COSC76/276 Artificial Intelligence
Fall 2022
Plan ahead with search problems and uninformed search

Soroush Vosoughi
Computer Science
Dartmouth College
Soroush@Dartmouth.edu

Reminders

- Thanks for the use of Ed Discussion!
  - Please continue using Ed Discussion (not emails) for your communication in the class
- SA-0 due 11:59pm ET on Friday
  - Office hours available on Canvas
- The tentative calendar for assignments and lectures for the next few weeks is on Canvas
  - SA-1 will be posted soon
  - PA-1 will be posted soon

Reminders

- Please enable email notification for Canvas messages.
- Late days!

SA0 Short first quiz/assignment

- SA0 Short quiz/assignment:
  - To get you up and running with the development environment
  - To ensure we are on the same page for your learning and expectations
  - Share initial thoughts on AI
- You will find it on Canvas

Discussion: what is the "correct" AI?

The science of making machines that:

Think like people	**Think rationally**
Act like people	**Act rationally**

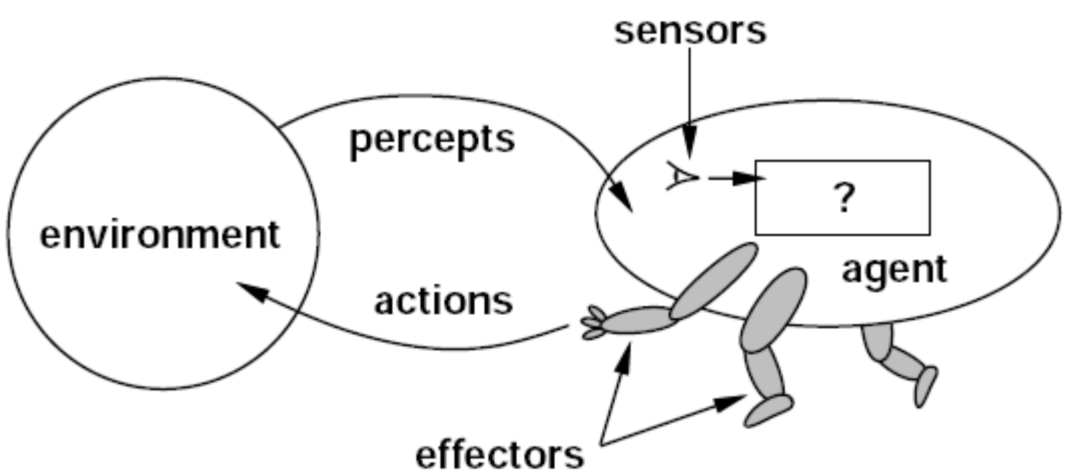
Your turn to interact

What does “rationally” mean?

- **Rational**: maximally achieving predefined goals
- **Utility** of outcomes are a measure for the goals
- **Rationality**: only concerns what decisions are made
- Being rational means **maximizing expected utility**

