Search

CPSC 433: Artificial Intelligence
Fall 2024

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

August 8, 2024

UNIVERSITY OF CALGARY

Outline

- Knowledge processing
- Search vs Computation
- Search
- Search Component Definitions
- Examples
- Graphs and Trees
- Problems that need solving when designing Search solution

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

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

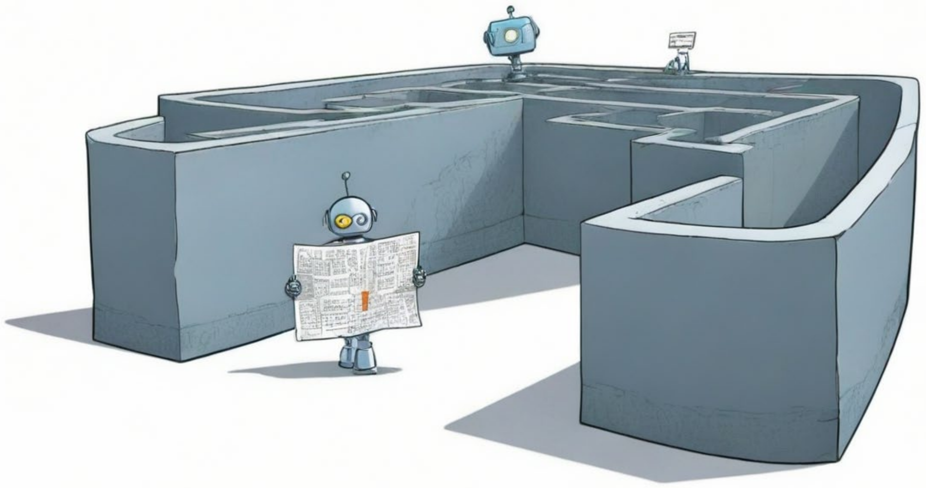
Search

Search Problems

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 (or that comes very near to it)
- 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
- …



UNIVERSITY OF CALGARY

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

General search knowledge

In this course we are going to first define a **general search paradigm**

- A bunch of formalization of the fundamental pieces we need for a search
- We'll discuss which pieces we need for this in the remainder of this slide deck

Then we will introduce **3 sub-variants** of this general search paradigm

- **Set-based search, And-tree-based search, Or-tree-based search**
- Each of these come with a **basic core** design that fulfills the **general search knowledge** requirement
- They also each come with some simpler components into which you fit **application knowledge** to make that search work for your specific problem

3 sub-variants

Set-based Search

- Good for local search (low impact on space needs)
- Good for greedy solutions like hill-climbing solutions (simple algorithms that don't need history)
- Often lose guarantee of optimization but often can find good solutions fast

And-tree-based Search

- Good for optimization problems
  - Structure an exhaustive search for all options and then return the optimal option
  - Tree can be bounded (pruned) (branch-and-bound algorithms – CPSC 413)
- Good for problems where you need to solve all sub-problems and combine them
- Take a lot of space and computation (but that's how we get optimal results)

Or-tree-based Search

- Good for finding one valid solution (like hard constraint satisfaction), but unlike set based search designed so we can keep a history so that we don't repeat steps when one path of search fails, less space than and-tree generally

Model

Basic Definitions (I)

Search Model

$A = (S, T)$

S set of possible states

$T \subseteq S \times S$ 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)

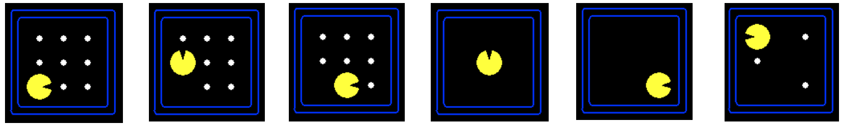
Search Model

$A = (S, T)$

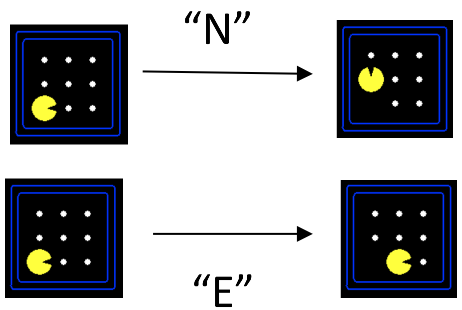
S set of possible states

$T \subseteq S \times S$ transitions between states

$S = \{s_1, s_2, s_3, s_4, s_5, s_6\}$



Two transitions from state 1



$(s_1, s_2) \in T$

$(s_1, s_3) \in T$

Process

Basic Definitions (II)

