ARTIFICIAL INTELLIGENCE
&
MACHINE LEARNING

SESSION NO : 7

Overview

- Uninformed vs. Informed Search
- Greedy Best-First Search
- A* Search Algorithm
- Heuristic Functions
- Admissibility & Consistency
- Comparison and Use Cases

Uninformed vs. Informed Search

Uninformed Search (Blind Search):

- Does **not use any domain-specific knowledge**.
- Only uses the information available in the problem definition (e.g., initial state, goal test).
- Examples: **Breadth-First Search (BFS)**, **Depth-First Search (DFS)**, **Uniform Cost Search (UCS)**.
- Often explores many irrelevant paths → **inefficient**.

Informed Search (Heuristic Search):

- Utilizes **problem-specific knowledge** (heuristics) to guide the search.
- More **efficient** by focusing on promising paths.
- Examples: **Greedy Best-First Search**, **A*** Search.
- Requires a good **heuristic function** h(n) for effectiveness.

Comparison Table

Feature	Uninformed Search	Informed Search
Knowledge Used	None (Blind)	Heuristic/Extra knowledge
Speed & Efficiency	Slower	Faster (with good heuristics)
Examples	BFS, DFS, UCS	Greedy, A*
Goal Direction	No	Yes

Classical Search – Informed

- **Classical Search** algorithms operate without any domain-specific knowledge. They include uninformed strategies such as Breadth-First Search, Depth-First Search, and Uniform Cost Search. These methods explore the search space blindly and can be inefficient for large or complex problems.
- **Informed Search** (also known as heuristic search) uses problem-specific knowledge to find solutions more efficiently. By incorporating **heuristics**—estimates of the cost from a given state to the goal—these algorithms significantly reduce the number of explored nodes.

Two important informed search strategies are:

1. **Greedy Best-First Search**, which uses heuristics to expand the node that appears closest to the goal.
2. **A*** Search, which combines the cost so far and the heuristic estimate to make balanced decisions.

Greedy Best-First Search

- Greedy Best-First Search works by evaluating the cost of each possible path and then expanding the path with the lowest cost.
- This process is repeated until the goal is reached.
- The algorithm uses a heuristic function to determine which path is the most promising.
- The heuristic function takes into account the cost of the current path and the estimated cost of the remaining paths.
- If the cost of the current path is lower than the estimated cost of the remaining paths, then the current path is chosen. This process is repeated until the goal is reached.

Steps of Greedy Best-First Search Algorithm

Here is a simplified step-by-step breakdown of the Greedy Best-First Search algorithm:

1. **Initialize:** Start from an initial node (often the root or starting point). Add this node to a priority queue.

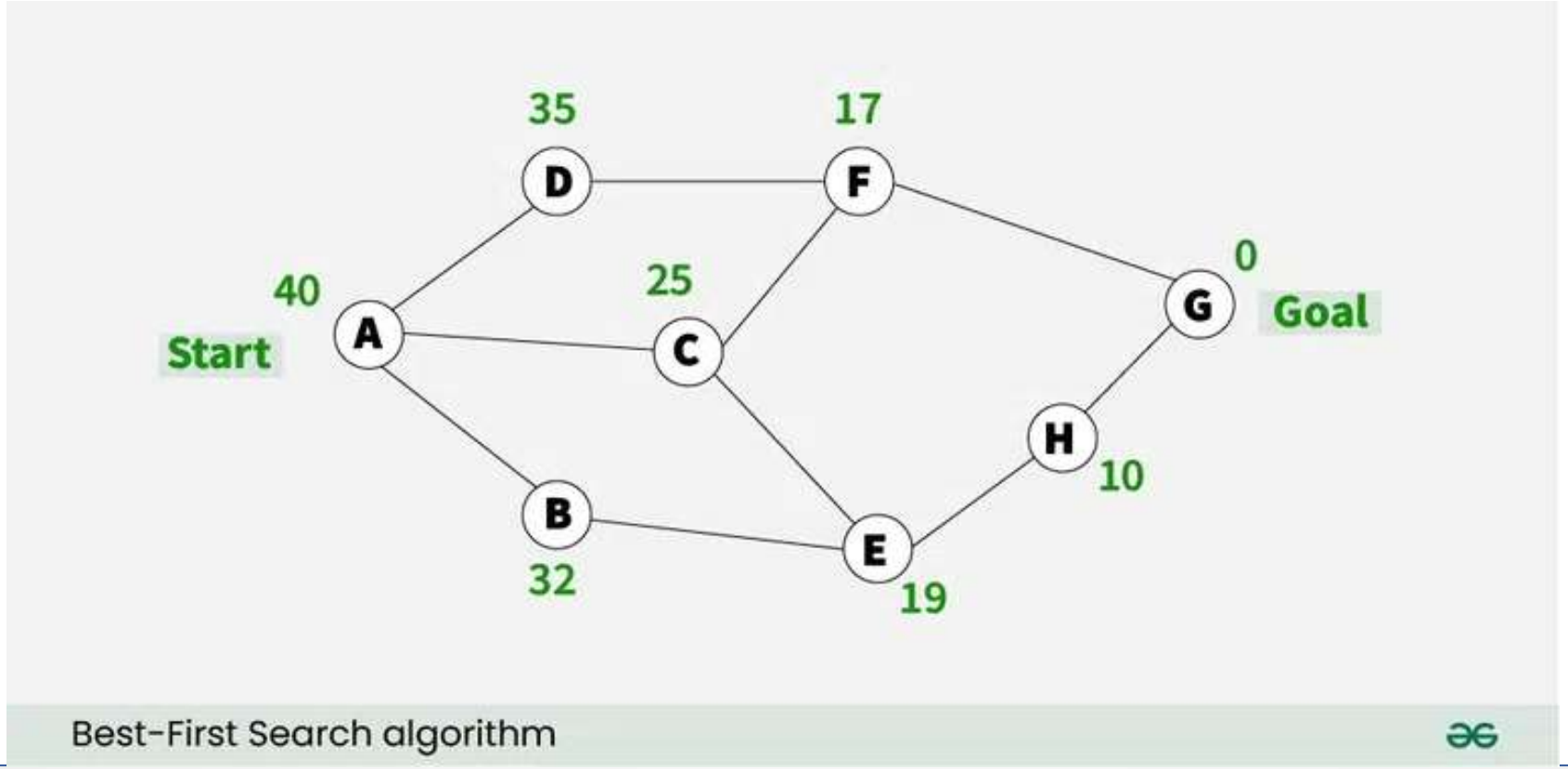
2. **Expand Nodes:** Evaluate all neighboring nodes of the current node. Assign each node a value based on a heuristic function, typically representing the estimated distance to the goal.

3. **Select Best Node:** From the priority queue, select the node with the lowest heuristic value (the node that appears closest to the goal).

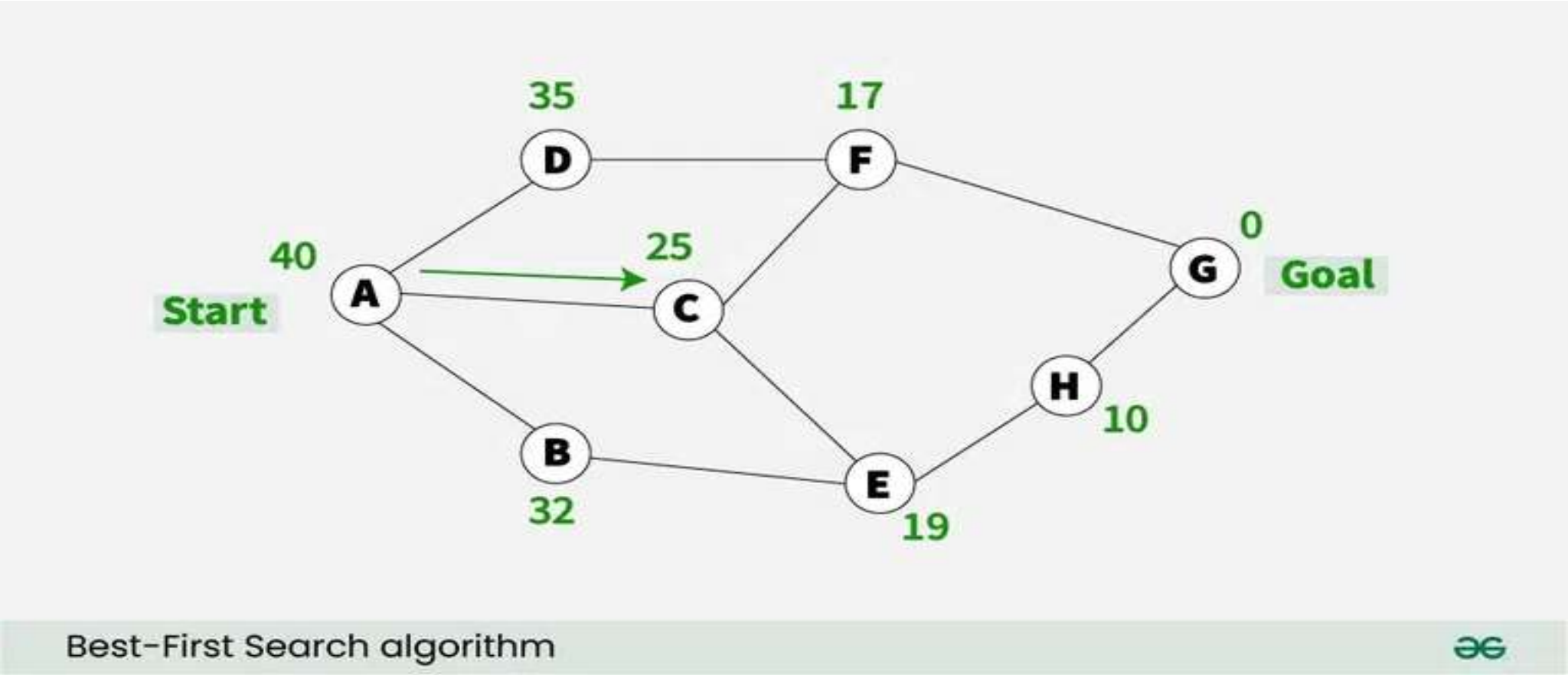
4. **Goal Check:** If the selected node is the goal, terminate the search.

5. **Continue:** If not, repeat steps 2 to 4 for the next node until the goal is reached or the queue is empty.

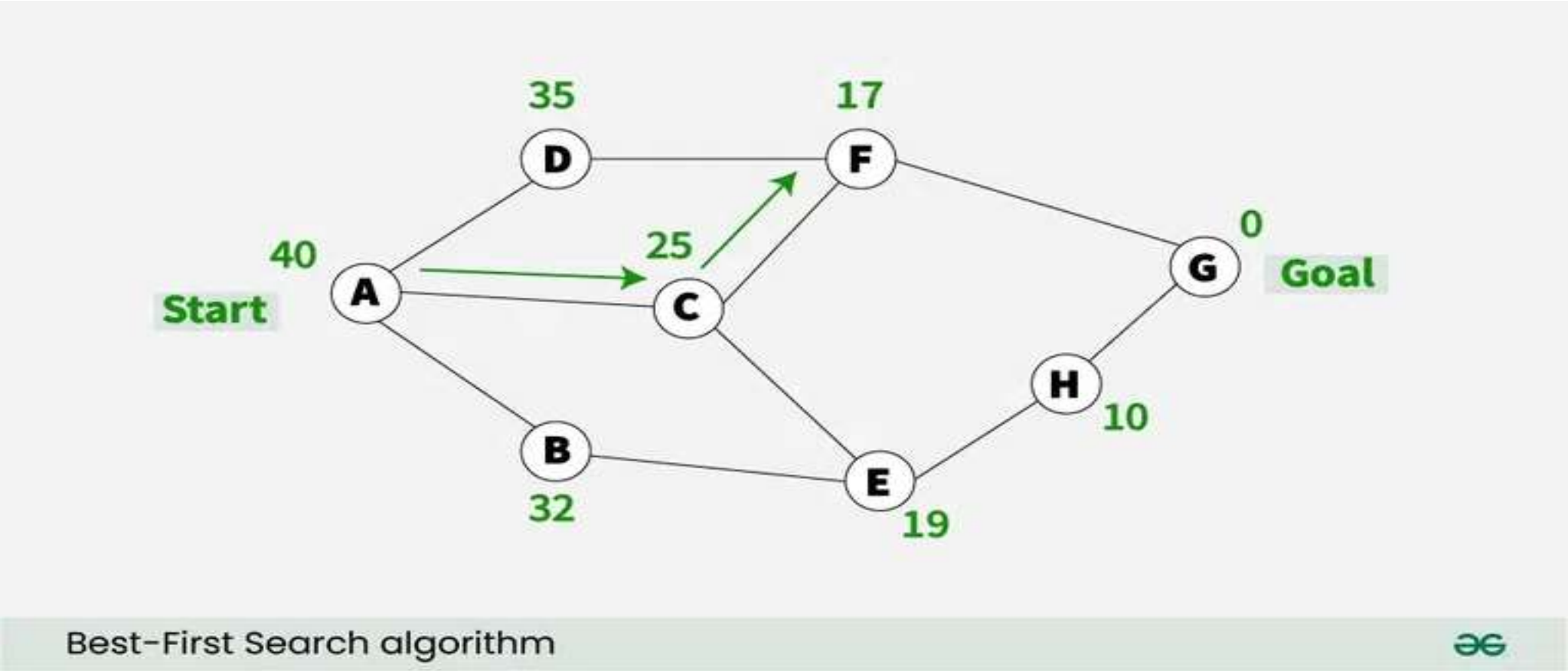
An example of the best-first search algorithm is below graph, suppose we have to find the path from A to G



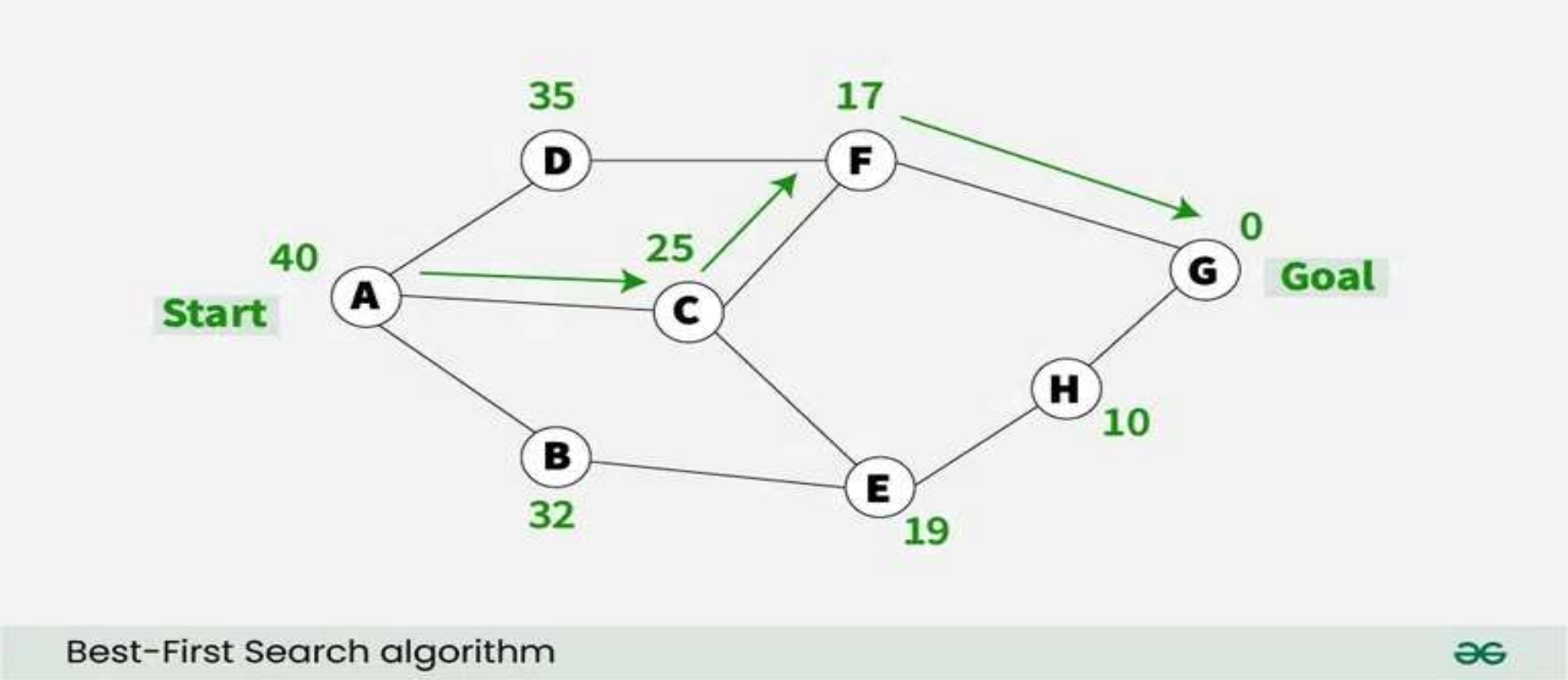
Contd..


35
17
D
F
40
25
0
Start
A
C
G
Goal
H
10
B
E
32
19
Best-First Search algorithm

Contd..


35
17
D
F
40
25
0
Start
A
C
G
Goal
H
10
B
E
32
19
Best-First Search algorithm

Contd..


35
17
D
F
40
25
0
Start
A
C
G
Goal
H
10
B
E
32
19
Best-First Search algorithm

Advantages of Greedy Best-First Search

- **Simple and Easy to Implement:** Greedy Best-First Search is a relatively straightforward algorithm, making it easy to implement.
- **Fast and Efficient:** Greedy Best-First Search is a very fast algorithm, making it ideal for applications where speed is essential.
- **Low Memory Requirements:** Greedy Best-First Search requires only a small amount of memory, making it suitable for applications with limited memory.
- **Flexible:** Greedy Best-First Search can be adapted to different types of problems and can be easily extended to more complex problems.
- **Efficiency:** If the heuristic function used in Greedy Best-First Search is good to estimate, how close a node is to the solution, this algorithm can be a very efficient and find a solution quickly, even in large search spaces.

A* Search Algorithm

Definition:

- A* is a **best-first search algorithm** that finds the shortest path from a start node to a goal node using both the actual cost and a heuristic estimate.
- The A* (A-star) algorithm is a powerful and versatile search method used in computer science to find the most efficient path between nodes in a graph.
- Widely used in a variety of applications ranging from pathfinding in video games to network routing and AI, A* remains a foundational technique in the field of algorithms and artificial intelligence.

- Formula:

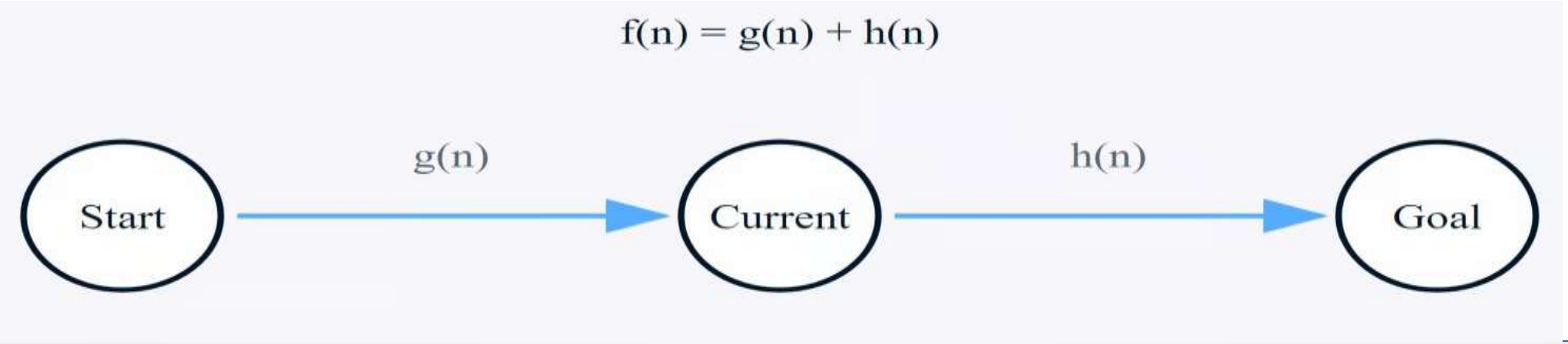
$$f(n)=g(n)+h(n)$$

- **g(n)** = Cost from the start node to the current node n
- **h(n)** = Heuristic estimate of the cost from n to the goal
- **f(n)** = Estimated total cost of the path through n

Key Concepts in A* Search

- **Nodes**: Points in your graph (like intersections on a map)
- **Edges**: Connections between nodes (like roads connecting intersections)
- **Path Cost**: The actual cost of moving from one node to another
- **Heuristic**: An estimated cost from any node to the goal
- **Search Space**: The collection of all possible paths to explore

The A* algorithm's efficiency comes from its smart evaluation of paths using three key components: g(n), h(n), and f(n). These components work together to guide the search process toward the most promising paths.



Heuristic Function In AI

- Heuristic functions are strategies or methods that guide the search process in AI algorithms by providing estimates of the most promising path to a solution. They are often used in scenarios where finding an exact solution is computationally infeasible.
- Heuristic functions play a critical role in artificial intelligence (AI), particularly in search algorithms used for problem-solving. These functions estimate the cost to reach the goal from a given state, helping to make informed decisions that optimize the search process. These techniques help find the most efficient path from a starting point to a goal, making them essential for applications such as navigation systems, game playing, and optimization problems.

Example:

Consider a GPS navigation system. The straight-line distance between two points acts as a heuristic, estimating the travel cost while ignoring obstacles like traffic or road closures.

Heuristic functions are the backbone of informed search strategies, enabling faster and more effective problem-solving in AI.

Contd..

Heuristic Search Algorithm in AI

Heuristic search algorithms leverage heuristic functions to make more intelligent decisions during the search process. Some common heuristic search algorithms include:

- **A* Algorithm**

  The A* algorithm is one of the most widely used heuristic search algorithms. It uses both the actual cost from the start node to the current node (g(n)) and the estimated cost from the current node to the goal (h(n)). The total estimated cost (f(n)) is the sum of these two values:
  f(n)=g(n)+h(n)$f(n)=g(n)+h(n)$

- **Greedy Best-First Search**

  The Greedy Best-First Search algorithm selects the path that appears to be the most promising based on the heuristic function alone. It prioritizes nodes with the lowest heuristic cost (h(n)), but it does not necessarily guarantee the shortest path to the goal.

- **Hill-Climbing Algorithm**

  The Hill-Climbing algorithm is a local search algorithm that continuously moves towards the neighbor with the lowest heuristic cost. It resembles climbing uphill towards the goal but can get stuck in local optima.

Admissibility & Consistency

- In Artificial Intelligence, especially within search and pathfinding algorithms, admissibility and consistency are crucial properties of heuristic functions.

- An admissible heuristic never overestimates the cost to reach the goal, while a consistent heuristic (also called monotone) satisfies an additional property related to the cost between neighboring nodes. While all consistent heuristics are admissible, not all admissible heuristics are consistent.

- The heuristic function h(n) is admissible if **h(n)** is **never larger** than **h*(n)** or if **h(n)** is always less or equal to the true value.
- If **A*** employs an admissible heuristic and **h(goal)=0**, then we can argue that **A* is admissible.**
- If the heuristic function is constantly tuned to be low with respect to the true cost, i.e. **h(n) ≤ h*(n)**, then you are going to get an optimal solution of 100%

Contd..

Admissible and Consistent Heuristics in A* Algorithm

- **Admissible Heuristic:** An admissible heuristic in the A* algorithm is a heuristic function that never overestimates the cost to reach the goal from any given node. In other words, the estimated cost provided by the heuristic function is always less than or equal to the actual cost. This property ensures that A* will always find the optimal solution.
- **Example:** Consider a simple grid with a start node and a goal node. The heuristic function h(n) estimates the straight-line distance from node n to the goal. If the actual cost to reach the goal is 10, the heuristic function should never estimate a cost greater than 10 for any node.
- **Consistent Heuristic:** A consistent heuristic, also known as a monotonic heuristic, satisfies the triangle inequality. This means that for any node n and its successor node m, the estimated cost from the start node to node n, plus the estimated cost from node n to node m, is always less than or equal to the estimated cost from the start node to node m.
- **Example:** In the same grid example, if the heuristic function estimates the cost from node A to the goal as 10, and from node A to node B as 6, then the estimated cost from node B to the goal should be less than or equal to 4 (10 - 6).
- Admissible heuristics guarantee optimality, while consistent heuristics improve the efficiency of the A* algorithm by reducing unnecessary exploration of nodes.

Comparison of Search Algorithms

Key Differences at a Glance

- **BFS**: Level-wise expansion. Optimal for unit cost. Not scalable to large problems.
- **DFS**: Fast and memory-efficient but risks getting stuck in deep branches.
- **UCS**: Expands cheapest path. Optimal, but like BFS, can be slow.
- **Greedy BFS**: Only follows heuristic h(n). Very fast, but not guaranteed optimal.
- **A***: Combines cost-so-far (g(n)) and heuristic estimate (h(n)). Best choice if heuristic is good.

Self Assessment Questions

1. Which of the following is an uninformed search algorithm?

A. A* Search
B. Greedy Best-First Search
C. Breadth-First Search
D. Hill Climbing

2. The function used in A algorithm is:

A. f(n)=h(n)f(n) = h(n)f(n)=h(n)
B. f(n)=g(n)f(n) = g(n)f(n)=g(n)
C. f(n)=g(n)+h(n)f(n) = g(n) + h(n)f(n)=g(n)+h(n)
D. f(n)=h(n)−g(n)f(n) = h(n) - g(n)f(n)=h(n)−g(n)

Self Assessment Questions

3. A heuristic is said to be admissible if it **of the following is an uninformed search algorithm?**

A. Always overestimates the cost
B. Uses past cost only
C. Never overestimates the cost
D. Uses random values

4. Which of the following is not guaranteed to find the optimal solution?

A. A*
B. Uniform Cost Search
C. Greedy Best-First Search
D. Breadth-First Search (unit cost)

Reference Books And Web Links

Text Books:

1. **Artificial Intelligence: A Modern Approach,** Stuart Russell & Peter Norvig, 4th Edition (2021), Pearson.
2. **Introduction to Artificial Intelligence,** Wolfgang Ertel, 2nd Edition (Springer).
3. **Elaine Rich & Kevin Knight, 'Artificial Intelligence', 3nd Edition, Tata McGraw Hill Edition, Reprint (2008)**

Web links:

- AIMA Python Repository (by Norvig)
- Khan Academy – Graph Search Algorithms
- CS50 AI (Harvard OpenCourseWare)
- Coursera – Artificial Intelligence Search Methods