Artificial Intelligence: Search Definitions

CPSC 433: Artificial Intelligence
Fall 2022

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

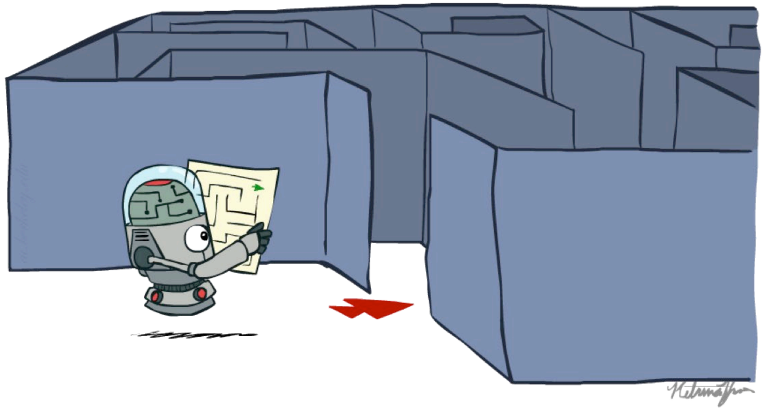
Thursday, September 15, 2022

UNIVERSITY OF CALGARY

Search: Basic Definitions

Search is at the core of nearly all 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
- …



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

Search Problems

Definitions

Basic Definitions (I)

Search Model $A = (S, T)$

S set of possible states

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

Search Problems Are Models

UNIVERSITY OF CALGARY

Basic Definitions (I)

Search Model $A = (S, T)$

S set of possible states

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

☞

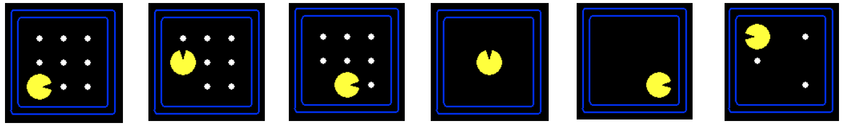
- Defines main data structure and possibilities (space)
- Tells us what the control can work with
- Limits the choices of the control

Search Problems Are Models

Basic Definitions (I)

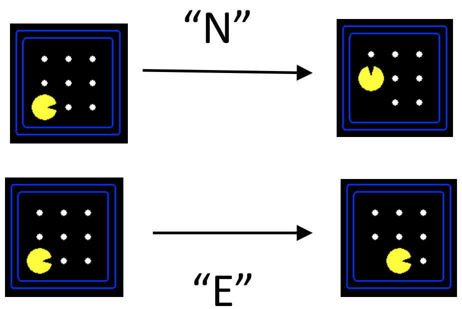
Search Model $A = (S, T)$

S set of possible states



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

Two transitions from state 1



☞

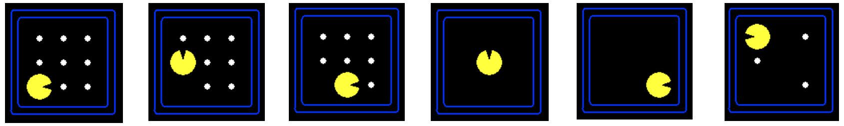
- Defines main data structure and possibilities (space)
- Tells us what the control can work with
- Limits the choices of the control

Basic Definitions (I)

Search Model $A = (S, T)$

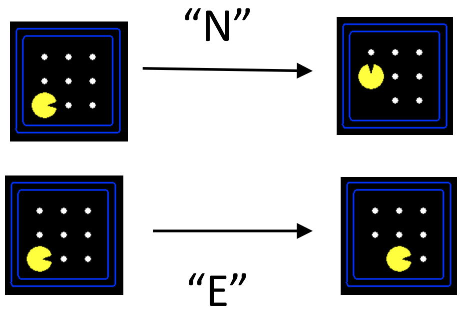
S set of possible states

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



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

Two transitions from state 1



$(s_1, s_2) \in T$
$(s_1, s_3) \in T$

☞

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

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 $(s,s') \in T, e \in Env$

Basic Definitions (II)

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

A	search model
Env	environment of process *(sometimes your configuration of algorithm)*
$K: S \times Env \rightarrow S$	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

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

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 (positively)
- Normally is generated out of user input

Basic Definitions (IV)

Search Derivation:

P applied on Ins leads to a sequence of states

$s_0, \ldots, s_i, \ldots$ with $K(s_i, e_i) = s_{i+1}, \quad s_i, s_{i+1} \in S, \; e_i \in Env$

Basic Definitions (IV)

Search Derivation:

P applied on Ins leads to a sequence of states

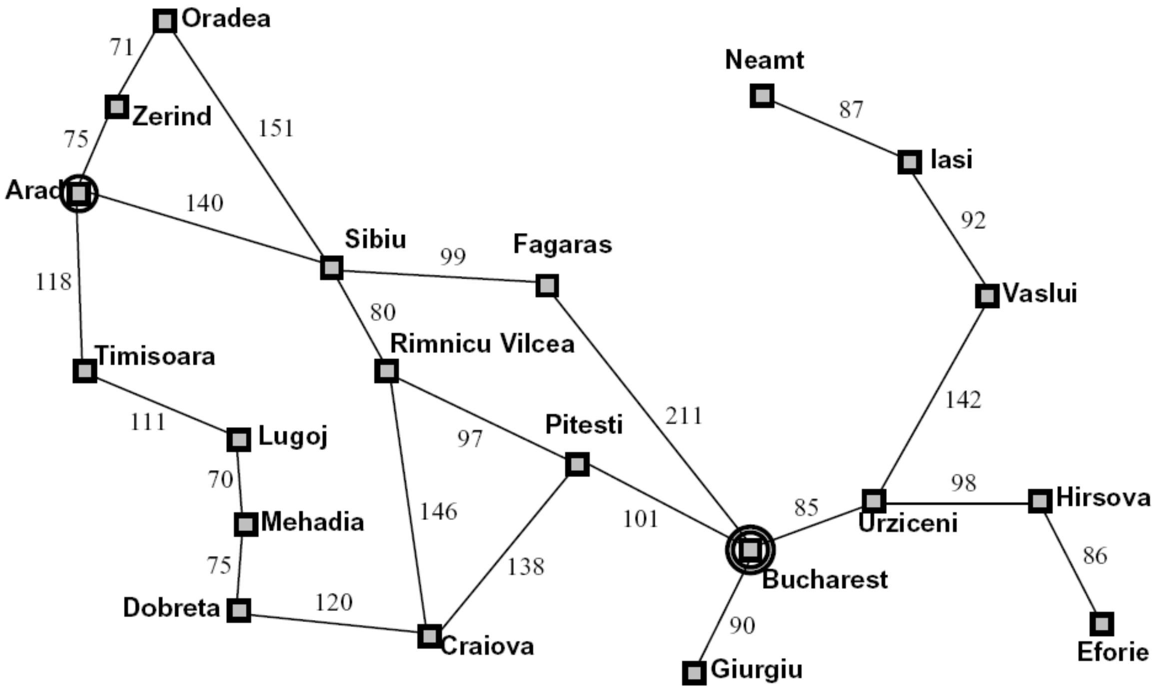
$s_0, \dots, s_i, \dots$ with $K(s_i, e_i) = s_{i+1}, \quad s_i, s_{i+1} \in S, \; e_i \in Env$

☞

- Protocols a search run
- 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

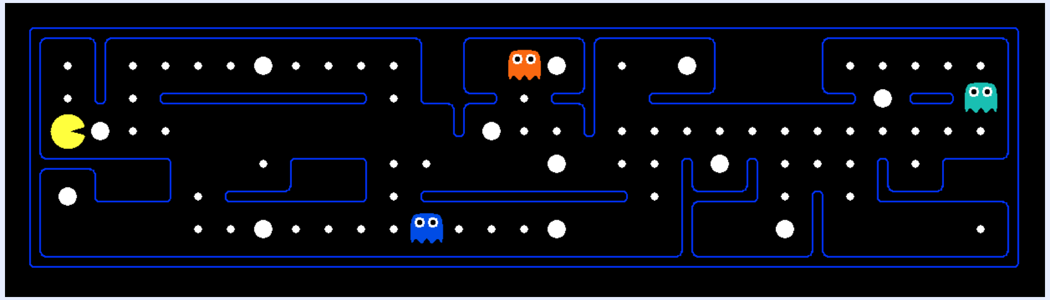
Example: Traveling in Romania



- State space model:
  - Cities
  - $'Bucharest' \in S$
- Transitions:
  - Roads: Go to adjacent city (cost = distance)
  - Ex. $('Bucharest', 'Urziceni') \in T$
- Start state :
  - $s_0 =' Arad'$
- Goal test:
  - $G('Bucharest') \rightarrow yes$
  - $G(s_i) \rightarrow no$ if $s_i \in S, s_i \neq' Bucharest'$

