Search

CPSC 383: Explorations in Artificial Intelligence and Machine Learning
Fall 2025

Jonathan Hudson, Ph.D
Associate Professor (Teaching)
Department of Computer Science
University of Calgary

August 27, 2025

UNIVERSITY OF CALGARY

Outline

- Knowledge processing
- Search vs Computation
- Search
- Search Component Definitions
- Examples
- Graphs and Trees
- Uniformed Search

Knowledge Processing

Knowledge Processing in general

- Task: use knowledge represented in system plus new knowledge and produce a result:
  - Add knowledge to knowledge base
  - Find inconsistencies in knowledge base
  - Answer user question

☞ make implicit knowledge explicit

- Approaches:
  - Search (produce a certain result or new consistent knowledge base)
  - Apply procedural knowledge (computation)

General Problems

- What parts of the knowledge base are needed?
- What parts of the knowledge base must be changed (frame problem)?
- What pieces of knowledge are applicable?
- What concrete piece of knowledge to choose next?

Search versus Computation

Search versus Computation

- Deep down in our computers everything is a computation

- On higher levels, there are different computation processes:

  1. Processes where each step is always necessary to achieve their goals
     ☞ computation

  2. **Processes where after they finished you can identify steps that did not contribute to achieving the goals**
     ☞ search

UNIVERSITY OF CALGARY

What does computation offer?

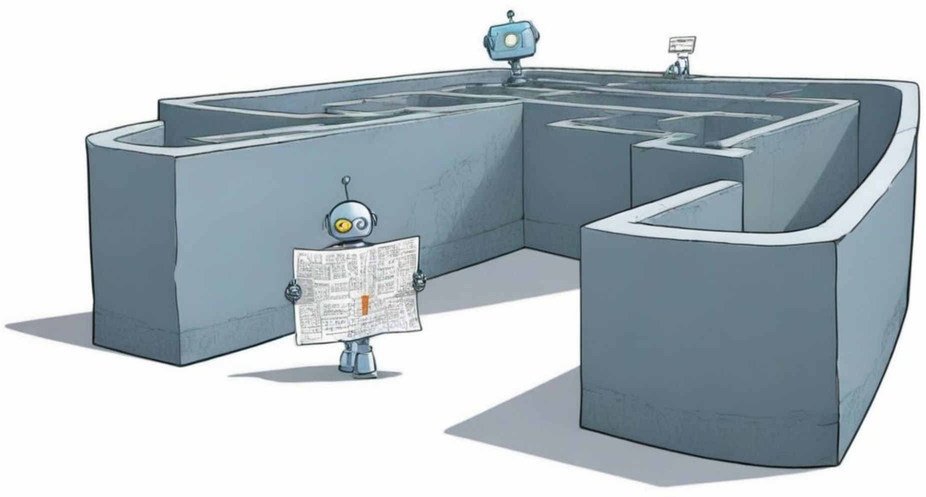
- Usually run time is predictable
- No dealing with choices
- No unnecessary steps
- Implicit knowledge representation
  - ☞ difficult to know what is going on
- Not always possible to achieve
  - ☞Nice to have, but in AI systems often not possible

Search

Search: Basic Definitions

Search is at the core of many systems that seem to be intelligent

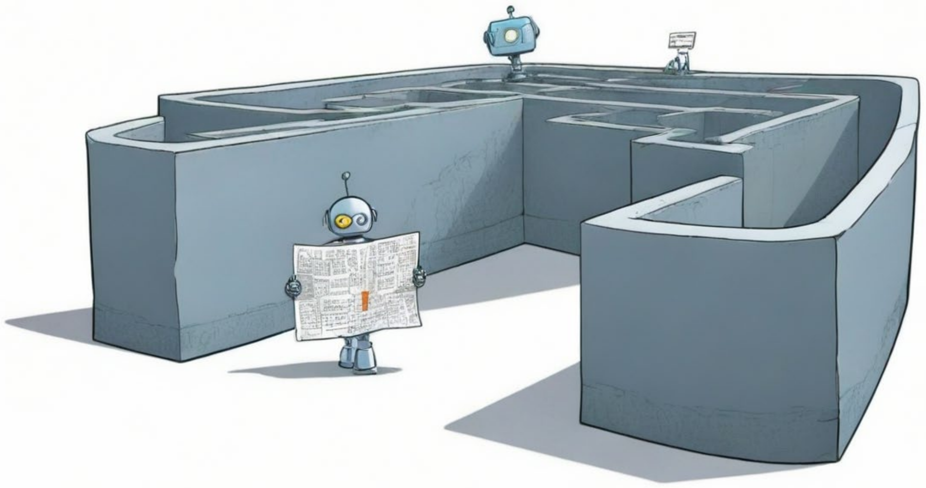
- Learning: search for a structure that explains/ predicts/justifies some experiences
- Planning: search for a series of decisions that best achieves a goal while fulfilling certain conditions
- Deduction: search for a justification for a certain fact
- Natural language understanding: search for the best interpretation of a text
- …



Search: Basic Definitions

Search is at the core of many systems that seem to be intelligent

- Learning: search for a structure that explains/ predicts/justifies some experiences
- Planning: search for a series of decisions that best achieves a goal while fulfilling certain conditions
- Deduction: search for a justification for a certain fact
- Natural language understanding: search for the best interpretation of a text
- …



How is "intelligence" achieved?

- By defining a good search model
- By finding good controls for search processes

But: do not expect your system to be good for every problem instance it can theoretically solve!

