
CS 232: Artificial Intelligence

Fall 2026

Prof. Carolyn Anderson
Wellesley College

Reminders

- ♦ Homework 3 will be released today
- ♦ Tutor help hours:
 - Aeka: Thursdays 7:30-8:30 in H103
 - Amber: Fridays 1:30-2:30 in Whiteboard Alley
 - Coco: Mondays 7-8 in Whiteboard Alley
- ♦ I have help hours Friday 3-4 in W423

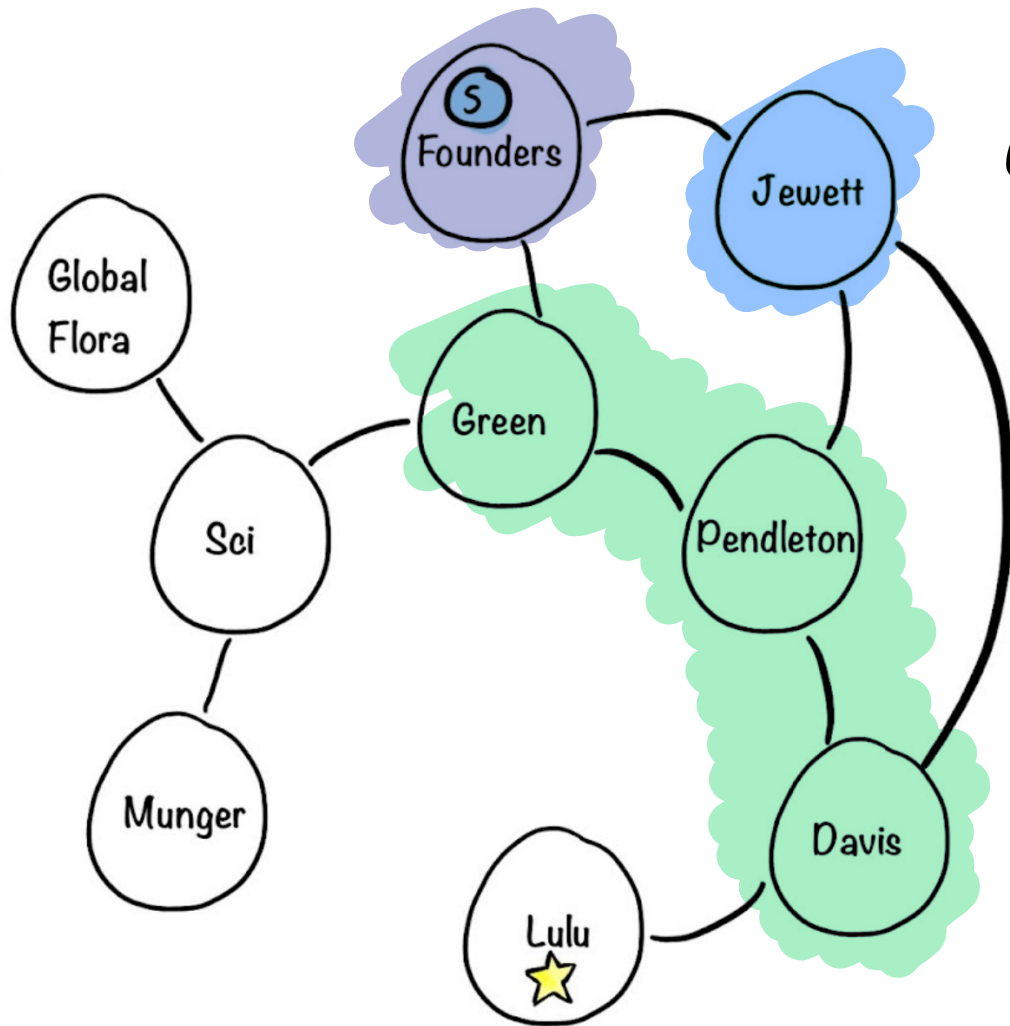
Homework 3: Pacman versus the world

The UC Berkeley AI course has a set of projects that revolve around the classic Pacman arcade game.

In this assignment, your goal is to fill in some of the functions related to search, which help Pacman find food pellets. This version of Pacman won't have any ghosts yet.