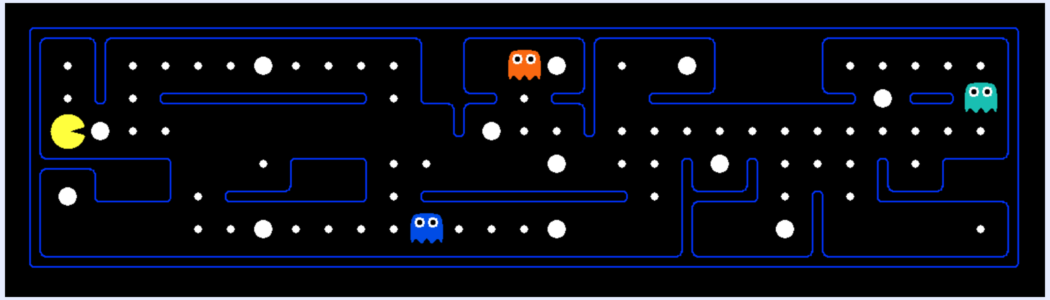
What's in a State Space?

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



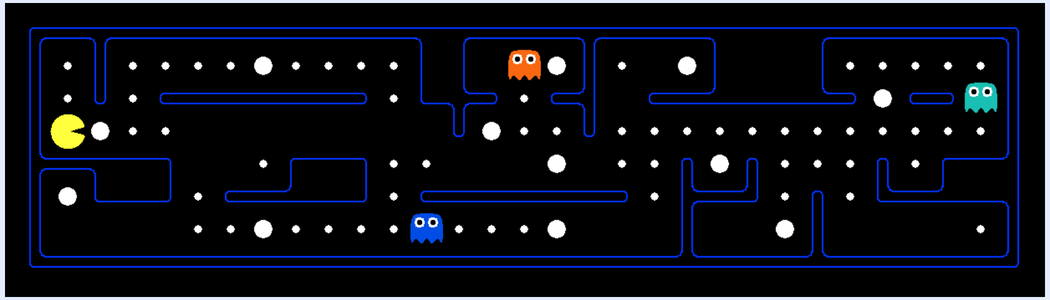
A search state keeps only the details needed for planning (abstraction)

- 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

UNIVERSITY OF CALGARY

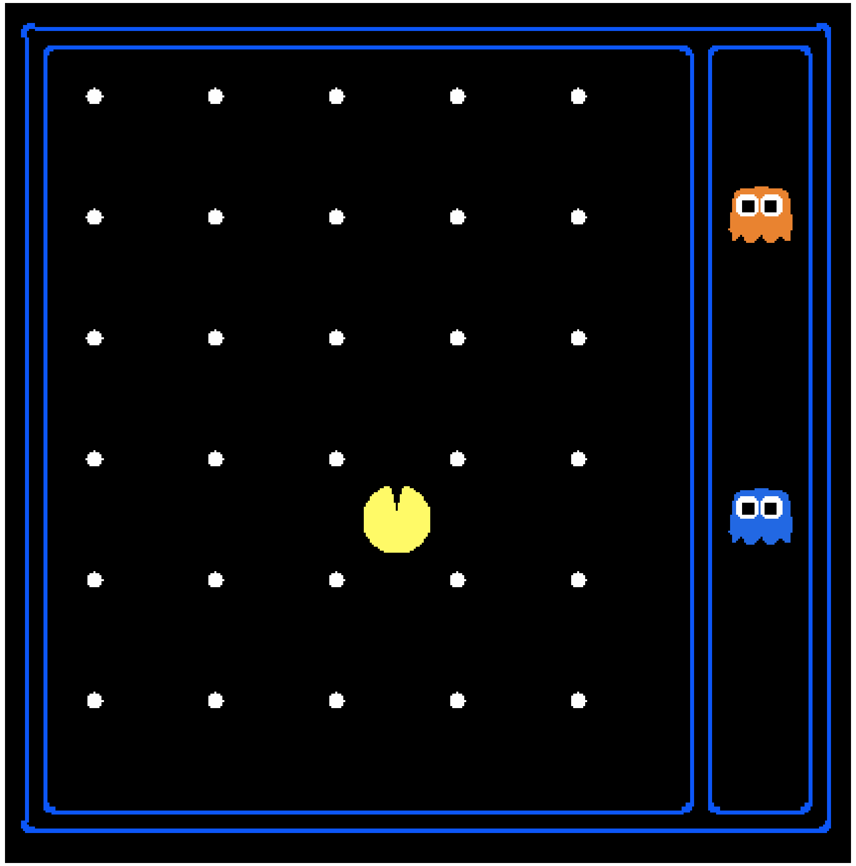
State Space Sizes?