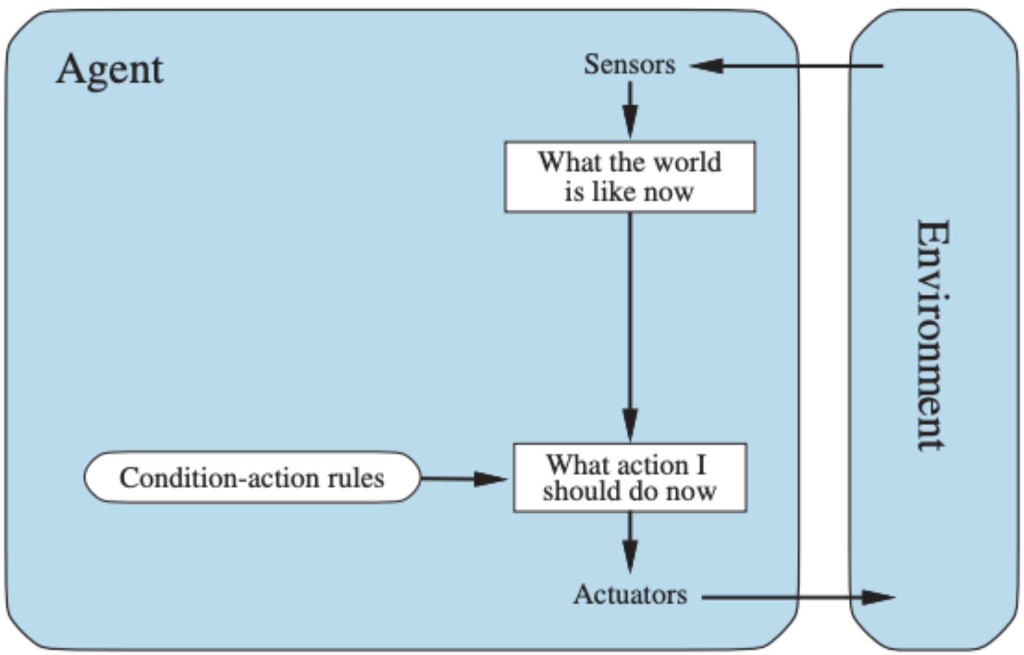
Rational agent

- An **agent** is an entity that *perceives* and *acts*.
- A **rational agent** selects actions that maximize its (expected) **utility**.



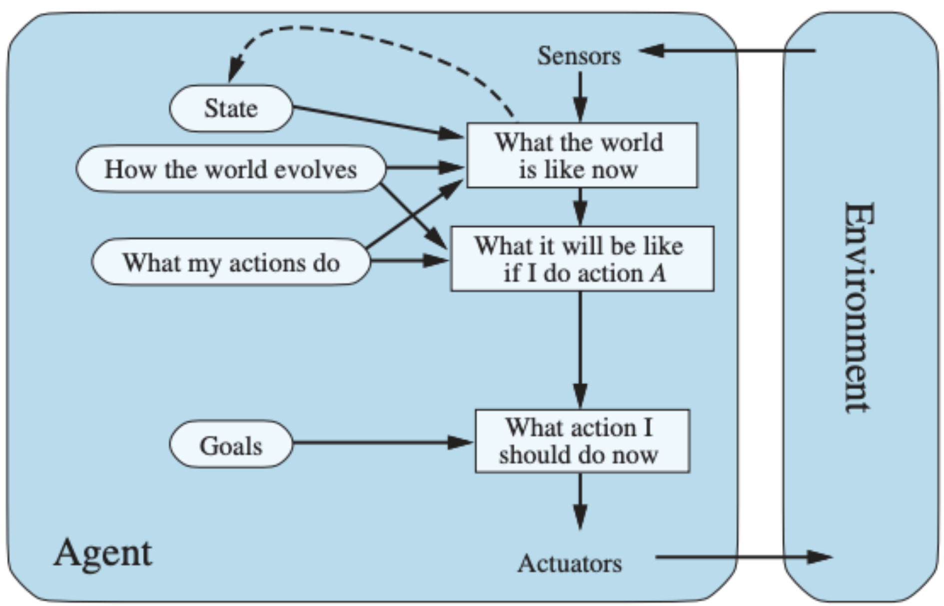
Reflex agents

- Choose action based only on current perception (and memory if available)
- Does not look ahead



Planning agent

- Actions based on looking at (hypothesized) consequences of actions
- A model is needed to see how the world and agent evolve given actions
- The goal needs to be explicit



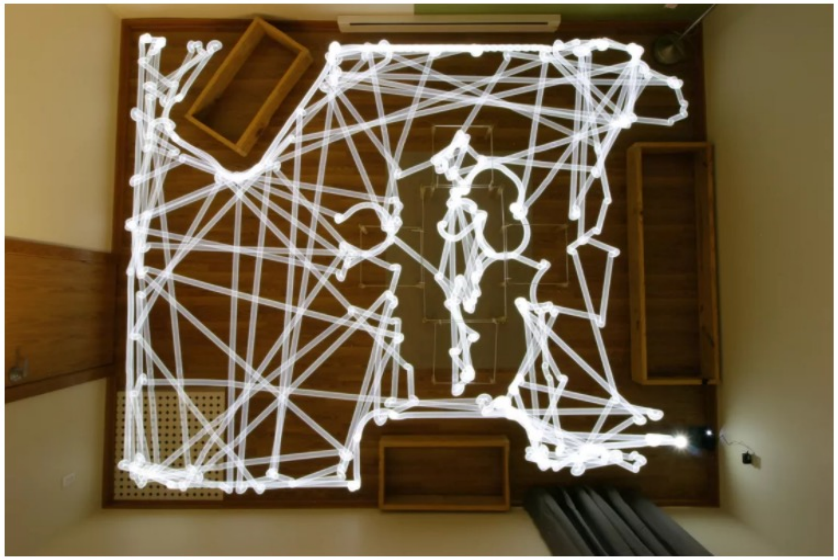
Which one is a better agent?

