

# *CSCE 580: Introduction to AI*

## Lecture 6: Search

---

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

5<sup>TH</sup> SEP 2024

**Carolinian Creed: “I will practice personal and academic integrity.”**

**Credits: Copyrights of all material reused acknowledged**

# Organization of Lecture 6

---

- Introduction Segment
  - Recap of Lecture 5
- Main Segment
  - Problem solving agent – goal directed
  - Problem formulation – abstraction, type of problems
  - Search
- Concluding Segment
  - Course Project Discussion
  - About Next Lecture – Lecture 7
  - Ask me anything

**Project 1: (Sep 5 – Oct 3)**

# Introduction Section

---

# Recap of Lecture 5

---

- Logic – First Order
- Inferencing
- Representation in the Large: ConceptNet, Cyc
- Trust Issues with Knowledge Representation

# Resume Exercise – Programming: Word Processing

---

1. Read your resume in text and get list of words, call `resume_words`
2. Plot a histogram of top 100 words, i.e., bar graph of words and counts
3. Have a list of common words – stop words
  1. Articles: a, an, the
  2. Propositions, determiners
4. Remove stop words from `resume_words`
5. Plot a histogram of top 100 words, i.e., bar graph of words and counts
6. Observation: Notice any difference?

# Resume Exercise – Programming: Word Tag Cloud

---

- Task 1
  - Take your resume
  - Create word tag cloud for your resume
- Task 2
  - Take all resumes in folder
  - Create word tag cloud for multiple (5 or more, even all) resumes
  - Inspect and compare images of tasks 1 and 2
- Observation: Notice any difference?

**Sample Code for Reference:**

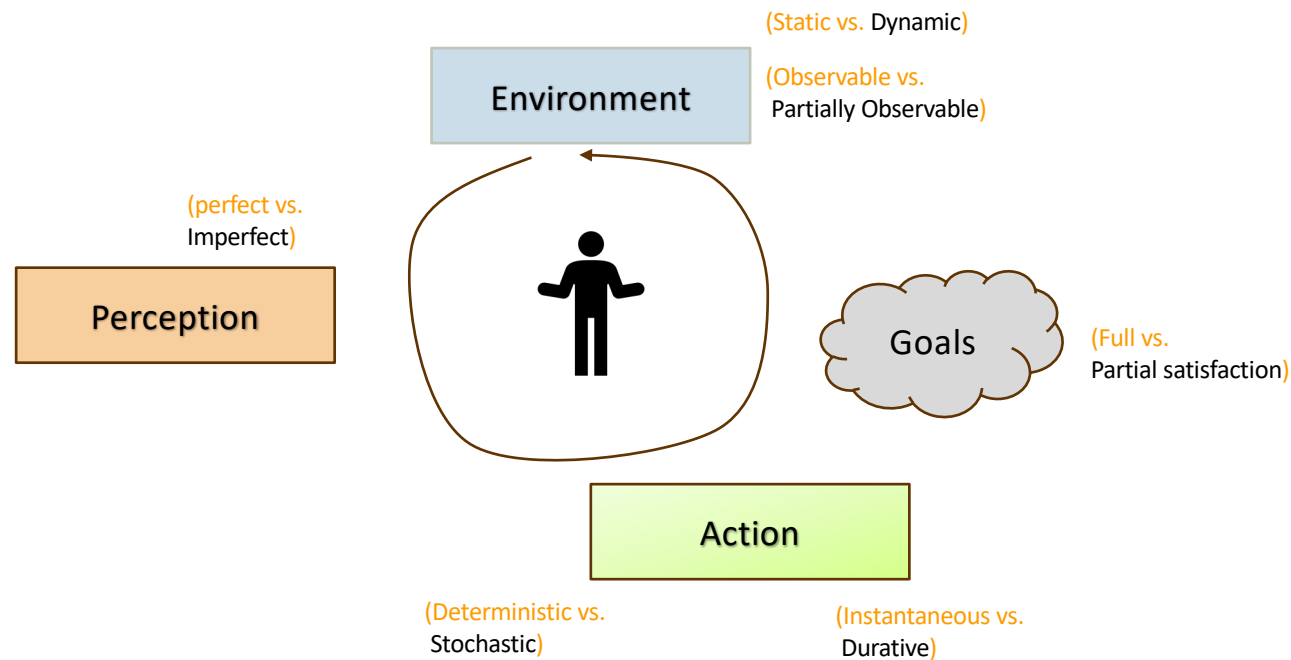
[https://github.com/biplav-s/course-nl-f22/blob/main/sample-code/l1-wordcloud/FirstLook\\_ClassSyllabusData.ipynb](https://github.com/biplav-s/course-nl-f22/blob/main/sample-code/l1-wordcloud/FirstLook_ClassSyllabusData.ipynb)

