# Problem Solving and Search

Andrea Danyluk
February 6, 2017

## Announcements

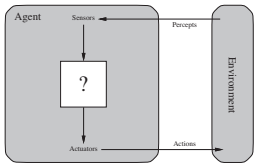
- Progamming Assignment 0: Python Tutorial
  - Optional / Ungraded
  - Posted last week
  - Due Thursday at 11pm
  - No CS Unix account? Let me know!

## Today's Lecture

- Agents
- Goal-directed problem-solving and search
- Uninformed search
  - Breadth-first
  - Depth-first
- Formulating a problem as a search problem

## Rational Agents

- An agent perceives and acts.
- "Doing the right thing" captured by a performance measure that evaluates a given sequence of environment states.



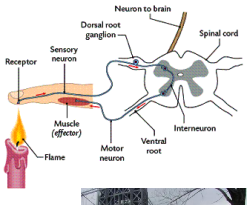

A rational agent:
selects an action that is expected to maximize the performance measure, given evidence provided by the percept sequence and whatever built-in knowledge the agent has.

- Rational ≠ omniscient
  - Percepts may not supply all relevant information
- Rational ≠ clairvoyant
  - Action outcomes may not be as expected
- Rational ≠ successful

[ Adapted from Russell]

## Reflex Agents

- Act on the basis of the current percept (and possibly what they remember from the past).
- May have memory or a model of the world's state.
- Do not consider future consequences of their actions.




[Adapted from CS 188 UC Berkeley]

## Goal-based Agents

- Plan ahead
- Ask "what if"
- Decisions based on (hypothesized) consequences of actions
- Have a model of how the world evolves in response to actions

[Adapted from CS 188 UC Berkeley]

## Building a goal-based agent

- Determine the percepts available to the agent
- Select/devise a representation for world states
- Determine the task knowledge the agent will need
- Clearly articulate goal(s)
  - Including what to optimize
- Select/devise a problem-solving technique so that the agent can decide what to do

## Search as a Fundamental Problem-Solving Technique

- Originated with Newell and Simon's work on problem solving in the late 60s.

## Search Problems

A search problem consists of

- A state space
  - A set of states
  - As set of actions
  - A transition model that specifies results of applying actions to states
    - Successor function: Result(s, a)
- An initial state
- A goal test

## An Example: the 8-Puzzle

| | | |
|---|---|---|
| 1 | 2 | 3 |
| | 8 | 7 |
| 6 | 5 | 4 |

Initial State

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

Goal State

# possible distinct states = 9! = 362,880 (but only 9!/2 reachable)

## Real world examples

- Navigation
- Vehicle parking
- Parsing (natural and artificial languages)
  - The old dog slept on the porch
  - The old dog the footsteps of the young

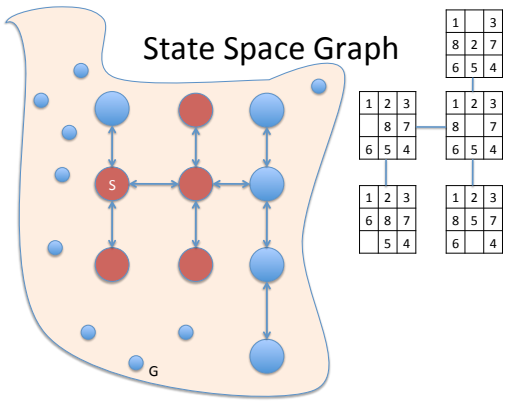
## And back to the 8-Puzzle

- States:
  - Puzzle configurations
- Actions:
  - Move blank N, S, E, or W
- Start state:
  - As given
- Goal test:
  - Is current state = specified goal state?

| | | |
|---|---|---|
| 1 | 2 | 3 |
| | 8 | 7 |
| 6 | 5 | 4 |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

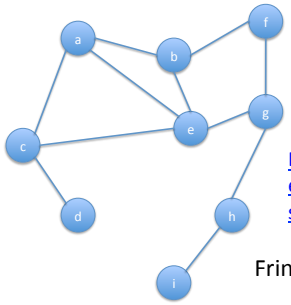
## State Space Graph




## Finding a solution in a problem graph

- Solving the puzzle = finding a path through the graph from initial state to goal state
- Simple graph search algorithms:
  - Breadth-first search
  - Depth-first search

## Breadth-first Graph Search: Review

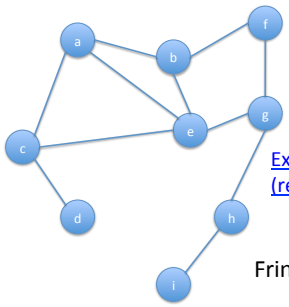



Start: a
Goal: i

Explore vertices (really edges) closest to the start state first.

Fringe is a FIFO queue

## Depth-first Graph Search: Review




Start: a
Goal: i

Explore deepest vertex (really edge) first.

Fringe is a LIFO stack

## Formalizing State Space Search

- A state space is a graph (V, E), where V is the set of states and E is a set of directed edges between states. The edges may have associated weights (costs).
- Our exploration of the state space using search generates a search tree.

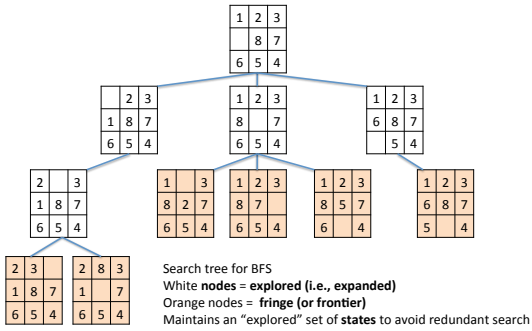
[Adapted from Eric Eaton]

## State Space Search in the AI World

- Rarely given a graph
- We don't build the graph before doing the search
  - Our search problems are BIG

## Search Trees

{N, E, S, W}



Search tree for BFS
White **nodes** = **explored (i.e., expanded)**
Orange nodes = **fringe (or frontier)**
Maintains an "explored" set of **states** to avoid redundant search

## Search Tree

- A "what if" tree of plans and outcomes
- Start state at the root node
- Children correspond to successors
- Nodes contain states; correspond to plans to those states
- Aim to build as little as possible
- Because we build the tree "on the fly" the representations of states and actions matter!

[Adapted from CS 188 UC Berkeley]

## Nodes in Search Trees

- A node in a search tree typically contains:
  - A state description
  - *A reference to the parent node*
  - The name of the operator that generated it from its parent
  - The cost of the path from the initial state to itself
  - Might also include its depth in the tree
- The node that is the root of the search tree typically represents the initial state

## Operators and Goal Tests

- Child nodes are generated by applying legal operators to a node
  - The process of expanding a node means to generate all of its successor nodes and to add them to the frontier.
- A goal test is a function applied to a state to determine whether its associated node is a goal node

## Solutions in Search Trees

- A solution is either
  - A sequence of operators that is associated with a path from start state to goal or
  - A state that satisfies the goal test
- The cost of a solution is the sum of the edge costs on the solution path
  - If all edges have the same (unit) cost, then the solution cost is just the length of the solution (i.e., the length of the path)

## Framing a Problem as Search

- 8 Queens
  - States?
  - Goal test?
  - Operators?