CS 232: Artificial Intelligence

Spring 2024

Prof. Carolyn Anderson
Wellesley College

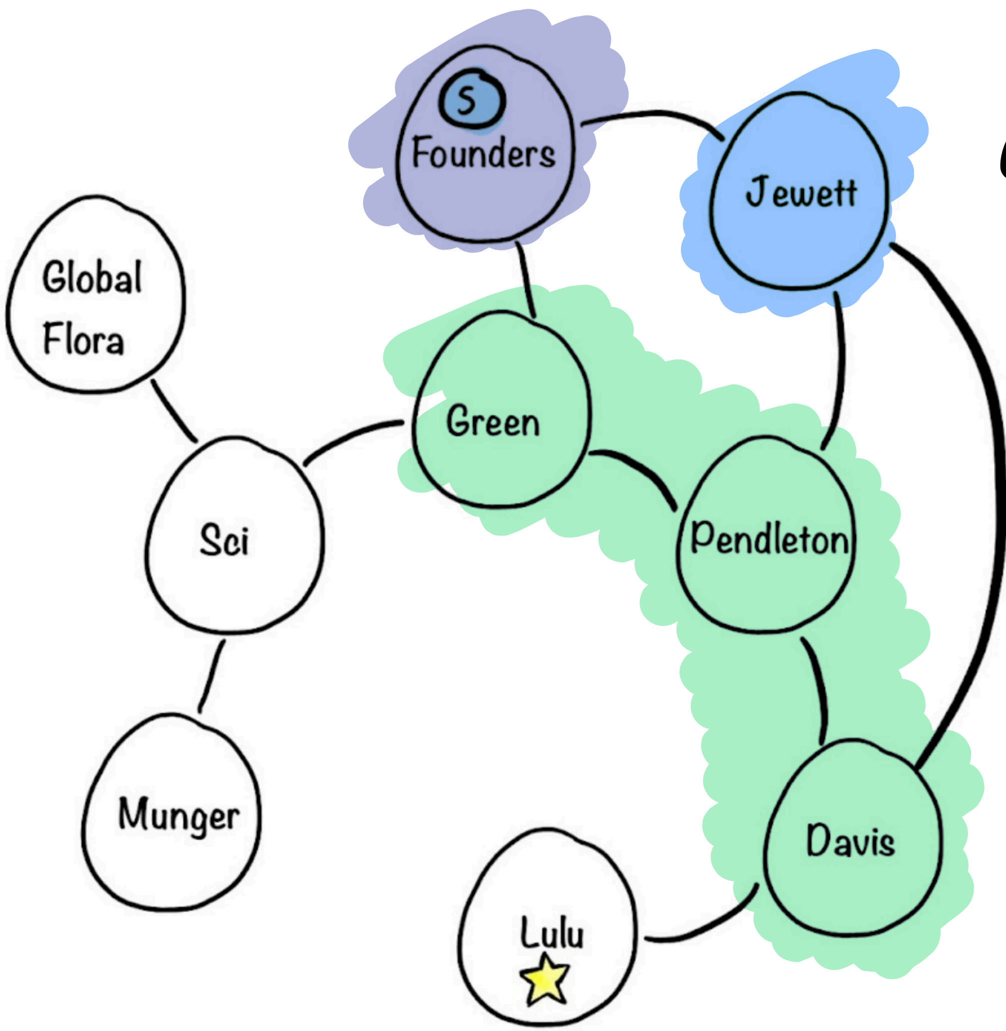
Reminders

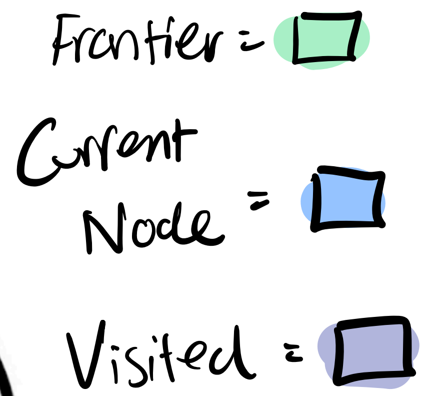
- Homework 3 will be released today
- Lepei has help hours Thursday
- I have help hours Friday