# Resume Exercise - Report

---

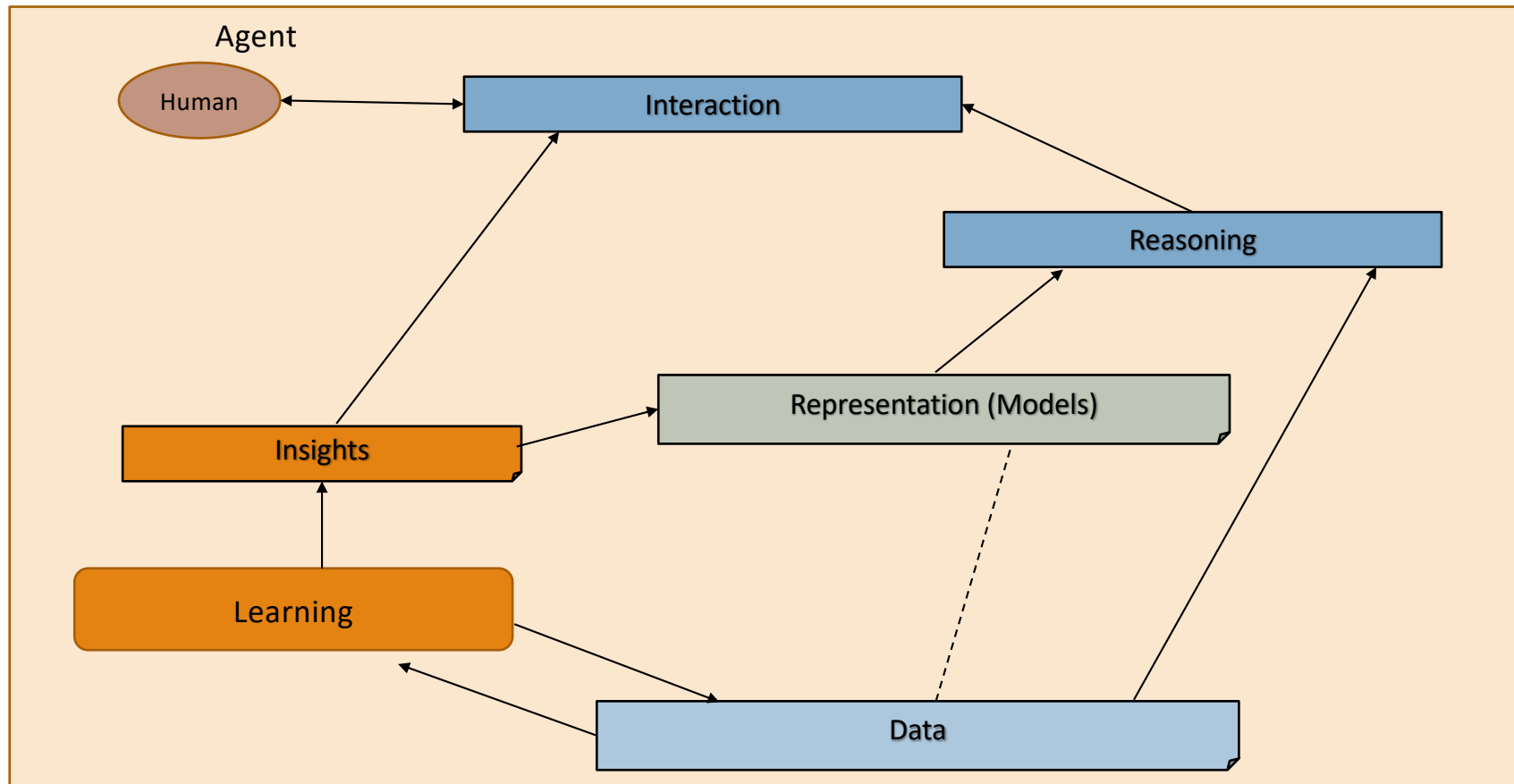
- T1: Put the code in your Github in sub-folder: “resume-exercise”
- T2: Document the finding on Github or Google drive as .pdf (.docx optional)  
(Create a Google doc in your Google drive called “Resume-Exercise”. In it,)
  - Show the two histograms
    - (top 100 and top-100 after stop words)
    - Comment on any difference you see
  - Show the two word tag clouds
    - (your resume and of the class)
    - Comment on any difference you see
- Do it by next Thursday – Sep 5, 2024

