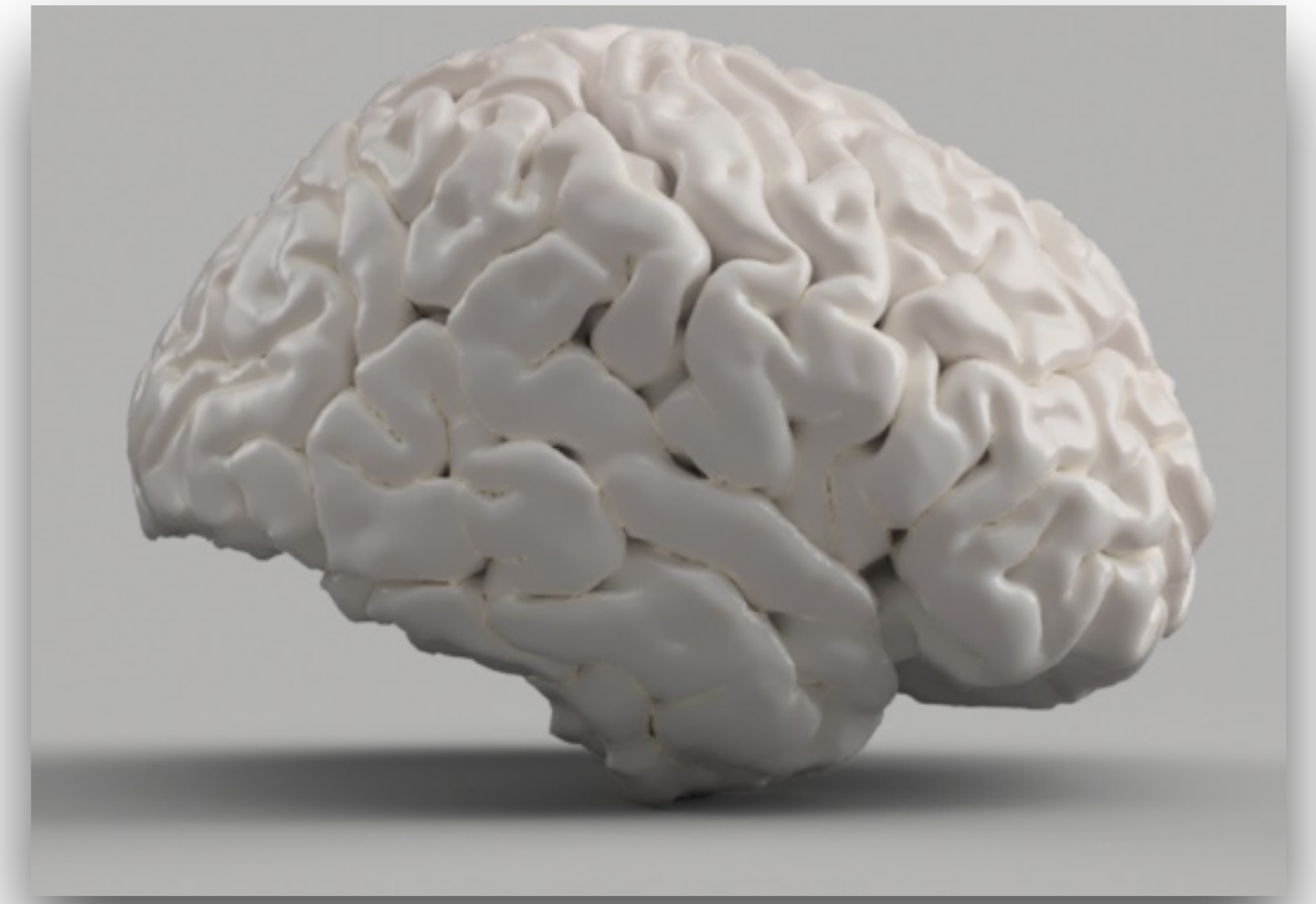
COGS109: Lecture 16



Search and AI

July 27, 2023

Modeling and Data Analysis

Summer Session 1, 2023

C. Alex Simpkins Jr., Ph.D.
RDPRobotics LLC | Dept. of CogSci, UCSD

Announcements

- New cape replacements logistics
- Assignments remaining A2, A3, D5, D6, D7, Q3, Q4, project
- Project checkpoint 2 EDA
- Project meetings to check in

Outline for this lecture

- Intro to AI
- Search methods

What is Artificial Intelligence (AI)?

- Human Intelligence has many definitions, but
  - **We can consider two important forms**
    - General intelligence (G)
    - Specific intelligence factors
- No perfect agreement of the AI definition
  - **Still evolving as a field**
- Broadly, one useful definition is *the field which attempts to use artificial devices (usually computers) to solve problems generally solved by humans*
  - **Some problems aren’t yet solved by computers OR people however, so keep this in mind**
- Another model!!!

