CSPB3202 Artificial Intelligence

Search

University of Colorado **Boulder**

Note: Some slides were adapted from the materials from Prof. Pieter Abbeel and Prof. Dan Klein at UC Berkeley, and Russell & Norvig's AI book.

Review- Agents in AI

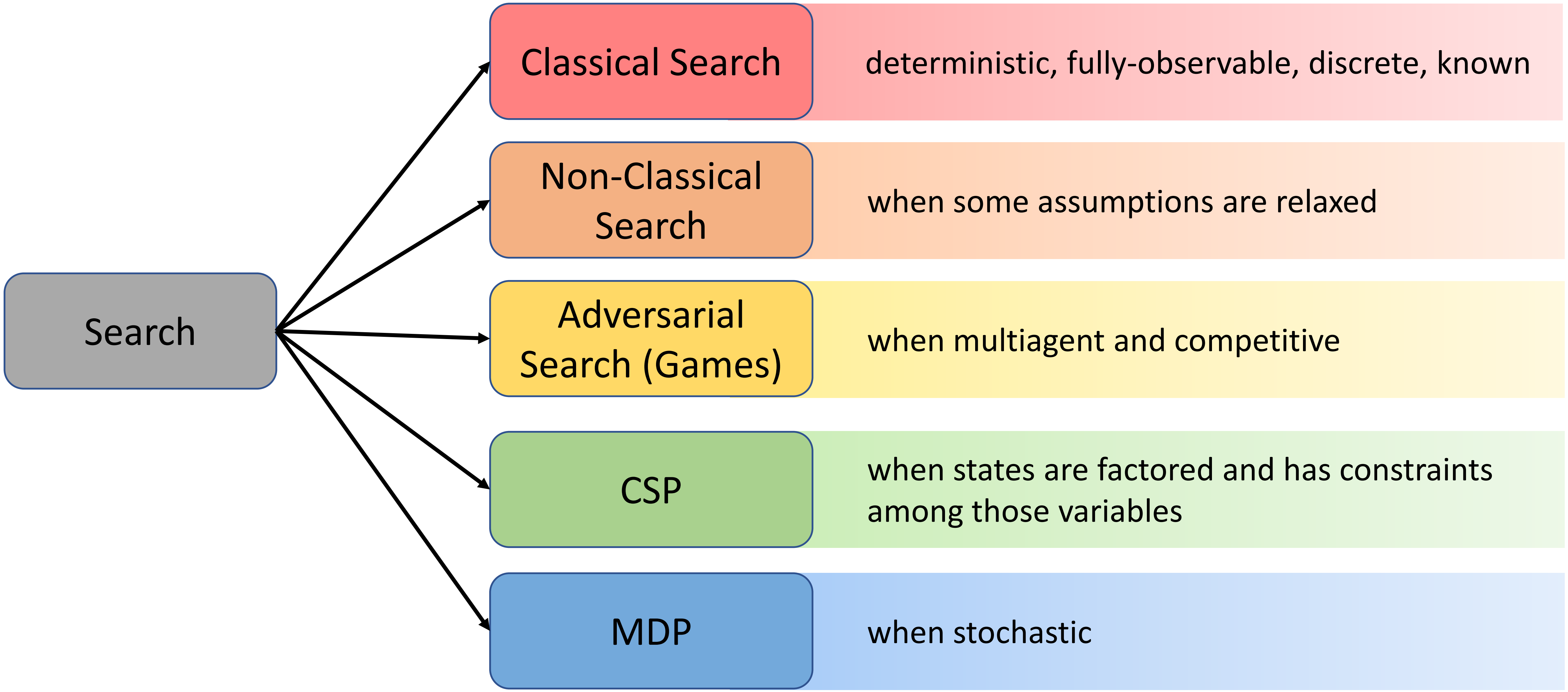
- An AGENT can sense the Environment and act to the environment according to its plan or an AI algorithm

