A* ALGORITHM

Search algo used to find shortest path from a start node to a goal node in a weighted graph
- Combines features of DIJKSTRA'S & Greedy Best-First search
- Applications
  - Robotics
  - Navigation system
  - Game development
  - Geographic Information System (GIS)

Components	Heuristic Function	Cost Function
· Start Node · Goal Node · Open List · Closed List	Estimates the cost from the current node to the goal	Actual cost from start node to the current node-weight

Evaluation Function: $f(n) = g(n) + h(n)$

- Initialization: Add the start node to the open list with $f(start) = h(start)$

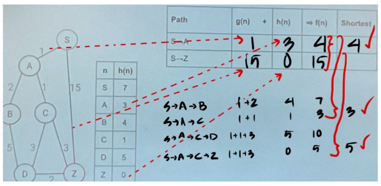
Main Loop: While the open list is not empty
- SELECT NODE: Choose the node with the lowest f value from the open list
- GOAL CHECK: If it's the goal node, reconstruct the path
- GENERATE SUCCESSOR: Evaluate each neighbor node
- UPDATE LISTS: Move the current node to the closed list and update the open list with new nodes

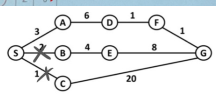
PATH RECONSTRUCTION: Trace back from the goal node to the start node to build the path

Time Complexity: $O(b^d)$
- b: branching factor
- d: depth of the shortest path

Space Complexity: $O(b^d)$
needs to store nodes in the open & closed lists

Better heuristics lead to more efficient search and reduced time complexity





Use this heuristic function:

n	$h(n)$
A	7
B	10
C	15
D	1
E	8
F	1
G	0
S	10

Path	g(n) +	h(n)	⇒ f(n)	Shortest
S	0	10	10	
S→A	3	7	10	10
S→B	2	10	12	×
S→C	1	15	16	×
S→A→D	3+6	1	10	10
S→A→D→F	3+6+1	1	11	11
S→A→D→F→G	3+6+1+1	0	11	11

HEURISTIC SEARCH

Heuristic Search Algorithms
- PURPOSE: Used to find optimal paths or solutions in various problems such as pathfinding & puzzle solving
- EXAMPLE: A* Algo is a popular heuristic search algo for finding the shortest path in graphs
- Key Concepts
  - Admissibility: Ensures the heuristic never overestimates the cost to reach the goal
  - Consistency: Ensures the heuristic satisfies the triangle inequality

- Consistent Heuristic: Heuristic Function h(n) is consistent or monotonic if for every node n and every successor n' of n reached
- Admissable Heuristic: h(n) is admissable if it never overestimates

guarantees optimality
- Example: In pathfinding problems, straight line distance (Euclidean distance) from a node to the goal is often admissable

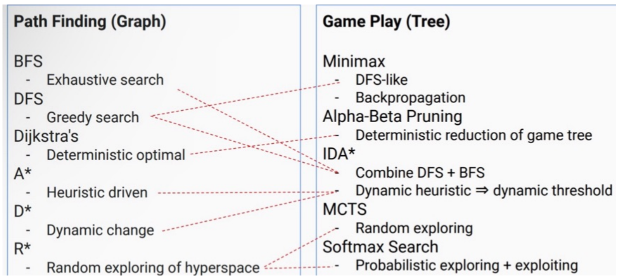
- Consistent heuristics ensure A* will expand nodes in proper order, guaranteeing completeness & optimality

MONTE CARLO — RANDOMIZED APPROACH

Monte Carlo Tree Search (MCTS)
- Heuristic search algo that uses randomness to explore the game tree
- It builds the tree incrementally, focusing on promising moves
- BASIC STEPS
  - Selection: Start at the root node (current game state) & recursively select child nodes using a strategy like Upper Confidence Boun[d] until you reach a leaf node
  - Expansion: If leaf node is not a terminal state, expand the tree by adding one or more child nodes, correspondin[g] to possible moves
  - Simulation: Perform a random playout from the newly expanded node until the game reaches a terminal state. The result of this random playout (W/L) provides an estimate of the value of the node
- Backpropagation:

Define the problem
- Start by clearly identifying the problem or quantity you want to estimate. This could be numerical result, such as the value of an integral, the outcome of a probabilistic process, or the approxamation of a constant

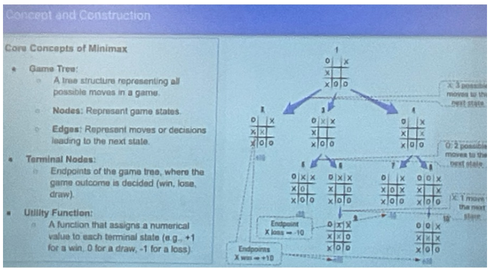
Set up a random model
- Translate problem into form where random variables or random sampling



MINIMAX

DECISION MAKING:

Minimax Algorithm: Used in two-player zero-sum games where one player's gain is equivalent to the other player's loss
↳ Recursive decision-making algo used for minimizing the possible loss in a worst-case scenario

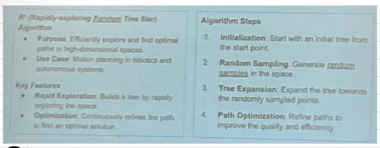


KEY FEATURES:
- COMPLETE: If a game tree is finite, Minimax will eventually find the optimal
- DETERMINISTIC: Always produces the same result given the same game state
- OPTIMAL: Find the best move assuming both play optimally

Time Complexity: $O(b^d)$, b = branching factor & d = tree depth
Space Complexity: $O(bd)$, depending on depth & breadth of game tree

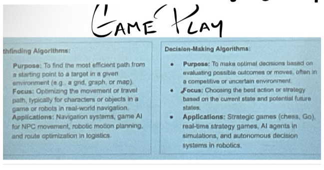
R* ALGORITHM

R* = Rapidly-exploring Random Tree Star



R* Heuristic
- R* may incorporate heuristic biases to guide the sampling process towards promising regions of the space
  - Goal biasing: increasing the probability of sampling near the goal to improve convergence towards the target
- Obstacle Avoidance: Adjusting sampling strategies to avoid areas with high obstacle density
- Informed Sampling: R* can use informed sampling technique to focus exploration efforts on regions with higher likelihood of leading to an optimal path
- Density Based Sampling: Sampling more frequently in regions with lower or higher density

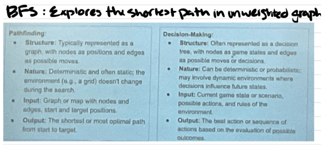
GAME PLAY



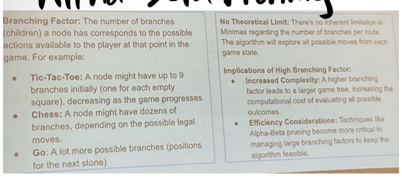
A*: Combines Dijkstra's algorithm with heuristics for efficient pathfinding

Dijkstra's: Finds the shortest path in graph w/o heuristics

BFS: Explores the shortest path in unweighted graph



Alpha-Beta Pruning



The sequence in which leaf nodes are evaluated during the Minimax algorithm can significantly impact

Alpha Cutoff (Maximizer)
- Alpha is the best score the maximizer can guarantee at that point in the tree

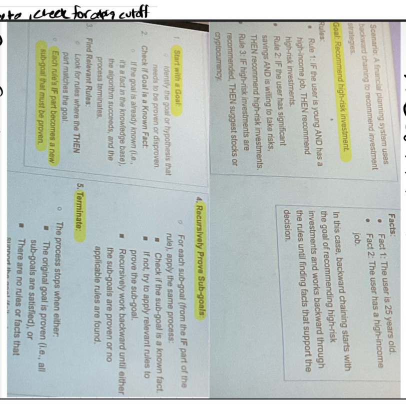
Beta Cutoff
- Beta is the best score the minimizer can guarantee at that point in the tree