What are some AI problems?

- Much early work done in game playing and theorem proving
  - **Why? Well-defined rules and algorithms, approaches that can be clearly laid out and executed**
  - **People that play games well and prove theorems are generally considered to be displaying intelligence**
    - Chess/checkers players
    - Geometry
    - Commonsense reasoning
      - **Navigating to class**

Task domains in AI (a few)

- Basic tasks
  - **Perception**
    - Vision, Speech
  - **Natural language**
    - Understanding, Generation, Translation
  - **Commonsense reasoning**
  - **Robot control**
- Formal tasks
  - **Games**
    - Chess, backgammon, checkers, go
  - **Mathematics**
    - Geometry, logic, integral calculus, proving properties of programs
- Expert tasks
  - **Engineering**
    - Design, fault finding, manufacturing planning
  - **Scientific analysis**
  - **Medical diagnosis**
  - **Financial analysis**

Rich and Knight, *Artificial Intelligence*

Questions to consider

- What are our basic assumptions about intelligence?
- What techniques can we use to solve AI problems?
- In what level of detail are we modeling (human or animal) intelligence?
- How do we know if we succeed making something intelligent?

The assumption

- Physical symbol system [Newell & Simon 1976]
  - **A set of physical patterns related in some physical way which may be part of other patterns referred to as symbol structures**
  - **System also contains processes which operate on the structures**
    - Creation, modification, reproduction, destruction
  - **A machine producing with time an evolving set of symbol structures, existing in a world broader than the mere set of symbols**
- **Physical symbol system hypothesis -** "a physical symbol system has the necessary and sufficient means for general intelligent action[Rich & Knight 1991]."

What is an AI technique?

- AI is a tremendously broad field of study
- AI techniques usually manipulate symbols as defined previously
  - **A general statement from all the past research is that intelligence requires knowledge**
    - Voluminous, hard to characterize, changing, organized how it will be used
  - **AI technique is one that exploits knowledge to solve a problem, where the knowledge is represented such that**
    - Captures generalizations (differs from data)
    - Understandable by people providing it
    - Modifiable to correct errors and adapt to changing world
    - Not context-dependent
    - May be accessed in a strategic way (because so much of it)

Model detail/level

- Are we working to produce a system that performs tasks the WAY people perform them?
- Are we working to produce a system that solves the same problem a person might in the easiest way possible?
- Both have been addressed by AI methods
  - **As cognitive scientists you will be surprised at the optimality of the human being from the context of, mechanical, dynamical, intellectual, computational, and other perspectives**
  - **Some things are more easily done with computers in a different way than a person might do them**
    - Nonsense syllables - storage and coupling to a stimulus syllable

Why would we model human performance?

- Testing cognitive theories of human mind
  - **PARRY [Colby, 1975] exploited a model of human paranoid behavior**
- Let computers understand human reasoning
  - **Reading comprehension**
- Let people understand computer reasoning
- Use people as models for solving problems, implement those solution methods in a way that benefits humankind (and generally the world/environment/etc of course)

Considering the last point and where we've just been…

- Artificial Neural Networks have been very popular models to test theories of human problem solving because of their structural parallel to the human brain's functionality
  - **Thus the model we developed of ANN can be used for AI models**
- Recently massively parallel computational systems are allowing more experimentation with high performance AI (with ANN implementation) models
  - **Think about your new computers - laptops have multiple processor cores now**
  - **nVidia's new 128 processor graphics card**

Evaluating model success - a criterion

- Turing test [Turing 1950] - method for determining whether a machine can think
  - **Needs 2 people and the machine**
  - **Interrogator in separate room**
    - Machine and other person communicate with them by typing questions and receiving typed responses
    - The goal of the machine is to convince the interrogator it is a person
      - **Success is interpreted as suggesting the machine can think**
- As of 2023 no machine has passed a formal Turing test, though MANY machines have fooled people into thinking they were persons
  - **ALICE example and chatbots**
  - **Foreign child**
  - **In chats, no active attempt to disprove chatter as a person**
  - **Tay the racist chatbot**

Other criteria

- Speed to perform a task
  - **Manufacturing (‘build to order’ computers) often use AI-based robotics to assemble products**
  - **Can often do what a skilled person might take hours to complete in a few minutes**
- Generally this is difficult to construct as single unifying statement
  - **Instead considering a particular instance with performance criteria which are more specific is the general approach**
  - **Humans again show their impressive adaptability by ability to solve such broad problems that defining all the problems concisely in one statement is very very difficult!!!**

So now what do we do to approach AI problems?

