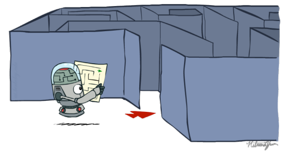
Announcements

- To be released by tomorrow morning
  - Slides
  - Access to edX Edge
  - Project 0: Python Tutorial
  - Math self-diagnostic on web --- optional, but important to check your preparedness for second half
  - Make sure you join the class on Piazza (so far … 28 enrolled)

1

CSC 481: Artificial Intelligence

Search, DFS



Instructor: Marco Alvarez

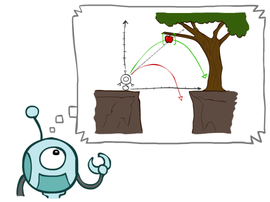
University of Rhode Island

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

2

Today

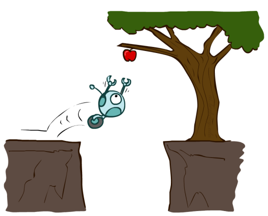
- Agents that Plan Ahead
- Search Problems
- Uninformed Search Methods
  - Depth-First Search



3

Reflex Agents

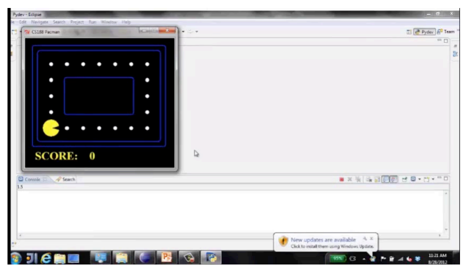
- Reflex agents:
  - Choose action based on current percept (and maybe memory)
  - May have memory or a model of the world's current state
  - Do not consider the future consequences of their actions
  - Consider how the world IS



[Demo: reflex optimal (L2D1)]
[Demo: reflex optimal (L2D2)]

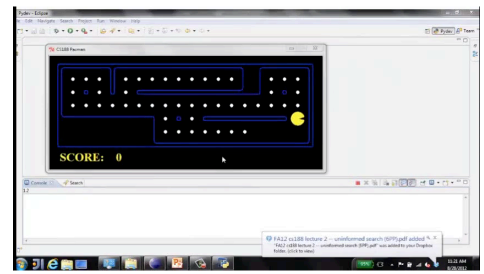
4

Video of Demo Reflex Optimal



5

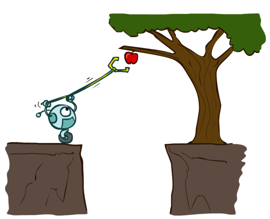
Video of Demo Reflex Odd



6

Planning Agents

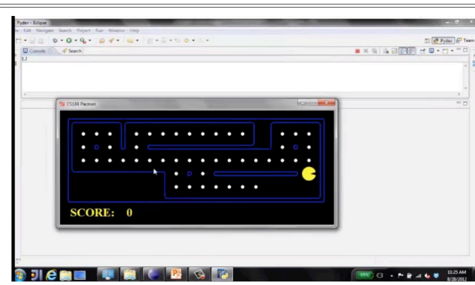
- Planning agents:
  - Ask "what if"
  - Decisions based on (hypothesized) consequences of actions
  - Must have a **model** of how the world evolves in response to actions
  - Must formulate a **goal** (test)
  - Consider how the world WOULD BE
- Optimal vs. complete planning
- Planning vs. replanning



[Demo: replanning (L2D3)]
[Demo: mastermind (L2D4)]

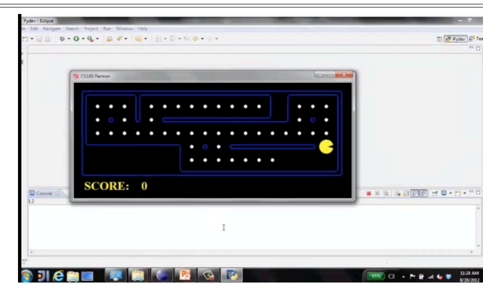
7

Video of Demo Mastermind



8

Video of Demo Replanning

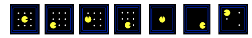
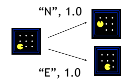