- A RATIONAL agent tries to maximize its performance measure

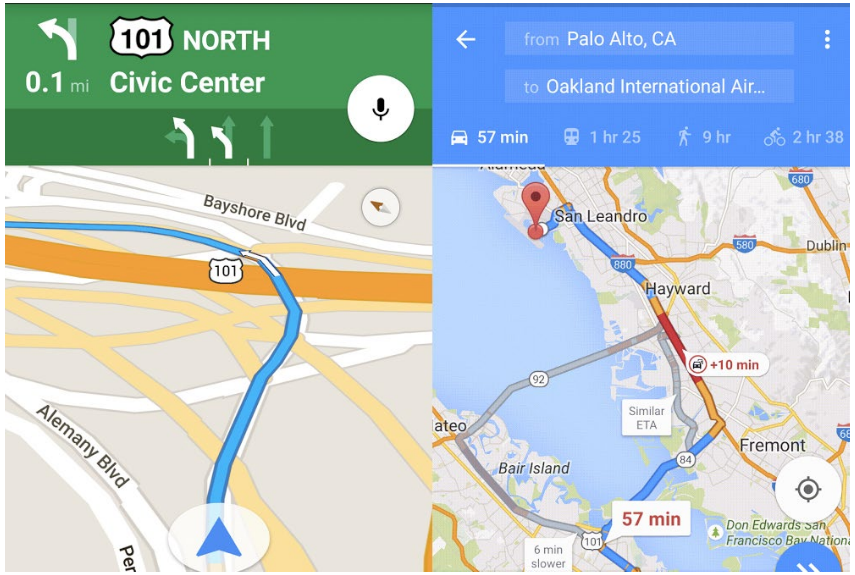
Review- Nature of Environments

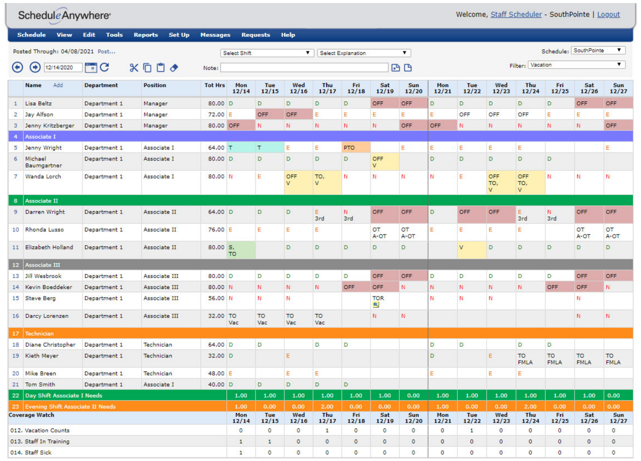
Criteria	Simpler	More complex
Sensor’s sensing ability	Fully observable	Partially observable
Number of agents involved	Single	Multi
State transition	Deterministic	Stochastic/Non-deterministic
Task	Episodic	Sequential
Does environment change while agent is thinking?	Static	Dynamic
Time	Discrete	Continuous
Agent’s state of knowledge	Dynamics/rules are known	Unknown

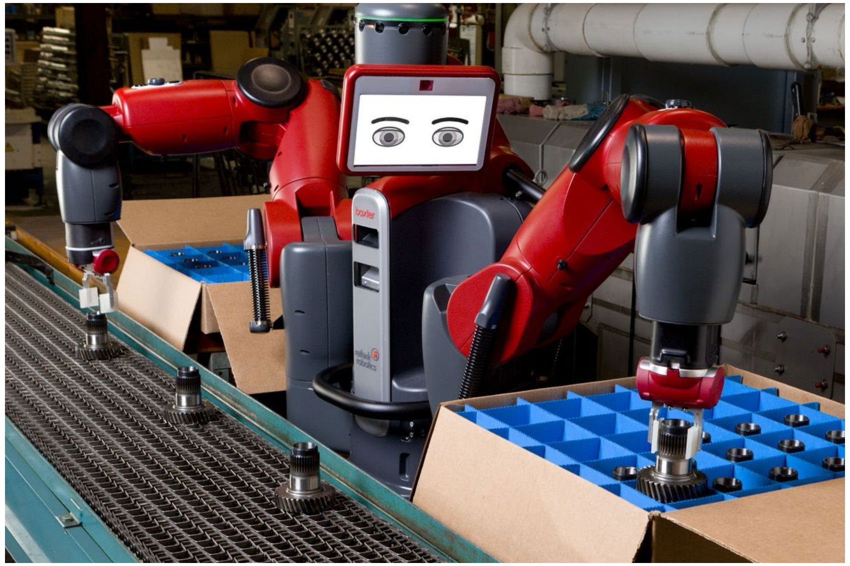
Overview- Search



Search Problems in the Real-World







What is a Search Problem?