Recap

Search Trees



Frontier = 

Current Node = 

Visited = 

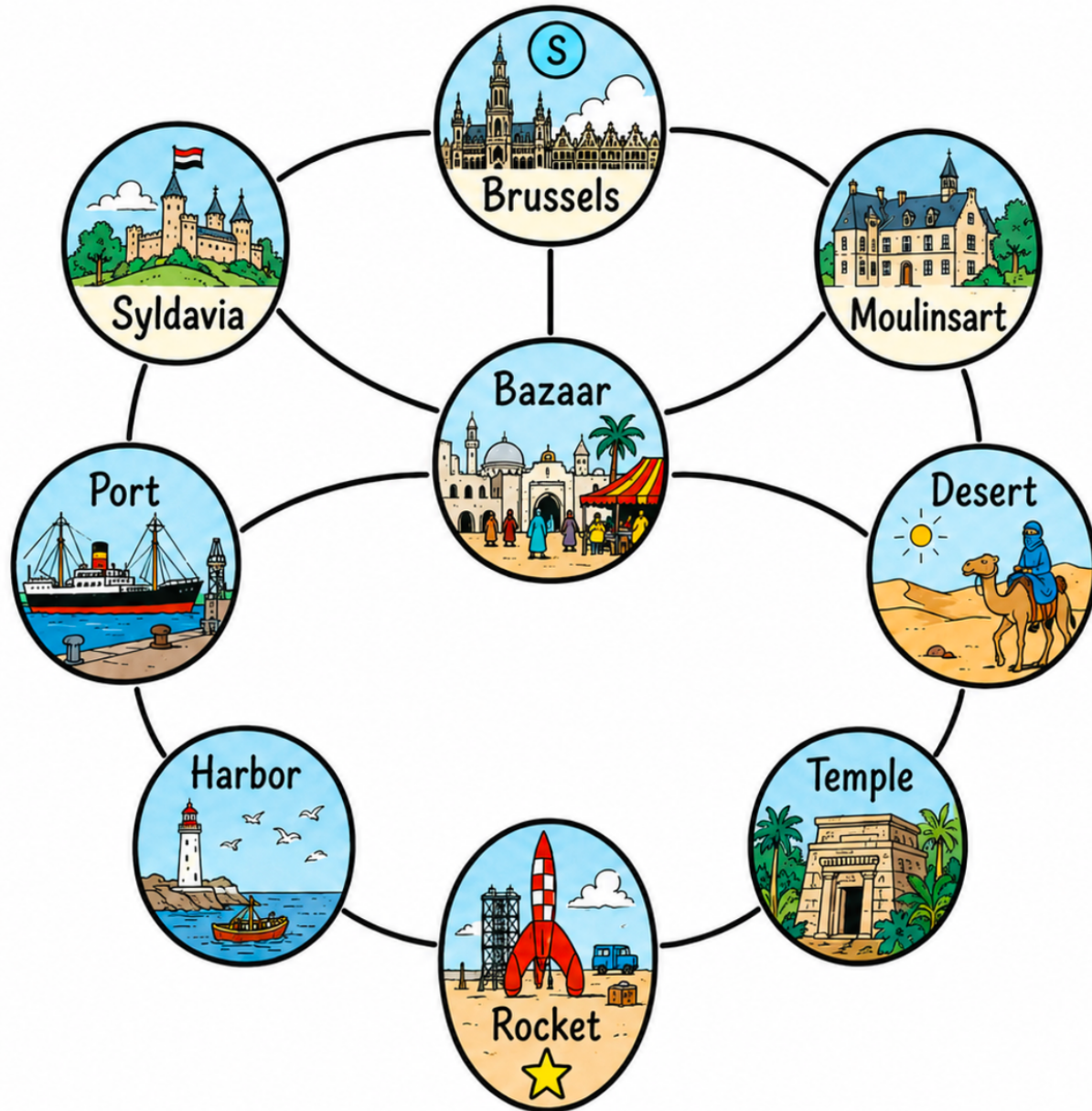
Choose leaf node from frontier for expansion according to the **search strategy**

Determines the search process

Graph Search

```
function GraphSearch(problem, strategy) return solution or failure
  Initialize frontier to start state of problem
  Initialize explored set to be empty
  do
    - if the frontier is empty then return failure
    - choose leaf for expansion according to strategy and
      remove from frontier
    - if node contains goal state then return solution
    - else:
      - add node to the explored set
      - expand the node and add resulting nodes to the
        frontier only if not already in the explored set
```


