Algorithmic Reasoning LLM Experiments on GSM8K

This notebook demonstrates how to compare different prompting strategies for solving math word problems from the GSM8K dataset using a large language model

We will explore six approaches:

- **No Prompt (just providing the question) : Provides only the question directly to the model without additional instructions or examples.**
- **Chain of Thought: Encourages the model to explicitly outline its step-by-step reasoning in the prompt.**
- **Algorithmic Structuring: Uses a structured set of instructions that guide the model through a predefined algorithmic flow.**
- **Tree/Graph of Thoughts: Guides the model to branch out into multiple solution paths or ideas, forming a tree or graph-like exploration.**
- **Decomposed Prompting: Breaks down the problem into smaller sub-questions or steps before synthesizing the final answer.**
- **Algorithmic Prompting: Provides clear, algorithm-like instructions (e.g., explicit arithmetic procedures) to ensure the model follows a strict rule-based approach.**

1. Setting Up Environment

Install and import necessary libraries

In [ ]:

```
!pip install datasets requests pandas matplotlib
```

```
Requirement already satisfied: datasets in /usr/local/lib/python3.11/dist-packages (3.5.1
)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (2.32.
3)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.1
0.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from
datasets) (3.18.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (fr
om datasets) (2.0.2)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11/dist-packages
(from datasets) (18.1.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.11/dist-packa
ges (from datasets) (0.3.8)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-packages (f
rom datasets) (4.67.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packages (from da
tasets) (3.5.0)
Requirement already satisfied: multiprocess<0.70.17 in /usr/local/lib/python3.11/dist-pac
kages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2025.3.0,>=2023.1.0 in /usr/local/lib/python3.11/d
ist-packages (from fsspec[http]<=2025.3.0,>=2023.1.0->datasets) (2025.3.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from d
atasets) (3.11.15)
Requirement already satisfied: huggingface-hub>=0.24.0 in /usr/local/lib/python3.11/dist-
packages (from datasets) (0.30.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from
datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (fr
om datasets) (6.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist
-packages (from requests) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (f
rom requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packa
ges (from requests) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packa
ges (from requests) (2025.1.31)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-p
ackages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (f
rom pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages
(from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-package
s (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (f
rom matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packag
es (from matplotlib) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packag
es (from matplotlib) (1.4.8)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from
matplotlib) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-package
s (from matplotlib) (3.2.3)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-
packages (from aiohttp->datasets) (2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-package
s (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (
from aiohttp->datasets) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packag
es (from aiohttp->datasets) (1.6.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-pack
ages (from aiohttp->datasets) (6.4.3)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-package
s (from aiohttp->datasets) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packag
es (from aiohttp->datasets) (1.20.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/di
st-packages (from huggingface-hub>=0.24.0->datasets) (4.13.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from
python-dateutil>=2.8.2->pandas) (1.17.0)
```

In [ ]:

```python
import requests              # For making API calls
import pandas as pd          # For data storage and manipulation
import matplotlib.pyplot as plt  # For plotting graphs
from datasets import load_dataset  # For loading the GSM8K dataset
import time  # For measuring API response time
import json
```

Adding token Limit

Max_tokens is a parameter that defines the maximum number of tokens the model is allowed to generate in its response.

If we dont keep a token limit then the model will extend the output even if it's not needed.

In relative to GSM8K dataset it contains grade-school level math word problems hence the average question is going to be 20 to 40 tokens

But since we are using prompting strategies it may vary,

Below is the estimate of token usage

Estimated Token Usage by Prompting Method

Prompting Method	Prompt Tokens	Response Tokens	Total Estimated Tokens

Prompting Method	Prompt Tokens	Response Tokens	Total Estimated Tokens
No Prompt	~30	~20	~50
Chain of Thought	~100–150	~150–250	~250–400
Algorithmic Structuring	~150–250	~250–300	~400–550
Tree/Graph of Thoughts	~200–300	~300–400	~500–700
Decomposed Prompting	~150–200	~250–350	~400–550
Algorithmic Prompting	~200	~200–300	~400–500

2: Data Loading and Preparation

Loading 200 problems from the GSM8K dataset so that we can use in our experiments.