- Figure out a strategic way to ***encode*** massive amounts of knowledge
- Figure out a strategic way to ***access*** that knowledge quickly and efficiently

A general approach to solving problems

- Define the problem carefully
  - **Specify all assumptions**
  - **Specify all given relevant information**
- Analyze the problem
- Isolate the knowledge needed to find solution
- Choose best method of solution (technique) and use it!

An important AI area - search

- Searches are a common part of life for most organisms
- Let’s introduce search with a game

Tower of Hanoi game

- Start with the configuration to left, finish with configuration on right
- Only top ring can be moved at a time, can only be put on a larger ring or empty peg

Problem space

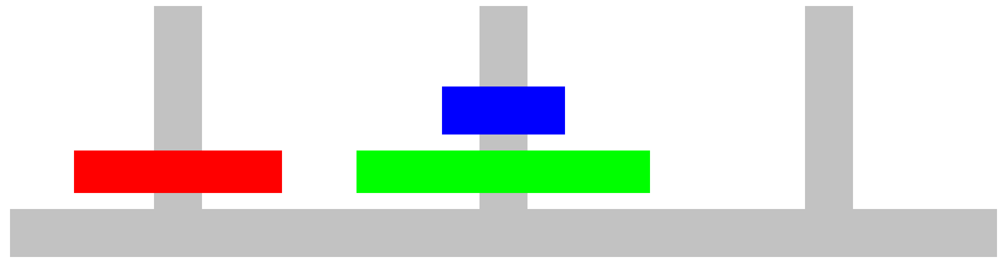
- **States** - the situations we encounter while attempting to solve the problem
- **Problem Space -** the set of all states for a problem
  - **Here it is the set of all possible configurations of the rings on the pegs**

Types of states

- **Initial states** - States where a given episode of problem solving starts
  - **With Hanoi problem one initial state**
  - **Other problems may have more**
- **Goal or solution states** - States that are considered solutions to the problem
  - **Again one solution state with Hanoi problem**
  - **Other problems may have more**
- **Failure or impossible states** - In some domains, there are states with the property that if they are ever encountered, the problem solving is considered a failure
  - **With Hanoi problem, any state in which the rule that rings can only be placed on bigger rings is violated**
  - **However often cast as constraints of the operators**
- *States can be explicitly (every possible state defined) or abstractly specified*

Operators

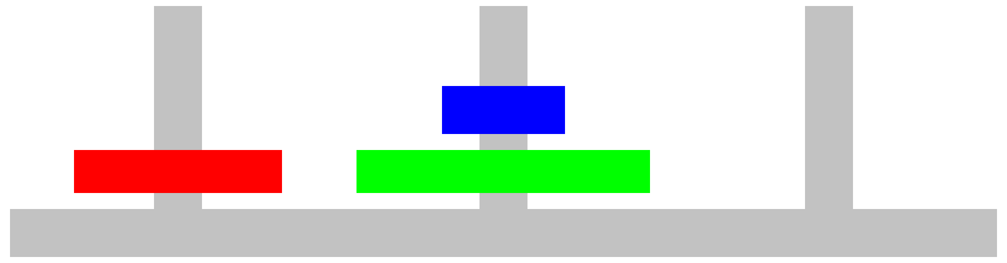
- We have to be able to manipulate states to make the problem useful
- **Operator -** can be applied to states in the problem domain, often an operator only acts on a subset of states



From this state, 3 possible operators can be applied

- Red ring can be moved to right hand peg
- Blue ring can be moved to left hand peg
- Blue ring can be moved to right hand peg

More on operators

- In this problem state

  - **The red ring cannot be moved to the middle peg because the blue ring is already there, and it is smaller.**
  - **The green ring can't be moved at all from this state.**
- When an operator is applied to a state, a new configuration of the problem domain, (a new *state*) is formed

What is a solution?

- **Solution to the problem domain** - A sequence of operators that can be performed from a given initial state, that doesn't pass through any failure states, and that leads to a goal state

Search for a solution

- We can apply several techniques
- The naïve approach
  - **Generate and test**
  - **Random search**
- Search spaces
  - **Breadth- and depth- first searches**

Generate and Test

1. generate a potential solution
2. see if it is in fact a solution
   - **(a) if so stop**
   - **(b) if not, return to 1.**

Issues with Generate and Test searches

- Problems with G.A.T. if
  - **If there are many possible solutions**
  - **If generating them is expensive, time consuming or dangerous**
- G.A.T. is useful if
  - **set of potential solutions isn't too big,**
  - **if it possible to try them quickly**
  - **if the more controlled approaches described in the next few slides can’t be used**
- Refinement
  - **Only try each solution once**
  - **easy if all of the possible solutions can be enumerated - you just try them in order**

Refinements of G.A.T.