DALL-E3 labeled chocolate cross-section from AI Weirdness blog

Recap

Search Trees





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

Determines the search process

Graph Search

```
function GRAPH-SEARCH(PROBLEM) returns a solution or failure
    initialize the FRONTIER using the start state of the PROBLEM
    initalize the EXPLORED set to be empty
    loop do
        if the FRONTIER is empty:
            return failure
        else:
            choose a leaf node by STRATEGY and remove it from FRONTIER
            if the node contains a goal state:
                return the corresponding solution
            else:
                add node to EXPLORED
                expand the chosen node to get set of children
                for each child:
                    if child not in FRONTIER and child not in EXPLORED:
                        add child to FRONTIER
```

BFS Closest to start — First in first out

DFS Farthest from start — Last in first out

Search Strategies

Review: ***Strategy*** = order of tree expansion

- Implemented by different queue structures (LIFO, FIFO, priority)

Dimensions for evaluation

- ***Completeness***- always find the solution?
- ***Optimality*** - finds a least cost solution (lowest path cost) first?
- ***Time complexity*** - # of nodes generated *(worst case)*
- ***Space complexity*** **- # of nodes simultaneously in memory** *(worst case)*

Time/space complexity variables

- *b*, *maximum branching factor* of search tree
- *d*, *depth* of the shallowest goal node
- *m*, maximum length of any path in the state space (potentially ∞)

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 >> d
- ☑ *but* only uses *O(**b*m) space***

Depth-limited search: A building block

Depth-First search *but with depth limit l.*

- i.e. nodes at depth l *have no successors.*
- No infinite-path problem!

If $l = d$ (by luck!), then optimal

- But:
  - **If $l < d$ then incomplete** ☹️
  - **If $l > d$ then not optimal** ☹️

Time complexity: $O(b^l)$

Space complexity: $O(bl)$ ☺️

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

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.

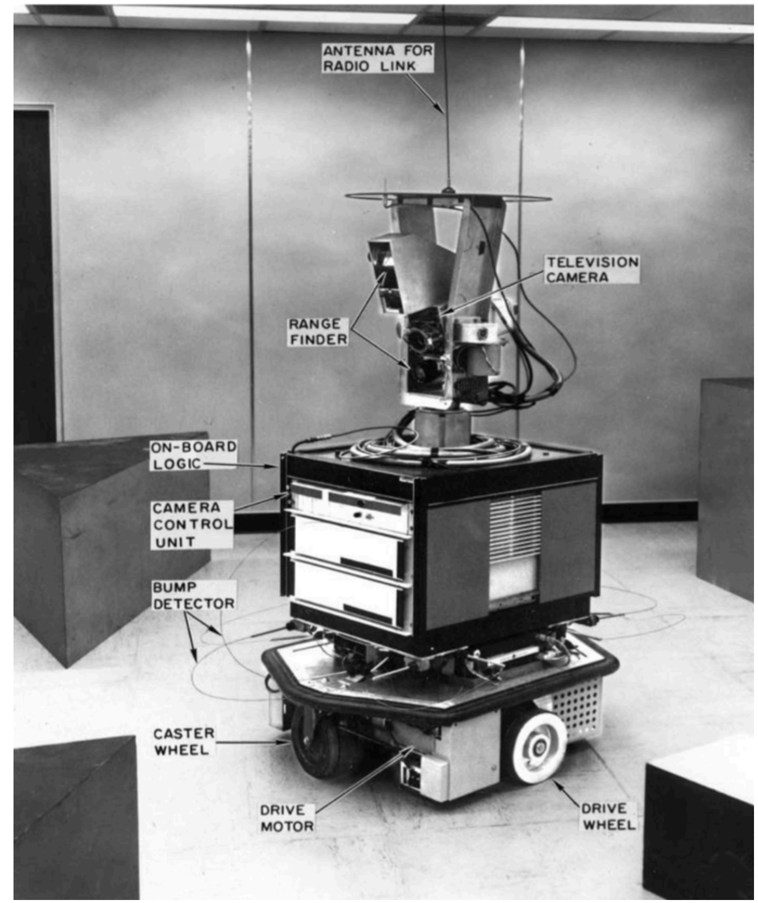
Informed Search

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.