- Problems that people actually care about
- Often very complex

- Simplified version
- Uses assumptions to simplify
- Mathematical formulation

Abstraction Level

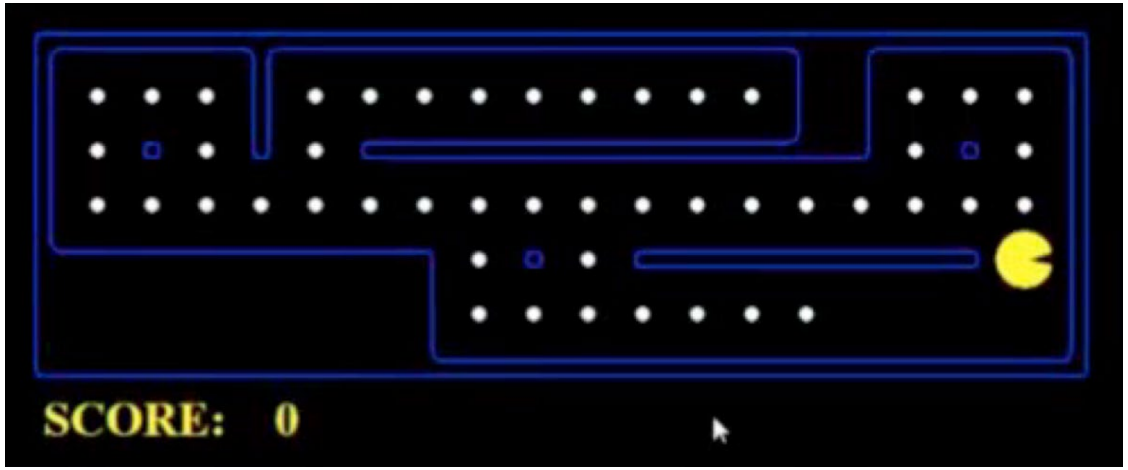
Europe googlemap

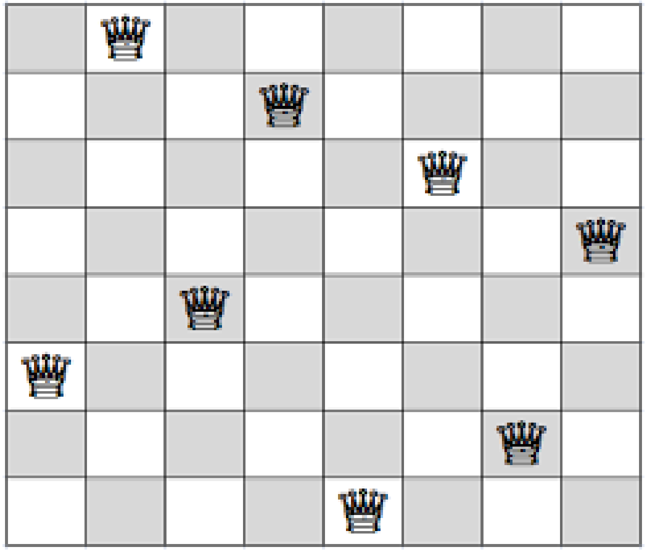
Eurail official map

Eurail cost map

What is a Toy Problem?

A problem that helps us to develop and test an AI algorithm



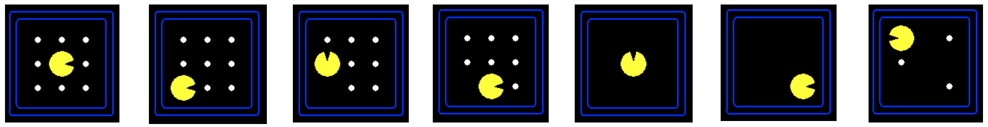


Search Problem Formulation

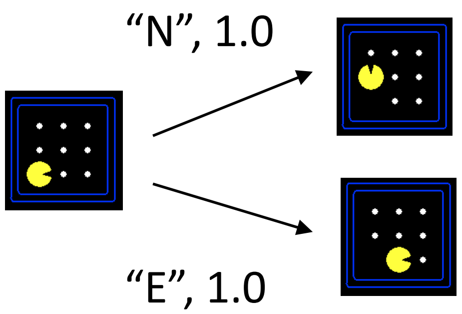
- Ingredients of a search problem
- Representation
- General search algorithm