- Generate solutions randomly, but to keep a list or an array of all of the solutions you have tried
- Before a new one is tried, to see if you have already tried that one
- But problem with this approach
  - **if it takes you a long time to solve the problem, your list of attempted solutions will grow, and you will spend most of your time checking whether you have tried a given solution before**
  - **If the set of possible solutions is really huge - or if it is infinite**
    - it is best to just generate solutions randomly and not check
    - chance of trying something twice is very small, and the overhead of checking is very high

Random search - using problem space to find a solution

- Assume that the program can store a representation of the set of states that it has encountered while trying to solve a problem
- The algorithm:

1. Start with the initial state.
Loop:
  2.a Choose an operator at random.
  2.b If the operator can't be applied,
    or yields a failure state,
    continue with the previous state at step 2.a.
  2.c If the result is a goal state, stop.
  2.d Otherwise, continue with the new state at step 2.a.

Issues with random search

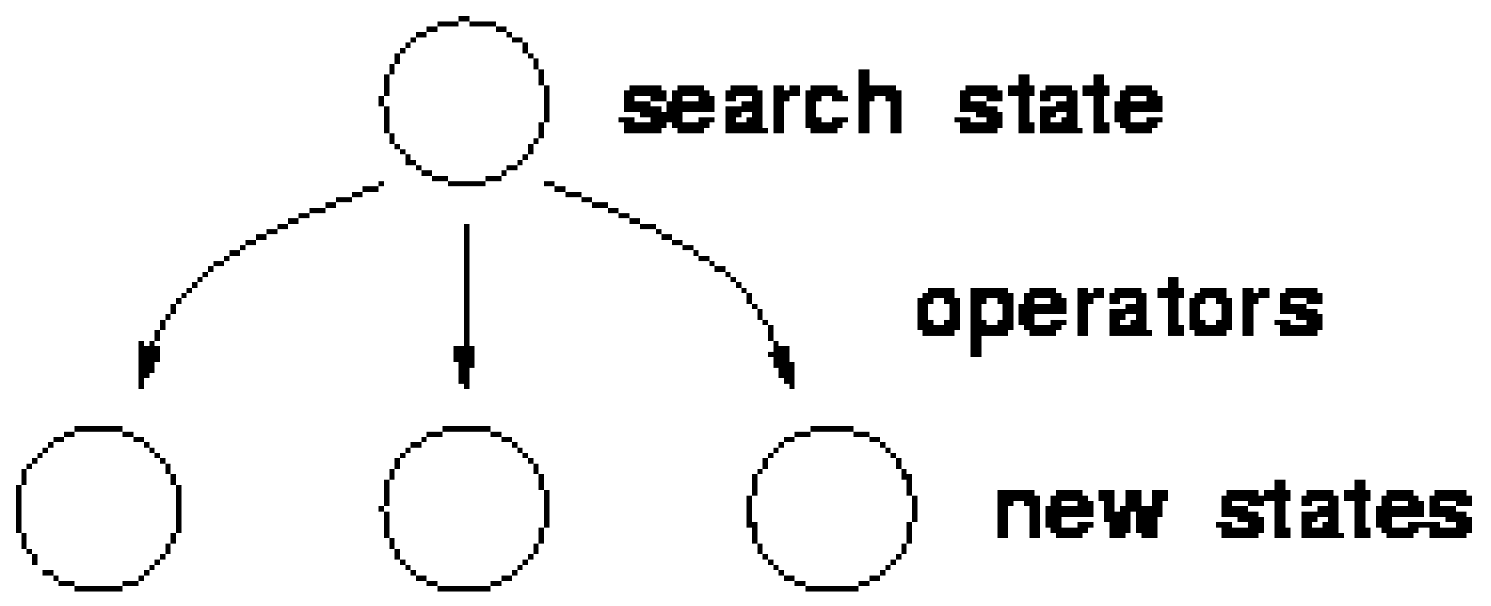
- It might get into an infinite loop generating the same states over and over again.
  - **Similar issues with generate and test, but worse - can get stuck in a loop where it keeps going back to the same state, then operating to go forward, and back again because the same operator is used**
- It might never generate the actual solution
  - **Some techniques are guaranteed to find a solution (if it exists), this one is not, nor is generate and test**
- It might take an arbitrarily long time to find a solution
  - **No guarantee as to how long it will take to find a solution, some techniques can guarantee finding solution in a finite time which is specified**

Solutions to problems with random search

- Avoiding the first problem
  - **We need a systematic way to explore the state space**
- Avoiding the second/third problems
  - **Find a method of determining which state most likely will ultimately lead to a solution state**