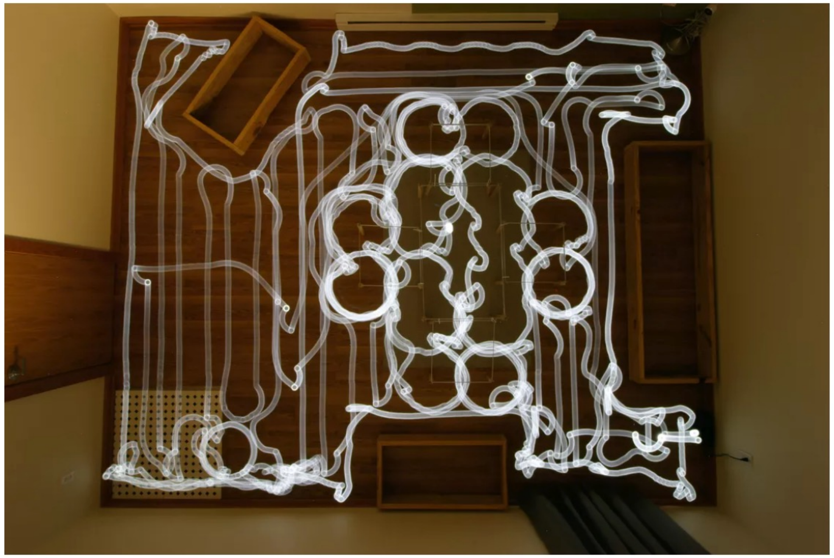
Your turn to interact

Recap: reflex vs planning agent

Reflex

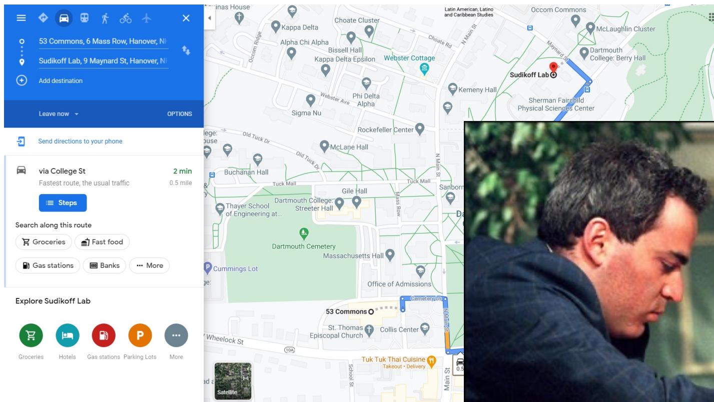
Planning

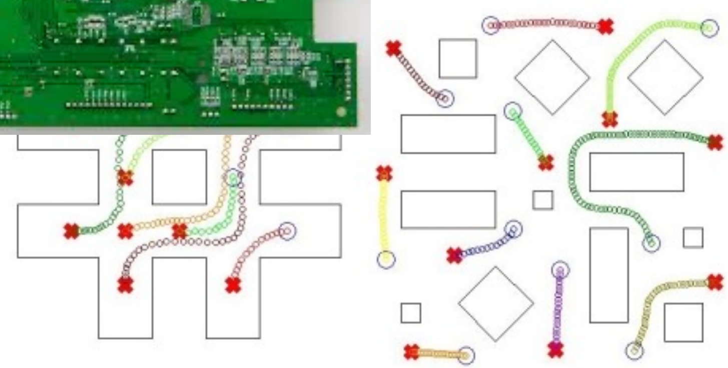




Source: https://www.cnet.com/home/kitchen-and-household/this-is-why-your-roombas-random-patterns-actually-make-perfect-sense/

