TASK 3

1. Metric Design: Evaluation Framework

Metric Formula Score = (1/N) ∑level (Success Ratelevel × Weightlevel + Partial Successlevel × Weightlevel × 0.7)

- Success Rate: Fraction of problems solved at each level.
- Partial Success: Fraction of problem complexity reduced (e.g., string length reduction).
- Weights: Level 0 (1), Level 1 (2), ..., Impossible (6).
- Normalization: N = Total prompts tested.

Why This Metric?

- Weighted Difficulty: Harder levels matter more (e.g., "Impossible" has 6x the weight of Level 0).
- Partial Credit: Rewards incremental progress (70% of full credit).
- Normalization: Adjusts for imbalanced testing (Gemini tested on 46 prompts vs. Claude's 6).
- No Penalties: Ignores parsing errors (blames API, not model capability).

2. Application to Data

Raw Scores and Normalization

Model	Prompting Technique	Raw Score	Total Prompts (N)	Normalized Score
Claude	Zero-shot	~5.38*	6	0.9
ChatGPT	Chain-of-Thought	~3.84*	6	0.64
Gemini	Zero-shot	0.4	46	0.0087
Llama 2	CoT	0.0	2	0.0

*Assumed based on partial success (e.g., Level 2 reduction from 5→3 chars).

Baseline Comparison

- Random Performance: Assume a random agent solves 10% of problems and reduces complexity by 10% on average.
- Normalized Baseline Score: ~0.1 (weights applied).
- Result: Claude (0.9) and ChatGPT (0.64) beat baseline; Gemini/Llama 2 fail.

3. Hypothesis-Driven Analysis

Why Claude Dominates Strengths:

- Iterative Reasoning: Excels in tasks requiring multi-step reduction (e.g., 17-length → 9-length string).
- Training Data: Likely fine-tuned on structured problem-solving (e.g., coding, math).

Weaknesses:

- Limited Testing: Scores on "Impossible" tasks rely on 1 prompt—may not generalize.

Why Gemini Underperforms API Constraints:

- Verbose Outputs: Generated long responses, causing parsing failures (not model's fault).
- Scripted Prompts: Lack of interactive refinement (vs. human-tested ChatGPT/Claude).

Potential:

- With parsing fixes, Gemini might match Claude on harder levels.

ChatGPT's Mixed Results Chain-of-Thought (CoT):

- Worked for Level 0-2 (simple tasks) but failed on harder levels due to reasoning depth limits.
- Example: Reducing a 5-length string to 3 is achievable; reducing a 25-length string to 13 requires deeper recursion.

Lookahead Prompting:

- Improved Level 2 performance (40% success) but lacked consistency (fails on Level 3+).

Llama 2's Failure Compute Limitations:

- Ran locally with limited RAM. Struggled with long inference times.

Model Architecture:

- Smaller variants (7B) lack the capacity for iterative tasks compared to Claude (larger?).

4. Creative Insights and Improvements

Hypothesis 1: Task Complexity vs. Model Architecture

- Observation: Claude succeeds on tasks requiring iterative reduction; Gemini/ChatGPT fail.
- Hypothesis: Transformer models with sliding window attention (e.g., Claude) handle long-term dependencies better than vanilla transformers (Gemini).
- Experiment: Test models on iterative tasks with varying recursion depths.

Hypothesis 2: Prompting Techniques as a Band-Aid

- Observation: CoT and lookahead improved performance but not enough for hard tasks.
- Hypothesis: Models lack internal memory for multi-step reasoning—prompting can't compensate.
- Improvement: Augment with external memory (e.g., vector DB for intermediate steps).

Hypothesis 3: The "Partial Success Trap"

- Observation: Partial success (e.g., halving a string) doesn't guarantee full solution.
- Hypothesis: Models get "stuck" in local minima (good enough reductions) without global planning.
- Improvement: Hybrid approach: Use ChatGPT for ideation + Claude for refinement.

5. Surprise Factor: Task-Specific Fine-Tuning

Creative Experiment:

1. Use Claude to generate synthetic training data (problem → solution pairs).
2. Fine-tune Llama 2 on this data to specialize in reduction tasks.
3. Compare fine-tuned Llama 2 vs. raw Claude.

Tests if smaller models can match larger ones with task-specific training.

Note -> Will require access to a computer with a good GPU for running Llama 2 locally.

6. Conclusion

Key Findings:

- Model Architecture > Prompting: Claude's design inherently suits iterative tasks.
- API Limitations Skew Results: Gemini's parsing errors mask true potential.
- Partial Success ≠ Progress: Models need mechanisms to "plan ahead."

Future Tests:

- Fix parsing issues for Gemini and retest.
- Benchmark models on harder tasks with more prompts.
- Explore hybrid prompting (CoT + lookahead) for Level 3+ tasks.