Uninformed Search Practice



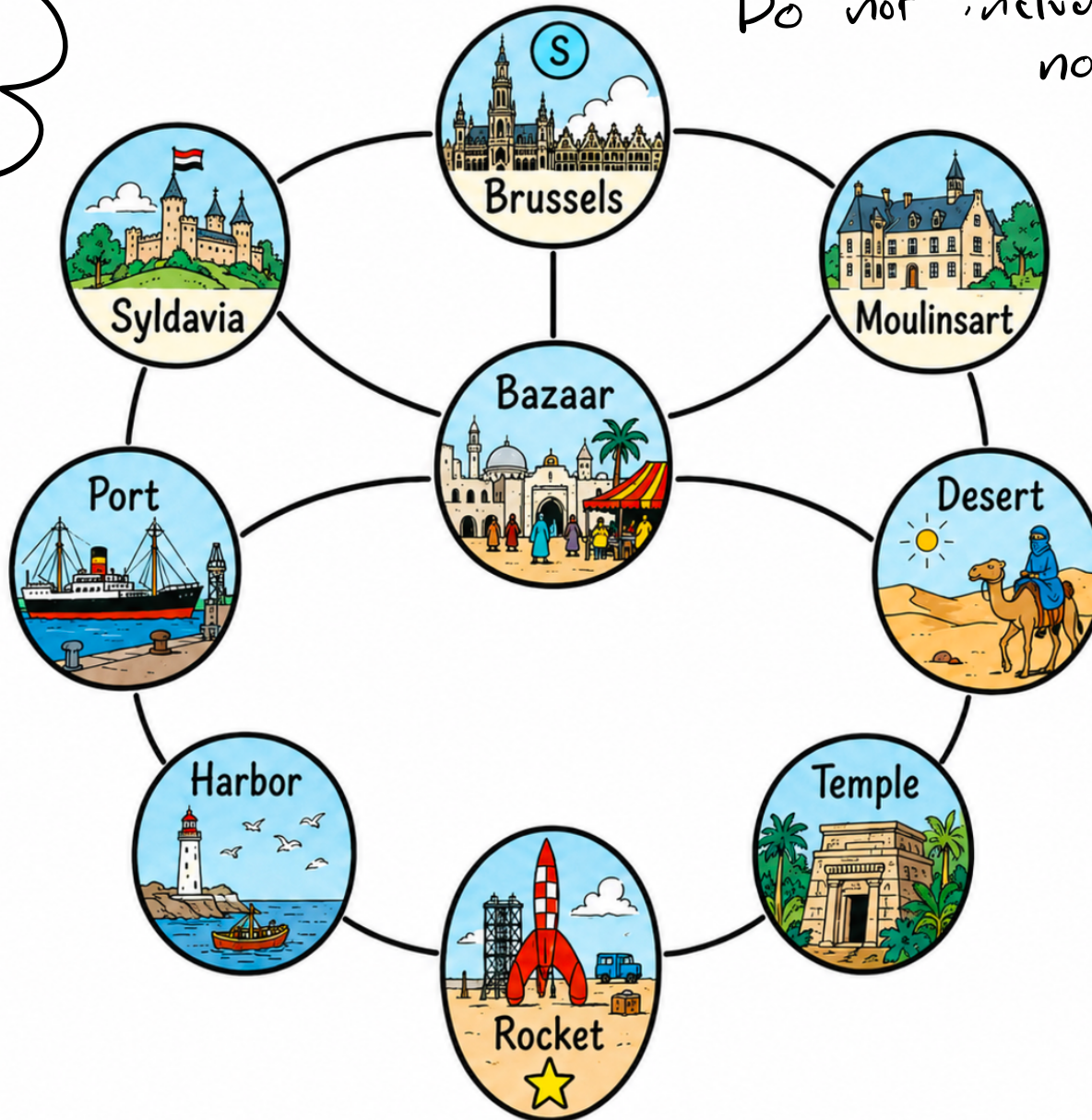
Uninformed Search Practice

BFS

Do not include duplicate nodes in frontier!

Frontier

~~Brussels~~
~~Bazaar~~
~~Moulinsart~~
~~Syldavia~~
~~Desert~~
~~Port~~
~~Temple~~
~~Harbor~~
Rocket ✱



Visited

Brussels
Bazaar
Moulinsart
Syldavia
Desert
Port
Temple
Harbor

Uninformed Search Practice

DFS

Do not include duplicate nodes in frontier!

Frontier

~~Brussels~~

Bazaar

Moulinsart

~~Syldavia~~

Port

Harbor

Rocket ★

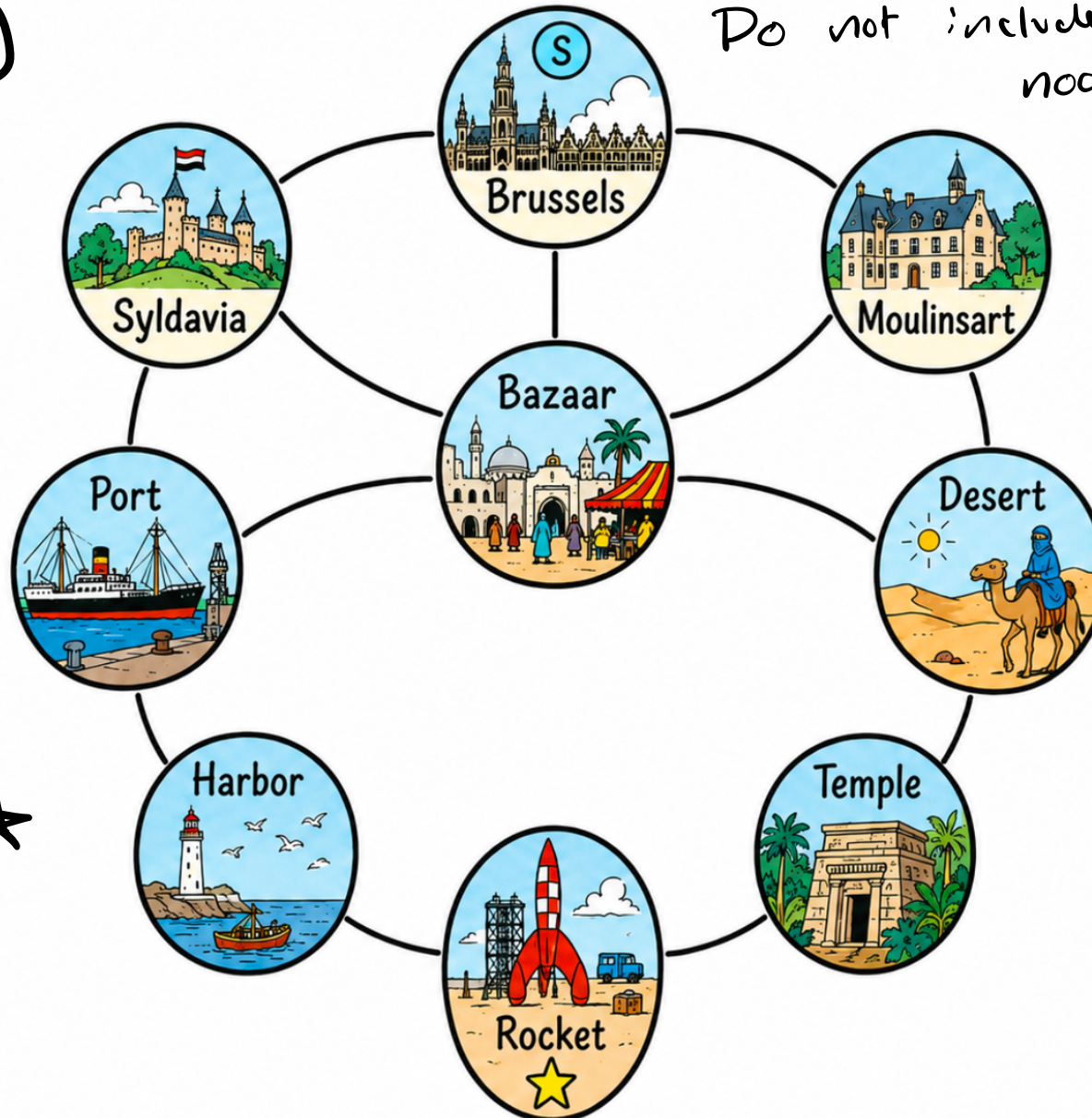
Visited

Brussels

Syldavia

Port

Harbor



Search Conundrum

Breadth-first

- ✓ Complete,
- ✓ Optimal
- ✗ *but* uses $O(b^d)$ space

Depth-first

- ✗ Not complete *unless m is bounded*
- ✗ Not optimal
- ✗ Uses $O(b^m)$ time; terrible if $m \gg d$
- ✓ *but* only uses $O(\mathbf{b*m})$ **space**

Iterative Deepening

Key Idea: Use depth-limited search, but keep increasing the depth limit.