In [ ]:

```
gsm8k = load_dataset("gsm8k", "main")
```

In [ ]:

```
print(gsm8k)
```

```
DatasetDict({
    train: Dataset({
        features: ['question', 'answer'],
        num_rows: 7473
    })
    test: Dataset({
        features: ['question', 'answer'],
        num_rows: 1319
    })
})
```

Inspecting the first item to identifty the format

In [ ]:

```
gsm8k_train = gsm8k["train"]
gsm8k_train[0]
```

Out[ ]:

```
{'question': 'Natalia sold clips to 48 of her friends in April, and then she sold half as
many clips in May. How many clips did Natalia sell altogether in April and May?',
 'answer': 'Natalia sold 48/2 = <<48/2=24>>24 clips in May.\nNatalia sold 48+24 = <<48+24
=72>>72 clips altogether in April and May.\n#### 72'}
```

Setting up 200 random problems for experiment from the dataset

In [ ]:

```
#Random selection and then selecting first 200
sampled_data = gsm8k_train.shuffle(seed=42).select(range(200))
```

In [ ]:

```
df_gsm8k = pd.DataFrame(sampled_data)
df_gsm8k.head()
```

Out[ ]:

	question	answer
0	Mimi picked up 2 dozen seashells on the beach....	Mimi has 2 x 12 = <<2*12=24>>24 sea shells.\nK...
1	Frankie's parents let him have many pets. He h...	He has 6 - 2 = <<6-2=4>>4 cats.\nHe has 4 - 1 ...
2	Olaf collects colorful toy cars. At first, his...	Dad gave Olaf 10 toy cars,\nMom has given Olaf...
		She spend $56 because 7 x 8

	question	answer
3	Emma's bank account has $100 in it. Each day o...	She spend $56 because 7 x 8 = <<7*8=56>>56\nSh...
4	Ezekiel hikes as a hobby. This past summer, he...	After the first day, Ezekiel had 50 - 10 = <<5...

Looking at the format, in later stages of the code we will have to trim the answer/response to just get the number for testing purposes for cleaner format

3: Prompt Definition

It is important to define the prompt correctly since the performance depends on how we ask the question

We will define function for each prompting technique and return a prompt string that will be sent to the model.

3.1 No Prompt

Raw question with no formatting or guidance.

In [ ]:

```
def build_prompt_no_prompt(question):
    return question.strip()
```

3.2 Chain of Thought

Encouraging the model to “think out loud” step by step.

In [ ]:

```
def build_prompt_chain_of_thought(question):
    return f"""You are a helpful assistant solving math problems step by step.

Problem: {question}
Let's think step by step."""
```

3.3 Algorithmic Structuring

Give explicit, structured steps like an algorithm, even if it’s a word problem.

In [ ]:

```
def build_prompt_algorithmic_structuring(question):
    return f"""Solve the following problem by following these steps:
1. Identify the known quantities.
2. Identify what needs to be solved.
3. Apply the appropriate mathematical operations.
4. Show each step clearly.
5. Provide the final answer.

Problem: {question}
Solution:"""
```

3.4 Tree/Graph of Thoughts

Encouraging branching exploration, multiple reasoning paths. Not all models support true "graph reasoning".

In [ ]:

```
def build_prompt_tree_of_thoughts(question):
```

```
def build_prompt_tree_of_thoughts(question):
    return f"""You're an expert reasoner. For this problem, explore multiple possible solution paths (like a decision tree), then converge on the correct answer.

Problem: {question}

Path 1:
Path 2:
Best Answer:"""
```

3.5 Decomposed Prompting

Breaking the problem into simpler sub-problems or questions.

In [ ]:

```
def build_prompt_decomposed_prompting(question):
    return f"""Break down this problem into smaller sub-questions. Solve each one before combining them into a final answer.

Problem: {question}
Sub-question 1:
Sub-question 2:
Final Answer:"""
```

3.6 Algorithmic Prompting

Algorithm-style instructions, especially for numeric tasks (e.g., addition, logic).

In [ ]:

```
def build_prompt_algorithmic_prompting(question):
    return f"""You are a calculator that solves problems by following exact algorithms.

Step-by-step, solve the following problem:
{question}

Algorithm Output:"""
```

Making a dictonary to access the right function by name later.

