Comparing Chat GPT-4o and OpenAI-o1 preview for High-Level Problem Solving

Amr Ibrahim, **Solha Park**, **David Zhao**

Stony Brook University

GitHub repository: https://github.com/AI-AmrIbrahim/High-Level-GPT-vs-OpenAI-ProblemSet

1 Introduction

Advances in large language models (LLMs), such as GPT-4o and OpenAI-o1, have enabled impressive capabilities in tackling complex tasks like advanced-level coding and PhD-level mathematics. However, a detailed comparison of their performance on high-level problems remains underexplored. This study evaluates their ability to solve challenges like LeetCode Hard coding problems and PhD-level data analysis problems, aiming to uncover their strengths, limitations, and implications for future AI development.

GPT-4o, a versatile and highly capable model, excels in complex reasoning and creative tasks, while OpenAI-o1 is optimized for efficiency in specific contexts. These design differences highlight the trade-offs between general-purpose models and those tailored for targeted applications.

This research focuses on comparing their performance in terms of speed, accuracy, and resource usage, using custom metrics to evaluate solution correctness, efficiency, and robustness. It also analyzes their problem-solving approaches to identify strengths, weaknesses, and suitability for specific tasks. The findings will offer insights into the capabilities of general-purpose versus optimized LLMs, providing guidance for selecting and developing AI systems for complex problem-solving applications.

2 Related Works

Our work builds on OpenAI's *"GPT-4 Technical Report"*, which highlights GPT-4's performance on professional and academic benchmarks, where it scored in the top 10%—an improvement over GPT-3.5, which ranked in the bottom 10%. Despite its advancements, GPT-4 showed weaknesses in benchmarks like HumanEval (Python coding tasks) and DROP (reading comprehension and arithmetic) [1]. OpenAI introduced the o1 model to address these shortcomings, incorporating extended reasoning processes similar to chain-of-thought reasoning. Our research expands on the o1-preview model's capabilities by evaluating its performance on LeetCode Hard problems and complex data analysis and qualifying exam questions.

3 Methodology

To evaluate GPT-4o and OpenAI-o1, we will test them on a diverse set of benchmark tasks, including LeetCode Hard coding challenges and PhD-level math problems. Tasks will be categorized by complexity and topic, and both models will be tested under identical, zero-shot conditions to ensure a fair comparison.

Challenges include defining success criteria for diverse tasks, evaluating complex problems in math and coding, and analyzing model reasoning for incorrect solutions. To address these, this study will use a balanced problem set, custom metrics, and detailed error analysis to ensure a thorough and unbiased evaluation of both models. Custom metrics will measure correctness (accuracy of solutions), efficiency (time and space complexity), reasoning quality (clarity and logic), and robustness (handling ambiguous or modified inputs). Error analysis will examine incorrect outputs and their proximity to correct reasoning, providing deeper insights into the models' capabilities.

Member	Responsibilities
Amr	● Github management ● LeetCode Problem Querying ● LeetCode Evaluation
David	● Math Problem Selection ● Math Rubric Development ● Math Evaluation
Solha	● Math Problem Selection ● Math Rubric Development ● Math Evaluation

Division of work among group

3.1 Leetcode Problems

Problem Selection:

A diverse set of LeetCode Hard problems was selected to ensure comprehensive coverage across various algorithmic challenges, including Dynamic Programming, Graph Traversals, and Binary Search. These problems were queried using the open-source Alfa LeetCode API (GitHub Repository). This API facilitated efficient retrieval of problem title slugs based on specific tags and difficulty levels. A complete list of problem tags and the corresponding problem counts queried is provided in Appendix A.

Problem Retrieval:

The problems were retrieved and stored using the following steps:

1. API Query: Python scripts were developed to interact with the Alfa LeetCode API (https://github.com/alfaarghya/alfa-leetcode-api), enabling the programmatic retrieval of problems filtered by tags and difficulty.
2. Problem Description: Each problem's description was saved as an HTML string, preserving any images or formatting present on the LeetCode platform. This ensured all essential context was available during solution evaluation.
3. Storage: Problems were queried and added to the dataset using the query_leetcode_problems() and get_problem_description_and_add() functions from our dataset manager tool (https://github.com/AI-AmrIbrahim/High-Level-GPT-vs-OpenAI-ProblemSet)

Metrics:

The performance of GPT-4o and OpenAI-o1-preview was evaluated using metrics derived from LeetCode's submission system:

- Runtime Beats: Percentage of submissions the solution was faster than.
- Memory Beats: Percentage of submissions the solution used less memory than.