First an introduction to search space

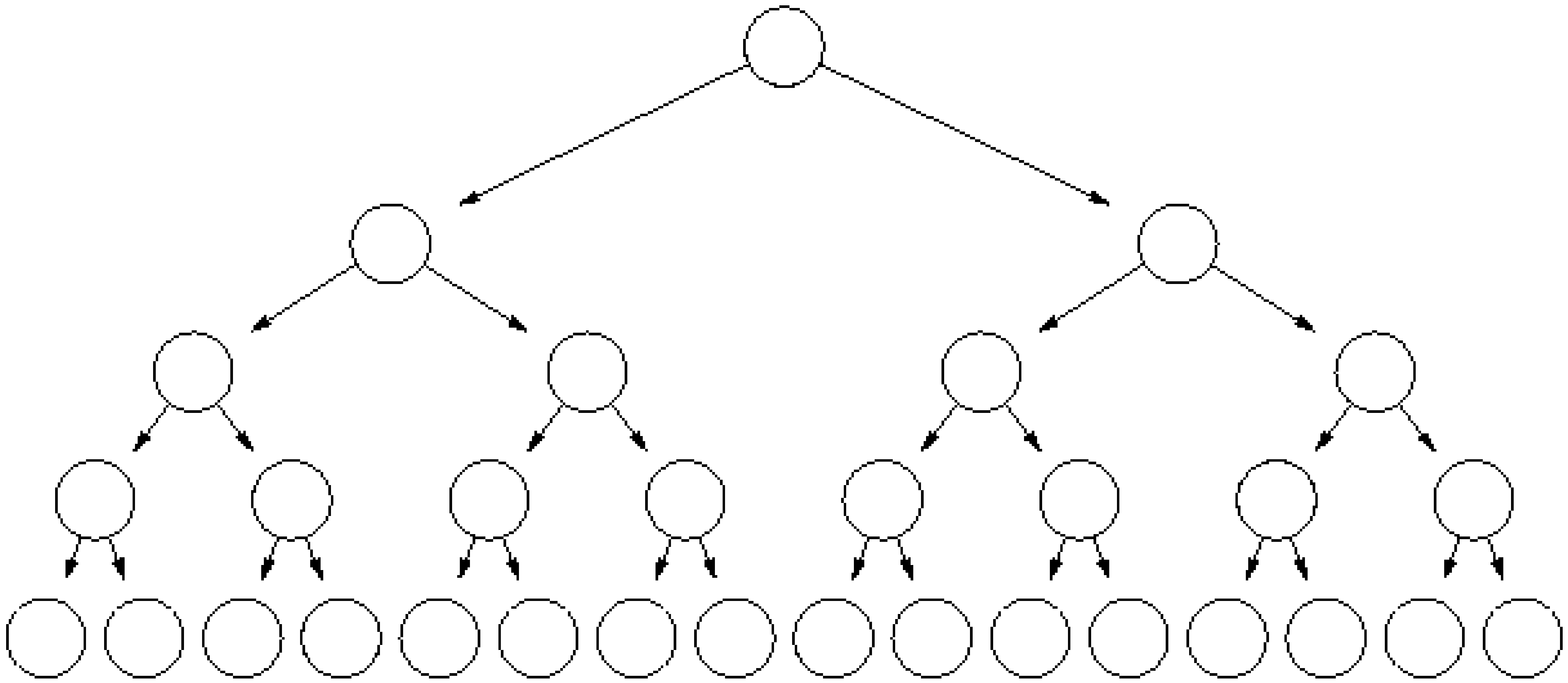
- **Search space** = problem space when a search algorithm is applied to the problem



Search space gets big fast!

- For 2 operators per state…2^n states per level, (2^(n+1)) - 1 total states

level	*states at this level*	*total states in tree*
0	1	1
1	2	3
2	4	7
3	8	15
4	16	31



Dealing with big search spaces

- Many techniques of AI attempt to deal with explosively sized search spaces
- Note also that the above operator equation is only unique states, it doesn't include possible repetitions!

