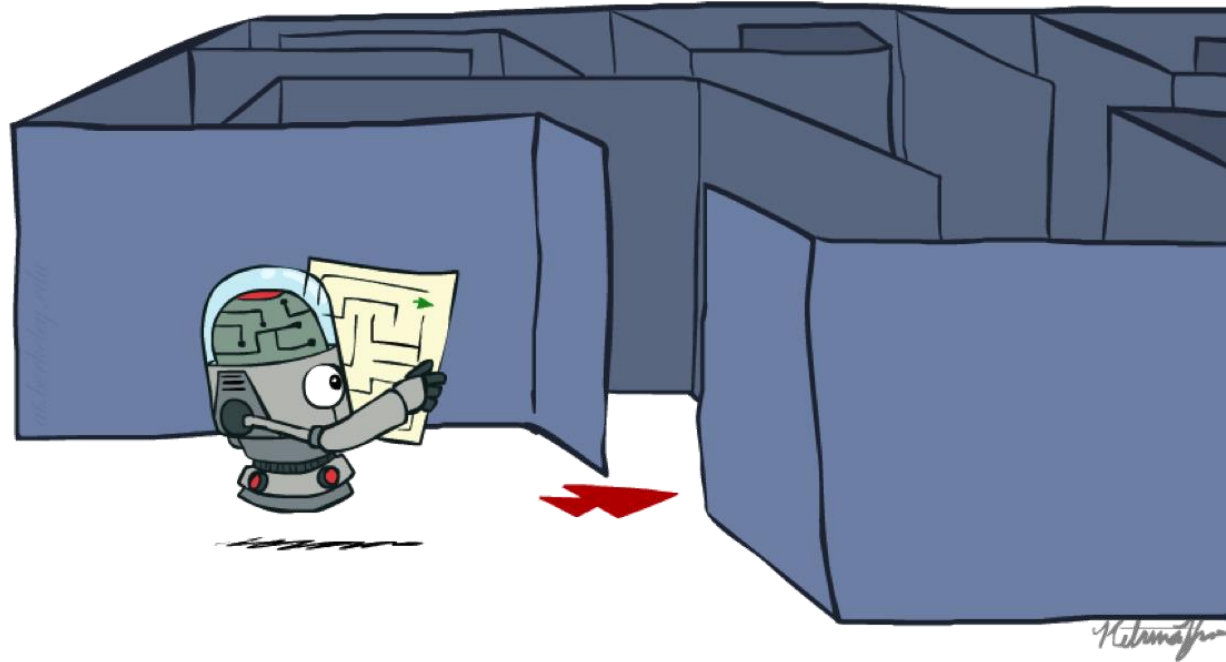


Announcements

- **Project 0: Python Tutorial**
 - Graded and corrected!
- **Homework 0: Math self-diagnostic**
 - Graded and corrected!
- **Homework 1: Search**
 - Will be released this week.
- **Project 1: Search**
 - Will be released this week.
 - Longer than most, and best way to test your programming preparedness
- **Office hours**
 - I have a new office when you want to meet, please check both for the next few weeks 😊
 - Current Office: Room 2207, Storey Innovation Center
 - New Office: Room 2235, Storey Innovation Center
 - Clarifications for homework and projects as well as technical questions
- **Make sure you fully submit your project and homework.**
 - You can team up, up to 2 students!

CSCE 580: Artificial Intelligence

Search



Pooyan Jamshidi

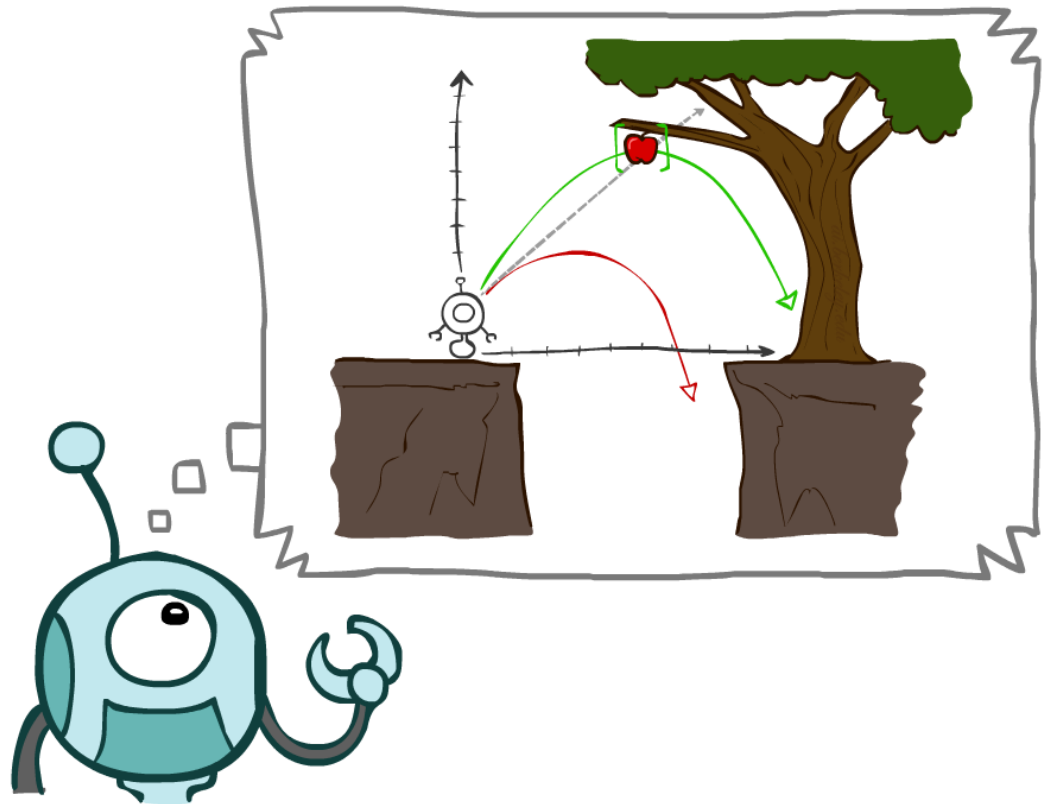
University of South Carolina

[These slides are mostly based on those of Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley, ai.berkeley.edu]



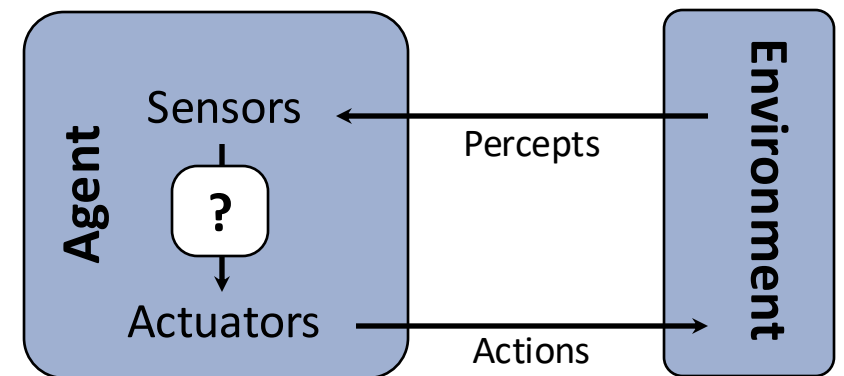
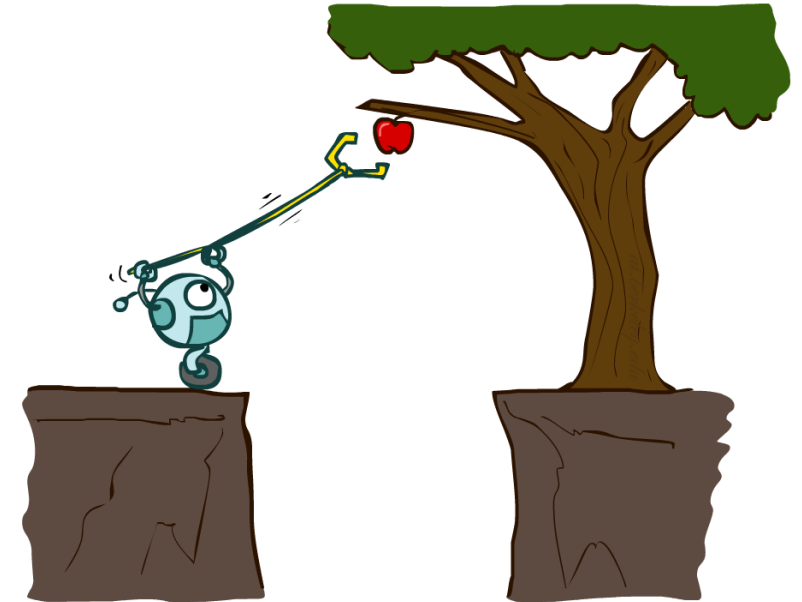
Today

- Agents that Plan Ahead
- Search Problems
- Uninformed Search Methods
 - Depth-First Search
 - Breadth-First Search
 - Uniform-Cost Search

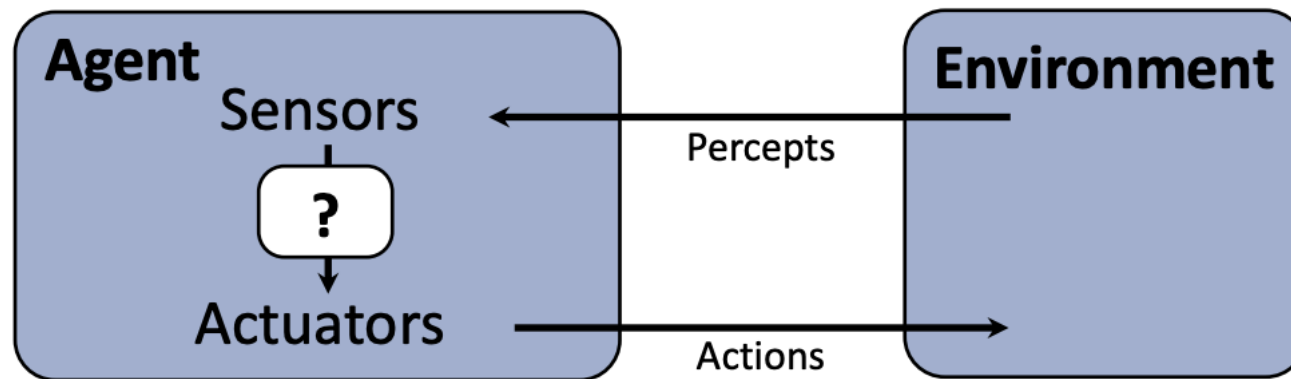
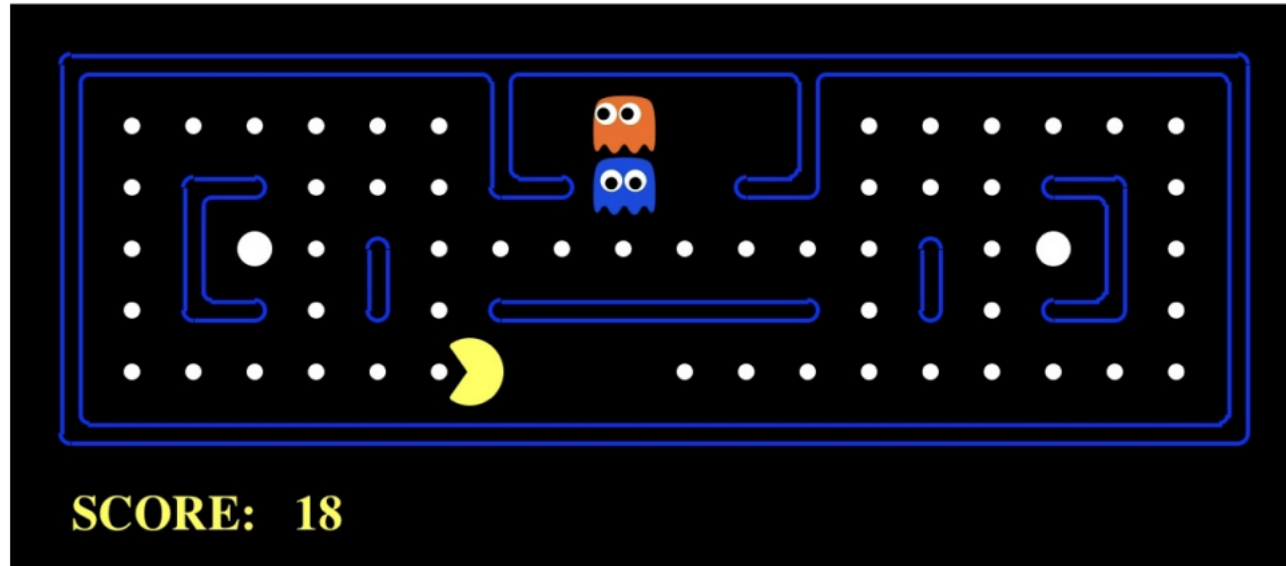


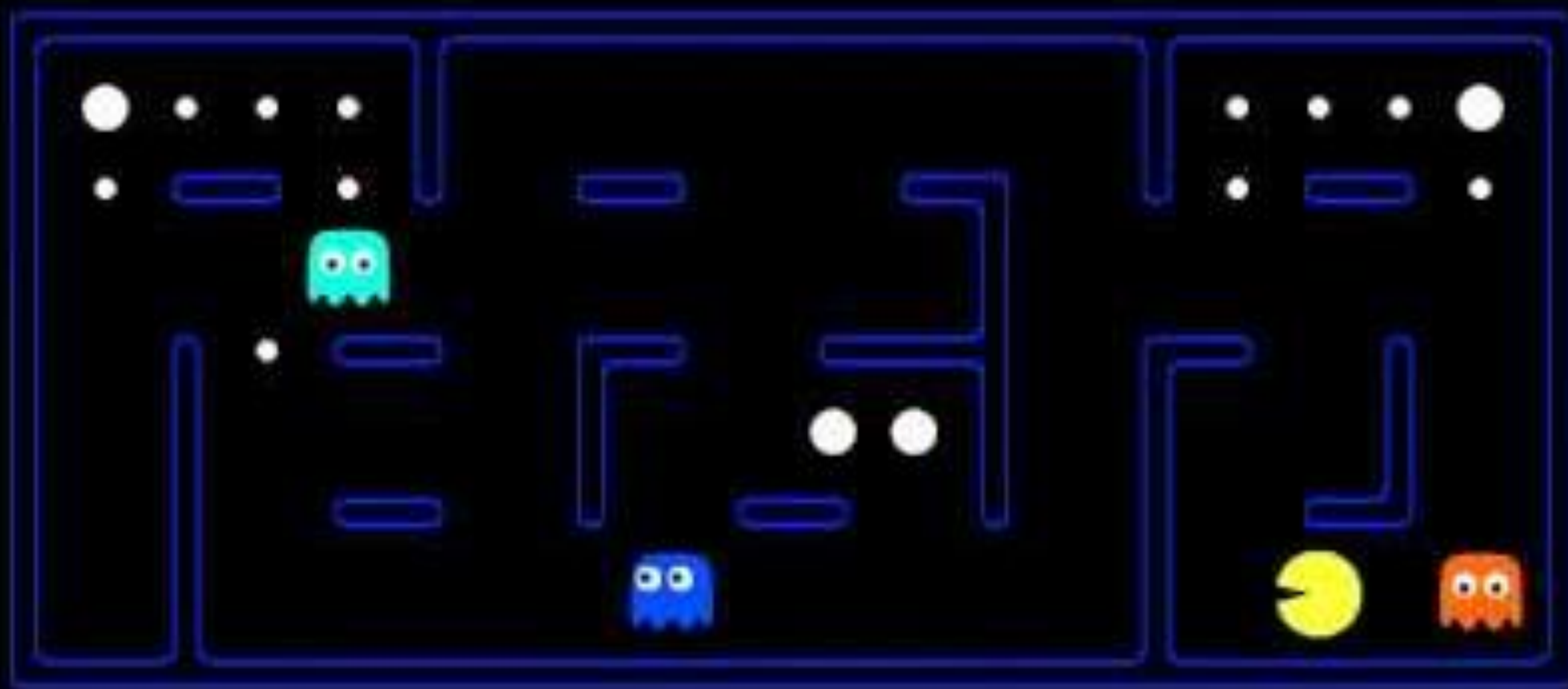
Designing Rational Agents

- An **agent** is an entity that *perceives* and *acts*.
- A **rational agent** selects actions that maximize its (expected) **utility**.
- Characteristics of the **percepts, environment, and action space** dictate techniques for selecting rational actions
- **This course** is about:
 - General AI techniques for a variety of problem types
 - Learning to recognize when and how a new problem can be solved with an existing technique



Pac-Man as an Agent





SCORE: 1282

Agents

- **A Goal of AI:** Build robust, fully autonomous agents in the real world

Intelligent (Autonomous) Agents: Examples

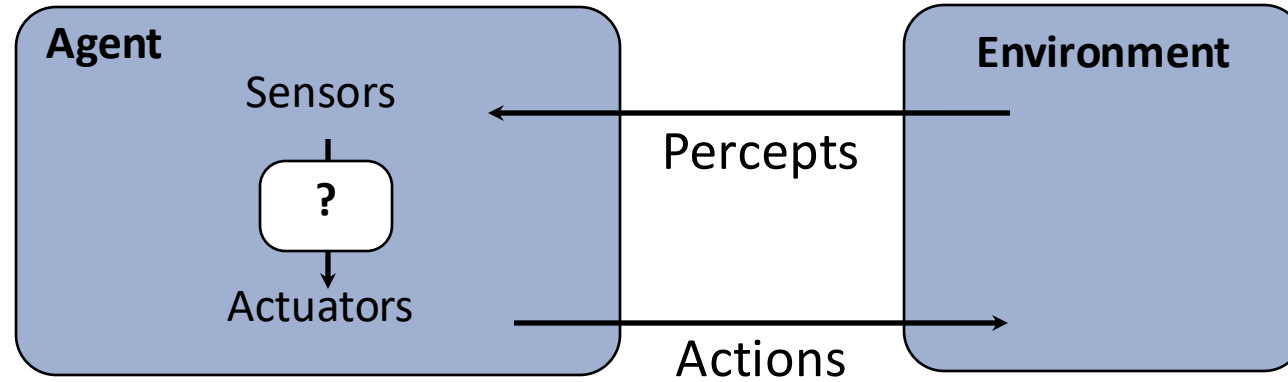
- Autonomous robot
- Information gathering agent
 - Find me the cheapest?
- E-commerce agents
 - Decides what to buy/sell and does it
- Air-traffic controller
- Meeting scheduler
- Computer-game-playing agent

Not Intelligent Agents

- Thermostat
- Telephone
- Answering machine
- Pencil
- Java object

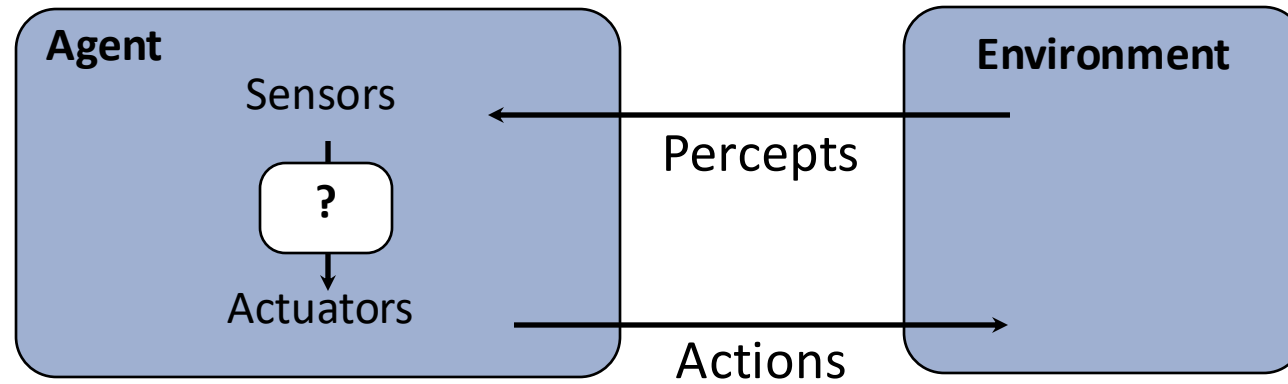
What is an Agent?

Agents and Environments



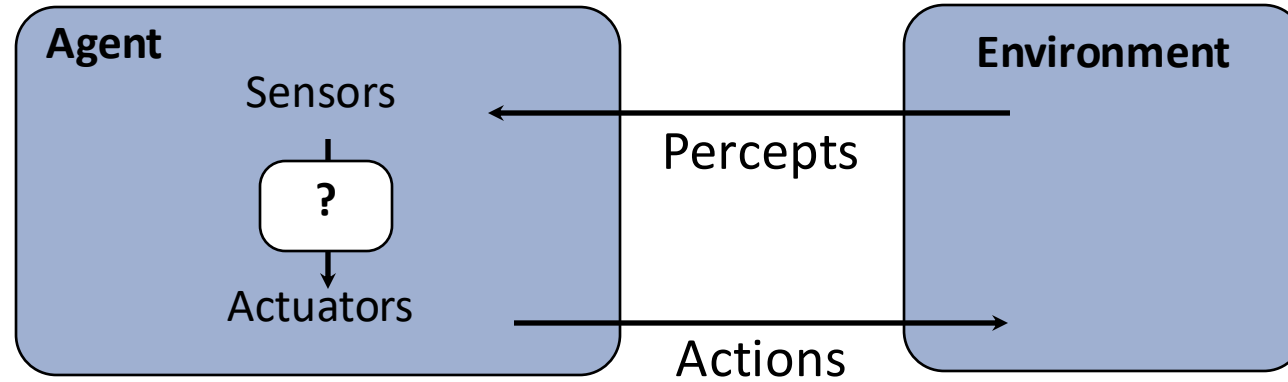
- An agent *perceives* its environment through *sensors* and *acts* upon it through *actuators*.

