CSCE 580: Introduction to AI
CSCE 581: Trusted AI

Lecture 6: Search

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

12TH SEP 2023

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

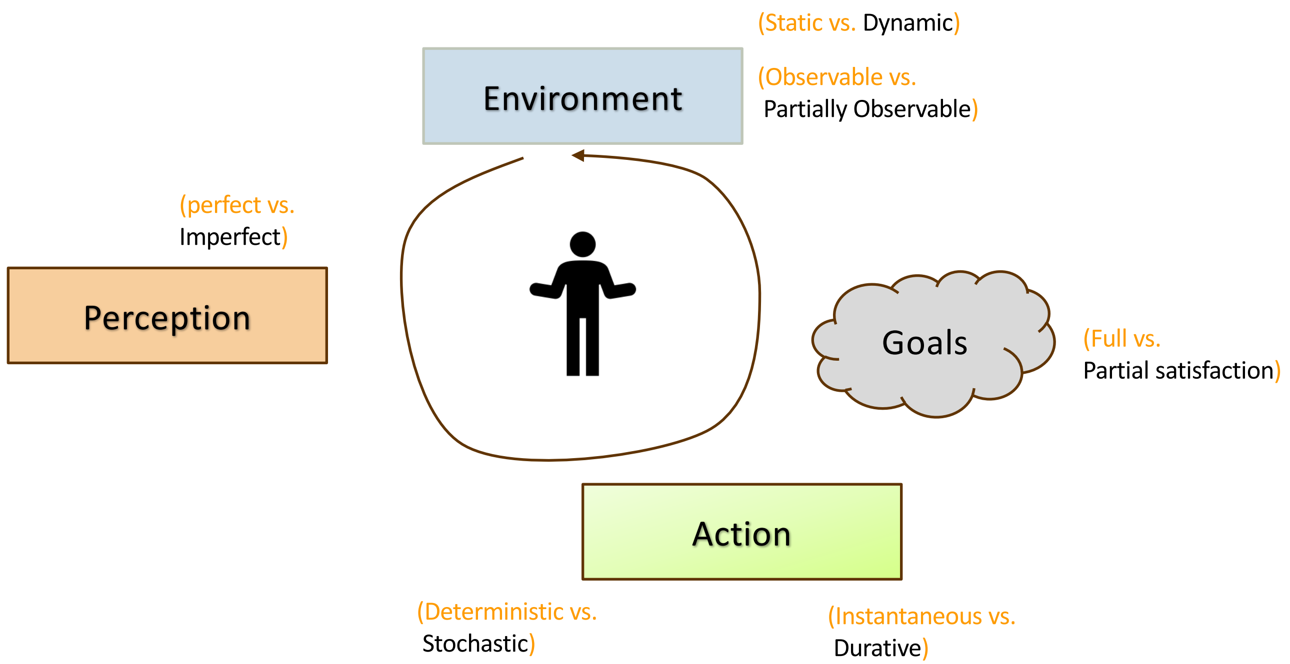
Sprint 1: (Sep 12 – Oct 5) STARTS

Introduction Section

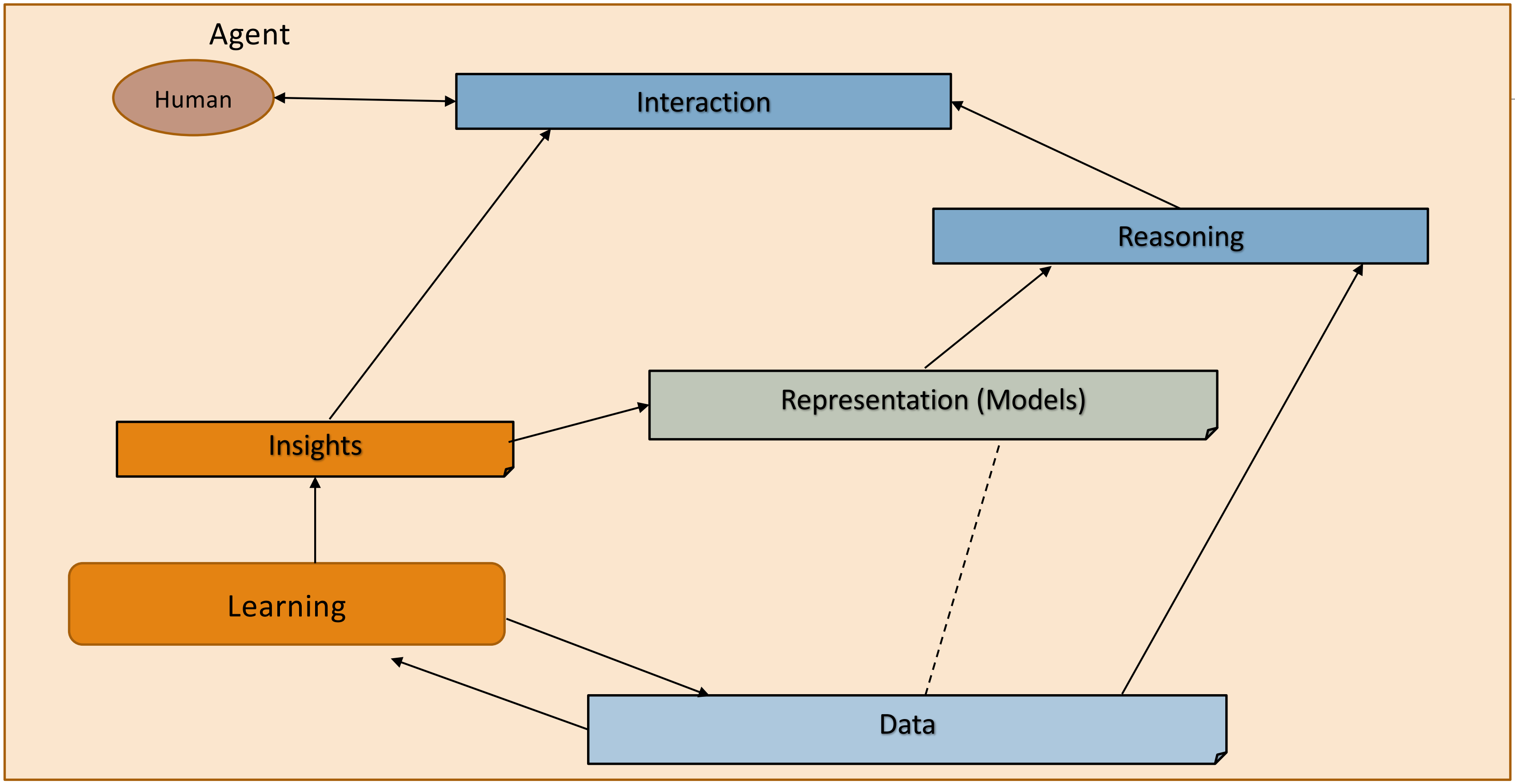
Recap of Lecture 5

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

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

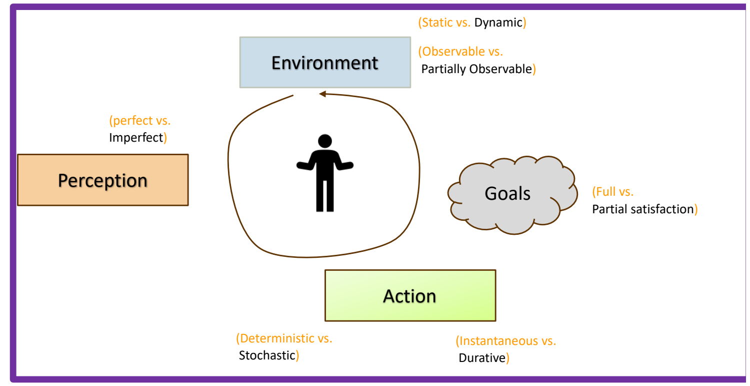
Example: Vacuum World

- 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

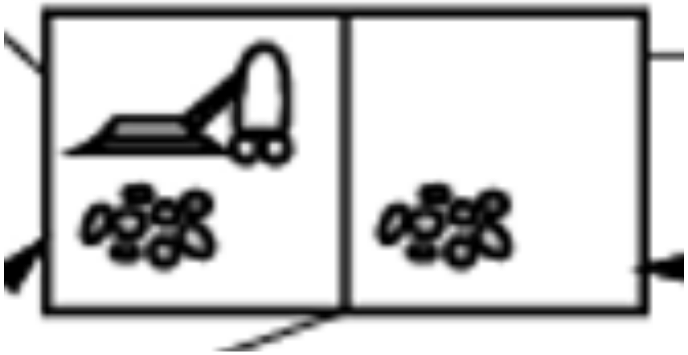
1. Goal Formulation: Have one or more (desirable) world states
2. Problem formulation: What actions and states to consider given goals and an initial state
3. Search for solution: Given the problem, search for a solution - a sequence of actions to achieve the goal starting from the initial state
4. Execution: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