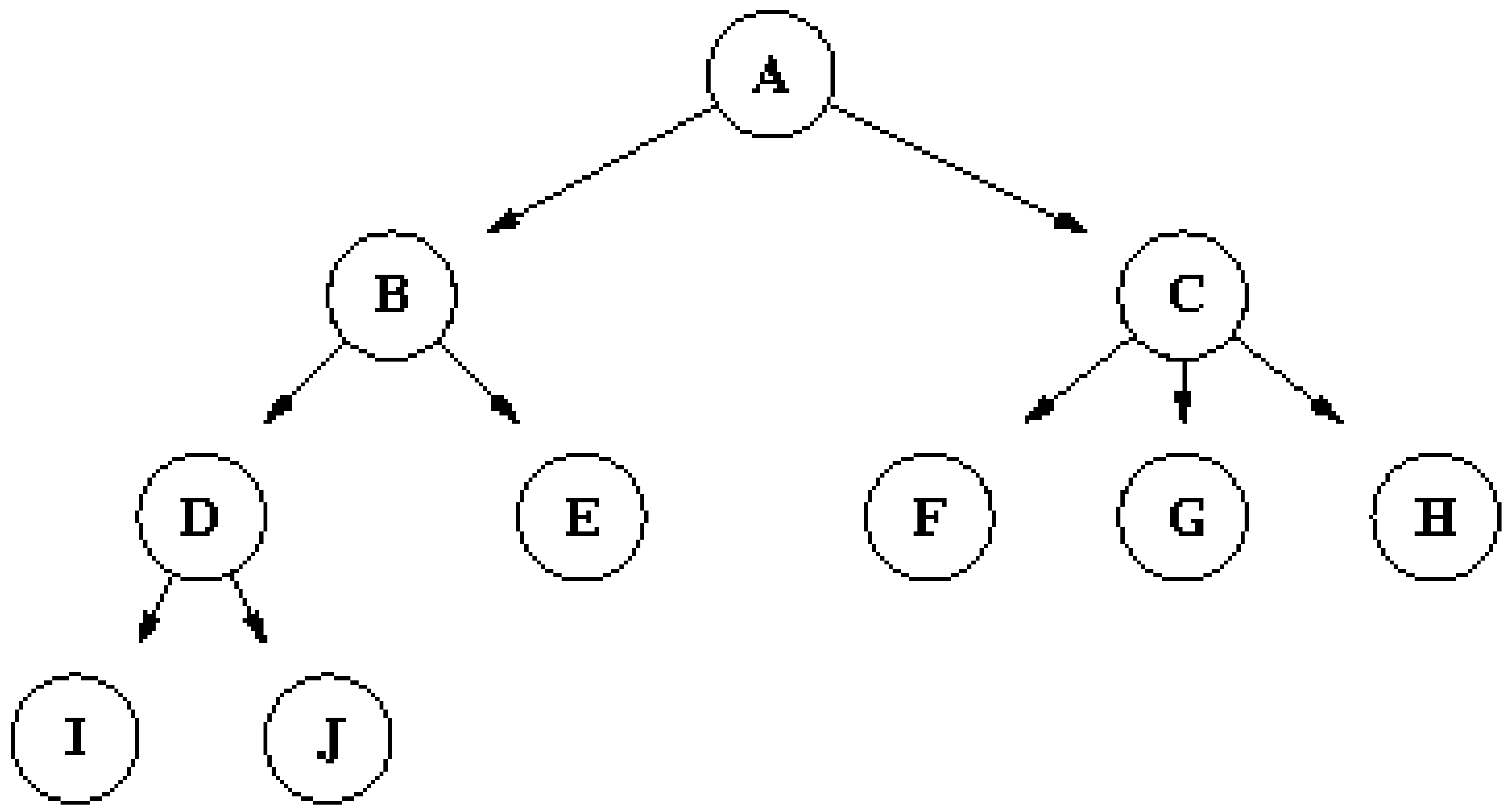
n	**2^n**	**2^(n+1)-1**
2	4	7
4	16	31
6	64	128
10	1024	2047
15	32768	65536
20	1048576	2097151
30	1073741824	2147483647

Finding the goal state

- start with the initial state A, and try to find the goal by applying operators to that state, and then to the states B and/or C that result, and so forth
- often one is also interested in
  - **finding the goal state as fast as possible**
  - **finding a solution that requires the minimum number of steps**
  - **finding a solution that satisfies some other requirements**