9

Search Problems

10

Search Problems

- A search problem consists of:
  - A state space 
  - A successor function (with actions, costs) 

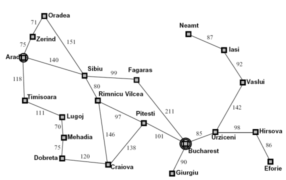
  - A start state and a goal test
- A solution is a sequence of actions (a plan) which transforms the start state to a goal state

11

Search Problems Are Models

12

Example: Traveling in Romania

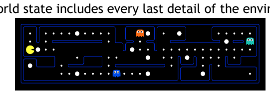


- State space:
  - Cities
- Successor function/Action:
  - Roads: Go to adjacent city with **cost** = distance
- Start state:
  - Arad
- Goal test:
  - Is state == Bucharest?
- Solution?

13

What's in a State Space?

The world state includes every last detail of the environment



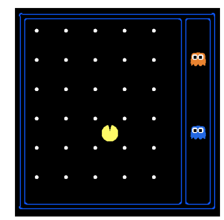
A search state keeps only the details needed for planning (abstraction)

- Problem: Pathing
  - States: (x,y) location
  - Actions: NSEW
  - Successor: update location only
  - Goal test: is (x,y)=END
- Problem: Eat-All-Dots
  - States: {(x,y), dot booleans}
  - Actions: NSEW
  - Successor: update location and possibly a dot boolean
  - Goal test: dots all false

14

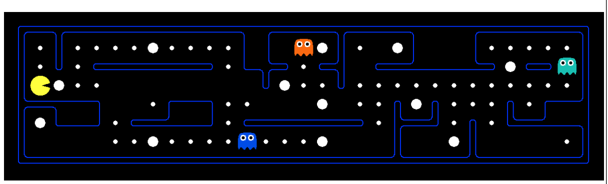
State Space Sizes?

- World state:
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW
- How many
  - World states?
    $120 \mathrm{x}(2^{30}) \mathrm{x}(12^2) \mathrm{x}4$
  - States for pathing?
    120
  - States for eat-all-dots?
    $120 \mathrm{x}(2^{30})$



15

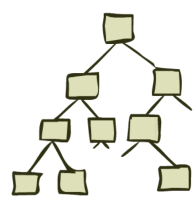
Quiz: Safe Passage



- Problem: eat all dots while keeping the ghosts perma-scared
- What does the state space have to specify?
  - (agent position, dot booleans, power pellet booleans, remaining scared time)

16

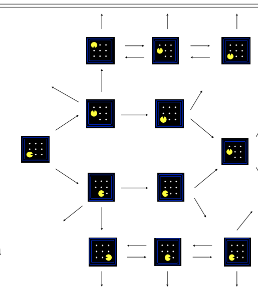
State Space Graphs and Search Trees



17

State Space Graphs

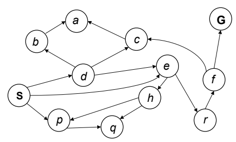
- State space graph: A mathematical representation of a search problem
  - Nodes are (abstracted) world configurations
  - Arcs represent successors (action results)
  - The goal test is a set of goal nodes (maybe only one)
- In a state space graph, **each state occurs only once!**
- We can rarely build this full graph in memory (it's too big), but it's a useful idea



18

State Space Graphs

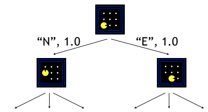
- State space graph: A mathematical representation of a search problem
  - Nodes are (abstracted) world configurations
  - Arcs represent successors (action results)
  - The goal test is a set of goal nodes (maybe only one)
- In a search graph, **each state occurs only once**!
- We can rarely build this full graph in memory (it's too big), but it's a useful idea



Tiny search graph for a tiny search problem

19

Search Trees



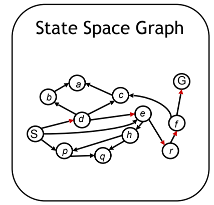
This is now / start

Possible futures

- A search tree:
  - A "what if" tree of plans and their outcomes
  - The start state is the root node
  - Children correspond to successors
  - Nodes show states, but correspond to PLANS that achieve those states
  - For most problems, we can never actually build the whole tree

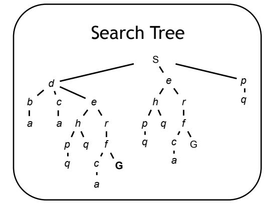
20

State Space Graphs vs. Search Trees



Each NODE in in the search tree is an entire PATH in the state space graph.