# Intelligent Agent Model





# Relationship Between Main AI Topics



# Where We Are in the Course

## CSCE 580/ 581 – In This Course

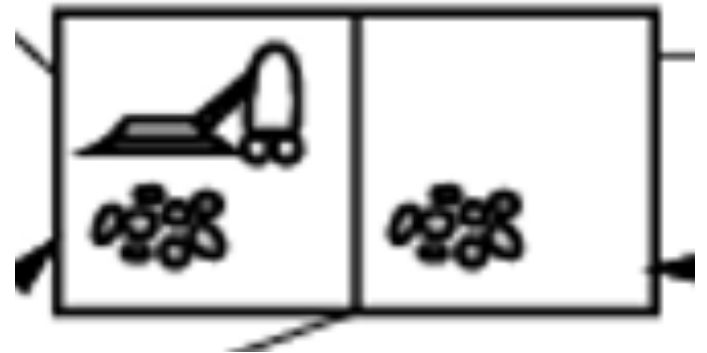
- Week 1: Introduction, Aim: Chatbot / Intelligence Agent
- Weeks 2-3: Data: Formats, Representation and the Trust Problem
- Week 4-5: Search, Heuristics - Decision Making
- Week 6: Constraints, Optimization – Decision Making
- Week 7: Classical Machine Learning – Decision Making, Explanation
- Week 8: Machine Learning - Classification
- Week 9: Machine Learning - Classification – Trust Issues and Mitigation Methods
- Topic 10: Learning neural network, deep learning, Adversarial attacks
- Week 11: Large Language Models – Representation, Issues
- Topic 12: Markov Decision Processes, Hidden Markov models - Decision making
- Topic 13: Planning, Reinforcement Learning – Sequential decision making
- Week 14: AI for Real World: Tools, Emerging Standards and Laws; Safe AI/ Chatbots

# Main Section

---

# Example: Vacuum World

- Situation
  - Two rooms
  - One robot
  - Dirt can be in any room
- Goal
  - Clean the rooms
- Actions
  - Move left, move right, clean

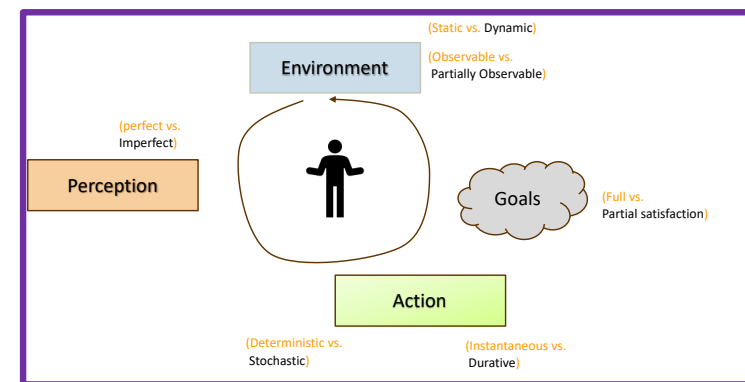


Adapted from:

1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

# Goal-directed Problem Solving Agents

1. Goal Formulation: Have one or more (desirable) world states
2. Problem formulation: What actions and states to consider given goals and an initial state
3. Search for solution: Given the problem, search for a solution - a sequence of actions to achieve the goal starting from the initial state
4. Execution: agent can execute actions in the solution

