📕 LangGraph Core Concepts

---

1. ⚡ Quick Recap: What is LangGraph?

- **LangGraph = Orchestration framework** for LLM workflows.
- It represents any **workflow as a graph**.
- **Key ideas:**
  - Each **node** = a task.
    - Example: LLM call, tool call, decision-making.
  - **Edges** = execution order (which task comes next).
- Execution flow:

1. Define workflow → Convert into graph.
2. Provide input to the **first node**.
3. Execution proceeds node by node (or parallel/branching/loops).

- Extra features:
  - Parallel execution
  - Loops
  - Branching (conditional flow)
  - Memory
  - Resumability (resume from failure point)
- Conclusion:
  - Ideal for **agentic** and **production-grade AI applications**.

---

3. 🧩 Core Concept 1 – LLM Workflows

- **Definition of Workflow:**
  A **series of tasks executed in order to achieve a goal**.
- Example: Automated hiring
  - Create JD → Post → Shortlist → Interview → Onboard.
- **Definition of LLM Workflow:**
  A workflow where many tasks depend on **LLMs**.
  - Example: Hiring flow (LLM used for JD writing, shortlisting, interviews).
- **General structure:**
  - Step-by-step process.
  - Each step = distinct task like:
    - Prompting
    - Reasoning
    - Tool calling
    - Memory access
    - Decision-making
- Workflows can be:
  - Linear
  - Parallel
  - Branching
  - Looped

🔑 Five Common LLM Workflows

1. **Prompt Chaining**
   - Sequentially call LLM multiple times.
   - Example: Topic → Generate outline → Write full report.
   - Add validation checks in between (e.g., word limit).
2. **Routing**
   - LLM acts as a **router** to decide which specialized model handles query.
   - Example: Customer support bot routes query to refund/tech/sales model.
3. **Parallelization**
   - Break one task into multiple subtasks → execute in parallel → merge results.
   - Example: YouTube content moderation:

        - Check community guidelines
        - Check misinformation
        - Check sexual content
    - Run in parallel, then aggregate decision.
4. **Orchestrator–Worker Pattern**
    - Similar to parallelization but **subtasks are not pre-defined**.
    - Orchestrator assigns tasks dynamically based on query.
    - Example: Research assistant → Query decides whether to search Google Scholar or Google News.
5. **Evaluator–Optimizer Loop**
    - Iterative refinement (like writing drafts).
    - Generator LLM → produces output.
    - Evaluator LLM → accepts/rejects with feedback.
    - Loop continues until evaluator approves.
    - Example: Writing emails/blogs/stories.

---

4. 🧩 Core Concept 2 – Graphs, Nodes, and Edges

- LangGraph represents workflows as **graphs**.
- **Example: UPSC Essay Practice Website**
    - Flow: Generate topic → User writes essay → Collect essay → Evaluate (clarity, depth, language) → Score → Congratulate or give feedback → Option for retry → Loop continues.
    - Represented as a graph:
        - Each step = **node**.
        - Flow = **edges**.
- **Nodes**
    - Each node = **single task**.
    - Behind the scenes → Node = Python function.
    - So LangGraph = set of Python functions connected via edges.
- **Edges**
    - Define execution order.
    - Types:
        - Sequential edges (step by step).
        - Parallel edges (tasks run simultaneously).
        - Conditional edges (branching).
        - Loop edges (repeat until condition).
- **Summary:**
    - Nodes = What to do.
    - Edges = When to do.

---

5. 🧩 Core Concept 3 – State

- **Definition:**
    - Shared memory that flows through workflow.
    - Holds all required data during execution.
- Characteristics:

1. Required for execution.
2. Evolves/updates over time.

- Example (Essay workflow):
    - Data points = essay text, topic, scores (clarity, depth, language), overall score.
- Properties:
    - **Accessible to all nodes**.
    - **Mutable** (nodes can update it).
    - Passed from one node → next as input.
- Implementation:
    - Defined before building graph.
    - Usually a **Typed Dictionary** (or Pydantic model).

---

6. 🧩 Core Concept 4 – Reducers

- **Problem:**
  - By default, nodes overwrite state values (mutable).
  - Example: Chatbot → user’s name erased after new message.
- **Solution: Reducers**
  - Define how updates apply to state.
  - Options:
    - Replace (default)
    - Add (append)
    - Merge
- Example:
  - Essay workflow → Keep all drafts (add), not just last one.
  - Chatbot → Store complete conversation history instead of overwriting.

---

7. 🧩 Core Concept 5 – Execution Model

- Inspired by **Google Pregel** (large-scale graph processing).
- **Steps:**
  1. **Graph Definition**
     - Define nodes, edges, and state.
  2. **Compile**
     - Check graph consistency (no orphan nodes).
  3. **Execution**
     - Start by invoking first node with initial state.
     - Node executes function → updates state → passes updated state to next node(s).
     - Process = **message passing** (via edges).
     - Parallel nodes execute together → called **superstep** (not just step).
     - Continues until no active nodes or messages remain.
- **Key Terms:**
  - **Message Passing** → transfer of state between nodes.
  - **Superstep** → one round of execution (may involve multiple parallel steps).

---

8. ✅ Summary

- **LangGraph = orchestration framework for building stateful, multi-step, intelligent LLM workflows.**
- Core concepts covered:
  1. **LLM Workflows** → definition + 5 common patterns.
  2. **Graphs, Nodes, Edges** → tasks & flow structure.
  3. **State** → shared evolving memory.
  4. **Reducers** → define update policies for state.
  5. **Execution Model** → inspired by Google Pregel (message passing, supersteps).
- With these concepts, upcoming coding videos will be easier to follow.