In [ ]:

```
prompt_builders = {
    "no_prompt": build_prompt_no_prompt,
    "chain_of_thought": build_prompt_chain_of_thought,
    "algorithmic_structuring": build_prompt_algorithmic_structuring,
    "tree_of_thoughts": build_prompt_tree_of_thoughts,
    "decomposed_prompting": build_prompt_decomposed_prompting,
    "algorithmic_prompting": build_prompt_algorithmic_prompting
}
```

To make sure those functions work as expected and each prompt is being constructed properly we will print the actual generated prompt text for a real question before sending it to the model

In [ ]:

```
sample_question = df_gsm8k.iloc[0]["question"]

for method_name, builder in prompt_builders.items():
    print(f"\n--- {method_name.upper()} ---\n")
    print(builder(sample_question))
```

```
--- NO_PROMPT ---
```

```
Mimi picked up 2 dozen seashells on the beach.  Kyle found twice as many shells as Mimi a
nd put them in his pocket. Leigh grabbed one-third of the shells that Kyle found.  How ma
ny seashells did Leigh have?

--- CHAIN_OF_THOUGHT ---

You are a helpful assistant solving math problems step by step.

Problem: Mimi picked up 2 dozen seashells on the beach.  Kyle found twice as many shells
as Mimi and put them in his pocket. Leigh grabbed one-third of the shells that Kyle found
.  How many seashells did Leigh have?
Let's think step by step.

--- ALGORITHMIC_STRUCTURING ---

Solve the following problem by following these steps:
1. Identify the known quantities.
2. Identify what needs to be solved.
3. Apply the appropriate mathematical operations.
4. Show each step clearly.
5. Provide the final answer.

Problem: Mimi picked up 2 dozen seashells on the beach.  Kyle found twice as many shells
as Mimi and put them in his pocket. Leigh grabbed one-third of the shells that Kyle found
.  How many seashells did Leigh have?
Solution:

--- TREE_OF_THOUGHTS ---

You're an expert reasoner. For this problem, explore multiple possible solution paths (li
ke a decision tree), then converge on the correct answer.

Problem: Mimi picked up 2 dozen seashells on the beach.  Kyle found twice as many shells
as Mimi and put them in his pocket. Leigh grabbed one-third of the shells that Kyle found
.  How many seashells did Leigh have?

Path 1:
Path 2:
Best Answer:

--- DECOMPOSED_PROMPTING ---

Break down this problem into smaller sub-questions. Solve each one before combining them
into a final answer.

Problem: Mimi picked up 2 dozen seashells on the beach.  Kyle found twice as many shells
as Mimi and put them in his pocket. Leigh grabbed one-third of the shells that Kyle found
.  How many seashells did Leigh have?
Sub-question 1:
Sub-question 2:
Final Answer:

--- ALGORITHMIC_PROMPTING ---

You are a calculator that solves problems by following exact algorithms.

Step-by-step, solve the following problem:
Mimi picked up 2 dozen seashells on the beach.  Kyle found twice as many shells as Mimi a
nd put them in his pocket. Leigh grabbed one-third of the shells that Kyle found.  How ma
ny seashells did Leigh have?

Algorithm Output:
```

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

> **Add blockquote**

4: Hugging Face API Integration

Connecting the notebook to Hugging Face Inference API (DeepSeek V3-0324) to send generated prompts to the model

In [ ]:

```
# ============================================
# Step 4: Hugging Face Router API Setup (Optimized - DeepSeek V3-0324)
# ============================================

HF_API_URL = "https://router.huggingface.co/novita/v3/openai/chat/completions"
HF_API_KEY = "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"

headers = {
    "Authorization": f"Bearer {HF_API_KEY}",
    "Content-Type": "application/json"
}

def query_huggingface_router(prompt, max_tokens=256, temperature=0.7):
    """
    Sends a prompt to Hugging Face Router API (DeepSeek V3-0324) and returns the generated response.
    """
    payload = {
        "model": "deepseek/deepseek-v3-0324",
        "messages": [
            {
                "role": "user",
                "content": prompt
            }
        ],
        "max_tokens": max_tokens,
        "temperature": temperature
    }

    try:
        start_time = time.time()
        response = requests.post(HF_API_URL, headers=headers, data=json.dumps(payload))
        elapsed_time = time.time() - start_time

        response.raise_for_status()

        result = response.json()
        generated_text = result['choices'][0]['message']['content']

        return {
            "response": generated_text.strip(),
            "time": elapsed_time
        }

    except Exception as e:
        print(f"Error during API call: {e}")
        return {
            "response": None,
            "time": None
        }
```

Step 5: Parallel Batch Inference and Response Collection

Sending each question with a particular prompting technique to generate a response and extract the final answer, and save the results to a CSV file — all done in parallel to speed things up.

