Chapter 1 – Intro

- Intelligence types:
  - Humanity: Turing Test Approach
    - How long will it take for a human opponent to tell that it's adversary isn't human?
  - Rationality
    - Does it always attempt to choose the option that will attain the best outcome? Or when there is uncertainty, the best expected outcome.

Chapter 2 – Intelligent Agents

- Vocab:
  - Agent – can be viewed as perceiving its environment through sensors and acting upon that environment through actuators
  - Percept – Agent's perceptual input (usually through sensors)
- PEAS:
  - Performance Measurement – How can we determine how well the agent performs its tasks?
  - Environment – What situation is the Agent going to be acting in?
  - Actuators – How will the Agent perform actions within its environment?
  - Sensors – How will the Agent gather information about its environment?
- Properties of task environments:
  - Fully Observable vs Partially Observable – Can the sensors detect all aspects that are relevant to the course of action?(Fully Observable) Can the sensors only detect part of the relevant data or noisy/inaccurate sensors? (Partially Observable)
  - Single Agent vs Multi Agent:
    - Single Agent:
      - An Agent doing a crossword puzzle
    - Multi Agent:
      - An Agent competing in a chess game.
  - Deterministic vs Stochastic – Is the next state of the environment of the environment fully determined by the current state and the action executed?(Deterministic) otherwise it is stochastic.
  - Episodic vs Sequential – Is the agent's experience divided into atomic episodes, where the next episode does not depend on the previous episode? This is episodic. If any decisions relies upon ones previous to it, it is sequential.
  - Static vs Dynamic – Does the environment change while the agent is deliberating? Then it is dynamic, otherwise it is static
  - Discrete vs Continuous – Is there a finite number of distinct states, percepts and actions? Then it is discrete, otherwise it is Continuous.

Chapter 3 – Problem Solving Agents

- Problem Formulation – The process of deciding what actions and states to consider given a goal
- Goal Formulation – Based on the current situation, and the agent's performance measure.
- Fully formulating a problem
  - The Initial State – Where/how the agent begins
  - Actions – What can the agent perform within this problem?
  - Transition Model/Successor – What does each action do? How to we move from one state to another state?

- State Space – The set of all states reachable from the initial state by any set of actions
- Goal Test – Determine if a current state is a goal state. (Based off predetermined goal)
- Path Cost – A numeric cost to each path from one state to another(Value of edges in graph)

- Measuring Algorithms
  - Completeness - Is it guaranteed to find a solution if there is one?
  - Optimality - Does it find an optimal solution?
  - Time complexity - How long does it take to find a solution?
  - Space complexity - How much memory is needed to perform the search?
- Branching Factor – the max number actions each node can take
- Depth – The shallowest goal node in any path of the state space
- Uninformed Search Algorithms:
  - BFS – Breadth First Search pg81
  - UCS – Uniform Cost Search – Djikstra's Algorithm pg83
  - DFS – Depth First Search pg85
  - DLS – Depth Limited Search – DFS with a depth limit pg87
  - IDS – Iterative Deepening(depth first) search – Gradually increase depth limit until goal is found pg88
  - BDS – Bidirectional Search – Run two simultaneous search 1 from initial and one back from goal. pg90
- Heuristics:
  - Additional information that allows an agent to learn for itself
  - Admissible – To be admissible a heuristic must NOT overestimate the true cost to reach the goal
  - Consistency/Monotonicity – The heuristic must be a consistent representation of the information. One node's heuristic can't vastly underestimate while another is close. (Triangle Inequality – Each side of a triangle can't be longer than the sum of the two other sides)
- Informed Search Strategies
  - Greedy Best First Search – Attempts to expand the node that is closes to the goal, only using the heuristic to make decisions. $f(n) = h(n)$ pg92
  - A* Search – $f(n) = g(n) + h(n)$ where $g(n)$ is the cost to get from one state and $h(n)$ is the heuristic to get from the node to the goal and $f(n)$ is the estimated cost of the cheapest solution through n. Identical to Djikstras but uses $g(n) + h(n)$ instead of just $g(n)$ pg93
- Memory Bounded Heuristic Search – We didn't discuss these, but they're in his notes.
  - Iterative Deepening A* (IDA*)
  - Recursive best-first search (RBFS)
  - Memory-bounded A* (MA*)
  - Simplified MA* (SMA*)

Chapter 5 – Adversarial Search

- Minimax algorithm: pg166
  - Effectively a DFS that assigns leaf nodes a value and every parent chooses the minimum or maximum value based off of the children's value. Traditionally the user will be a Max node (denoted with a upright triangle) and the opponent will be a min node (denoted with a downward triangle)
- alpha-beta ($alpha, \beta$) pruning pg167
  - Used in conjunction with the minimax algorithm.
  - On a Max node, you check if the Max value of your children are $\geq \beta$ if so, then return
  - On a Min node, you check the Min value of your children are $\leq \alpha$, if so, then return.
  - WATCH A VIDEO IT CAN EXPLAIN BETTER THAN THIS EVER WILL
- Search vs Lookup – If there is a discrete amount of moves that can be held within memory, it is computationally less expensive to just do a table lookup for the best next move rather than computing it.
- Drawing Game Trees – Be able to draw a game tree (at least 2 ply). Specifically look at tic-tac-toe accounting for symmetry.

Chapter 6 – Constraint Satisfaction Problems

- Constraint Satisfaction Problems – A problem solved when each variable has a value that has satisfied all constraints on the variable
- Formulating a CSP:
  - Variables
  - Domain for the Variables – The possible values for the variables
  - Constraints – The rules for validation on the variables
  - Solutions are Valid variable assignments
- Binary Constraints – Relates two variables (e.g. $SA \neq NSW$). Can be represented as a constraint graph. All ternary (and n-ary) constraints can and should be reduced to binary. Unary constraints can and should reduce the domain of a variable, by doing so, removing the Unary constraint.
- Arc Consistency (concept)– Every value in its domain satisfies the binary constraints. Can be performed to reduce the domain and the search space within the CSP. pg209
- Strategies to Solving CSPs
  - Incremental – Start with no variables assigned. Assign variables one by one ensuring that the variable satisfies all constraints
  - Constraint Propagation – Use constraints to reduce the legal values for a variable, reducing the domain of the variable
  - Arc Consistency(algorithm) – AC3 – If a node loses a state in the domain, check it's neighbors to ensure that it is still solvable, reducing the potential domains of the neighbors. pg 209
  - Backtracking Search – An incremental approach and backtracks when no legal moves are available. pg 215
- Choosing Heuristics for CSPs
  - Minimum Remaining Values – Choose a variable that has the fewest options in the domain. Attempts to get a dead end as soon as possible (if a dead end exists)
  - Degree – Choose the variable that has the most children. This option will reduce the domain of the most neighbors
  - Least Constraining Value – Choose the value for the variable that will least reduce the domain of its neighbors
  - Forward Checking/Inference – When a variable is assigned, it checks arc consistency reducing the domain of the neighbors (see AC3)
  - Min Conflicts – Choose a value that violates the fewest constraints