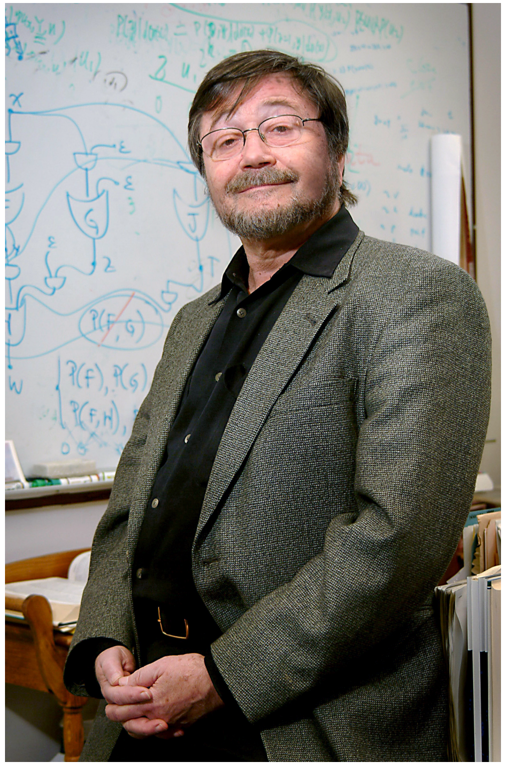
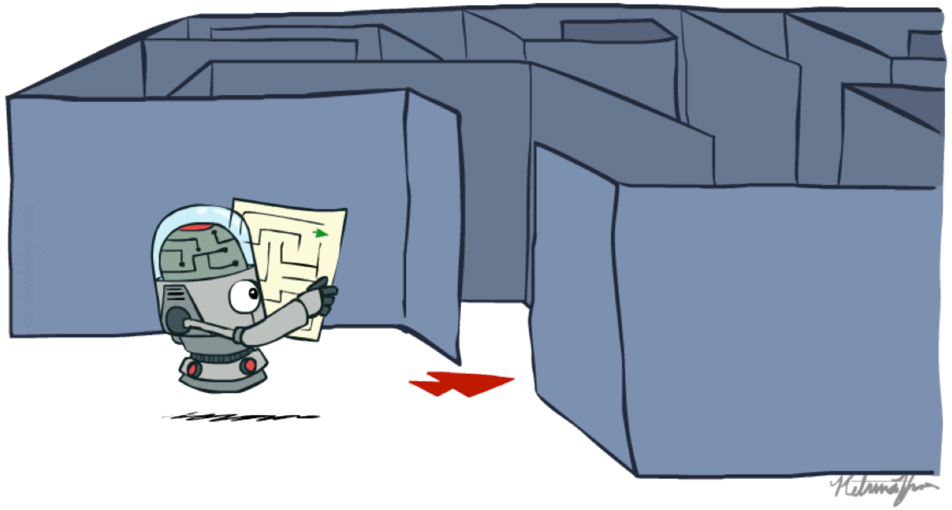


Artificial Intelligence Uninformed Search



CS 444 – Spring 2021

Dr. Kevin Molloy
Department of Computer Science
James Madison University

JAMES MADISON UNIVERSITY

Much of this lecture is taken from
Dan Klein and Pieter Abbeel AI class

Announcements

- HW 0 is due tonight (from last lecture and Chp 2 in Russell/Norvig).
- PA 0 is due tomorrow Friday, Jan 22
- PA 1 is released. You should start on the project now.
- No quiz this week (want to get back some HW to you first).

Learning Objectives for Today

- More on Reflex and other agents
- Define a search problem
- Uninformed Search Methods:
  - Depth-first search
  - Breadth-first search
  - Iterative Deeping Search
  - Uniform-Cost search

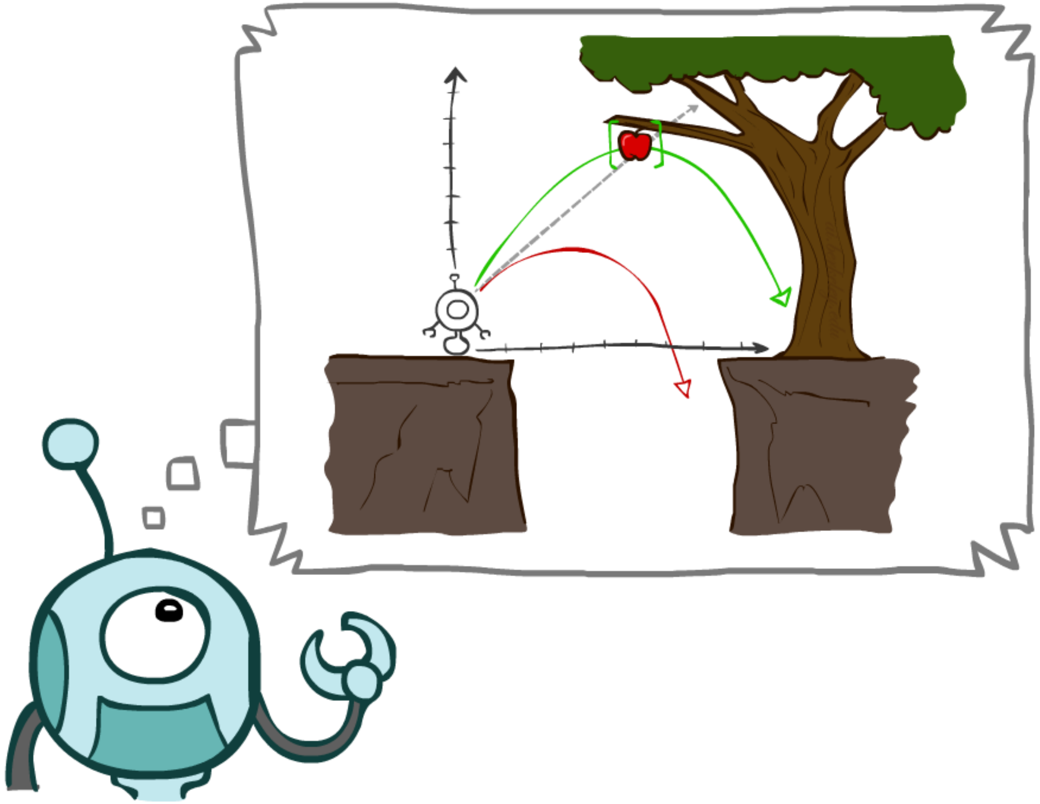


Figure from Berkley AI

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
- Summary: They consider how the world IS right now.

Question: Can a reflex agent be rational?

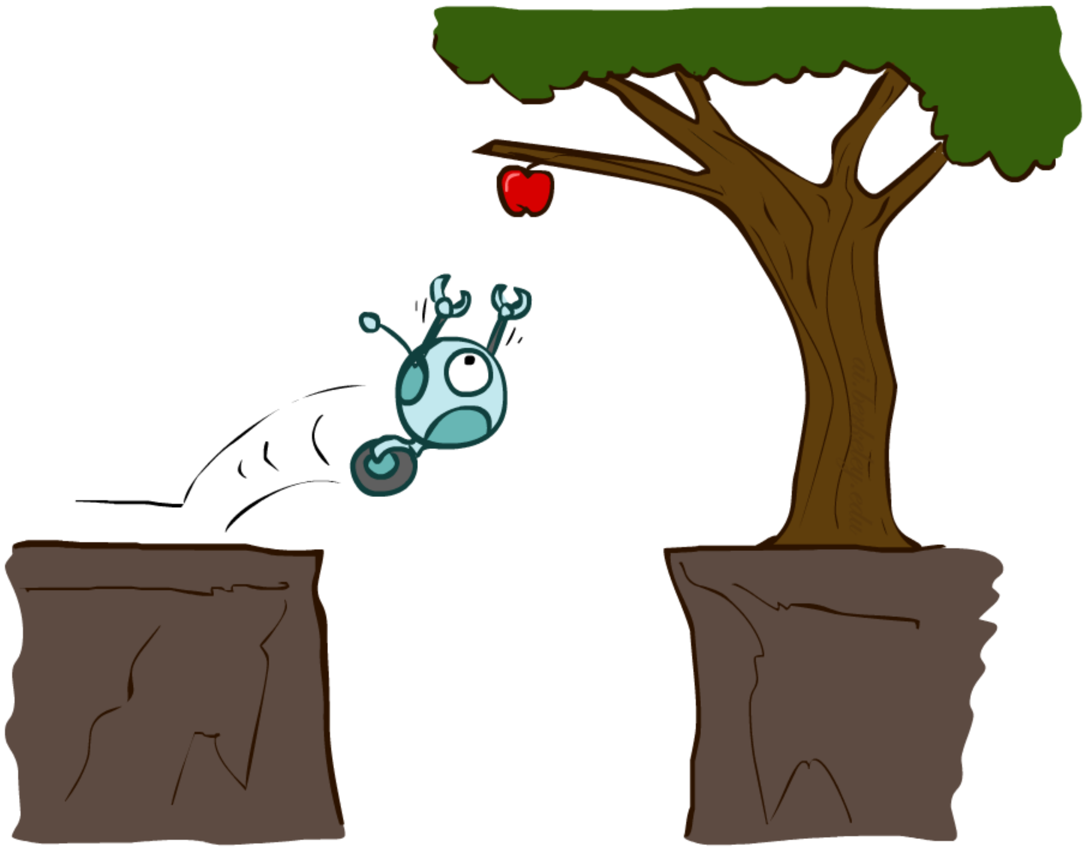
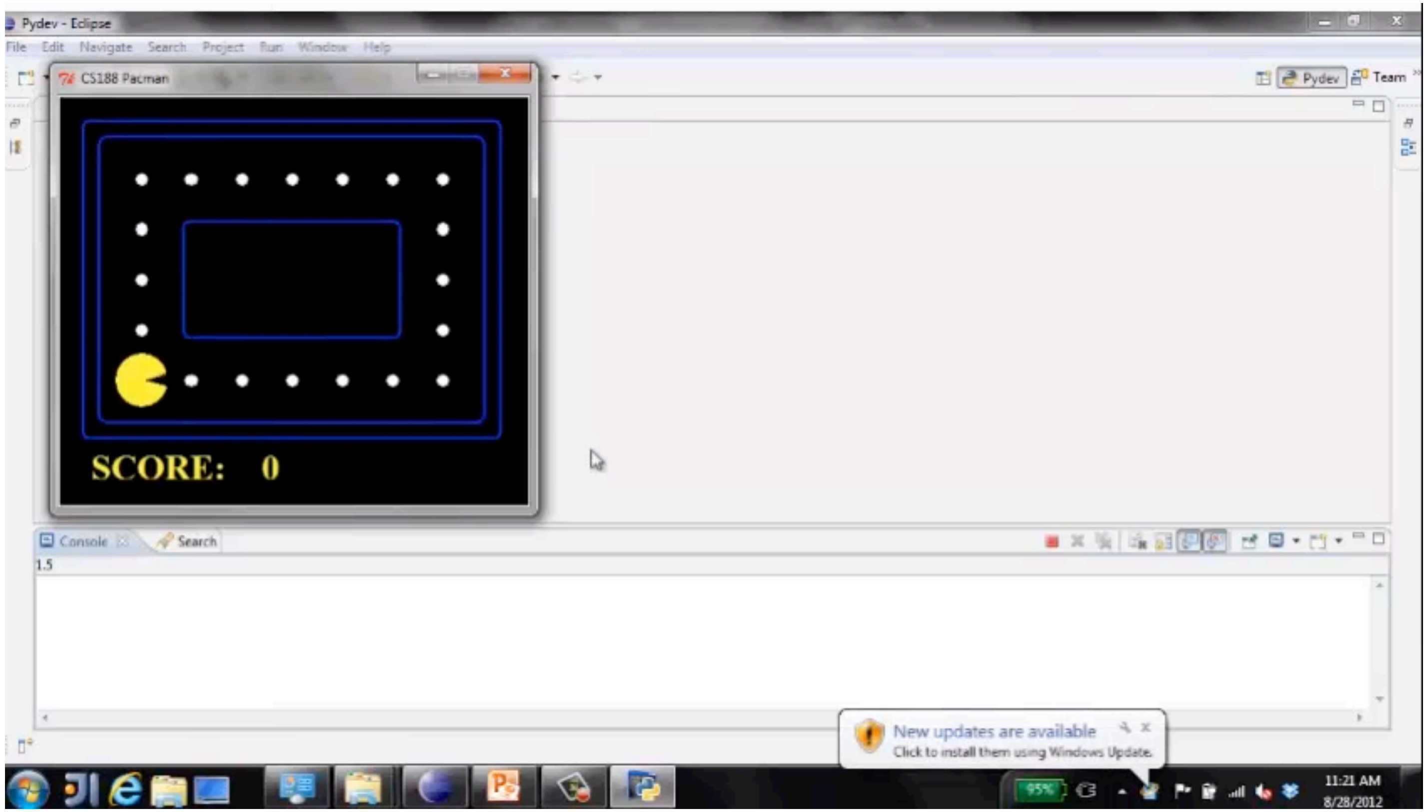


Figure from Berkley AI

Video of Reflex Agent that is Optimal

Question: Can a reflex agent be rational?