In [ ]:

```
# ============================================
# Step 5: Parallel Batch Inference and Data Collection (Questions)
# ============================================

import concurrent.futures
```

```
# Helper to extract GSM8K ground truth final answer
def extract_gsm8k_answer(answer_string):
    if "####" in answer_string:
        return answer_string.split("####")[-1].strip()
    else:
        return answer_string.strip()

results = []

# Function to process a single question
def process_single_question(method_name, builder_function, question, true_answer, idx):
    full_prompt = builder_function(question)

    # Show full prompt being sent
    print(f"□ [{method_name}] - Processing Q{idx+1}")
    print("□ Prompt Preview:\n", full_prompt[:500])

    response = query_huggingface_router(full_prompt)
    model_response = response["response"]
    response_time = response["time"]

    if model_response:
        print(f"□ Response Received. (Time taken: {round(response_time, 2)} sec)\n")
    else:
        print(f"⚠ No response for Q{idx+1}\n")

    return {
        "method": method_name,
        "question": question,
        "prompt_sent": full_prompt,
        "model_response": model_response,
        "true_answer": true_answer,
        "response_time": response_time
    }

# Loop over each prompting technique
for method_name, builder_function in prompt_builders.items():
    print(f"\n□ Starting method: {method_name.upper()}...\n")

    question_list = []
    true_answer_list = []

    for idx, row in df_gsm8k.head(200).iterrows():  # head(5) for quick test
        question_list.append((method_name, builder_function, row["question"], extract_gs
m8k_answer(row["answer"]), idx))

    # Use ThreadPoolExecutor to parallelize calls
    with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
        future_to_question = {executor.submit(process_single_question, *q): q for q in q
uestion_list}

        for future in concurrent.futures.as_completed(future_to_question):
            result = future.result()
            results.append(result)

# Save results
final_df = pd.DataFrame(results)
final_df.to_csv("/content/gsm8k_test_results.csv", index=False)
print("\n□ All results saved to /content/gsm8k_test_results.csv successfully!")
```

```
Streaming output truncated to the last 5000 lines.

Problem: Telegraph Road goes through multiple states and is 162 kilometers long. Pardee R
oad is 12000 meters long. How many kilometers longer is Telegraph Road than Pardee Road?

Path 1:
Path 2:
Best Answer:
□ Response Received. (Time taken: 10.43 sec)

□ [tree_of_thoughts] - Processing Q103
□ Prompt Preview:
 You're an expert reasoner. For this problem, explore multiple possible solution paths (l
ike a decision tree), then converge on the correct answer.

Problem: Alison has half as much money as Brittany. Brittany has 4 times as much money as
Brooke. Brooke has twice as much money as Kent. If Kent has $1,000, how much money does A
lison have?

Path 1:
Path 2:
Best Answer:
□ Response Received. (Time taken: 10.89 sec)

□ [tree_of_thoughts] - Processing Q104
□ Prompt Preview:
 You're an expert reasoner. For this problem, explore multiple possible solution paths (l
ike a decision tree), then converge on the correct answer.

Problem: Simon has 20% more legos than Bruce, who has 20 more than Kent.  If Kent has 40
legos, how many does Simon have?

Path 1:
Path 2:
Best Answer:
□ Response Received. (Time taken: 11.01 sec)

□ [tree_of_thoughts] - Processing Q105
□ Prompt Preview:
 You're an expert reasoner. For this problem, explore multiple possible solution paths (l
ike a decision tree), then converge on the correct answer.

Problem: Layla is training for a bicycle race. She rode her bicycle slowly to the high sc
hool, rode four miles around the running track at her faster, race pace, and then rode ba
ck home, slowly, on the same route. If her total mileage was 10 miles, how far is it, in
miles, from Layla's house to the high school?

Path 1:
Path 2:
Best Answer:
□ Response Received. (Time taken: 11.48 sec)

□ [tree_of_thoughts] - Processing Q106
□ Prompt Preview:
 You're an expert reasoner. For this problem, explore multiple possible solution paths (l
ike a decision tree), then converge on the correct answer.

Problem: Rebecca bought 22 items for her camping trip, including tent stakes, packets of
drink mix, and bottles of water.  She bought 3 times as many packets of drink mix as tent
stakes.  She also bought 2 more bottles of water than she did tent stakes.  How many tent
stakes did she buy?

Path 1:
Path 2:
Best Answer:
□ Response Received. (Time taken: 11.84 sec)

□ [tree_of_thoughts] - Processing Q107
□ Prompt Preview:
 You're an expert reasoner. For this problem, explore multiple possible solution paths (l
ike a decision tree), then converge on the correct answer.

Problem: A porcelain vase was originally priced at $200 but went on sale for 25% off. If
Donna bought the porcelain vase and paid 10% sales tax, how much did she pay in total?

Path 1:
Path 2:
Best Answer:
□ Response Received. (Time taken: 11.62 sec)

□ [tree_of_thoughts] - Processing Q108
□ Prompt Preview:
 You're an expert reasoner. For this problem, explore multiple possible solution paths (l
 You are a calculator that solves problems by following exact algorithms.

Step-by-step, solve the following problem:
Every 4 weeks, Helen hand washes her silk pillowcases.  It takes 30 minutes to hand wash
all of them.  In 1 year, how much time does Helen spend hand washing her pillowcases?

Algorithm Output:
□ Response Received. (Time taken: 7.92 sec)

□ [algorithmic_prompting] - Processing Q200
□ Prompt Preview:
 You are a calculator that solves problems by following exact algorithms.

Step-by-step, solve the following problem:
On our last vacation, I bought 4 times as many tetras as Tim bought clowns. Tim bought tw
ice as many clowns as Rick bought guppies. If Rick bought 30 guppies, how many animals di
d we buy on our last vacation?

Algorithm Output:
□ Response Received. (Time taken: 7.07 sec)

□ Response Received. (Time taken: 5.96 sec)

□ Response Received. (Time taken: 7.33 sec)

□ Response Received. (Time taken: 8.4 sec)

□ Response Received. (Time taken: 6.95 sec)

□ Response Received. (Time taken: 10.38 sec)

□ Response Received. (Time taken: 10.84 sec)

□ Response Received. (Time taken: 10.29 sec)

□ Response Received. (Time taken: 9.71 sec)

□ Response Received. (Time taken: 12.07 sec)


□ All results saved to /content/gsm8k_test_results.csv successfully!
```