No free lunch theorem:

For every search system there is a search instance that shows the worst case behavior

Definitions

Basic Definitions (I)

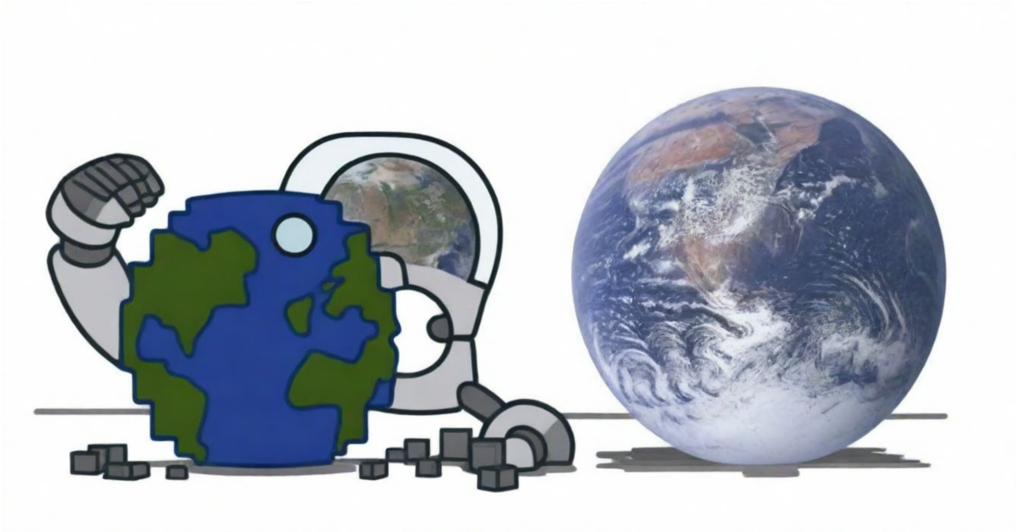
Search Model $A = (S, T)$

S	set of possible states
T	transitions between states

☞

- Defines main data structure and possibilities (space)
- Tells us what the control can work with
- Limits the choices of the control

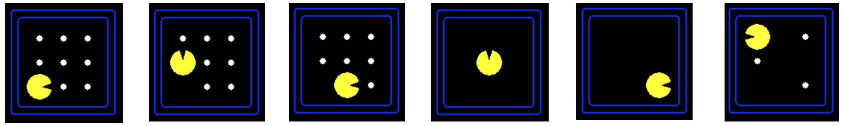
Search Problems Are Models



Basic Definitions (I)

Search Model $A = (S, T)$

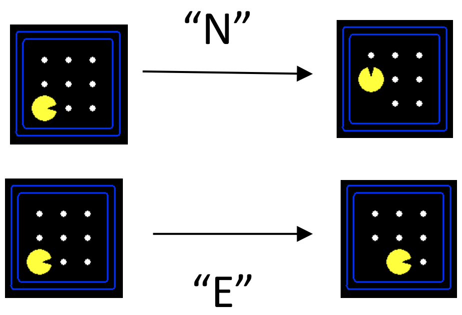
S	set of possible states
T	transitions between states



☞

- Defines main data structure and possibilities (space)
- Tells us what the control can work with
- Limits the choices of the control

Two transitions from state 1



UNIVERSITY OF CALGARY

Basic Definitions (II)

Search Process $P = (A, Env, K)$

A	search model
Env	environment of process *(sometimes your configuration of algorithm)*
K	search c(K)ontrol is a function transitioning from current state to next state

☞

- Defines how to deal with indeterminism of search model.
- Has to deal with all possible states and all searches you want to perform

UNIVERSITY OF CALGARY

Basic Definitions (III)

Search Instance $Ins = (s_0, G)$:

s_0 start state for the instance

$G(state) \rightarrow \{yes, no\}$ goal condition (function on current state that halts)

☞

- Defines concrete input for a search run
- Defines when search ends (positively)
- Normally is generated out of user input

Basic Definitions (IV)

Search Derivation:

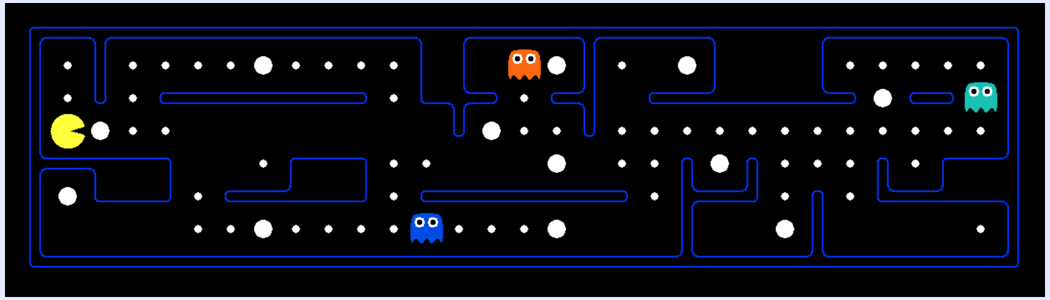
P applied on Ins leads to a sequence of states

- Needed to analyze quality of search control
  - distinguish between necessary and unnecessary steps
  - compare with shortest possible sequence of states that leads to a solution
- Might be looked at to determine solution

Examples

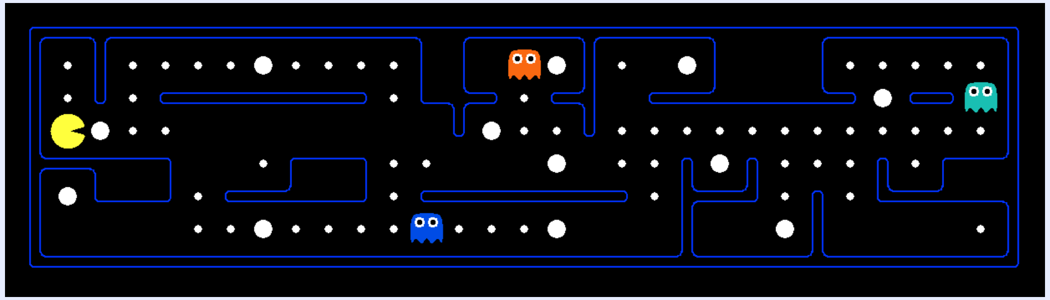
What's in a State Space?

