1_1 Introduction

- Knowledge Representation
  - Reasoning : Draw conclusions from knowledge
    - Deduction
      - Proofs (mathematical formalization)
    - Belief Revision
    - Causality
  - Knowledge:
    - Facts
      - Symbollic (Logic)
      - Numeric (Probability)
    - Uncertain (Beliefs)
      - Derived from probability theory

Natural Language Processing

- Classical: Analyze languages and use knowledge of world to reason
- Modern: Rely on data, use machine learning
  - Superficial: Learn patterns of translations
  - Observation: Exploit symmetry

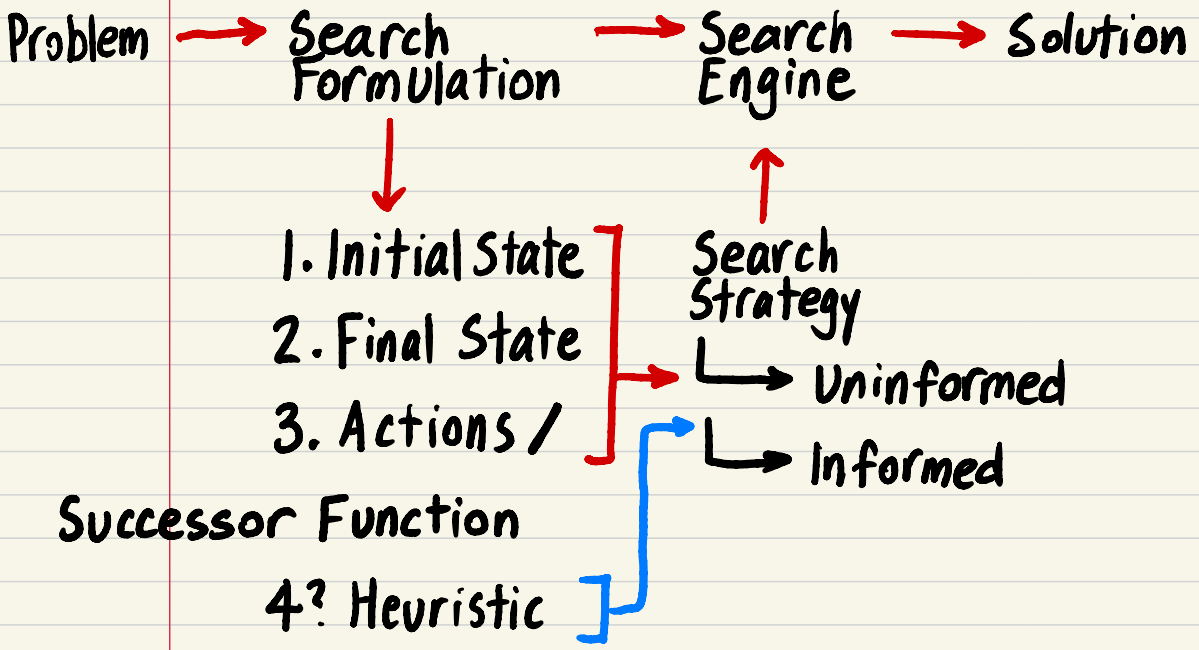
Machine Learning

- Supervised Learning: Label data, tell model to imitate patterns of labelled data
- Unsupervised Learning

Issues with AI

- Fairness: Training AI with biased data
- Explainability: AI cannot justify its actions

1_2 Problem Solving Using Search



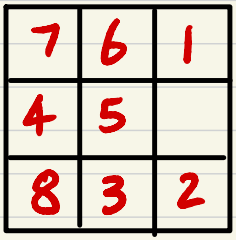
Constraint Satisfaction Problems

- Problems where we only care about the solution
  - Notion of constraints & knowledge
    - Knowledge interpretted as constraints
      - Behavior dictated by data and constraints

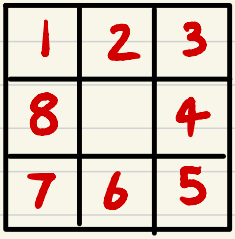
Search Formulation

Ex: 8 Puzzle

- Initial State



- Final State



- Actions
  - 8 tiles can move in 4 directions
  - Blank tile can move in 4 directions

Search Trees

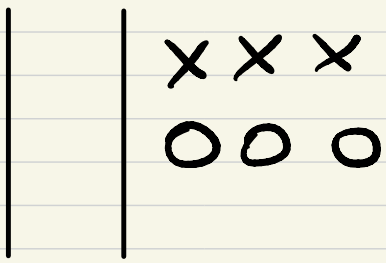
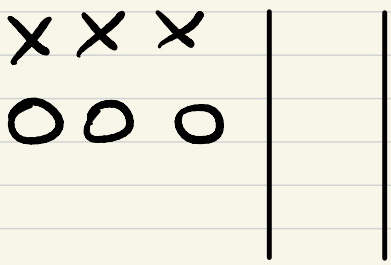
Search trees: Not an actual tree, but rather a conceptualization

- Each action is associated with a cost
- Optimality
  - Optimal Cost
  - Amount of work done by algorithm
    - Efficiency (Time & Space)

Boat Problem

The Game
- Move all X, O to other side of river
- Rules
  - Boat takes at most 2 creatures
    - Boat cannot travel empty
  - Boat cannot have more O's than X's on one side at a time

Search Formulation

- Initial State
  - 
- Final State
  - 
- Actions
  - Boat moves:

    {XX, XO, OO, X, O}
  - Successor Function:

    Returns legal states given an inputted state

What contributes to the difficulty of a search problem?

- Branching Factor: Maximum number of states you can reach from a given state / generated by the successor function
- State Space: Number of possible states
- Depth of optimal solution

Constraint Satisfaction Problems

Ex: 8 queens on 8 x 8 chessboard

- Initial State: Empty Board
- Final State : N/A
- Actions:
  - Test if board is in final state or not

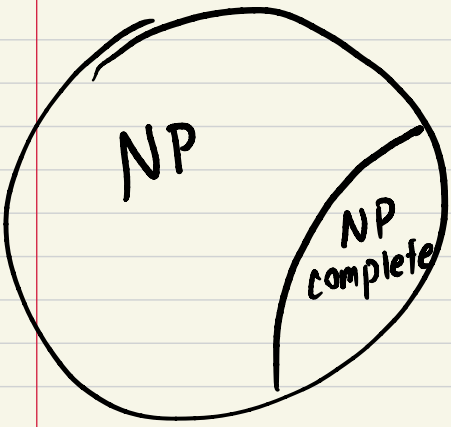
Note:

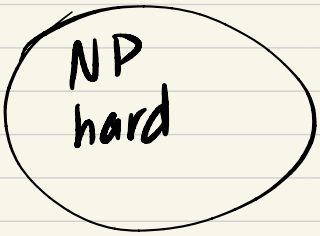
- Search tree depth is finite

SAT

- Given boolean variables, find assignments such that no constraints are violated

↳ Prototypical NP-Complete Problem





2_1 Blind/Uninformed Search Strategies

Terminology:

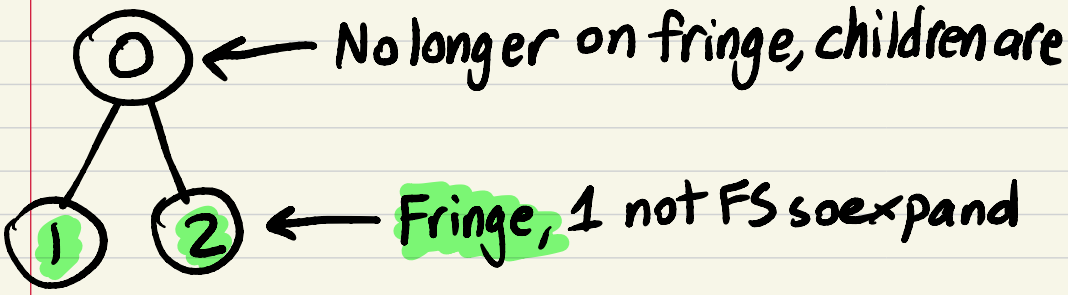
- Frontier / fringe
  - ↳ Nodes that are not explored yet (waiting to be expanded)
- Generate children
  - ↳ Add successor function outputs (children) to fringe
- Expanding a node
  - ↳ 2-Step Process
    - 1. Check if node is final state
    - 2. Generate its children

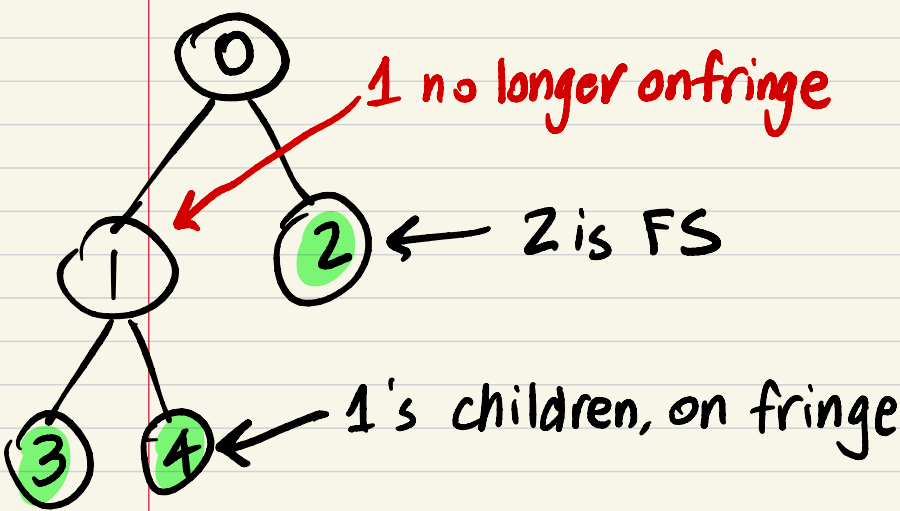
Search strategy: Criteria to decide which node on the fringe to expand next

Note: Check for final state happens during <u>expansion</u>, not generation

Ex: Let 2 be FS


0
Fringe, not FS
so expand


0
No longer on fringe, children are
1
2
Fringe, 1 not FS so expand


0
1 no longer on fringe
1
2
2 is FS
3
4
1's children, on fringe

Search Strategy Criteria

- Completeness
  - ↳ Will the search strategy terminate (If there is no solution)
- Time Complexity
  - ↳ Big O, number of nodes expanded
- Space Complexity
  - ↳ Big O, measured in maximum number of nodes in memory
- Optimality
  - ↳ Does it always find the least cost solution?
    - ↳ DIFFERENT FROM SHORTEST PATH

Parameters to define time/space complexity

- branching factor
  - largest number of children any node has
- maximum depth/height of tree
- depth of least-cost solution

Formula to calculate number of nodes given b & d

$$N(b,d) = O(b^d) = \left(\frac{b}{b-1}\right) b^d$$

Note: Last level of tree has more nodes than rest of tree/dominates

Strategies: BFS

Criterium for selecting nodes off fringe:

↳ Shallowest node first

- Maintain fringe as queue

↳ Generated nodes in end, nodes to expand in front

- BFS will not expand a node at level L unless it has expanded all nodes at earlier levels

↳ Optimal and Complete for Uniform Cost

BFS Overview

↳ Complete

↳ Time Complexity = $O(b^d)$

↳ Space Complexity = $O(b^d)$ (bad)

↳ Optimal

Uniform Cost Search

- Expand node with lowest path cost

UCS Overview

- Complete
- Optimal

] if no zero cost steps, greater than ϵ

- Time
- Space

] $O(b^{\lceil C^*/\epsilon \rceil})$

C^* = Optimal solution cost

ϵ = Smallest possible step

C^*/ϵ = max number of steps

DFS

Criterium for selecting nodes off fringe:

↳ Deepest node first

- Maintain fringe using stack or recursion

DFS Overview

↳ Not complete

↳ Runs infinitely if no solution

↳ Time Complexity = $O(b^m)$

↳ Space Complexity = $O(b \cdot m)$

↳ Not Optimal

Depth Limited Search

- DFS with Depth Cap
  - DFS stops at depth/cutoff L

DLS Overview
- Completeness
  - If (L >= d)
    - Complete
  - else (L < d)
    - Not Complete
- Time Complexity = $O(b^L)$
- Space Complexity = $O(b \cdot L)$
- Optimality
  - If (L == d)
    - Optimal
  - else (L != d)
    - Not Optimal

Iterative Deepening

ID Overview:

- Complete
- Time Complexity = $O(b^d)$
- Space Complexity = $O(b \cdot d)$
- Optimal

Formula to calculate number of nodes given b & d FOR ITERATIVE DEEPEN

$= \left(\frac{b}{b-1}\right)^2 b^d$

- bigger branching factor means less difference between ID and DFS