جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قسم الانظمة الطبية الذكية

Lecture: (6)

A and A* Search Algorithms

Subject: Artificial Intelligence

Class: Third

Lecturer: Dr. Maytham N. Meqdad

A (Best-First) Search Algorithm

- **Description**: The A (Best-First) search algorithm is a greedy search that attempts to expand the most promising node based on a heuristic function, **h(n)h(n)h(n)**, which estimates the cost from the current node to the goal.
- **How It Works**:
  - It evaluates nodes based only on the heuristic function, h(n)h(n)h(n).
  - Nodes with lower heuristic values are prioritized.
  - Because it doesn't consider the actual path cost to reach the node, it may find a solution but not necessarily the shortest one.
- **Complexity**: The performance depends on the heuristic used, but it is generally **O(bd)O(b^d)O(bd)**, where bbb is the branching factor and ddd is the depth of the solution.
- **Pros**:
  - Simple and fast with a good heuristic.
  - Useful for problems where finding any solution quickly is the main goal rather than the optimal path.
- **Cons**:
  - May not find the shortest path.
  - The solution's quality is highly dependent on the heuristic.

A* Search Algorithm

- **Description**: A* search is an improvement on A (Best-First) by combining both path cost and heuristic information. It evaluates each node based on a function, f(n)f(n)f(n), defined as:

  f(n)=g(n)+h(n)f(n) = g(n) + h(n)f(n)=g(n)+h(n)

  where:

  - g(n)g(n)g(n) is the actual cost to reach the current node nnn from the start node.
  - h(n)h(n)h(n) is the heuristic cost estimate from node nnn to the goal node.
- **How It Works**:
  - The algorithm begins at a start node and explores paths, prioritizing nodes with the lowest f(n)f(n)f(n) value.
  - It keeps track of the path cost using g(n)g(n)g(n) and the estimated cost-to-goal using h(n)h(n)h(n).
  - The algorithm expands the node with the lowest f(n)f(n)f(n) value until it reaches the goal node.

- **Optimality**: A* is optimal and complete if the heuristic h(n)h(n)h(n) is **admissible** (never overestimates the true cost to reach the goal) and **consistent** (satisfies the triangle inequality).
- **Complexity**: Its complexity depends on the heuristic function and is generally **O(bd)O(b^d)O(bd)**, where bbb is the branching factor and ddd is the depth of the optimal solution. However, it performs more efficiently than uninformed search algorithms with a good heuristic.
- **Pros**:
  - Finds the optimal solution if the heuristic is admissible.
  - Balances between path cost and heuristic, which makes it efficient.
  - Widely used in applications where the shortest path is required, such as map navigation, robotics, and games.
- **Cons**:
  - Memory-intensive as it stores all nodes in memory.
  - May be slower for complex heuristics or large graphs.

Key Differences Between A and A* Search

Feature	A (Best-First)	A*
Evaluation Function	h(n)h(n)h(n)	f(n)=g(n)+h(n)f(n) = g(n) + h(n)f(n)=g(n)+h(n)
Optimality	Not guaranteed	Guaranteed with an admissible heuristic
Completeness	Complete only with finite graphs	Complete if h(n)h(n)h(n) is admissible
Efficiency	Greedy but may not find shortest	Finds shortest path with an admissible heuristic

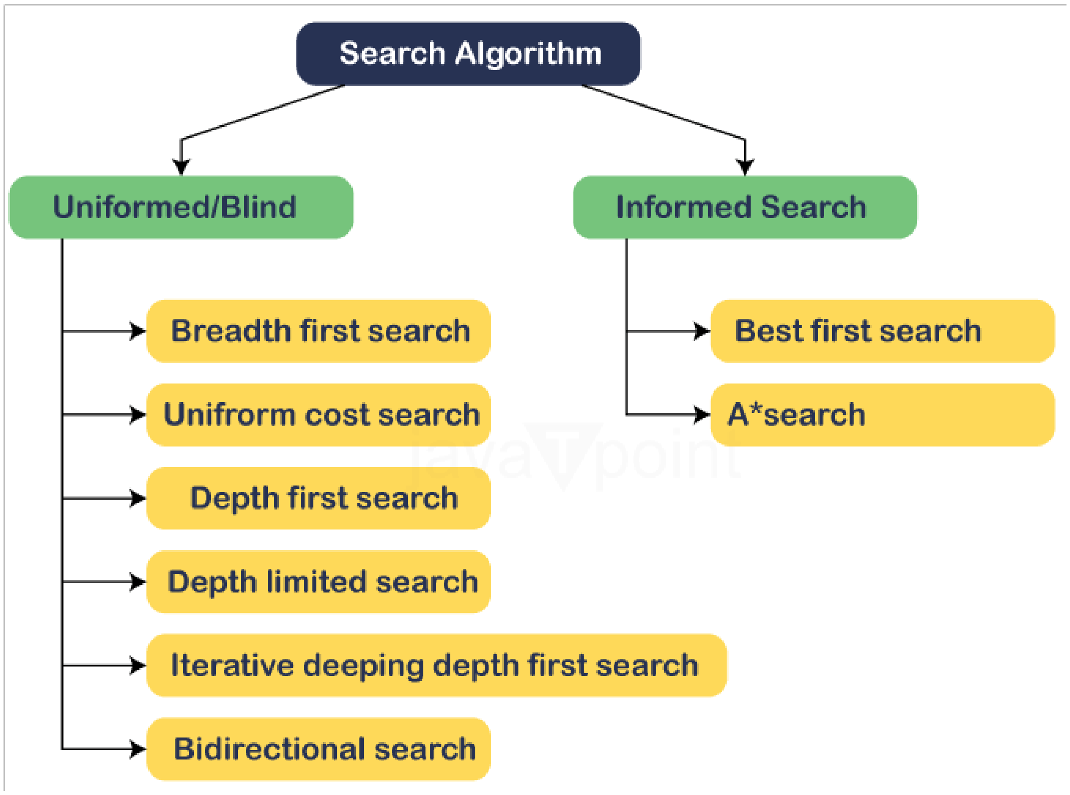
Example Use Case

In a grid-based pathfinding problem (e.g., navigating through a maze), **A*** would choose paths that minimize both the actual distance traveled and the estimated distance to the goal. In contrast, **A (Best-First)** would simply choose the path that seems to be closest to the goal, potentially leading to dead ends or suboptimal paths.