12 x 10 grid (dot and spaces)

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

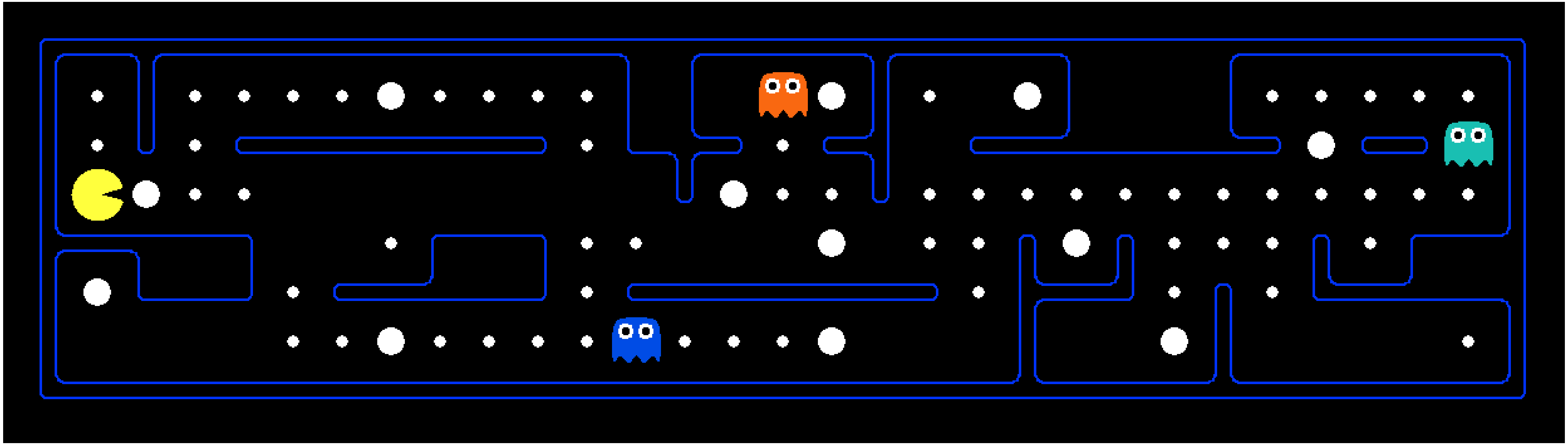
- World state:
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing: NSEW

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



UNIVERSITY OF CALGARY

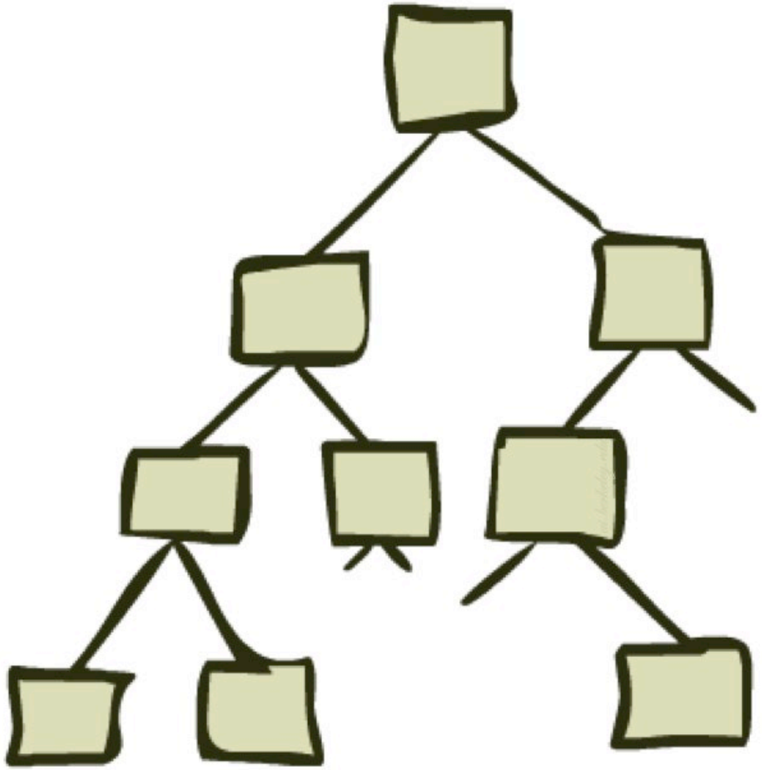
Safe Passage



- 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?