The world state includes every last detail of the environment



What’s in a State Space?

The world state includes every last detail of the environment



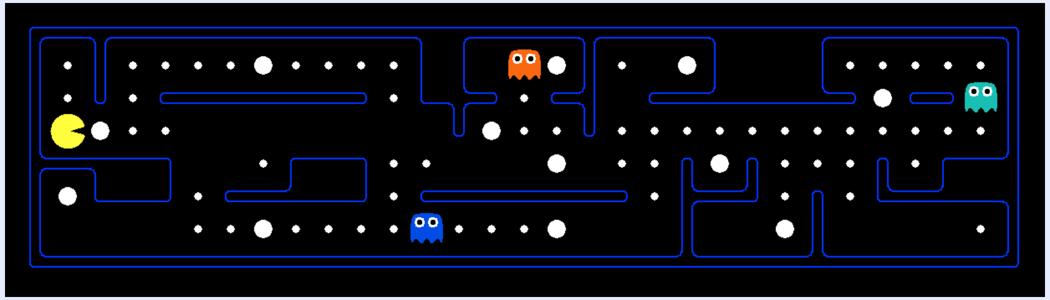
A search state keeps only the details needed for planning (abstraction)

- Problem: Pathing
  - States: (x,y) location (make up S)
  - Actions: NSEW (help us decide T)
  - Successor: update location only (make T)
  - Goal test: is (x,y)= END (make G)

UNIVERSITY OF CALGARY

What's in a State Space?

The world state includes every last detail of the environment



A search state keeps only the details needed for planning (abstraction)

- Problem: Pathing
  - States: (x,y) location (make up S)
  - Actions: NSEW (help us decide T)
  - Successor: update location only (make T)
  - Goal test: is (x,y)= END (make G)
- Problem: Eat-All-Dots
  - States: {(x,y), dot booleans}
  - Actions: NSEW
  - Successor: update location and possibly a dot boolean
  - Goal test: dots all false

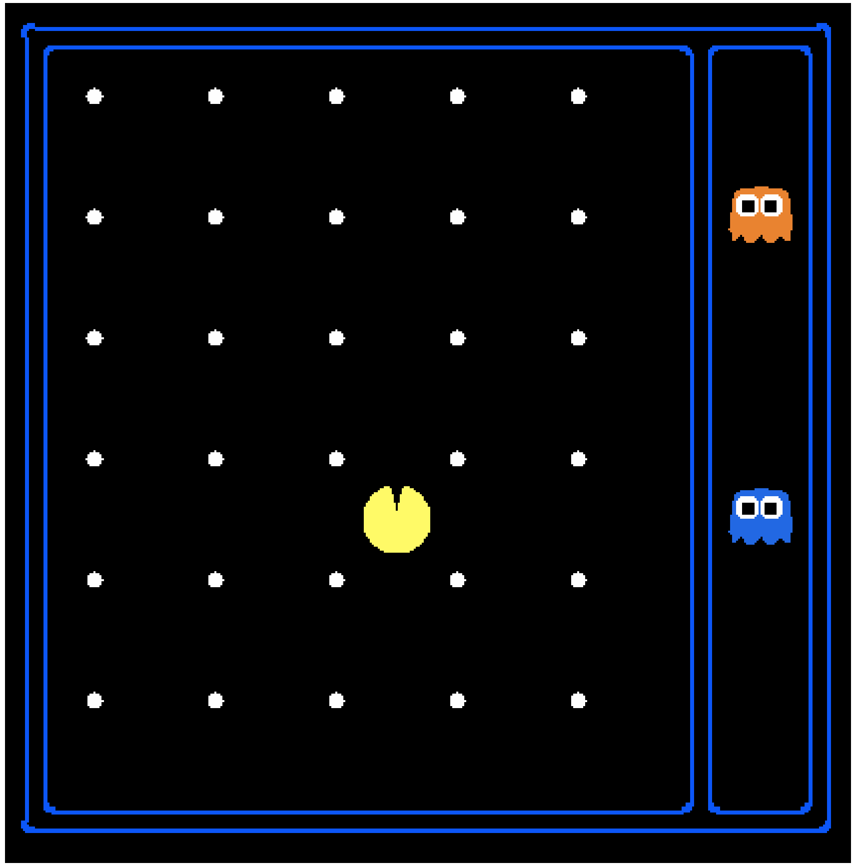
State Space Sizes?

