# CIS 471/571(Fall 2023): Introduction to Artificial Intelligence

# Lecture 2: Uninformed Search

Thanh H. Nguyen

Most slides are by Pieter Abbeel, Dan Klein, Luke Zettlemoyer, John DeNero, Stuart Russell, Andrew Moore, or Daniel Lowd
Source: http://ai.berkeley.edu/home.html

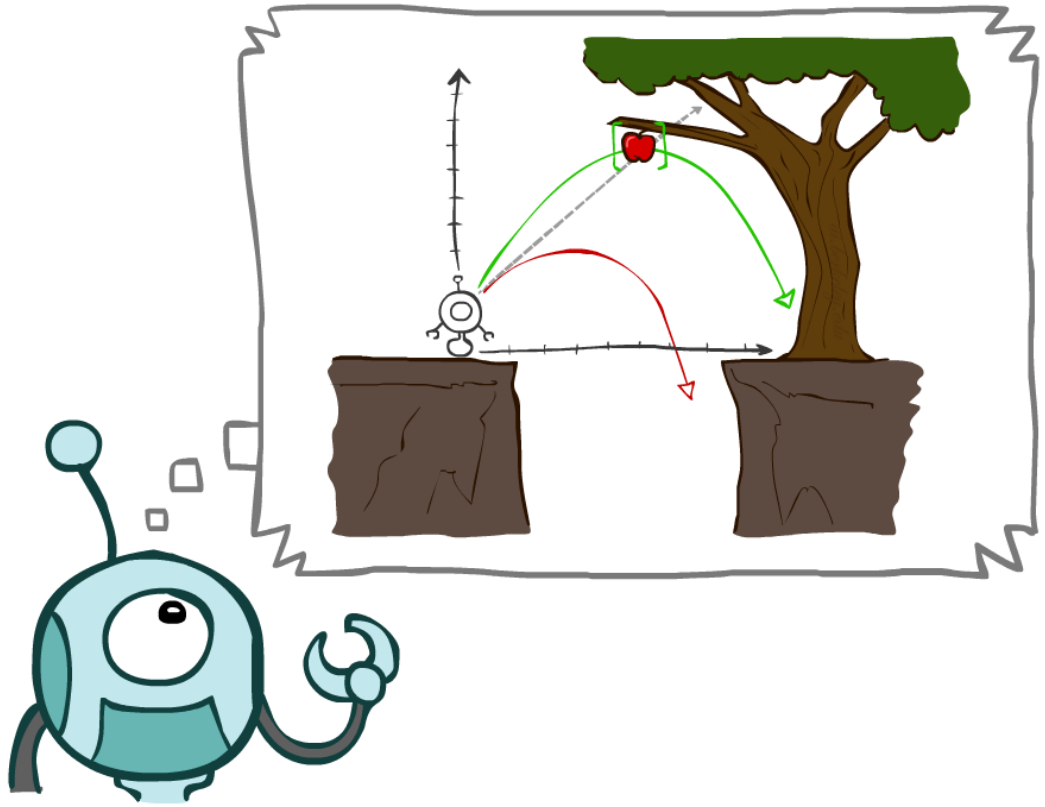
# Announcement

- Project 1
  - Deadline: Oct 16th, 2023

- Written Assignment 1
  - Will be posted tomorrow
  - Deadline: Oct 11th, 2023

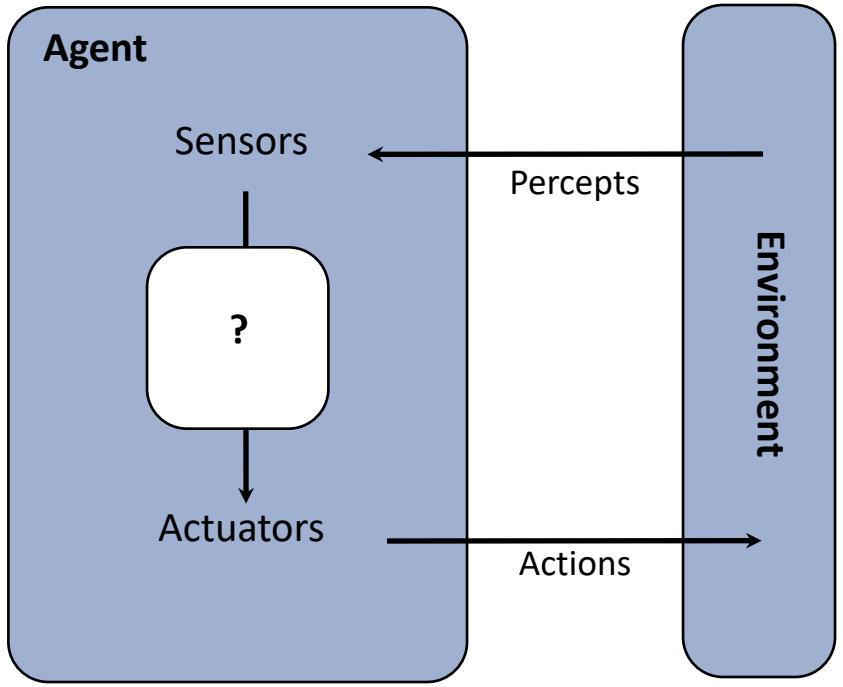
# Today

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



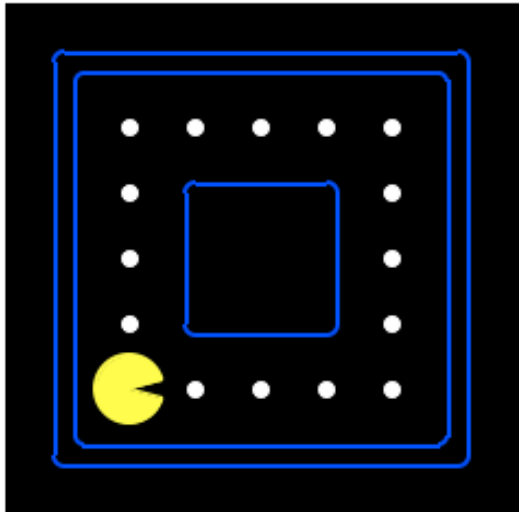
# Rational Agents

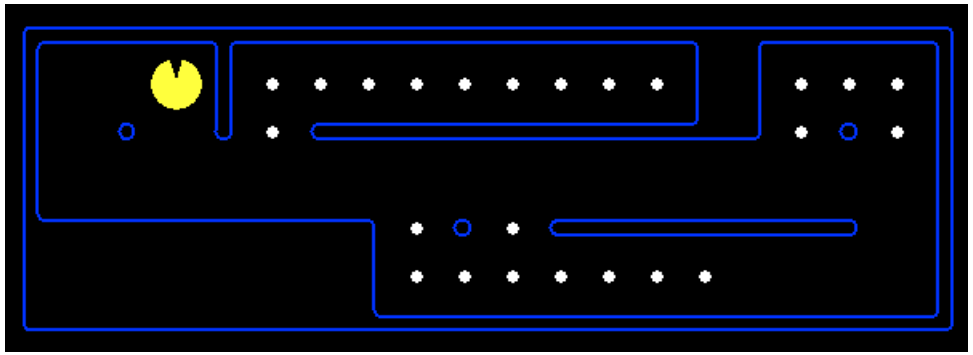
- An **agent** is an entity that *perceives* and *acts*.
- A **rational agent** selects actions that maximize its **utility function**.
- Characteristics of the **percepts, environment,** and **action space** dictate techniques for selecting rational actions.