Strategy: Move towards the nearest food pellet



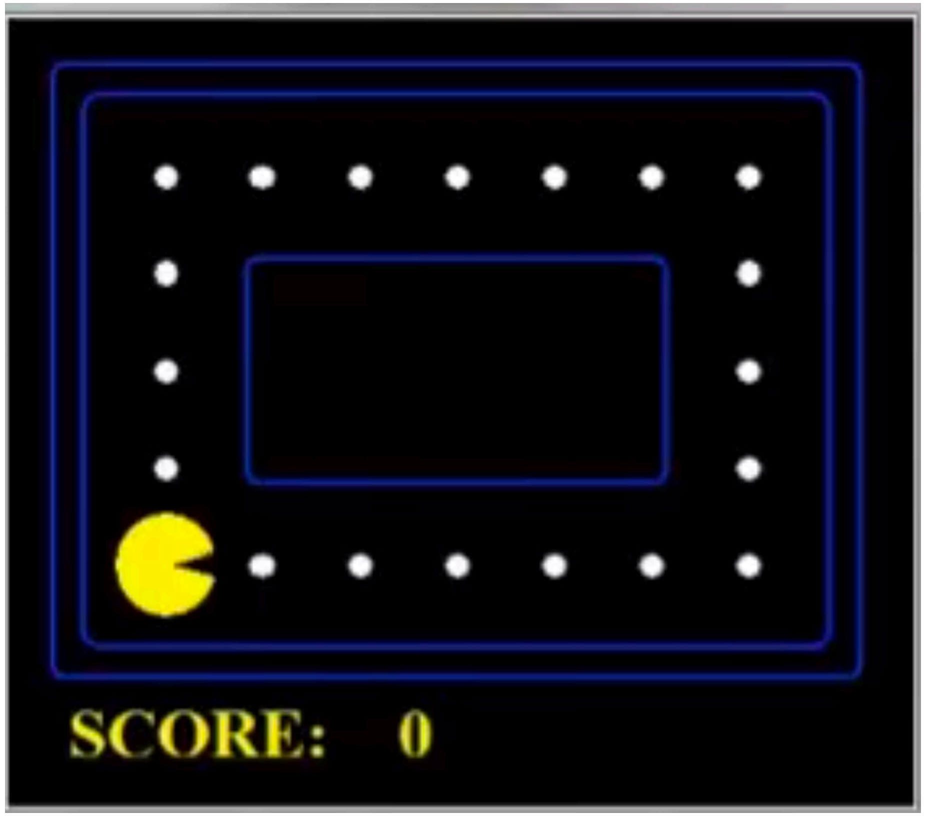


Figure from Berkley AI

Reflex Agent in a Different Environment

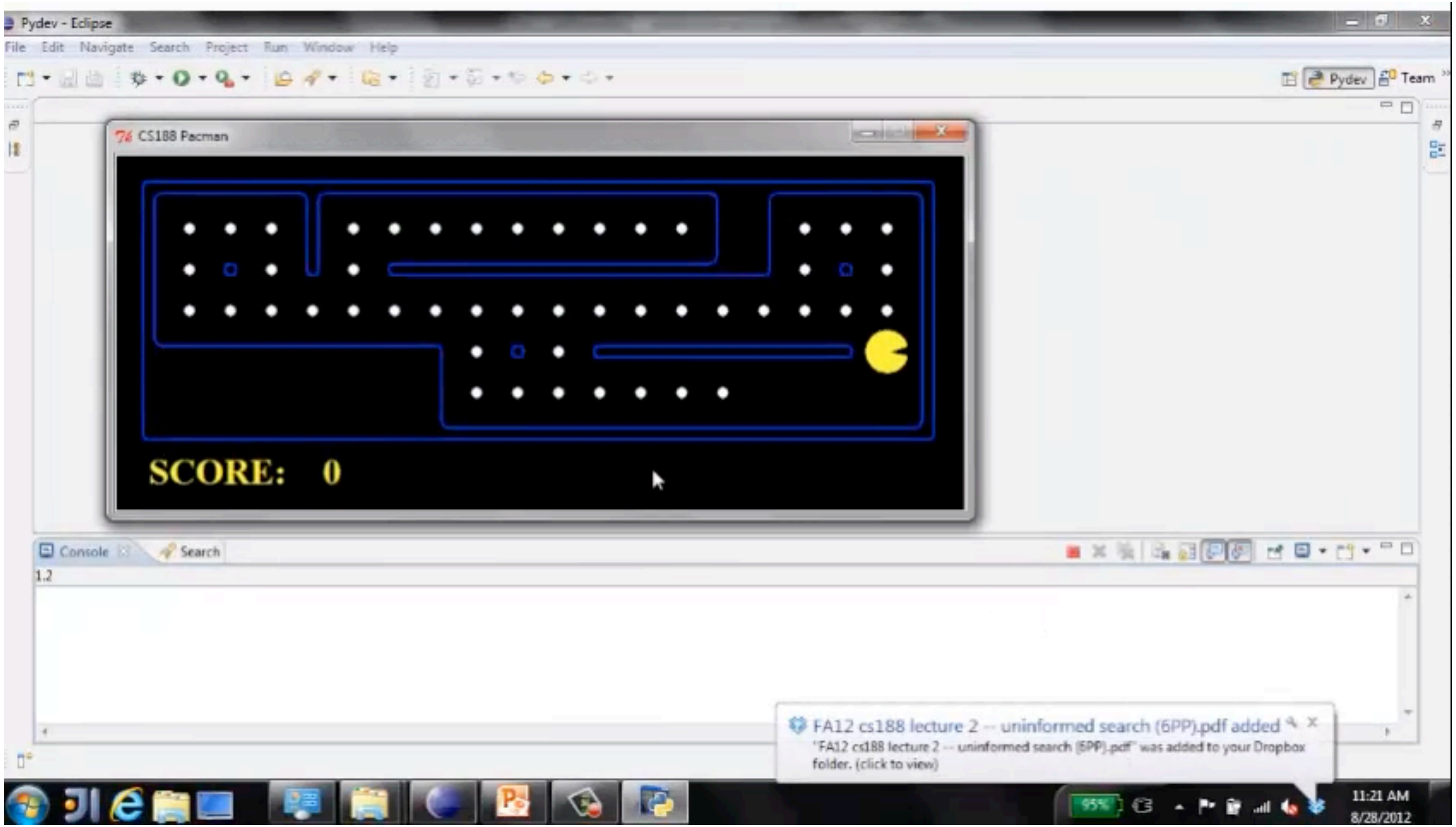
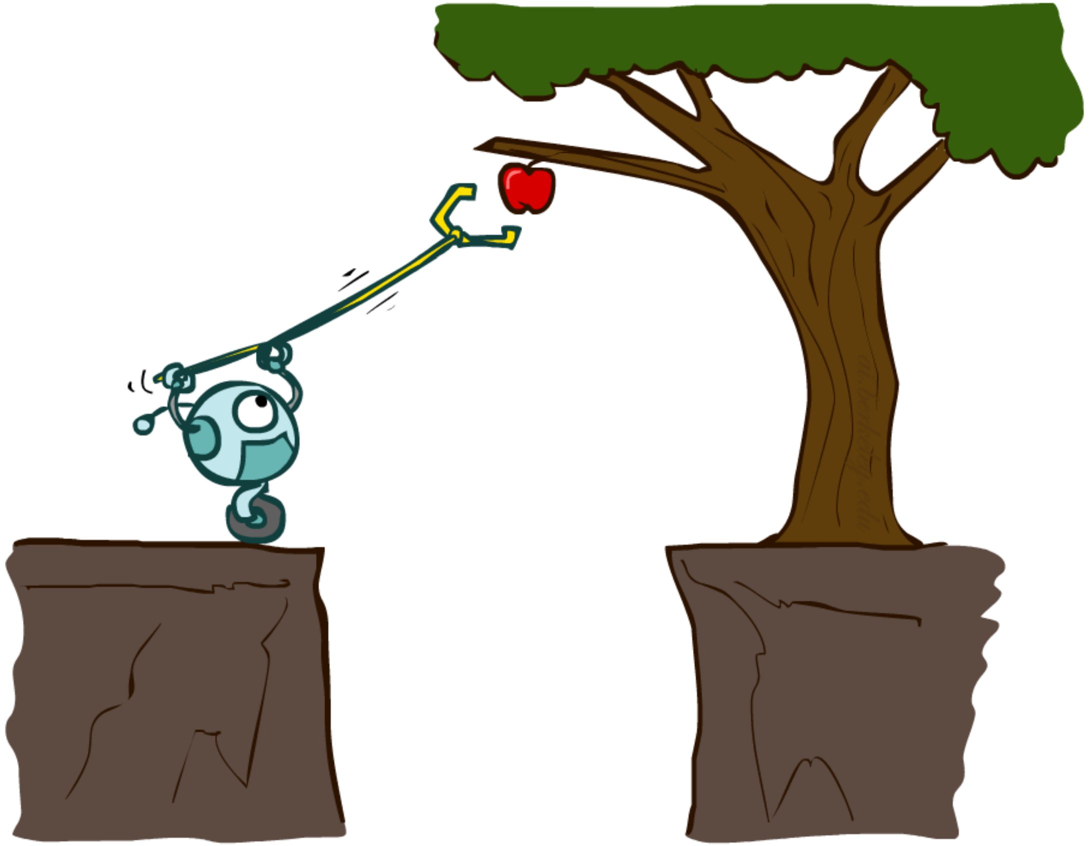


Figure from Berkley AI

Planning Agents

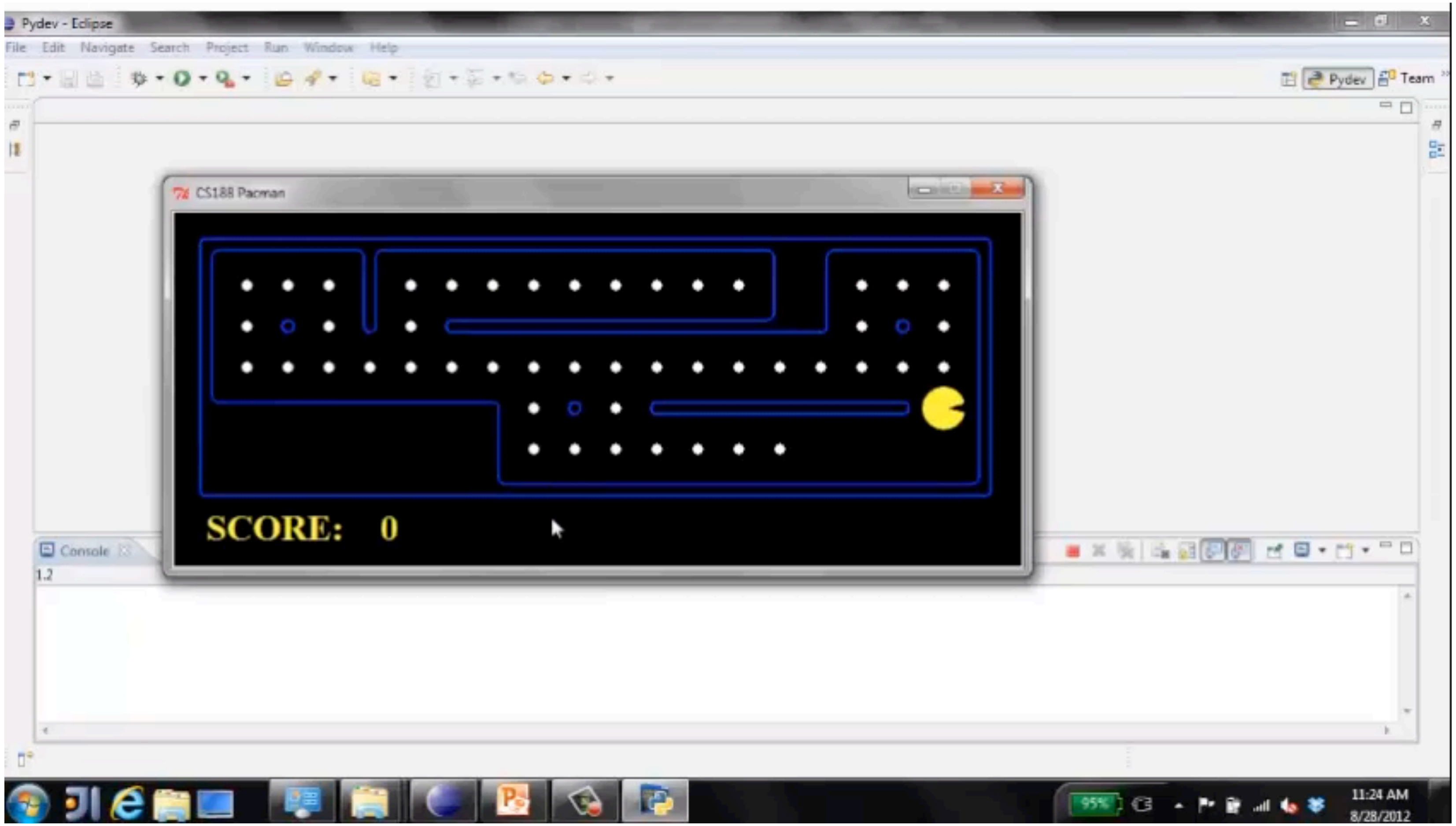
- Planning agents:
  - Ask "what if"
  - Decisions based on (hypothesized) consequences of actions
  - Must have a model of how the world evolves in response to actions
  - Must formulate a goal (test)
  - Consider how the world WOULD BE
- Optimal vs. complete planning
- Planning vs replanning



Much of this slide from Berkley AI

Replanning in Pacman

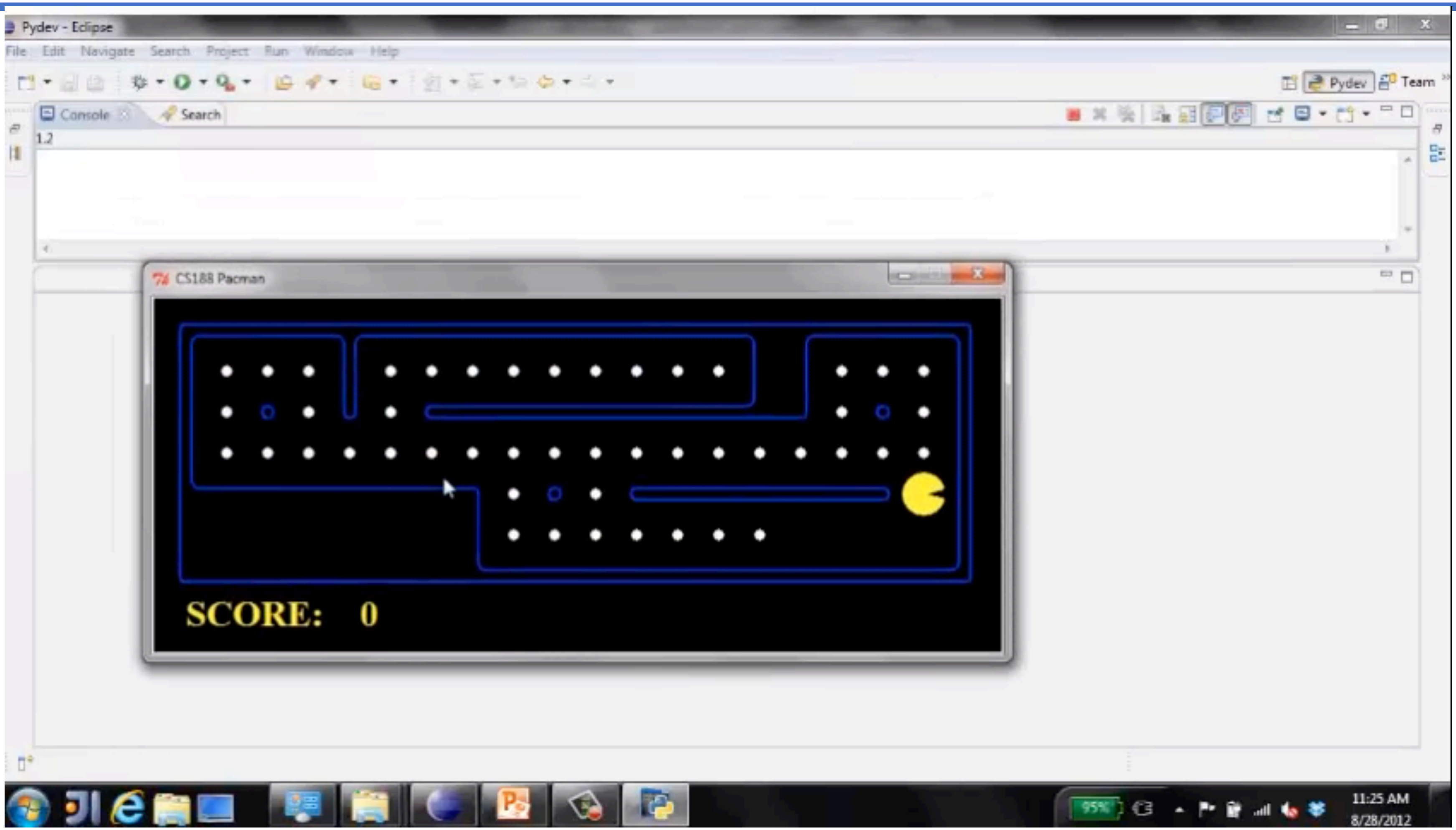
After eating each dot, plan again (replan) on how to get to the nearest dot.