State Space Graphs and Search Trees

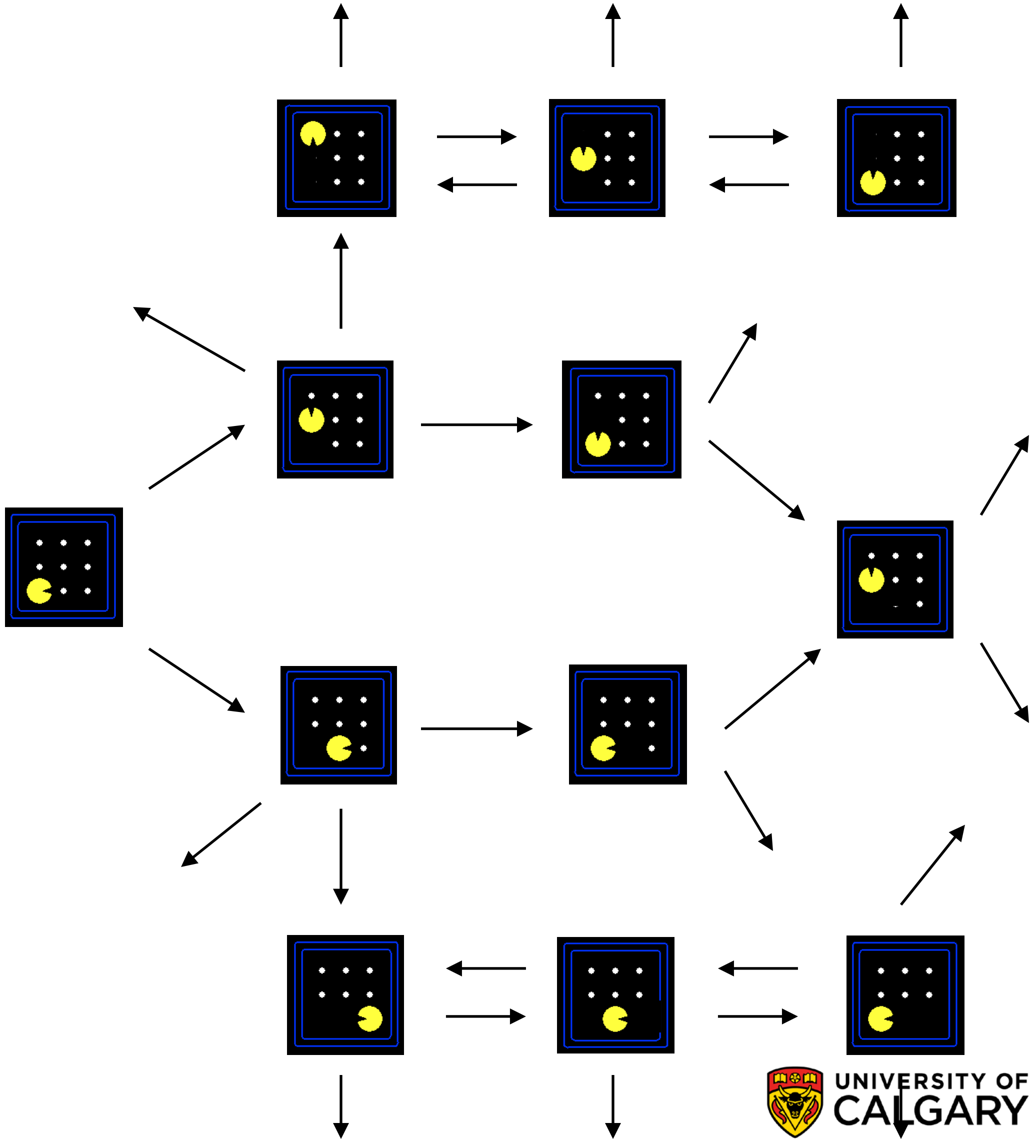


State Space Graphs

- 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

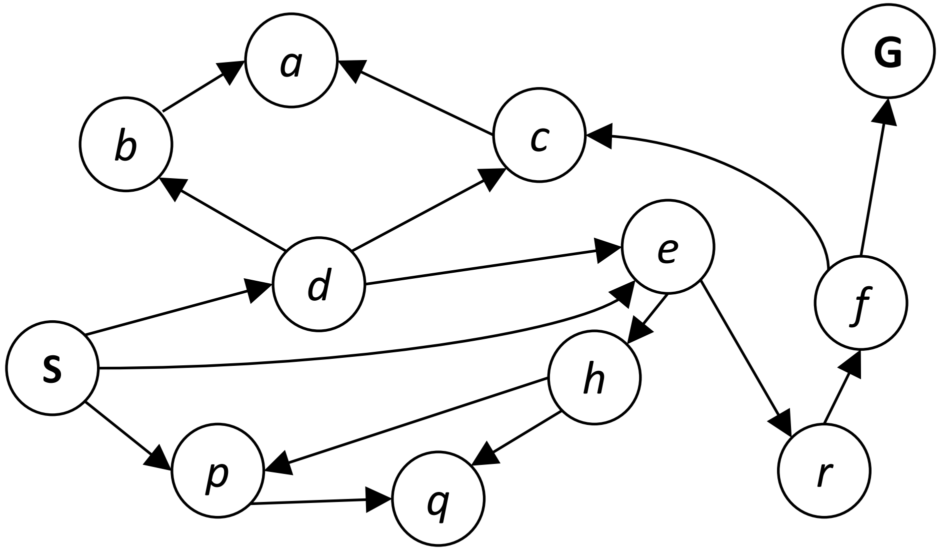


State Space Graphs

- 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)