- World state:
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW

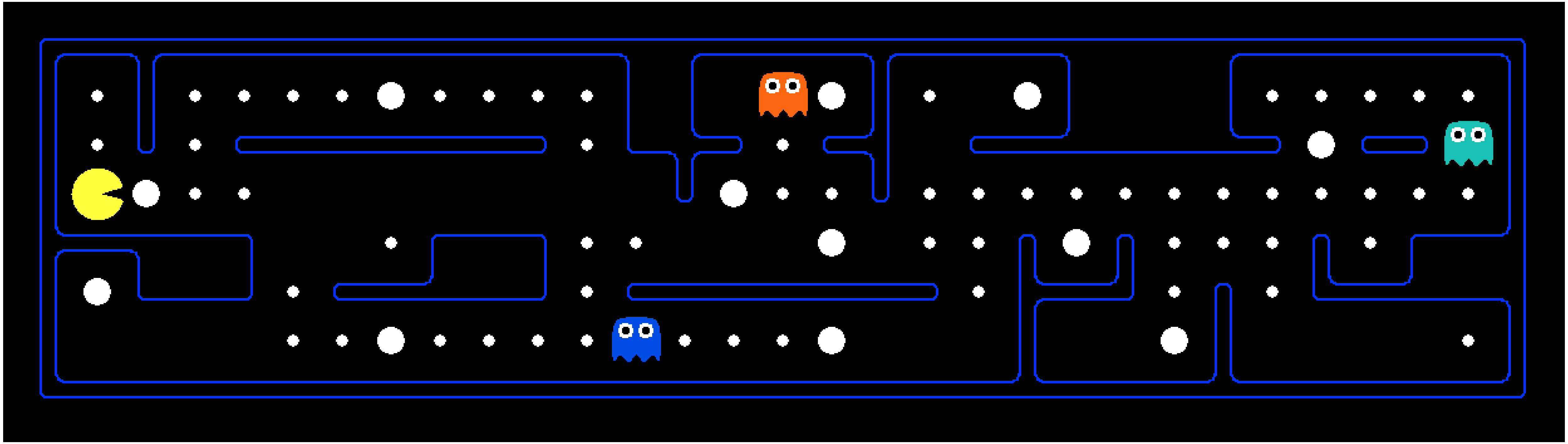
- How many
  - World states?
    $120x(2^{30})x(12^2)x4$
  - States for pathing?
    120
  - States for eat-all-dots?
    $120x(2^{30})$

12 x 10 grid (dot and spaces)

._._._._._ is one row (10 spots)



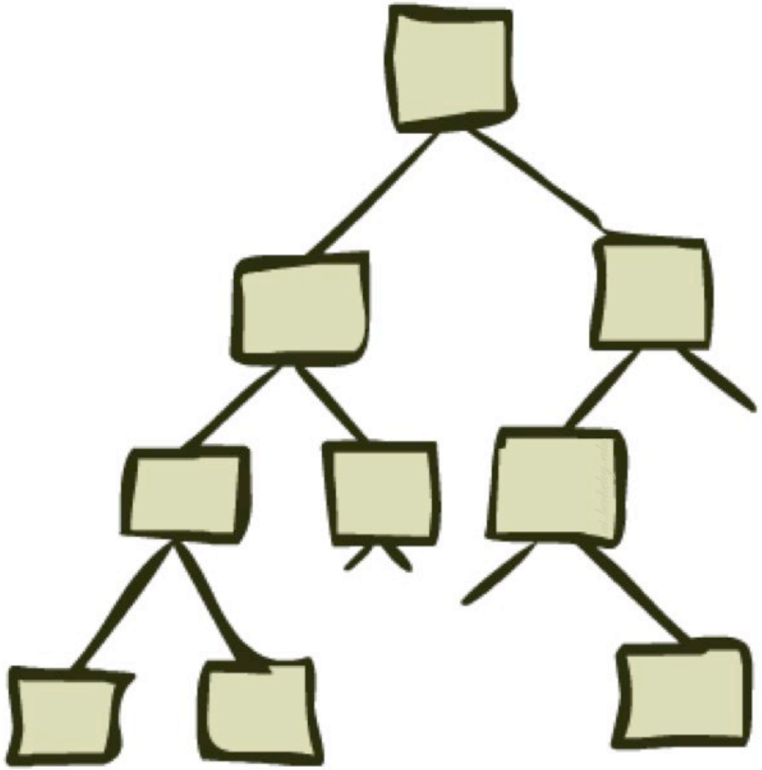
Safe Passage



- Problem: eat all dots while keeping the ghosts perma-scared
- What does the state space have to specify?
  - (agent position, dot booleans, power pellet booleans, remaining scared time)

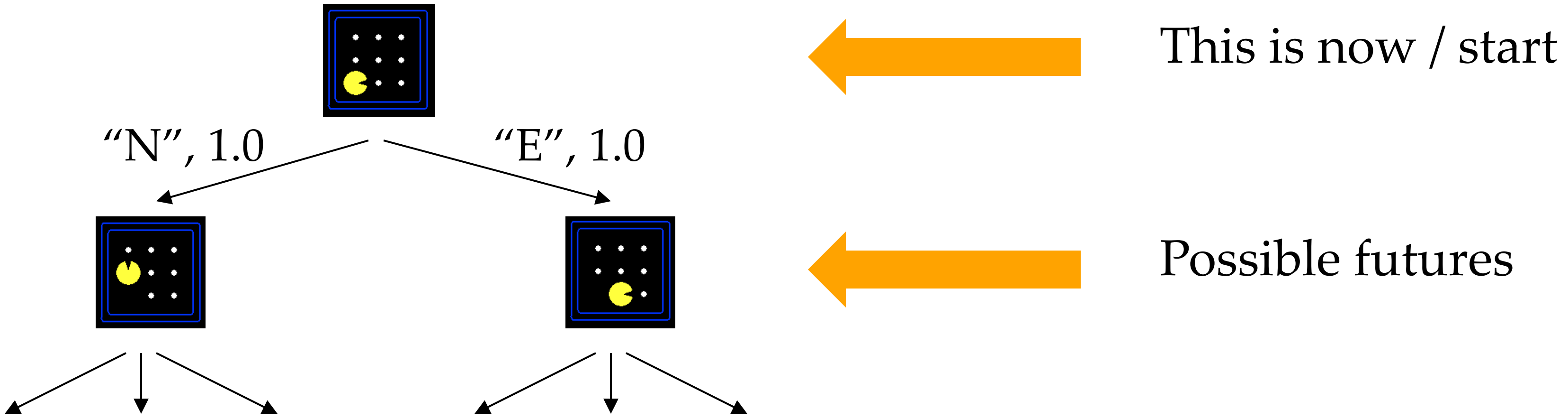
Graphs? Trees?