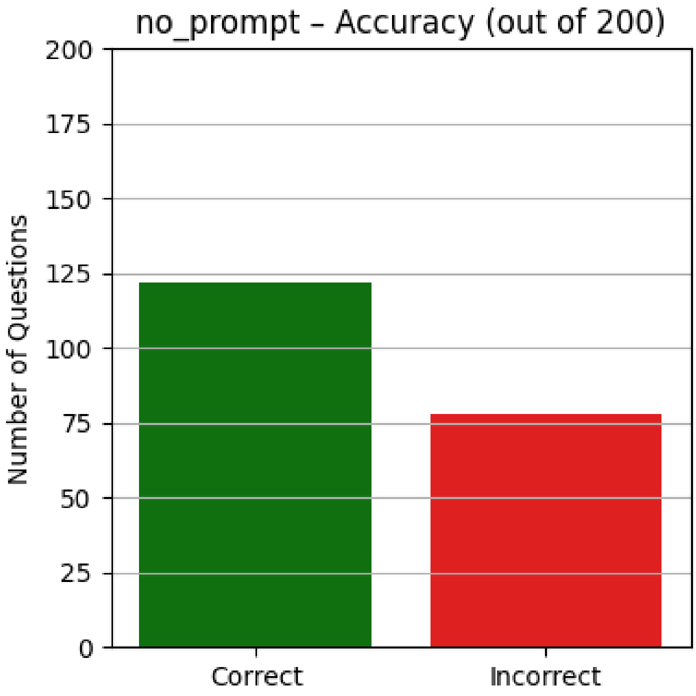
Step 6: Accuracy Evaluation and Visualization

This step compares the model's predicted answers with the actual GSM8K answers for each prompting technique, calculates accuracy, and visualizes the results using bar charts.