Question: Does this agent act rationally/optimally?

Complete Planning

Construct a global plan to get all the dots. Evaluate all plans and pick the optimal plan.

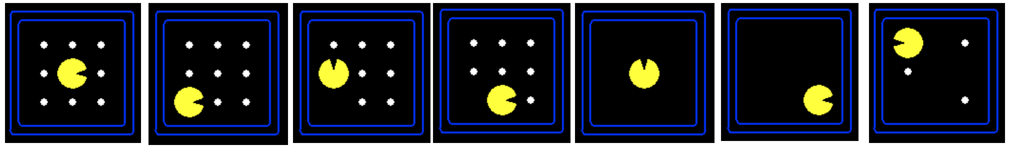


Question: Is this really an option generally speaking?

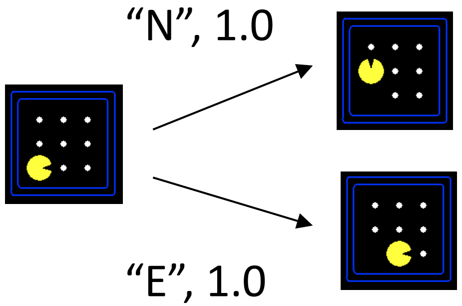
Search Problems

A **search problem** consists of:

- A state space



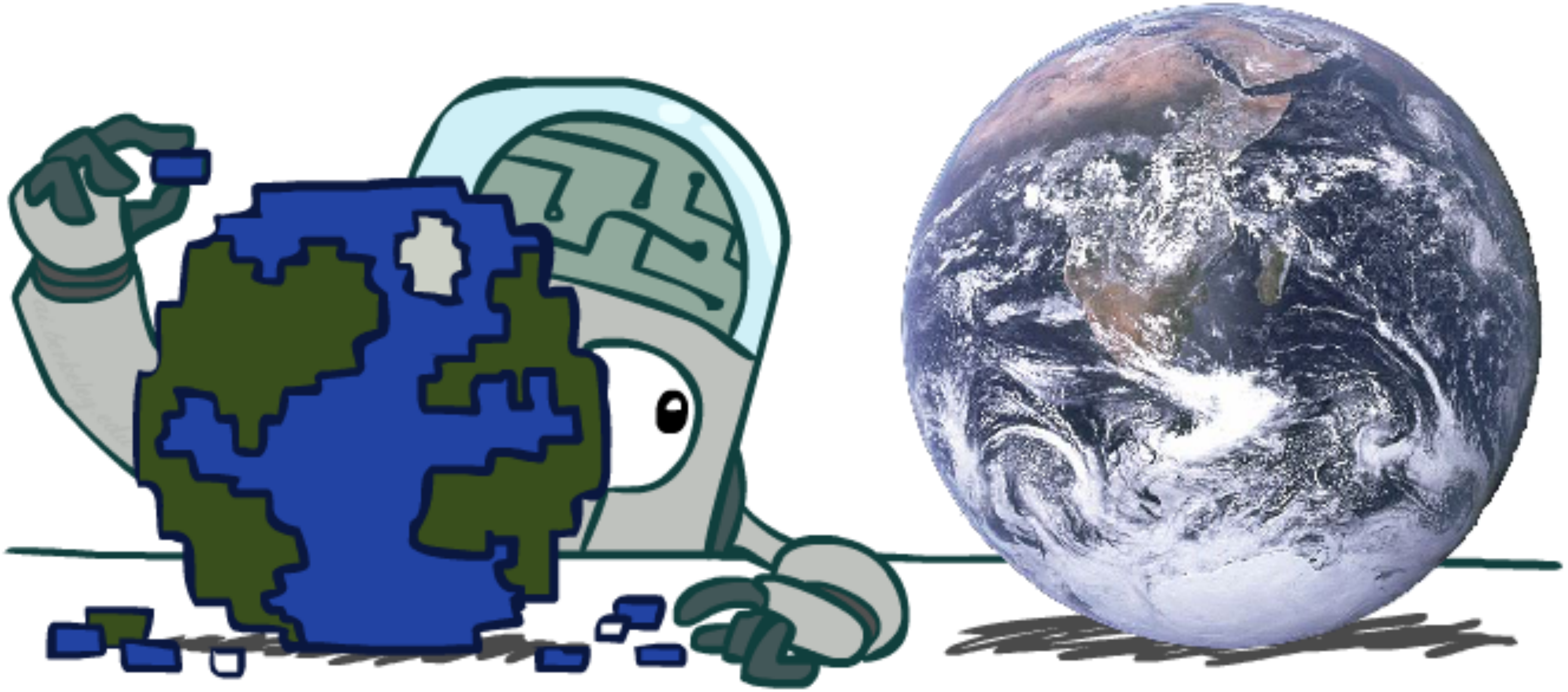
- A successor function (with actions, costs)



- 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.

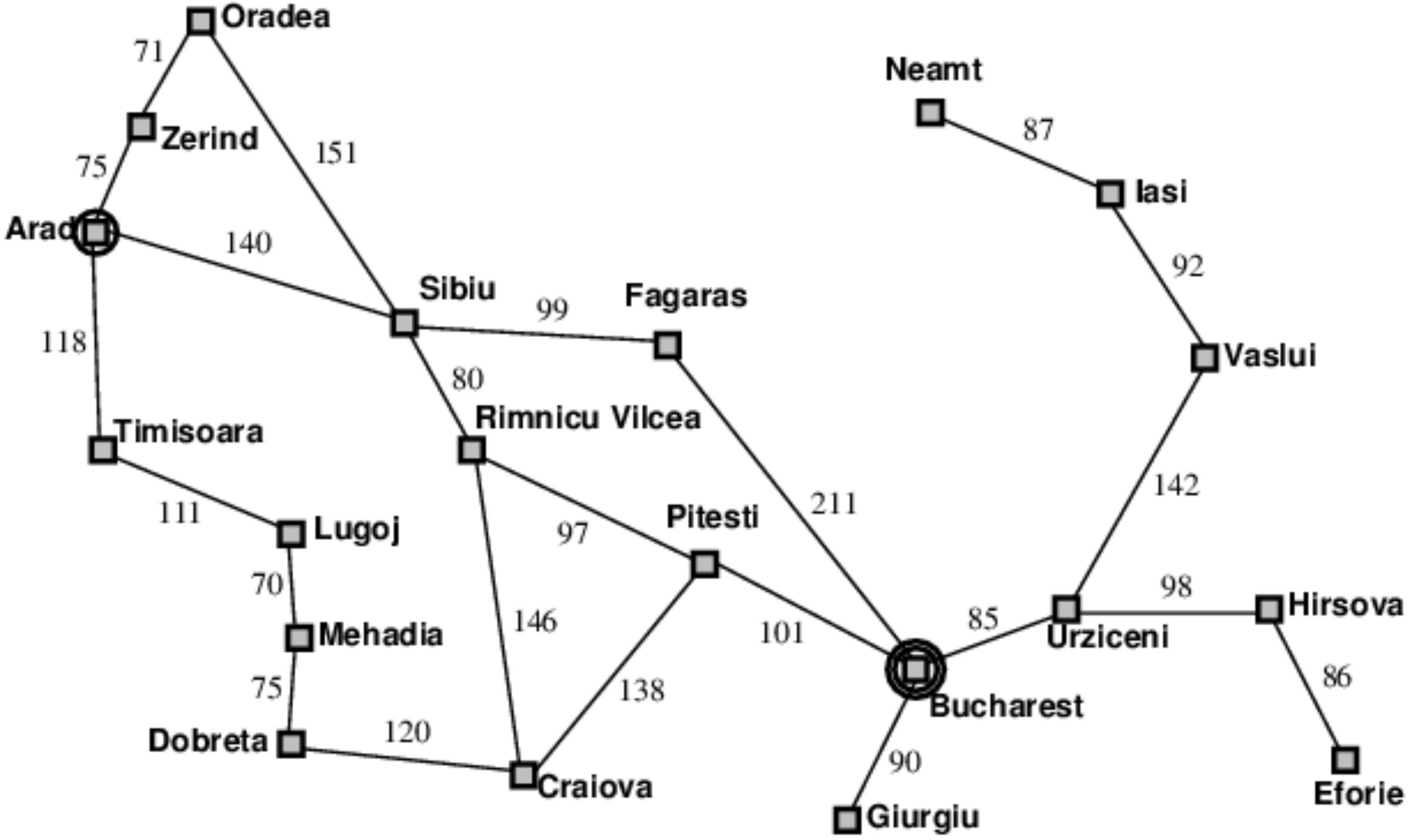
Search Problems are Models



Models are discrete approximations of the real world.

Much of this slide from Berkley AI

Example: Traveling in Romania



State space:

- Cities

Successor function:

- Roads connecting adjacent city with cost = distance

Start state:

- Arad

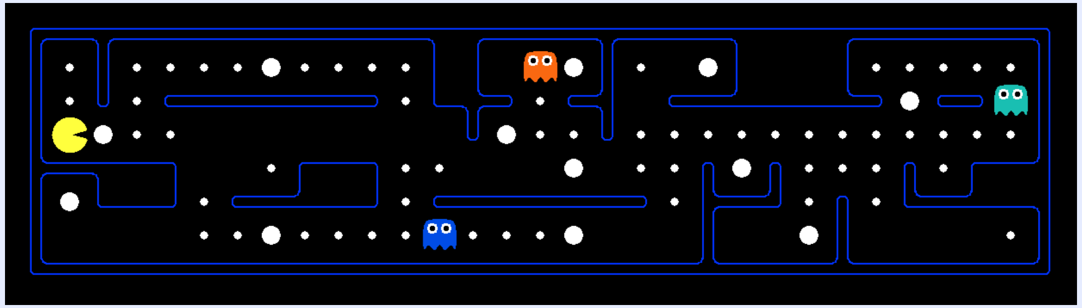
Goal test:

- Is state == Bucharest

Solution: How do we solve this problem?

What is 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

Goal Test: (x,y) == END

Problem: Eat All Dots

States: (x,y) , dots (booleans)

Actions: NSEW

Successor: Update location and dots

Goal Test: No dots (all false)

Much of this slide from Berkley AI

State Space Size?

World State:

Agent position: 120

Food: 30

Ghost positions: 12

Agent facing: NSEW

How many world states?

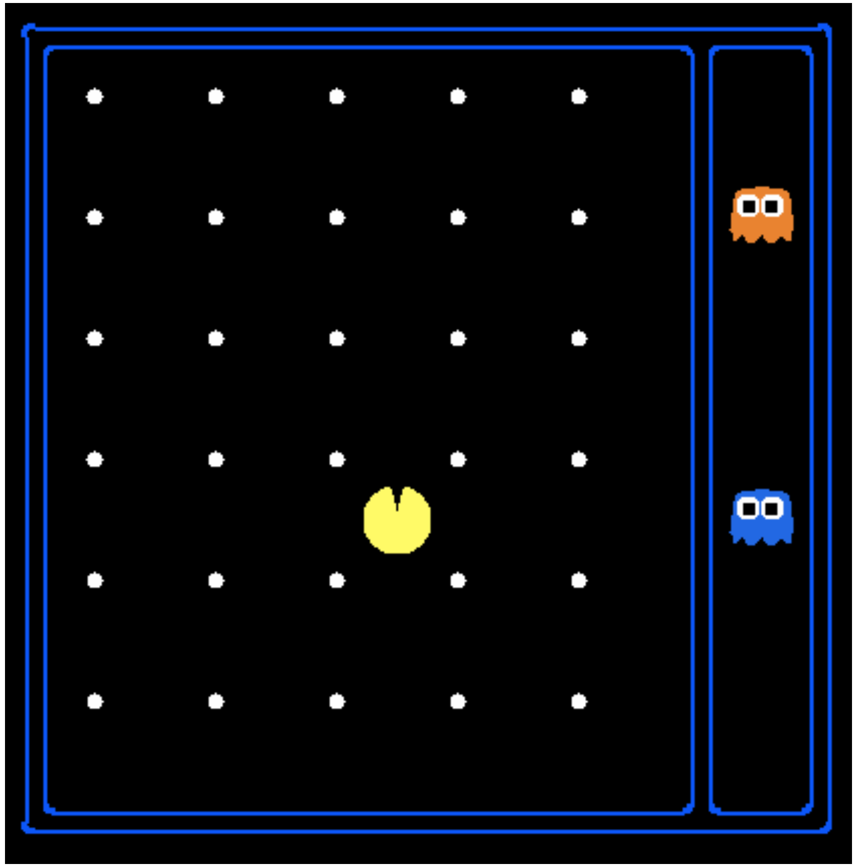
$120 \times 2^{30} \times 12^{2} \times 4$

States for pathing?