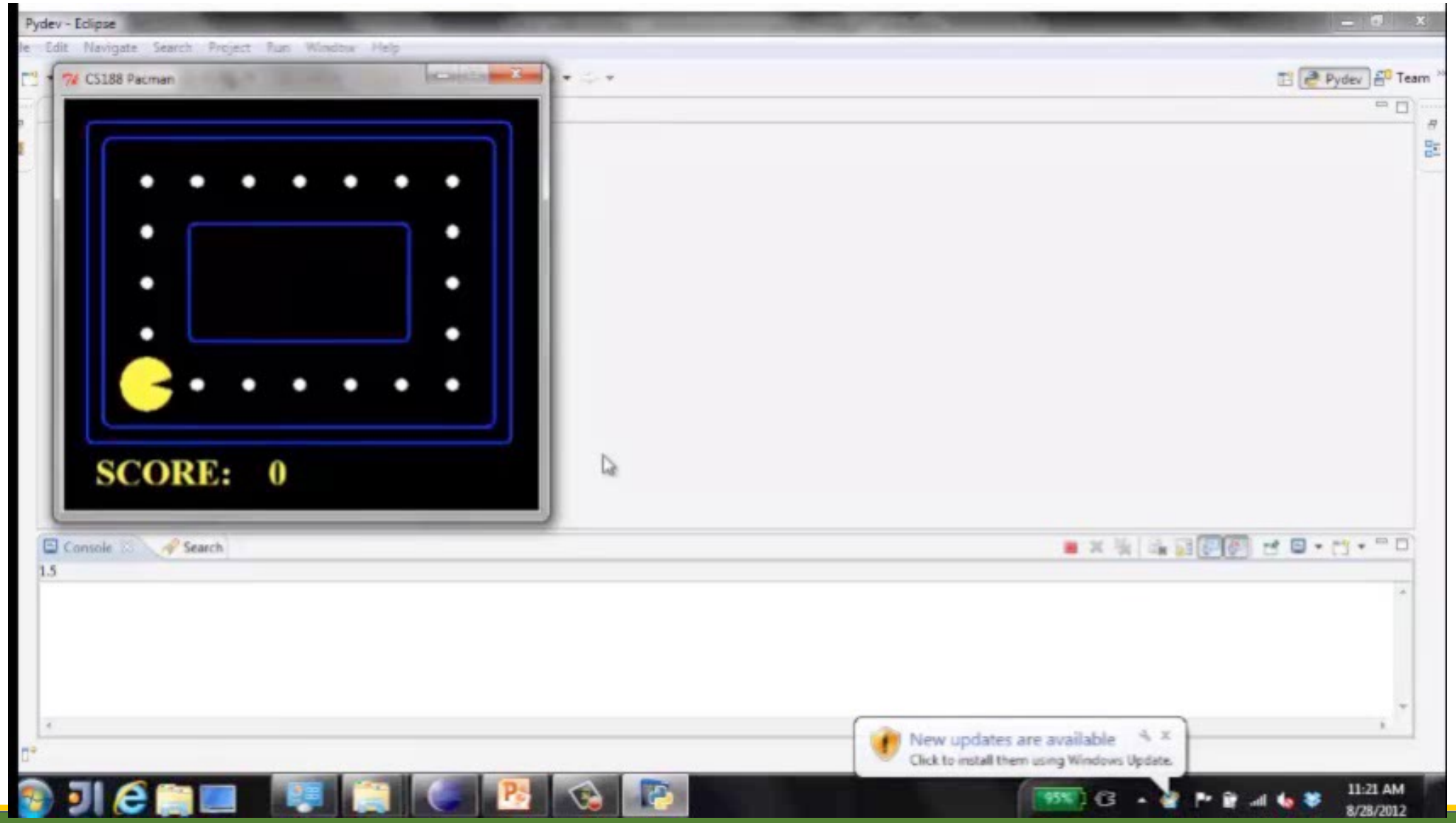
# Reflex Agents

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

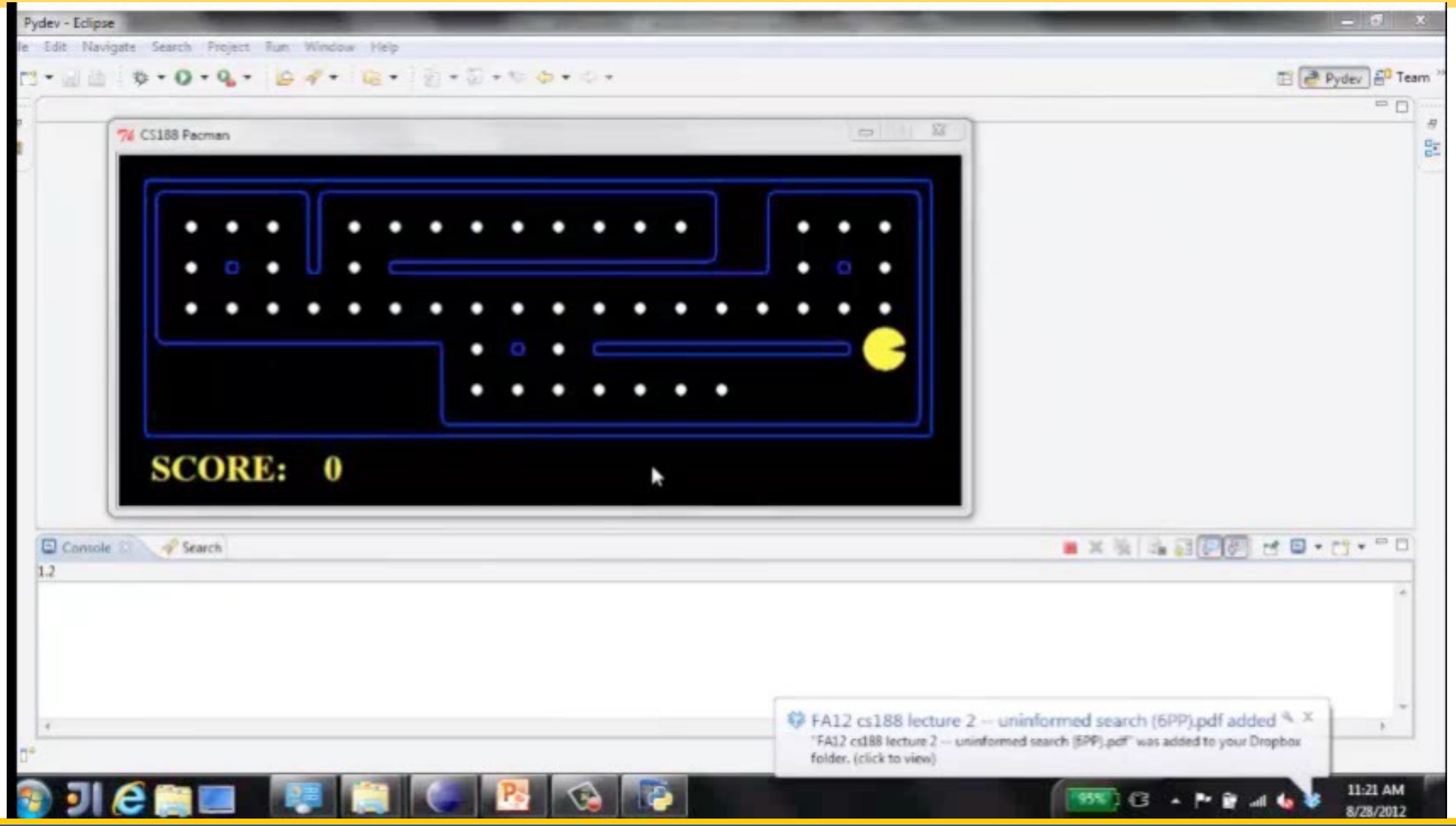




# Video of Demo Reflex Optimal

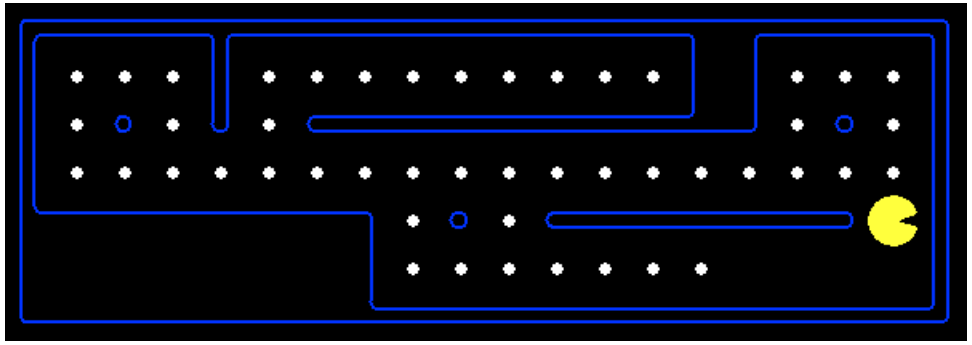


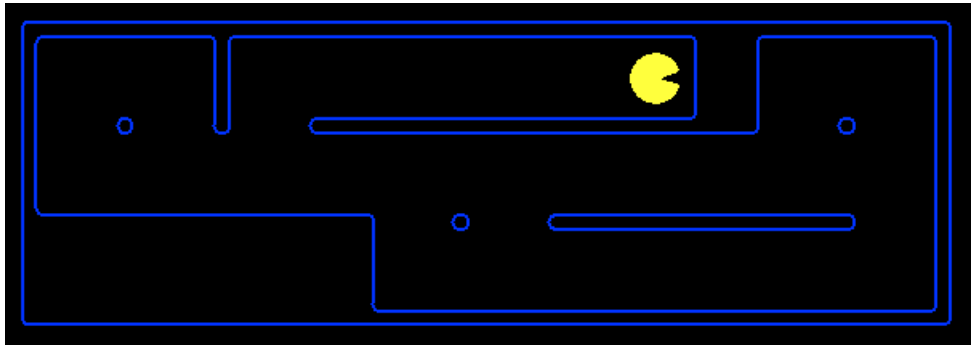
# Video of Demo Reflex Odd



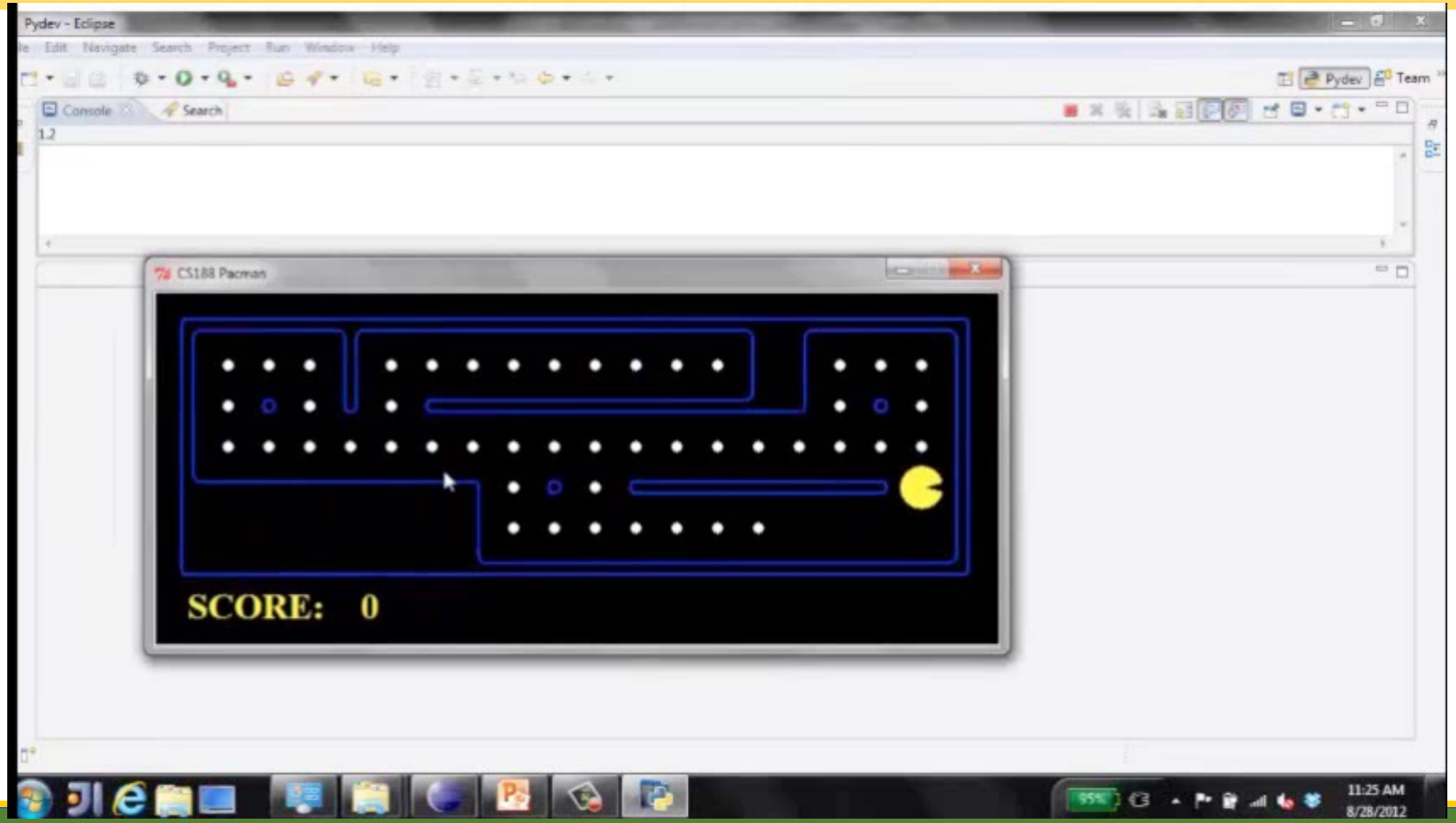
# Goal-based Agents

- 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



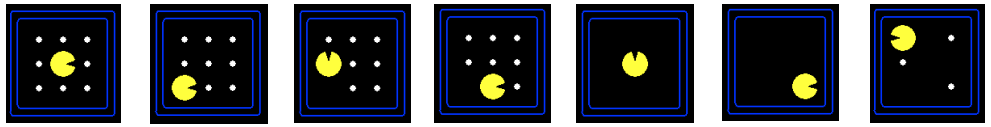


# Video of Demo Mastermind

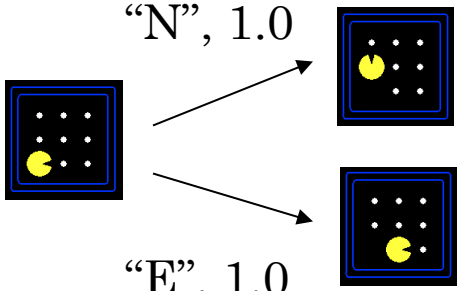


# Search Problem

- A search problem consists of:
  - A state space



  - A successor function (with actions, costs)