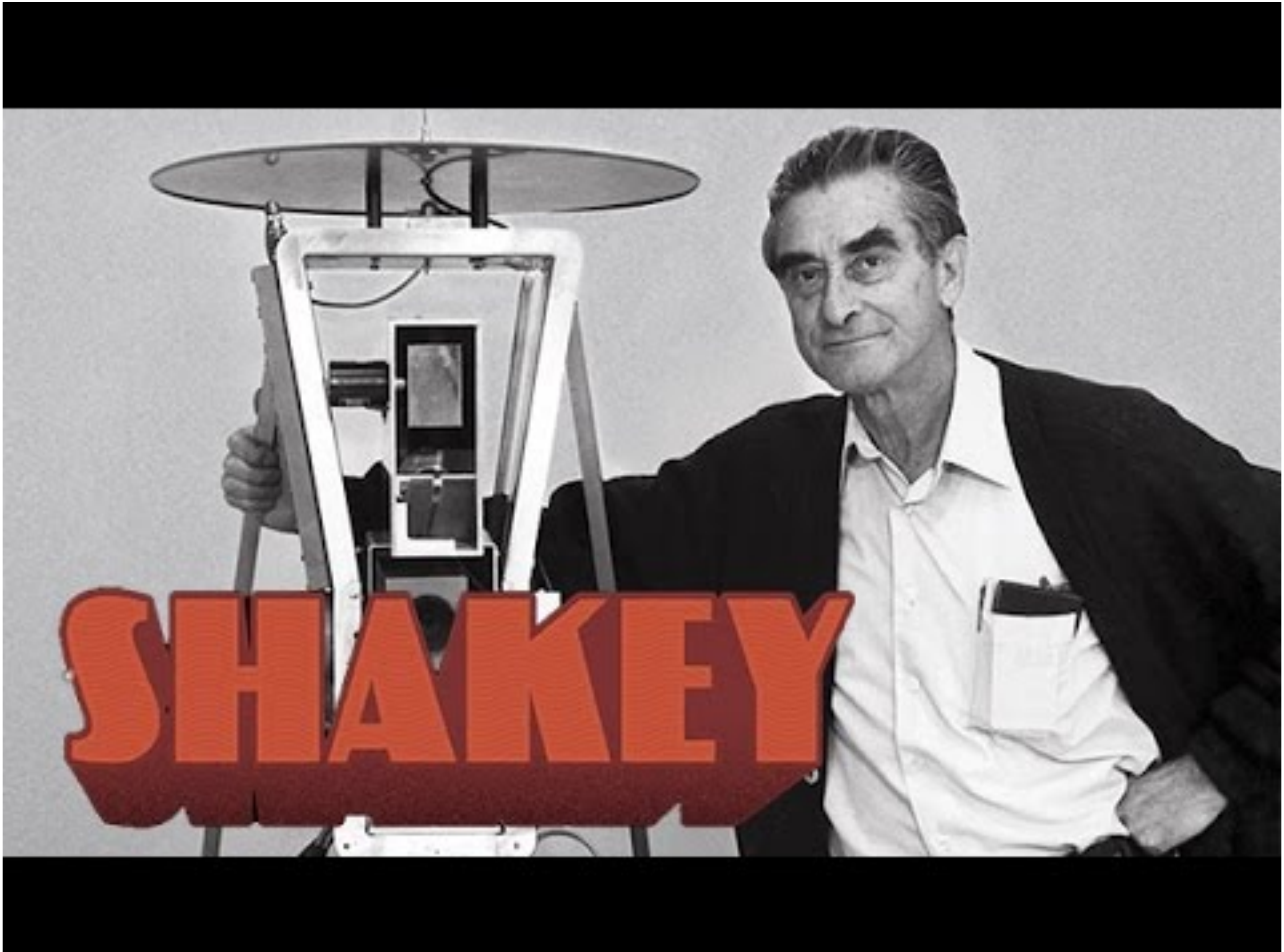
Hints will come as part of a **heuristic function** denoted h(n).

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.