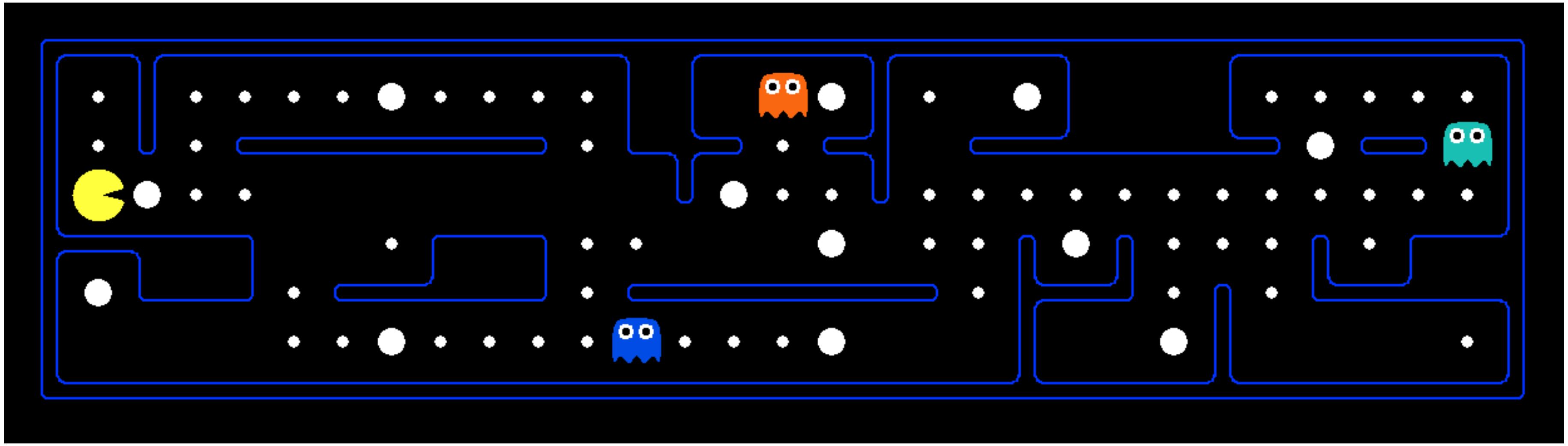
120

States for eat-all-the-dots?

$120 \times 2^{30} = 128{,}849{,}018{,}880$



Quiz: Safe Passage



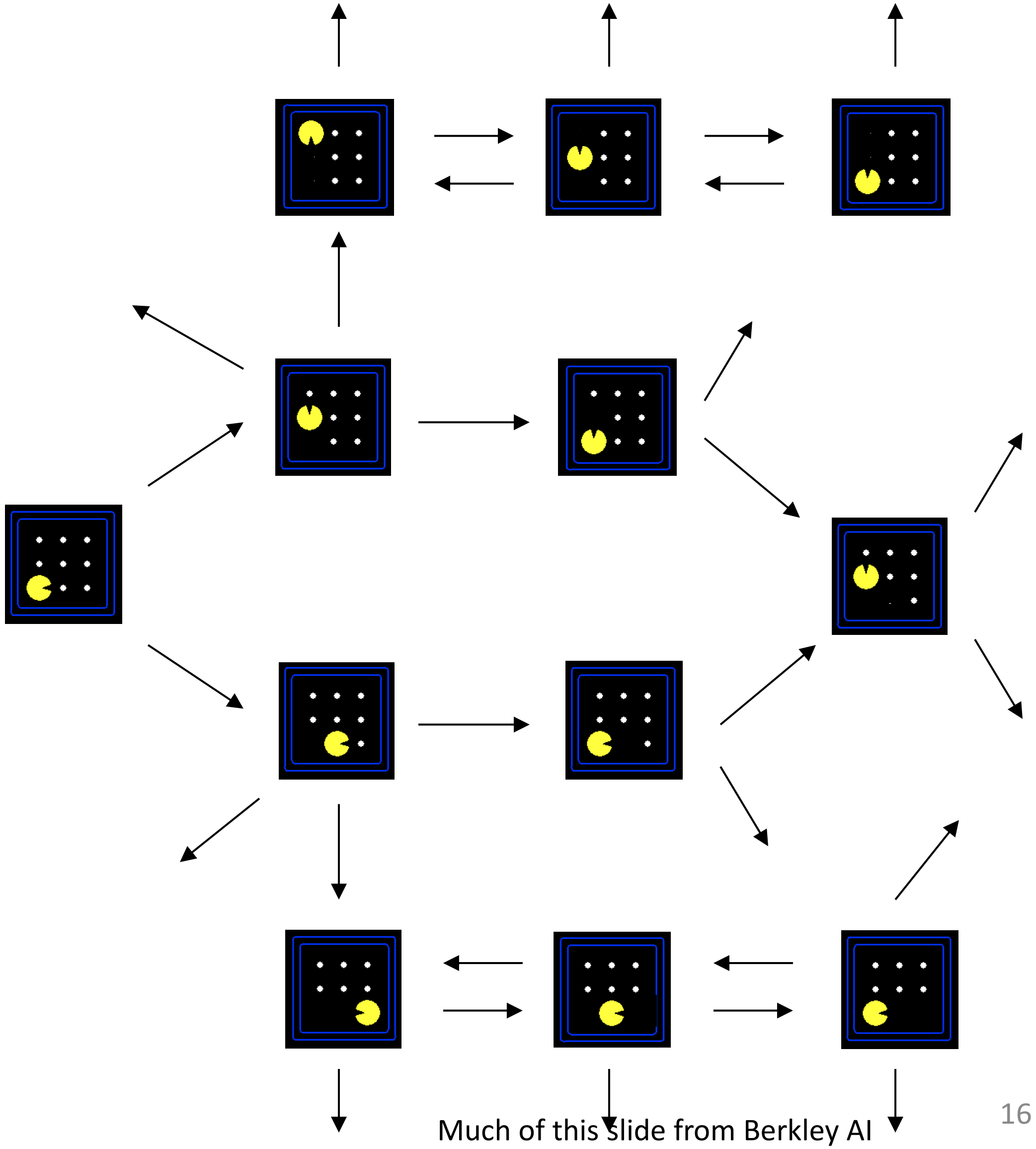
- Problem: eat all dots while keeping the ghosts perma-scared
- What does the state space have to specify?
  - agent position
  - dot booleans
  - power pellet booleans
  - remaining scared time

State Space Graph

- 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!

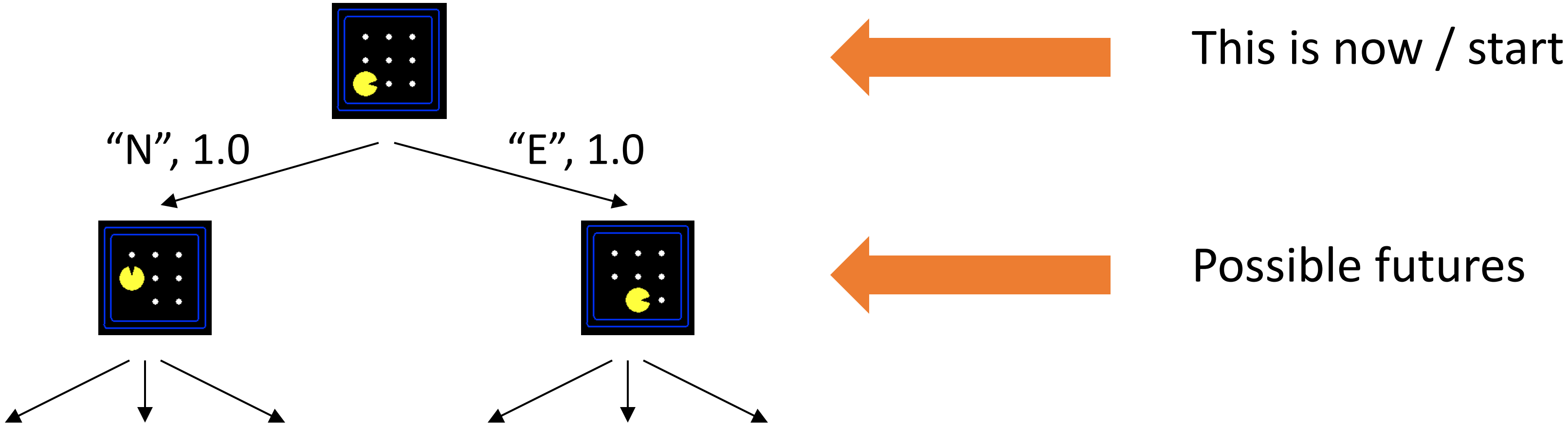
- We can rarely build this full graph in memory (it's too big), but it's a useful idea



Much of this slide from Berkley AI

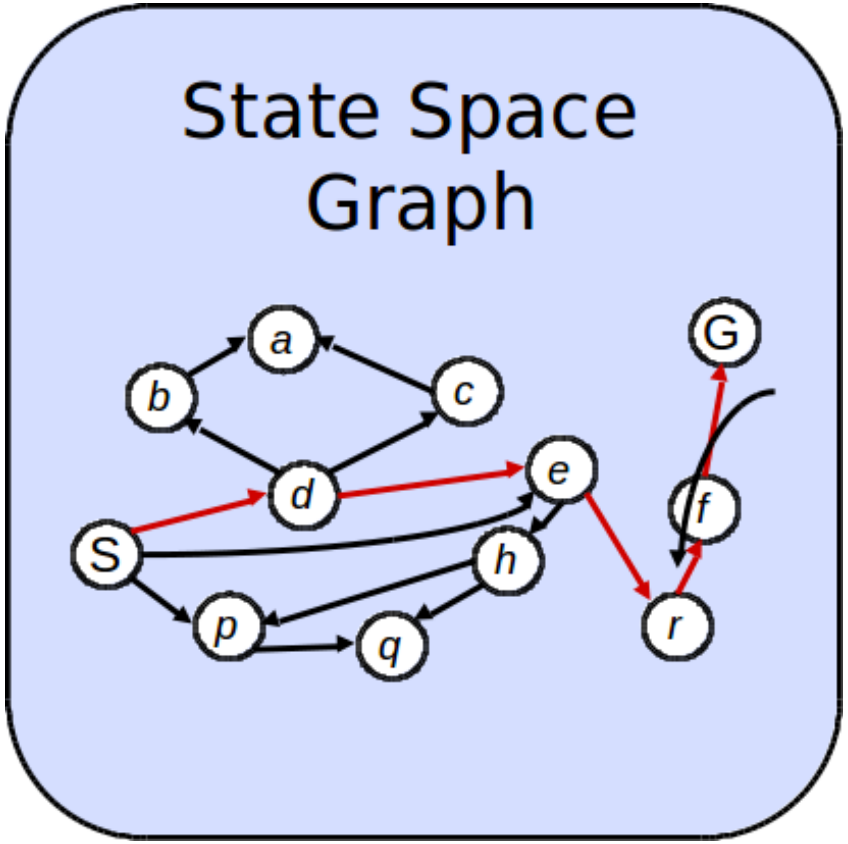
JAMES MADISON UNIVERSITY

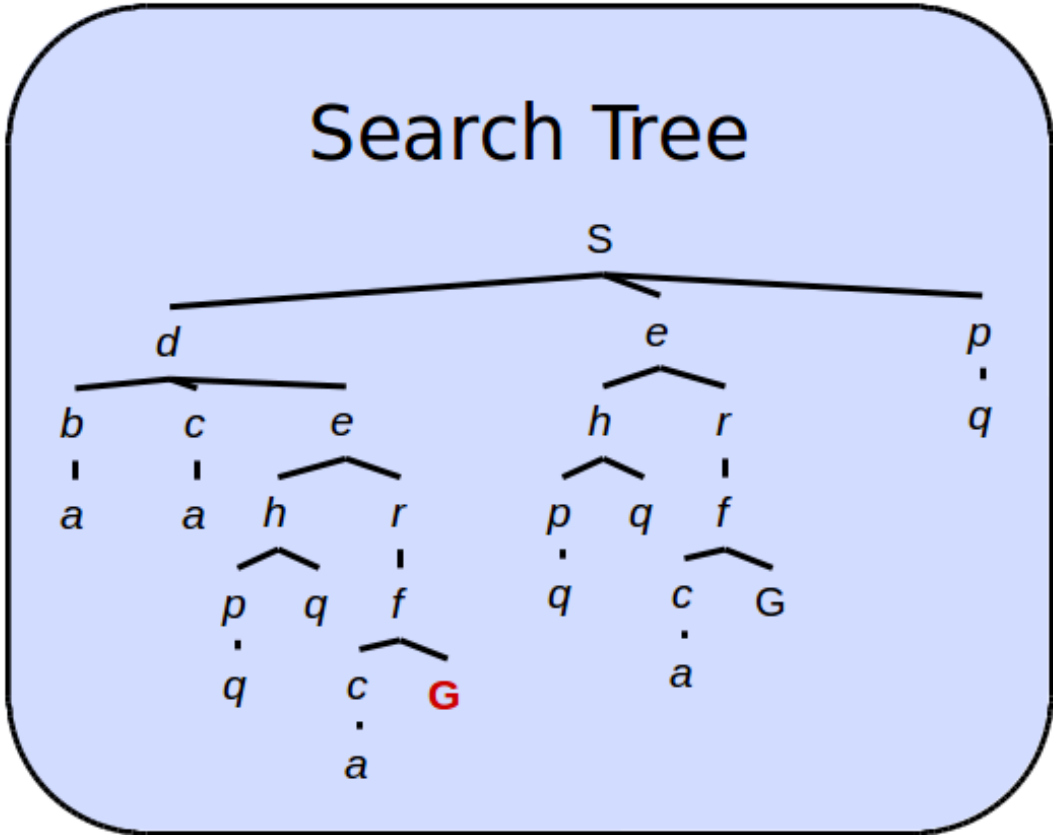
Search Trees



- 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

State Space Graph vs. Search Trees



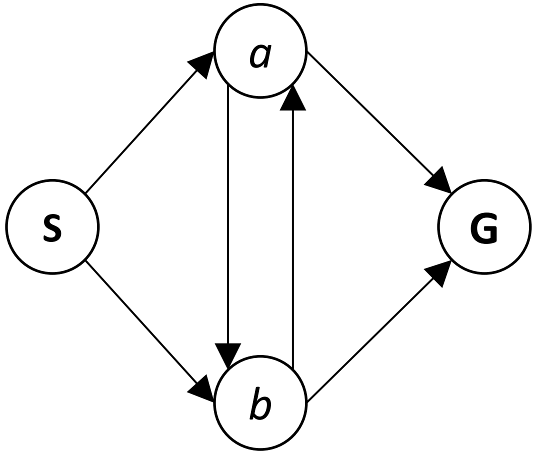


Each NODE in the search tree is an entire PATH in the state space graph.