Search Process

$P = (A, Env, K)$

A search model

Env environment of process

(sometimes your configuration of algorithm)

$K\colon S \times Env \rightarrow S$ search control is a function K transitioning from current state to next state (based on possible additional environment input)

$K(s, e) = s'$ where $(s, s') \in T, e \in Env$

☞

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

Instance

Basic Definitions (III)

Search Instance

$Ins = (s_0, G)$

$s_0 \in S$ start state for the instance

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

$G(s_i) = result$ where $s_i \in S$, and result is **yes** if search is done, **no** otherwise

☞

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

Derivation

Basic Definitions (IV)

Search Derivation:

$\boldsymbol{P}$ applied on $\boldsymbol{Ins}$ leads to a sequence of states

$s_0, \ldots, s_i, \ldots$

with $\boldsymbol{K}(s_i, e_i) = s_{i+1}, \quad s_i, s_{i+1} \in \boldsymbol{S}, \; e_i \in \boldsymbol{Env}$

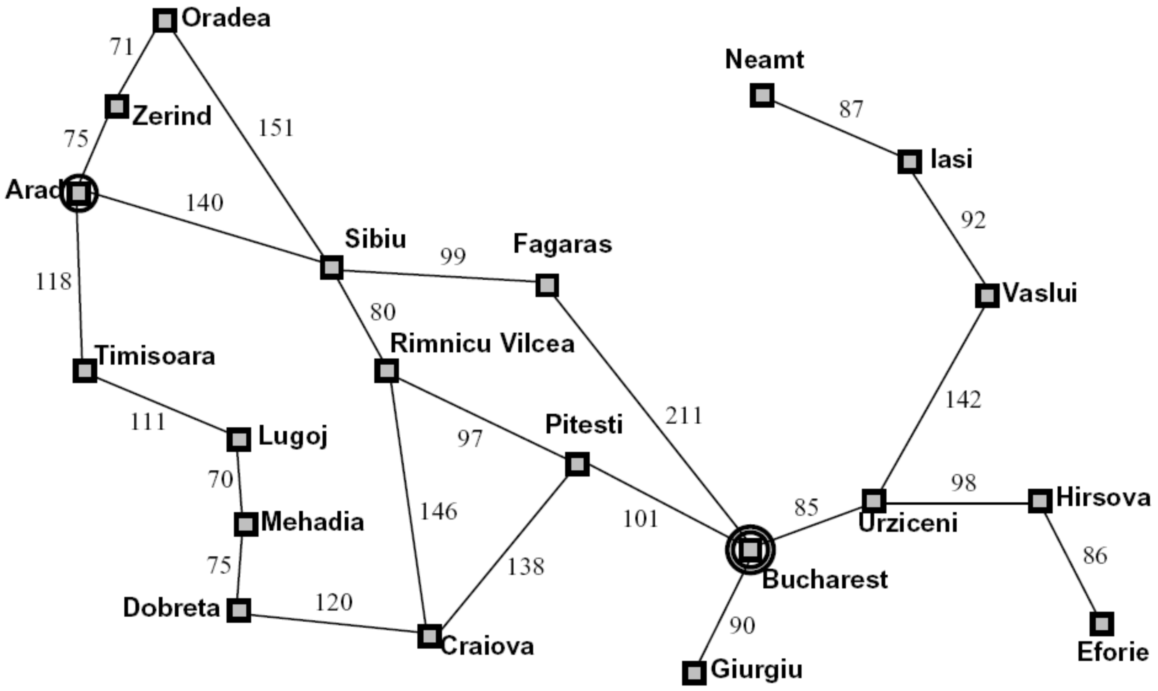
☞

1. Protocols a search run
2. 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
3. Might be looked at to determine solution

UNIVERSITY OF CALGARY

Examples

Example: Traveling in Romania



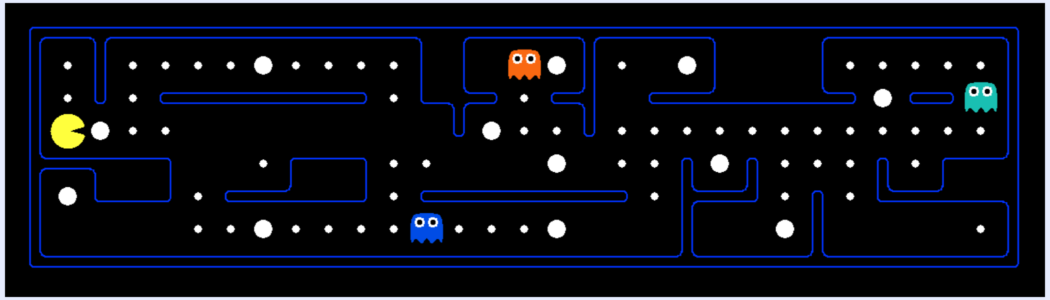
Travel from Arad to Bucharest

- State space model:
  - Cities = S
- Transitions:
  - Roads = T
    - Go to adjacent city
    - cost = distance
- Start state :
  - $s_0 = 'Arad'$
- Goal test:
  - $\boldsymbol{G}('Bucharest') \rightarrow \boldsymbol{yes}$
  - $\boldsymbol{G}(state) \rightarrow \boldsymbol{no}$
    - if state $!= 'Bucharest'$

