🔍 1. Search Concepts

In **AI**, search is about exploring possible actions to reach a goal.
The agent doesn't always know the answer — it has to **search** for it.

💡 **Analogy**:
Search is like finding a book in a giant library *without Google*:

- Start at the entrance
- Walk aisle by aisle
- Try different paths until you find the right shelf

📝 2. Problem Formulation

To create a **search problem**, we define **5 ingredients**:

1. **Initial state** → Where we start.
    - Example: *"I'm at home."*
2. **Actions** → What moves are possible.
    - Example: *"Take bus, drive car, walk."*
3. **Transition model** → What happens when we take an action.
    - Example: *"If I take the bus, I move to the bus stop."*
4. **Goal test** → How do we know we're done?
    - Example: *"I reached the university."*
5. **Path cost** → The "expense" of the path (time, money, distance).
    - Example: Driving costs fuel, walking costs time.

💡 **Analogy**: Solving a **maze** → start, allowed moves, walls, exit condition, and steps taken.

🌐 3. Search Space Definition

- The **search space** = all possible states reachable from the start.
- Each state is a **node** in a graph/tree.
- **Edges** represent actions.

💡 **Analogy**: The search space is like the **map of the entire library**, with every aisle and shelf shown.

⚙️ 4. Types of Search Algorithms

There are two main categories:

🔲 Uninformed (Blind) Search

- Agent has **no additional knowledge** — just explores systematically.
- **Examples**: BFS, DFS, Uniform Cost Search
- **Analogy**: Looking for a friend on campus by checking *every classroom one by one*, without knowing their schedule.

🔷 Informed (Heuristic) Search

- Uses extra knowledge (**heuristics = educated guesses**).
- **Examples**: Greedy Best-First Search, A*
- **Analogy**: Instead of checking all classrooms, you **guess** your friend is probably in the computer lab.

🌳 5. Breadth-First Search (BFS)

📖 Definition

- BFS expands the **shallowest (closest)** nodes first.
- Explores **level by level** in the search tree.

📌 Properties

- **Complete**: ✅ If a solution exists, BFS will find it.
- **Optimal**: ✅ Finds the shortest path (*if step cost = 1*).
- **Cons**: ❌ Uses a lot of **memory** (stores all nodes at a level).

🎯 Example

Looking for the **nearest coffee shop** in a new city:

- First check all shops 1 block away
- If not found → check 2 blocks away
- Continue outward until found

💡 **Analogy**: Like **waves spreading in water** — BFS explores outward equally in all directions.

📊 Comparison Table: BFS vs DFS

Property	BFS	DFS
Strategy	Explore level by level	Explore one path deep before backtracking
Memory	High (stores many nodes)	Low (just the path stack)
Completeness	✅ Yes	❌ Not always (may get stuck in infinite path)
Optimality	✅ Yes (shortest path if uniform cost)	❌ Not always

Property	BFS	DFS
Analogy	Checking all nearby classrooms first	Going into one building's basement all the way down

✅ Key Takeaways

- **Problem formulation** = define states, actions, transitions, goal, cost.
- **Search space** = all possible states (nodes in a tree/graph).
- **Uninformed search** = no hints (BFS, DFS).
- **BFS** = complete + optimal, but memory expensive.