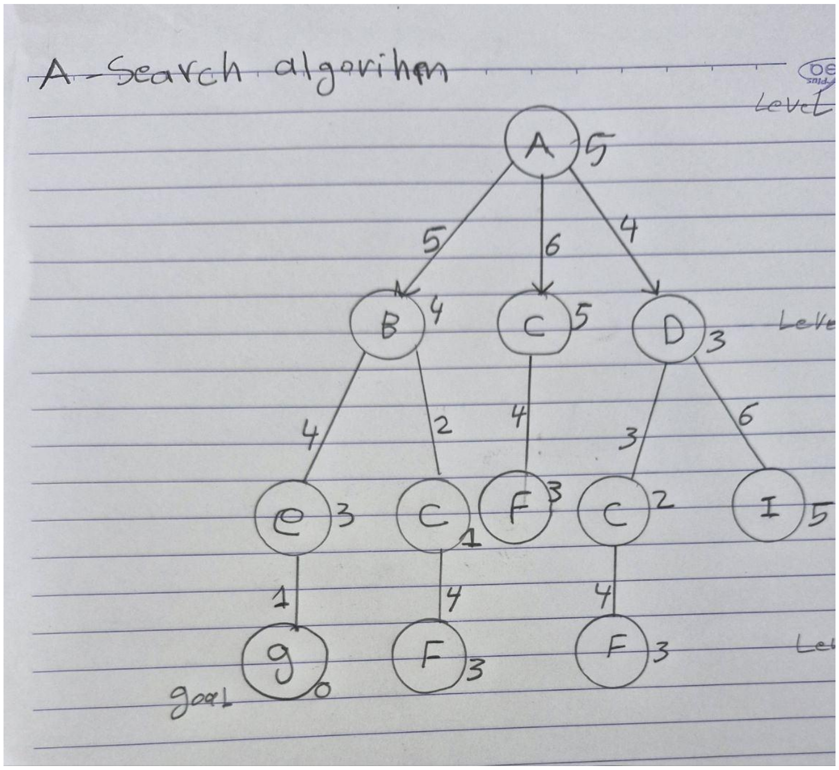
Applications of A*

- **Game Development**: For non-player character (NPC) pathfinding.
- **Navigation Systems**: Finding the shortest route between locations.
- **Robotics**: For obstacle avoidance and path planning in a dynamic environment.

A* is widely preferred due to its ability to provide optimal paths efficiently with the right heuristics, making it a powerful choice in various path finding scenarios.



A Search Algorithm



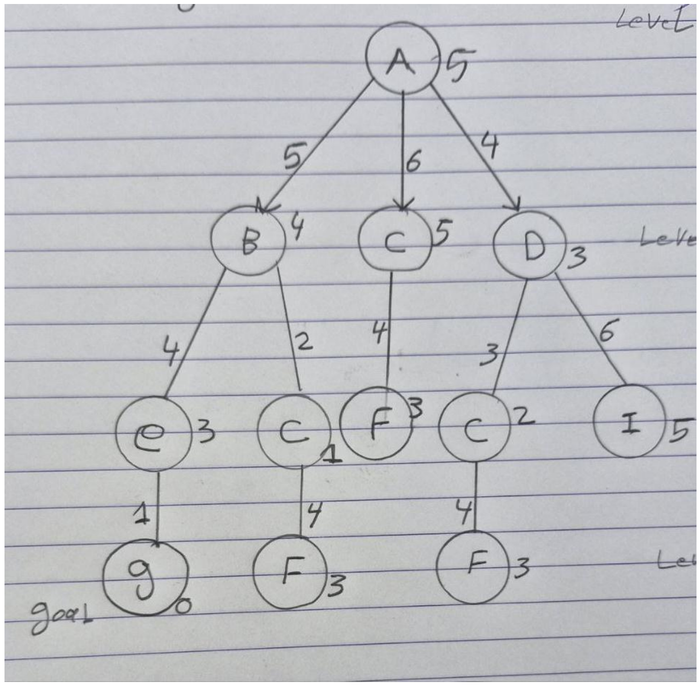
Open	Closed
[A5]	[]
[D4,B5,C6]	[A5]
[C4,B4,I7]	[A5,D4]
[B5,F6,I7]	[A5,D4,C4]
[C3,E5,F6,I7]	[A5,D4,C4,B3]
[E5,F6,I3]	[A5,D4,B5,C3]
[G3,F6,I7] STOP	[A5,D4,B5,C3,E5]
	[A5,D4,B5,C3,E5,G3]

G3 → GOAL

A0 → D4 → B9 → C2 → E6 → G1

=0+4+9+2+6+1=22

A* Search Algorithm



Open	Closed
[A8]	[]
[B6,D7,C9]	[A8]
[E7,C4,D7]	[A8,B6]
[G0,C4,D7] STOP	[A8,B6,E4]
	[A6,B6,E4,G0]

G0→ GOAL
PATH: A0→B5→E4→G1
=0+5+4+1=10