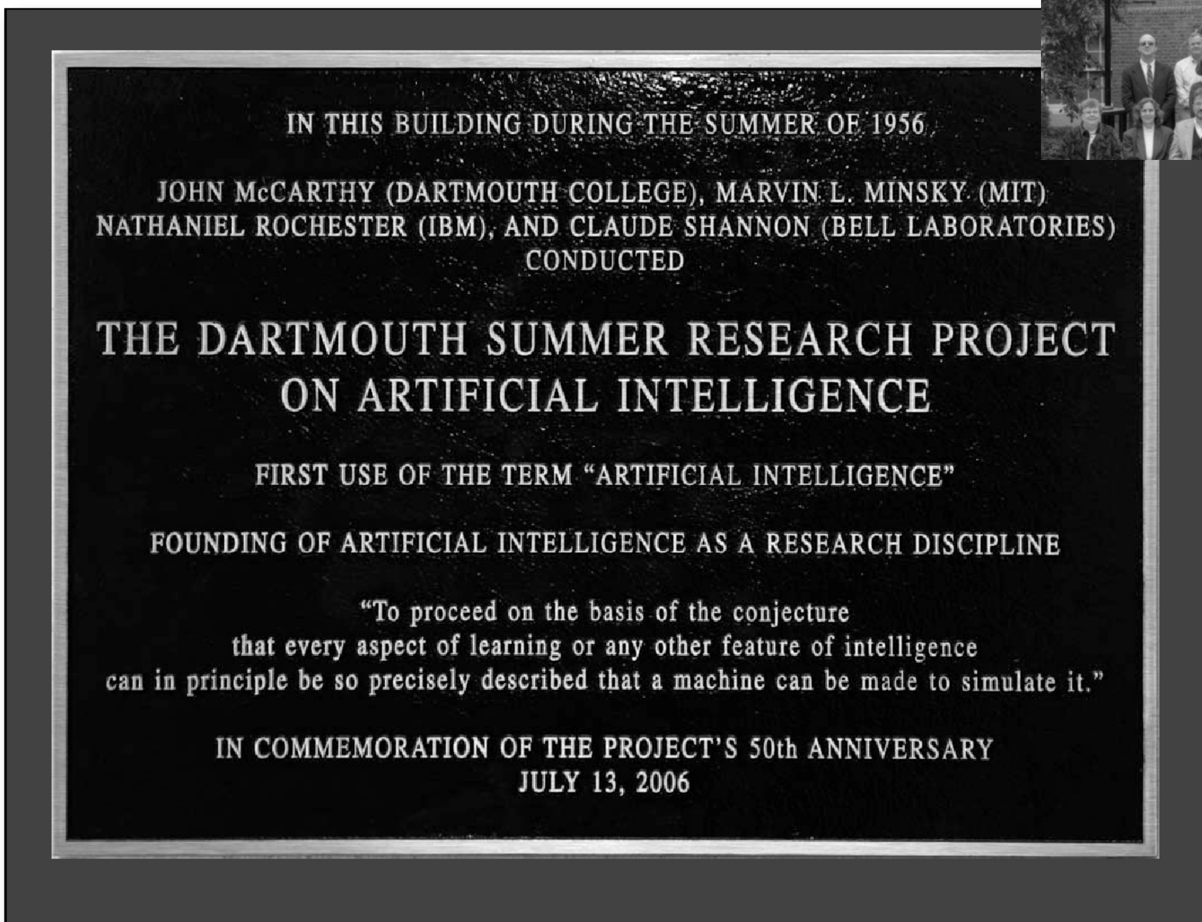
Type of problems we'll look at





Where Did AI Originate From?

AI Coined at Dartmouth



Additional readings

- AIMA book Chapter 1 and 2
- (Will be reported on Canvas too in the calendar)

Today's learning objectives

How do we make rational agents plan ahead?