In [ ]:

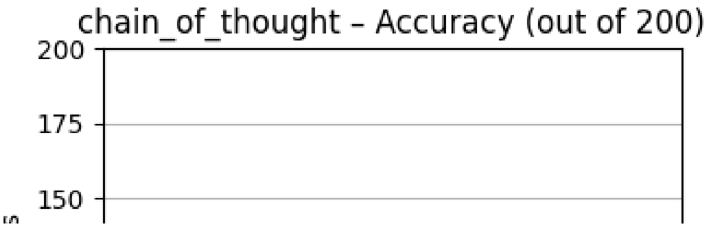
```python
# ============================================
# Visualize Per-Method Correct vs Incorrect Counts
# ============================================

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re

# Reload the dataset (if needed)
df = pd.read_csv("/content/gsm8k_test_results.csv")  # change path if needed

# Helper to extract final number from text
def extract_numeric_answer(text):
    if pd.isna(text):
        return None
    numbers = re.findall(r"\d+(?:\.\d+)?", str(text))
    return numbers[-1] if numbers else None

# Clean and compare
df["predicted_numeric"] = df["model_response"].apply(lambda x: extract_numeric_answer(st
r(x)))
df["true_numeric"] = df["true_answer"].apply(lambda x: extract_numeric_answer(str(x)))
df["correct"] = df.apply(lambda row: str(row["predicted_numeric"]) == str(row["true_nume
ric"]), axis=1)

# List of prompting techniques
prompting_methods = df["method"].unique()

# Plot bar graph for each method
for method in prompting_methods:
    method_df = df[df["method"] == method]
    correct_count = method_df["correct"].sum()
    incorrect_count = len(method_df) - correct_count

    # Bar plot for the method
    plt.figure(figsize=(4, 4))
    sns.barplot(x=["Correct", "Incorrect"], y=[correct_count, incorrect_count], palette=
["green", "red"])
    plt.title(f"{method} - Accuracy (out of {len(method_df)})")
    plt.ylabel("Number of Questions")
    plt.ylim(0, len(method_df))
    plt.tight_layout()
    plt.grid(axis="y")
    plt.show()
```

```
<ipython-input-354-6384c675f08e>:36: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=["Correct", "Incorrect"], y=[correct_count, incorrect_count], palette=["g
reen", "red"])
```

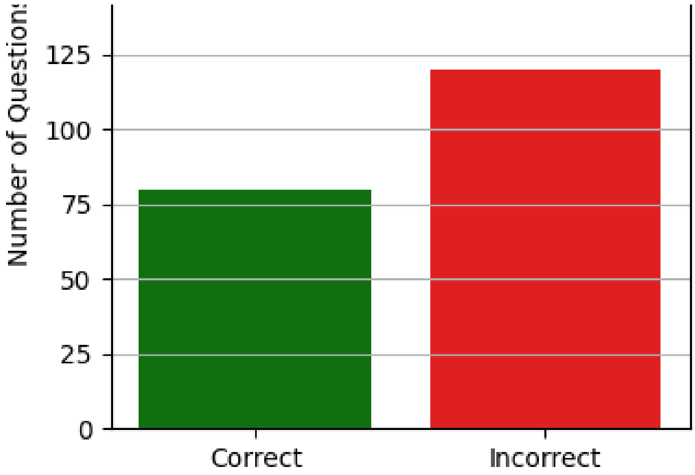


no_prompt - Accuracy (out of 200)

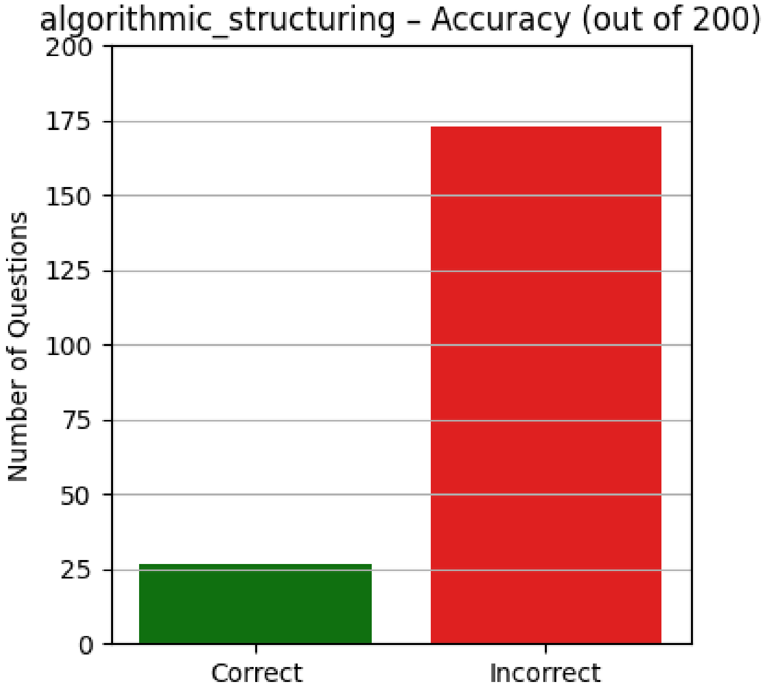
```
<ipython-input-354-6384c675f08e>:36: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=["Correct", "Incorrect"], y=[correct_count, incorrect_count], palette=["g
reen", "red"])
```