Ingredients of a Search Problem

- A search problem consists of:
  - A state space



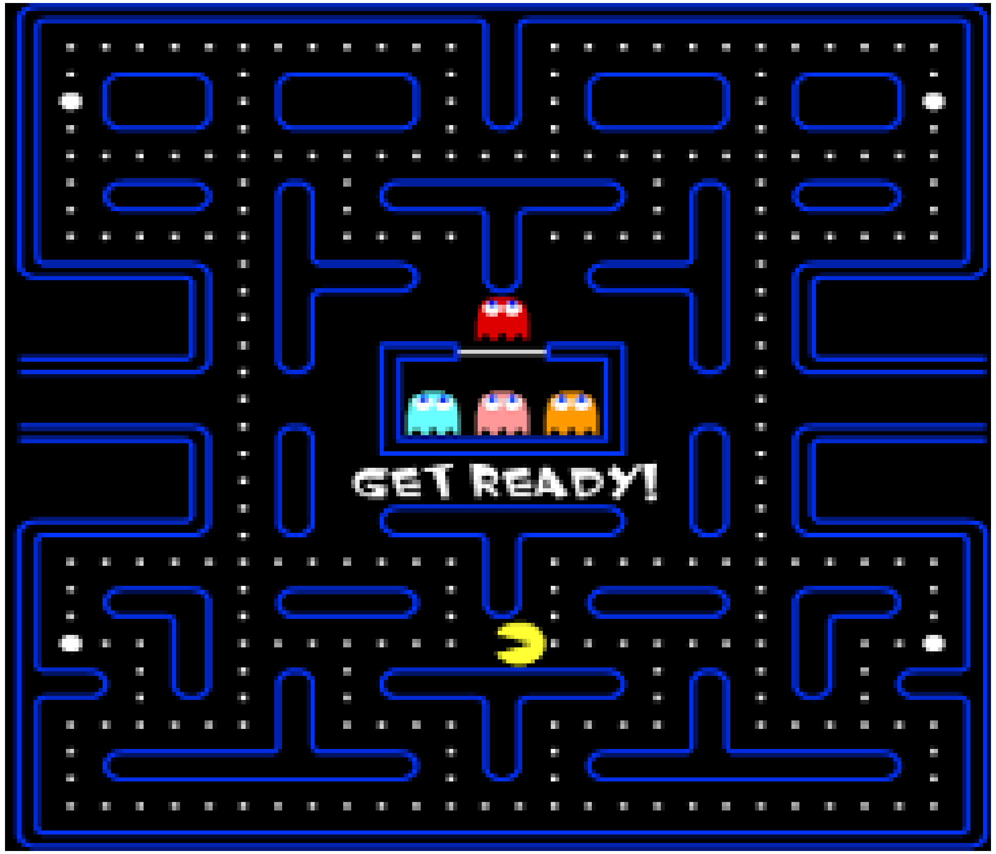
  - A successor function (with actions, costs)