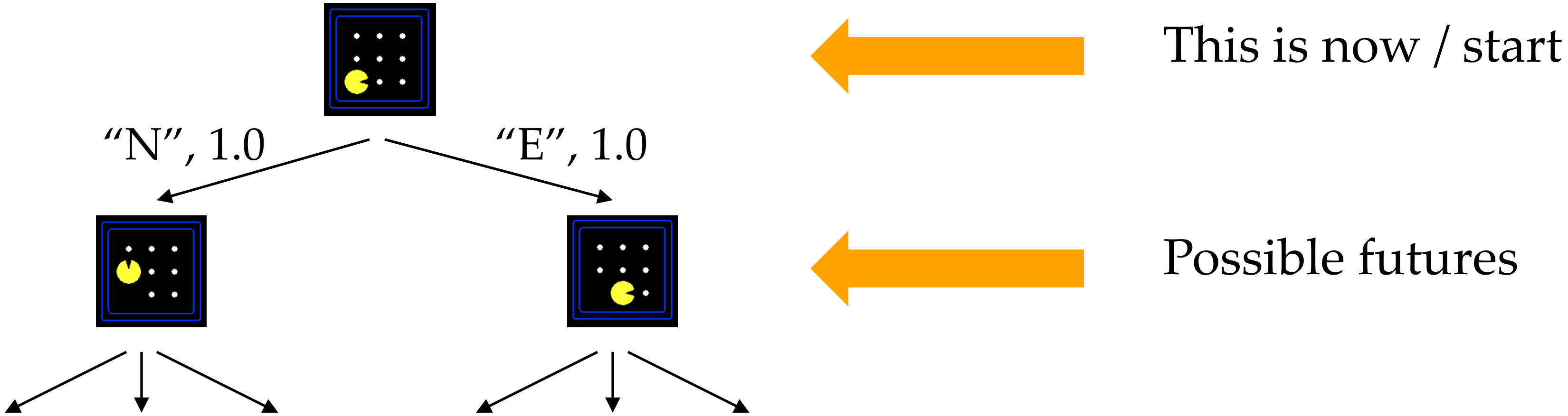
- In a state space graph, each state occurs only once!

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



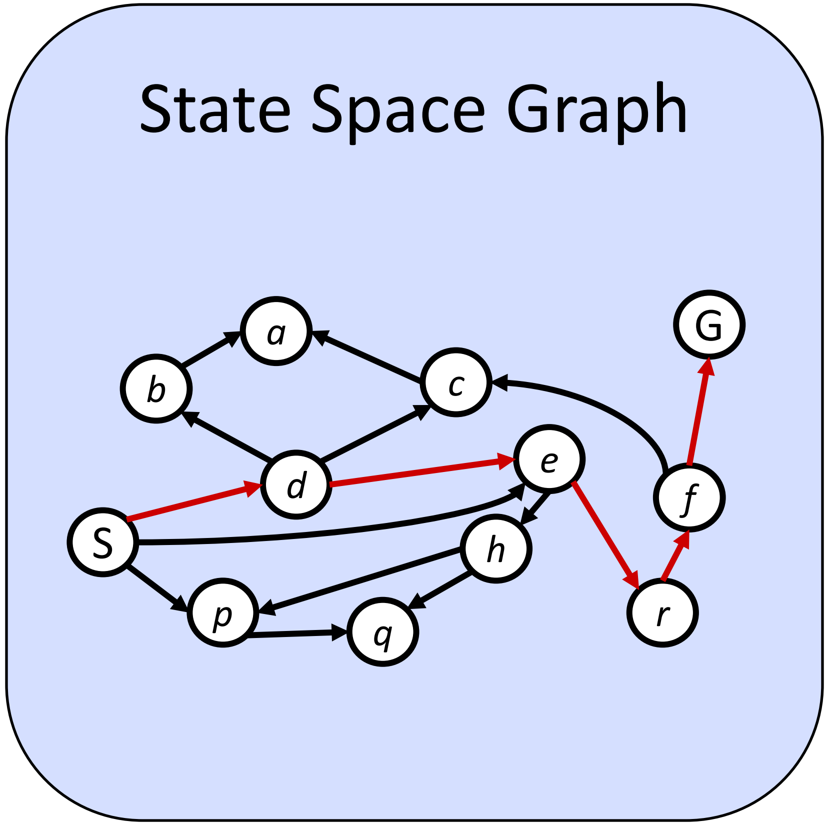
Tiny search graph for a tiny search problem