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

Quiz: State Space Graphs vs. Search Tree

Consider this 4-state graph:



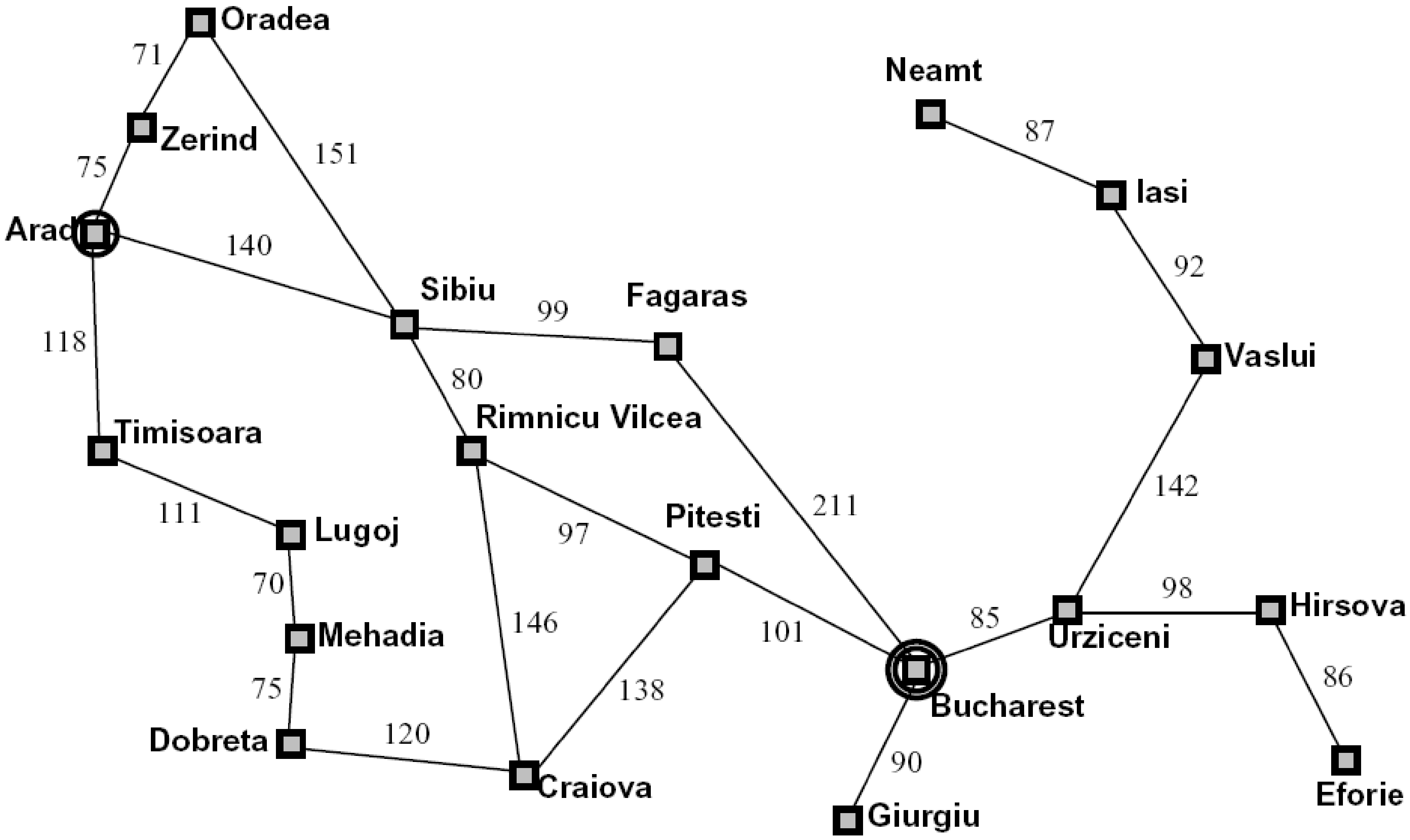
How big is its search tree (from S)?

∞

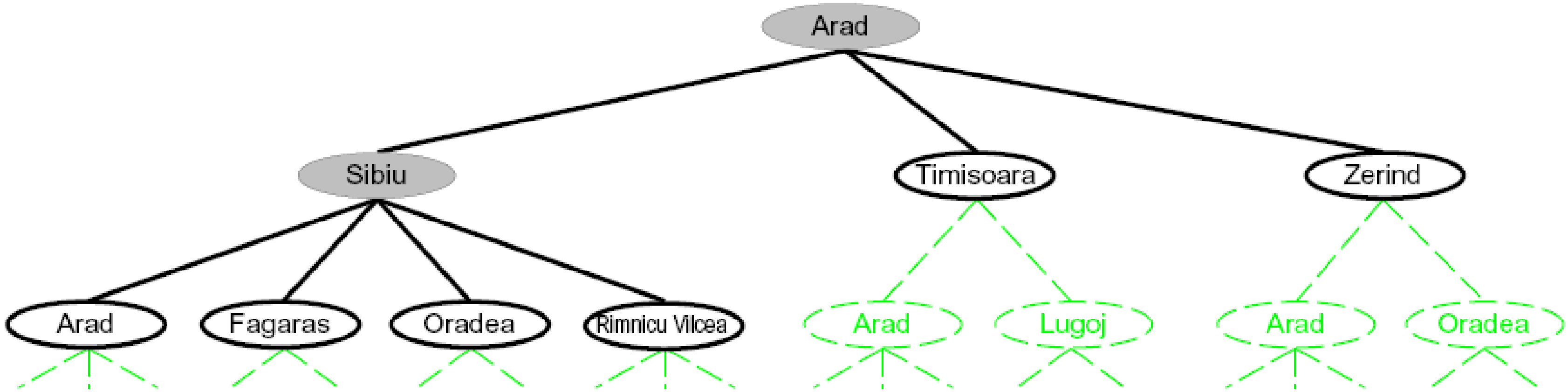
Important: Lots of repeated structure in the search tree!

Slide from Russell/Norvig

Tree Search: Example with Romania



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

JAMES MADISON UNIVERSITY

Slide from Russell/Norvig

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?

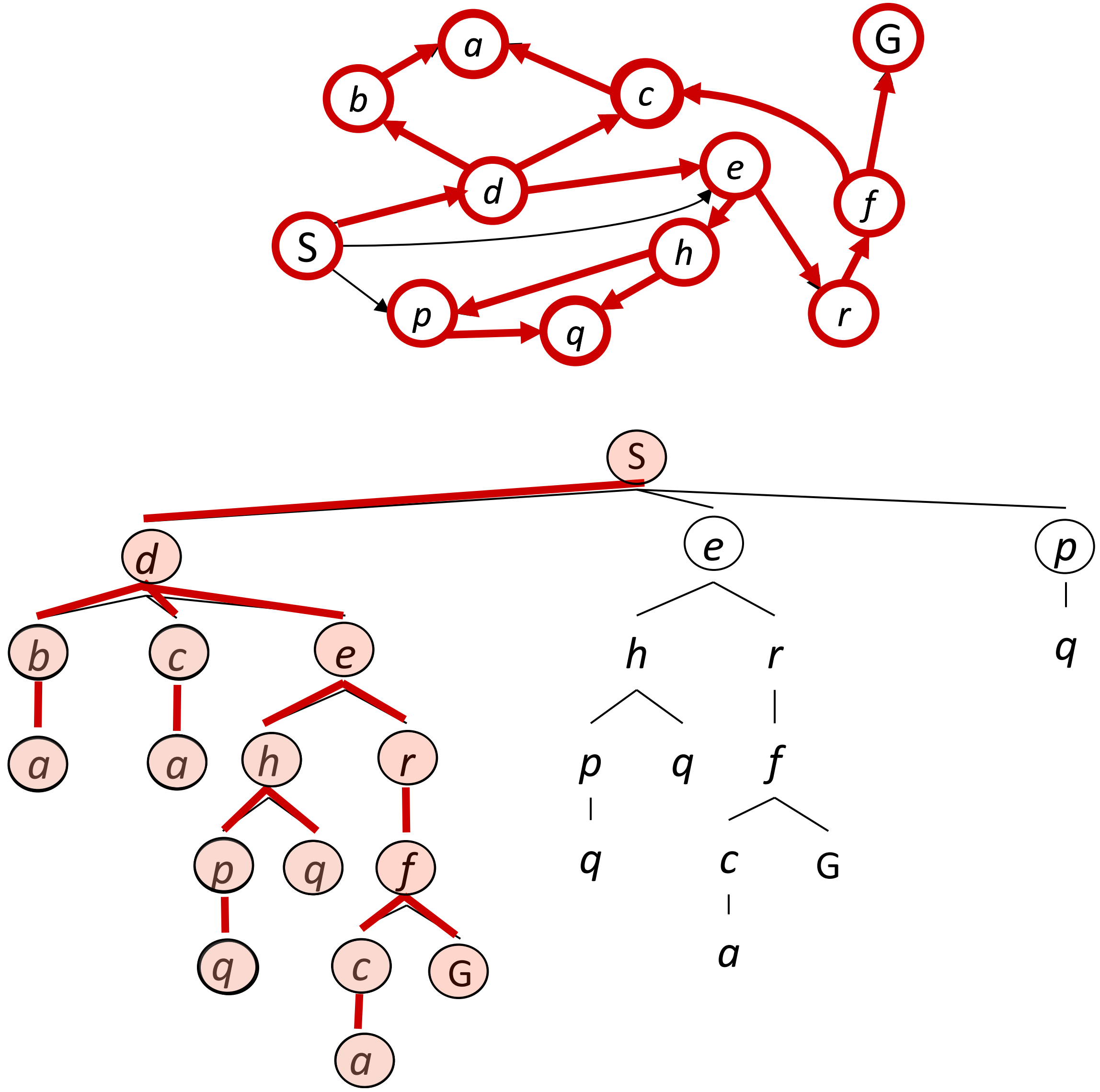
Depth-First Search (DFS)

Much of this slide from Berkley AI

DFS

Strategy: expand a deepest node first

Implementation: Fringe is a LIFO stack

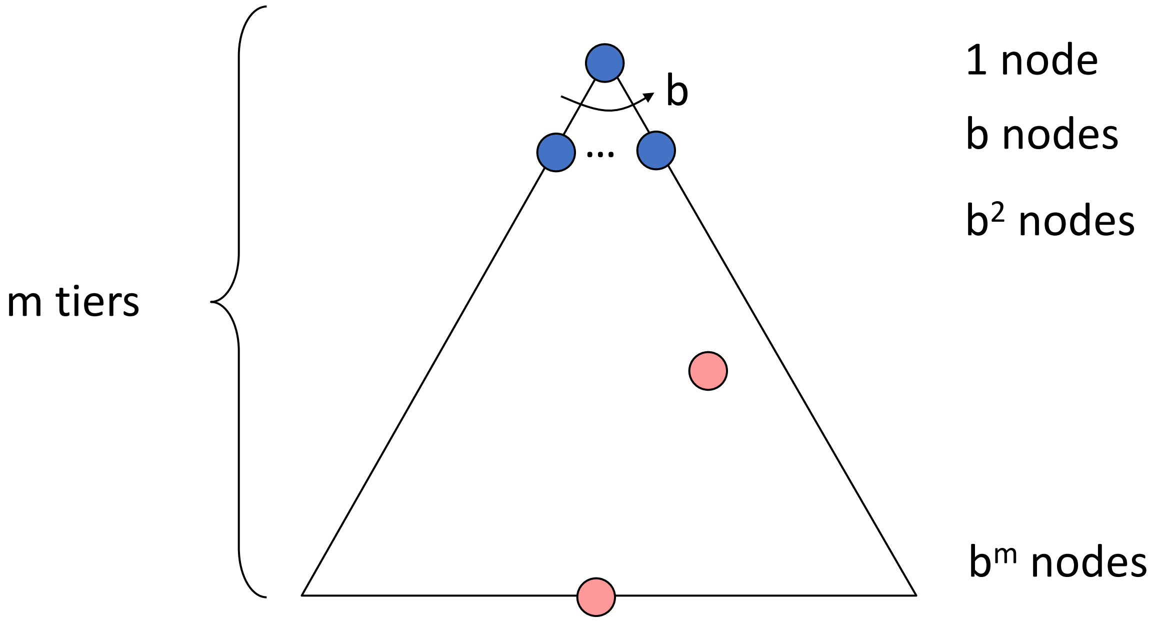


Search Algorithm Properties

- 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

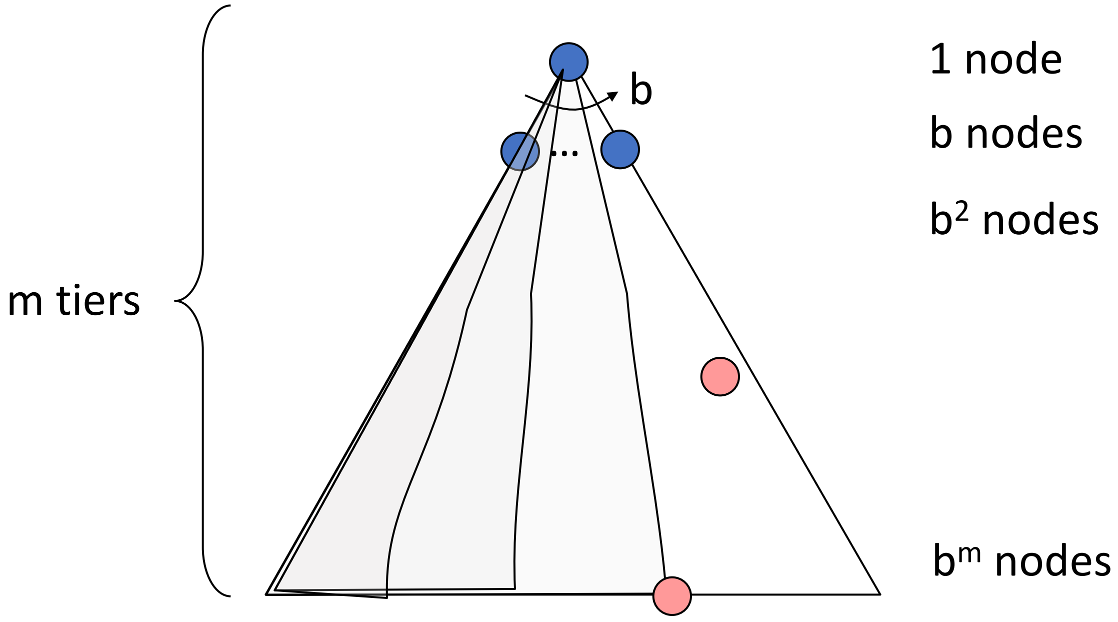
- Number of nodes in entire tree?
  - $1 + b + b^2 + \ldots.\ b^m = O(b^m)$