UNIVERSITY OF CALGARY

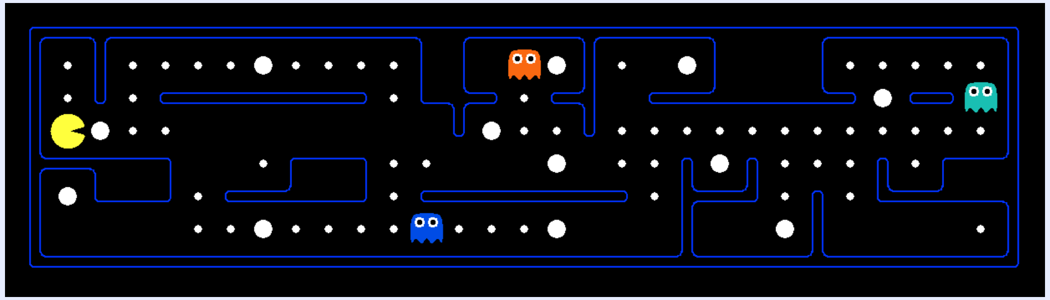
What's in a State Space?

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



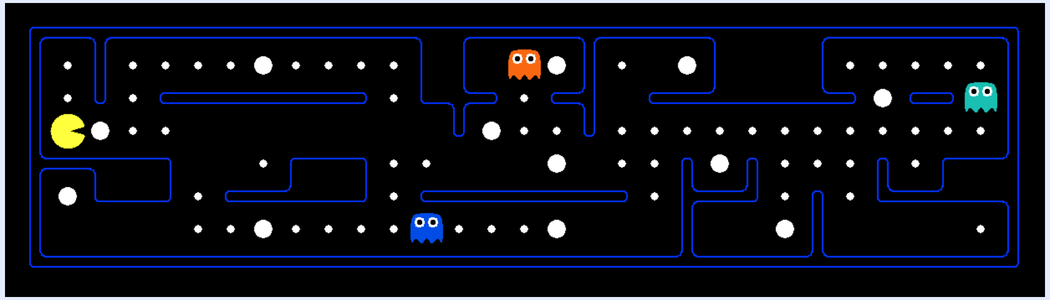
A search state keeps only the details needed for planning (abstraction)

- 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)

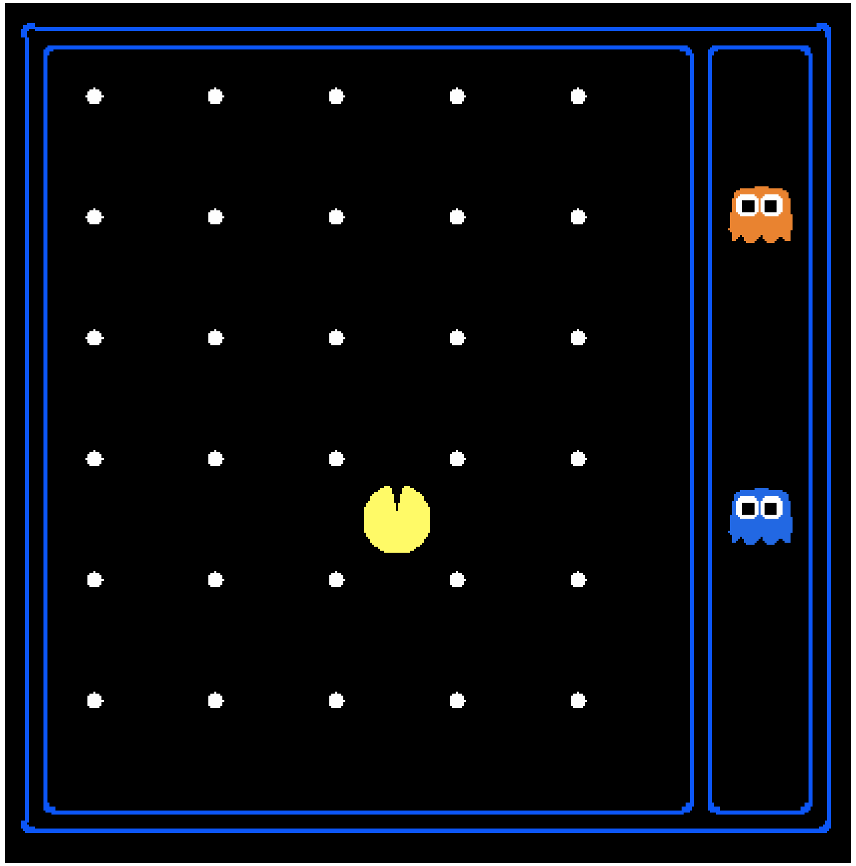
- Problem: Pathing
  - States: (x,y) location (make up S)
  - Actions: NSEW (help us decide T)
  - Successor: update location only (make T)
  - Goal test: is (x,y)= END (make G)
- Problem: Eat-All-Dots
  - States: {(x,y), **dot booleans**}
  - Actions: NSEW
  - Successor: update location and **possibly 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?
    $120x(2^{30})$