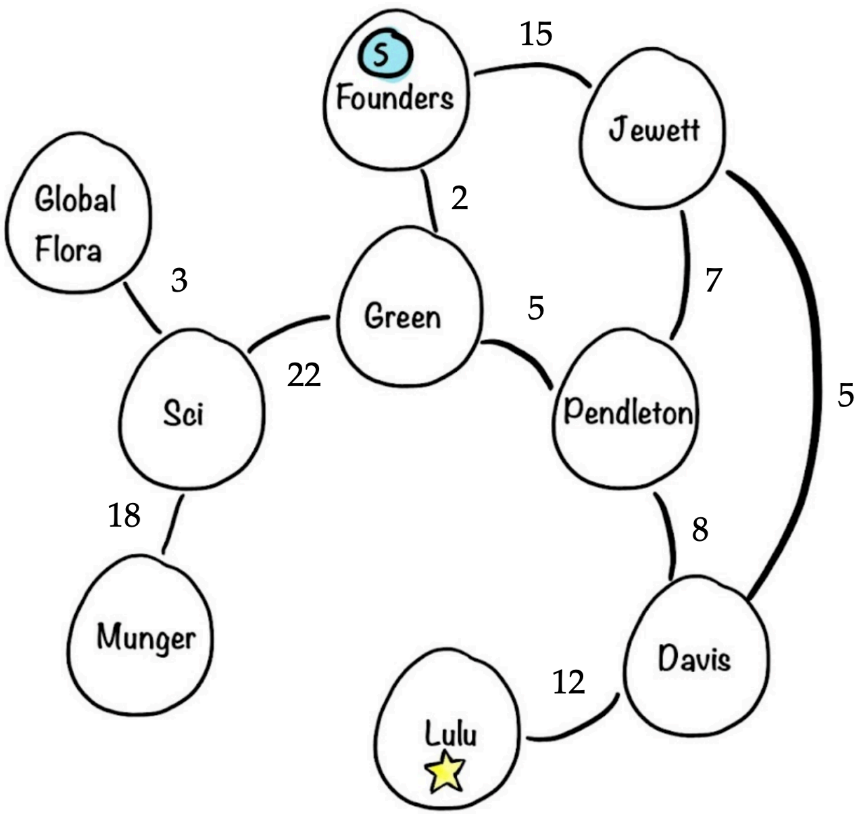
Slides adapted from Chris Callison-Burch


SHAKEY

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

↑ Node

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

C(i,j) = Cost of going from N_i to N_j

C(Fanders, Jewett) = 15

↑ Nodes ↑ Nodes

g(Sci) = C(Fanders, Green) + C(Green, Sci)

If N_0 is the start state, then $g(N_3)$ = C(0,1) + C(1,2) + C(2,3)

Uniform-cost search (UCS)

Extend BFS:

Expand the node with the lowest path cost

Frontier:

priority queue ordered by g(n)

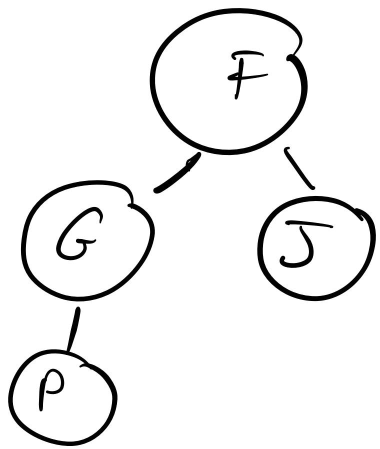
Subtle difference:

Test if a node is the goal when it is <u>expanded</u>

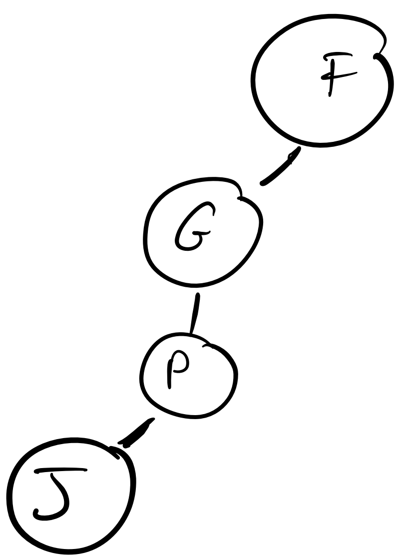
<u>not</u> when it is added to the frontier

Uniform-cost search (UCS)

Before Updating Jewett Cost:



After Updating Jewett Cost:



Frontier

~~Green : 2~~

~~Pendleton : 7~~

~~Jewett : 15 14~~

~~Davis : 15~~

Sci : 24

Lulu : 27

Visited

~~Franklin~~

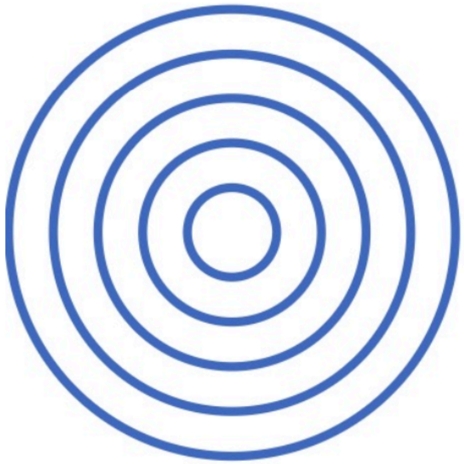
Green

Pendleton

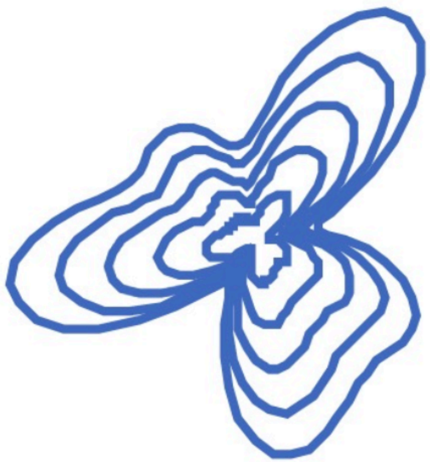
Jewett

Davis

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...

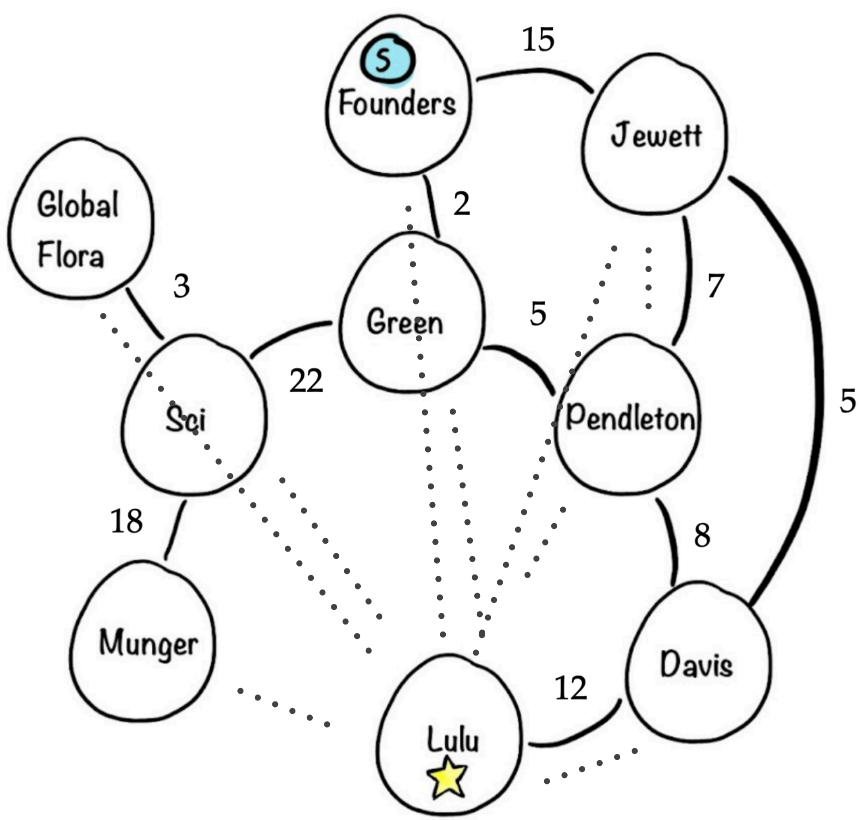
- Node expansion based on *an estimate* which *includes distance to the goal*
- General approach of informed search:
  - *Best-first search*: node selected for expansion based on an *evaluation function* ***f(n)***
    - ✓ ***f(n)*** includes *estimate* of distance to goal ***(new idea!)***
- Implementation: Sort frontier queue by this new ***f(n)***.
  - Special cases: **greedy search**, and ***A* search***