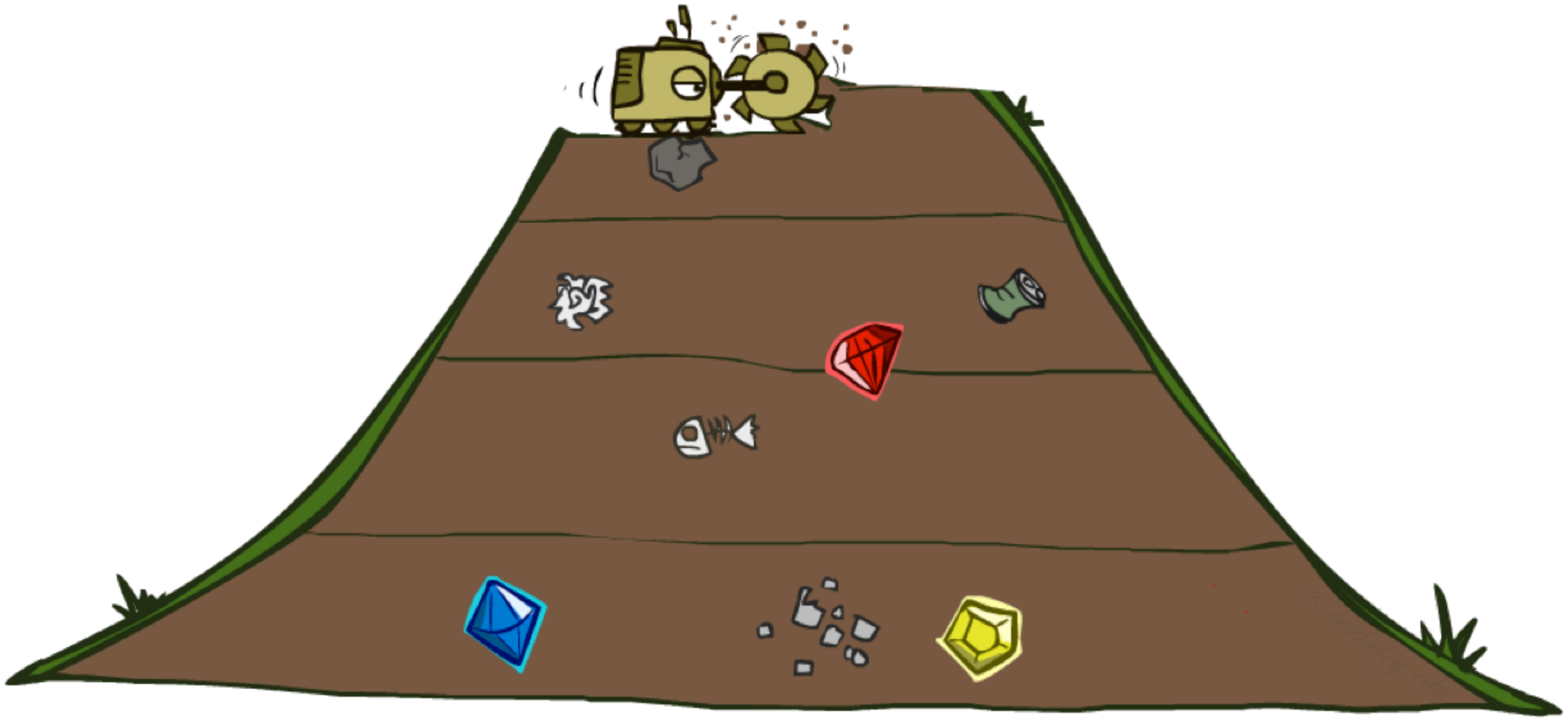
Much of this slide from Berkley AI and Russell/Norvig

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



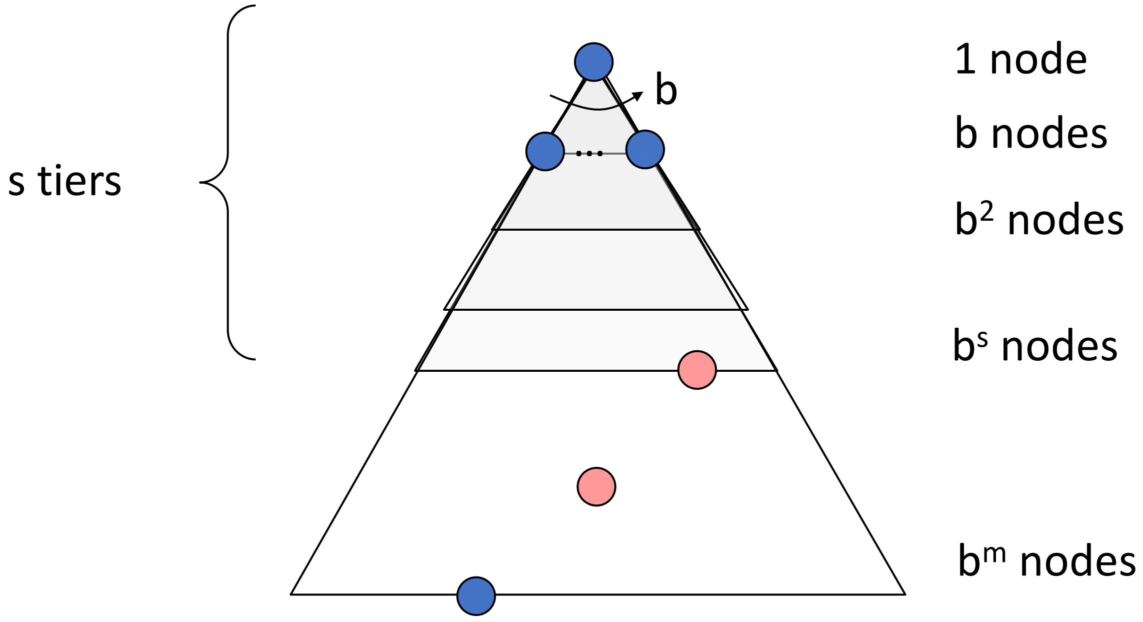
Breadth-First Properties



Much of this slide from Berkley AI

BFS Properties

- What nodes does BFS expand?
  - Processes all nodes above shallowest solution
  - Let depth of shallowest solution be s
  - Search takes time $O(b^s)$
- How much space does the fringe take?
  - Has roughly the last tier, so $O(b^s)$
- Is it complete?
  - s must be finite if a solution exists, so yes!
- Is it optimal?
  - Only if costs are all 1 (more on costs later)



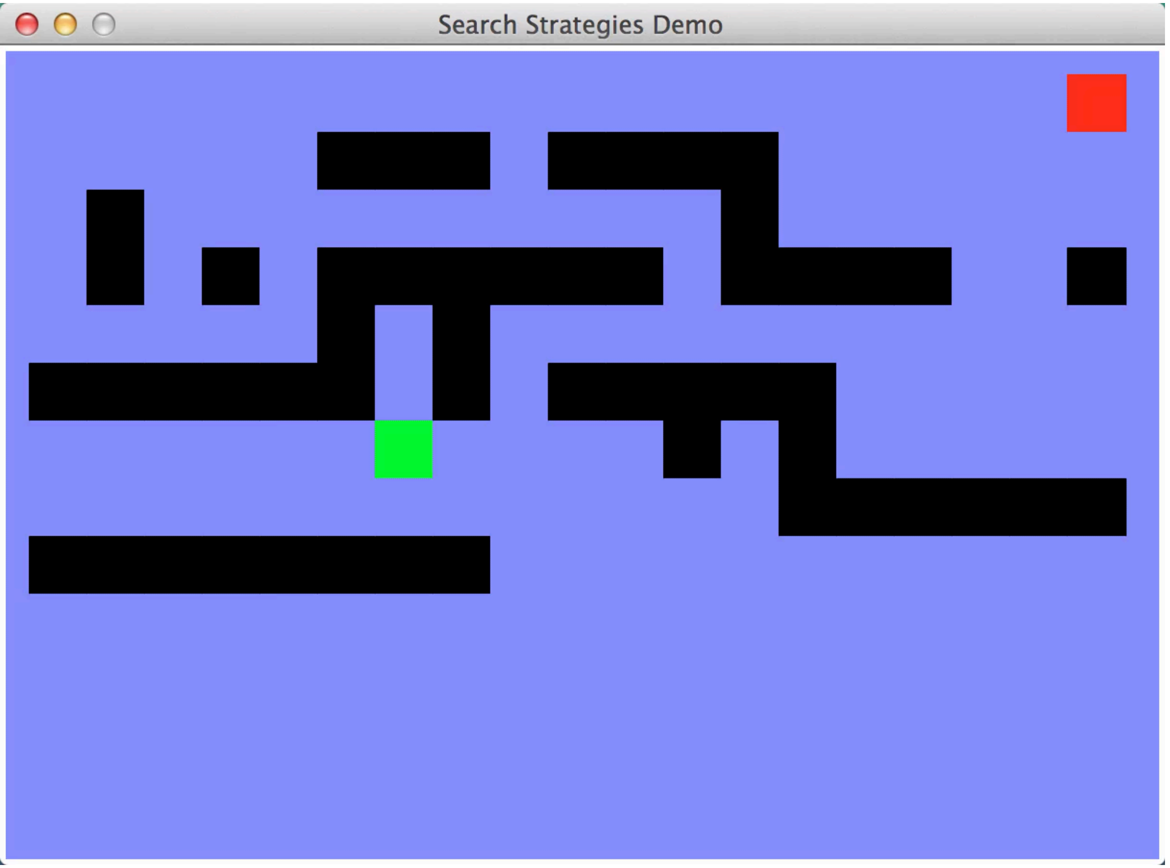
Much of this slide from Berkley AI and Russell/Norvig

Quiz: DFS vs BFS

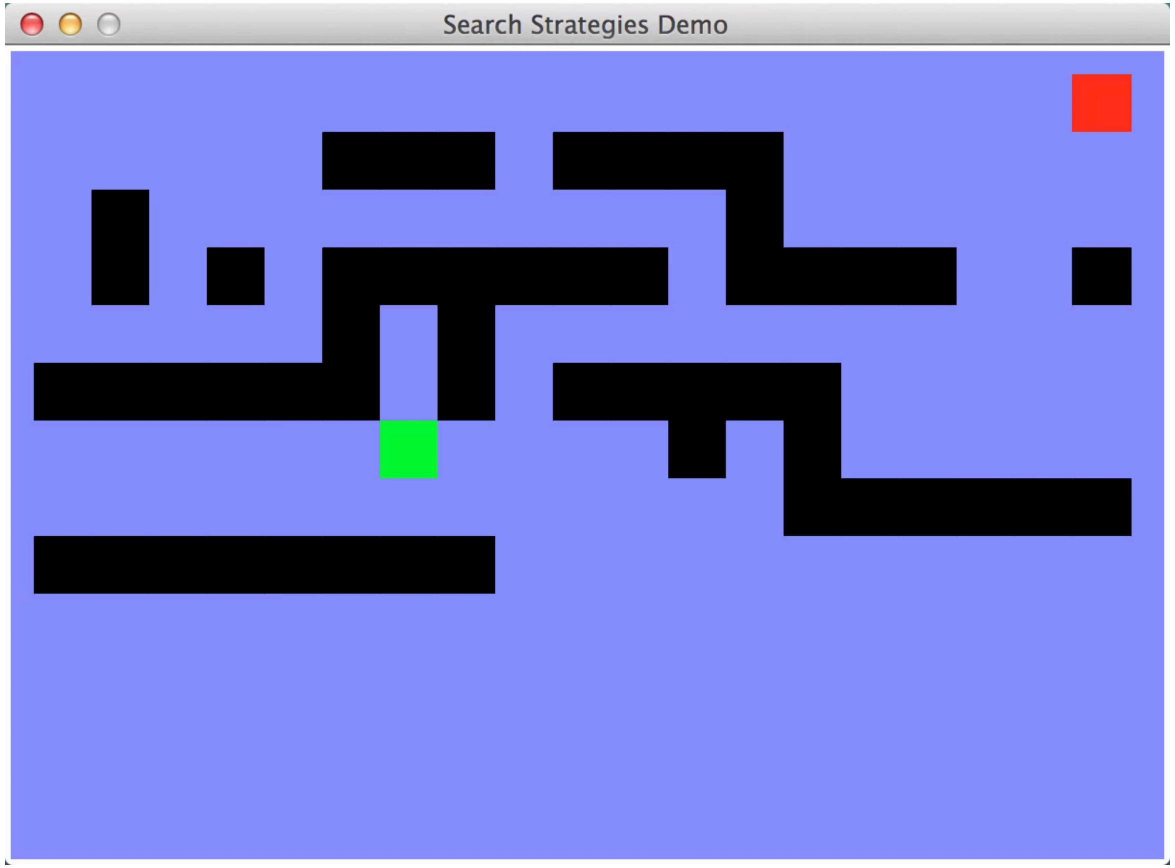
- When will BFS outperform DFS?

- When will DFS outperform BFS?