- Model problems as a search problem
- Define tree-search algorithm
- Identify properties of search algorithms
- Implement uninformed search methods
- Determine properties of those search methods

Outline

- Search problems
- Uninformed search algorithms (tree-search, without memory)

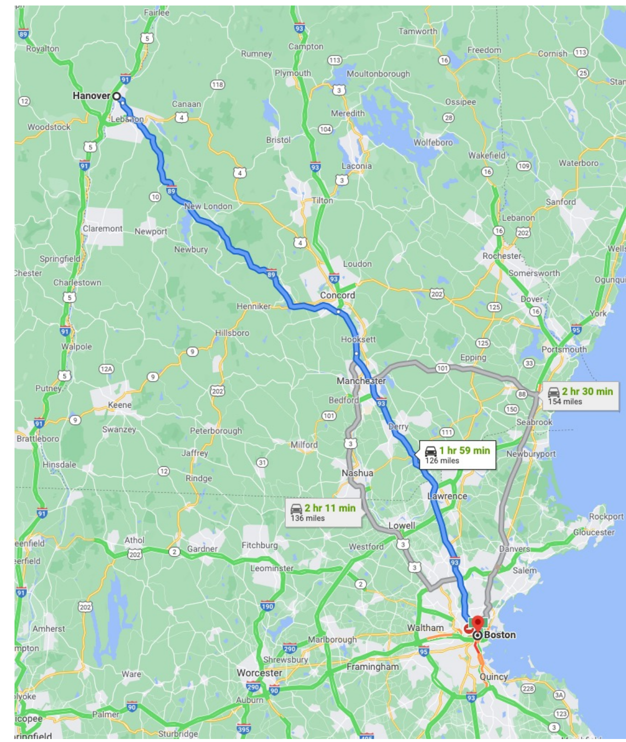
Outline

- Search problems
- Uninformed search algorithms (tree-search, without memory)

How can we plan?

- We need a model:
  - How should the agent's **state be represented**? – state space
  - What are the **actions** for the system? – action space
  - How does the state **change** in response to the action?

