Lean 4 Main Workflow Overview:

Architecture

Task Initialization

- Tasks are identified by their ID from the `task_ids` list
- Initial configurations include utilization of RAG to extract the top k (default to 3) most relevant source chunk (token_size=256) for reference.
- Function information is extracted from the template (description.txt and task.lean)

Agent Roles

- Planning Agent: (1) Analyzes problems, creates implementation strategies for the initial try and (2) revises plans based on previous attempt and error message.
- Generation Agent: Produces Lean 4 code and proofs based on the planning agent's suggestion.
- Verification Agent: Tests code execution, analyzes errors, suggests improvements

RAG Integration

- Uses semantic search to find relevant Lean 4 examples
- Retrieves specific examples for:
    - General implementation patterns
    - Proof techniques
    - Error-specific solutions
- Adapts search queries based on error messages
- Sources are retrieved from the web with the help of Perplexity and Claude, following the instruction that they are derived from Lean4 documentations or community platforms.

Workflow Loop

The pipeline implements an iterative approach (defaults to max 1 iterations; note the original value was 3, which results in ~7 mins of runtime for one task. Since there has been no significant improvement result, I have aborted multi-iterations):

1. Planning Phase
    - Initial analysis of problem description and template
    - Plan revision based on previous errors when applicable
    - Strategy development for implementation and proof

2. Implementation Phase
    - Code generation guided by the plan and examples

- Error tracking to avoid repeating mistakes
- Verification through execution of Lean code

3. Proof Phase
   - Proof generation based on successful implementation
   - Incorporation of proof-specific examples
   - Error analysis and reflection for subsequent attempts

4. Verification
   - Full solution testing with both implementation and proof
   - Error-specific diagnostic feedback
   - Solution iteration until success or max attempts reached

Tracking and Feedback

- Records all attempts, errors, and solutions
- Saves intermediate and final results to files
- Maintains history to avoid repeating the same mistakes

Output (for visualization)

- Final solution combines verified function implementation and proof
- Results are saved in a structured format for evaluation (as txt file)
- Execution logs provide insights into the reasoning process (this requires running “make test >> logs/test_n.txt”, i.e. this is just recording the terminal message)

Note:

- The RAG database is obtained through scraper.py file. To obtain the documents, run python3 scraper.py --append-all (on the first try) and --rebuild for all subsequent runs. Other log arguments are less necessary and were used for validation during construction.
- submission.zip only contains all non ,py/.pkl files as per the suggestion made by GSI on Ed.