Visualize Search



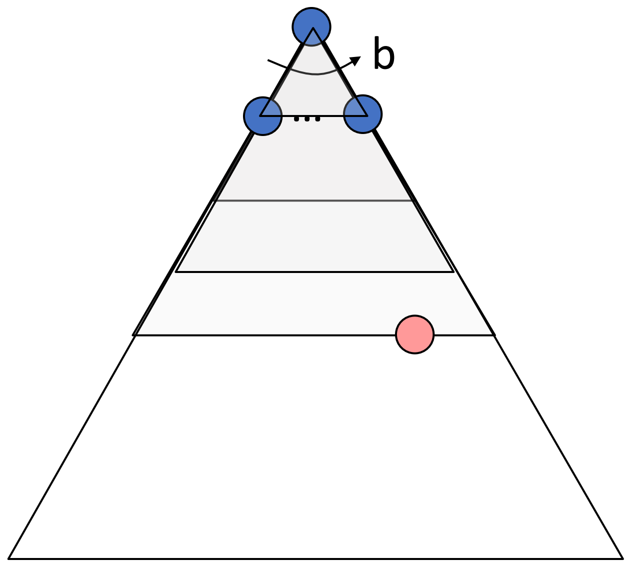
DFS



BFS

Iterative Deeping Search (IDS)

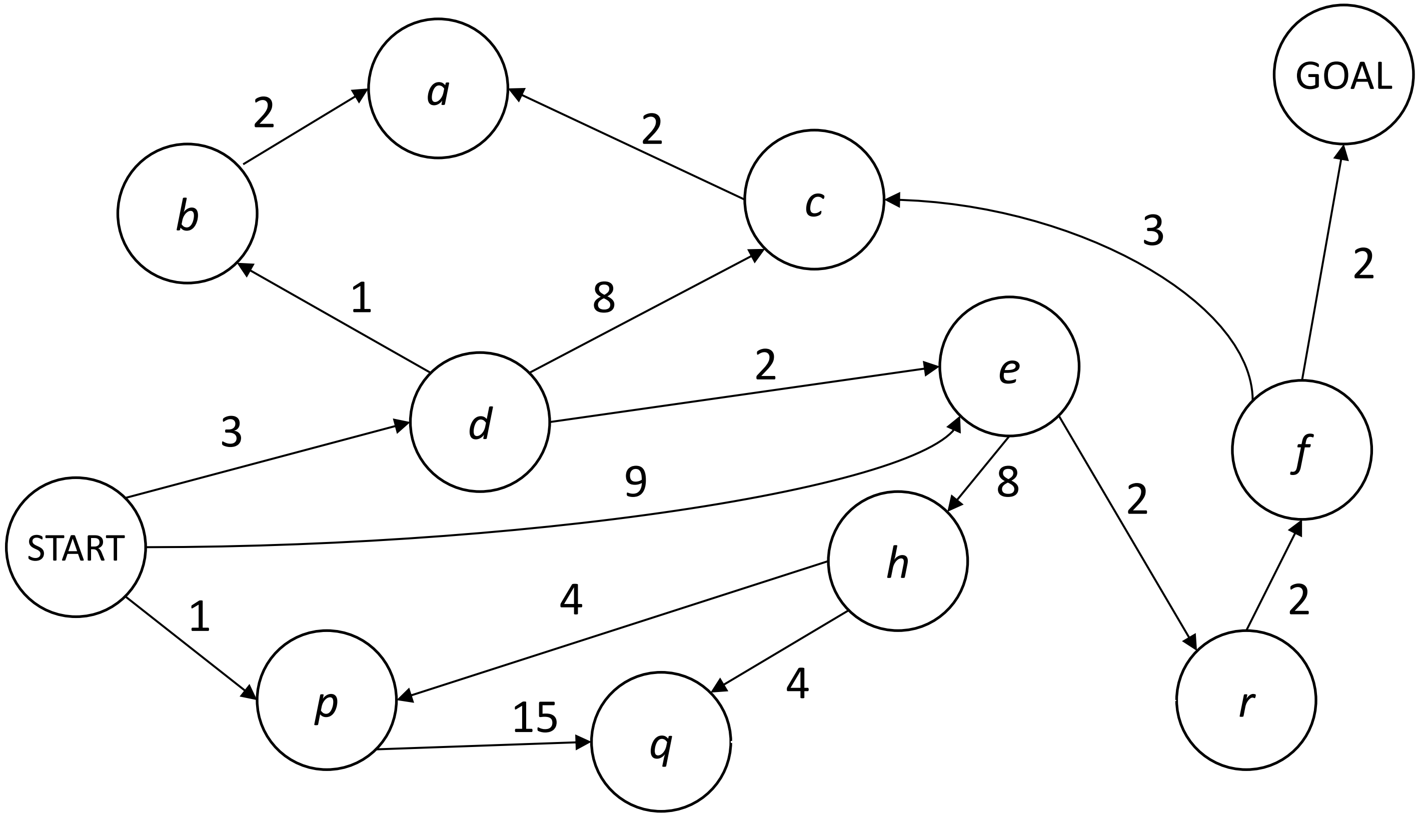
- Idea: get DFS's space advantage with BFS's time / shallow-solution advantages
  - Run a DFS with depth limit 1. If no solution…
  - Run a DFS with depth limit 2. If no solution…
  - Run a DFS with depth limit 3. …..



- Isn't that wastefully redundant?
  - Generally, most work happens in the lowest level searched, so not so bad!

Much of this slide from Berkley AI and Russell/Norvig

Cost-Sensitive Search

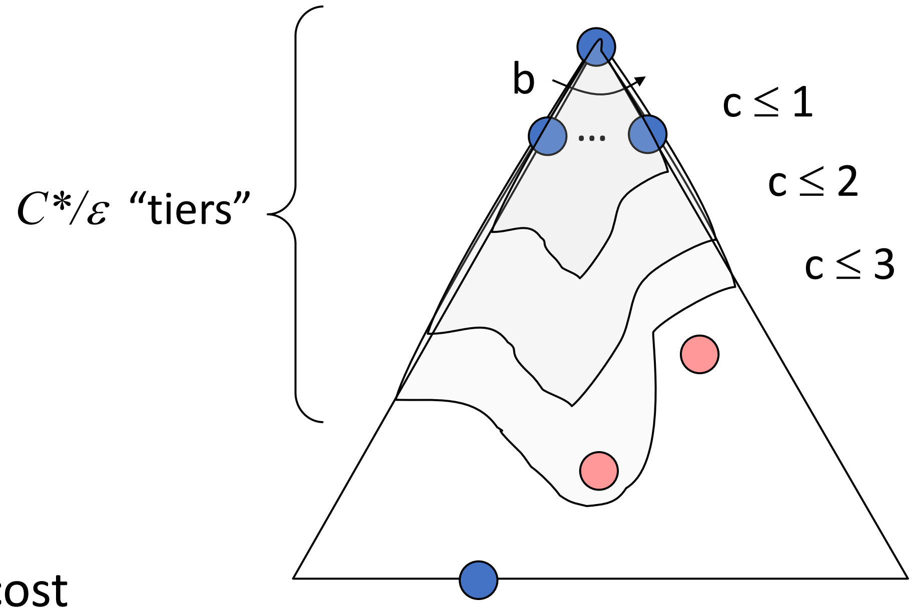


BFS finds the shortest path in terms of number of actions. It does not find the least-cost path. We will now cover a similar algorithm which does find the least-cost path.

Much of this slide from Berkley AI and Russell/Norvig

Uniform Cost Search (UCS)

- What nodes does UCS expand?
  - Processes all nodes with cost less than cheapest solution!
  - If that solution costs C^* and arcs cost at least ε, then the "effective depth" is roughly C^*/ε
  - Takes time O($b^{C^*/\varepsilon}$) (exponential in effective depth)
- How much space does the fringe take?
  - Has roughly the last tier, so O($b^{C^*/\varepsilon}$)
- Is it complete?
  - Assuming best solution has a finite cost and minimum arc cost is positive, yes!
- Is it optimal?
  - Yes! (Proof next lecture via A*)



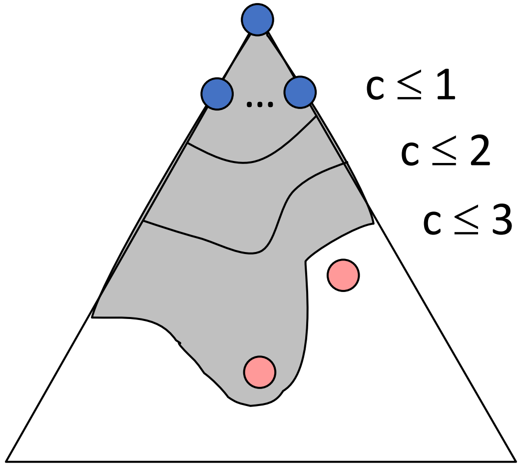
Uniform Cost Issues

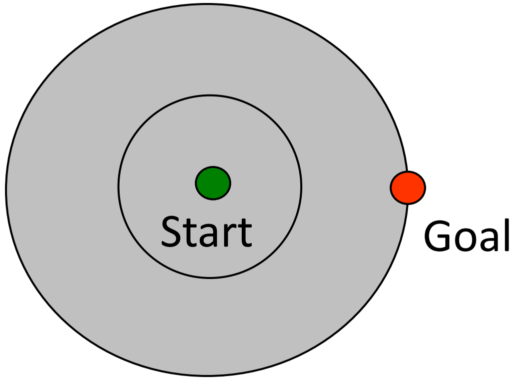
- Remember: UCS explores increasing cost contours

- The good: UCS is complete and optimal!

- The bad:
  - Explores options in every “direction”
  - No information about goal location

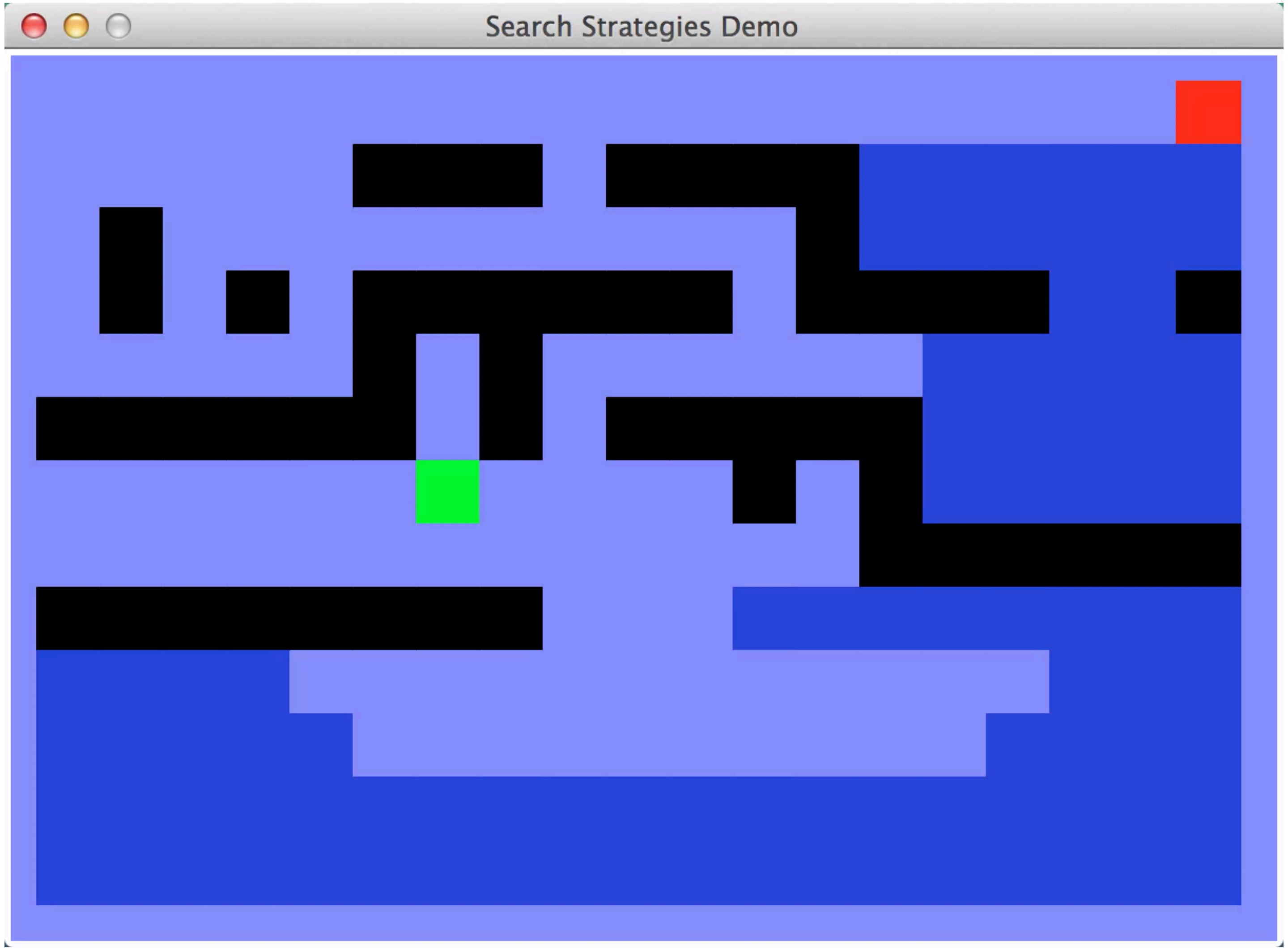
- We’ll fix that soon!