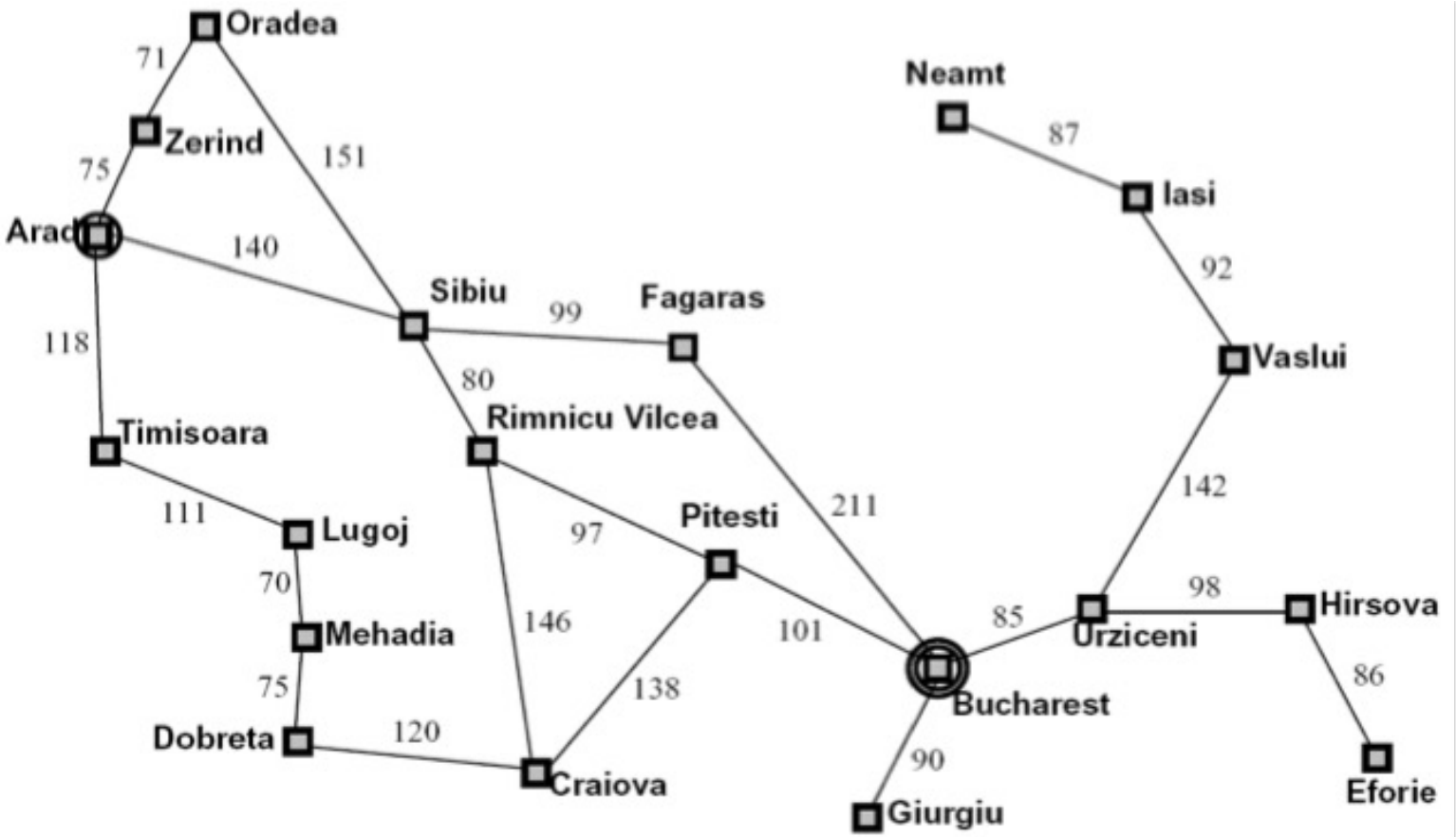



  - A start state and a goal test

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

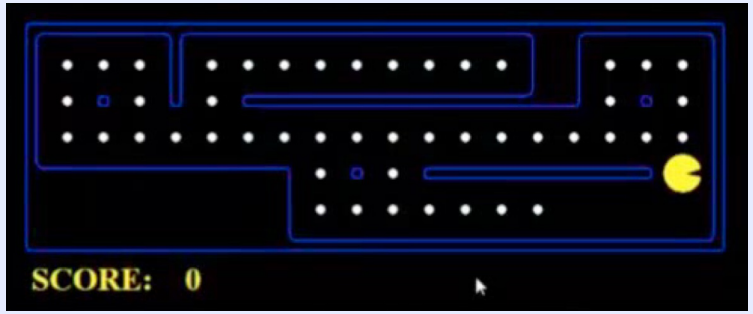
# Example: Romania

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



# What is in State Space

The world state includes every last detail of the environment




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

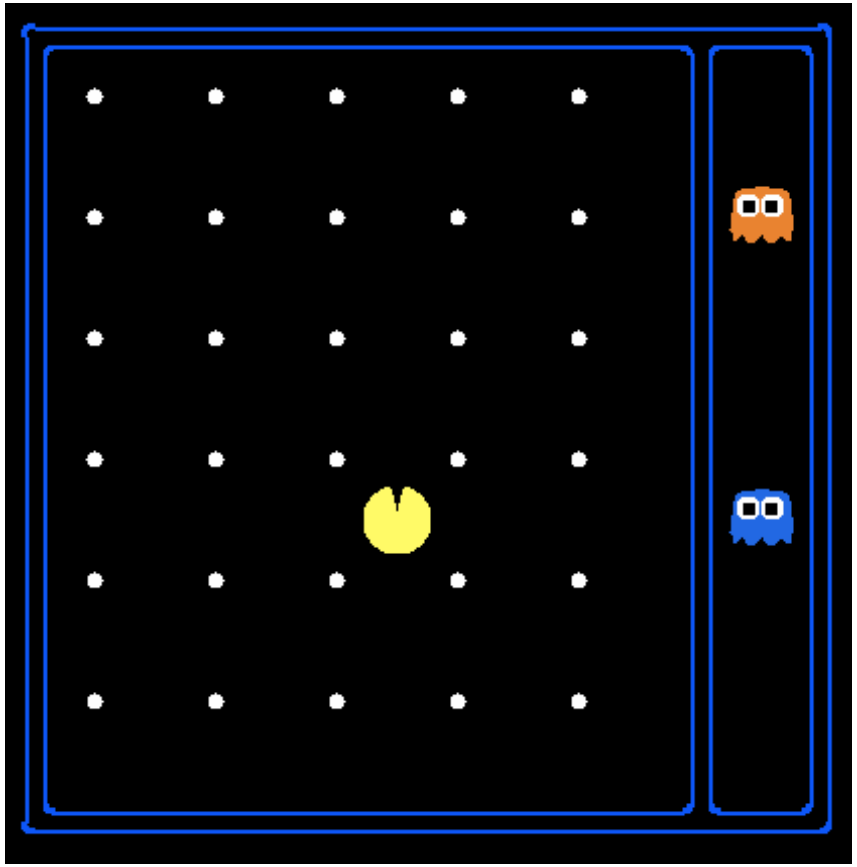
- Problem: Eat-All-Dots
  - States: {(x,y), dot booleans}
  - Actions: NSEW
  - Successor: update location and possibly a dot boolean