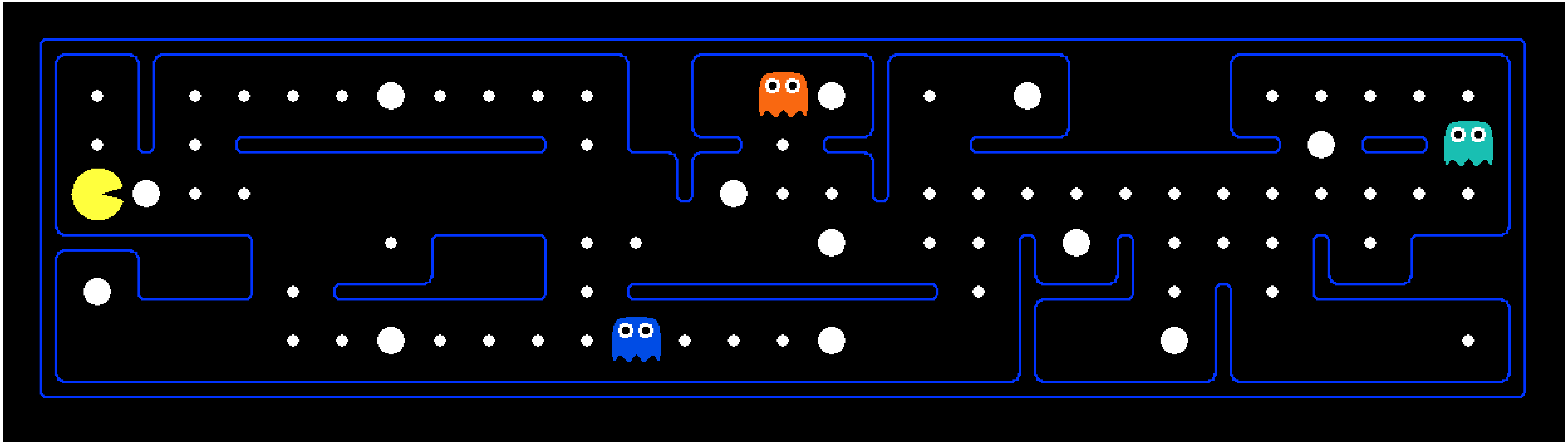
12 x 10 grid (dot and spaces)