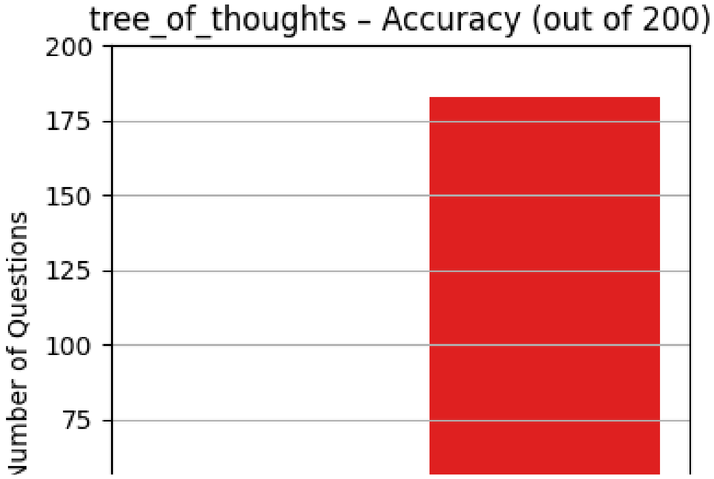
chain_of_thought - Accuracy (out of 200)

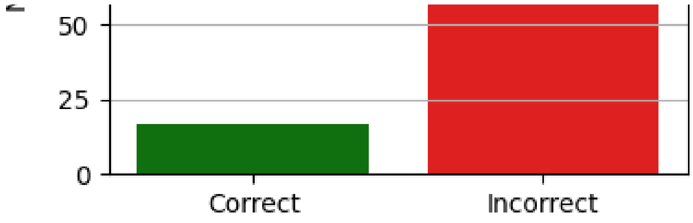


```
<ipython-input-354-6384c675f08e>:36: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=["Correct", "Incorrect"], y=[correct_count, incorrect_count], palette=["g
reen", "red"])
```

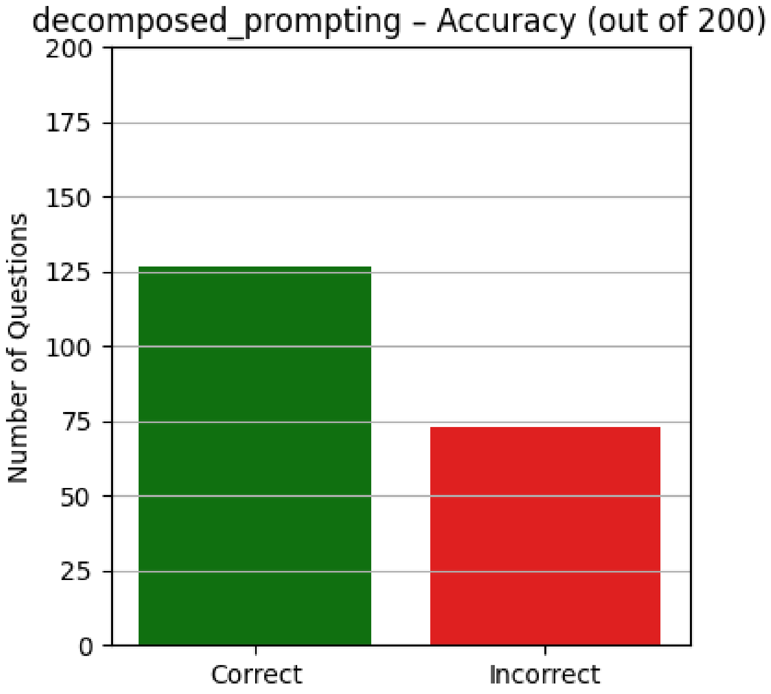

algorithmic_structuring – Accuracy (out of 200)

```
<ipython-input-354-6384c675f08e>:36: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=["Correct", "Incorrect"], y=[correct_count, incorrect_count], palette=["g
reen", "red"])
```