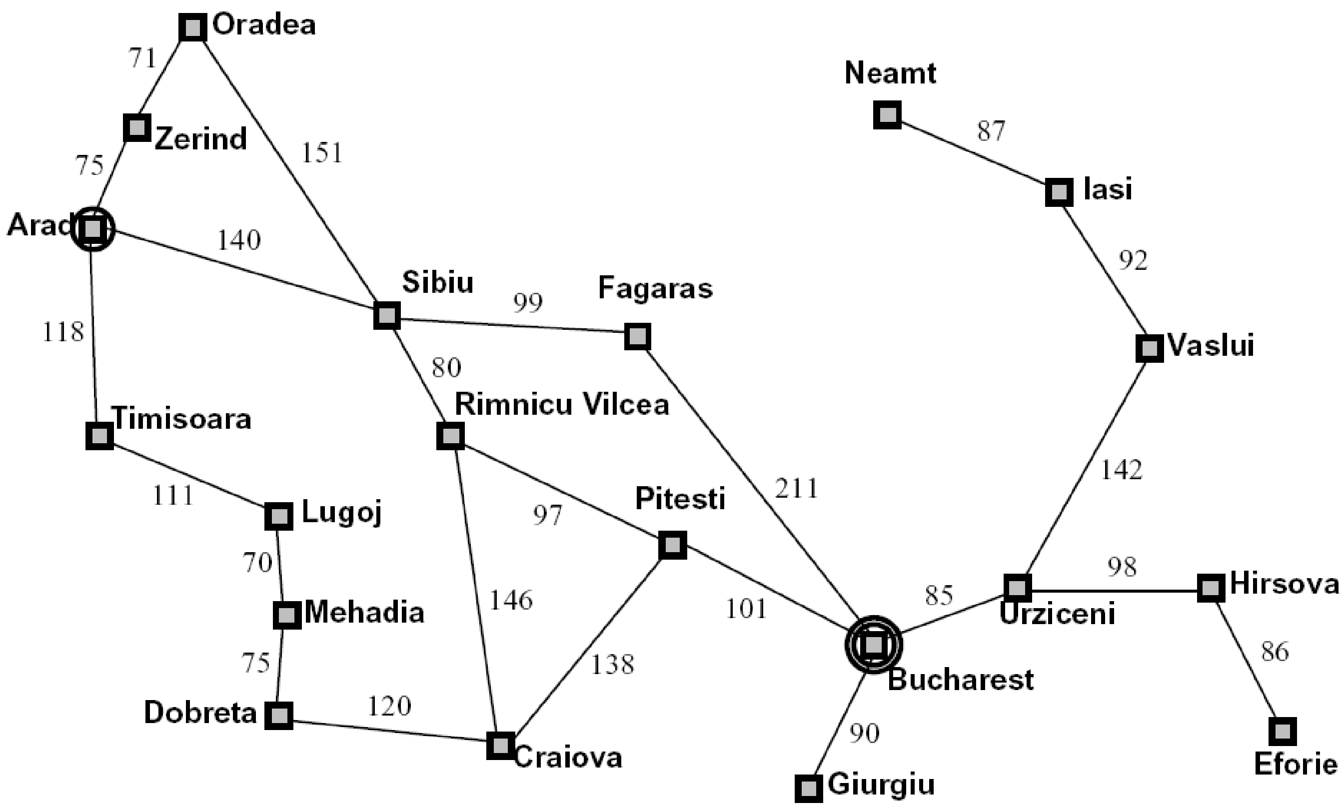
Discussion: example



- We want a self-driving car to travel between cities connected via highways
  - What are the agent' states?
  - What are the agent's actions?
  - How does the state change in response to the input?

Discuss

Example: Traveling in Romania



- State space: cities
- Successor function: go to adjacent city
- Cost: distance between cities
- Start state: Arad
- Goal test: is state == Bucharest?

State space graph

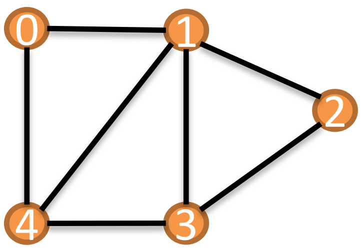
- State space graph: A mathematical representation of a search problem
  - States are (abstracted) world configurations
  - Arcs represent successors (action results)
  - The goal test is a set of goal states

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

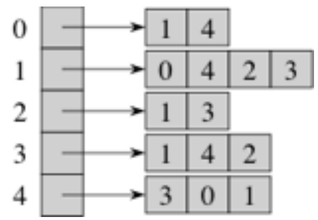
- The full graph is typically too big to store in memory

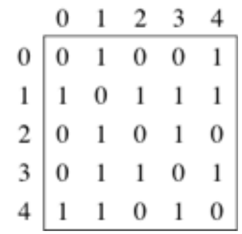
Reminder (from CS10): Graph representations

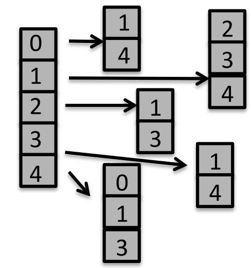
Anyone remembers?



{{0,1},
{0,4}, {1,2},
{1,3}, {1,4},
{2,3}, {3,4}}







Method	Edge List	Adjacency List	Adjacency Matrix	Adjacency Map
`in/outDegree(v)`	O(m)	O(1)	O(n)	O(1)
`in/outNeighbors(v)`	O(m)	$O(d_v)$	O(n)	$O(d_v)$
`hasEdge(u,v)`	O(m)	$O(\min(d_u,d_v))$	O(1)	O(1)
`insertVertex(v)`	O(1)	O(1)	$O(n^2)$	O(1)
`removeVertex(v)`	O(m)	$O(d_v)$	$O(n^2)$	$O(d_v)$
`insertEdge(u,v,e)`	O(1)	O(1)	O(1)	O(1)
`removeEdge(u,v)`	O(m)	O(1)	O(1)	O(1)

Best performance is shown in red

n = number of nodes (5), m = number of edges (7), d_v = degree of node v