Search Trees



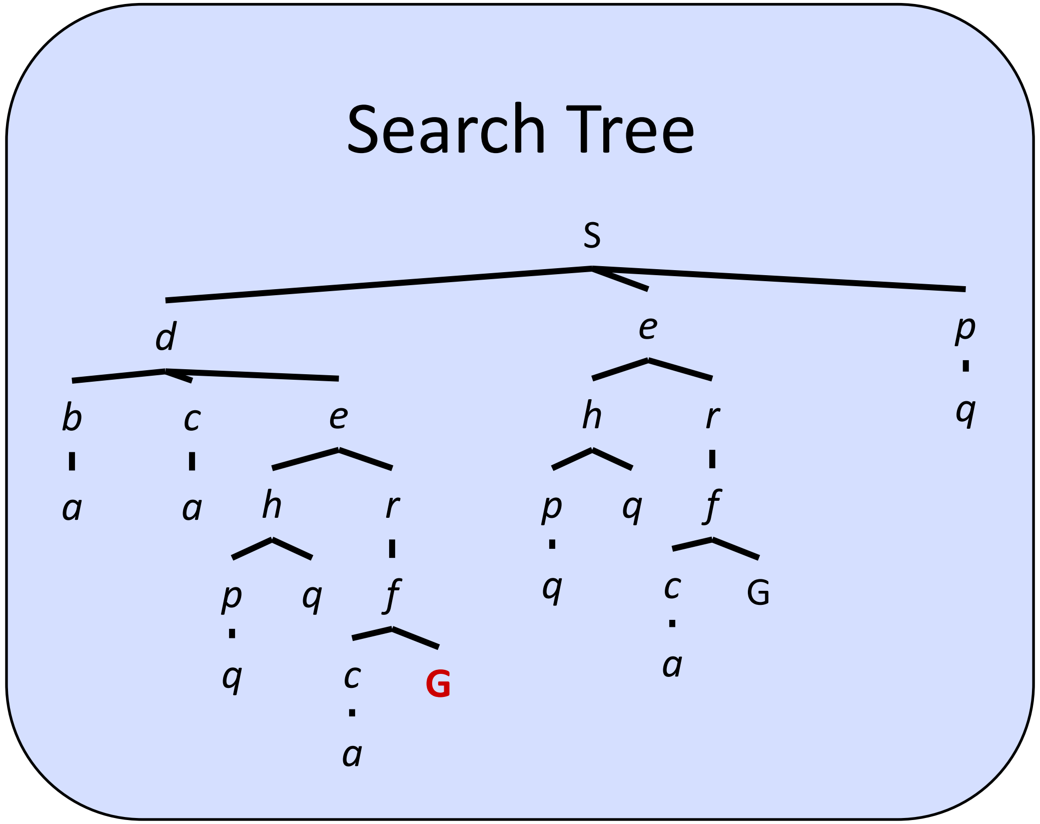
- A search tree:
  - A "what if" tree of plans and their outcomes
  - The start state is the root node
  - Children correspond to successors
  - Nodes show states, but correspond to PLANS that achieve those states
  - For most problems, we can never actually build the whole tree

State Space Graphs vs. Search Trees



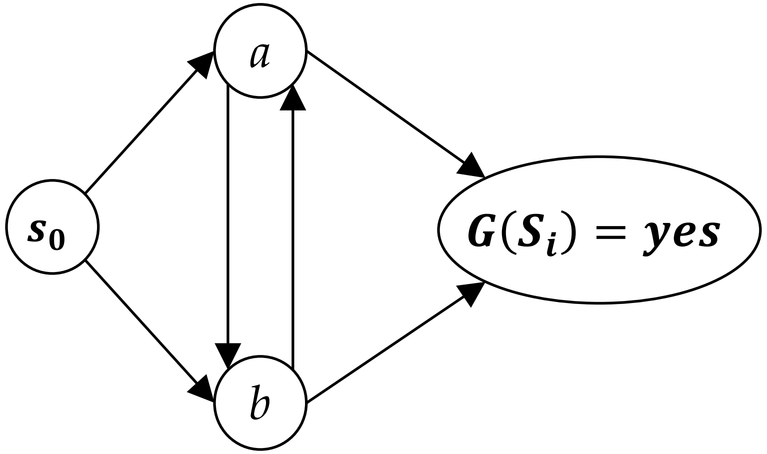
Each NODE in in the search tree is an entire PATH in the state space graph.