Greedy Best-First Search

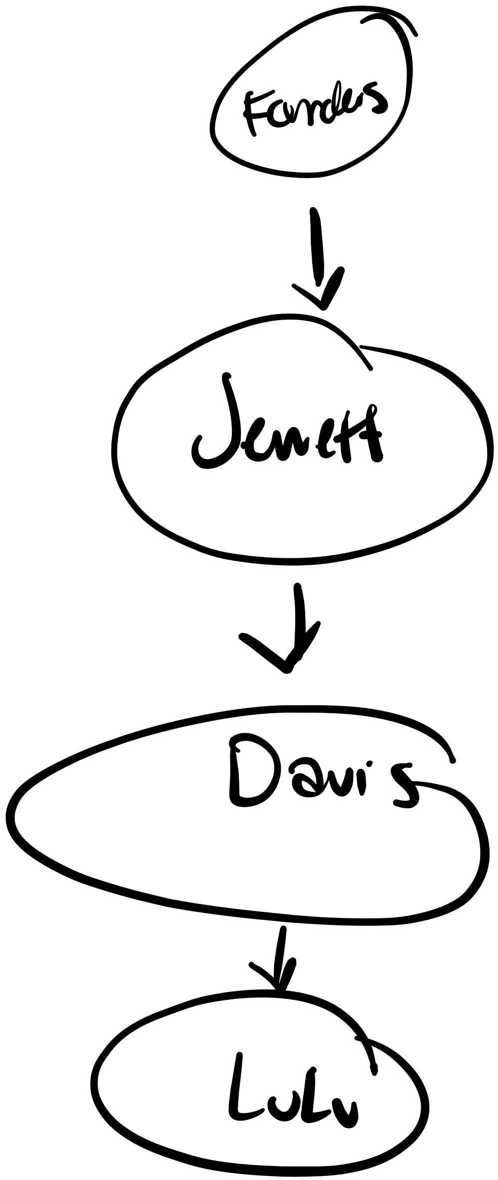
Greedy Best First Search

Idea: expand the node that is **estimated** to be closest to the goal.
Ignore the **actual cost g(n)** and rely totally on the **heuristic h(n).**

In our navigation example, this is like relying on the straight-line distance from a building to the goal (estimated from Google Maps).



Greedy Best-First Search



Frontier	Visited
~~Jewett - 15~~	~~Founders~~
~~Davis - 10~~	Jewett
~~LuLu - 0~~	Davis
Pendleton - 16	LuLu
Green - 20	

Building	Straight-line Distance to LuLu
Davis	10
Founders	30
Global Flora	40
Green	20
Jewett	15
LuLu	0
Munger	25
Pendleton	16
Sci	15

Properties of Greedy Best First Search

Complete? Yes

Optimal? No!