In addition to raw performance metrics, composite metrics were calculated for deeper insights:

- Simple Average: Arithmetic mean of runtime and memory beats:

$$\text{Simple Average} = \frac{\text{Runtime Beats} + \text{Memory Beats}}{2}$$

- Weighted Average: Runtime beats were weighted more heavily due to memory being less critical in modern, cloud-based systems:

$$\text{Weighted Average} = 0.6 \times \text{Runtime Beats} + 0.4 \times \text{Memory Beats}$$

Evaluation:

The evaluation process was implemented as follows:

1. Prompting and Solution Generation: A consistent, standardized prompt was used across all problems to ensure fairness. The prompt emphasized correctness, efficiency, and clarity, aligning with LeetCode's judging criteria. This prompt remained unchanged throughout the study to maintain consistency shown in Appendix B for reference.
2. LeetCode Evaluation: Solutions were submitted to the LeetCode platform to obtain runtime and memory performance metrics.
3. Feedback and Manual Annotation: Each solution was manually reviewed and annotated with feedback regarding its performance. Solutions scoring below 70% in either Runtime Beats or Memory Beats were flagged for optimization, while solutions scoring 70% or

higher were considered correct without requiring improvements. This threshold was set arbitrarily based on high expectations of LLMs in problem-solving tasks.

3.2 Data Analysis Problems

Problem Selection:

A diverse set of data analysis problems was selected from *Statistics and Data Analysis* by Tamhane and Dunlop (Pearson) [2], covering a variety of topics: sampling distribution of statistics, basic concepts of interference, inference for single samples, inference for two samples, inference for proportions and count data, simple linear regression and correlation, analysis of single-factor experiments, and nonparametric statistical methods. In addition, we also had past data analysis qualifying exam problems included [3].

Metrics:

The performance of GPT-4o and OpenAI-o1-preview was evaluated using a custom math rubric, as our problem involved advanced mathematical concepts, and linguistic analysis alone was insufficient. We opted for a more sophisticated human evaluation approach. The rubric included five distinct categories that assessed the logical reasoning involved in solving a data analysis problem: correctness of the final answer, correctness of intermediate steps, clarity and depth of explanation, completeness, and use of appropriate methods. Each category was weighted differently to emphasize the importance of intermediate steps and logical flow in problem-solving. The resulting math score is calculated as follows:

$$\begin{aligned}\text{Math Score} &= 0.25 \times \text{correctness\_final} + 0.3 \times \text{correctness\_steps} + 0.2 \times \text{clarity\_explanation} \\ &\quad + 0.15 \times \text{completeness} + 0.1 \times \text{appropriate\_methods}\end{aligned}$$

For each of these categories, it was graded on a 5-point Likert scale, with specific descriptions of each category and score shown in the rubric, shown in Appendix C.

Evaluation:

The evaluation process for the prompting and solution generation followed a consistent, standardized prompt that was used across all problems to ensure fairness. The prompt has a context as an expert statistician and mathematician and asks for detailed steps solving the problem and writing solutions in LaTeX for mathematical expressions. This prompt remained unchanged throughout the study to maintain consistency included in Appendix D for reference.

4 Results

4.1 Leetcode Problems

The o1-preview model shows improvements across all metrics as shown in Table 1. The improvements are shown in green where the mean and median improvements are increasing and standard deviation improvements are shown as decreasing.

	Runtime Beats	Memory Beats	Simple Average	Weighted Average
GPT-4o Mean	49.92	39.40	44.66	45.72
GPT-4o Median	59.45	41.02	55.63	58.65
GPT-4o StDev	40.08	36.51	35.10	35.62

	Runtime Beats	Memory Beats	Simple Average	Weighted Average
o1-preview Mean	53.53 (**+3.61**)	43.10 (**+3.70**)	48.31 (**+3.65**)	49.36 (**+3.64**)
o1-preview Median	66.05 (**+6.60**)	41.05 (**+0.03**)	56.19 (**+0.56**)	60.65 (**+2.00**)
o1-preview StDev	38.12 (**-1.96**)	35.59 (**-0.92**)	32.41 (**-2.69**)	32.88 (**-2.74**)

Table 1. Summary Statistics for GPT-4o and o1-preview Leetcode Performance Metrics

The next step is to test whether these improvements are statistically significant at $\alpha = 0.05$. The distribution of differences for these metrics were non-normal from the Shapiro–Wilk test, therefore we used the Wilcoxon signed rank sum test. For all different scoring metrics, we do not have evidence that the o1-preview model performed better than the GPT-4o model as shown in Table 2.

	Wilcoxon statistic	p-value
Runtime Beats	2354	0.1508
Memory Beats	2769	0.1524
Simple Average	2754	0.1651
Weighted Average	2737	0.1802

Table 2. One-sided paired Wilcoxon signed rank sum test statistic and p-value at $\alpha = 0.05$ comparing o1-preview and GPT-4o for LeetCode metric differences

Figure 1 shows the distribution of feedback comparing o1-preview and GPT-4o. Feedback covered various leetcode outputs including: runtime error, time limit exceeded, wrong answer, and various values for runtime and memory percentage beats. Overall o1-preview performed better than GPT-4o by having less wrong answers however that came with unique issues such as solution in Python not SQL and more runtime errors.

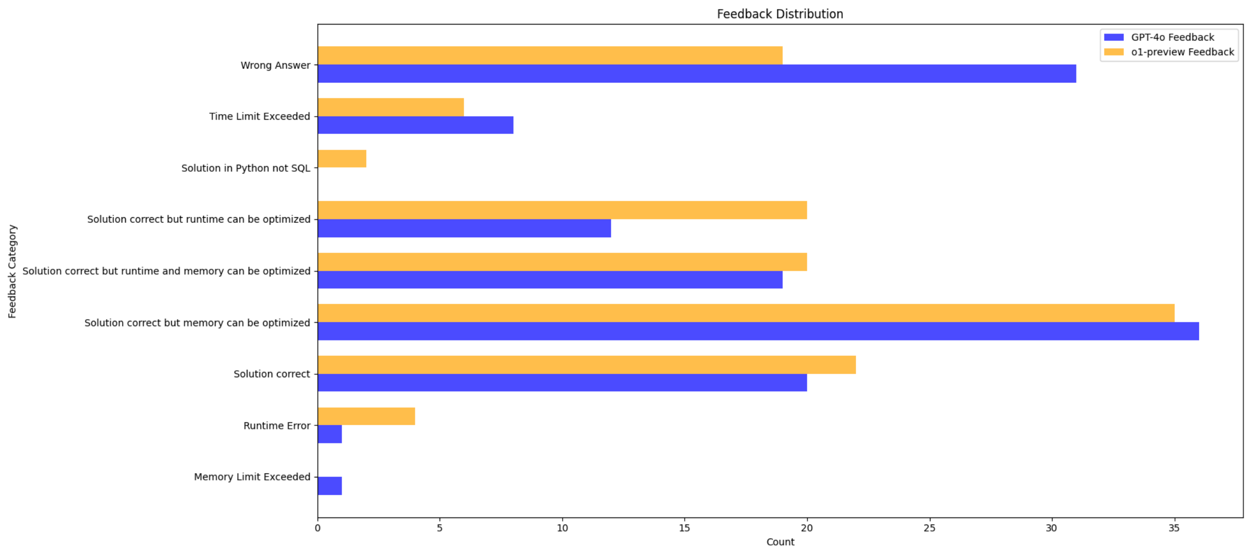


Figure 1. Distribution of Feedback Scores for GPT-4o and o1-preview

Figure 2 demonstrates that the differences in performance for each metric are predominantly centered around zero, indicating a largely comparable performance between the two models. The mean difference, represented by the red dashed line in each plot, reinforces this observation, as it hovers close to zero across all metrics. The symmetric shape of the distributions suggests a balanced variance, with no extreme outliers heavily favoring either model. Notably, the spread is slightly more pronounced for the simple and weighted average metrics, reflecting more variability in overall task performance. These results align with the hypothesis that while o1-preview may outperform GPT-4o in some scenarios, the differences are not overwhelmingly consistent across all tasks. This analysis serves as a visual corroboration of the statistical findings, confirming that the observed performance differences are generally modest and task-dependent.

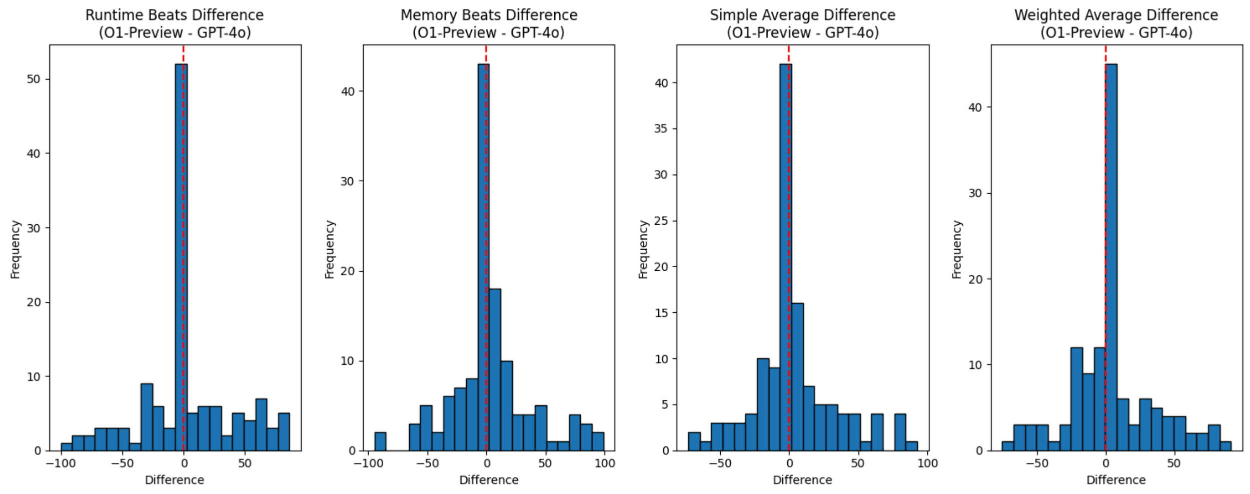


Figure 2. Difference in Leetcode Performance Metrics (o1-preview - GPT-4o)

4.2 Data Analysis Problems

After evaluating the solutions provided by GPT-4o and o1-preview, the o1-preview model demonstrated improvements across all statistics, as shown in Table 3. These improvements are highlighted in green, with increases in mean and median values and a decrease in standard deviation.

	GPT-4o	o1-preview
Mean	4.4377	4.7451 (**+0.3074**)
Median	4.5	5.0 (**+0.5**)
StDev	0.5399	0.4186 (**-0.1213**)

Table 3. Summary Statistics for GPT-4o and o1-preview Math Performance Metrics

Figure 3 illustrates the differences in performance on the math metric, showing a unimodal distribution slightly skewed to the right. The dotted vertical line at 0 represents no difference. The skewness is supported by the observation that the mean difference exceeds the median difference, as detailed in Table 4.

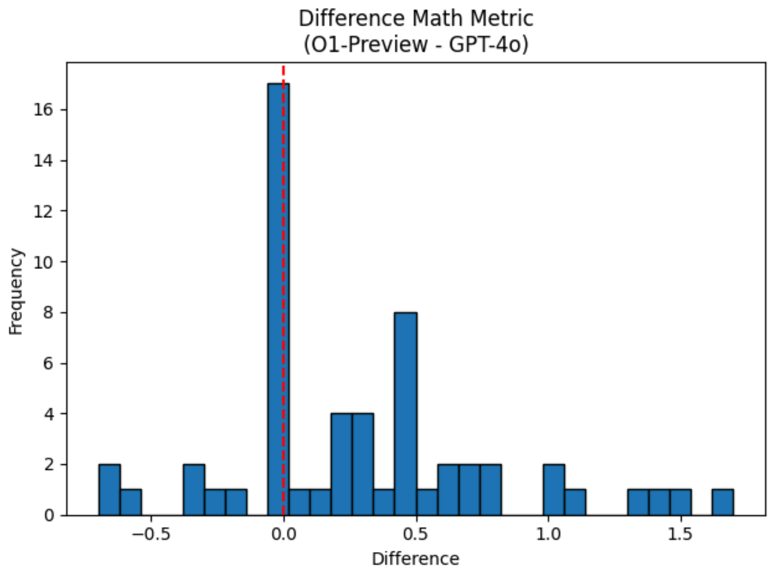


Figure 3. Difference in Math Performance Metrics (o1-preview - GPT-4o)

To determine whether these improvements are statistically significant at $\alpha = 0.05$, we conducted a Wilcoxon signed-rank test, indicated by the Shapiro–Wilk test and the distribution showing non-normality in Figure 3. The test yielded a p-value of 0.000036, which is below the significance level threshold. Thus, we have strong evidence that the o1-preview model outperforms GPT-4o on data analysis problems.

	Diff Stats
Mean	0.3075
Median	0.2000
StDev	0.5087
Wilcoxon statistic	736.5
p-value	0.000036

Table 4. One-sided paired Wilcoxon signed rank sum test statistic and p-value at α = 0.05 comparing o1-preview and GPT-4o for math metric differences

5 Discussion

Our research faced challenges due to financial constraints, limiting the number of LeetCode and math problems we could test. Additionally, the process was labor-intensive, requiring manual inputs to calculate solution percentiles and perform human evaluations for math problems. Our results showed that o1-preview excels in mathematical reasoning but is less suited for complex coding tasks. This disparity may be influenced by the relative nature of the coding evaluation metric used.

LeetCode metrics, such as runtime and memory beats, are inherently relative measures, as they compare a solution's performance to the distribution of all other submitted solutions for a specific problem. These scores fluctuate as new entries are added, potentially improving or worsening the relative ranking of a solution. While useful for gauging competitive performance, their dynamic nature introduces inconsistency, especially when problems accumulate more diverse or optimized submissions over time. A more robust metric would focus on absolute performance measures, such as runtime in milliseconds and memory usage in kilobytes, which remain fixed regardless of external changes. These metrics allow for consistent comparisons across different models and datasets, ensuring that evaluations are not influenced by shifts in the community's performance distribution.

Additionally, one of the major challenges encountered in this study was the lack of an API or automated process to run the generated solutions directly on LeetCode. This limitation required manually copying and pasting each generated solution into the LeetCode interface, retrieving the resulting metrics, and then manually inputting those values into the dataset manager. This highly repetitive and time-intensive process consumed the majority of our time during data collection, detracting from efforts that could have been spent on deeper analysis or model development. As a next step, developing tools or leveraging browser automation frameworks to streamline this workflow would significantly reduce manual effort and allow future research to focus more on improving and evaluating models rather than on tedious data handling tasks.

For the math problems, GPT-4o demonstrated a solid understanding of definitions and equations but often made simple mathematical errors. For instance, in a problem involving the expected value of a sampling distribution, it correctly defined the expected value but introduced inconsistencies when substituting values for the numerator, leading to an incorrect intermediate step but a correct final answer.

1. **Expected Value $E(\bar{X})$:**

$$E(\bar{X}) = \sum \bar{x} \cdot P(\bar{x}) = 1.5 \cdot \frac{1}{10} + 2 \cdot \frac{1}{10} + 2.5 \cdot \frac{1}{5} + \ldots + 4.5 \cdot \frac{1}{10}$$

Calculating this:

$$E(\bar{X}) = \frac{1.5 + 2 + 10 + 12 + 14 + 12 + 9}{20} = \frac{60}{20} = 3$$

A notable difference between the models is that o1-preview rounds its numerical calculations to more digits than GPT-4o, resulting in answers closer to the correct range. When given the same prompt, GPT-4o tends to provide a broad, general explanation without performing the numerical computations, whereas o1-preview demonstrates improved precision and directly addresses areas where GPT-4 previously struggled, as highlighted in its technical report. For future research, incorporating evaluations by domain experts in mathematics, statistics, and computer science can help reduce the subjectivity of the metric system.

Appendix

Appendix A: Problem tags and counts of LeetCode Hard Problems

LeetCode Question Tag	Number of Questions Queried out of 15
Array	15
String	15
Hash Table	15
Dynamic Programming	15
Math	15
Sorting	15
Greedy	15
Depth-First Search	15
Database	1
Binary Search	10
Matrix	15
Tree	15
Recursion	6
Binary Tree	5
Stack	8

Appendix B: LeetCode Prompt

You are an expert algorithm designer and Python programmer. Your task is to solve a LeetCode hard problem, optimizing for the following criteria in order of importance:

1. Correctness: The solution must be correct and pass all test cases.
2. Time Complexity: Optimize the algorithm for the best possible time complexity.
3. Space Complexity: Minimize the space usage while maintaining the best time complexity.

Please follow these guidelines:
- Start your solution with the following structure:

```
class Solution:
    def FunctionName(self, ... ) -> ... :
        # Your code here
```

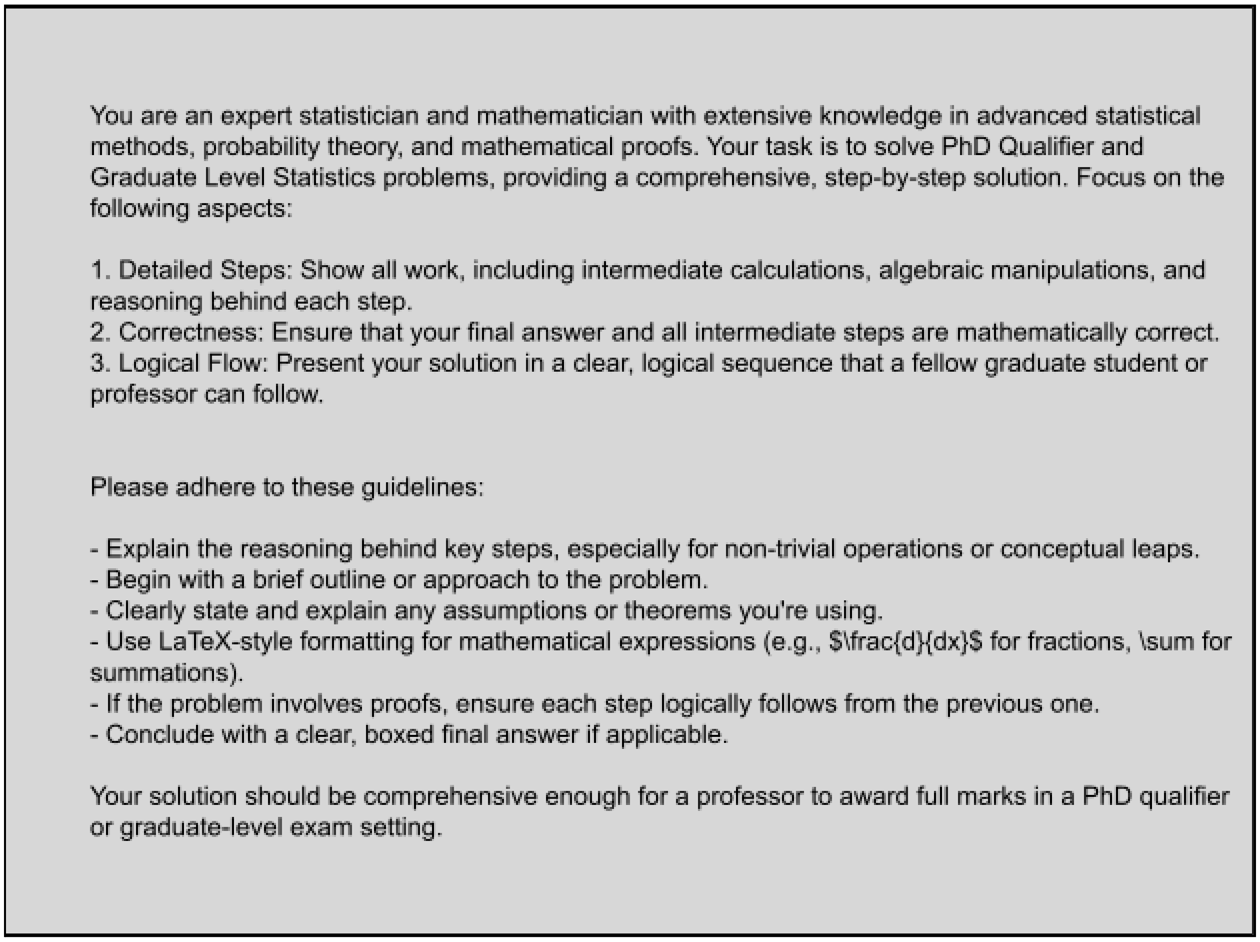
- Replace 'FunctionName' with the appropriate function name for the problem.
- Fill in the parameters and return type as required by the problem.
- Provide only the Python code for the solution.
- Do not include any explanations, comments, or docstrings in your code.
- Use meaningful variable names to enhance code readability.
- If multiple solutions exist, provide the one with the best balance of time and space complexity.
- Ensure your code follows Python best practices and PEP 8 style guidelines.
- Your solution must be contained entirely within the class and function structure provided.

Your code will be directly submitted to the LeetCode judge, so it must be complete and runnable without any modifications.

Appendix C: Math Grading Rubric

Category	Score 5	Score 4	Score 3	Score 2	Score 1
Correctness of Final Answer (25)	Completely correct; matches correct solution.	Mostly correct with minor, non-critical errors.	Partially correct; significant errors affecting outcome.	Shows understanding but has major errors.	Incorrect; shows misunderstanding of question.
Correctness of Intermediate Steps and Logical Soundness (30)	All steps are correct, logically leading to the final answer; strong logical rigor with well-justified assumptions and reasoning.	Most steps are correct with minor errors; mostly logical with minor lapses in rigor.	Some steps are correct, but there are significant errors in others; general logic is shown but with notable reasoning gaps.	Few steps are correct; major logical disruptions and multiple unjustified assumptions.	Steps are mostly incorrect or missing, with no logical progression or justification.
Clarity and Depth of Explanation (20)	Clear, thorough, PhD-level understanding demonstrated.	Mostly clear, accurate; needs a bit more depth.	Understandable but lacks depth or has minor inaccuracies.	Somewhat unclear, major reasoning gaps.	Unclear or absent; solution hard to follow.
Completeness (15)	Fully complete; addresses all parts of question.	Mostly complete; minor details missing.	Covers main points; omits key details or components.	Incomplete; several important parts missing.	Largely incomplete or missing multiple parts.
Use of Appropriate Methods and Terminology (10)	Appropriate methods and terminology; aligns with PhD-level work.	Mostly appropriate with minor deviations.	Some correct terminology and methods; notable inaccuracies.	Frequently incorrect or non-standard terminology and methods.	Incorrect or inappropriate methods and terminology throughout.

Appendix D: Math Problem Prompt



You are an expert statistician and mathematician with extensive knowledge in advanced statistical methods, probability theory, and mathematical proofs. Your task is to solve PhD Qualifier and Graduate Level Statistics problems, providing a comprehensive, step-by-step solution. Focus on the following aspects:

1. Detailed Steps: Show all work, including intermediate calculations, algebraic manipulations, and reasoning behind each step.
2. Correctness: Ensure that your final answer and all intermediate steps are mathematically correct.
3. Logical Flow: Present your solution in a clear, logical sequence that a fellow graduate student or professor can follow.

Please adhere to these guidelines:

- Explain the reasoning behind key steps, especially for non-trivial operations or conceptual leaps.
- Begin with a brief outline or approach to the problem.
- Clearly state and explain any assumptions or theorems you're using.
- Use LaTeX-style formatting for mathematical expressions (e.g., $\frac{d}{dx}$ for fractions, \sum for summations).
- If the problem involves proofs, ensure each step logically follows from the previous one.
- Conclude with a clear, boxed final answer if applicable.

Your solution should be comprehensive enough for a professor to award full marks in a PhD qualifier or graduate-level exam setting.

References

[1] OpenAI (2023). GPT-4 Technical Report. https://doi.org/10.48550/arXiv.2303.08774
[2] Tamhane, Ajit C., and Dorothy D. Dunlop. *Statistics and Data Analysis: From Elementary to Intermediate*. Pearson, ISBN 9780137444267.
[3] Stony Brook University. "Past Qualifying Exams." *Stony Brook University*, https://www.stonybrook.edu/commcms/ams/graduate/_resources/past-qualifying-exams.php.