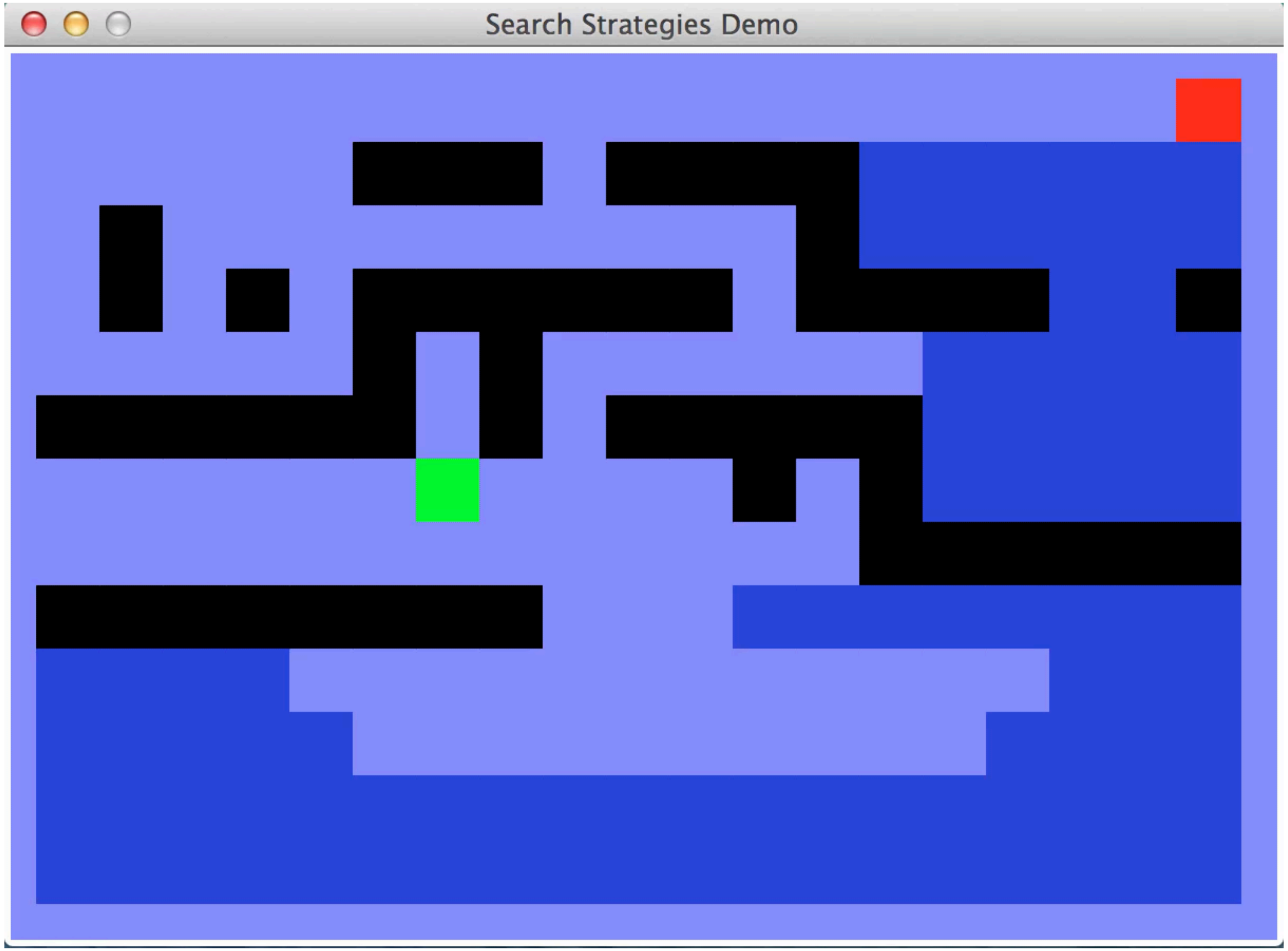
Much of this slide from Berkley AI and Russell/Norvig

Quiz: Which Search Algorithm is this?

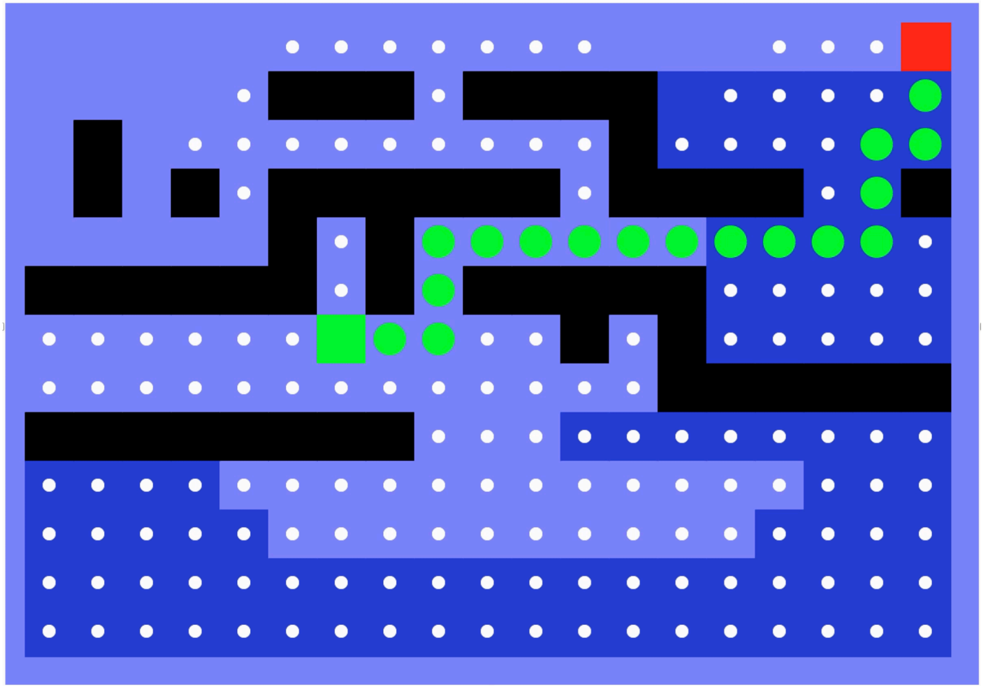


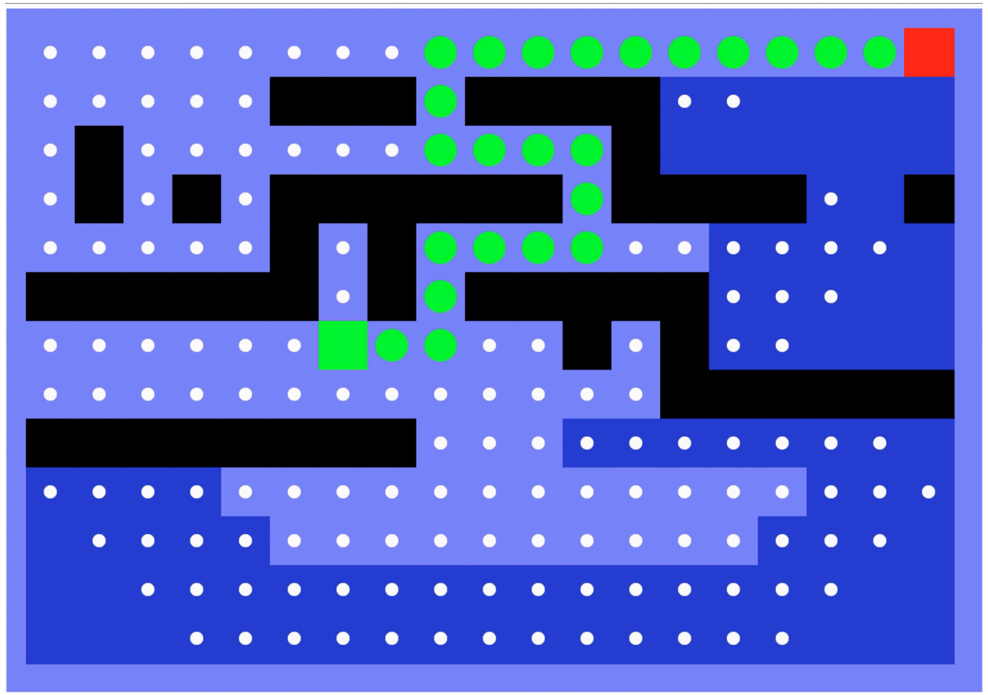
JAMES MADISON UNIVERSITY

Quiz: Which Search Algorithm is this?

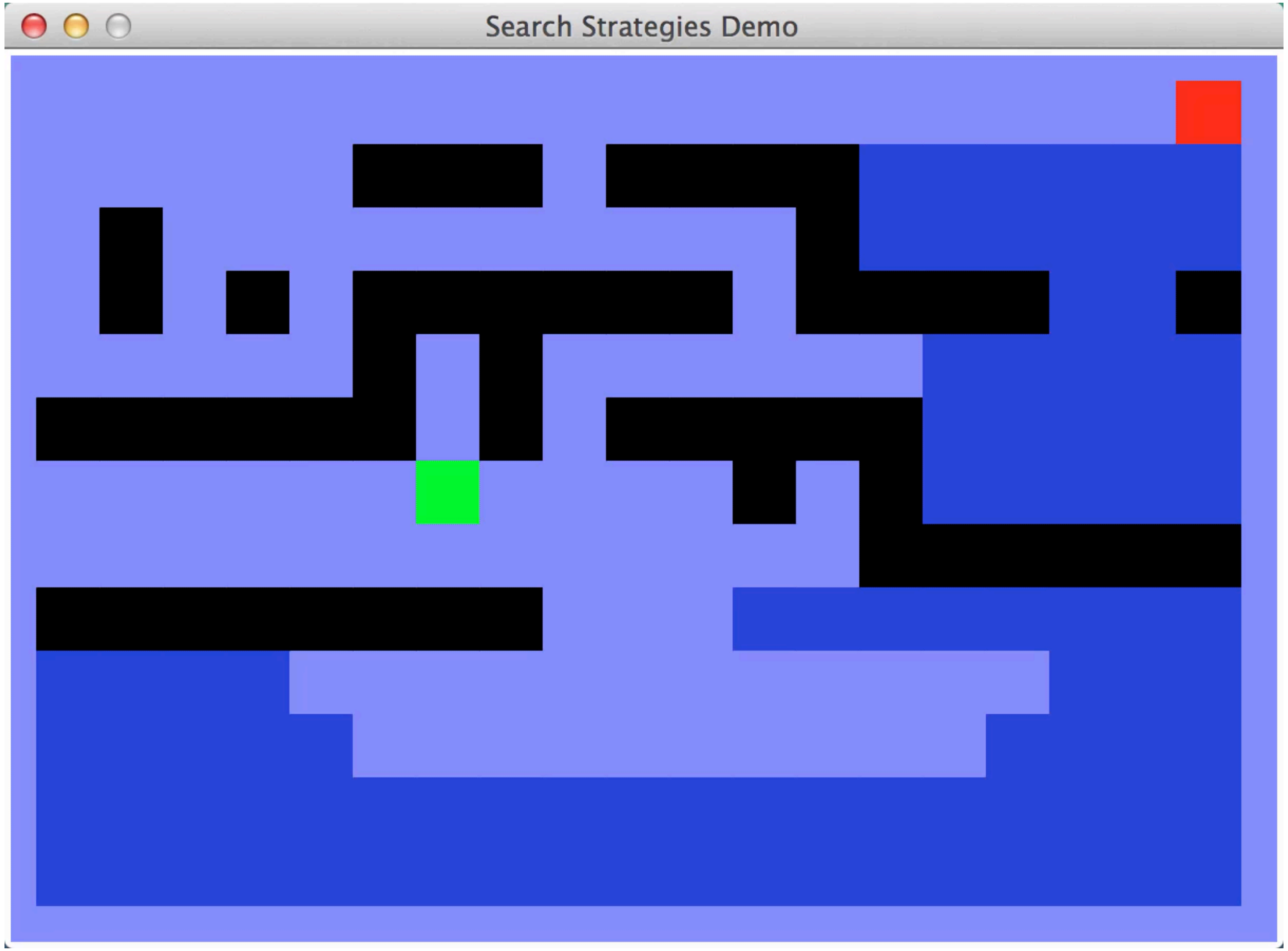


Quiz: Which Search Algorithm is this?



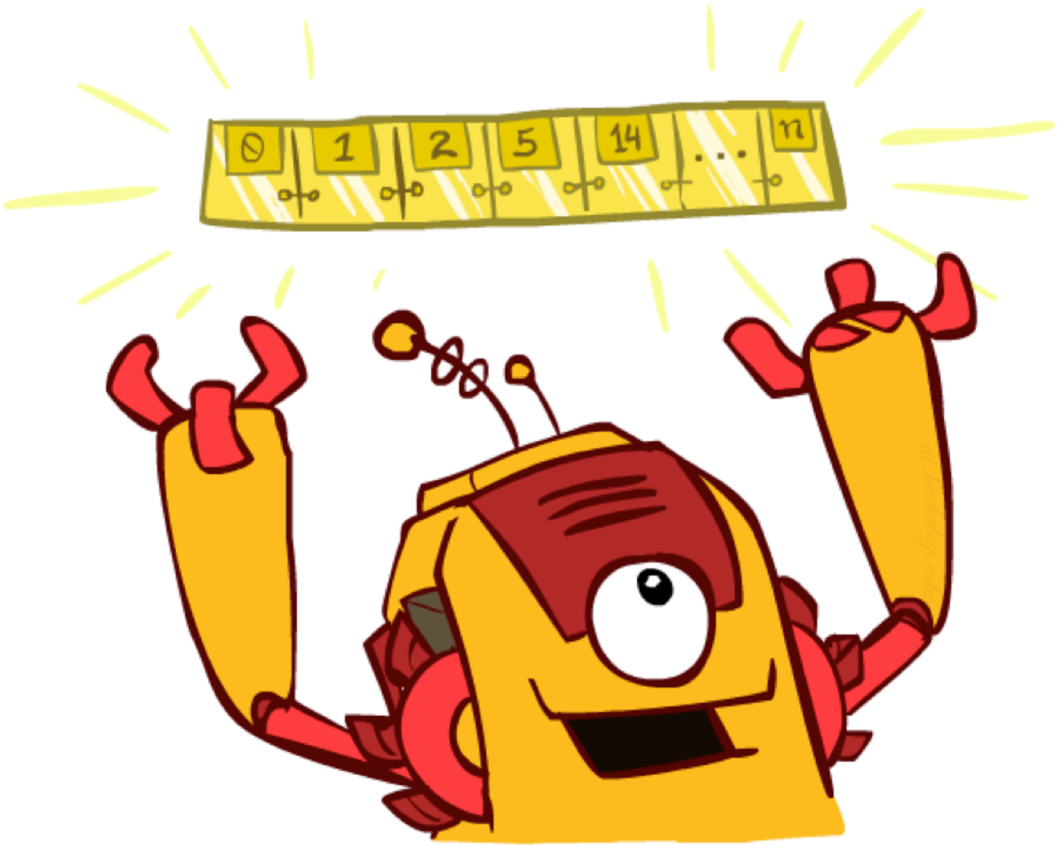


Quiz: Which Search Algorithm is this?



Differences are only in Fringe Management

- All these search algorithms are the same except for fringe strategies
  - Conceptually, all fringes are priority queues (i.e. collections of nodes with attached priorities)
  - Practically, for DFS and BFS, you can avoid the log(n) overhead from an actual priority queue, by using stacks and queues
  - Can even code one implementation that takes a variable queuing object



Much of this slide from Berkley AI and Russell/Norvig

For Next Time

- HW 0 is due tonight.
- HW 0.5 is coming soon (instead of the quiz)
- PA 1 is released. You should start on the project now.