F → J → D → L : 32

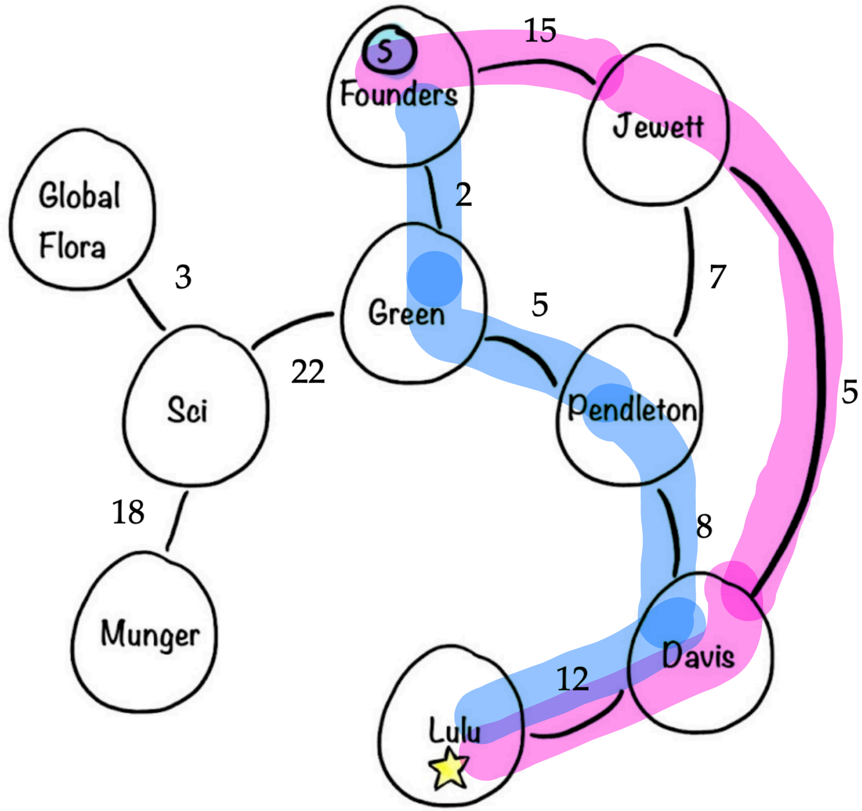
F → J → P → D → L : 42

F → G → P → D → : 27

Properties of Greedy Best First Search

Complete? **YES**

Optimal? **NO!**



Path found:
Founders -> Jewett -> Davis -> LuLu
Cost: 32

Best path:
Founders -> Green -> Pendleton -> Davis ->LuLu
Cost: 27

A* Search

A* search

Key Idea: Avoid expanding paths that are known to be expensive, but expand most promising paths first.

Simple: $f(n) = g(n) + h(n)$

$g(n)$: actual cost so far of going from start to n

$h(n)$: estimated cost of reaching goal from n

$f(n)$: estimated total cost of start to goal through n

Implementation: frontier is a priority queue

Key concept: Admissible heuristics

A heuristic h(n) is admissible if it never overestimates the cost to reach the goal.

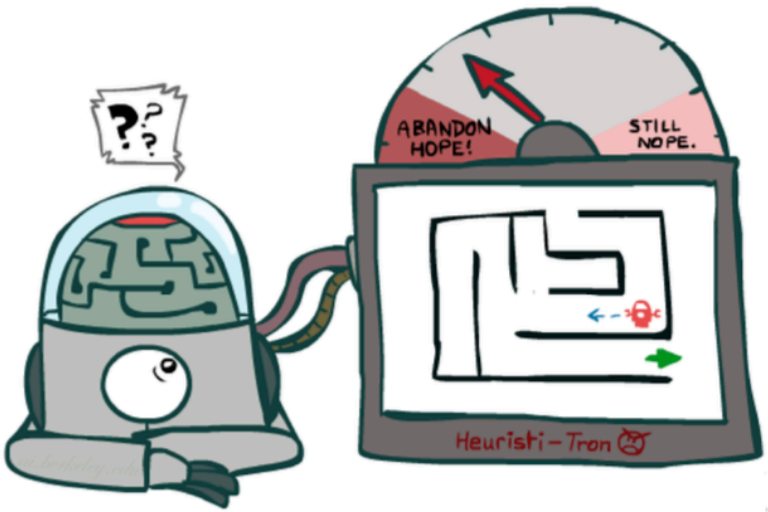
Formally, $\forall n$, n is a node:

$h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from n to goal

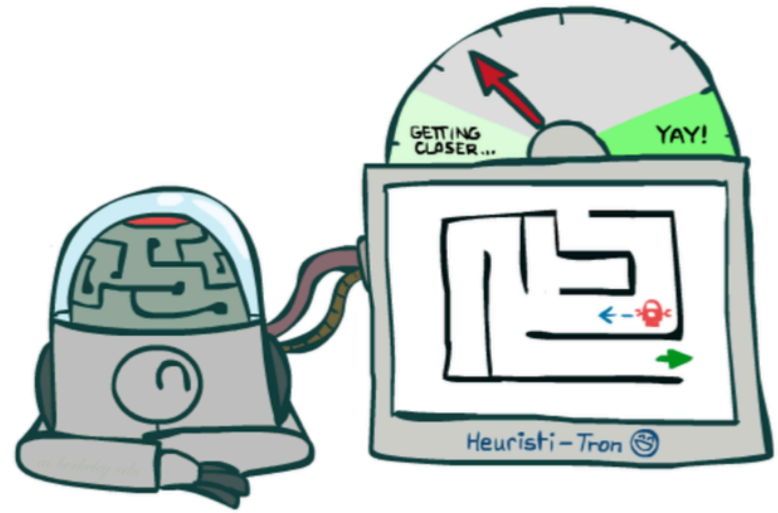
$h(n) \geq 0$ $h(goal) = 0$

If h(n) is admissible, A* is optimal

Idea: Admissibility



Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the frontier



Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs

A* Search Example

Frontier

~~Green: 2+20~~
~~Pendleton: 7+16=23~~
~~Davis: 15+10=25~~
~~Lulu: 27+0~~
Jewett: ~~15~~ 14 + 15 = ~~30~~ 29
Sci: 24+15=39