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

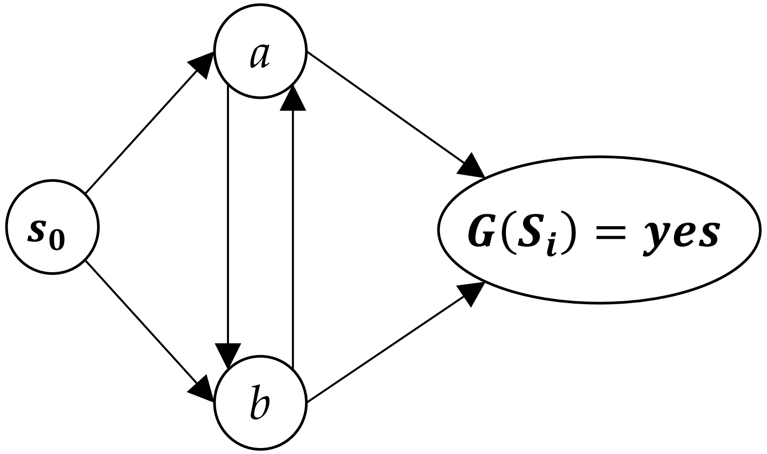
Search Trees



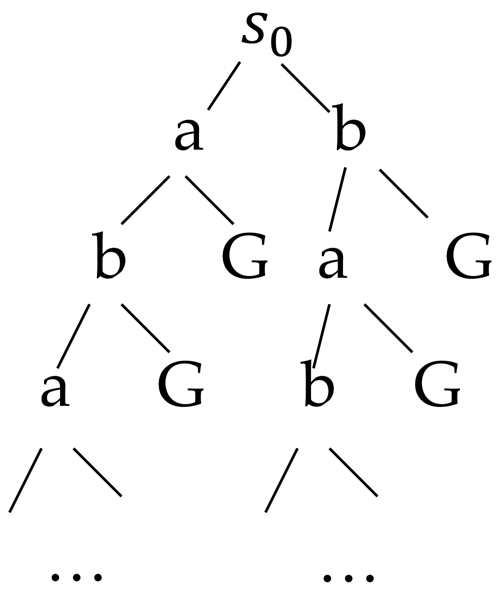
- A search tree:
  - A “what if” tree of plans and their outcomes
  - The start state is the root node
  - Children correspond to successors
  - Nodes show states, but correspond to PLANS that achieve those states
  - For most problems, we can never actually build the whole tree

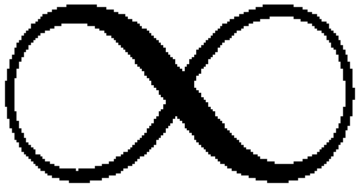
State Space Graphs vs. Search Trees

Consider this 4-state graph:



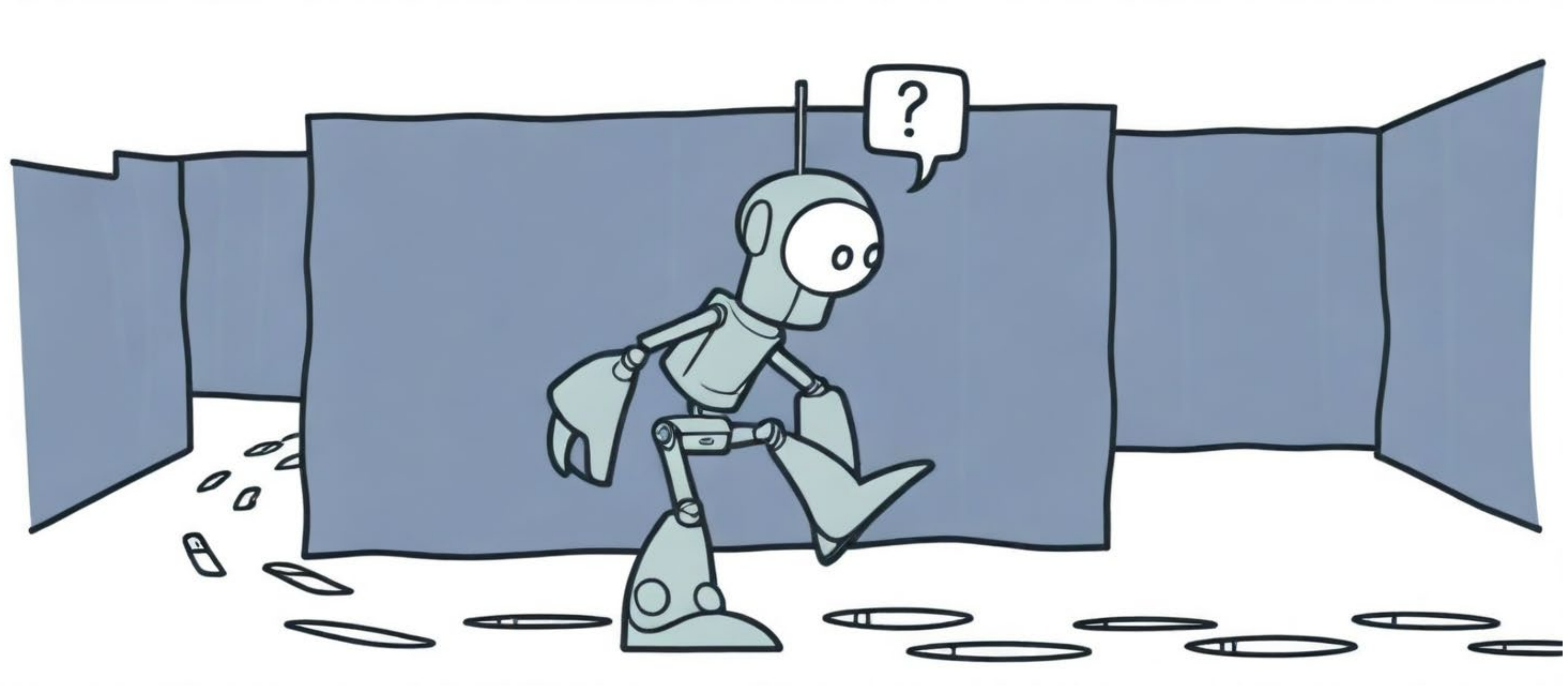
How big is its search tree (from S)?