  - Goal test: dots all false

# State Space Size

- Search Problem:
  Eat all of the food

- Pacman positions: 10 x 12 = 120
- Pacman facing: up, down, left, right
- Food Count: 30
- Ghost positions: 12

- How many
- World states? $120*(2^{30})*(12^2)*4$
- States for pathing? 120
- States for eat-all-dots? $120*(2^{30})$

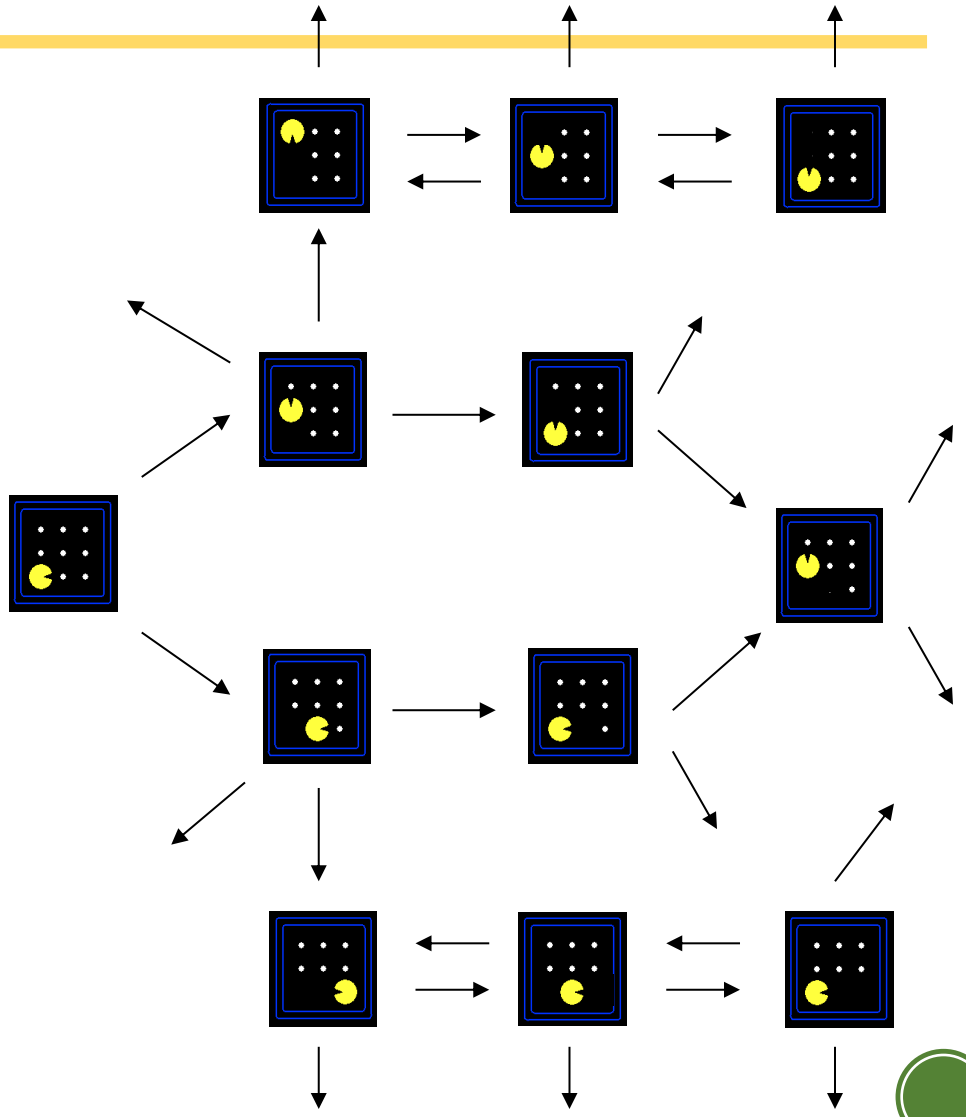


# State Space Graphs

- 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

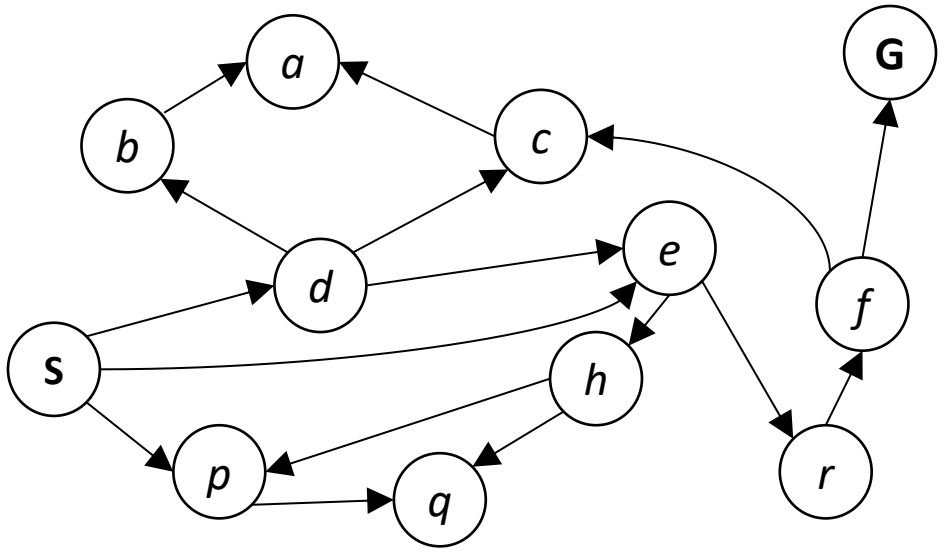


# State Space Graphs

- 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)