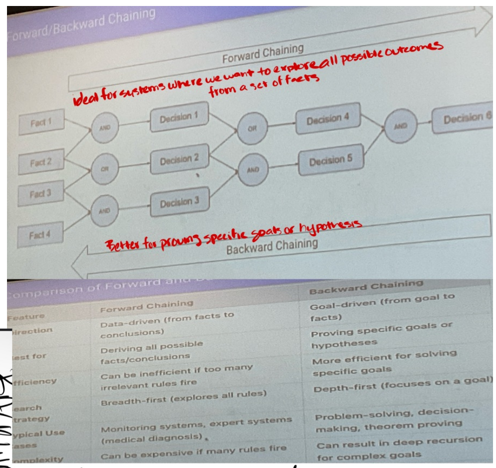
Initialize Alpha & Beta → Alpha = $-\infty$ & Beta = ∞

Maximizer's Turn (MAX NODE)
↳ For each child of the current node,
Evaluate the child node, update alpha, check for beta cutoff

Minimizer's Turn (MIN NODE)
↳ For each child of the current node,
Evaluate the child node, update beta, check for alpha cutoff

Depth First Search
↳ focuses on a specific goal and it's subgoals, going deeper in reasoning tree
↳ Efficient for goal-oriented programs
↳ If goal is cyclically dependent on itself, infinite loops can occur





BACKWARD CHAINING

Forward Chaining: Data-Driven (starts with known facts & use rules)
- Initialize Knowledge Base
  - Start with set of facts
  - Define a set of if-then rules
- Match Rules
  - For each rule, check if conditions are satisfied by current facts in base
    - This is called Rule Matching
- Apply Rules
- Update Knowledge Base
- Repeat
- Terminate

Problem Solving Algorithms, Expert Systems Solutions, Knowledge Representation Reasoning
Problem Solving Algorithms
Knowledge/Heuristics -> numbers
use numbers to drive the problem solving strategy
Expert Systems Solutions
Knowledge -> Logic-> Data Structure
Run data through logic -> Flow facts through data structure, to make decisions
Knowledge Representation Reasoning
Knowledge/Relationships -> Taxonomy & Ontology
Web Data -> Semantically Linked Data
Querying -> Reasoning

Learning-Based AI vs Classic AI
Learning-Based AI (knowledge driven rules)
Rules and data are put into classical programming, which outputs answers
Rules =?= Knowledge -> Intelligence
Classic AI (data driven data)
Data and answers put into machine learning and output “rules” - Training phase
Data is put into Inferencing (“rules”) and outputs Answers - Inference phase

ML Problems and Applications
House Price Prediction
Data - Features like square footage, number of bedrooms, neighborhood, etc.
Label - Price of the house
Model - Linear Regression
Answer - Prediction of the house price (Regression)
Customer Churn Prediction
Data - Features like customer age, contract length, monthly charges, customer service interactions, etc.
Label - Whether the customer will leave the service
Model - Logistic Regression or Decision Trees
Answer - Prediction of whether customer will churn (Classification)

<u>Data vs Label for Learning</u>
Dog vs Cat
Data: Image features extracted from pixels or convolution layers (images of dogs & cats)
Label: The class (1 for cat, 0 for dog)

Handwritten Digit Recognition (MNIST dataset)
Data: Images of handwritten digits (28x28 pixel images)
Label: The actual digit (0-9) corresponding to the image
Model: Convolutional Neural Network (CNN) or k-Nearest Neighbors (k-NN)
Answer: Classification on digit (Classification)

Images of various objects
Data: Small images of various objects like airplanes, cars, bird, cat, deer, dog, etc.
Label: The actual name of the object corresponding to the image
Model: CNN
Answer: Classification of object

Object Recognition (labeling is very important)
Face Detection vs Face Recognition
Automatic License Plate Recognition

Content Derivation
Colorization
Supervised Learning
Type of machine learning where a model is trained on labeled data
Model is from human knowledge
Parameters are from training (examples)
Key Algos
Linear Regression - Regression problems
predicting house price based on its size and location
Logistic Regression - Binary classification problems
Predicting whether a customer will buy a product (y/n , 1/0)
Decision Trees - Classification and Regression
Predicting which loan applicants are likely to default
Support Vector Machines (SVM) - Classification
Classifying emails as spam or not spam
Linear Regression
Regression - return to a former or less developed state
A measure of the relation between the mean value of one variable (e.g. output) and corresponding values of other variables (e.g. time and cost)
In context of stats and ml
Regression refers to a type of predictive modeling technique that estimates the relationships between a dependent variable (outcome/target) and one or more independent variables (predictors or features)
The primary goal of regression is to predict the value of the dependent variable based on the input feature
Regression tries to “fit” a model to observed data in such a way that we can make predictions about future data points
Goal of linear regression is to find the model parameters that best fit the data

Logistic Regression
Classification algorithm used to predict binary outcomes based on 1 or > independent variables
Formula : P(y=1/x)= 1 / (1+e^-(B0+B1x1+...+Bnxn)
Output: probability value between 0 and 1, which is converted into a class label
Example: Predicting whether a student will pass/fail based on study hours
Sigmoid Function = Logistic Function
Primary reason for using the sigmoid function is that it maps any real-valued number into a range between 0 and 1
Crucial for logistic regression because we want to interpret output as prob
ô(z)=1/(1+e-z), where z = (B0+B1X)
As z approaches -∞, sigmoid approaches 0
as z approaches +∞, sigmoid approaches 1
This property makes it perfect for estimating the prob of binary outcome
Function has an S-shaped curve, providing a smooth transition between 0 and 1
Allows model to represent probabilities in a way that reflects uncertainty
Near midpoint, the output changes rapidly, making it sensitive to changes in input
Purpose: Used for predicting a binary outcome based on one or more independent vars
Assumptions
Dependent Variable is binary (0,1)
Independent Variables can be continuous or categorical
Assumes a logistic relationship between independent and the log odds of the dependent variable
Data Role:
Similar to linear regression, but dependent variable is categorical
Model estimates probability that given input point belongs to a particular category based on the independent variables
Example: medical diagnosis scenario; independent variables = age, symptoms, test results to predict if patient has disease or not (1/0)

Importance of Data
Quality - quality of data directly impacts the accuracy of the model
Feature Selection - choosing wrong independent variables can distort the predictions
Volume - Sufficient volume helps in capturing the underlying patterns & relationships

Decision Tree
Construction
Splitting - dataset split based on attribute that leads to highest information gain
Recursive Partitioning - process continues recursively, splitting each subset furth
Stopping Criteria - Algo stops when all data points are perfectly classified or other predefined stopping rules are met (max depth, minimum samples per leaf)
Prediction - Once built, tree can predict output by navigating from root to leaf node for new input data
Information Gain
Entropy
Gini Index - Measure of impurity or disorder used in decision trees
Represents how often a randomly chosen element from the set would be incorrectly classified if it was randomly labeled according to the distribution of labels in the set
Definition
Gini(s) = 1 - sum of i = 1 to n of
Weighted Gini
Gini(Split) = sum of 1 to k ((|Si|/|S|) x Gini(si))
S = original set before split
Si = one of the k groups formed by the split
|Si| = size (number of elements) in group Si

Rete Algorithm
Ensuring efficient pattern matching even when dealing with a large set of ruels and facts
Decouple rule evaluation from execution sequencing
Alpha-Beta Network/Node
Alpha Node - These handle simple tests on individual facts
Comparing a fact’s attribute to a constant value
Beta Node - These combine multiple facts
When a rule has conditions that depend on more than one fact
Rete avoids re-evaluating all rules and facts each time new data is introduced
Using a data structure (called the Rete Network)
Convert Rules - > Objects
Advantages
Speed as it takes the advantage of structural similarity
Disadvantages
Conflict Resolution
<u>Definition</u>: When multiple rules are applicable at the same time, conflict resolution strategies determine which rule to apply
With forward chaining, the Rete network would create partial matches for each of these rules based on incoming patient data. As new symptoms are added, the network efficiently updates natchez. When multiple rules are ready to fire, a conflict resolution strategy determines which diagnosis to make first, based on priority or specificity
This combination of the rete algorithm and conflict resolution strategies allows systems to efficiently handle large numbers of rules and facts
Strategies
Recency, specificity, priority, random
Recency- Rule that matches the most recent data is given priority
Specificity - More specific rules are given priority
Priority Rules - predefined priorities assigned by the system designer
Random - If no clear priority exists
Algos
Pqueue, salience, heuristic evaluation
Expert Systems Examples
Production Rules (Rule-Based Systems)
Semantics
Frames
Taxonomy is a group of related thing
Define categories within a single domain
Ontology captures multidimensional relationships
Connects taxonomies to provide rich information about the business environment
Interrelationships among the entities in the taxonomy are the essence of ontology
Therefore, the ontology reflects a broad view of the business and a detailed description of its components

Semantic web stack
Ontology management tools
Ontology based data access
Speciailized ontologies and models
Linked data platforms
Owl reasoners
Querying , ontologies, rules, taxonomies,
Data interchange
Syntax
Identigiers, character Set

RDF - Resource Description Framework - Standard for representing data on the web
Flexible and powerful framework for representing structured data
Enable linking od diverse datasets, making it essential for the semantic web
General taxonomy framework for representing interconnected data on the web
RDF statements are used for describing and exchanging metadata
Which enables standardized exchange of data based on relationships
Developed by W3C as a key component of the Semantic Web
Purpose
Represent Information in a structured way so machines can understand and process relationships between entities
Enables the creation of linked data,
Basics
RDF Data Model is built around triples
Example: (John, hasName, “John Smith”)
Subject, Predicate, Object
RDF uses XML syntax for structuring triples
RDFS (S=Schema)
Extends RDF by providing semantics for concepts like classes and properties

OWL (Web Ontology Language)
Based on rdf but more expressive
Formal language designed for representing complex knowledge about things relationships and categories on the web
Purpose
Enables machines to understand and reason over structured data
Standardized by the W3C for the Semantic Web
RDF doesn’t have the power to reason, just to capture and structure
Key Concepts and Levels
Classes, individuals, properties, and axioms
Class - Categories or types of things (person, organization)
Subclassing (person->employee->John)
Individual - Specific instance of classes (John, ABC_Corp)
Properties - Relationship between things (worksFor)
Object Properties
Reasoning - process of inferring new knowledge from explicit facts and rules
If John is an instance of manager, oWL inferencing can conclude that John is also an employee and person
Applications
Healthcare & biomedical ontologies, Gene ontology, Google’s knowledge graph

RDF vs OWL
Rdf = smaller domain, mostly on taxonomies (classifying)
Structure: Simple Triples (subject-predicate-objects)
Better for lightweight flexible knowledge graphs where simplicity is key, and there's no deep reasoning or complex logic structures
Owl = Bigger domain, Bigger scope
Structure: triples with extended semantics (classes, properties, restrictions)
Suited for applications where you need to enforce strict ontological rules and enable reasoning engines to draw conclusions automatically from defined knowledge graph

1. Artificial Intelligence (AI) - Classical Approaches

- **Brief History of AI**
  - **Alan Turing (1950):** Introduced the concept of the "universal machine" and the Turing Test.
  - **John McCarthy (1956):** Coined the term "Artificial Intelligence" at the Dartmouth Conference.
  - **Early Programs (1951):** Christopher Strachey wrote the first AI program, a checkers game.
- **Ups and Downs of AI**
  - **First AI Winter (1970s):** Reduced funding and interest due to unmet expectations.
  - **Second AI Winter (Late 1980s - Early 1990s):** Expert systems struggled with scalability.
  - **Resurgence in the 1980s:** Development of expert systems like MYCIN and XCON revived AI interest.

2. Search-Based Problem Solving

- **Pathfinding Algorithms**
  - **Breadth-First Search (BFS):** Explores nodes at the current depth before moving deeper. Ideal for unweighted graphs.
  - **Depth-First Search (DFS):** Explores as far as possible along a branch before backtracking.
  - **Dijkstra's Algorithm:** Uses a priority queue to explore the cheapest path first. Efficient for finding the shortest path in weighted graphs.
  - *A Algorithm:** Combines Dijkstra's algorithm with a heuristic. Finds shortest paths efficiently with heuristic guidance.
  - *D Algorithm:** Finds paths in dynamic environments and updates efficiently when obstacles change.
  - *R Algorithm:** Used in high-dimensional motion planning, explores random points in space and refines the optimal path.

3. Game Playing

- **Minimax Algorithm:**
  - Used in two-player games like chess or tic-tac-toe to find the optimal move assuming the opponent plays optimally.
- **Alpha-Beta Pruning:**
  - Optimizes Minimax by pruning branches that don't affect the final decision.
- **Monte Carlo Tree Search (MCTS):**
  - Uses random simulations to explore possible game states, focusing on the most promising moves.
- **Softmax Search:**
  - Balances exploration and exploitation by probabilistically selecting moves, allowing exploration of less-promising moves.

4. Knowledge Representation and Reasoning

- **Forward Chaining:**
  - A data-driven approach that applies rules to known facts to derive new facts.
- **Backward Chaining:**
  - A goal-driven approach that starts with a goal and works backward to find supporting facts.
- **Rete Algorithm:**
  - Efficient pattern matching algorithm used in large-scale rule-based systems to match facts with rules.

5. Machine Learning (ML)

- **Supervised Learning**
  - **Decision Trees:** Splits data based on features that lead to the highest information gain.
  - **Random Forests:** An ensemble of decision trees that reduces overfitting and improves accuracy.
  - **Gradient Boosting:** Sequentially builds models to correct errors of previous models. XGBoost is a popular, efficient implementation.
- **Neural Networks**
  - **Perceptron:** A simple linear classifier.
  - **Multilayer Perceptron (MLP):** A network with multiple hidden layers, trained using backpropagation and gradient descent.

6. Heuristic Search

- **Admissibility:**
  - Ensures that a heuristic never overestimates the true cost to reach the goal.
- **Consistency:**
  - Ensures the heuristic obeys the triangle inequality, preserving optimality.
- **Backward Heuristic:**
  - Used in dynamic environments to propagate changes from the goal backward, efficiently updating paths.

7. Knowledge Representation

- **Ontologies:**
  - Structures that define relationships between concepts and connect taxonomies for reasoning over linked data.
- **Taxonomy vs. Ontology:**
  - **Taxonomy:** Categorizes items within a domain.
  - **Ontology:** Defines relationships between taxonomies and connects domains for richer data representation.

8. Learning-Based AI vs. Classical AI

- **Classic AI (Expert Systems):**
  - Rule-based systems that use predefined logic to infer conclusions from data.
- **Learning-Based AI (Machine Learning):**
  - Data-driven models that learn patterns from data without pre-programmed rules. Includes neural networks, decision trees, and ensemble learning.

9. Comparing Algorithms

- **Pathfinding Algorithms:**
  - **Strengths:** Effective in static environments, providing optimal paths.
  - **Limitations:** Struggle in dynamic environments where conditions change.
- **Decision-Making Algorithms:**
  - **Strengths:** Handle dynamic and competitive environments well, suitable for complex games.
  - **Limitations:** Computationally expensive and may require approximations in large search spaces.