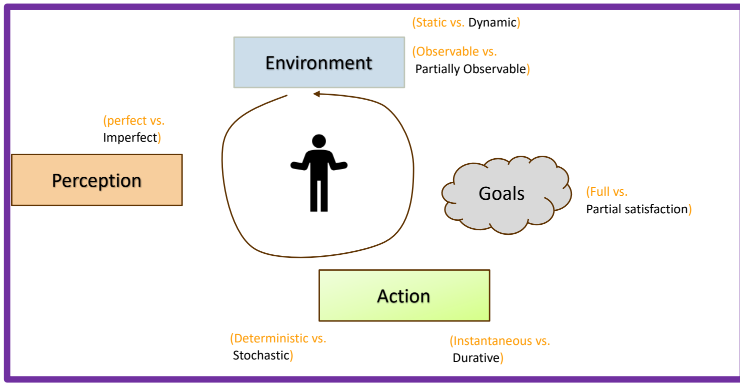
- Model: an abstract representation of the problem
  - “All models are wrong, but some are useful”
- What to capture, what to avoid
  - Only the necessary details needed to solve the problem
- In the example, we can avoid
  - For concepts
    - Size of rooms or robot
    - Quantity of dirt
  - For actions
    - 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

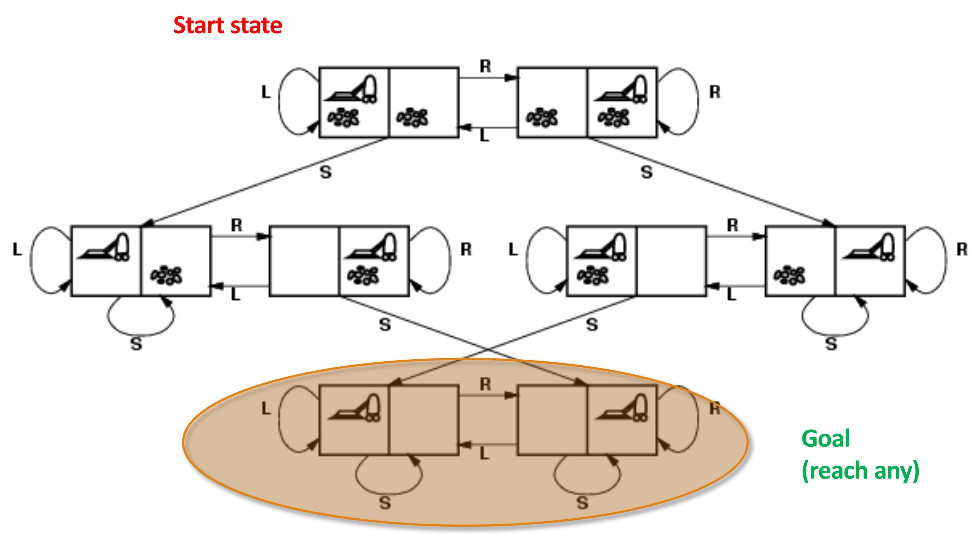
Open Loop v/s Closed Loop Systems

- 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 • Initial state • Goal state	8 possible world states *(2room x 2dirt location x 2clean?)* • Any • No dirt at all locations
Actions • Transition model • Action cost	Left, Right, Suck • Action transition (edges) • 1