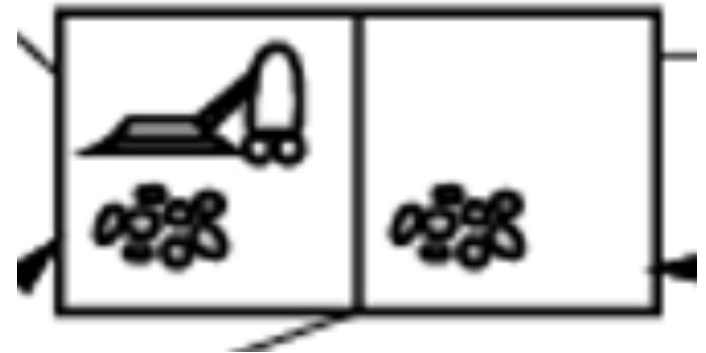


Adapted from:

1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

# Modeling and Abstraction Consideration

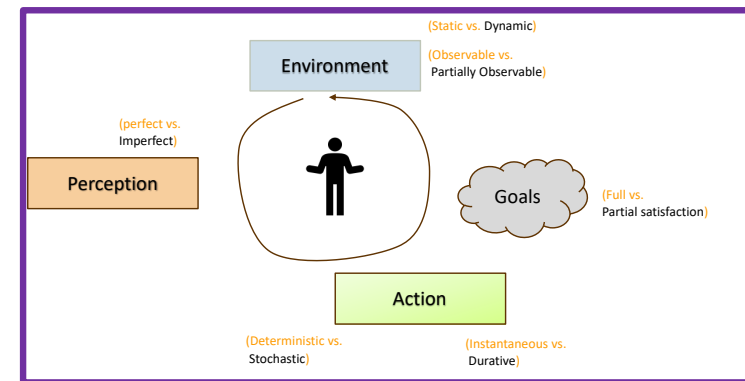
- Model: an abstract representation of the problem
  - “All models are wrong, but some are useful”
- What to capture, what to avoid
  - Only the necessary details needed to solve the problem
- In the example, we can avoid
  - For concepts
    - Size of rooms or robot
    - Quantity of dirt
  - For actions
    - Time taken to clean
    - Charging/ recharging time
    - Doing nothing – staying at the same place?



- Concepts
  - Two rooms
  - One robot
  - Dirt can be in any room
- Goal
  - Clean the rooms
- Actions
  - Move left, move right, clean

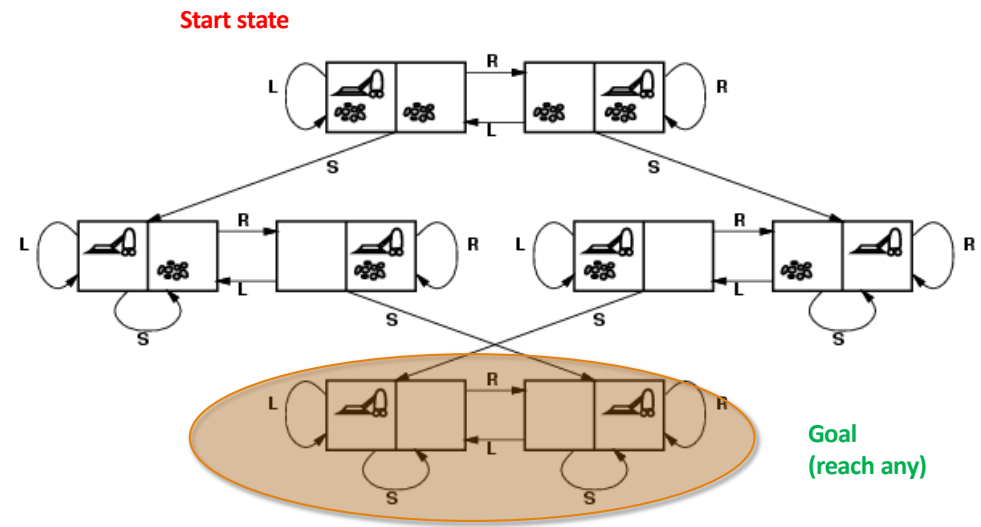
# Open Loop v/s Closed Loop Systems

- Open loop
  - Assuming the world will not change, after a solution is found, one can simply execute it one action at a time
- Closed loop
  - If the world keeps changing, after a solution is found, one cannot ignore perception when executing actions
  - The solution has to be relooked whenever an action is being executed. New solutions may have to be found at each step again.