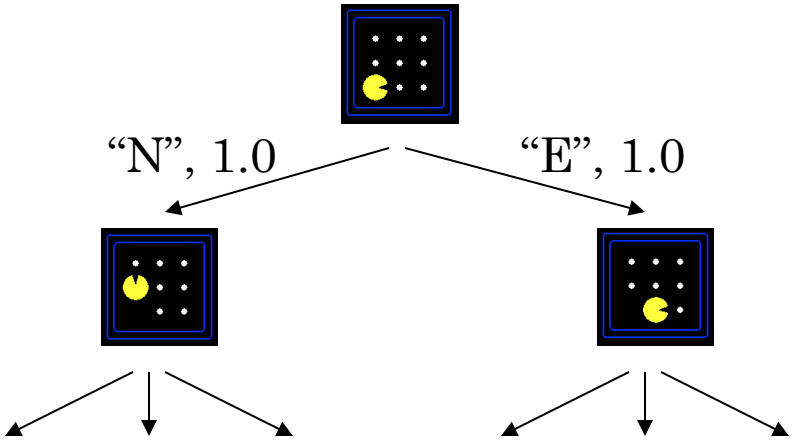
- In a state space graph, each state occurs only once!

- 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



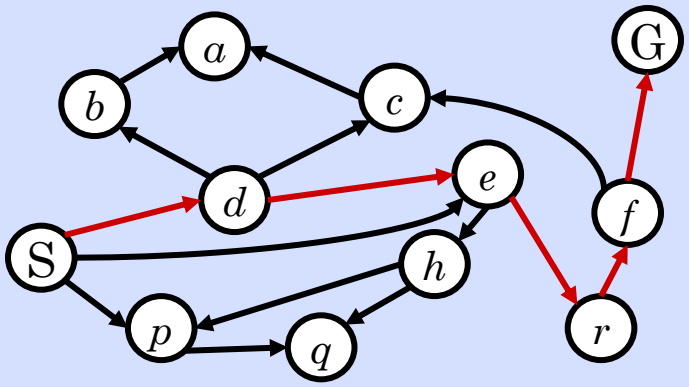

This is now / start

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

# State Space Graphs vs. Search Trees

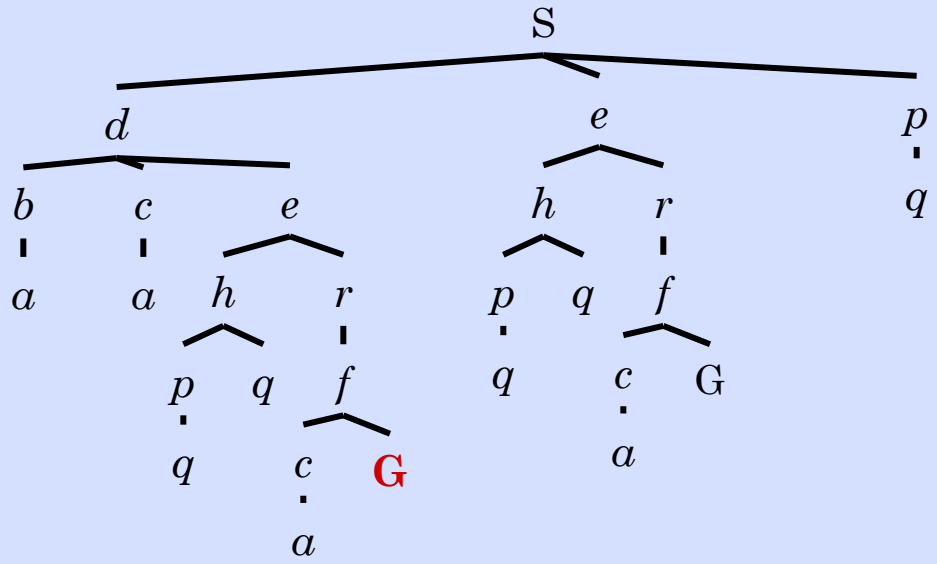
## State Space Graph




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

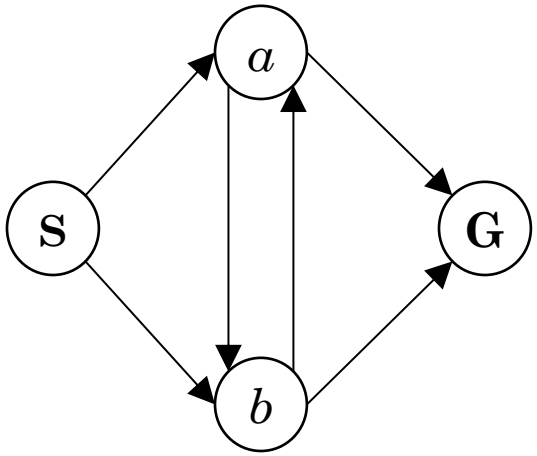
*We construct both on demand – and we construct as little as possible.*

## Search Tree



# Quiz: State Space Graphs vs. Search Trees

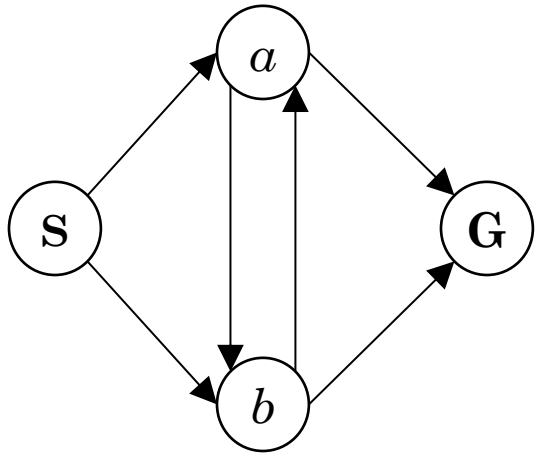
Consider this 4-state graph:



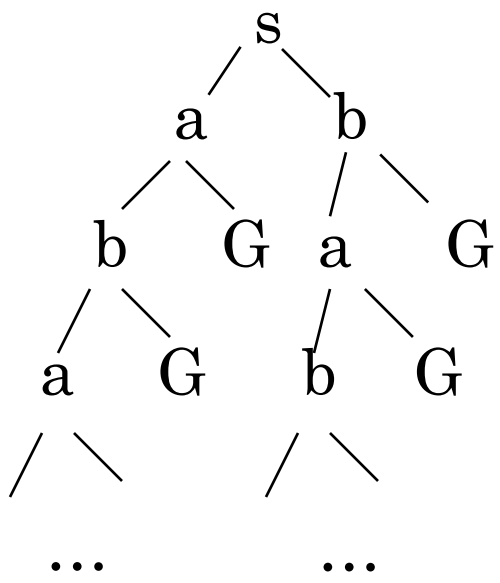

How big is its search tree (from S)?

