilon}$
- $O(b^{C^*/\varepsilon})$ time and space
- Complete $\triangleq$ yes if finite cost and minimum arc cost
- Optimal $\triangleq$ yes (A*)