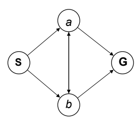
We construct both on demand - and we construct as little as possible.



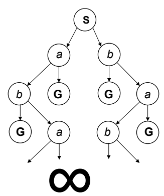
21

Quiz: State Space Graphs vs. Search Trees

Consider this 4-state graph:



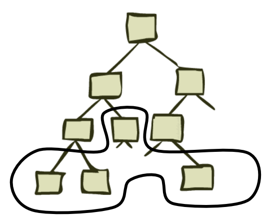
How big is its search tree (from S)?



Important: Lots of repeated structure in the search tree!

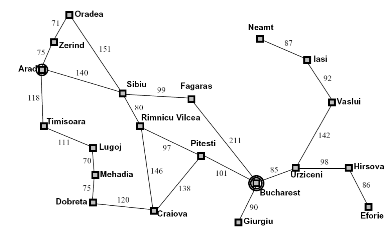
22

Tree Search



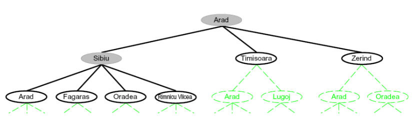
23

Search Example: Romania



24

Searching with a Search Tree



- Search:
  - Expand out potential plans (tree nodes)
  - Maintain a fringe of partial plans under consideration
  - Try to expand as few tree nodes as possible

25

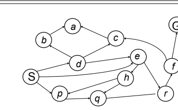
General Tree Search

```
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?

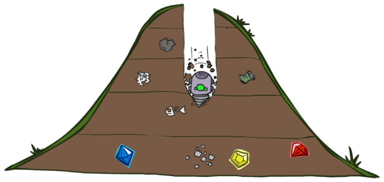
26

Example: Tree Search



27

Depth-First Search

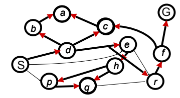


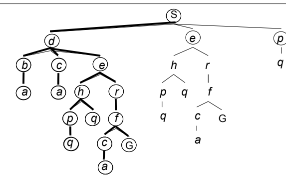
28

Depth-First Search

Strategy: expand a deepest node first

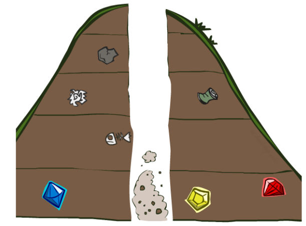
Implementation: Fringe is a LIFO stack





29

Search Algorithm Properties



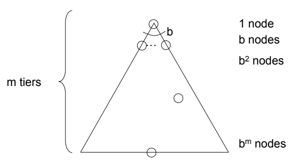
30

Search Algorithm Properties

- Complete: Guaranteed to find a solution if one exists?
- Optimal: Guaranteed to find the least cost path?
- Time complexity?
- Space complexity?

- Cartoon of search tree:
  - b is the branching factor
  - m is the maximum depth
  - solutions at various depths

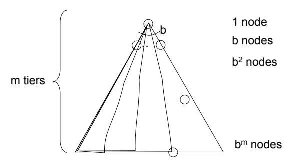
- Number of nodes in entire tree?
  - $1 + b + b^2 + \ldots. b^m = O(b^m)$



31

Depth-First Search (DFS) Properties

- What nodes DFS expand?
  - Some left prefix of the tree.
  - Could process the whole tree!
  - If m is finite, takes time $O(b^m)$
- How much space does the fringe take?
  - Only has siblings on path to root, so O(bm)
- Is it complete?
  - m could be infinite, so only if we prevent cycles (more later)
- Is it optimal?
  - No, it finds the "leftmost" solution, regardless of depth or cost



32