- ✓ Complete

- ✓ Optimal (*if depth limit increased by 1 each time*)

Summary of algorithms

Criterion	Breadth-First	Depth-First	Depth-limited	Iterative deepening
Complete?	YES	NO	NO	YES
Time	b^d	b^m	b^l	b^d
Space	b^d	bm	bl	bd
Optimal?	YES	NO	NO	YES

b = Branching factor
 d = depth of nearest goal
 m = maximum depth of search space
 l = depth limit

Informed Search

Uninformed Search

Uses only information available in problem definition

Informally:

Uninformed search: All non-goal nodes in frontier look equally good

Informed search: Some non-goal nodes can be ranked above others.

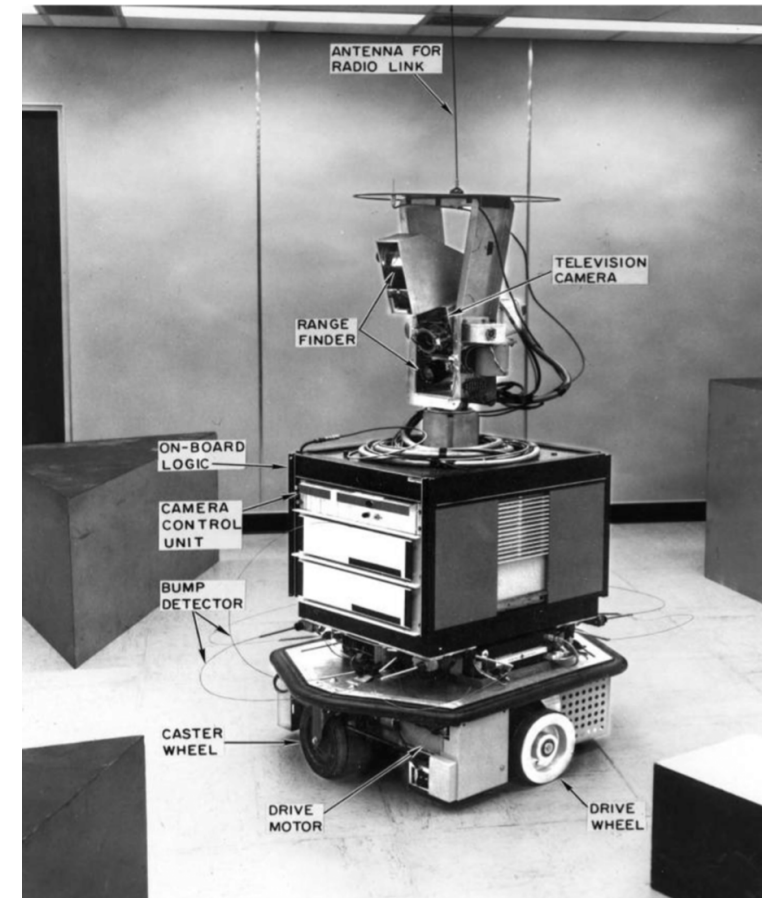
Informed Search

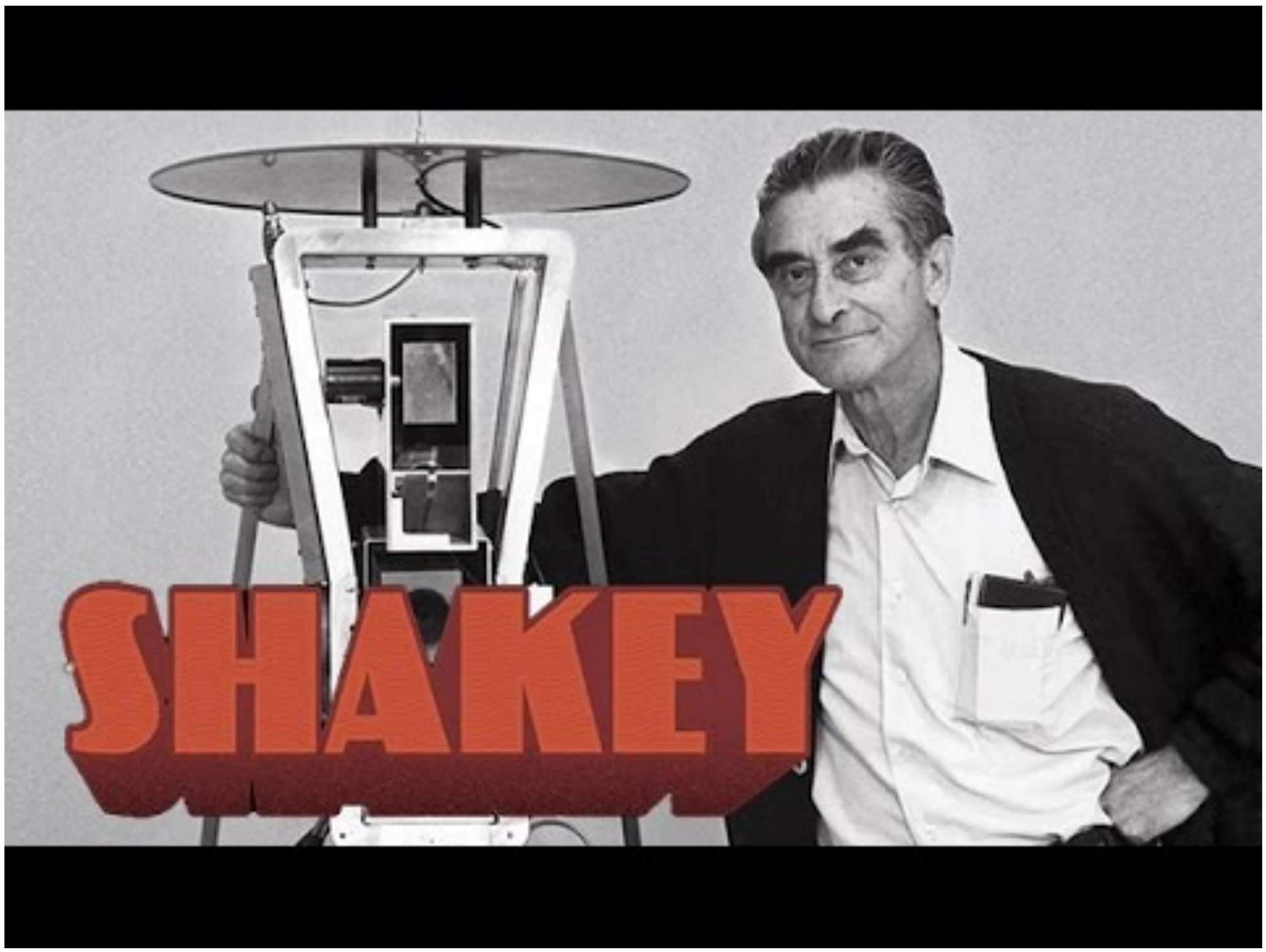
An **informed search** strategy uses **domain-specific information** about the location of the goals in order to find a solution **more efficiently** than uninformed search.

Hints will come as part of a **heuristic function** denoted $h(n)$.

One of the most famous informed search algorithms is **A*** which was developed for **robot navigation**.

Shakey the robot was developed at the Stanford Research Institute from 1966 to 1972.



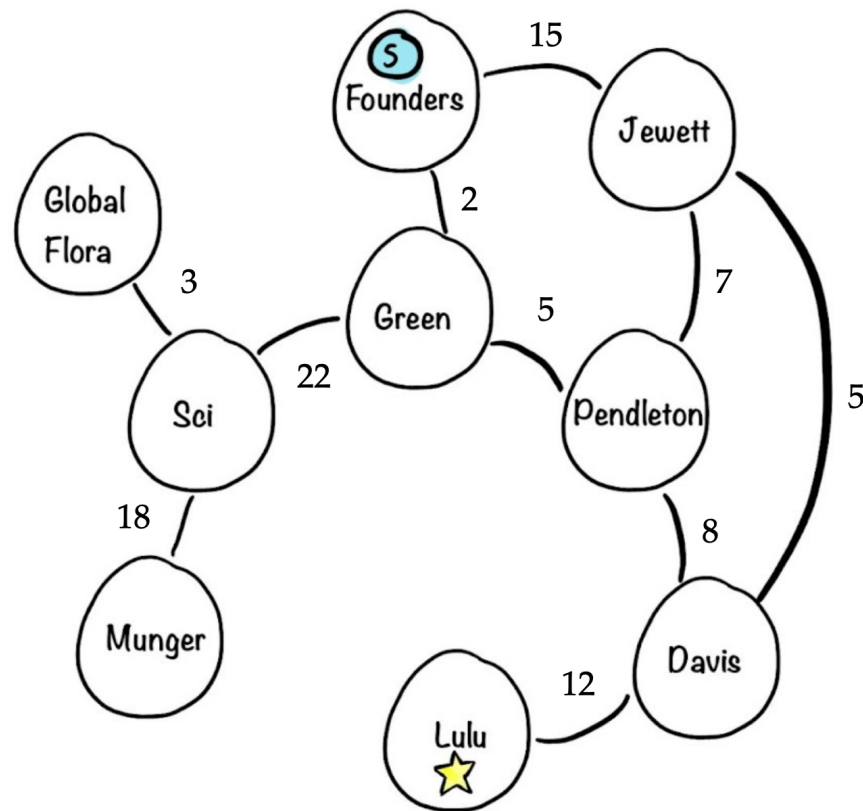


SHAKY

Motivation: Navigation Tasks

So far, we have assumed that all actions have the same cost.

But this isn't true for many applications.



Some of these buildings are much closer than others!

$g(N)$: the Path Cost Function

- **Our assumption so far: All moves equal in cost**
 - Cost = # of nodes in path-1
 - $g(N) = \text{depth}(N)$ in the search tree

$g(N)$: tells us the cost of
reaching N from the start
state

N_0, N_1, N_2 = nodes visited on a path
from N_0 to N_2

$C(i,j)$: Cost of going from Node i to Node j

$$g(N_2) = C(0,1) + C(1,2)$$

$g(N)$: the Path Cost Function

- **Our assumption so far: All moves equal in cost**
 - Cost = # of nodes in path-1
 - $g(N) = \text{depth}(N)$ in the search tree
- More general: Assigning a (potentially) unique cost to each step
 - N_0, N_1, N_2, N_3 = nodes visited on path p from N_0 to N_3
 - $C(i,j)$: Cost of going from N_i to N_j
 - If N_0 the root of the search tree,

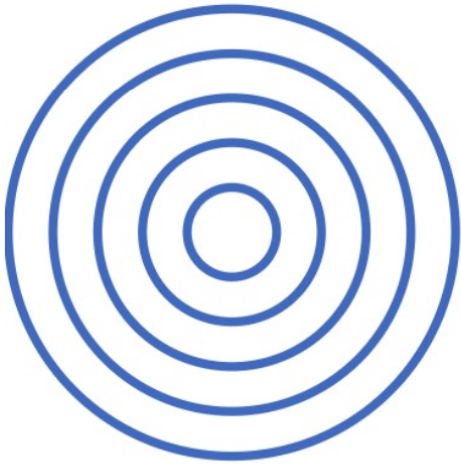
$$g(N_3) = C(0,1) + C(1,2) + C(2,3)$$

Uniform-Cost Search

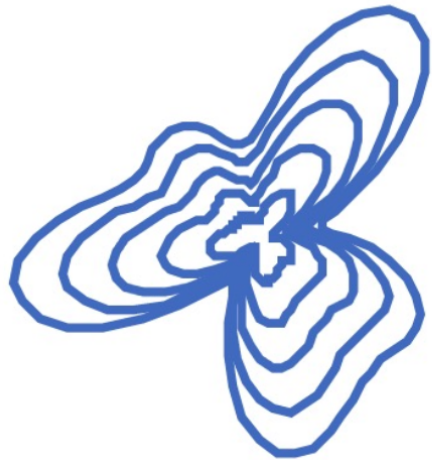
Like to BFS / DFS, but
w/ a different node selection
strategy:

- Expand the node w/ the lowest cost path
- Implementation: priority queue ordered by $\hat{g}(n)$

Shape of Search



- **Breadth First Search** explores equally in all directions. Its frontier is implemented as a FIFO queue. This results in smooth contours or “plys”.



- **Uniform Cost Search** lets us prioritize which paths to explore. Instead of exploring all possible paths equally, it favors lower cost paths. Its frontier is a priority queue. This results in “cost contours”.

A Better Idea

- ◆ Node expansion based on an **estimate** which includes distance to the goal.
- ◆ General approach of informed search:
 - ◆ Best-first search: node selected for expansion based on an **evaluation function $f(n)$**
 - ◆ $f(n)$ includes an estimate of the distance to the goal