Agents and Environments



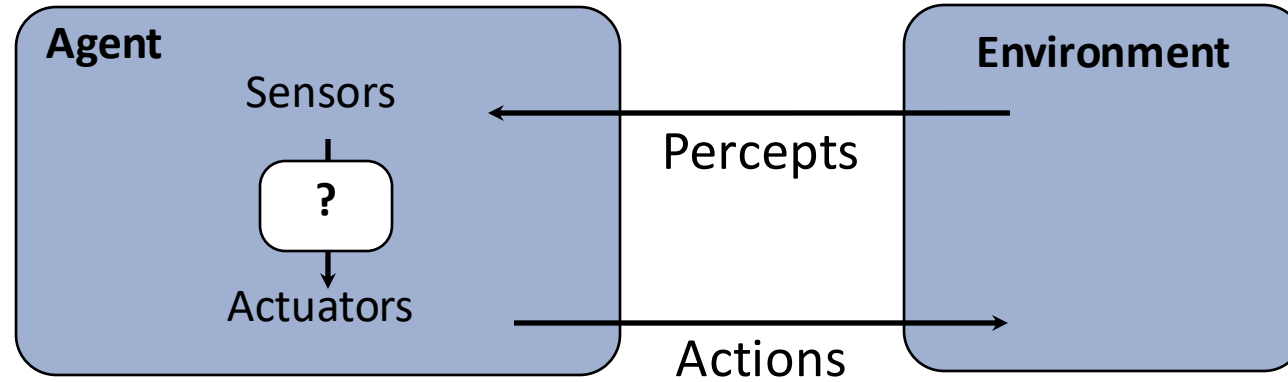
- Are humans agents?
- Yes!
 - Sensors = vision, audio, touch, smell, taste, ...
 - Actuators = muscles, secretions, changing brain state

Agents and Environments



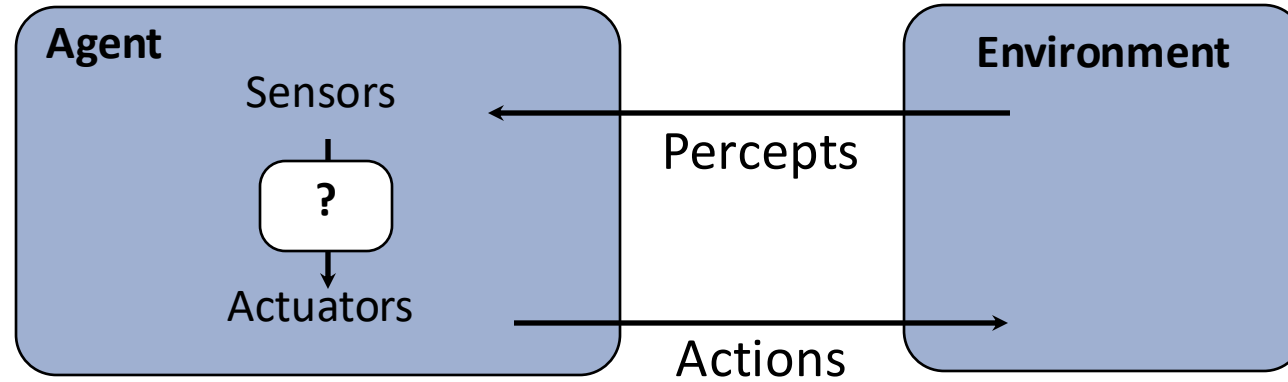
- Are Robots agents?
- Yes!
 - Sensors = cameras, laser range finders, GPS
 - Actuators = various motors

Agents and Environments



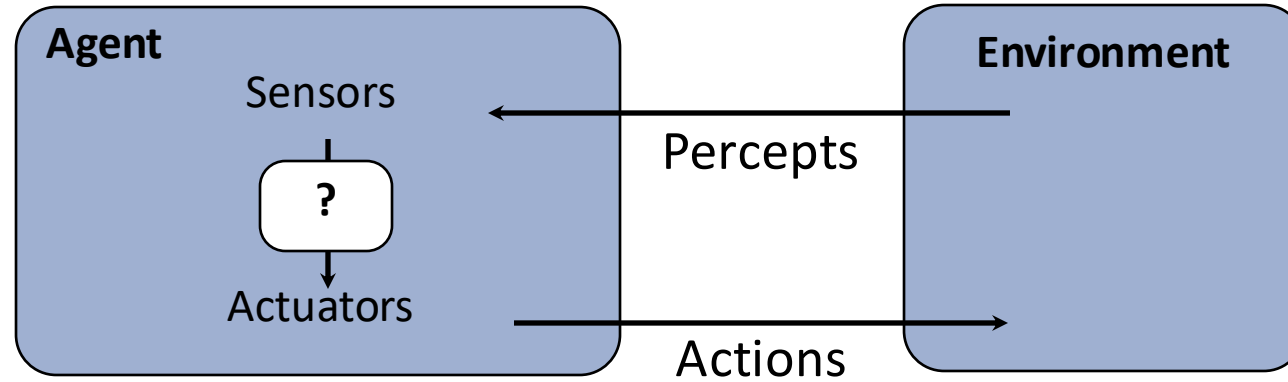
- Are pocket calculators agents?
- Yes!
 - Sensors = key state sensors
 - Actuators = digit display

Agents and Environments



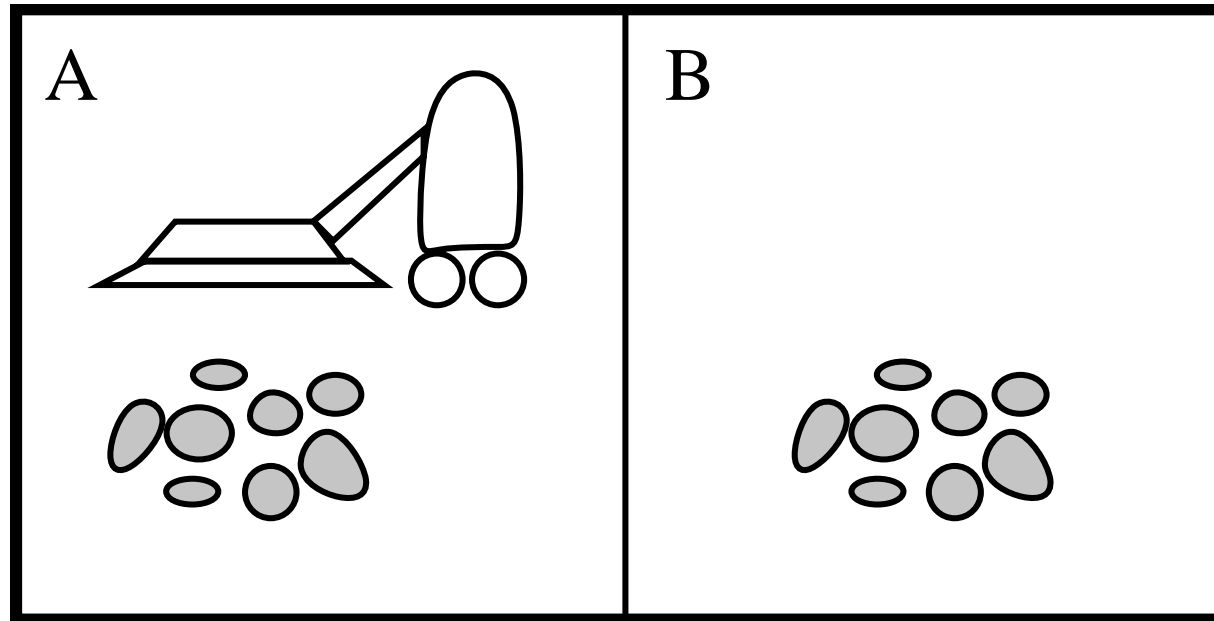
- AI is more interested in agents with substantial computation resources and environments requiring nontrivial decision making

Agents and Environments



- AI is more interested in agents with **substantial computation resources** and **environments** requiring **nontrivial decision making**

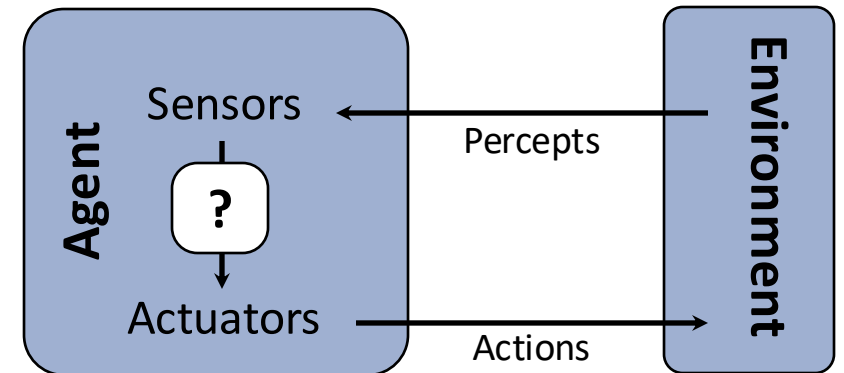
Example: Vacuum world



- Percepts: [location,status], e.g., [A,Dirty]
- Actions: Left, Right, Suck, NoOp

Rational Agents

- A **rational agent** selects actions that maximize its (expected) **utility**.
- Utility or performance measure of a vacuum-cleaner agent:
 - amount of dirt cleaned up
 - amount of time taken
 - amount of electricity consumed
 - amount of noise generated
 - etc.



Rational vs. Irrational Agents




- **Rational Agent:** Selects actions that maximize expected utility based on knowledge and available information.
 - *Example:* A chess AI choosing the best possible move after evaluating all options.
- **Irrational Agent:** Makes decisions without optimizing for utility or ignoring available information.
 - *Example:* Random move generator in a game of chess.

What is Utility in AI?




- **Utility** is a measure of how desirable a particular outcome is.
- **Expected Utility:** The sum of utilities of all possible outcomes, weighted by their probabilities.
- **Examples of Utility in Different Scenarios:**
 - **Robotics:** Minimize energy use while maximizing task efficiency.
 - **Recommendation Systems:** Maximize user satisfaction with suggested content.
 - **Healthcare AI:** Maximize diagnostic accuracy while minimizing time.

Rational Agents in Real Life

- **Self-Driving Cars:**

- Maximize passenger safety 
- Minimize fuel consumption 
- Avoid traffic delays 

- **Delivery Drones:**

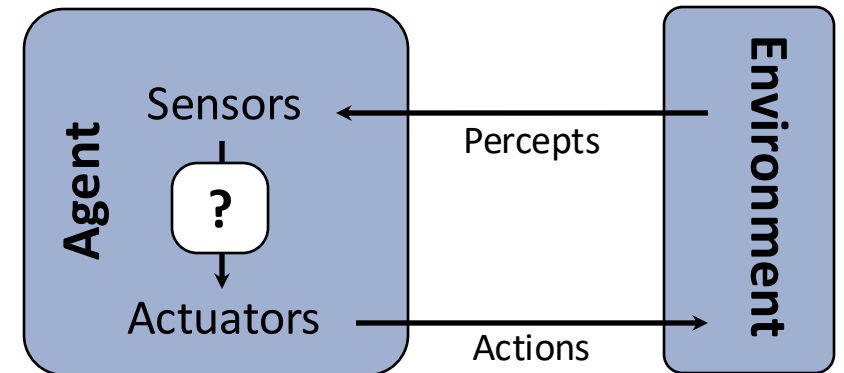
- Maximize delivery speed 
- Minimize battery consumption 
- Avoid obstacles or hazards 

Activity: Is This Agent Rational?

- *“Consider the following scenarios. Are these agents rational? Why or why not?”*
 1. A **thermostat** keeps the room at a constant temperature by turning on/off the heater.
 2. A **navigation app** suggests the longest possible route to avoid all traffic lights.
 3. A **robot vacuum** keeps cleaning the same spot repeatedly.

Rational Agents

- A **rational agent**
 - acts appropriately given goals and circumstances
 - is flexible to changing environments and goals
 - learns from experience
 - makes appropriate choices given perceptual and computational limitations
- Characteristics of the **percepts, environment,** and **action space** dictate techniques for selecting rational actions.



Rational Agents

- Are rational agents **omniscient**?
 - No – they are limited by the available percepts
- Are rational agents **clairvoyant**?
 - No – they may lack knowledge of the environment dynamics
- Do rational agents **explore and learn**?
 - Yes – in unknown environments these are essential
- So rational agents are not necessarily successful, but they are **autonomous**.

Discussion Item

- A realistic agent has finite amount of computation and memory available. Assume an agent is killed because it did not have enough computation resources to calculate some rare event that eventually ended up killing it. Can this agent still be rational?

Announcements

- Homework 1: Search
 - Released today---Thursday, Feb 13.
 - It will be due on Monday, Feb 24 (late day on Friday, Feb 28)
 - We may extend due dates depending on when we finish the lecture on search 😊
- Project 1: Search
 - It will be released on Monday, Feb 17.
 - It will be due on Monday, Feb 24 (late day on Friday, Feb 28)
 - Please wait until you receive a notification about it from us!
 - We may extend due dates, if necessary, but do not rely on them!

Please take a picture of the announcement in class
and post it to Piazza (official)/GroupME (unofficial) for those who could not attend!

Announcements

- I encourage teams of two students to do the projects
- I encourage pair programming
- DO NOT SEPARATE THE TASKS BETWEEN EACH OTHER!
- Please pay attention to pinned posts on Piazza



PAIR PROGRAMMING

PEAS: Performance measure, Environment, Actuators, Sensors

PEAS: Pacman

- Performance measure
 - -1 per step; + 10 food; +500 win; -500 die;
- Environment
 - Maze, food, ghosts, ...
- Actuators
 - Pacman's body and mouth
- Sensors
 - Some sort of Vision (Entire state is visible)



PEAS: Automated Taxi

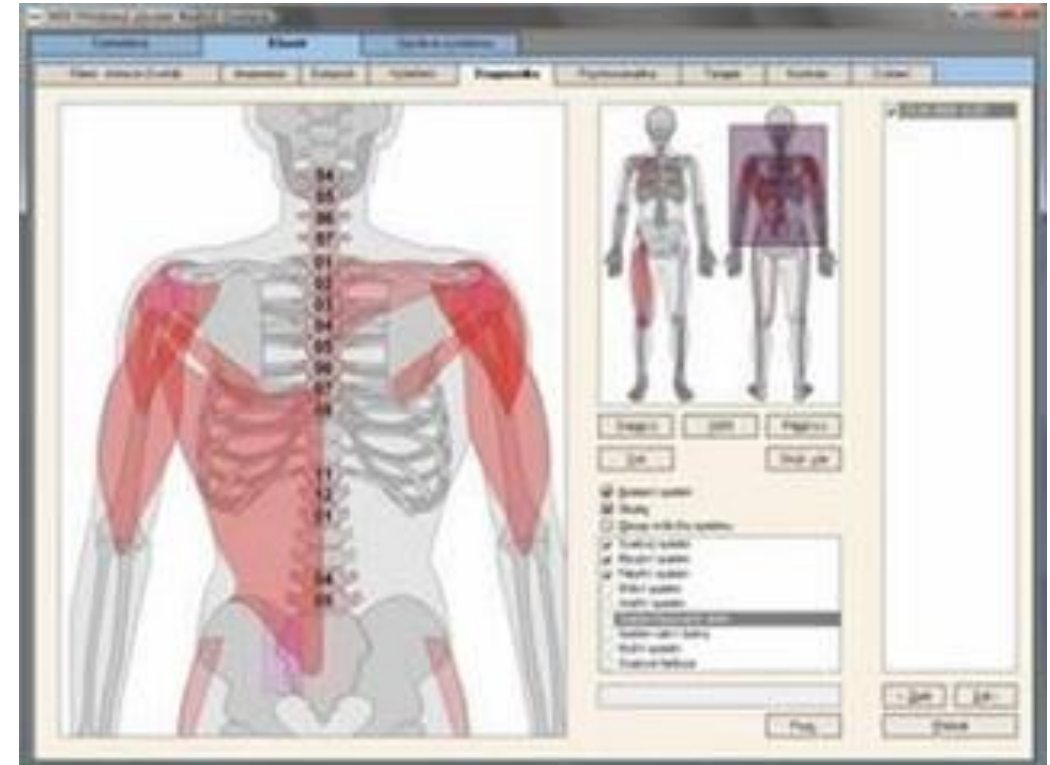
- Performance measure
 - Income, happy customer, vehicle costs, fines, insurance premiums
- Environment
 - US streets, other drivers, customers
- Actuators
 - Steering, brake, gas, display/speaker
- Sensors
 - Camera, radar, accelerometer, engine sensors, microphone



Image: <http://nypost.com/2014/06/21/how-google-might-put-taxi-drivers-out-of-business/>

PEAS: Medical Diagnosis System

- Performance measure
 - Patient health, cost, reputation
- Environment
 - Patients, medical staff, insurers, courts
- Actuators
 - Screen display, email
- Sensors
 - Keyboard/mouse



Environment Types

Environment Types

- **Fully Observable** (vs. Partially Observable)
- **Deterministic** (vs. Stochastic)
- **Episodic** (vs. Sequential)
- **Static** (vs. Dynamic)
- **Discrete** (vs. Continuous)
- **Single-Agent** (vs. Multi-Agent):

Fully Observable vs. Partially-Observable Domains

- **Fully-observable:** The agent has access to all information in the environment relevant to its task.
- **Partially-observable:** Parts of the environment are inaccessible

Pacman	Crossword	Backgammon	Pick&Place Robot	Diagnosis	Taxi
Fully	Fully	Fully	Partially	Partially	Partially

Deterministic vs. Stochastic Domains

If an agent knew the initial state and its action, could it predict the resulting state? The dynamics can be:

- **Deterministic:** the resulting state is determined from the action and the state
- **Stochastic:** there is uncertainty about the resulting state

Pacman	Crossword	Backgammon	Pick&Place Robot	Diagnosis	Taxi
Deterministic	Deterministic	Stochastic	Stochastic	Stochastic	Stochastic

Episodic vs Sequential Domains

- **Episodic:** Current action is independent of previous actions.
- **Sequential:** Current choice of action will affect future actions

Pacman	Crossword	Backgammon	Pick&Place Robot	Diagnosis	Taxi
Sequential	Sequential	Sequential	Episodic	Sequential	Sequential

Static vs Dynamic Domains

- **Static:** Environment does not change while the agent is deliberating over what to do
- **Dynamic:** Environments does change

Pacman

Crossword

Backgammon

Pick&Place
Robot

Diagnosis

Taxi

Static

Static

Static

Dynamic

Dynamic

Dynamic

Discrete vs Continuous Domains

- **Discrete**: A limited number of distinct, clearly defined states, percepts, actions, and time steps (otherwise **continuous**)

Pacman	Crossword	Backgammon	Pick&Place Robot	Diagnosis	Taxi
Discrete	Discrete	Discrete	Continuous	Continuous	Continuous

Single-agent vs. Multi-agent Domains

- Does the environment include other agents?
- If there are other agents whose actions affect us
 - It can be useful to explicitly model their goals and beliefs, and how they **react** to our actions
- Other agents can be: cooperative, competitive, or a bit of both

Pacman

Crossword

Backgammon

Pick&Place
Robot

Diagnosis

Taxi

Multi

Single

Multi

Single

Single

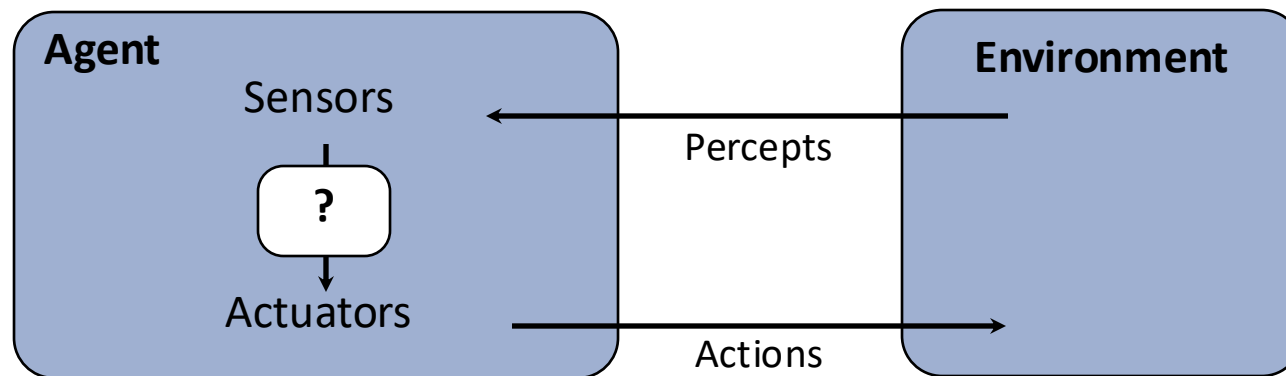
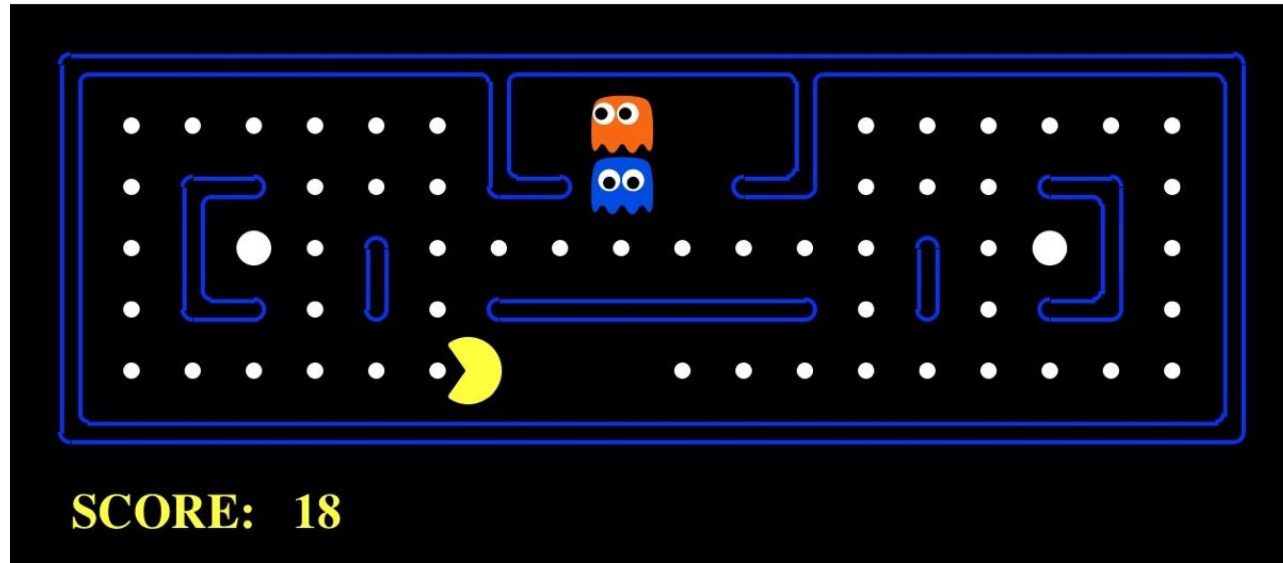
Multi