# Quiz: State Space Graphs vs. Search Trees

Consider this 4-state graph:

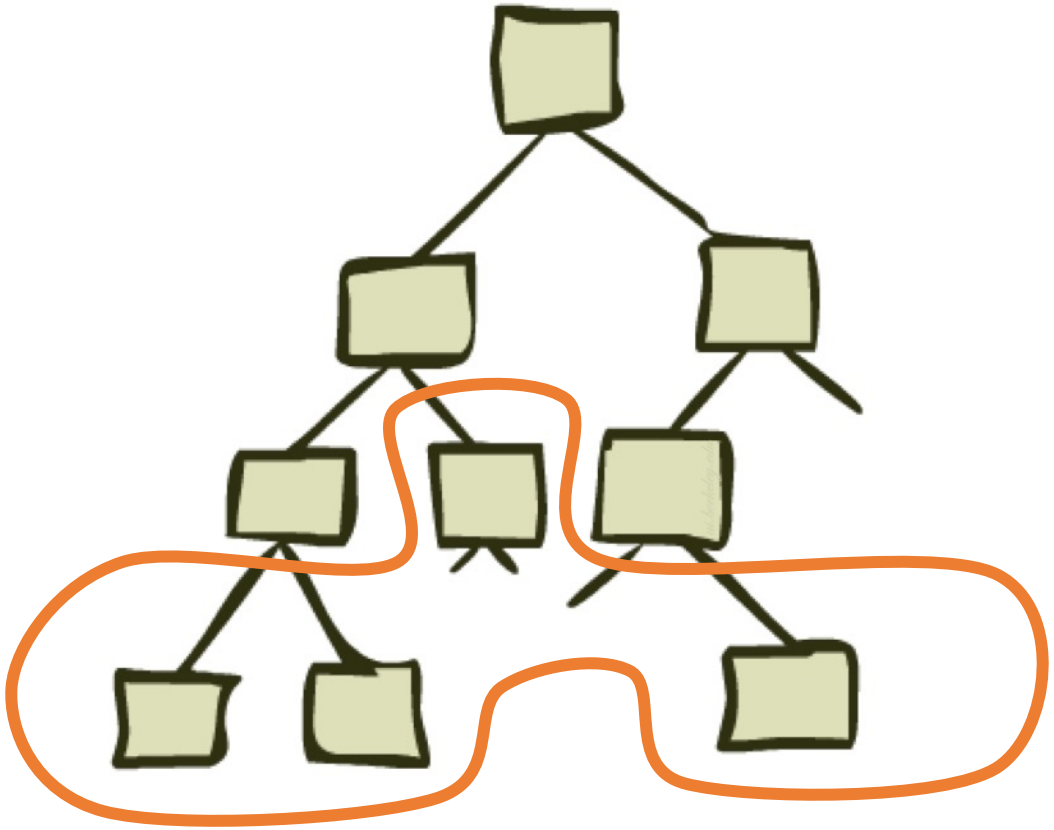



How big is its search tree (from S)?

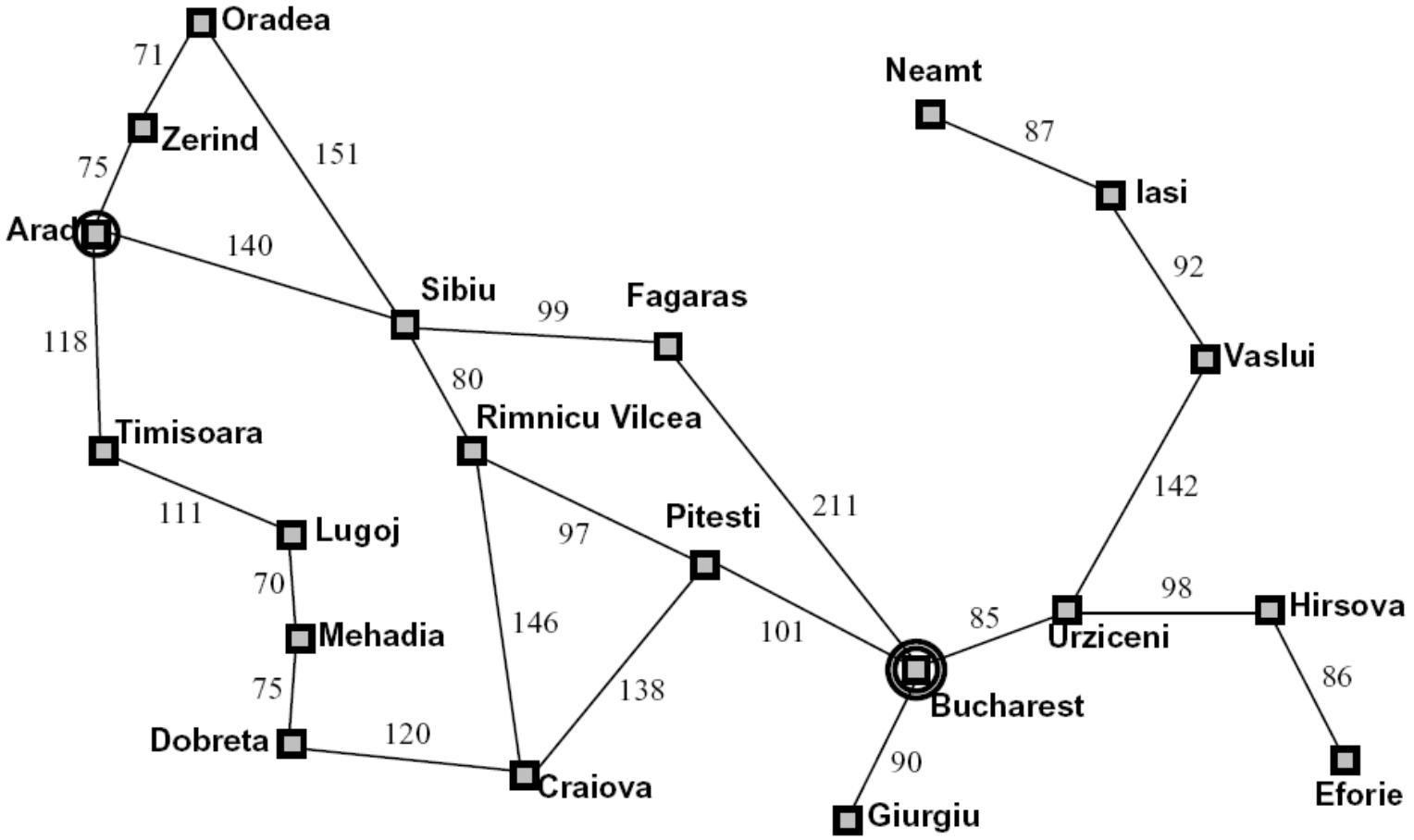



Important: Lots of repeated structure in the search tree!