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

Type of Problems

Standardized Problems

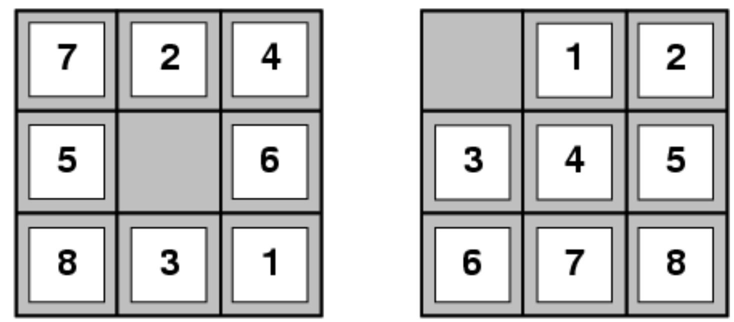
- Grid world
- Sliding tile
- Sokoban
- Chess
- …

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	

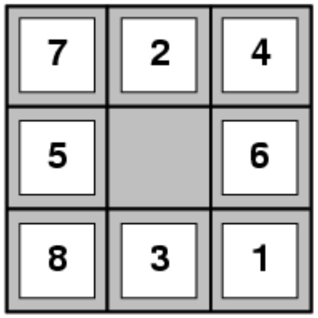


Start State Goal State

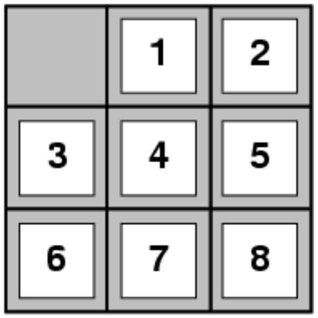
Adapted from:
Russell & Norvig, AI: A Modern Approach

Exercise: Sliding 8-tile Puzzle

States • Initial state • Goal state	Location of tiles • Any (given) • All numbers sorted, Empty tile in corner (given)
Actions • Transition model • Action cost	move blank left, right, up, down • Blank transition (edges) • 1



Start State



Goal State

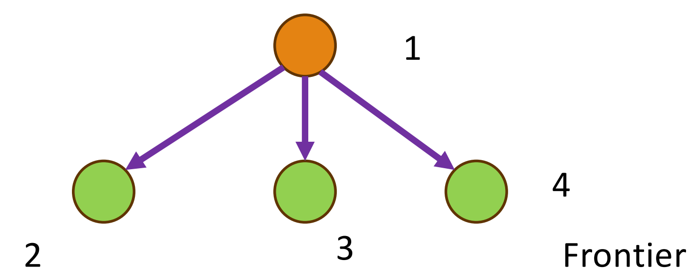
Adapted from:
1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

Search

Search Basics

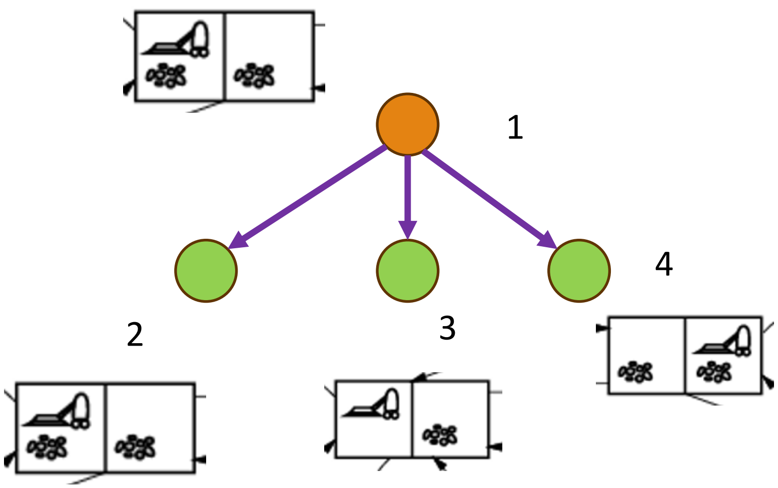
- input: a problem with states and actions
- output: solution(s) or flag for failure