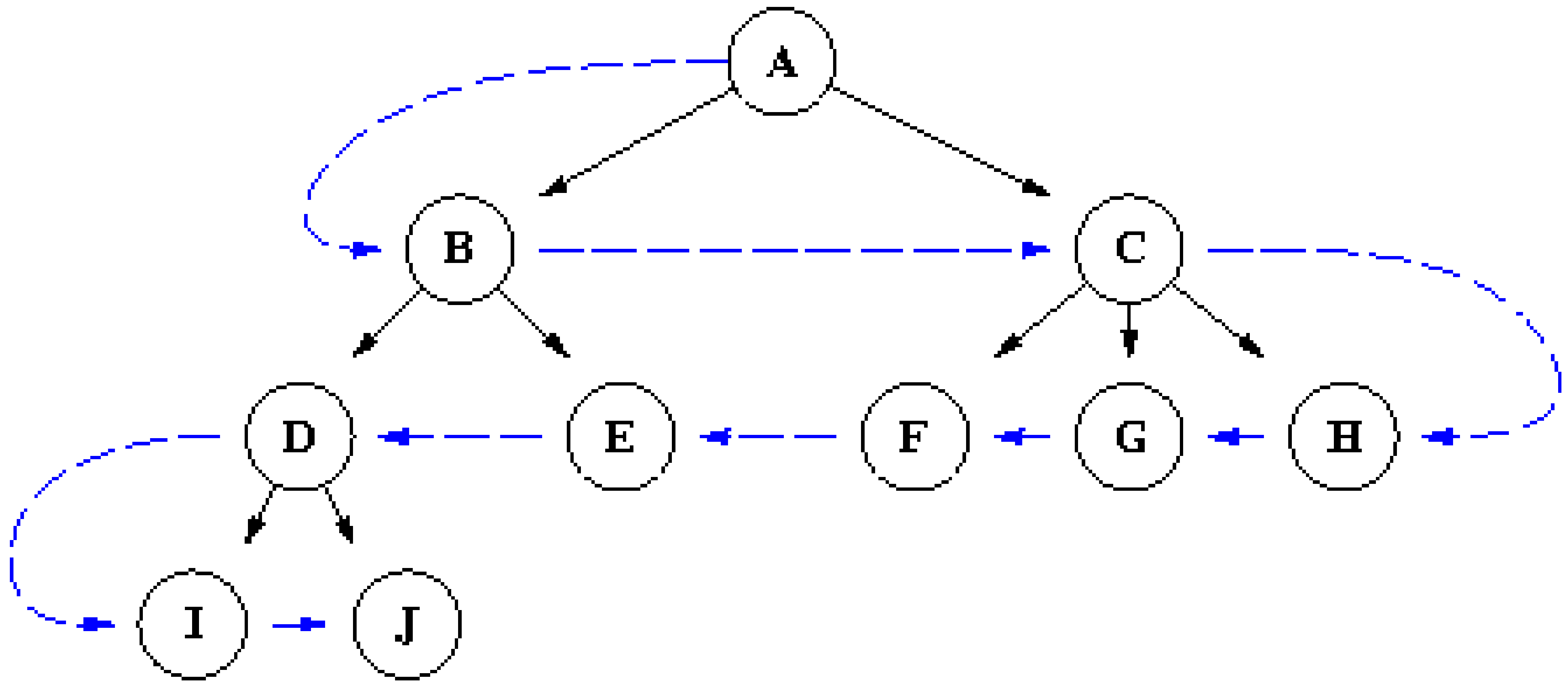
Breadth and depth first searches

- Consider the following space
  - **G is the goal**
  - **no operators apply to the states I, J, E, F and H**
  - **Of course we only start with A, not knowing the rest**



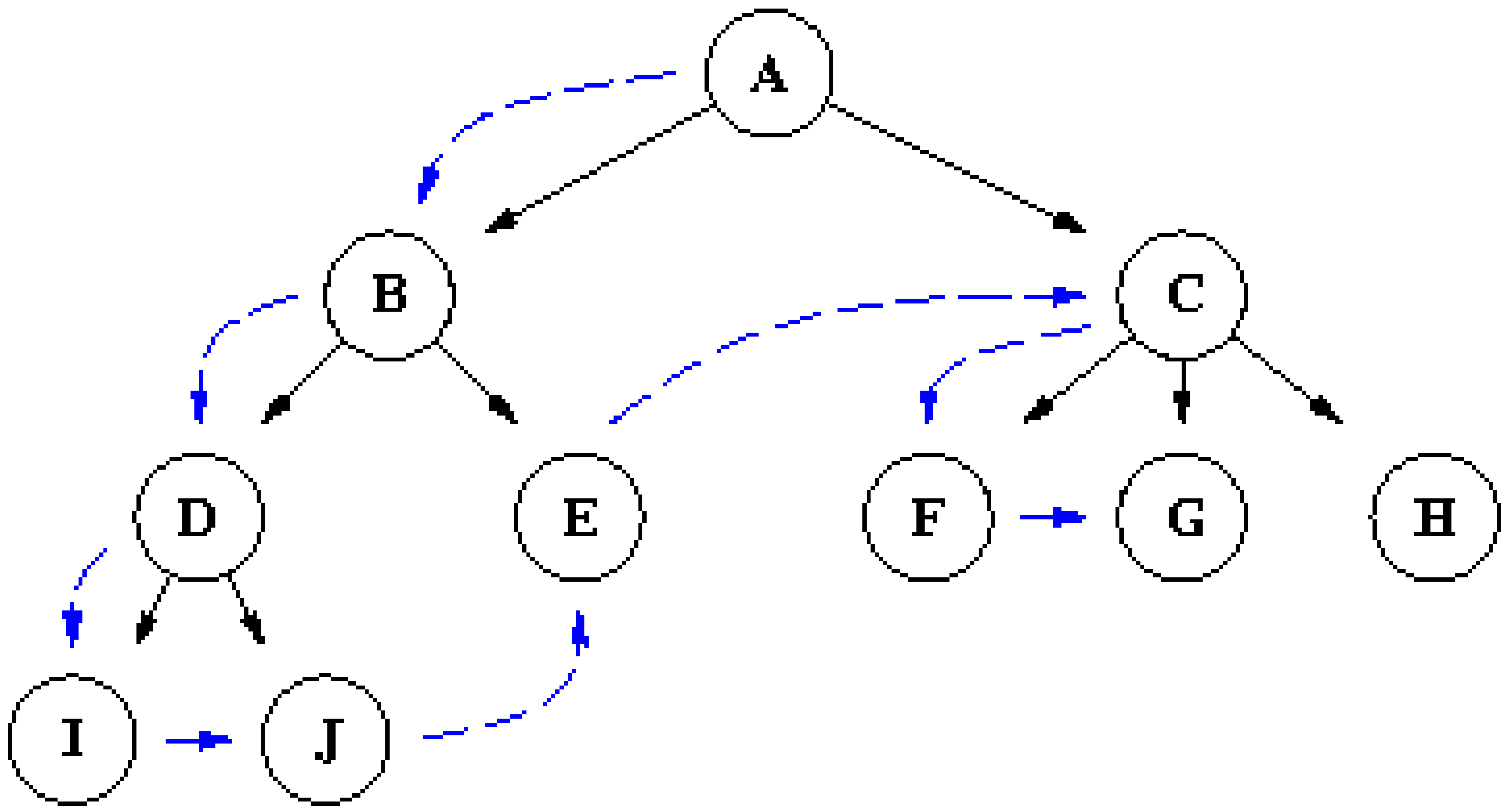
Breadth-first search

- all of the states at one level of the tree are considered before any of the states at the next lower level
- order in which the states at a given level are considered is not necessarily that shown in the diagram



Depth-first search

- After operators are applied to a state, one of the the resultant states is considered next
- If a node is a failure node or there are no operators that apply to it, the next node to be considered might be in the level above that of the current node
- We assumed that when a state is considered, all of the applicable operators are applied to the state. This isn't always necessary



Some useful properties of Breadth- and Depth- First Searches to know

- Simplicity is practical!
- Breadth-first search can be proved to possess the following properties

1. **If a solution exists in the search space, Breadth-first search will (eventually) find it**
2. **Breadth-first search will find the shortest possible solution, measured in terms of number of operator applications**

More properties

- Breadth-first search may take a while computationally, though it will find the path to the shortest answer by checking all the other possible paths
  - **If a solution is at level N, Breadth-first search will consider all the states down through level N before any further level, so if minimal solution at N, it will find it**
  - **But if N (minimal solution) is big, with 2 operators per state, Breadth-first search considers 2^N different states before solving the problem, whereas depth-first dives straight there (hopefully)**
- Depth-first search does not necessarily satisfy either of the above two properties
  - **In cases of infinite search space, might go down one branch and not come back even if the solution is the 2nd level next over!**
  - **When space is smaller, depth-first is generally faster**
  - **Also good when most of space is failure states**

Implementing searches

- You will read about a general search in more depth
- algorithm can be tailored to various types of searches including what we just mentioned (due to time)
  - **It's only a few pages on the website**
  - **Sample code will be available**

A quick intro to a general search algorithm

1.1 Make a "bag" containing the initial state.

  2.1 If the bag of states is empty, the search fails.

  2.2 Otherwise, remove a state from the bag.

  2.3 If that state achieves the goal, the search succeeds.

  2.4 Loop through the set of operators:

   3.3 If the operator applies to the state, apply the operator and add the resulting state to the bag.

  2.5 Continue at step 2.1.

There are many other, optimized forms of search

- Read about these as part of your final assignment
  - **Heuristic search**
  - **Best-first search**
  - **Hill climbing**
  - **Minimizing cost**
  - **A* search**
  - **Beam search**
  - **Two-way search**
  - **Island search**

Relation to optimization

- We've already learned about search methods
  - **Discussion of bracketing and golden ratio/section**
  - **Gradient descent**
  - **Minimization in general**
    - Local vs. global solutions
- Now you have a sense of how these methods are important for AI concepts, and how to extend them into studies of cognition, behavior, and the human condition

The evolution of our concepts of modeling and data analysis in this course

- We've developed an approach to modeling and analyzing physical systems which can be applied to cognitive modeling from many perspectives
  - **Capturing, processing and analyzing data**
    - Data processing, filtering, manipulation
      - **Computational tools, from pen and paper to matlab**
  - **Visualization of that data**
  - **Modeling behavioral data**
    - Formulating the problem
    - Statistical
    - Computational
    - Evaluation criterion formulation (how well did the model perform?)
  - **Communication of results**
    - Reading others' results and insights
    - Creating reports, collecting results and presenting them clearly

Where to go from here

- You now have basic tools you can apply to a variety of problems
- Depending on the direction you intend to go, the most important thing to do is to apply any techniques you really want to learn well
  - **Practice**
- Many topics have been introduced
  - **Choose references given and read**