Important: Lots of repeated structure in the search tree!

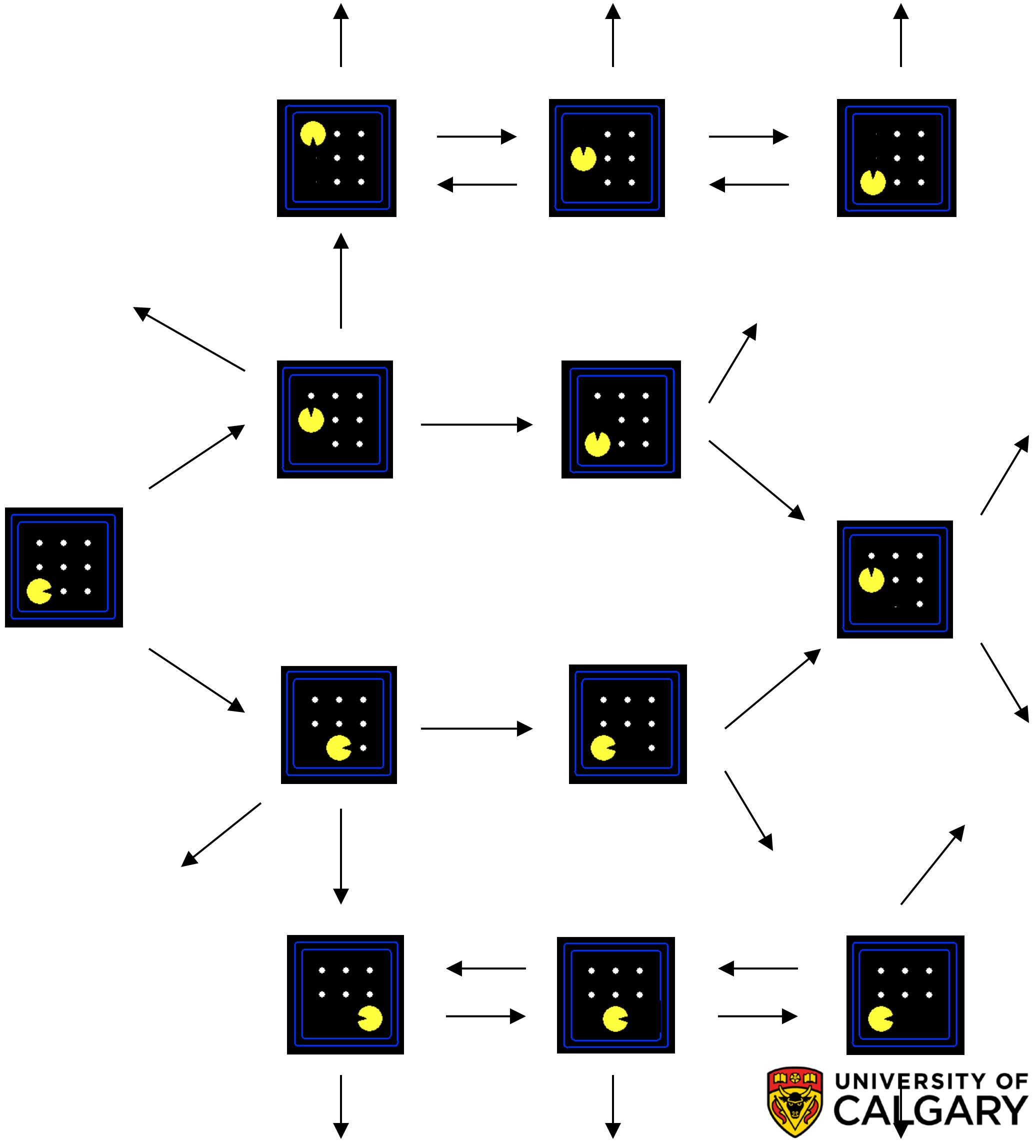
Graph Search



UNIVERSITY OF CALGARY

State Space Graphs

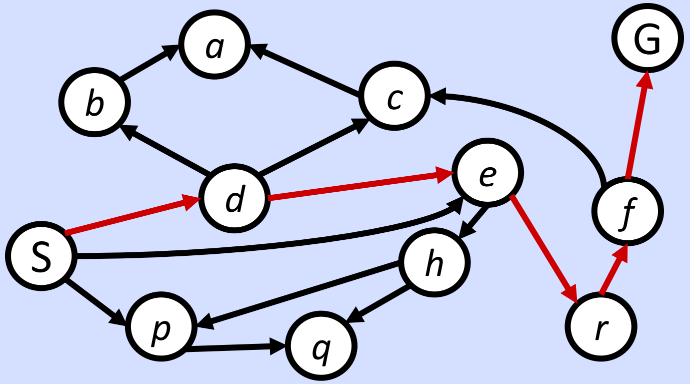
- In a state space graph, each state occurs only once!
- We can rarely build this full graph in memory (it's too big), but it's a useful idea