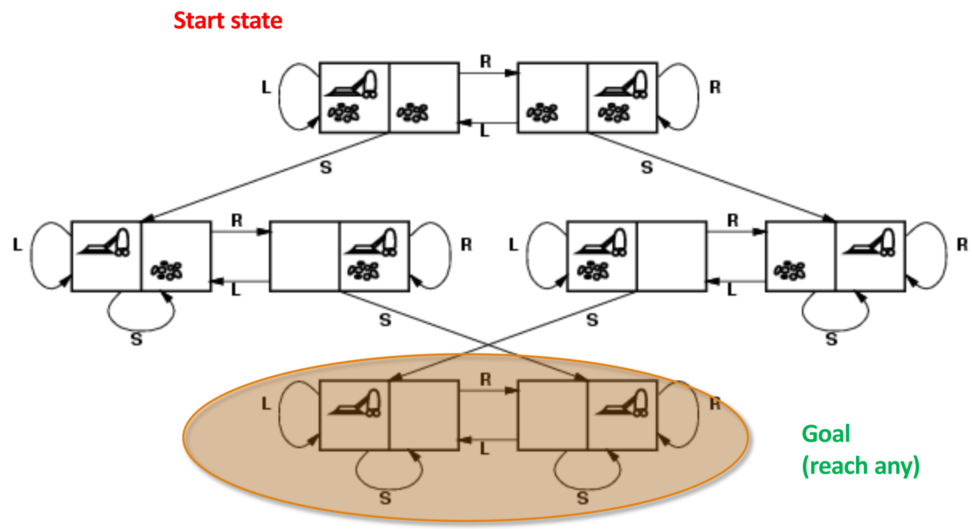
- Concepts:
  - **Node**: corresponds to a state of the problem
  - **Edges**: transition between states
  - **Expand**: consider actions in the state (ACTIONS) and transition model. Generate new nodes corresponding to resulting states (RESULT)
  - **Explore**: check when a node meets goal condition



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

Formulating a Problem

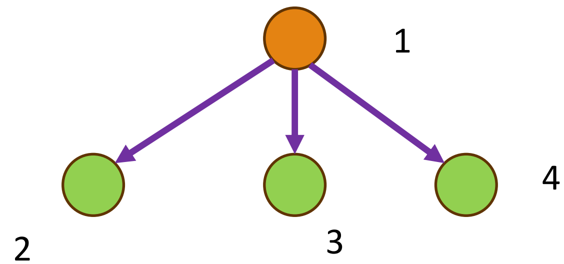




Adapted from:
1. Russell & Norvig, AI: A Modern Approach
2. Bart Selman's CS 4700 Course

Tree-search Algorithms

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

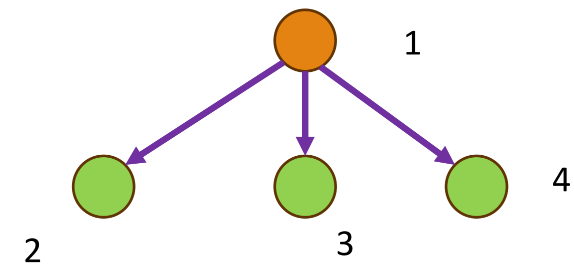


function TREE-SEARCH(*problem, strategy*) **returns** a solution, or failure
  initialize the search tree using the initial state of *problem*
  loop do
    if there are no candidates for expansion **then return** failure
    choose a leaf node for expansion according to *strategy*
    if the node contains 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

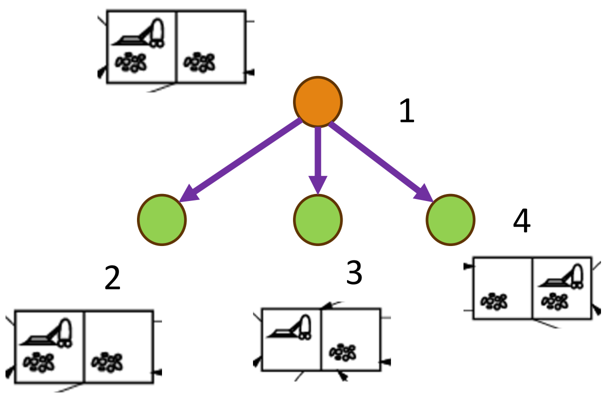
Implementing a Tree-Search Algorithm

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

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

Source: Russell & Norvig, AI: A Modern Approach

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 ∞)

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

Project Discussion: What Problem Fascinates You ?

- Data
  - Water
  - Finance
  - …
- Analytics
  - Search, Optimization, Learning, Planning, …
- Application
  - Building chatbot
- Users
  - Diverse demographics
  - Diverse abilities
  - Multiple human languages

Project execution in sprints

- Sprint 1: (Sep 12 – Oct 5)
  - Solving: Choose a decision problem, identify data, work on solution methods
  - Human interaction: Develop a basic chatbot (no AI), no problem focus
- Sprint 2: (Oct 10 – Nov 9)
  - Solving: Evaluate your solution on problem
  - Human interaction: Integrated your choice of chatbot (rule-based or learning-based) and methods
- Sprint 3: (Nov 14 – 30)
  - **Evaluation:** Comparison of your solver chatbot with an LLM-based alternative, like ChatGPT

Project Discussion: Dates and Deliverables

Project execution in sprints

- Sprint 1: (Sep 12 – Oct 5)
  - Solving: Choose a decision problem, identify data, work on solution methods
  - Human interaction: Develop a basic chatbot (no AI), no problem focus
- Sprint 2: (Oct 10 – Nov 9)
  - Solving: Evaluate your solution on problem
  - Human interaction: Integrated your choice of chatbot (rule-based or learning-based) and methods
- Sprint 3: (Nov 14 – 30)
  - **Evaluation:** Comparison of your solver chatbot with an LLM-based alternative, like ChatGPT

- Oct 12, 2023
  - Project checkpoint
  - In-class presentation
- Nov 30, 2023
  - Project report due
- Dec 5 / 7, 2023
  - In-class presentation

Skeleton: A Basic Chatbot

- Run in an infinite loop until the user wants to quit
- Handle any user response
  - User can quit by typing “Quit” or “quit” or just “q”
  - User can enter any other text and the program has to handle it. The program should write back what the user entered and say – “I do not know this information”.
- Handle <u>known</u> user query types // Depends on your project
  - “Tell me about N-queens”, “What is N ?”
  - “Solve for N=4?”
  - “Why is this a solution? ”
- Handle <u>chitchat</u> // Support at least 5, extensible from a file
  - “Hi” => **“Hello”**
  - …
- ***Store session details in a file***

<u>**Illustrative Project**</u>

1. **Title**: Solve and explain solving of n-queens puzzle
2. **Key idea**: Show students how a course project will look like
3. **Who will care when done**: students of the course, prospective AI students and teachers
4. **Data need**: n: the size of game; interaction
5. **Methods**: search
6. **Evaluation**: correctness of solution, quality of explanation, appropriateness of chat
7. **Users**: with and without AI background; with and without chess background
8. **Trust issue**: user may not believe in the solution, may find interaction offensive (why queens, not kings? …)

Project Discussion: Illustration

1. Create a private Github repository called “CSCE58x-Fall2023-<studentname>-Repo”. Share with Instructor (biplav-s) and TA (kausik-l)
2. Create Google folder called “CSCE58x-Fall2023-<studentname>-SharedInfo”. Share with Instructor (prof.biplav@gmail.com) and TA (lakkarajukausik90@gmail.com)
3. Create a Google doc in your Google repo called “Project Plan” and have the following by next class (Sep 5, 2023)

1. **Title**: Solve and explain solving of n-queens puzzle
2. **Key idea**: Show students how a course project will look like
3. **Who will care when done**: students of the course, prospective AI students and teachers
4. **Data need**: n: the size of game; interaction
5. **Methods**: search
6. **Evaluation**: correctness of solution, quality of explanation, appropriateness of chat
7. **Users**: with and without AI background; with and without chess background
8. **Trust issue**: user may not believe in the solution, may find interaction offensive (why queens, not kings? …)

Reference: Project Rubric

- **Project results** – 60%
  - Working system ? – 30%
  - Evaluation with results superior to baseline? – 20%
  - Considered related work? – 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 announced policy (right) - up to 30%

Milestones and **Penalties**

- Oct 12, 2023
  - Project checkpoint
  - In-class presentation
  - **Penalty: presentation not ready by Oct 10, 2023 [-10%]**
- Nov 30, 2023
  - Project report due
  - **Project report not ready by date [-10%]**
- Dec 5 / 7, 2023
  - In-class presentation
  - **Project presentations not ready by Dec 4, 2023 [-10%]**

About Next Lecture – Lecture 7

Lecture 7: Searching for Problem Solving

- Search
- Heuristics
- Quiz 1 will be released, due on Monday, Sep 18 (end-of-day)