  - A start state and a goal test

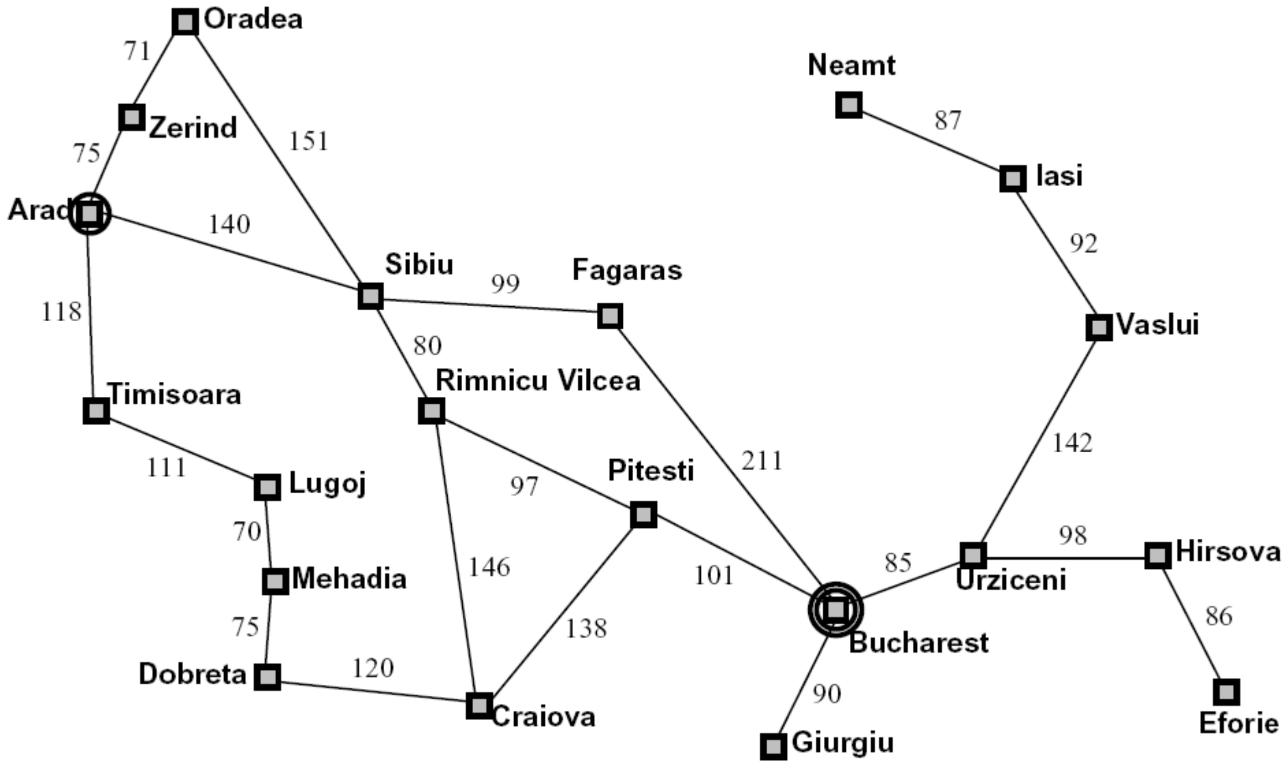
- A solution is a sequence of actions (a plan) which transforms the start state to a goal state

Example: Pacman



- State space:
  - Configurations
- Successor function:
  - Move by 1 unit
- Start state:
  - Initial Configuration
- Goal test:
  - Is food left == None
- Terminal state:
  - Is Alive == No || Is Goal achieved == Yes
- Cost: time

