Project Proposal: Evaluating LLM Program Efficiency

Problem: The rapid rise of Large Language Models (LLMs) like GPT-4 has introduced new capabilities in code generation, allowing developers to automate tasks such as writing, optimizing, and debugging code. While there exist evaluations of LLMs based on the syntactic correctness and accuracy of generated code, an important metric has been overlooked: the efficiency of the generated programs. Given that code generation may result in various implementations of the same algorithm, the most efficient solution in terms of runtime, memory usage, and overall computational complexity is often not guaranteed. This paper will seek to evaluate the efficiency of LLM generated programs through various metrics to determine their feasibility, especially in algorithm generation. Current evaluations do not consider metrics such as resource efficiency, leading to the generation of inefficient or resource-heavy programs.

Challenges:

1. **Identifying Efficiency Metrics:** Defining metrics such as runtime, memory consumption, and CPU usage that apply across languages and domains.
2. **Dataset Curation:** It may be difficult to find datasets to perform our evaluation that contain both: a holistic range of problems spanning all subject domains and a human-produced solution that is known to be the most optimal.
3. **Automating Analysis:** Evaluating efficiency involves both static analysis (e.g., complexity analysis, memory profiling) and dynamic analysis (e.g., measuring runtime performance on specific hardware). Building automated tools to perform these analyses consistently and reliably will be a significant technical challenge.
4. **Framework Development:** There will likely be technical challenges & software design involved in creating a framework to run these tests, similar to EvalPlus' framework architecture.

Approach:

1. **Metric Definition:** We will focus on identifying key efficiency metrics that reflect program quality beyond correctness, including runtime performance (measured through execution time), memory usage (tracked using profiling tools), redundant variable storage (through static code analysis), and algorithmic efficiency (using Big-O complexity analysis).
2. **Data Curation:** After determining metrics for evaluation, we will need to search for suitable prompt-code pairs that depict highly efficient code for comparison to LLM-generated code.
3. **Metric Comparison:** Utilizing the metrics determined earlier, compare the output/efficiency of the LLM generated code (if it is correct) to the output/efficiency of the "gold label" code.
4. **Refining Metrics:** Identify where LLMs fall short and which metrics are most predictive of suboptimal performance.

Evaluation:

1. **Metrics Accuracy:** We will validate the accuracy of our selected efficiency metrics (runtime, memory usage, redundant variable storage, and algorithmic efficiency) by

comparing the LLM-generated code against human “gold label” solutions. Success will be determined by the extent to which our metrics can distinguish between efficient and inefficient implementations across various algorithms.
2. **Baseline Comparison:** We will compare the LLM-generated code with "gold label" human-produced code using our efficiency metrics. Success will be indicated by our framework's ability to identify cases where LLMs generate suboptimal code, providing clear insights into areas where LLMs fall short or excel in efficiency.
3. **Dataset Quality:** The curated dataset of prompt-code pairs should be comprehensive, covering a broad range of problem domains with corresponding efficient human solutions. Success will be measured by the dataset's applicability and usefulness in further research and development within this field.

Timeline:

Week 1-2 (Sep 20 - Oct 3)	Week 3-4 (Oct 4 - Oct 17)	Week 5-6 (Oct 18 - Oct 31)	Week 7-8 (Nov 1 - Nov 14)	Week 9 (Nov 15 - Nov 21)	Week 10 (Nov 22 - Nov 29)	Week 11 (Nov 30 - Dec 2)
Metric Definition & Data Curation	Framework Design & Implementation	Framework Integration & Initial Testing	Metric Evaluation & Framework Refinement	Final Testing & Analysis	Report Writing & Final Deliverables	Presentation Preparation & Final Refinements
Aditya: Metric definition and literature review Arnav: Data curation and dataset collection	Aditya: Framework architecture and static analysis Arnav: Dynamic analysis implementation	Aditya: Static tools integration Arnav: Dynamic tools integration and testing	Aditya: Static analysis evaluation Arnav: Dynamic analysis evaluation	Aditya: Final testing and static metrics analysis Arnav: Final testing and dynamic metrics analysis	Aditya: Report on metric definition and static analysis Arnav: Report on data curation and dynamic analysis	Aditya and Arnav: Presentation preparation and report finalization

Deliverables

1. Extended EvalPlus Framework: A potential extension of the EvalPlus system to evaluate LLM-generated code based on defined efficiency metrics.
2. Efficiency Evaluation Dataset: A curated dataset of programs for benchmarking and further research on code efficiency.
3. Selected Metrics for LLM Code Generation: A set of the top metrics for comparing LLM code generation to efficient human-written code.
4. Final Report: A comprehensive report detailing our methodology, evaluation results, and insights on the comparative efficiency of LLM-generated and human-written code.

This project aims to enhance AI-assisted programming by providing tools and methodologies for optimizing code generation efficiency.