We construct both on demand – and we construct as little as possible.



State Space Graphs vs. Search Trees

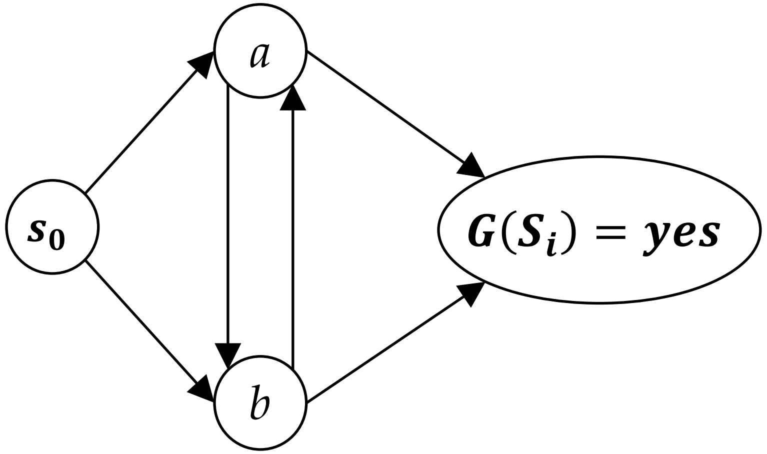
Consider this 4-state graph:



How big is its search tree (from S)?

State Space Graphs vs. Search Trees

Consider this 4-state graph:

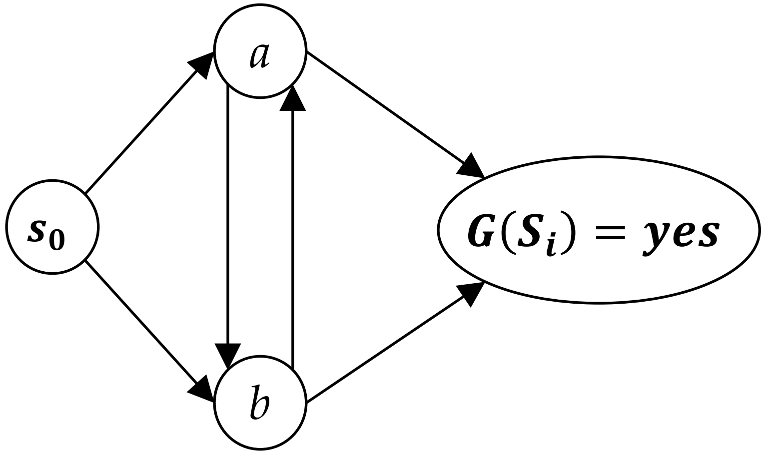


How big is its search tree (from S)?

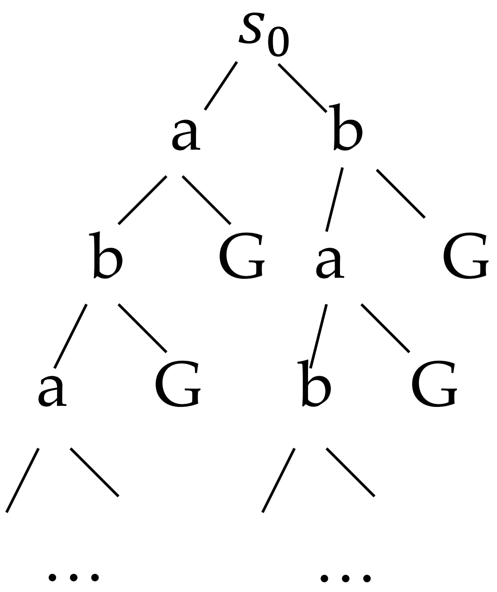
∞

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!

Problems that need solving

Problems to solve when designing search model and process

- Combine application knowledge and general search knowledge (from search paradigms)
- 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: General Comments (I)

In general, they contain information about

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

Search States: General Comments (II)

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: 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

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)

Onward to …
Set-based Search

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY