**Simplified AI Lecture 4 Summary: Searching Algorithm in AI**

**What is a Searching Algorithm?**

* Searching algorithms help AI find solutions to problems
* They help the AI look through possible paths to find the best one
* There are different types of search algorithms for different situations

**Types of Search Algorithms in AI**

**Uninformed Search (Blind Algorithm)**

* Does not have any background information about the problem
* Number of steps and path cost are unknown
* The agent only knows when it reaches a goal
* Examples: BFS, DFS, UCS, Bidirectional Search

**Informed Search (Heuristic Search)**

* Uses background information about the problem
* Agent has estimates about how close it is to the goal
* More efficient than uninformed search
* Examples: Greedy Best-First, A\* Search

**Uninformed Search Strategies**

**Breadth-First Search (BFS)**

* Explores all nodes at the current depth before moving to the next level
* Uses a FIFO queue (First In, First Out)
* Guarantees a solution if one exists (complete algorithm)
* Good for finding the shortest path in an unweighted graph

**Uniform-Cost Search (UCS)**

* Prioritizes the lowest-cost path when multiple paths exist
* Visits nodes in order of increasing path cost
* Ensures optimality if there are no negative costs
* Ensures completeness if states are finite with no zero-weight loops
* Acts like BFS when all transitions have equal costs
* Uses path cost g(n) = sum of edge costs to reach the current node

**Depth-First Search (DFS)**

* Explores branch nodes deeply before backtracking
* Goes as far as possible along one branch before going back
* Uses a LIFO stack (Last In, First Out)
* Good for maze-like problems where the goal might be deep
* Not guaranteed to find the shortest path

**Bidirectional Search**

* Searches from both the start and goal at the same time
* The two searches meet at a common node in the middle
* Reduces the search space significantly
* Works well when there's a single goal state

**Informed Search Strategies**

**Greedy Best-First Search**

* Always expands the node closest to the goal according to a heuristic
* Uses a heuristic function h(n) to estimate the cost to the goal
* Evaluation function f(n) = h(n)
* Uses a priority queue for implementation
* Very fast but not guaranteed to find the optimal solution
* Can get stuck in loops or dead ends

**A\* Search**

* Combines the advantages of UCS and Greedy Best-First Search
* Uses evaluation function f(n) = g(n) + h(n)
  + g(n) is the path cost from the start to the current node
  + h(n) is the estimated cost from the current node to the goal
* UCS keeps the solution cost low
* Greedy Best-First helps find a solution quickly
* A\* combines these approaches for an efficient search

**Search Strategy Comparison**

* BFS: Explores level by level, finds shortest path in unweighted graphs
* DFS: Explores deeply before backtracking, good for mazes
* UCS: Finds shortest path in weighted graphs, considers path costs
* Greedy Best-First: Very fast, but may not find optimal solution
* A\*: Efficient and optimal when using an admissible heuristic

**Important Concepts**

* Complete algorithm: Guarantees finding a solution if one exists
* Optimal solution: The solution with the lowest cost
* Heuristic function: An estimate of the cost to reach the goal
* Path cost: Sum of the edge costs to reach a particular node