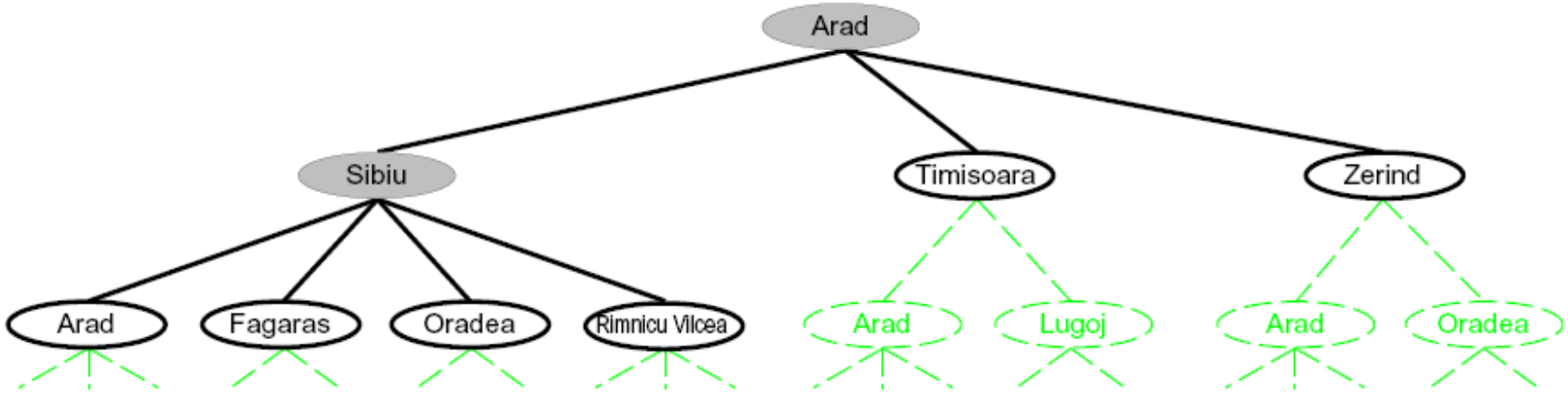
# Tree Search



# Search Example: Romania



# Searching with a Search Tree

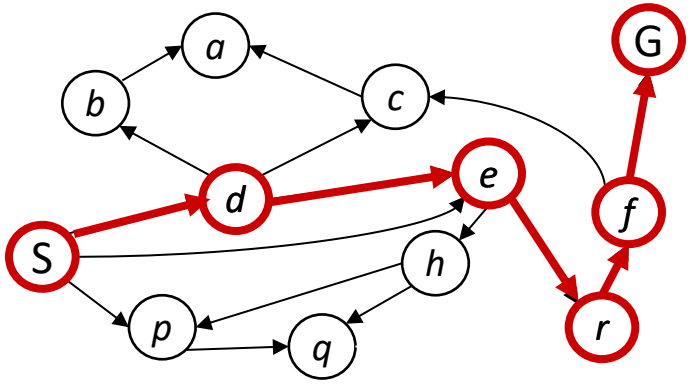


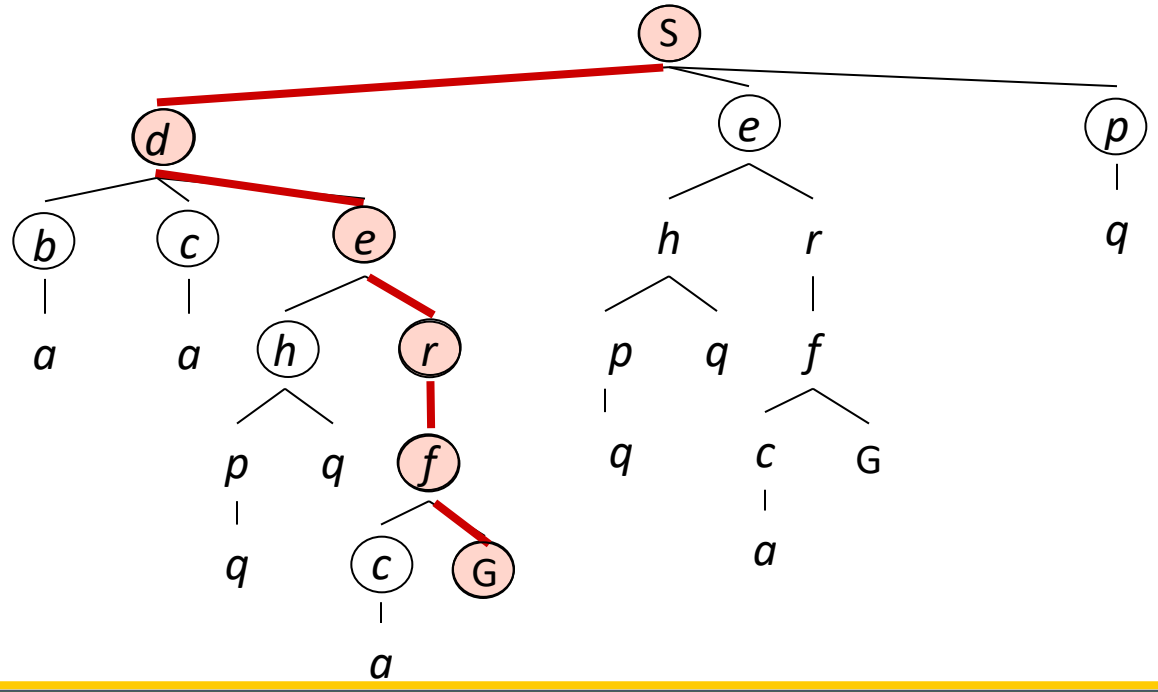

- Search:
  - Expand out potential plans (tree nodes)
  - Maintain a fringe of partial plans under consideration
  - Try to expand as few tree nodes as possible

# General Tree Search

- Tree Search
  - Initialize the *root node* of the search tree with the *start state*
  - While there are unexpanded leaf nodes (fringe):
    - Choose a leaf node (strategy)
    - If the node contains a goal state:
      return the corresponding solution
    - Else: expand the node and add its children to the tree
- Important ideas:
  - Fringe
  - Expansion
- Strategy: which fringe nodes to explore?

# 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~~