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



UNIVERSITY OF CALGARY

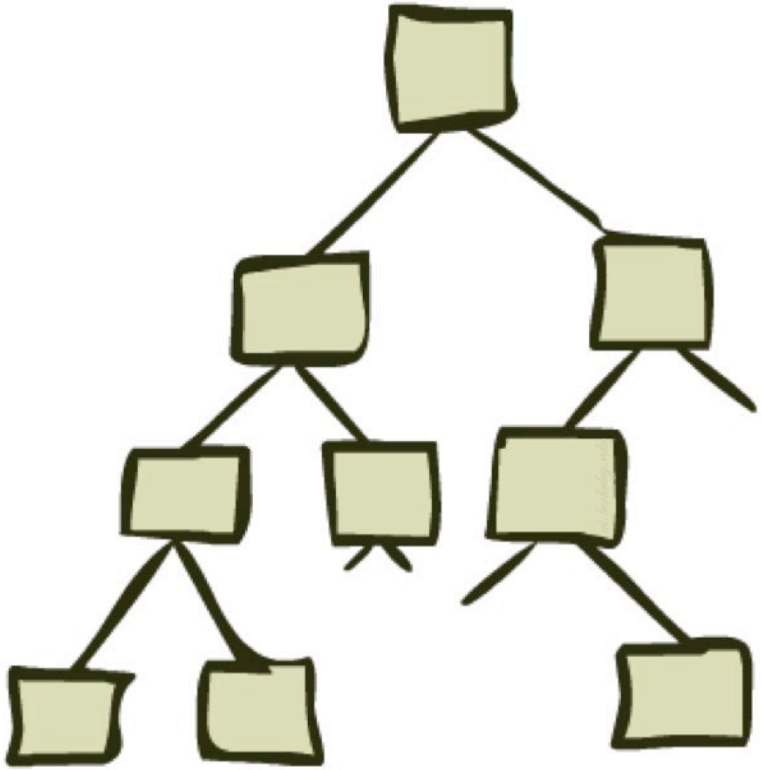
Safe Passage



- 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**)

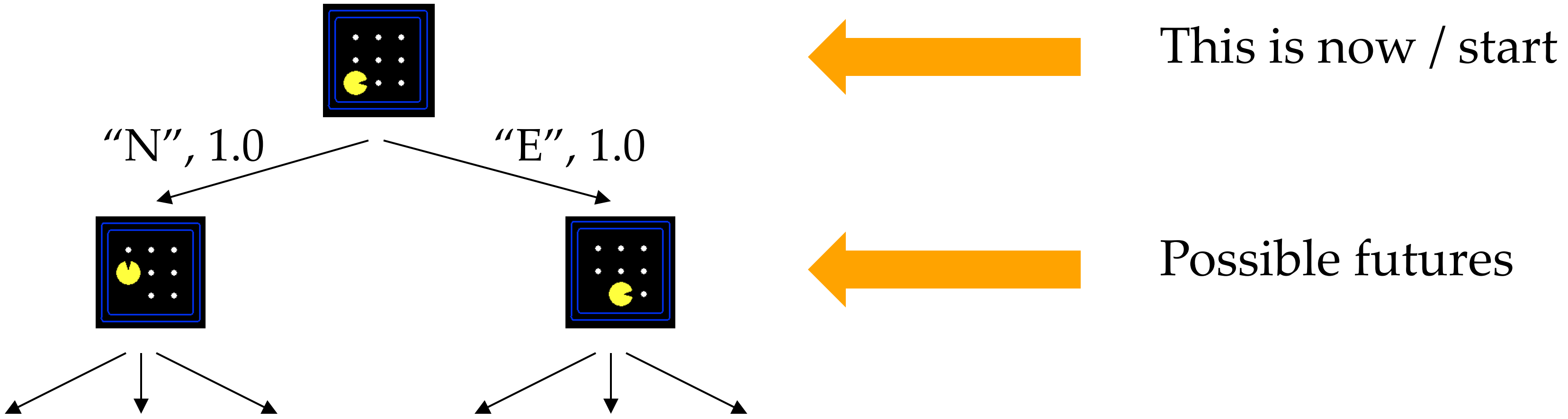
Graphs? Trees?