Environment Types: Summary

	Pacman	Crossword	Backgammon	Pick&Place Robot	Diagnosis	Taxi
Fully or Partially Observable	Fully	Fully	Fully	Partially	Partially	Partially
Deterministic or Stochastic	Deterministic	Deterministic	Stochastic	Stochastic	Stochastic	Stochastic
Episodic or Sequential	Sequential	Sequential	Sequential	Episodic	Sequential	Sequential
Static or Dynamic	Static	Static	Static	Dynamic	Dynamic	Dynamic
Discrete or Continuous	Discrete	Discrete	Discrete	Continuous	Continuous	Continuous
Single-agent or Multiagent	Multi	Single	Multi	Single	Single	Multi

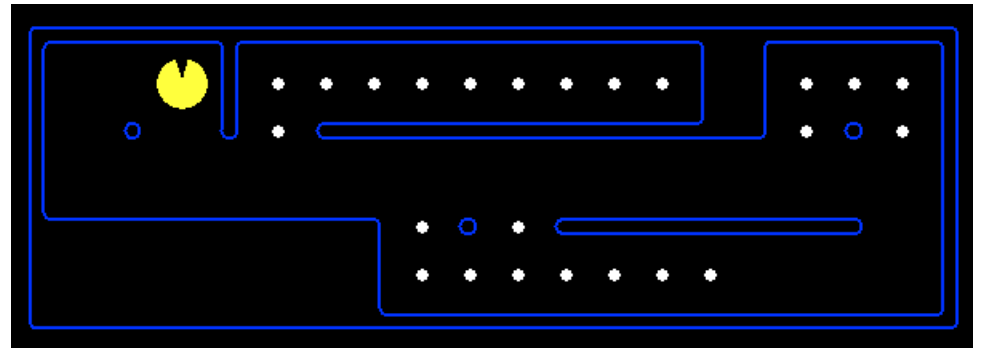
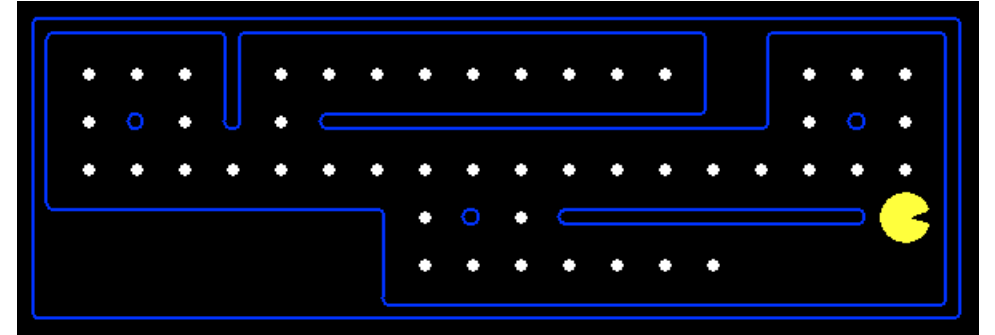
Agent Types

Pac-Man as an Agent

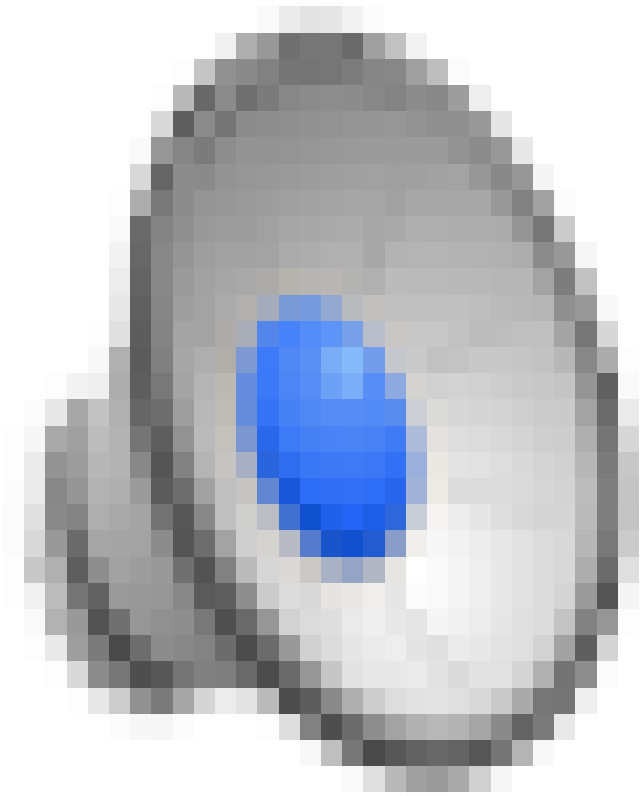


Reflex Agents

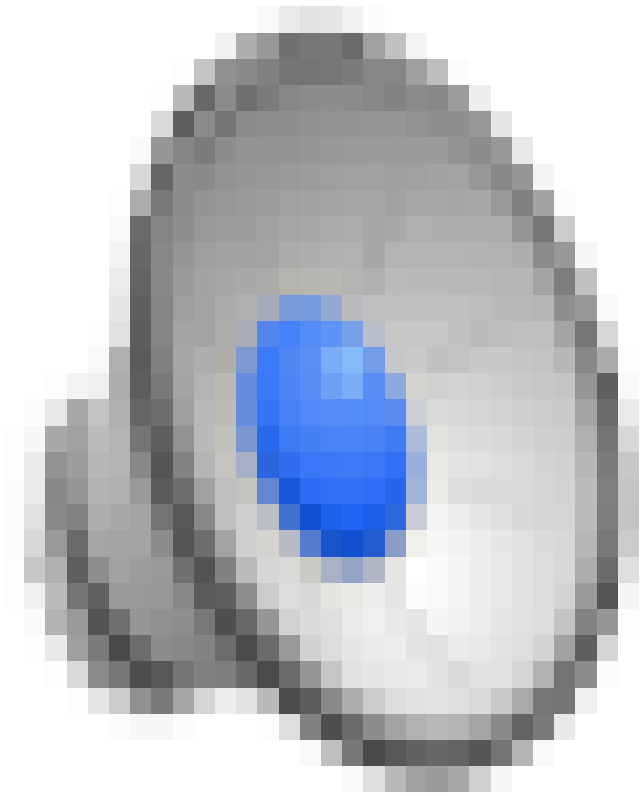
- Reflex agents:
 - Choose action based on current percept (and maybe memory)
 - May have memory or a model of the world's current state
 - Do not consider the future consequences of their actions
 - Consider how the world IS
- Can a reflex agent be rational?



Video of Demo Reflex Optimal

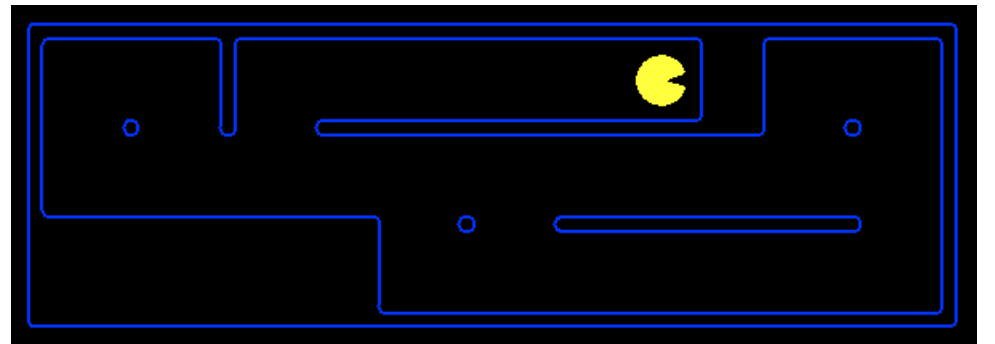
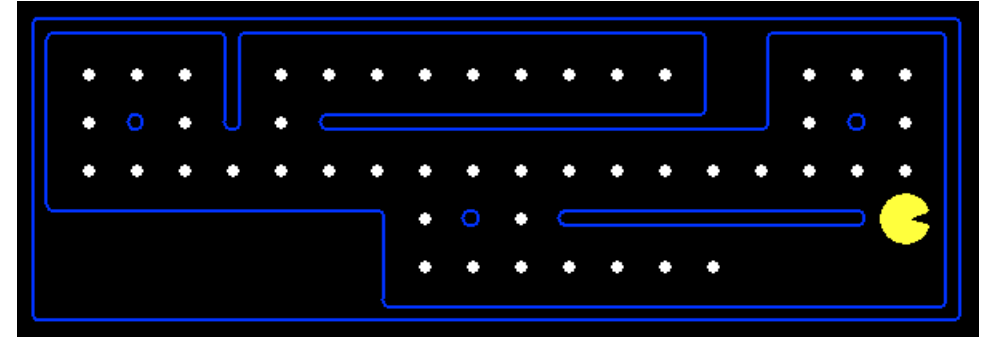


Video of Demo Reflex Odd

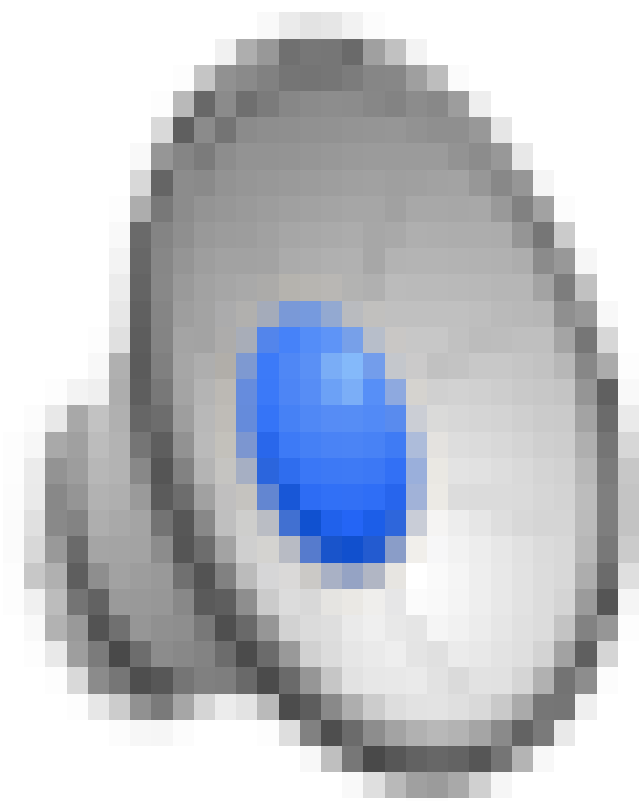


Planning Agents

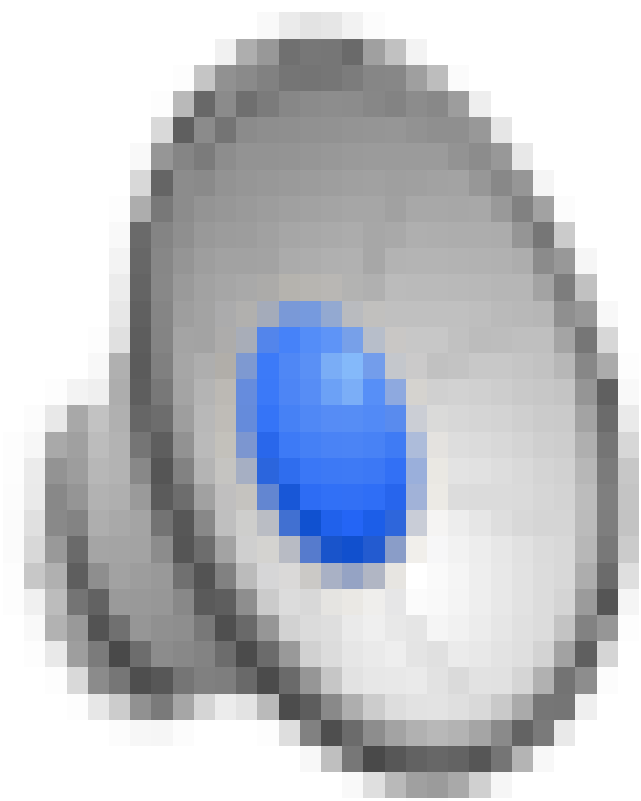
- Planning agents:
 - Ask “what if”
 - Decisions based on (hypothesized) consequences of actions
 - Must have a model of how the world evolves in response to actions
 - Must formulate a goal (test)
 - Consider how the world **WOULD BE**
- Optimal vs. complete planning
- Planning vs. replanning



Video of Demo Re-planning



Video of Demo Mastermind

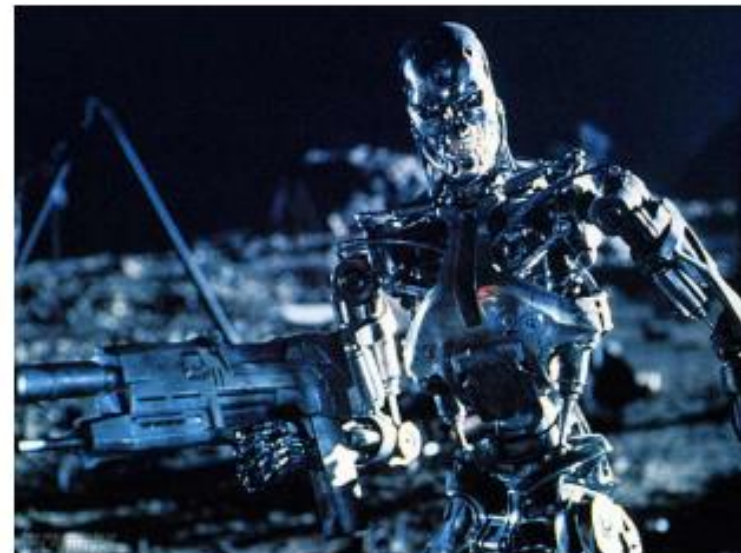




SCORE: 1282

Ethics and Implications

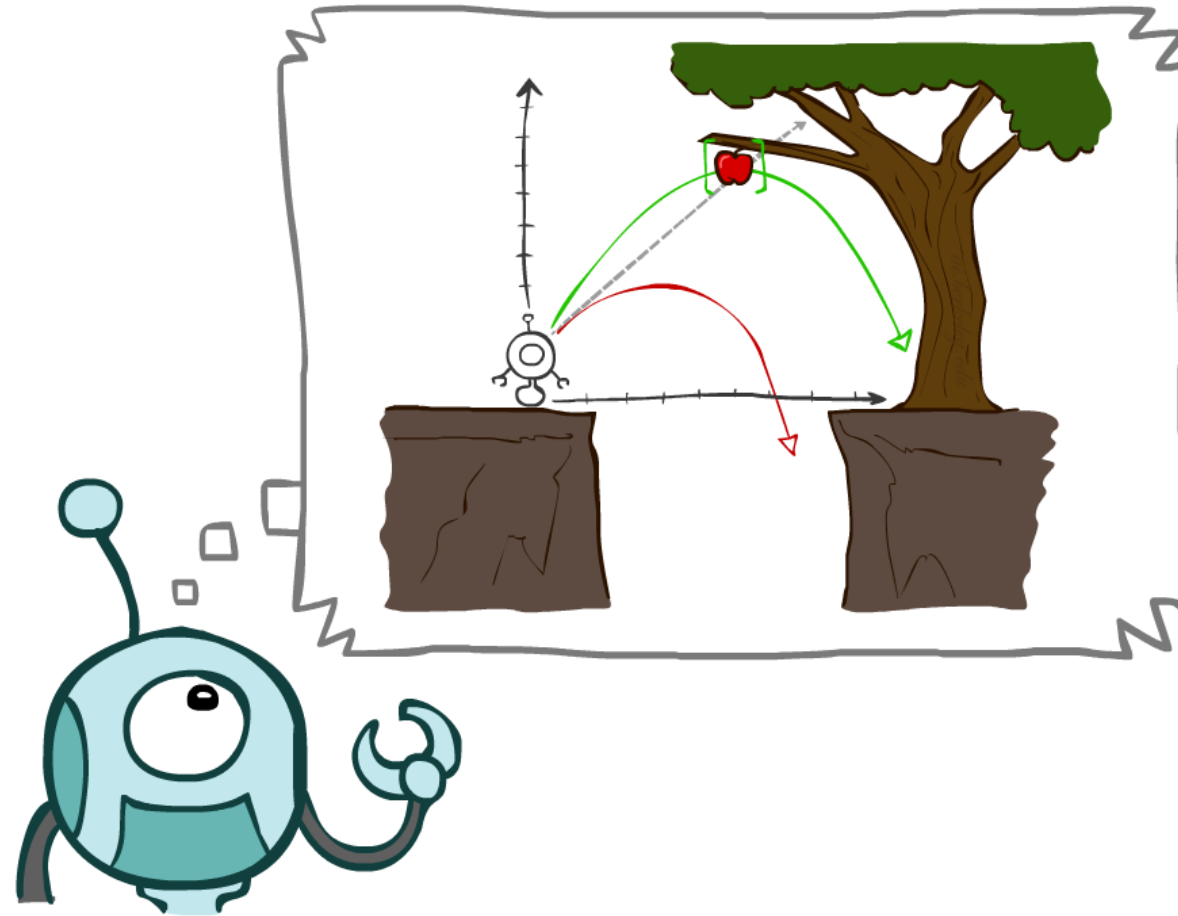
- Robust, fully autonomous agents in the real world
- What happens when we achieve this goal?



Ethics and Implications

- Who is liable if a robot driver has an accident?
- What will we do with super-intelligent machines?
- Would such machines have conscious existence? Rights?
- Can human minds exist indefinitely within machines (in principle)?

Agents that Plan

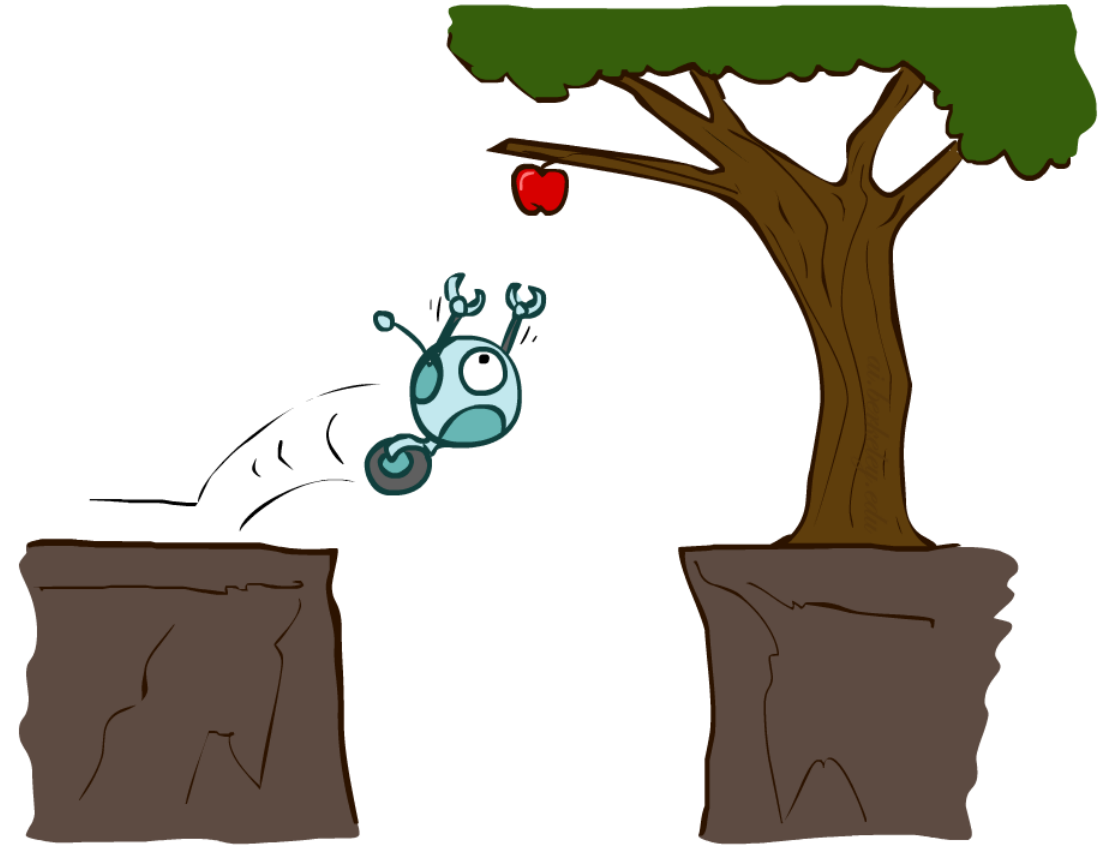


Agents that Plan Ahead (In Depth)

- **Reflex Agents:**
 - React immediately to percepts (e.g., thermostat).
- **Planning Agents:**
 - Predict consequences of actions before acting (e.g., Chess AI).