UNIVERSITY OF CALGARY

State Space Graphs vs. Search Trees

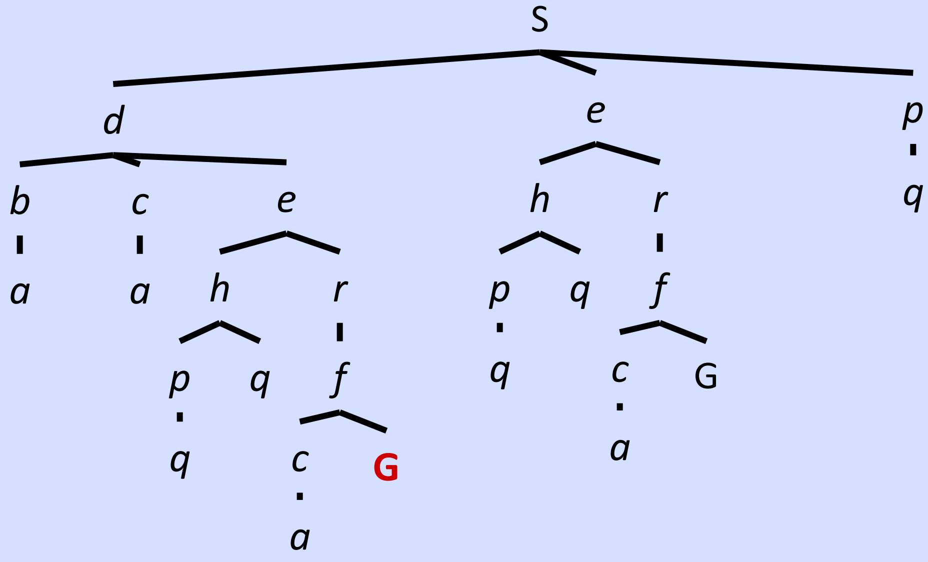
State Space Graph



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



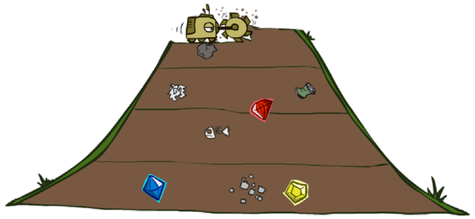
Uniformed Search

Search strategies

- A K(control) is defined by picking the order of node expansion for the derivation
- Strategies are evaluated along the following dimensions:
  - completeness—does it always find a solution if one exists? (can we trust it!)
  - optimality—does it always find a least-cost solution? (is it the best!)
  - time complexity (how long!)
  - space complexity (how taxing on my storage!)
- Time and space complexity are measured (for trees which are most common)
  - b—maximum branching factor of the search tree
  - d—depth of the least-cost solution
  - m—maximum depth of the state space (may be ∞)

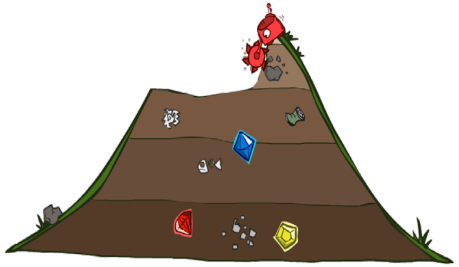
Uniformed Search strategies

Uninformed strategies use only the information available in the problem definition



- Breadth-first search
  - Explore level closest to root first

- Depth-first search
  - Explore farthest from root first



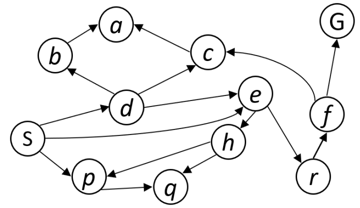
- Uniform-cost search
  - Explore lowest cost region next

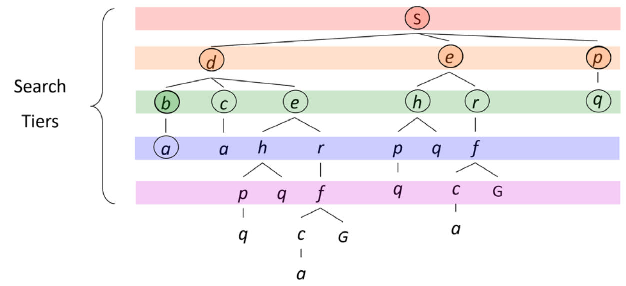
UNIVERSITY OF CALGARY

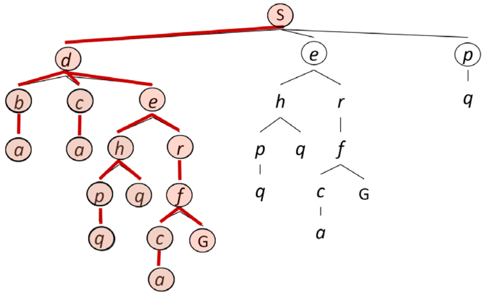
Uniformed Search strategies

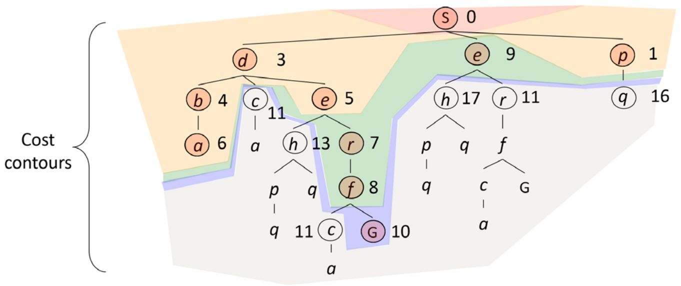
Uninformed strategies use only the information available in the problem definition