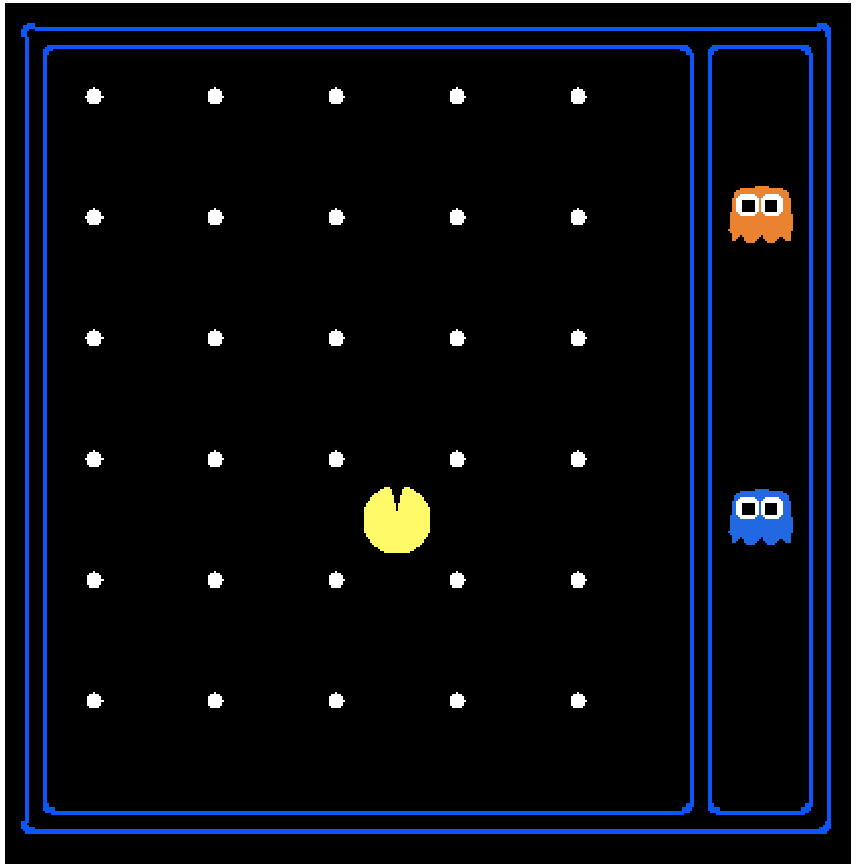
Example: Traveling in Romania



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

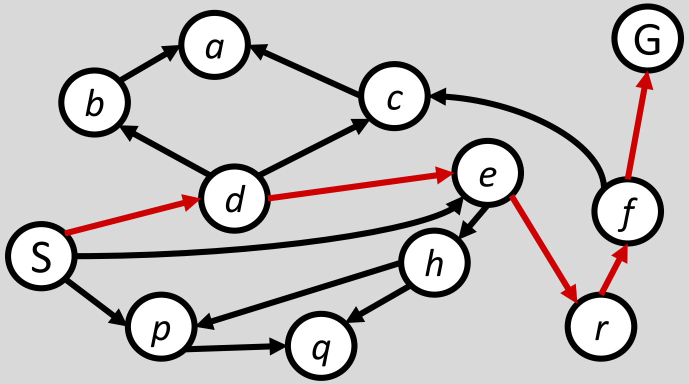
State Space Sizes

- 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?
    $120 \times (2^{30})$

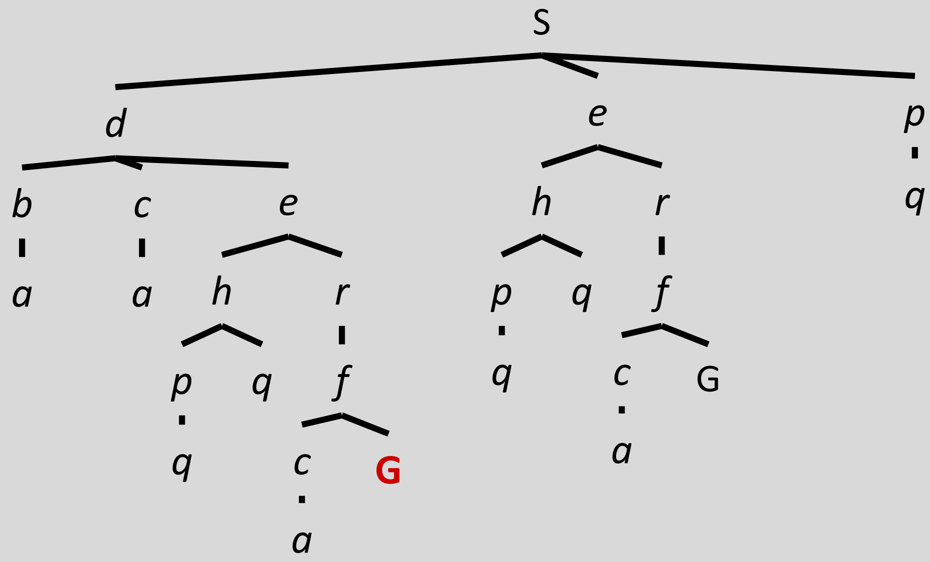


Search Representation

State Space Graph



Search Tree

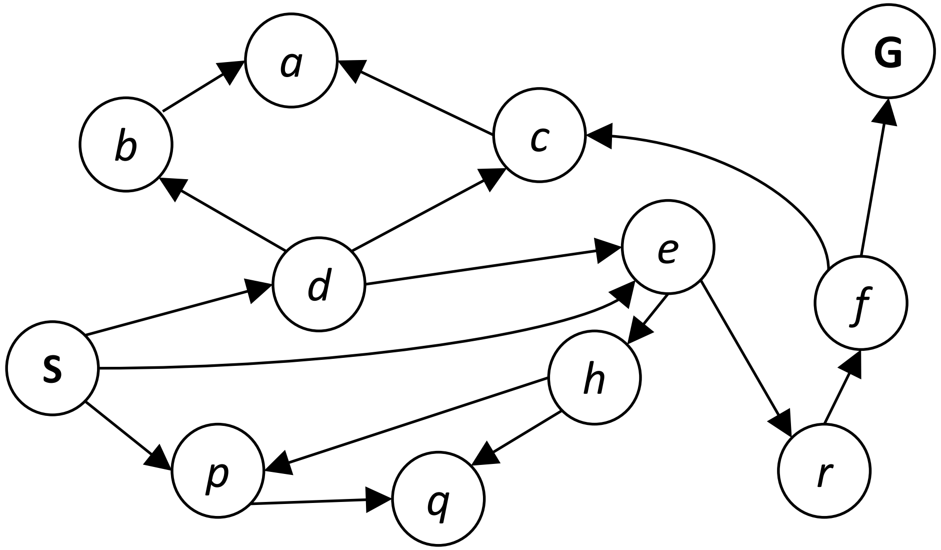


State Space Graphs

- 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)

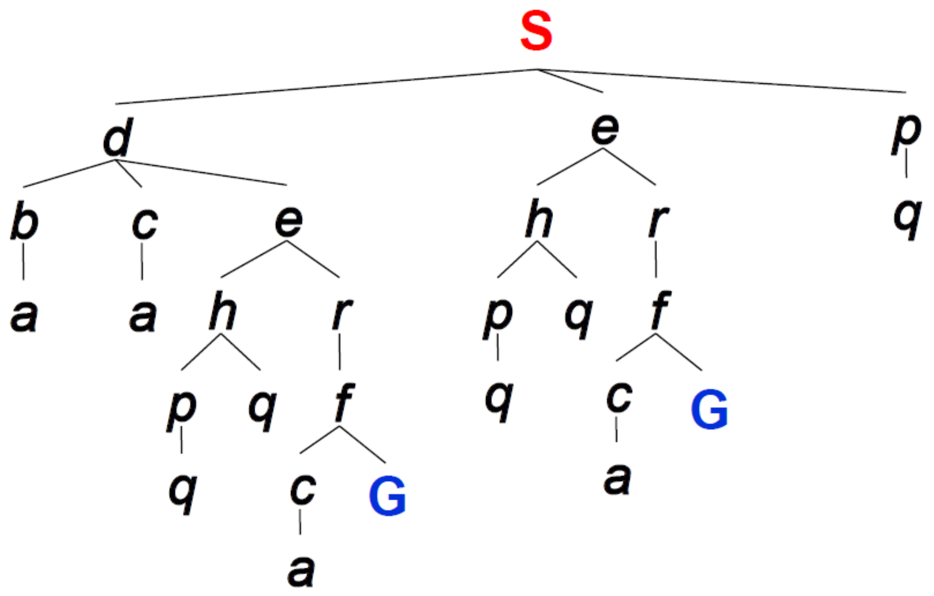
- In a state space graph, each state occurs only once!

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



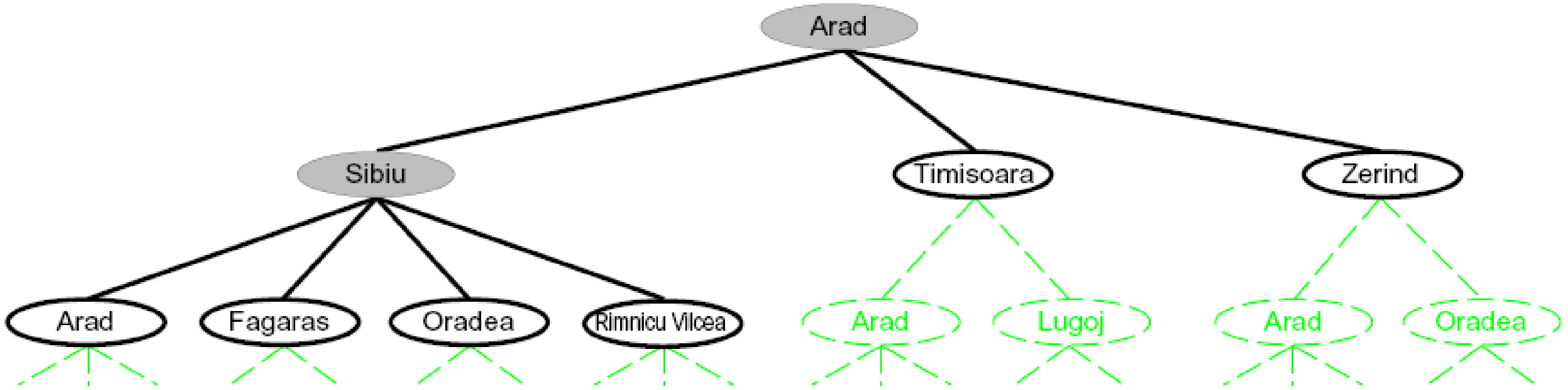
Tiny state space graph for a tiny search problem

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

Searching with a Search Tree



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

What we want in a Tree Search

- Can we get to the goal with the fewest effort (without having to expand all the nodes)?

- Which Fringe Node to Explore First?

  -> Search Strategy