Visited

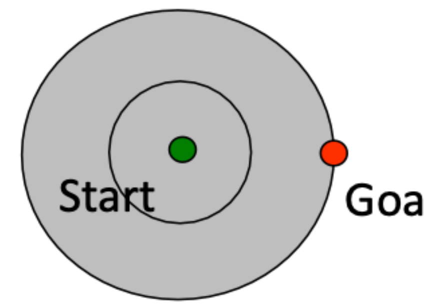
Founders
Green
Pendleton
Davis
LuLu

Founders → Green → Pendleton → Davis → LuLu

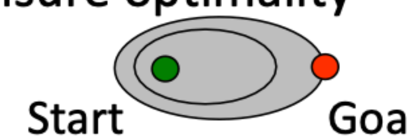
Building	Straight-line Distance to LuLu
Davis	10
Founders	30
Global Flora	40
Green	20
Jewett	15
LuLu	0
Munger	25
Pendleton	16
Sci	15

UCS vs A* Contours

Uniform-cost expands equally in all "directions"



A* expands mainly toward the goal, but does hedge its bets to ensure optimality



A* Applications

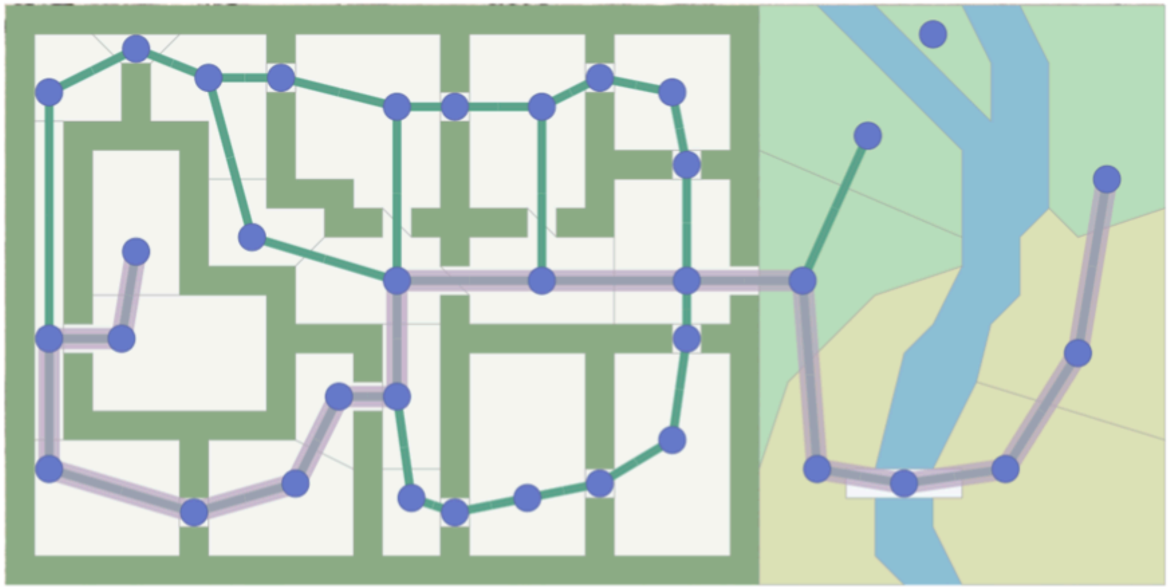
Pathing / routing problems (A* is in your GPS!)

Video games

Robot motion planning

Resource planning problems

…



Heuristics

Heuristic Functions

For the 8-puzzle

- **Avg. solution cost is about 22 steps**
    - **(branching factor ≤ 3)**
- **(branching factor ≤ 3)**
- **A good heuristic function can reduce the search process**

Slides adapted from Chris Callison-Burch

Admissible Heuristics

For the 8-puzzle:

$h_{oop}(n)$ = number of out of place tiles

$h_{md}(n)$ = total Manhattan distance (i.e., # of moves from desired location of each tile)

$h_{oop}(S) = 8$

$h_{md}(S) = 3+1+2+2+2+3+3+2 = 18$

Key: Admissibility

Inadmissible (pessimistic) heuristics break optimality by pushing good plans too far back on the frontier, which means they may never get expanded.

Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs. That means that the true best plan will always be expanded.