- Breadth-first search
  - Explore level closest to root first
  - **stores a lot of state, waste time on early**
  - **but complete**
- Depth-first search
  - Explore farthest from root first
  - **less state, may get lost in infinite trees**
- Uniform-cost search
  - Explore lowest cost region next
  - **medium state, could get lost in infinite trees depending on cost measure**

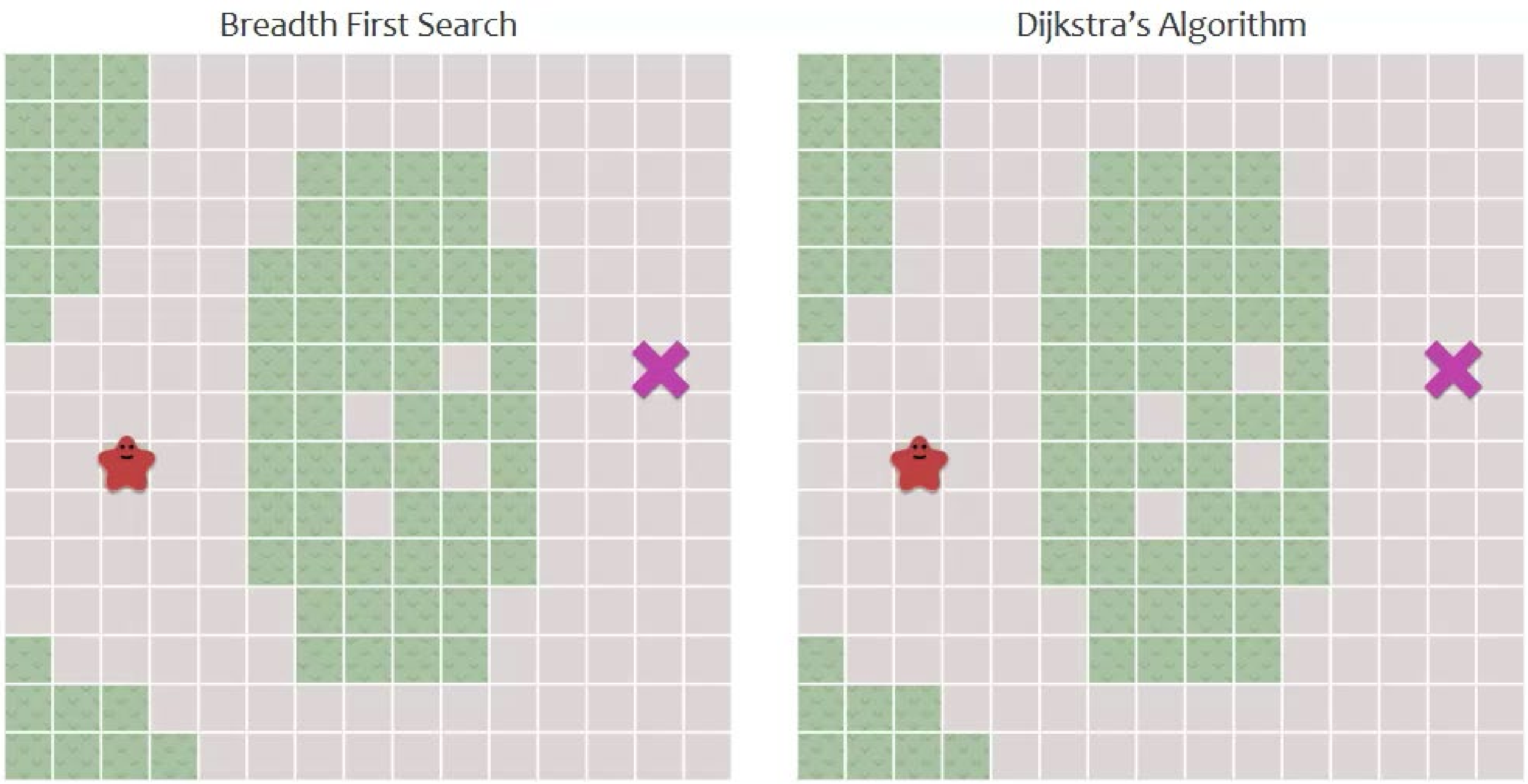








Uniformed Search strategies



Uniformed Search strategies

- Depth-limited search
  - depth to level L
  - **least state, won't get lost in infinite trees**
  - **might not go deep enough to find answer**

- Iterative deepening search
  - Do depth limited to increasing depth level L
  - **less state, more time, 1 to infinity**

Summary

Summary

- Search is a process where after they finished you can identify steps that did not contribute to achieving the goals
- Intelligence Achieved by
  - defining a good search model
  - finding good controls for search processes
- There is always a worst case
- You need to define a Model and Process, you start these on an Instance which creates a Derivation (history of you search)
- Search spaces are often really really large (This is fundamental problem)
- Can use graphs and trees to manage search space exploration
- Uniformed search (structure of problem only) includes breadth, depth, and simple variants of these two

Next…path-finding

Jonathan Hudson, Ph.D.
jwhudson@ucalgary.ca
https://cspages.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY