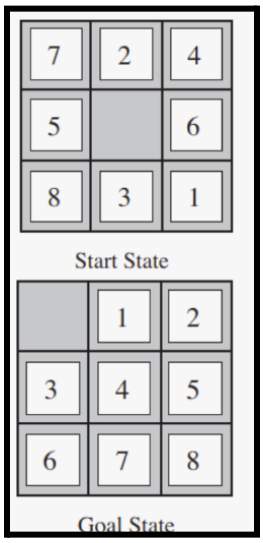
INFORMED SEARCH AND HEURISTIC FUNCTIONS

Informed Search Strategy

- **What is it?** A search strategy that uses extra knowledge beyond just the problem definition.
- **Why use it?** It finds solutions faster and more efficiently than **uninformed search**.
- **Example:** The algorithm knows how far the goal is, how much it costs to get there, and the best path to take.

Heuristic Function



- **What is it?** A function that helps decide the most promising path in informed search.
- **How does it work?** It estimates how close the current state is to the goal.
- **Key points:**
  - It may not always find the perfect solution, but it finds a good one in a reasonable time.
  - It is represented as **h(n)**, where:
    - h(n) = estimated cost from current state to the goal.
  - Always positive.
  - **Formula:** f(n)=h(n)
  - f(n) is the heuristic cost and h(n) is the estimated cost from current state to the goal which is less than or equal to f(n).
  - Example: **8-puzzle problem**, where:
    - **h1** = the **number of misplaced tiles**, all of the eight tiles are out of position, so the start state would have **h1 = 8**
    - **h2** = the sum of the distances of the tiles from their goal positions. This is called Manhattan distance.

Pure Heuristic Search

- The simplest type of heuristic search.
- Uses two lists:
  1. **OPEN List** → Nodes that are yet to be expanded.
  2. **CLOSED List** → Nodes that have already been expanded.
- Steps:
  1. Start with an OPEN list.
  2. Pick the node with the lowest **h(n)**.
  3. Expand it and add it to the CLOSED list.
  4. Repeat until the goal is found.

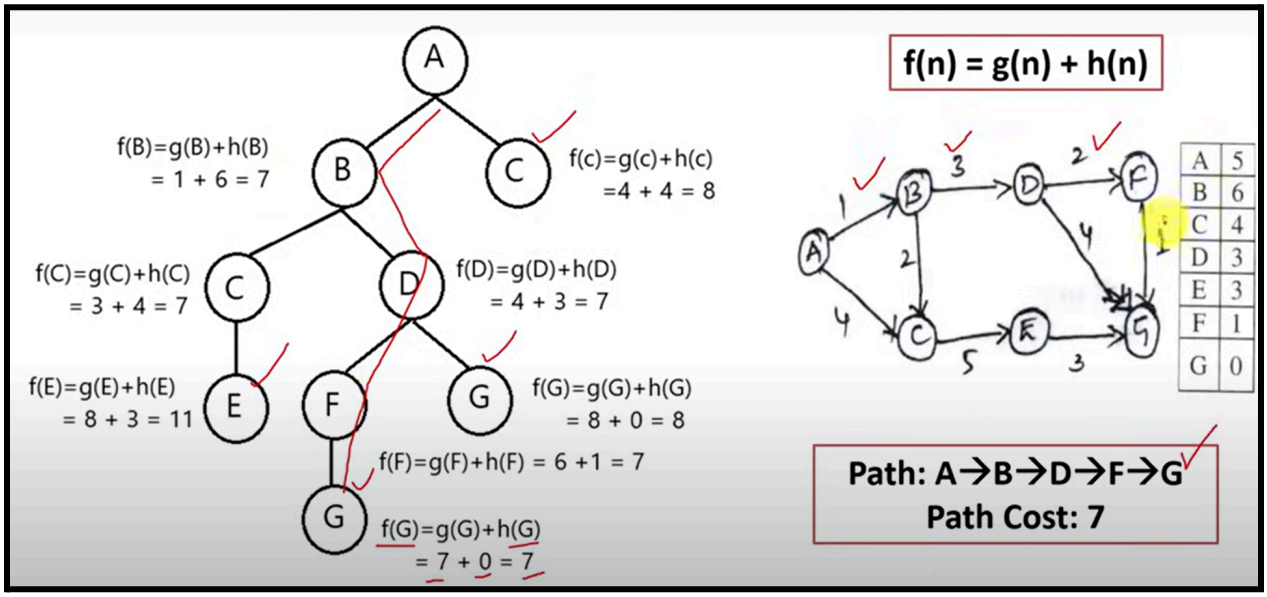
A Search Algorithm*

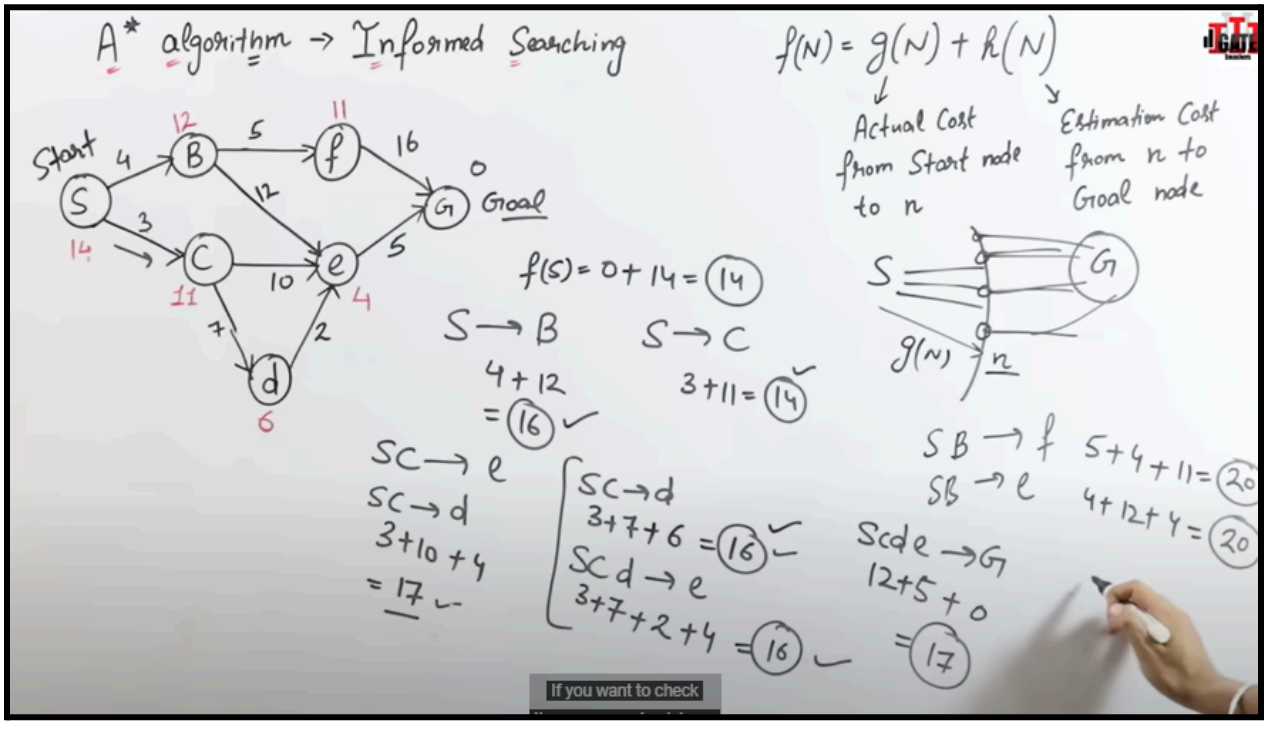
- **What is it?** A popular search algorithm that finds the shortest path.

- **How does it work?** Uses both:
    - **g(n)** = Cost from start to current node **n**.
    - **h(n)** = Estimated cost from node **n** to goal.
    - **Total cost:** f(n)=g(n)+h(n)
- *Why use A?** It finds the best path while exploring fewer nodes.

Steps of A Algorithm:*

1. Put the initial node(start) in the OPEN list.
2. Among the neighbours, Pick the node with the smallest **f(n) = g(n) + h(n)**.
3. If it's the goal node, stop.
4. Expand the node, add successors to the OPEN list.
5. Repeat until the goal is found.





Memory-Bounded Search

- Sometimes A* requires too much memory. These alternatives use **less memory**:
    - *Iterative Deepening A (IDA)***:
        - Like A* but with limited memory.
        - Explores nodes multiple times but saves memory.
    - *Memory-Bounded A (SMA)***:
        - Works like A* but keeps only limited nodes in memory.
        - If memory is too small, it returns the best possible solution.

Iterative Improvement Algorithms

- Used in **optimization problems** where the path isn't important.
- **Types:**
    1. **Hill-Climbing (Gradient Descent)**: Keeps improving the current state until the best solution is found.
    2. **Simulated Annealing**: Inspired by the heating & cooling of metals to avoid getting stuck in local optima.

Advantages and Disadvantages of A Search Algorithm*

✅ Advantages of A*

1. **Guaranteed to find the optimal solution** (if the heuristic is admissible and consistent).
2. **Efficient** – Explores fewer nodes than uninformed search strategies like BFS or DFS.
3. **Uses both actual cost (`g(n)`) and estimated cost (`h(n)`)**, making it more accurate.
4. **Flexible** – Can adapt based on the heuristic function used.
5. **Widely used** in AI, robotics, and game pathfinding (e.g., Google Maps, video games).

❌ Disadvantages of A*

1. **High memory usage** – Stores all generated nodes in memory, making it impractical for very large graphs.
2. **Slow in complex problems** – If many nodes have similar `f(n)`, A* may expand many nodes before finding the goal.
3. **Heuristic-dependent** – Performance heavily depends on how good the heuristic function is.
4. **Computationally expensive** – Requires sorting and updating priority queues frequently.
5. **Not suitable for dynamic environments** – If obstacles or paths change frequently, A* needs to recompute from scratch.

Summary Table

Feature	Advantage	Disadvantage
Optimality	Finds the best solution	Can be slow for complex problems
Efficiency	Explores fewer nodes than BFS/DFS	High memory consumption
Heuristic Use	Guides search intelligently	Needs a good heuristic to work well
Flexibility	Works in many AI applications	May not work well in dynamic environments

---