# Formulating a Problem

States	8 possible world states <i>(2room x 2dirt location x 2clean?)</i>
<ul style="list-style-type: none"> <li>Initial state</li> <li>Goal state</li> </ul>	<ul style="list-style-type: none"> <li>Any</li> <li>No dirt at all locations</li> </ul>
Actions	Left, Right, Suck
<ul style="list-style-type: none"> <li>Transition model</li> <li>Action cost</li> </ul>	<ul style="list-style-type: none"> <li>Action transition (edges)</li> <li>1</li> </ul>



Adapted from:

1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course



# Type of Problems

---

## Standardized Problems

- Grid world
- Sliding tile
- Sokoban
- Chess
- ...

## Real-World Problems

- Route finding
- Robotic / space craft navigation
- Protein design: find a sequence of amino acids that will fold into a 3D protein structure
- Dialog generation: how to give an effective answer that a person can understand
- ...

# Exercise: Sliding 8-tile Puzzle

## States

- Initial state
- Goal state

## Actions

- Transition model
- Action cost

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Adapted from:  
Russell & Norvig, AI: A Modern Approach

# Exercise: Sliding 8-tile Puzzle

<b>States</b> <ul style="list-style-type: none"><li>• Initial state</li><li>• Goal state</li></ul>	<b>Location of tiles</b> <ul style="list-style-type: none"><li>• Any (given)</li><li>• All numbers sorted, Empty tile in corner (given)</li></ul>
<b>Actions</b> <ul style="list-style-type: none"><li>• Transition model</li><li>• Action cost</li></ul>	move blank left, right, up, down <ul style="list-style-type: none"><li>• Blank transition (edges)</li><li>• 1</li></ul>

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Adapted from:

1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

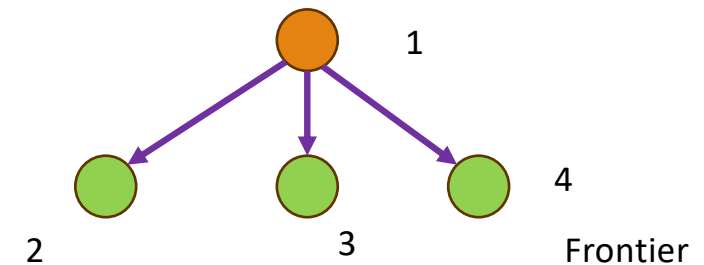
# Search

---

# Search Basics

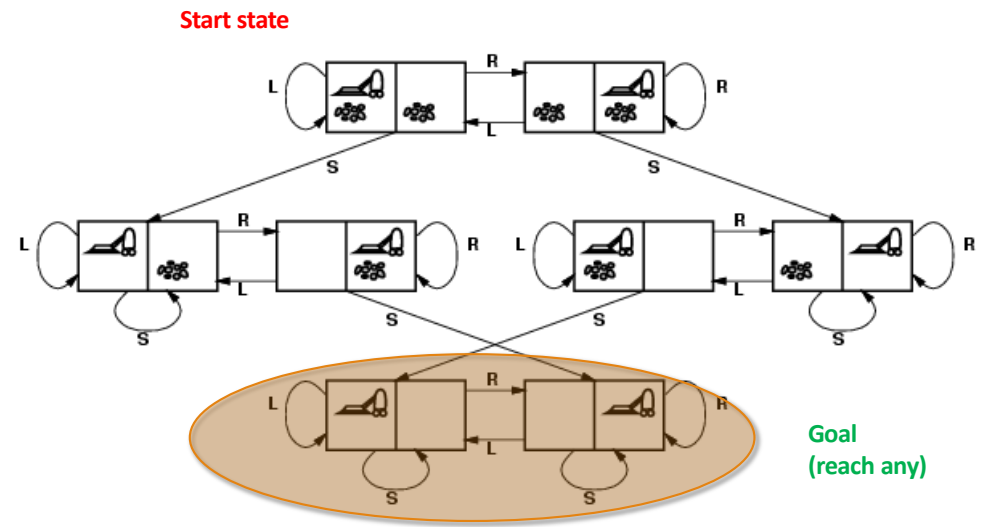
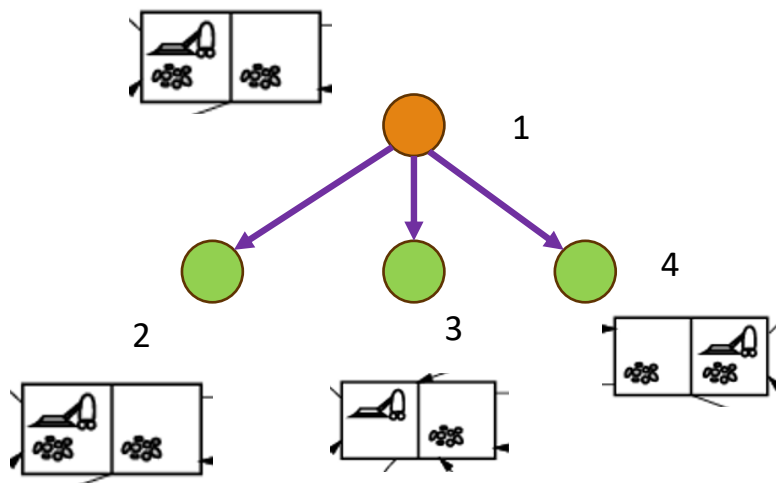
---

- input: a problem with states and actions
- output: solution(s) or flag for failure
- Concepts:
  - **Node**: corresponds to a state of the problem
  - **Edges**: transition between states
  - **Expand**: consider actions in the state (ACTIONS) and transition model. Generate new nodes corresponding to resulting states (RESULT)
  - **Explore**: check when a node meets goal condition



Node 1 has been **Reached**, Nodes {2,3 4} constitute Node 1's **Frontier**

# Formulating a Problem



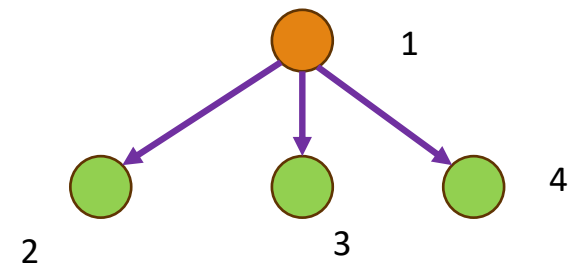
Adapted from:

1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

# Tree-search Algorithms

**Basic idea:** simulated exploration of state space by generating successors of already-explored states (a.k.a. ~ **expanding** states)

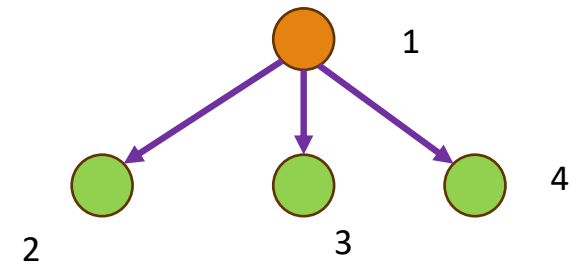
```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
```



Adapted from:  
1. Russell & Norvig, AI: A Modern Approach  
2. Bart Selman's CS 4700 Course

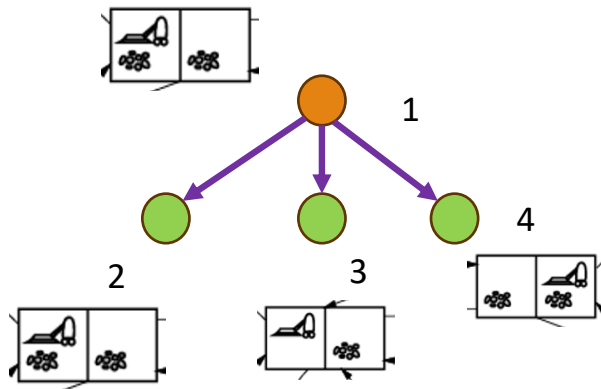
# Implementing a Tree-Search Algorithm

- Data structure
  - node.STATE: the state to which the node corresponds
  - node.PARENT: the node in the tree that generated this node
  - node.ACTION: the action that was applied to the parent to generate this node
  - node.PATH-COST: the total cost of the path from the initial state to this node
- Queue to store frontier
  - Is-EMPTY(frontier): true/ false depending on whether frontier is empty
  - POP(frontier): remove the top node from the frontier and return it
  - TOP(frontier): returns the top node from the frontier but does not remove it
  - ADD(node, frontier): insert node into its proper place in the queue
- Queue:
  - priority queue – removes the node with minimum cost according to some evaluation function
  - FIFO queue – first in, first output. Used in breadth first search
  - LIFO queue - last in, first output. Used in depth first search





# Best-First Search



```

function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
  node  $\leftarrow$  NODE(STATE=problem.INITIAL)
  frontier  $\leftarrow$  a priority queue ordered by f, with node as an element
  reached  $\leftarrow$  a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node  $\leftarrow$  POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s  $\leftarrow$  child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s]  $\leftarrow$  child
        add child to frontier
  return failure
  
```

```

function EXPAND(problem, node) yields nodes
  s  $\leftarrow$  node.STATE
  for each action in problem.ACTIONS(s) do
    s'  $\leftarrow$  problem.RESULT(s, action)
    cost  $\leftarrow$  node.PATH-COST + problem.ACTION-COST(s, action, s')
    yield NODE(STATE=s', PARENT=node, ACTION=action, PATH-COST=cost)
  
```

# Examples of Search Strategies

---

- Uninformed
  - Depth first
  - Breadth first
- Informed (Heuristic)
  - Greedy best first search
  - A\* search

# More on Search Strategies

---

- A search strategy is defined by picking the **order of node expansion**.
- Strategies are evaluated along the following dimensions:
  - **completeness**: does it always find a solution if one exists?
  - **time complexity**: number of nodes generated
  - **space complexity**: maximum number of nodes in memory
  - **optimality**: does it always find a least-cost solution?
- Time and space complexity are measured in terms of
  - ***b***: maximum branching factor of the search tree
  - ***d***: depth of the least-cost solution
  - ***m***: maximum depth of the state space (may be  $\infty$ )

Adapted from:

1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

# Exercise and Code

---

- Search Methods
  - From Book: AI – A Modern Approach,  
<https://github.com/aimacode/aima-python/blob/master/search.ipynb>

Source: Russell & Norvig, AI: A Modern Approach

# Lecture 6: Summary

---

- We talked about
  - Goal-directed problem solving agents
  - How to formulate problem formulations
  - Search concepts
    - Problems of controlled robot navigation, 8-tile
  - Search strategies

# Concluding Section

---

# Course Project

---

# Discussion: Projects

---

- New: two projects
  - Project 1: model assignment
  - Project 2: single problem/ llm based solving / fine-tuning/ presenting result



# Project Discussion

---

1. Go to Google spreadsheet against your name
2. Enter model assignment name and link from (<http://modelai.gettysburg.edu/> )

1. Create a private Github repository called “CSCE58x-Fall2024-<studentname>-Repo”. Share with Instructor (biplav-s) and TA (vishalpallagani)
2. Create Google folder called “CSCE58x-Fall2024-<studentname>-SharedInfo”. Share with Instructor ([prof.biplav@gmail.com](mailto:prof.biplav@gmail.com)) and TA ([vishal.pallagani@gmail.com](mailto:vishal.pallagani@gmail.com))
3. Create a Google doc in your Google repo called “Project Plan” and have the following by next class (Sep 5, 2024)

## Timeline

1. Title:
2. Key idea: (2-3 lines)
3. Data need:
4. Methods:
5. Evaluation:
6. Milestones
  1. // Create your own
7. Oct 3, 2024

# Reference: Project 1 Rubric (30% of Course)

## Assume total for Project-1 as 100

- **Project results** – 60%
  - Working system ? – 30%
  - Evaluation with results superior to baseline? – 20%
  - Went through project tasks completely ? – 10%
- **Project efforts** – 40%
  - Project report – 20%
  - Project presentation (updates, final) – 20%
- **Bonus**
  - Challenge level of problem – 10%
  - Instructor discretion – 10%
- **Penalty**
  - Lack of timeliness as per your milestones policy (right) - up to 30%

## Milestones and Penalties

- Project plan due by Sep 5, 2024 [-10%]
- Project deliverables due by Oct 3, 2024 [-10%]
- Project presentation on Oct 8, 2024 [-10%]

# About Next Lecture – Lecture 7

---

# Lecture 7: Searching for Problem Solving

---

- Search
- Heuristics
- Quiz 1 will be released, due on Thursday, Sep 12 (end-of-day)