• Project 1: Constants
• Lab: Evals, Project 2
• Lab 8: due Friday @ midnight
Goal Based Agents

- Goal-based agents:
  - Plan ahead
  - Ask “what if”
  - Decisions based on (hypothesized) consequences of actions
  - Must have a model of how the world evolves in response to actions
  - Act on how the world WOULD BE
Search Problems

• A search problem consists of:
  – A state space
  – A successor function
  – A start state and a goal test

• A solution is a sequence of actions (a plan) which transforms the start state to a goal state
Example: Romania

- **State space:**
  - Cities
- **Successor function:**
  - Go to adj city with cost = dist
- **Start state:**
  - Arad
- **Goal test:**
  - Is state == Bucharest?
- **Solution?**
State Space Graphs

• State space graph: A mathematical representation of a search problem
  – For every search problem, there’s a corresponding state space graph
  – The successor function is represented by arcs

• We can rarely build this graph in memory (so we don’t)
State Space Sizes?

- Search Problem: Eat all of the food
- Pacman positions: $10 \times 12 = 120$
- Food count: 30
A search tree:
- This is 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
- For most problems, we can never actually build the whole tree
A search problem consists of:

- A state space
- A successor function
- A start state and a goal test

A solution is a sequence of actions (a plan) which transforms the start state to a goal state.
Example: Romania

- State space:
  - Cities
- Successor function:
  - Go to adj city with cost = dist
- Start state:
  - Arad
- Goal test:
  - Is state == Bucharest?
- Solution?
• **Search:**
  – Expand out possible plans
  – Maintain a *fringe* of unexpanded plans
  – Try to expand as few tree nodes as possible
General Tree Search

```plaintext
function Tree-Search(problem, strategy) returns a solution, or failure
initialize the search tree using the initial state of problem
loop do
  if there are no candidates for expansion then return failure
  choose a leaf node for expansion according to strategy
  if the node contains a goal state then return the corresponding solution
  else expand the node and add the resulting nodes to the search tree
end
```

• Important ideas:
  – Fringe
  – Expansion
  – Exploration strategy
General Tree Search

function Tree-Search(problem, strategy) returns a solution, or failure
initialize the search tree using the initial state of problem
loop do
  if there are no candidates for expansion then return failure
  choose a leaf node for expansion according to strategy
  if the node contains a goal state then return the corresponding solution
  else expand the node and add the resulting nodes to the search tree
end

• Important ideas:
  – Fringe
  – Expansion
  – Exploration strategy

• Main question: which fringe nodes to explore?
Example: Tree Search
State Graphs vs. Search Trees

We construct both on demand – and we construct as little as possible.

Each NODE in the search tree is an entire PATH in the problem graph.
States vs. Nodes

- Nodes in state space graphs are problem states
  - Represent an abstracted state of the world
  - Have successors, can be goal / non-goal, have multiple predecessors
- Nodes in search trees are plans
  - Represent a plan (sequence of actions) which results in the node’s state
  - Have a problem state and one parent, a path length, a depth & a cost
  - The same problem state may be achieved by multiple search tree nodes

Problem States

Search Nodes

Depth 5

Depth 6
Depth First Search

Strategy: expand deepest node first

Implementation: Probably want to use recursion
Breadth First Search

Strategy: expand shallowest node first

Implementation: Fringe is a FIFO queue