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

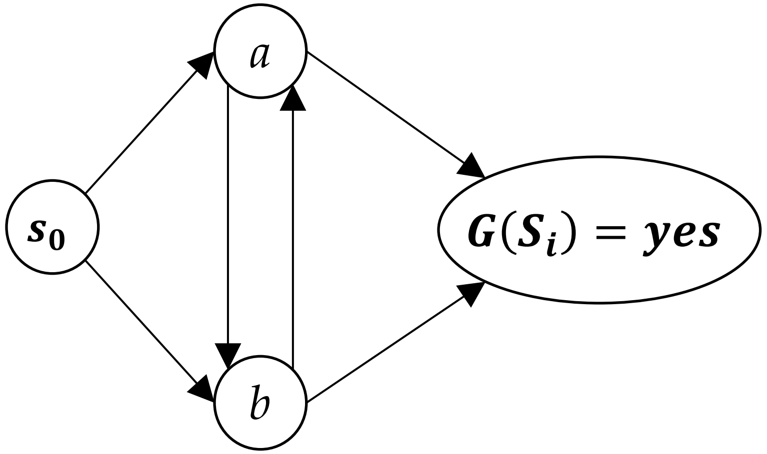
Search Trees



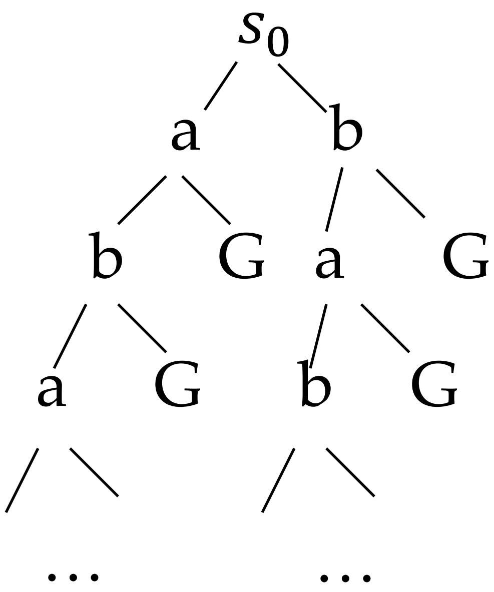
- A search tree:
  - A “what if” tree of plans and their outcomes
  - The start state is the root node
  - Children correspond to successors
  - Nodes show states, but correspond to PLANS that achieve those states
  - For most problems, we can never actually build the whole tree

State Space Graphs vs. Search Trees

Consider this 4-state graph:



How big is its search tree (from S)?



∞

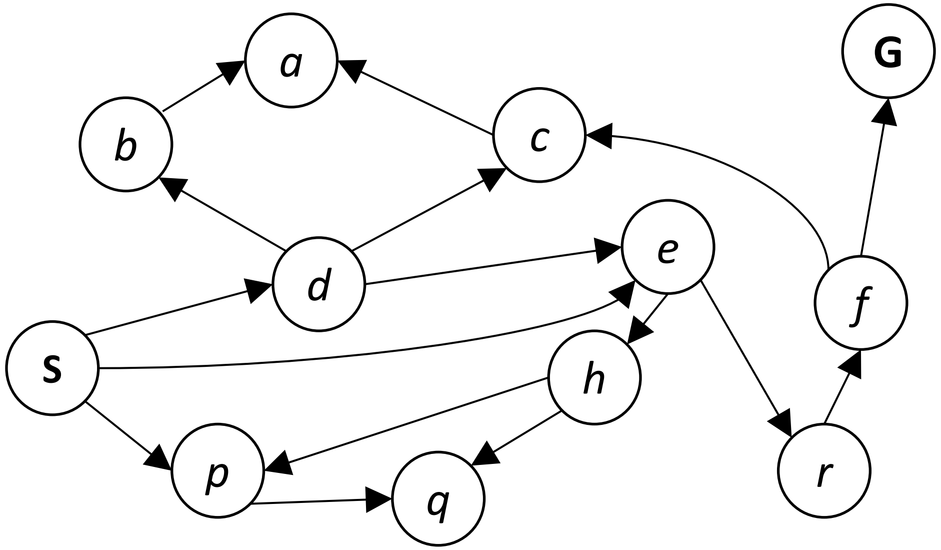
Important: Lots of repeated structure in the search tree!

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 search graph for a tiny search problem