Reflex Agents

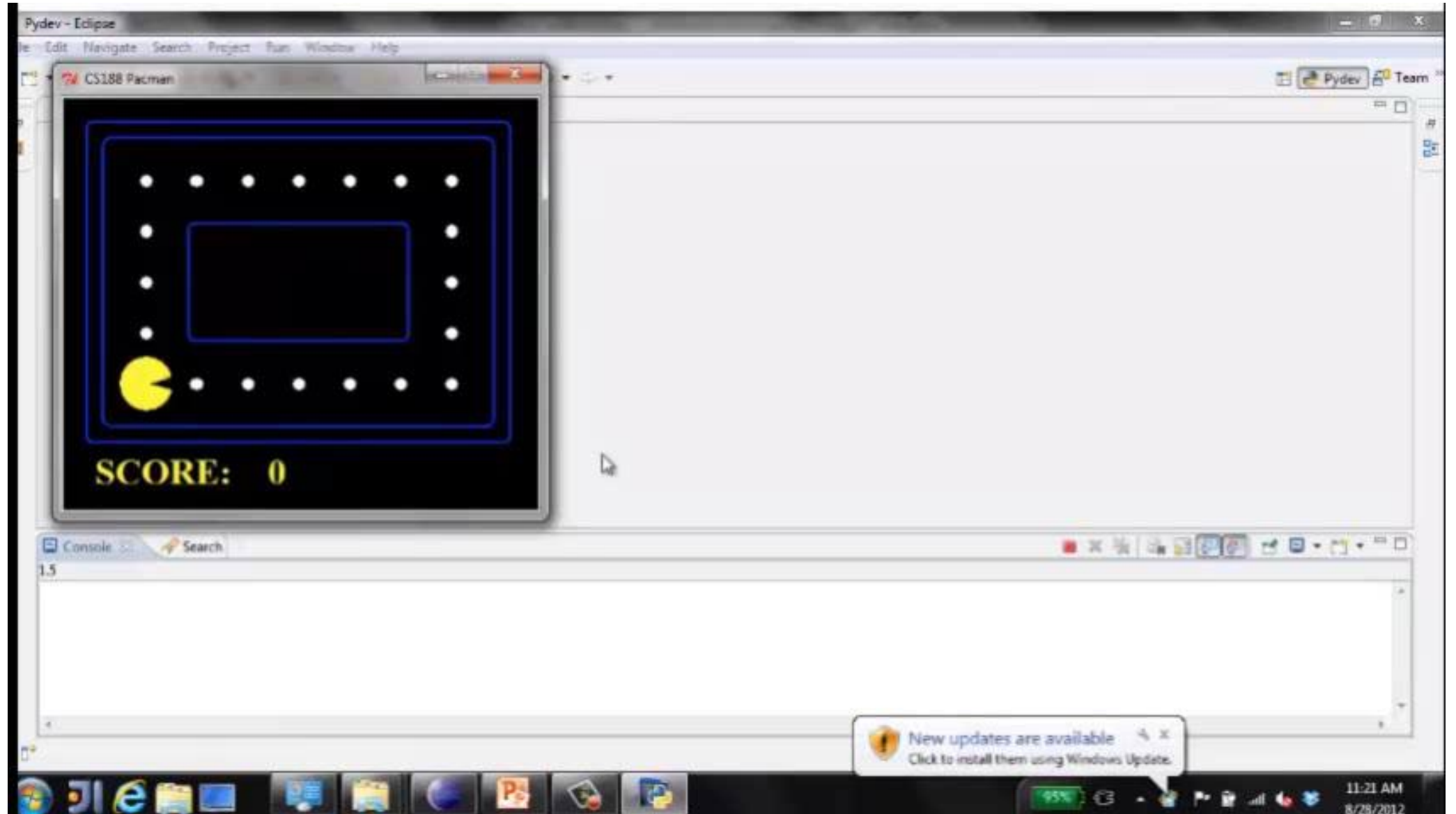
- Reflex agents:
 - Choose action based on current percept (and maybe memory)
 - May have memory or a model of the world's current state
 - Do not consider the future consequences of their actions
 - Consider how the world IS
- Can a reflex agent be rational?



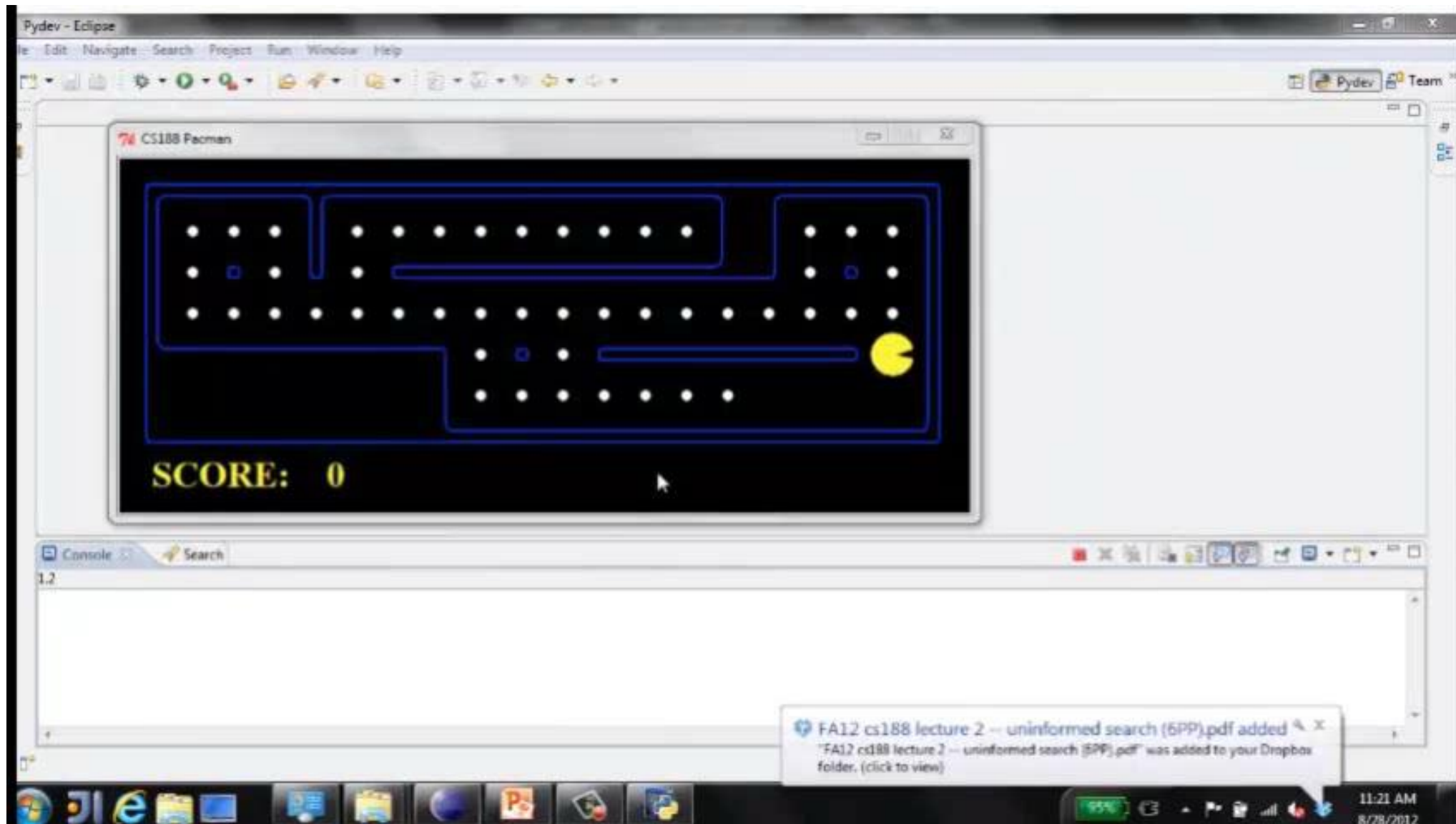
[Demo: reflex optimal (L2D1)]

[Demo: reflex optimal (L2D2)]

Video of Demo Reflex Optimal

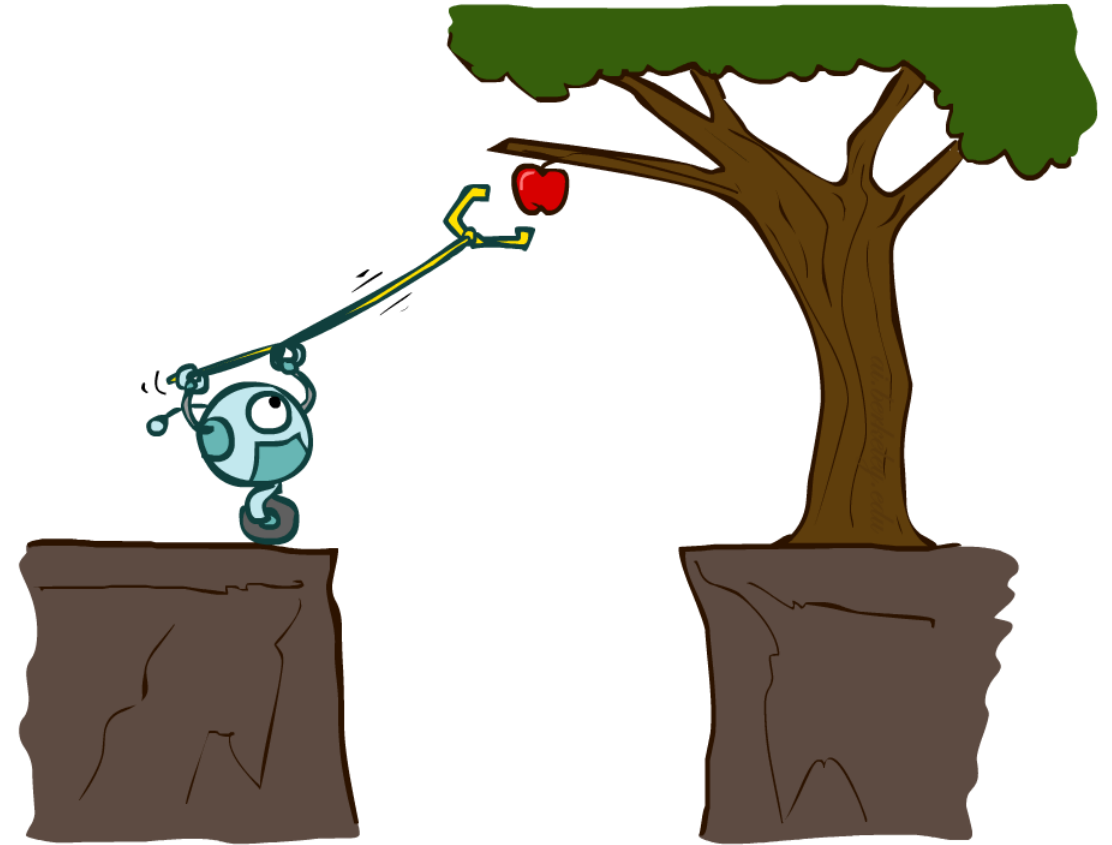


Video of Demo Reflex Odd



Planning Agents

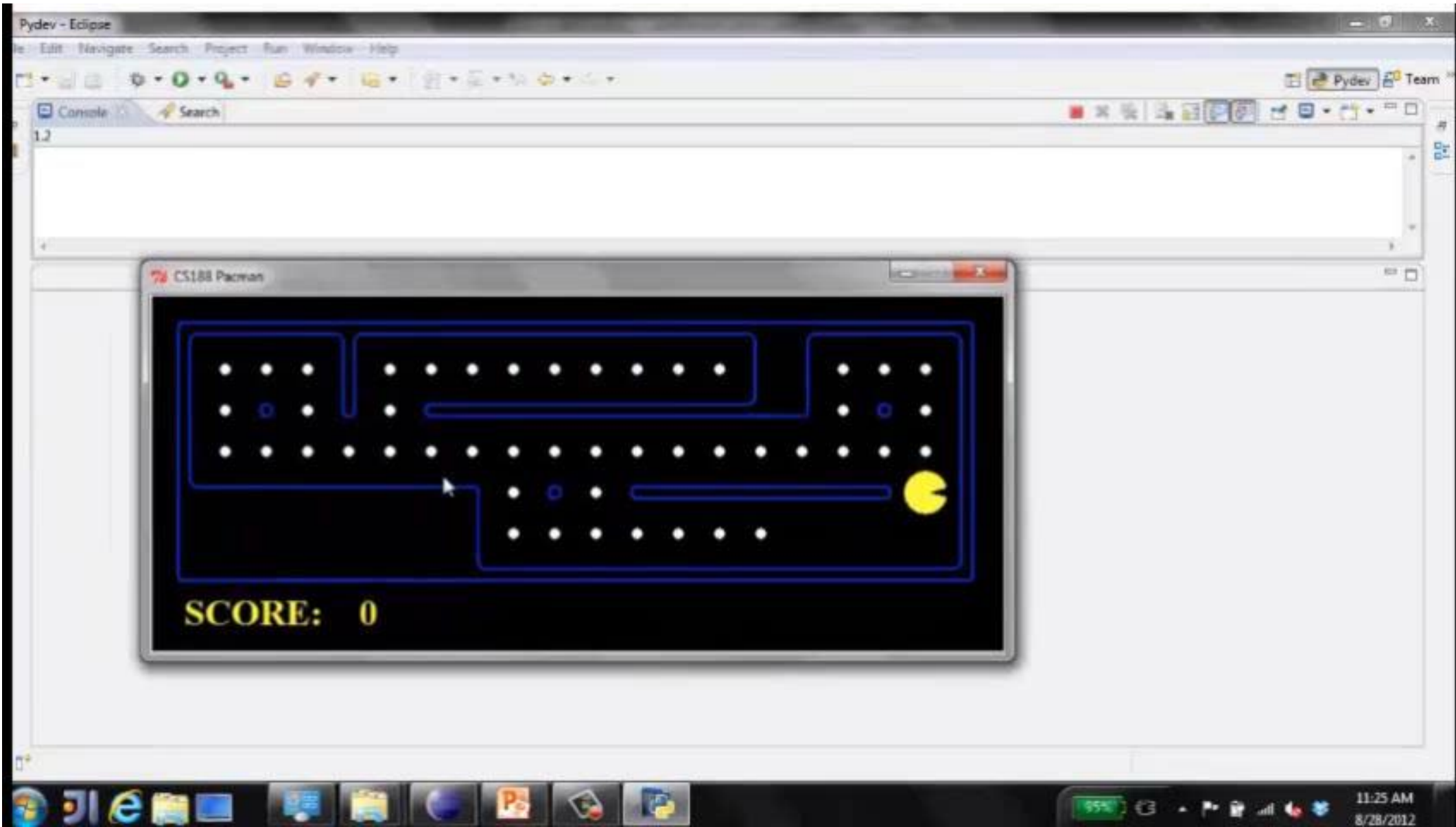
- Planning agents:
 - Ask “what if”
 - Decisions based on (hypothesized) consequences of actions
 - Must have a model of how the world evolves in response to actions
 - Must formulate a goal (test)
 - Consider how the world **WOULD BE**
- Optimal vs. complete planning
- Planning vs. replanning



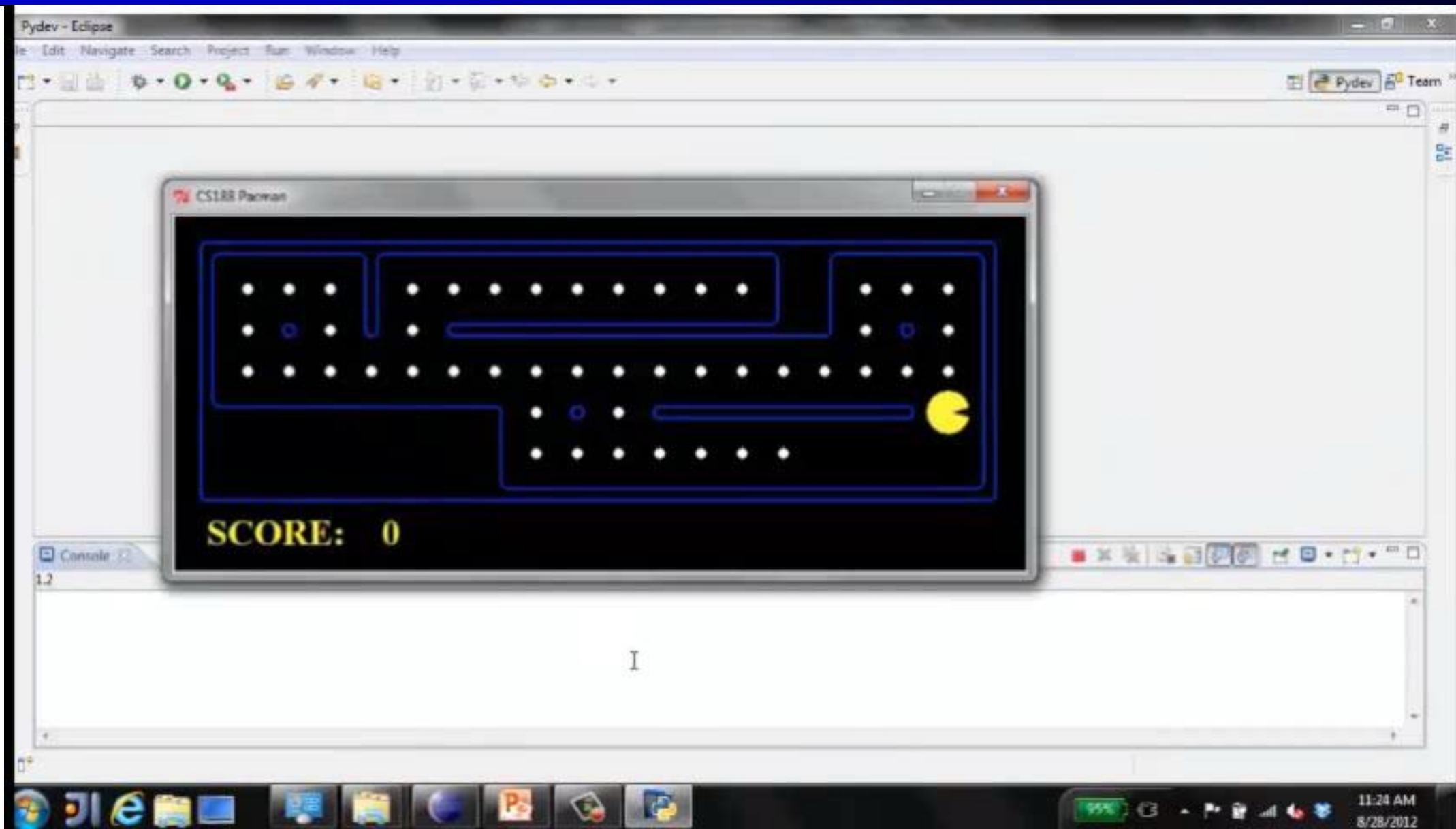
[Demo: re-planning (L2D3)]

[Demo: mastermind (L2D4)]

Video of Demo Mastermind



Video of Demo Replanning



Class Discussion

1. Can you think of real-world examples of planning agents?
2. Can you think of examples where both reflex and planning behaviors are combined in real-world AI systems?

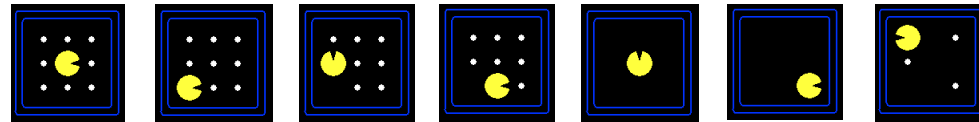
Search Problems



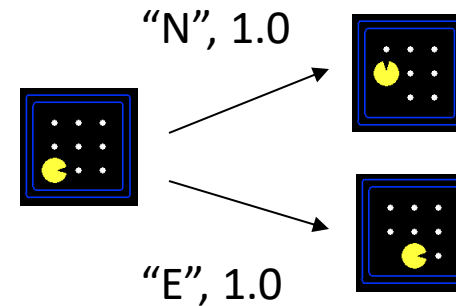
Search Problems

- 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

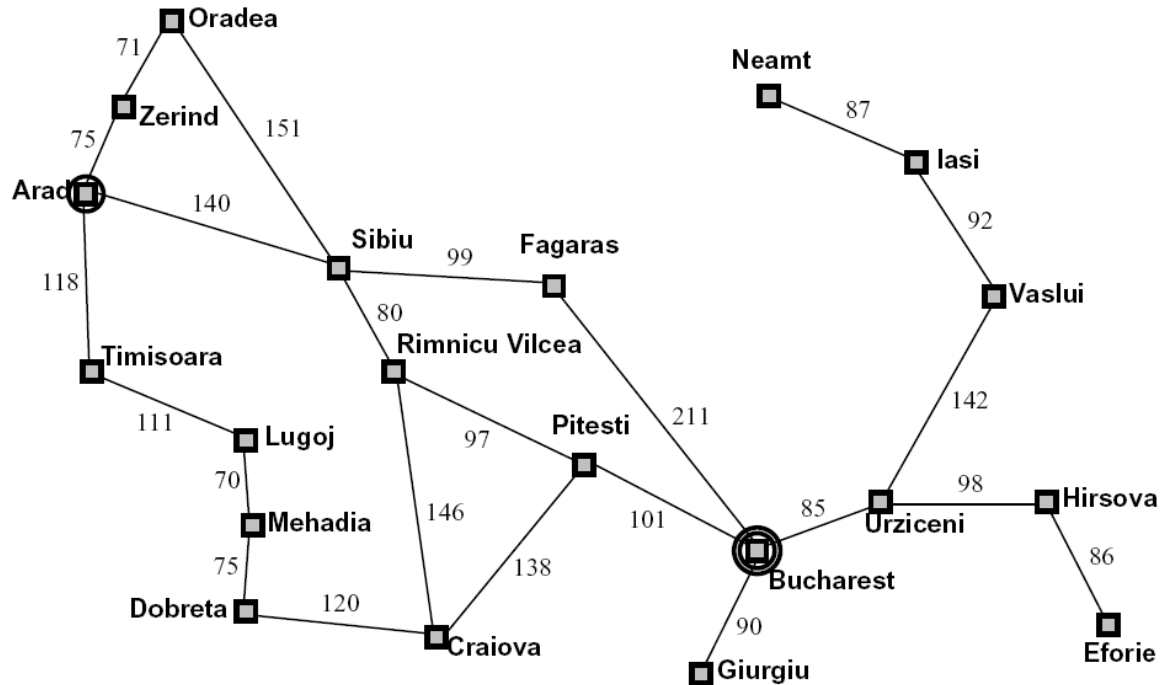
Formalizing Search Problems

- **State Space:** All possible configurations of the problem.
- **Successor Function:** Possible actions from each state.
- **Start State & Goal Test:** Where the search begins and how success is defined.
- **Cost Function:** Measures the cost of a path (if applicable).
- **Example:** Traveling between cities or solving mazes.

Search Problems Are Models



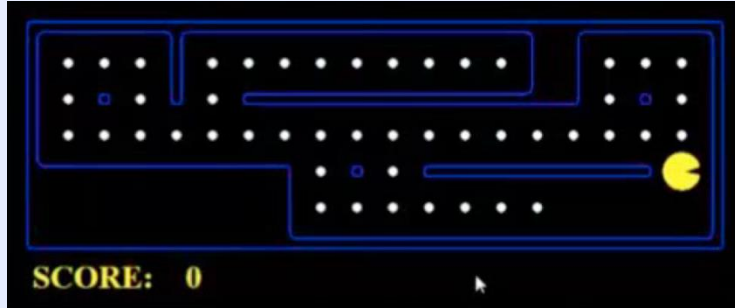
Example: Traveling in Romania



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

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

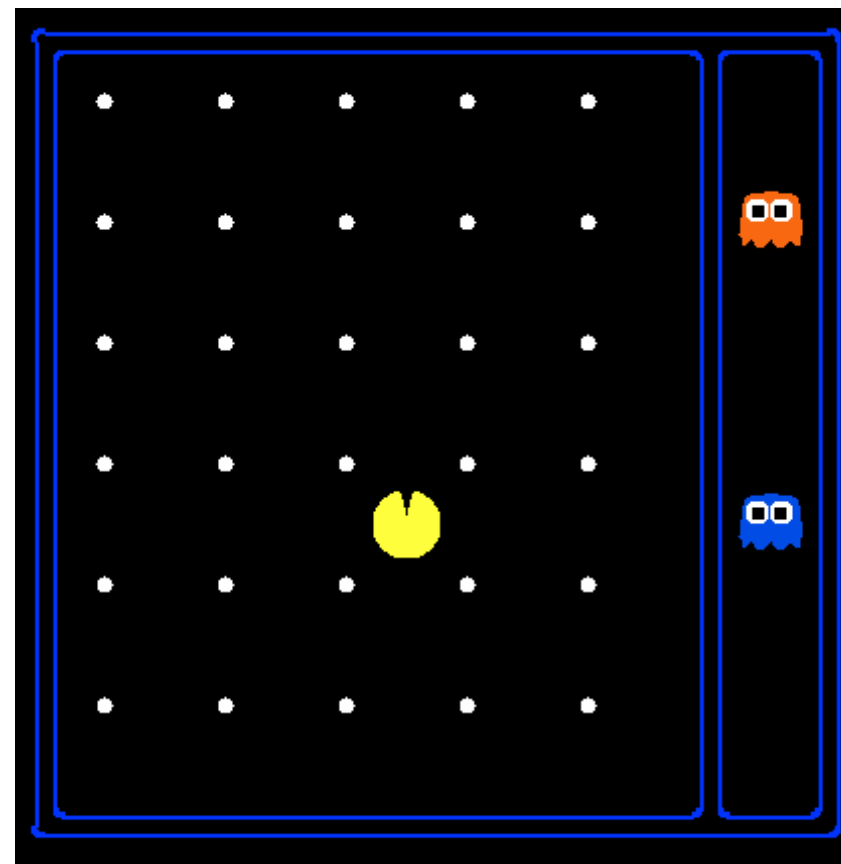
- States: (x,y) location
- Actions: NSEW
- Successor: update location only
- Goal test: is $(x,y)=END$

■ Problem: Eat-All-Dots

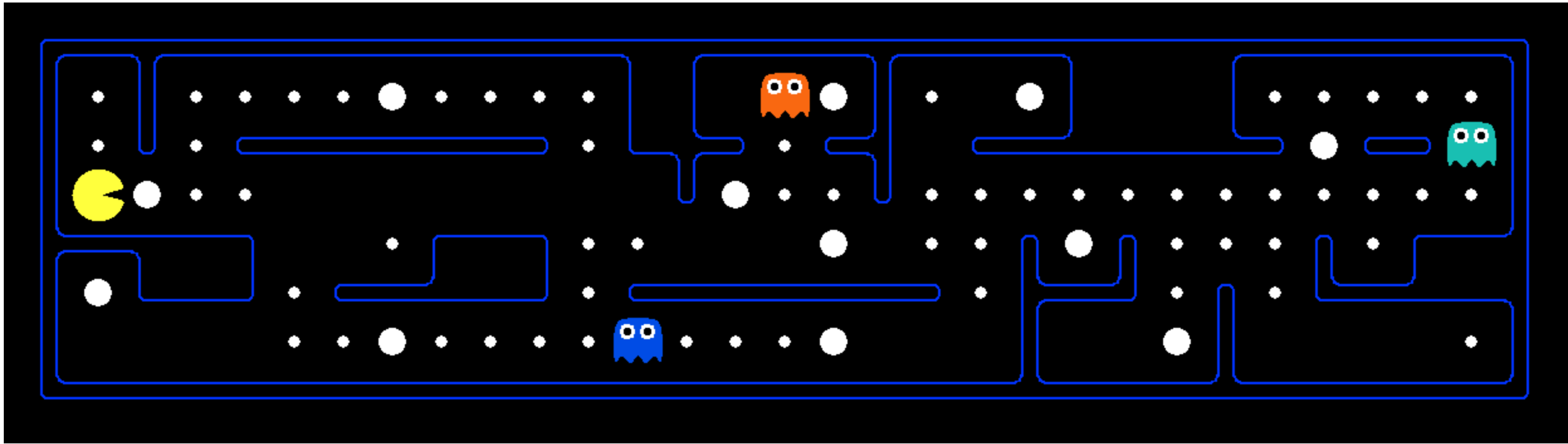
- States: $\{(x,y), \text{dot booleans}\}$
- Actions: NSEW
- Successor: update location and possibly a dot boolean
- Goal test: dots all false

State Space Sizes?

- World state:
 - Agent positions: 120
 - Food count: 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-dots?
 $120 \times (2^{30})$

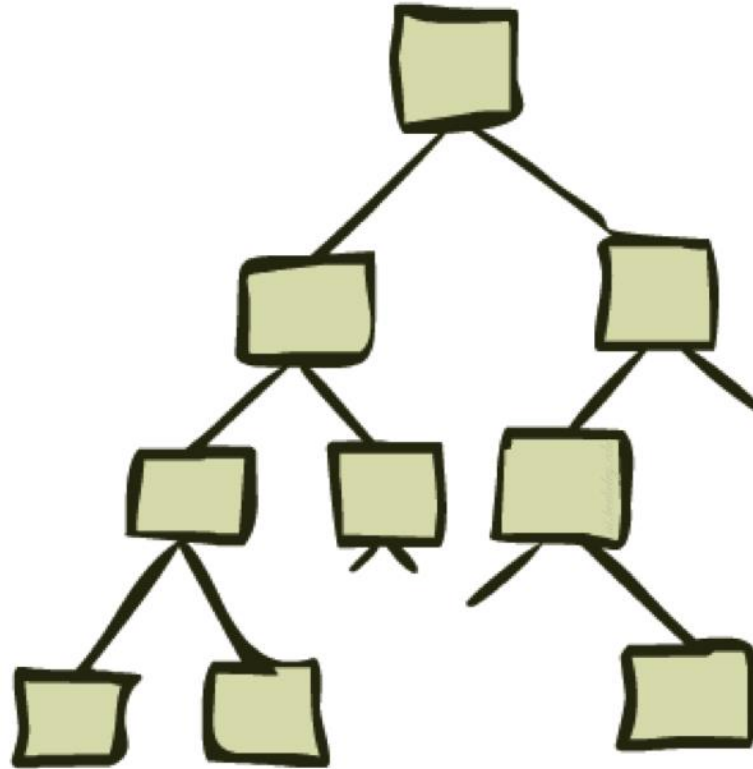


Quiz: Safe Passage



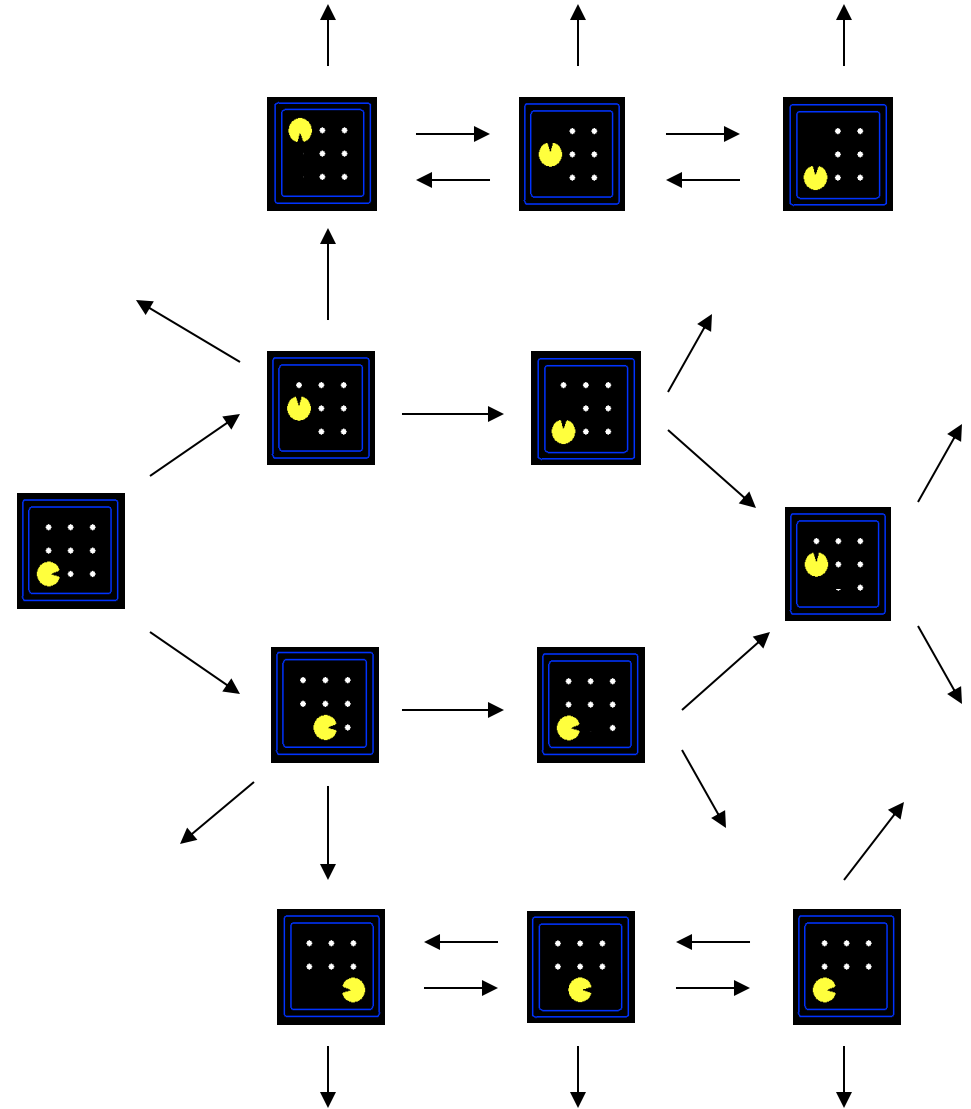
- Problem: eat all dots while keeping the ghosts perma-scared
- What does the state space have to specify?
 - (agent position, dot booleans, power pellet booleans, remaining scared time)

State Space Graphs and Search Trees



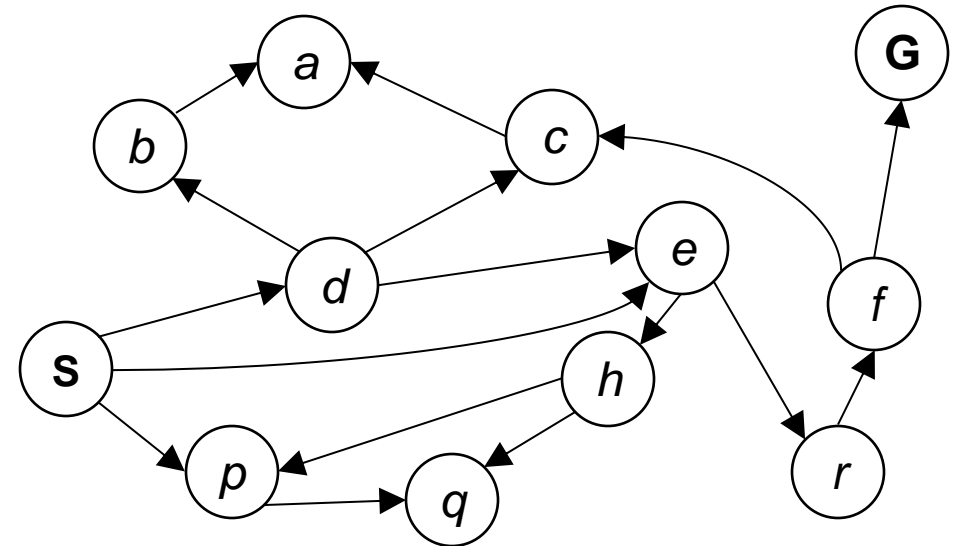
State Space Graphs

- 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



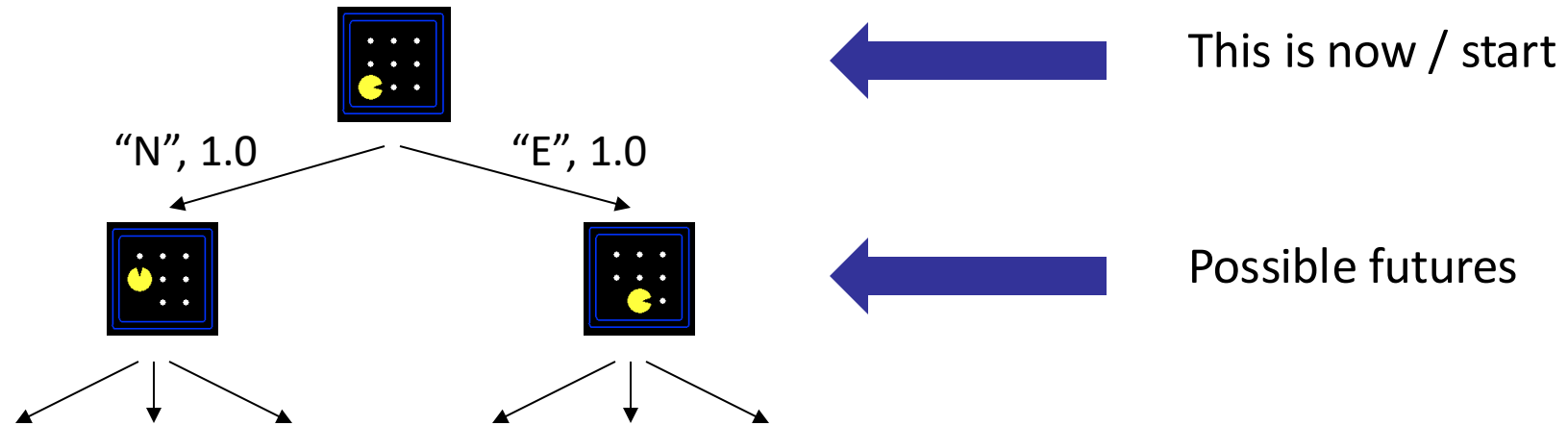
State Space Graphs

- 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



Tiny state space graph for a tiny search problem

Search Trees

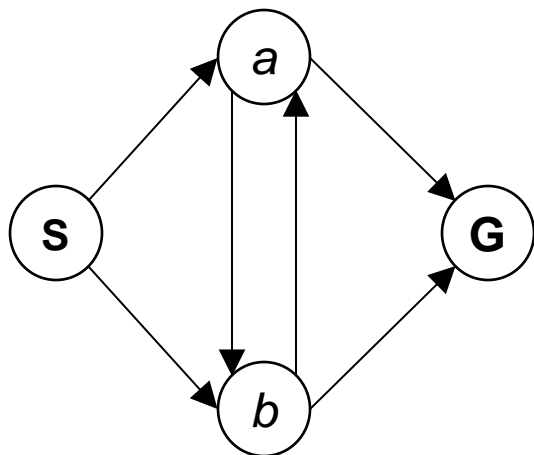


- A search tree:

- 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

Quiz: State Space Graphs vs. Search Trees

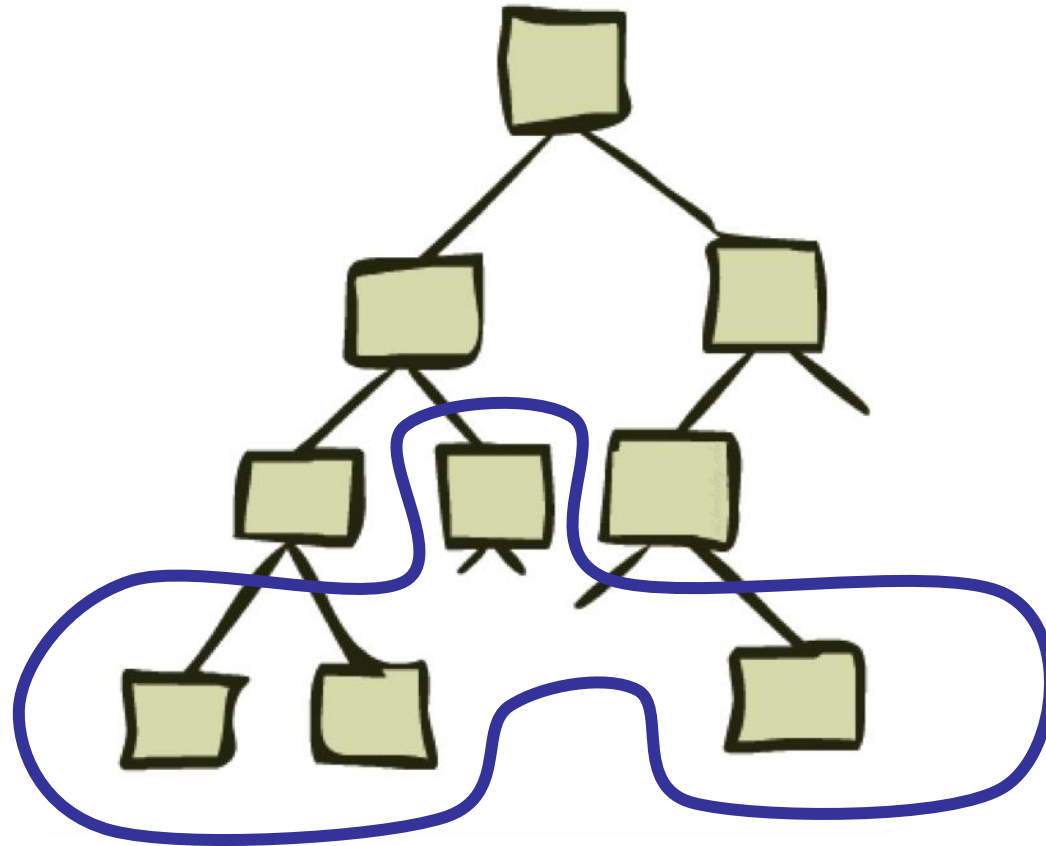
Consider this 4-state graph:



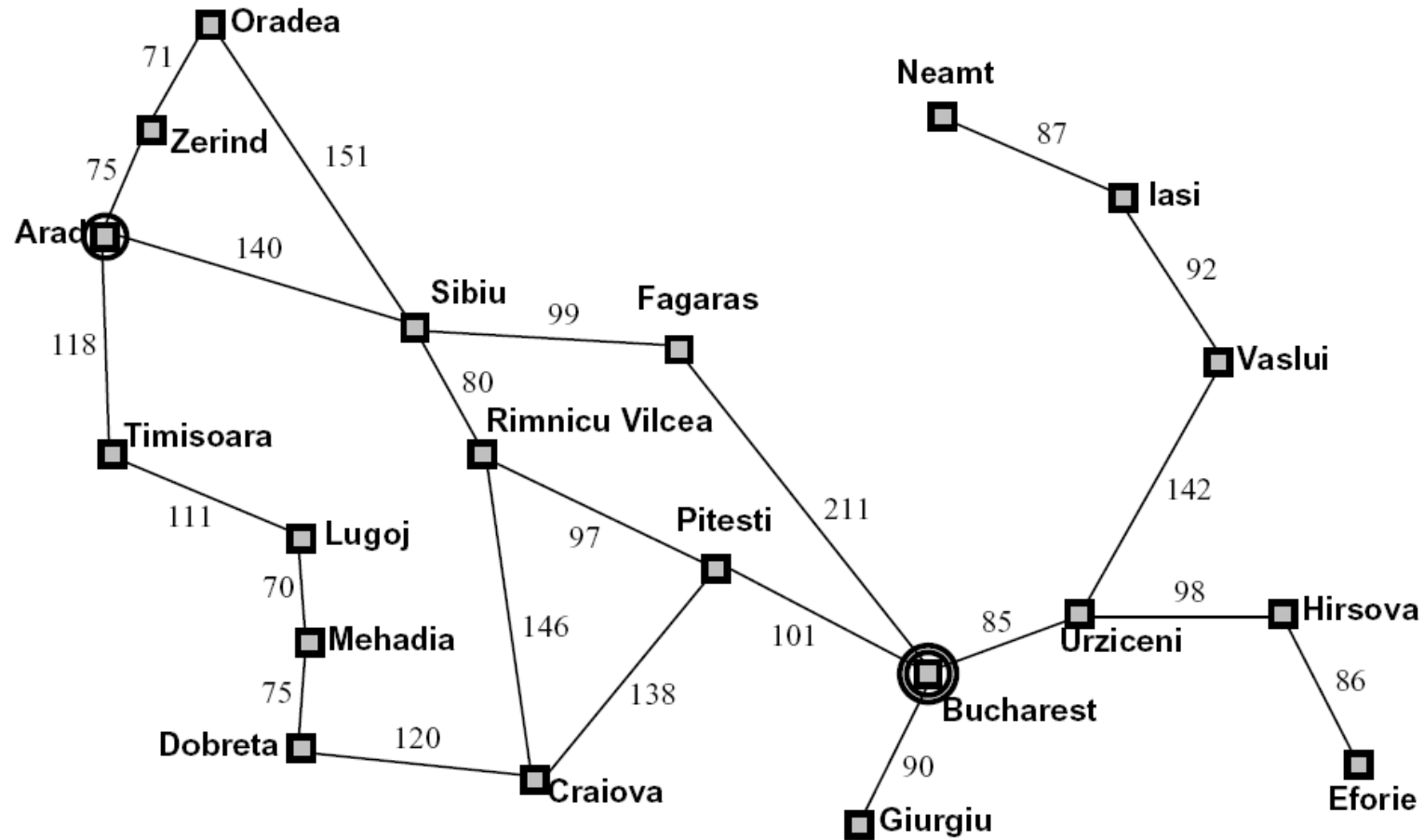
How big is its search tree (from S)?



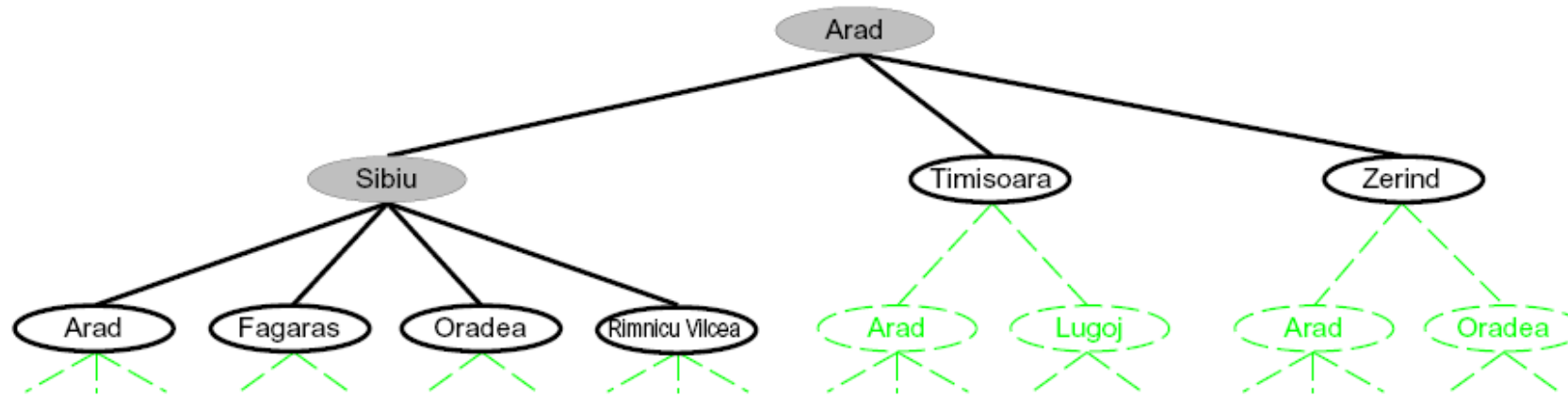
Tree Search



Search Example: Romania



Searching with a Search Tree



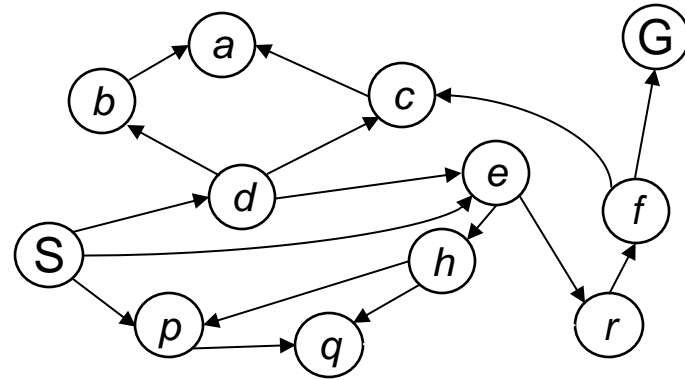
- Search:
 - Expand out potential plans (tree nodes)
 - Maintain a **fringe** of partial plans under consideration
 - Try to expand as few tree nodes as possible

General Tree Search

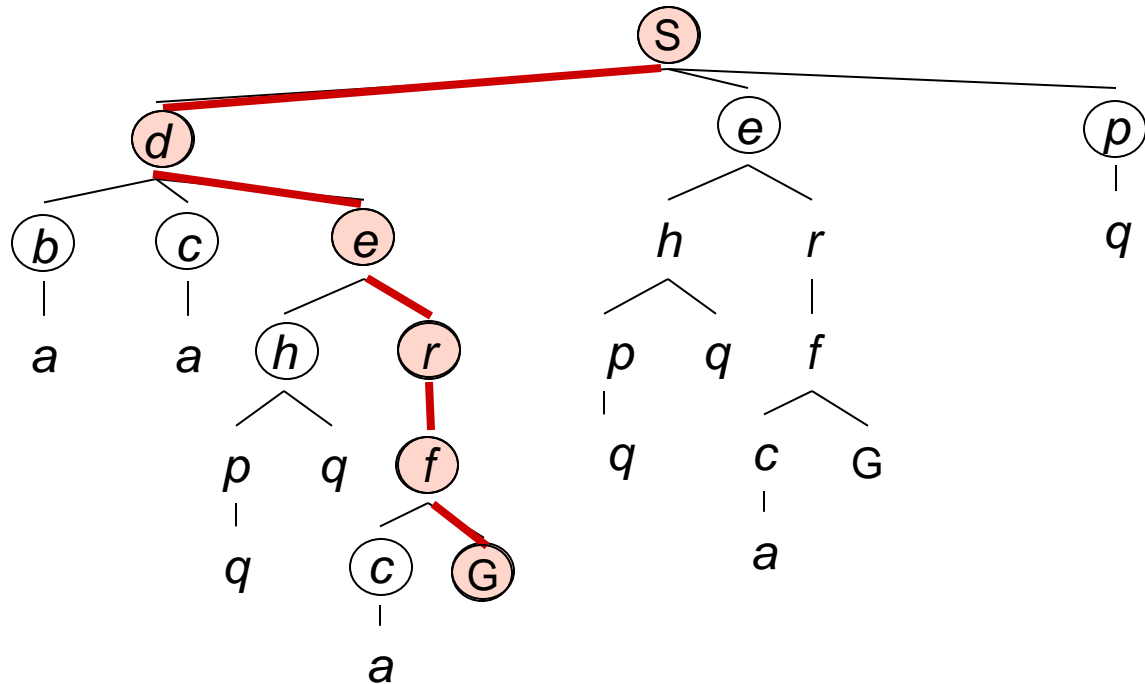
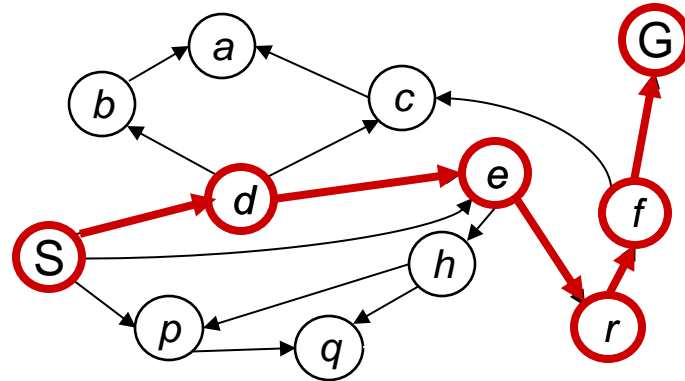
```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal 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



Example: Tree Search



- ~~s~~
- ~~s → d~~
- s → e
- s → p
- s → d → b
- s → d → c
- ~~s → d → e~~
- s → d → e → h
- ~~s → d → e → r~~
- ~~s → d → e → r → f~~
- s → d → e → r → f → c
- ~~s → d → e → r → f → G~~

Depth-First Search



Depth-First Search (DFS) in Action

- **How It Works:**

- Explores as far as possible along a branch before backtracking.

- **Advantages:**

- Low memory usage; good for deep solutions.

- **Disadvantages:**

- May get stuck in infinite loops; not optimal.

- **Visualization:**

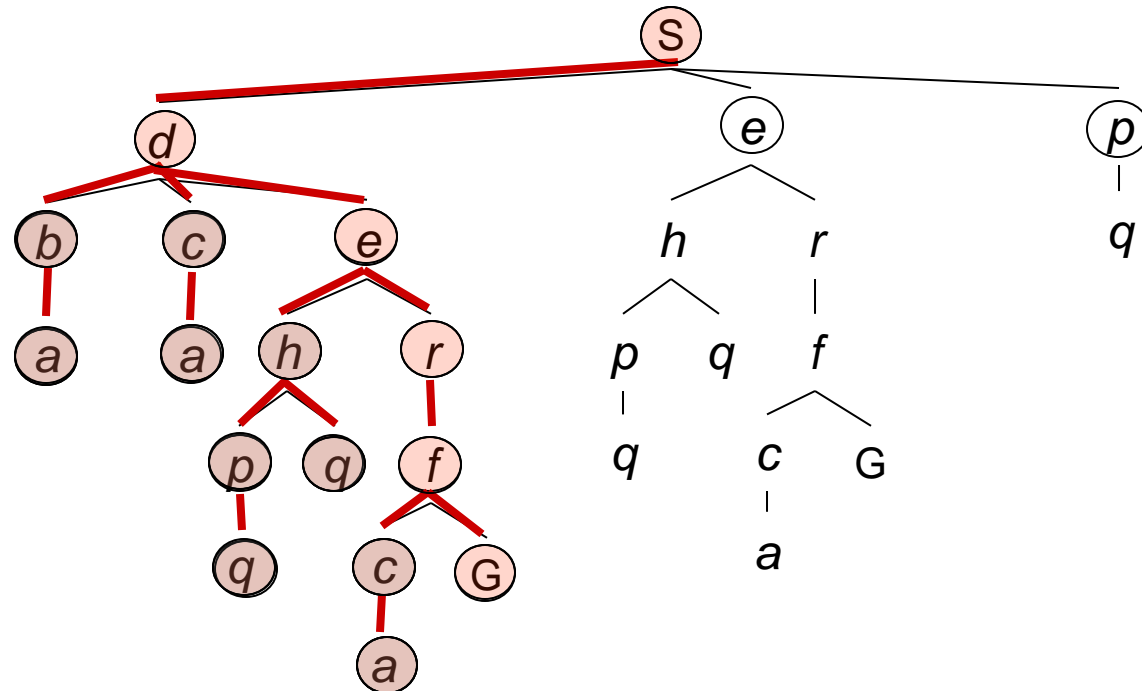
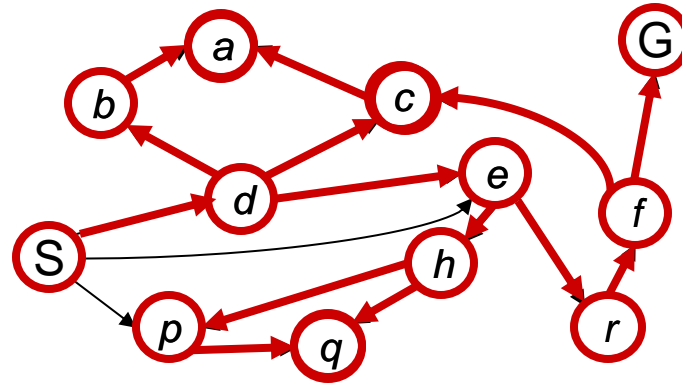
- Simple tree traversal showing DFS paths.

Depth-First Search

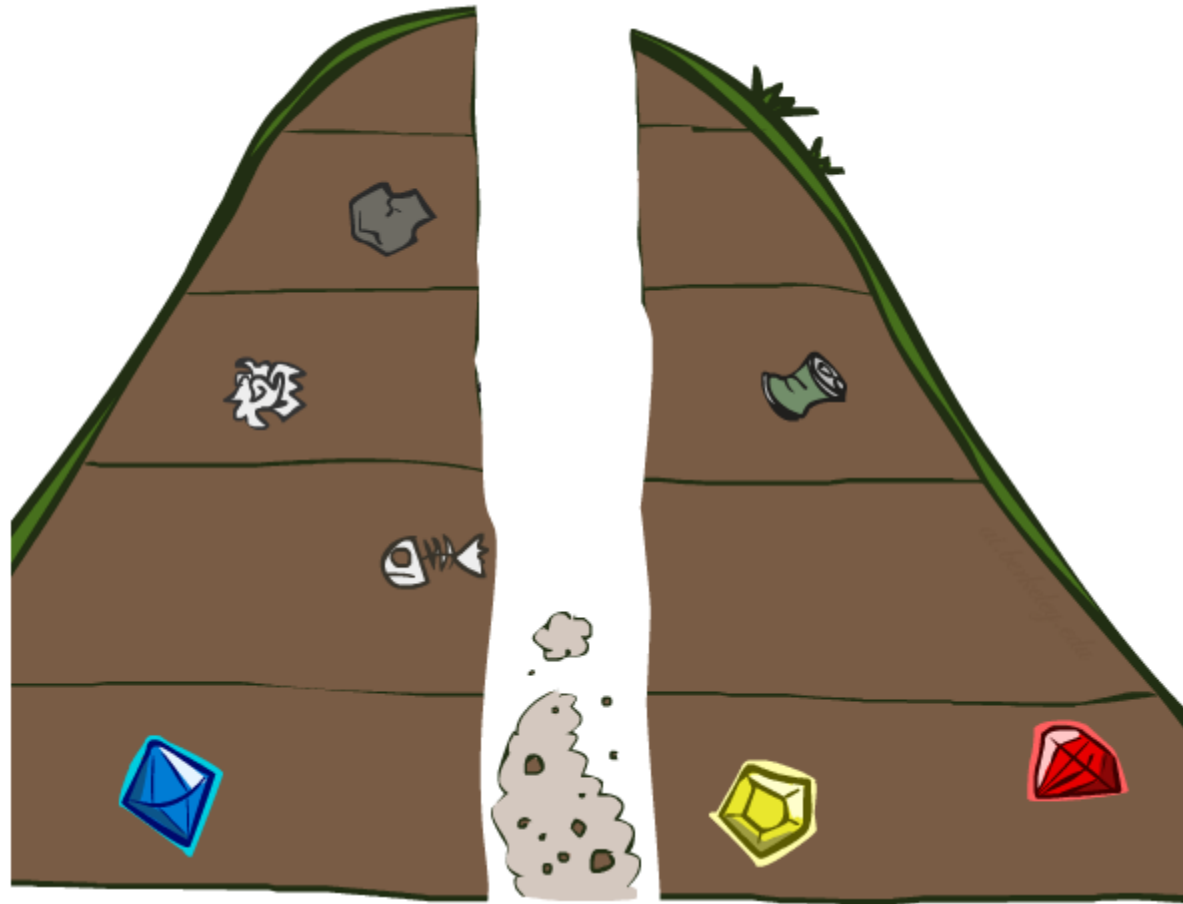
Strategy: expand a deepest node first

Implementation:

Fringe is a LIFO stack

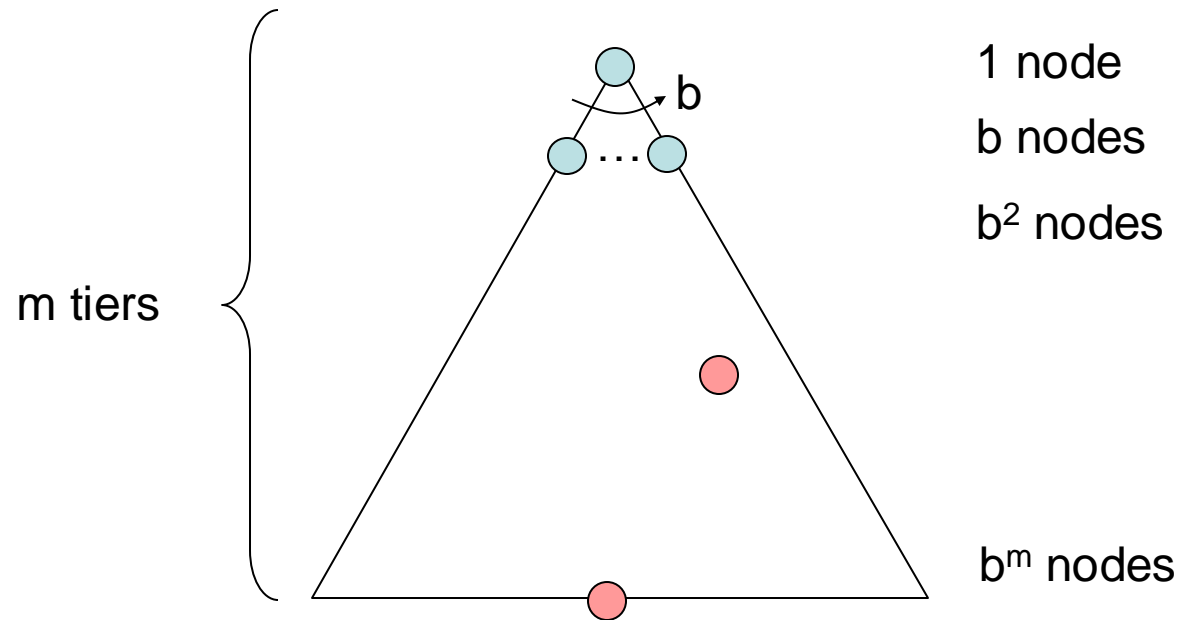


Search Algorithm Properties



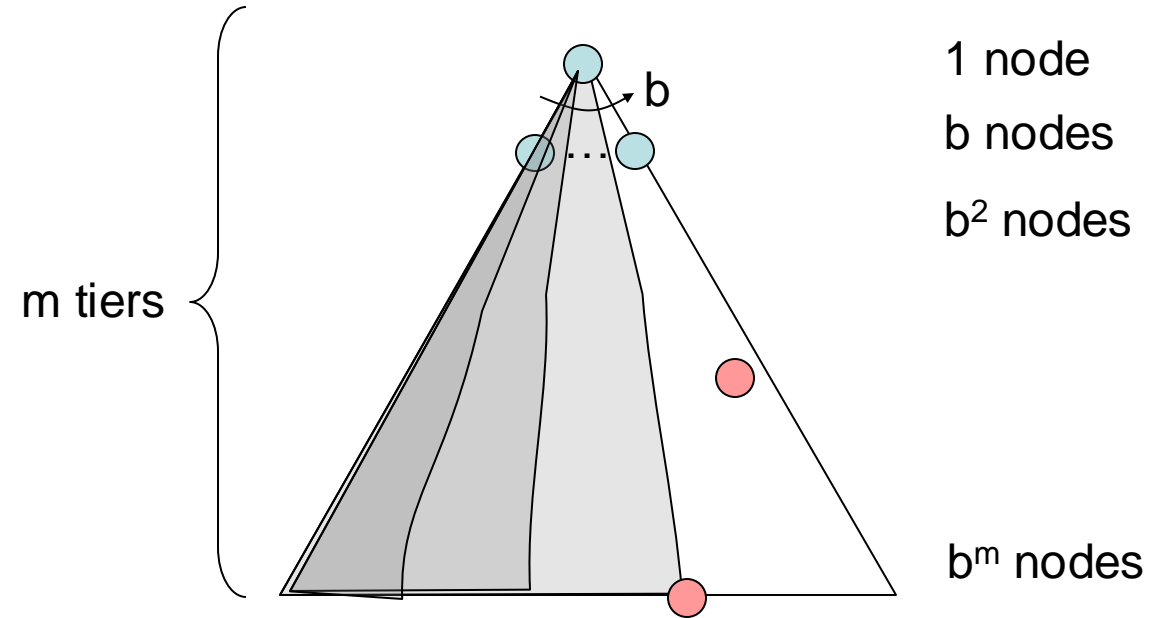
Search Algorithm Properties

- 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?
 - $1 + b + b^2 + \dots + b^m = O(b^m)$

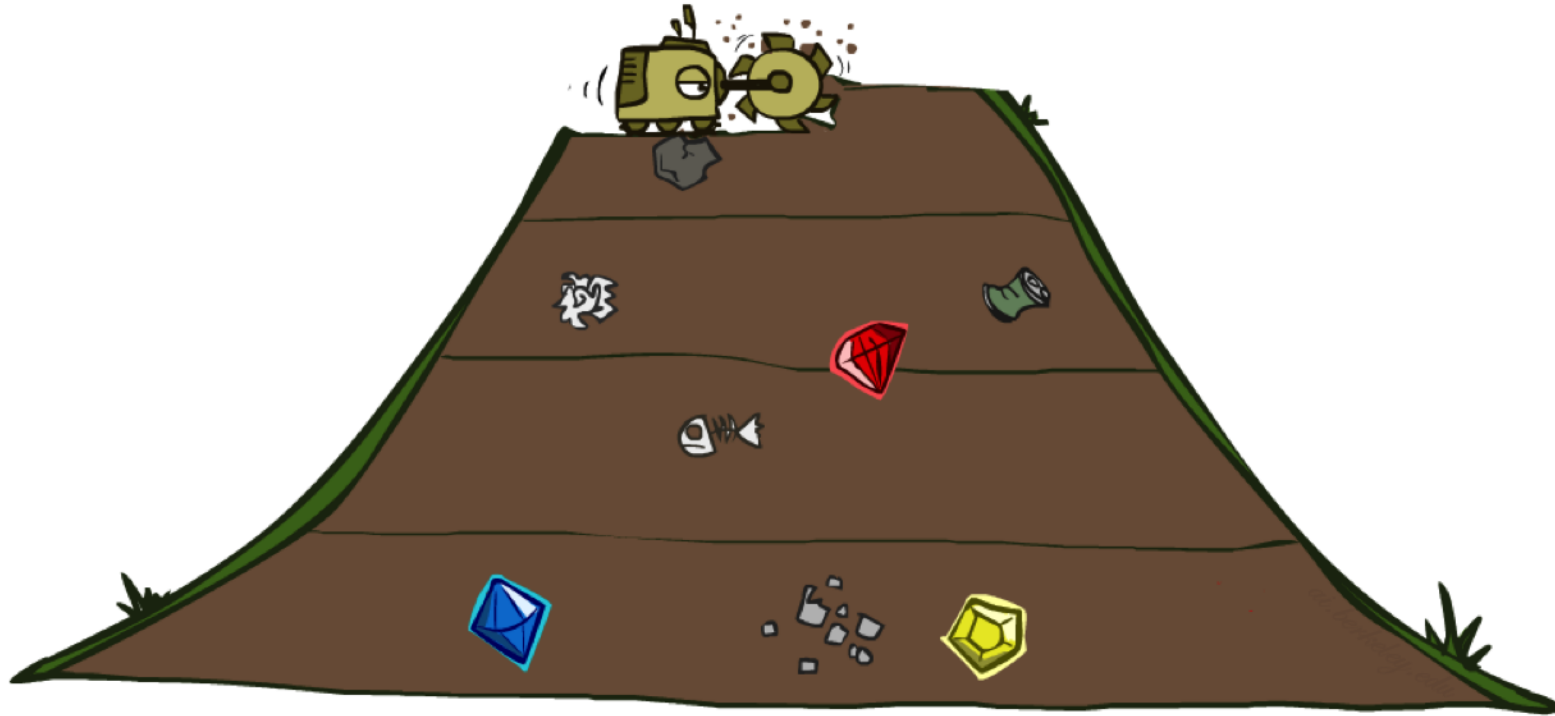


Depth-First Search (DFS) Properties

- 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



Breadth-First Search



Breadth-First Search (BFS) in Action

- **How It Works:**

- Explores all nodes at the current depth before moving deeper.

- **Advantages:**

- Finds the shortest path; complete.

- **Disadvantages:**

- High memory consumption.

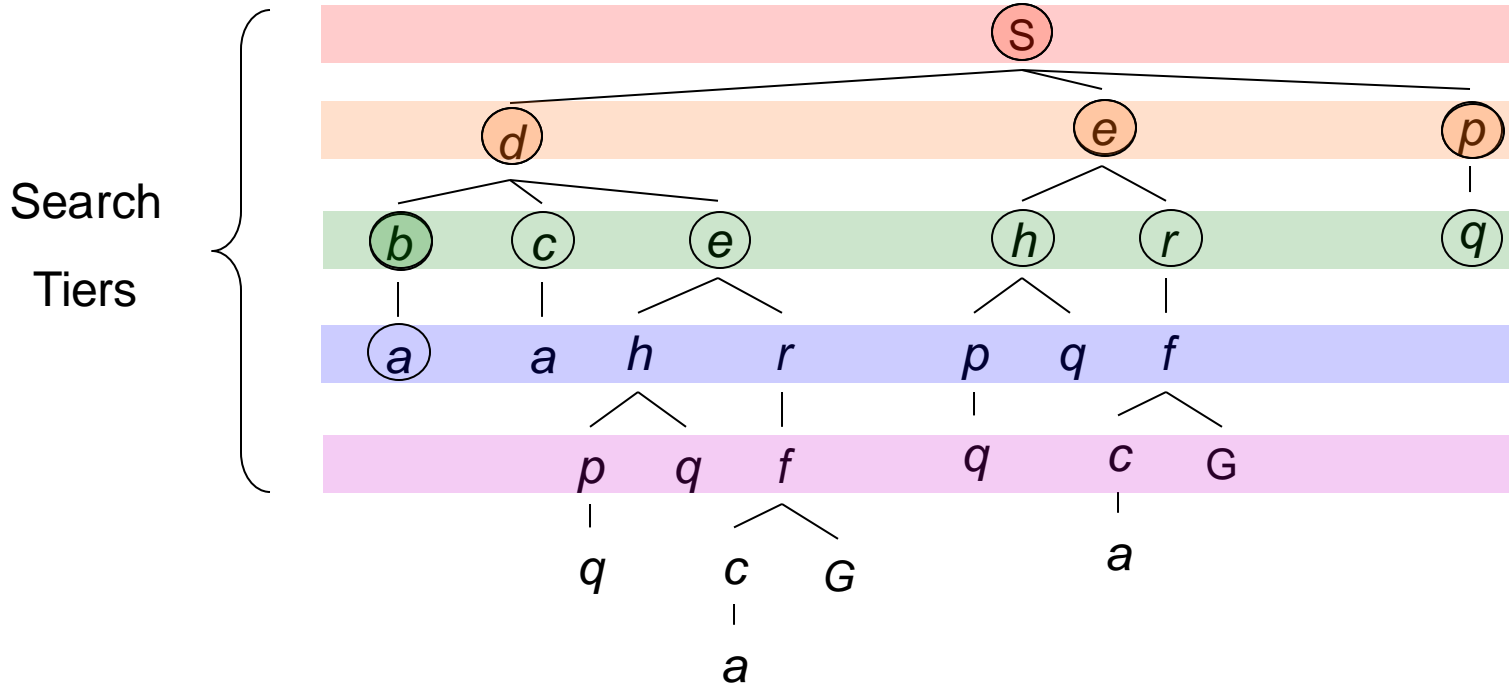
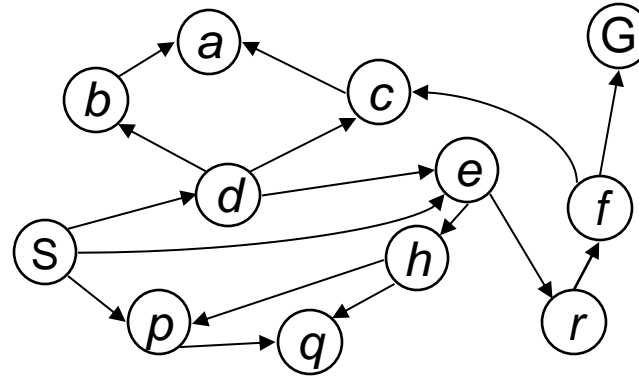
- **Visualization:**

- Compare with DFS using the same problem.

Breadth-First Search

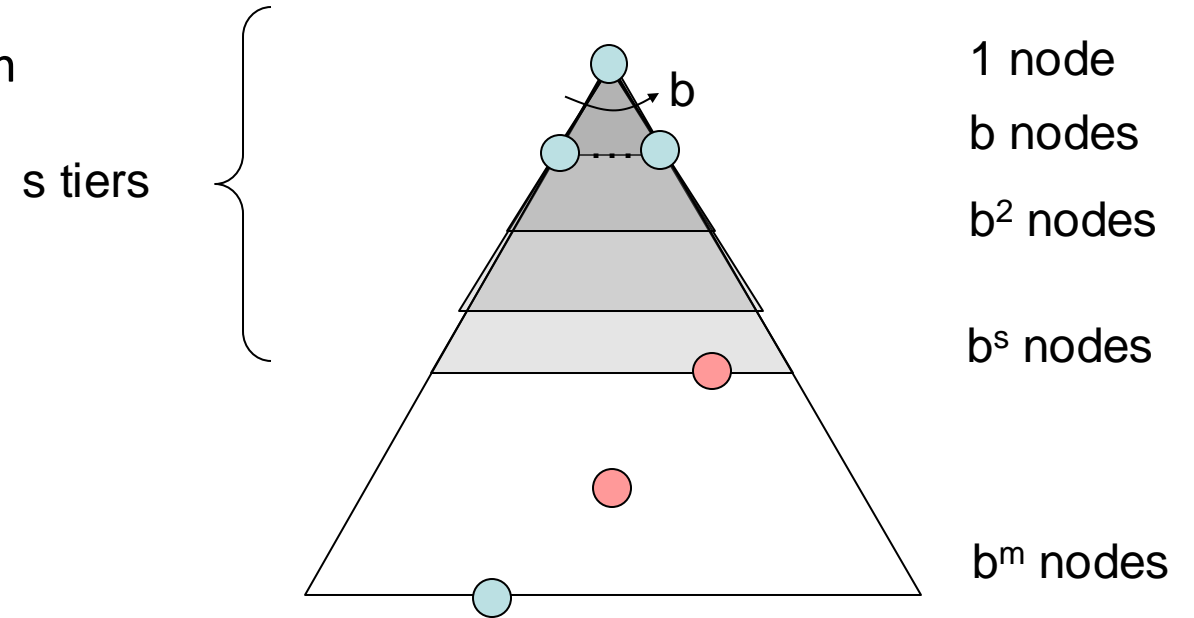
Strategy: expand a shallowest node first

Implementation: Fringe is a FIFO queue

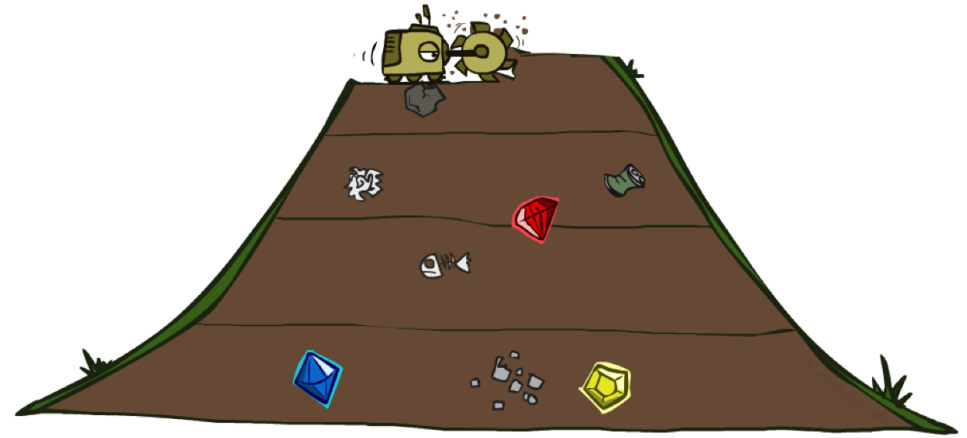


Breadth-First Search (BFS) Properties

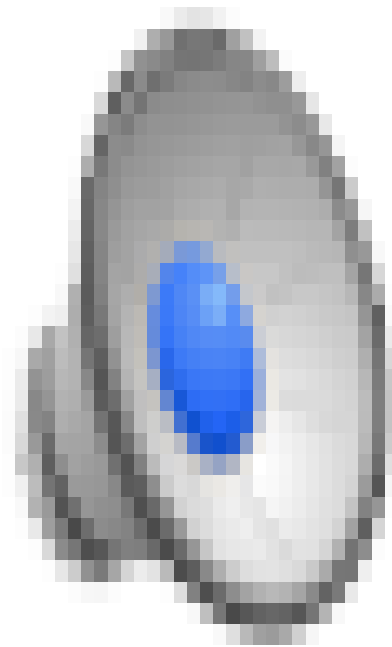
- What nodes does BFS expand?
 - Processes all nodes above shallowest solution
 - Let depth of shallowest solution be s
 - Search takes time $O(b^s)$
- How much space does the fringe take?
 - Has roughly the last tier, so $O(b^s)$
- Is it complete?
 - s must be finite if a solution exists, so yes!
-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



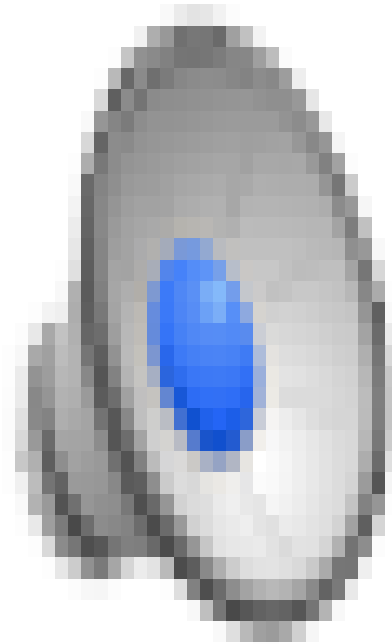
Quiz: DFS vs BFS



Video of Demo Maze Water DFS/BFS (part 1)



Video of Demo Maze Water DFS/BFS (part 2)

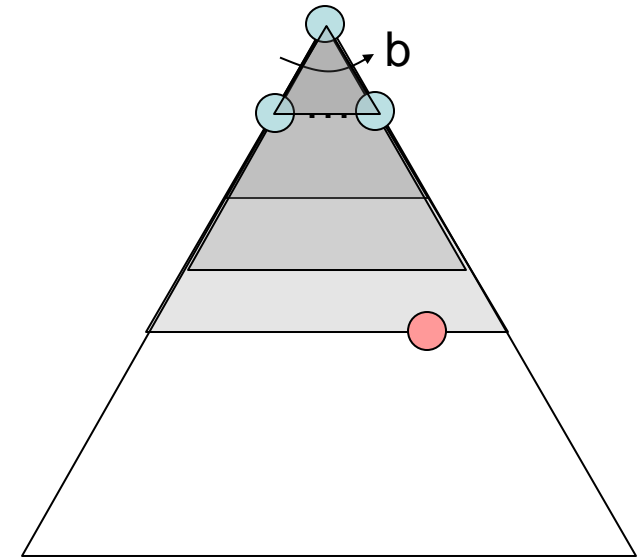


Quiz: DFS vs BFS

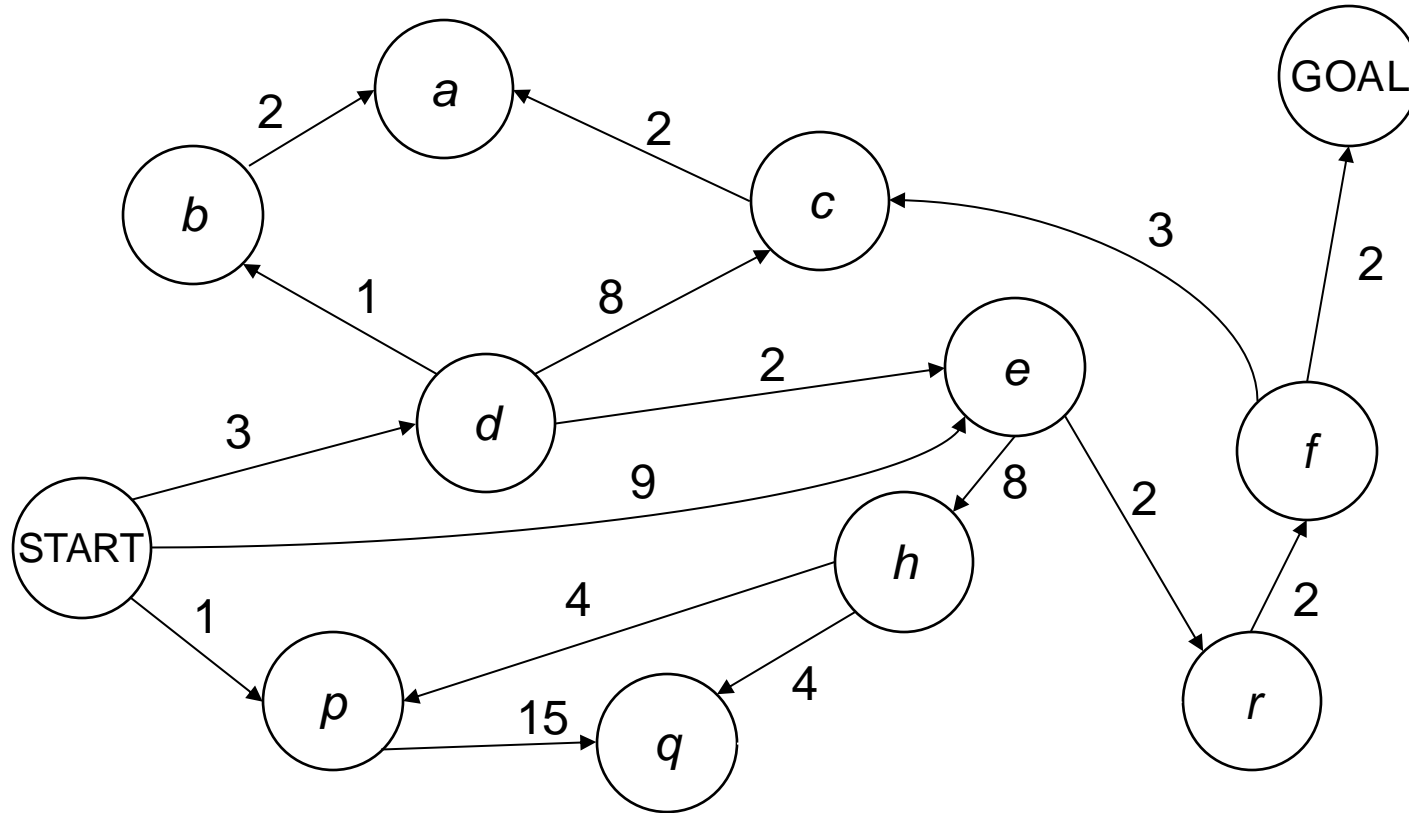
- When will BFS outperform DFS?
- When will DFS outperform BFS?

Iterative Deepening

- 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!

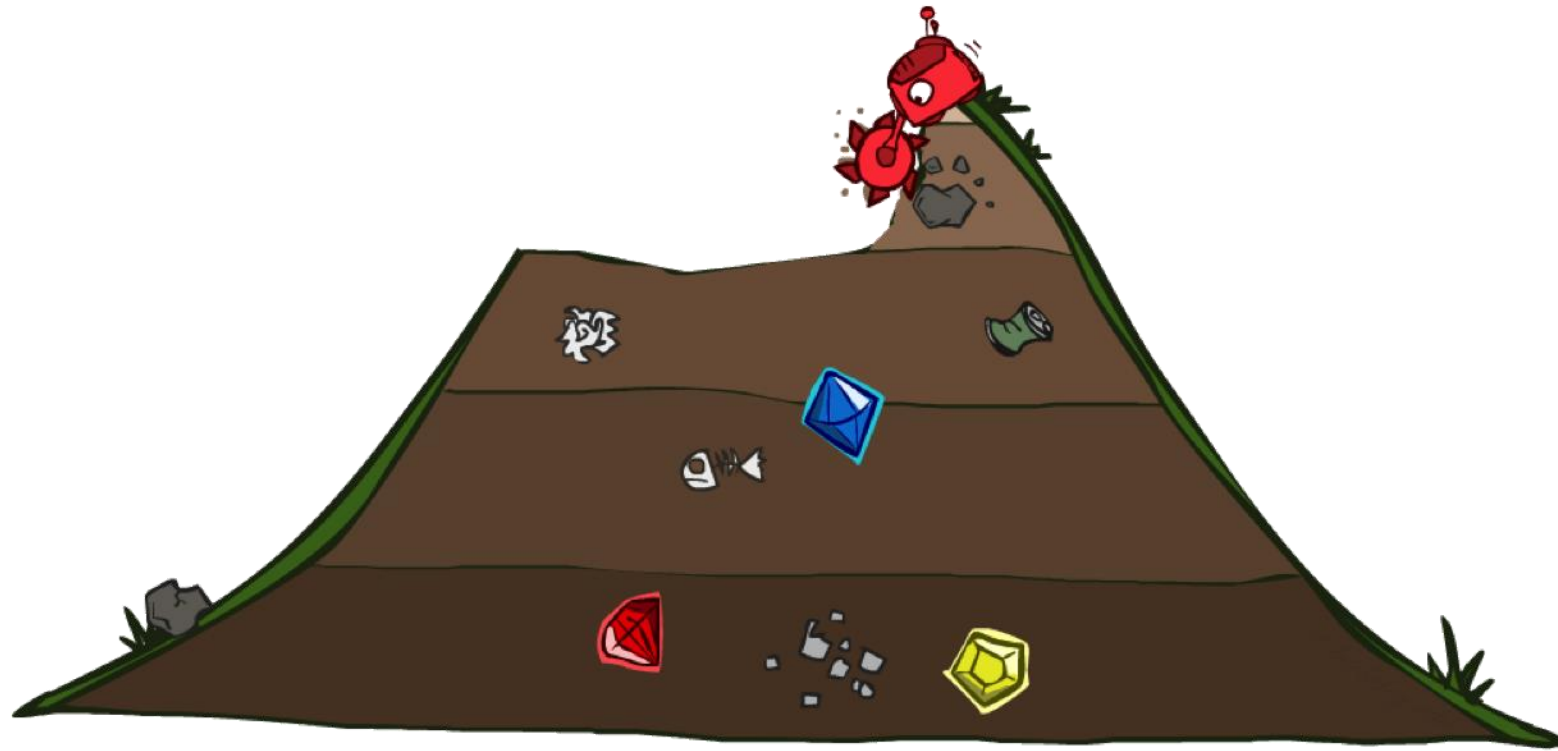


Cost-Sensitive Search



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.

Uniform Cost Search



Uniform-Cost Search (UCS) Explained

- **How It Works:**

- Expands the node with the lowest cumulative cost first using a priority queue.

- **Advantages:**

- Finds least-cost paths; handles varying path costs.

- **Disadvantages:**

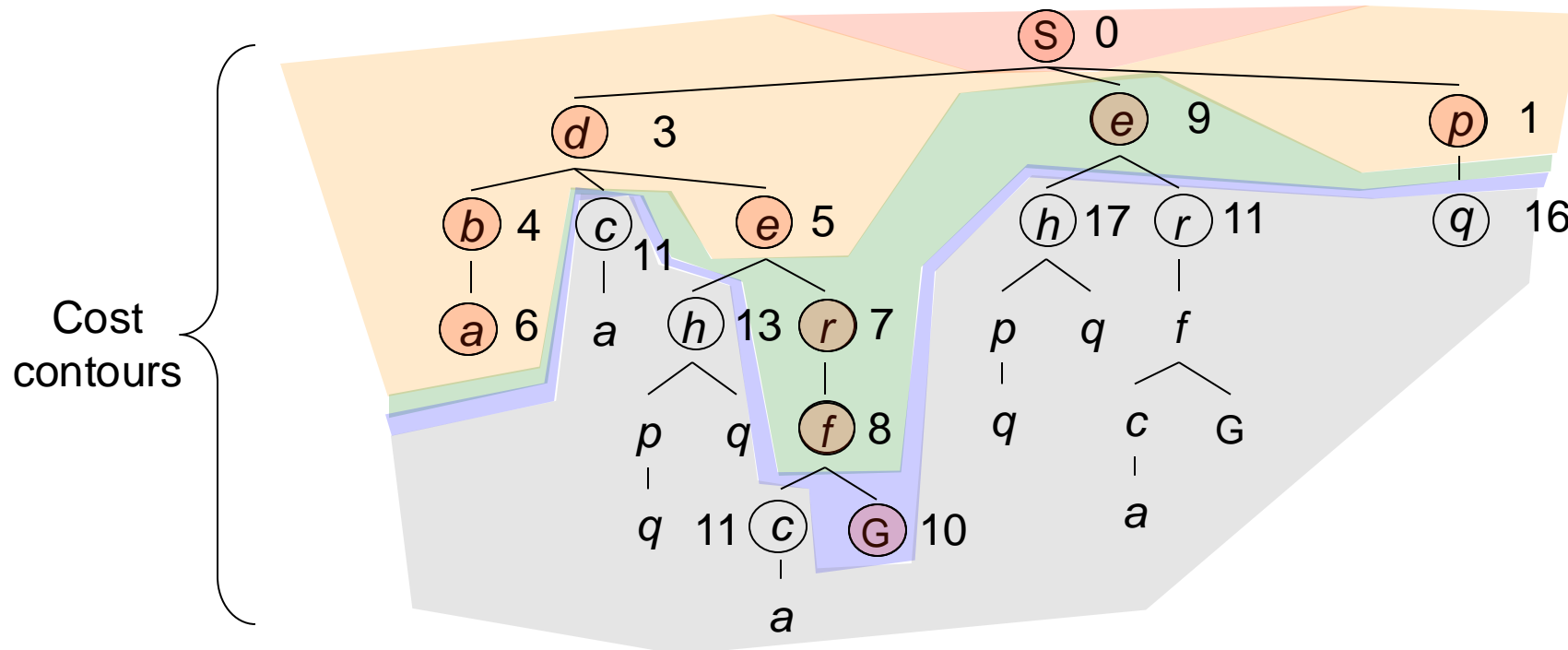
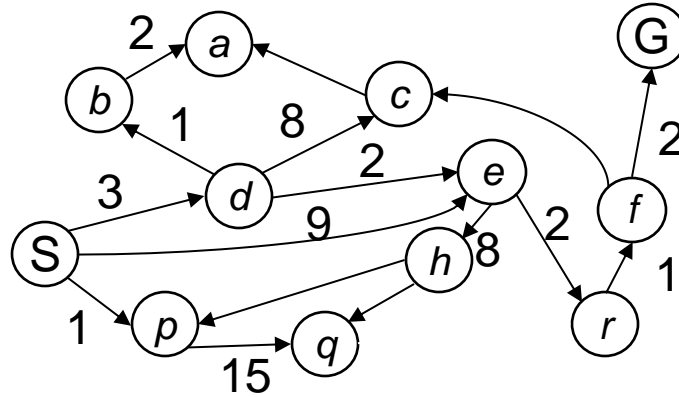
- Can be inefficient if all paths have similar costs.

- **Example:** Finding the cheapest flight across multiple cities.

Uniform Cost Search

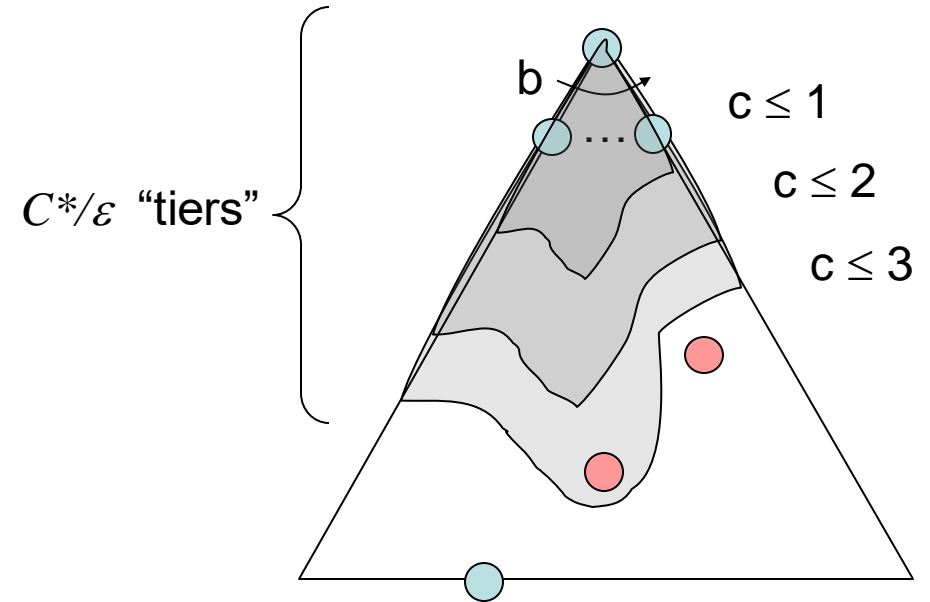
Strategy: expand a
cheapest node first:

Fringe is a priority queue
(priority: cumulative cost)



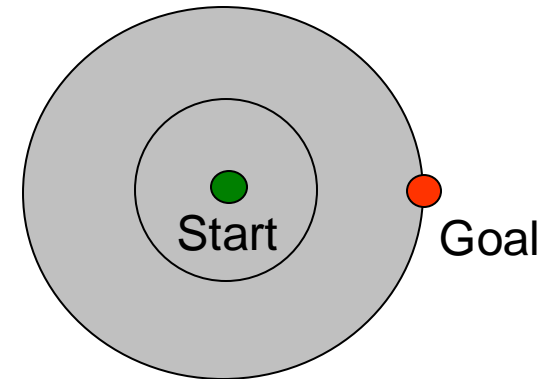
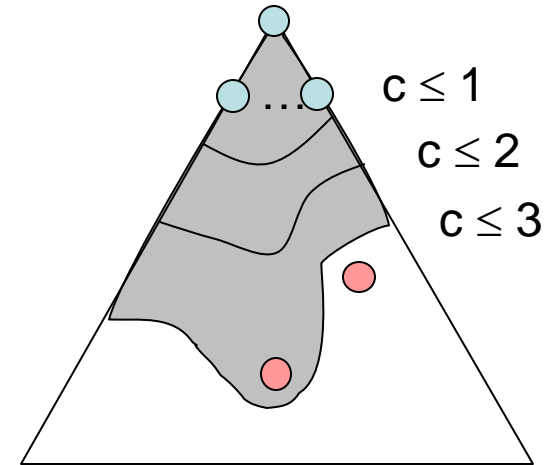
Uniform Cost Search (UCS) Properties

- 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 $O(b^{C^*/\epsilon})$ (exponential in effective depth)
- How much space does the fringe take?
 - Has roughly the last tier, so $O(b^{C^*/\epsilon})$
- 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^*)



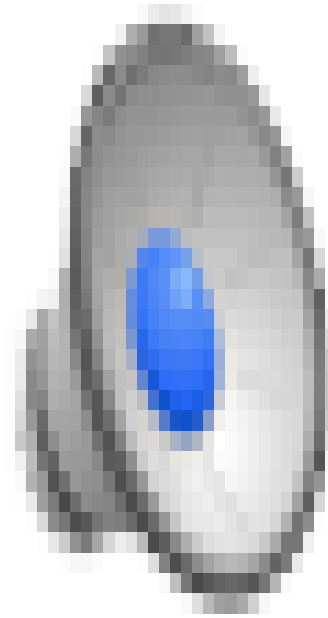
Uniform Cost Issues

- 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!

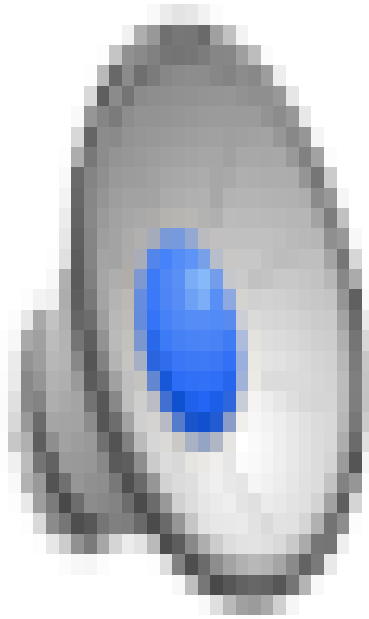


[Demo: empty grid UCS (L2D5)]
[Demo: maze with deep/shallow water DFS/BFS/UCS (L2D7)]

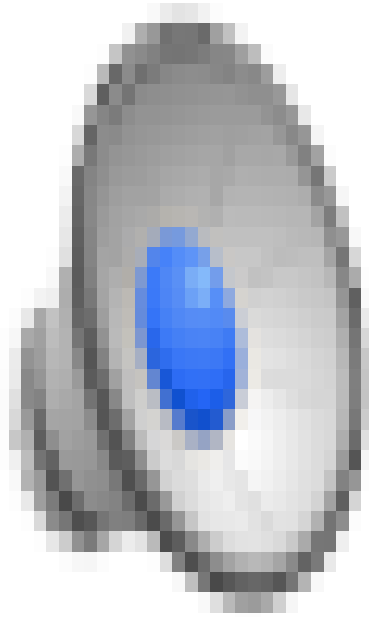
Video of Demo Empty UCS



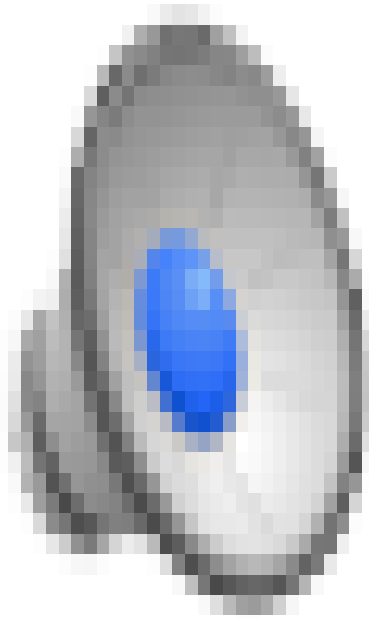
Video of Demo Maze with Deep/Shallow Water --- DFS, BFS, or UCS? (part 2)



Video of Demo Maze with Deep/Shallow Water --- DFS, BFS, or UCS? (part 1)

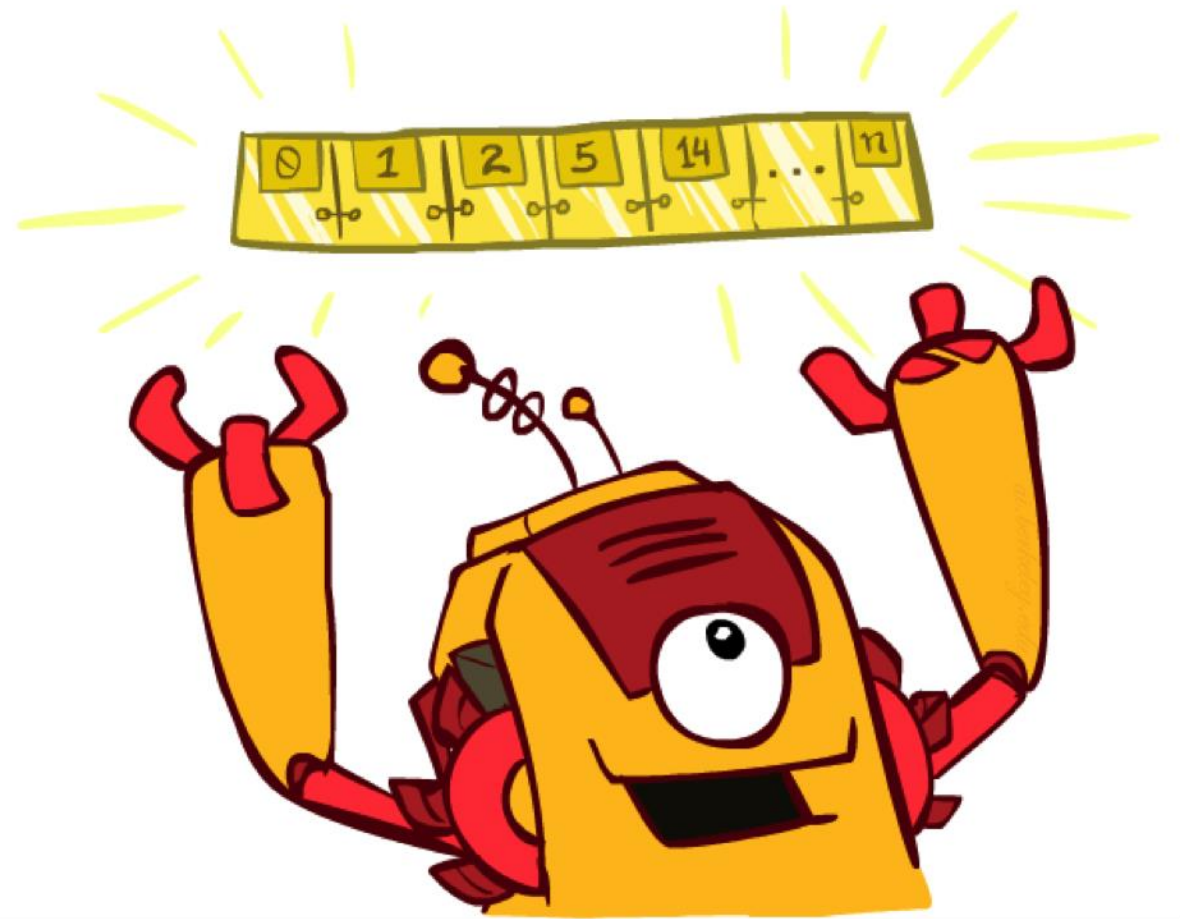


Video of Demo Maze with Deep/Shallow Water --- DFS, BFS, or UCS? (part 3)

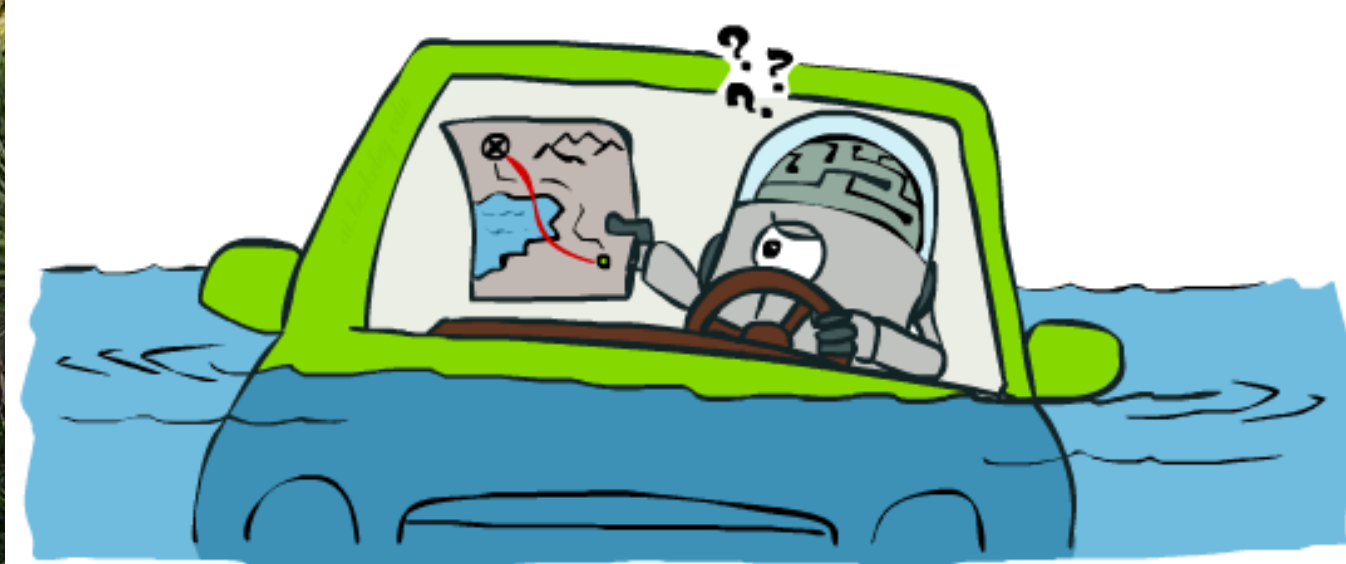
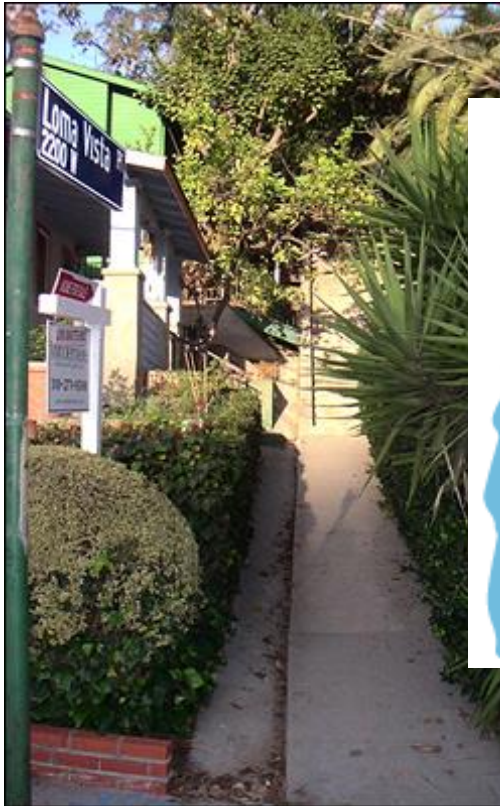


The One Queue

- 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



Search Gone Wrong?



Start: Haugesund, Rogaland, Norway
End: Trondheim, Sør-Trøndelag, Norway
Total Distance: 2713.2 Kilometers
Estimated Total Time: 47 hours, 31 minutes

nrk.no/alltidmoro

Search and Models

- Search operates over models of the world
 - The agent doesn't actually try all the plans out in the real world!
 - Planning is all “in simulation”
 - Your search is only as good as your models...