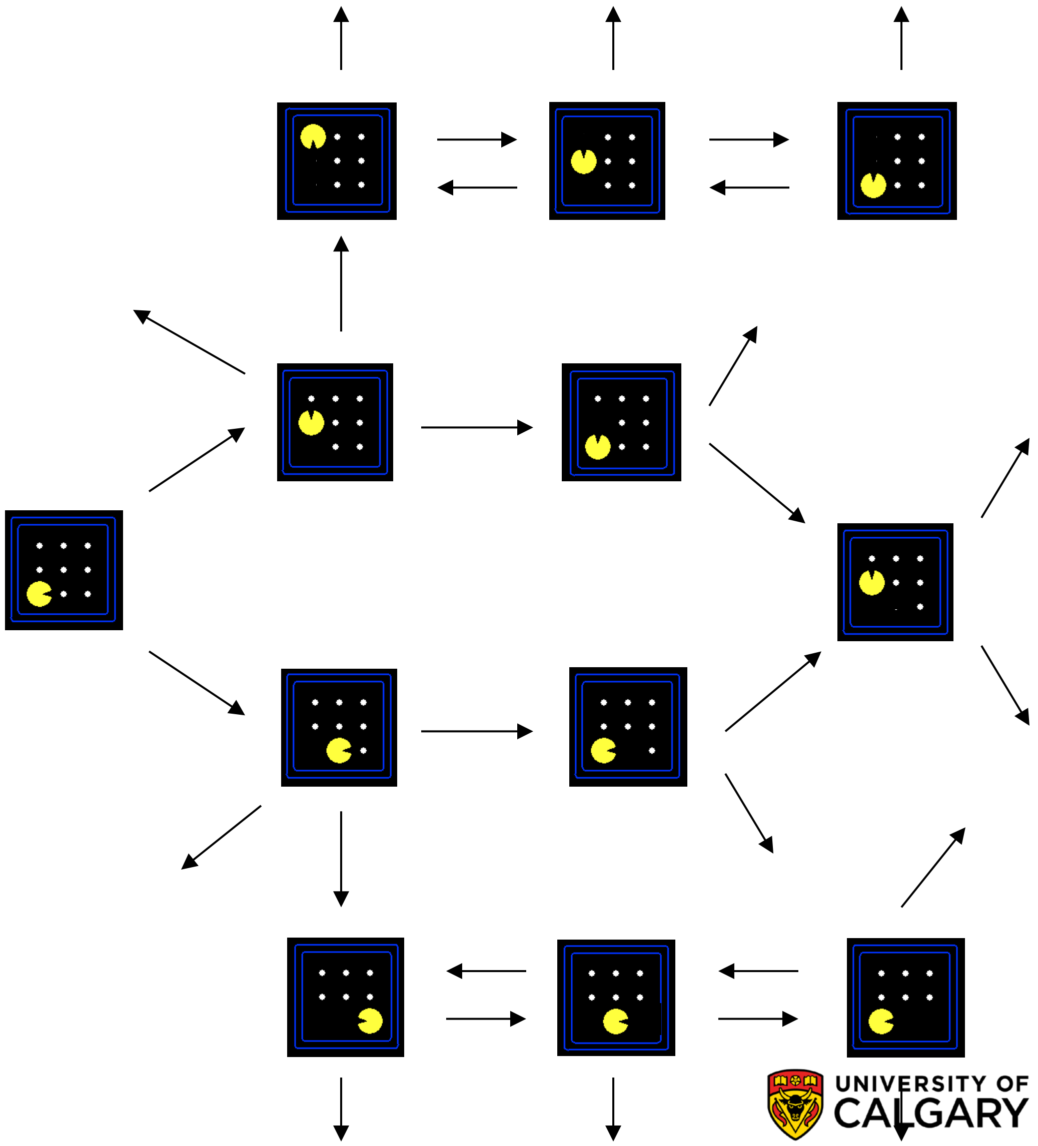
UNIVERSITY OF CALGARY

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)

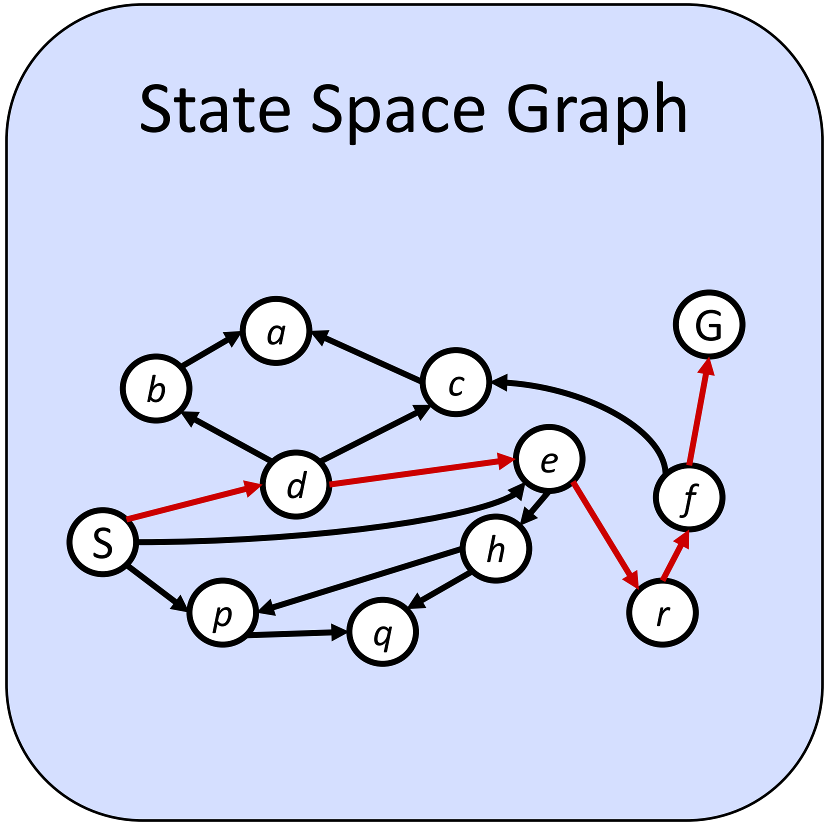
- In a state space graph, each state occurs only once!

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



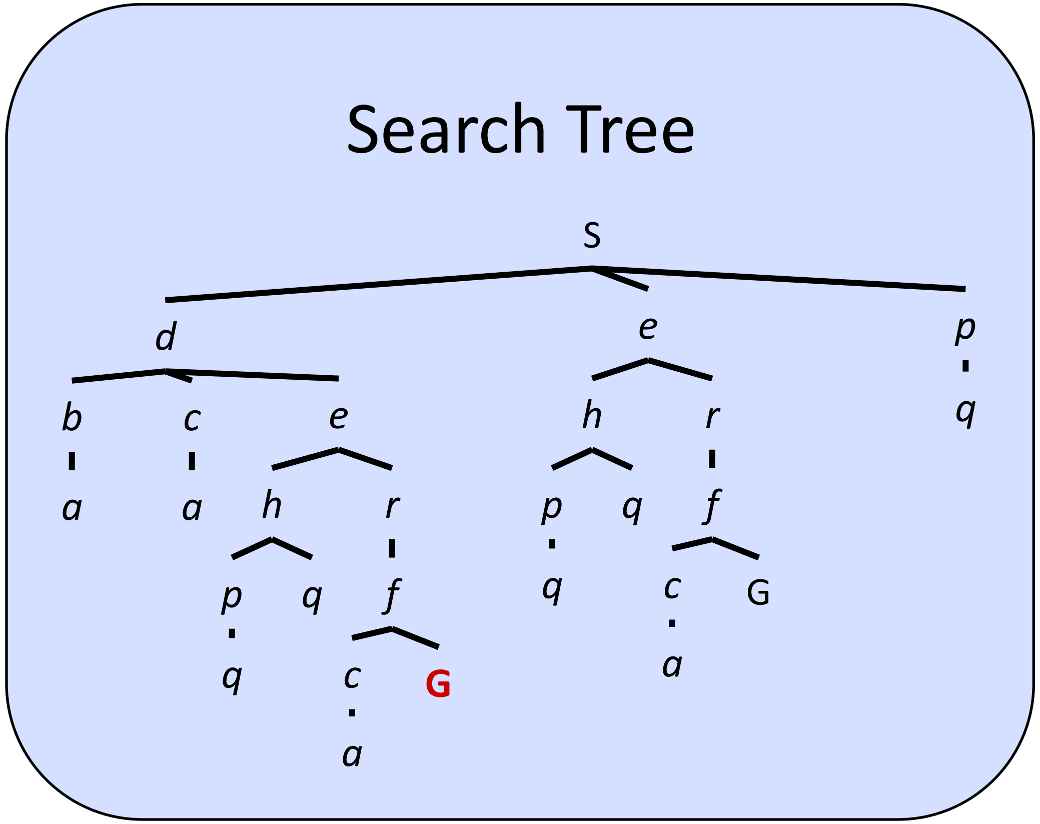
UNIVERSITY OF CALGARY

State Space Graphs vs. Search Trees



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.



Problems that need solving

Problems to solve when designing search model and process

- Combine
  1. **application knowledge** and
  2. **general search knowledge** (from search paradigms [ex. set, and, or, …])

- Define what input knowledge is necessary
- Define outside influences

- Select search paradigm
- Define search control knowledge
  ☞ part from application, part from paradigm

- Look for limitations in knowledge

Search States

Search States: General Comments

In general, they contain information about

- application
- past search
- future possibilities
- particular user interest (i.e. input; instance).

State vs Environment

- Data from outside of knowledge base and given instance
  ☞ **environment**
  - Example: new sensor data, changes in the world the system acts in, new tasks to be scheduled

- Data that never changes during search
  ☞ **environment**
  - Example: cost-profit vectors

- Data describing internal beliefs, (partial) solutions, results of reasoning and everything not mentioned above
  ☞ **state**

Transitions

Transitions: General Comments (I)

In general, they connect two states:

- Directed relation: (s_1,s_2) means you can go from s_1 to s_2 (not vice versa)

- Based on rules from
  - Application area
  - Semantics of states

Transitions: General Comments (II)

Big problem:

relation, i.e. there might be many states you can go to from a particular state

☞ the less the better

Use of more application knowledge in both states and rules for transitions can reduce number of potential successor states.

But: you can lose short search derivations and even correctness and completeness of algorithm

☞ less transitions **vs** better search control

Processes

Search Processes: General Comments

- Main tasks
  - Selection of the next search state
  - Integration of environment information

- Usually, many processes possible to a given search model
  - ☞ selection of search control essential for efficiency of search system

☞(will return to search controls but first talk about types of search)

Summary

UNIVERSITY OF CALGARY

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
- What problems do we have to solve when designing search solutions

Next…set-based search

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

UNIVERSITY OF CALGARY