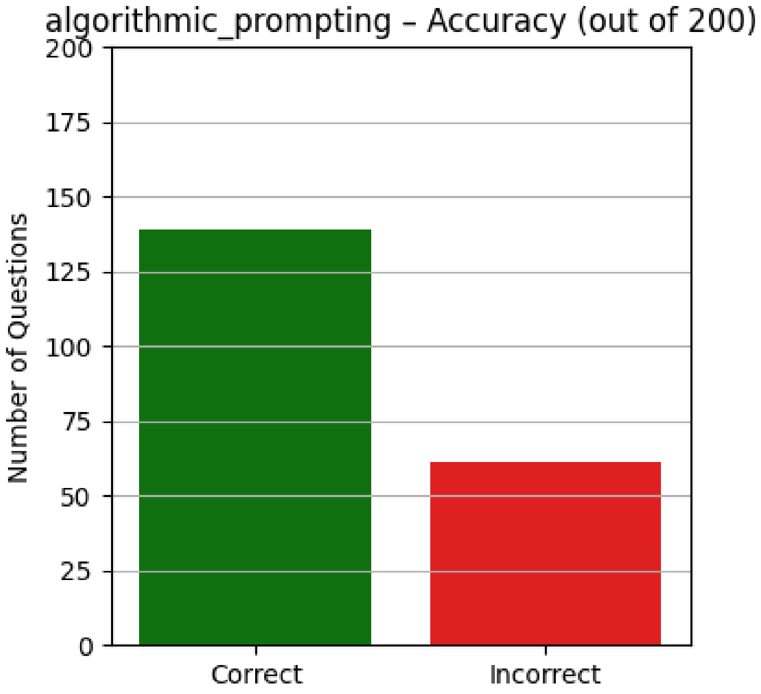

tree_of_thoughts – Accuracy (out of 200)



```
<ipython-input-354-6384c675f08e>:36: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=["Correct", "Incorrect"], y=[correct_count, incorrect_count], palette=["g
reen", "red"])
```



```
<ipython-input-354-6384c675f08e>:36: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=["Correct", "Incorrect"], y=[correct_count, incorrect_count], palette=["g
reen", "red"])
```



In [ ]:

```
# ============================================
# Display Model Outputs vs True Answers (Side-by-Side)
# ============================================

import pandas as pd
import re

# Load the results CSV
df = pd.read_csv("/content/gsm8k_test_results.csv")  # change path if needed

# Helper: Extract numeric answers
def extract_numeric_answer(text):
    if pd.isna(text):
        return None
    numbers = re.findall(r"\d+(?:\.\d+)?", str(text))
    return numbers[-1] if numbers else None

# Apply numeric extraction
df["predicted_numeric"] = df["model_response"].apply(lambda x: extract_numeric_answer(st
r(x)))
df["true_numeric"] = df["true_answer"].apply(lambda x: extract_numeric_answer(str(x)))

# Compare prediction to ground truth
df["correct"] = df.apply(lambda row: str(row["predicted_numeric"]) == str(row["true_nume
ric"]), axis=1)

# Columns to view
view_cols = ["method", "question", "model_response", "true_answer", "predicted_numeric",
"true_numeric", "correct"]

# Create summary DataFrame for inspection
summary_table = df[view_cols].copy()

# Display full table
summary_table
```

Out[ ]:

	method	question	model_response	true_answer	predicted_numeric	true_numeric	correct
0	no_prompt	Don throws 3 darts. One is a bullseye worth 5...	Let's calculate Don's final score step by step...	75	75	75	True
1	no_prompt	Mimi picked up 2 dozen seashells on the beach....	Let's break down the problem step by step to f...	16	16	16	True
2	no_prompt	Ezekiel hikes as a hobby. This past summer, he...	Let's break down Ezekiel's three-day hike step...	15	15	15	True
3	no_prompt	Olaf collects colorful toy cars. At first, his...	Let's break down the problem step by step to f...	196	2	196	False
4	no_prompt	TreQuan is throwing rocks in the river and he ...	To determine the total width of the splashes T...	7	4	7	False
...	...	...	...	...	...	...	...
1195	algorithmic_prompting	Annie has some money. The restaurant next door...	Let's solve the problem step by step.\n\n1. **...	132	132	132	True
1196	algorithmic_prompting	Ruth prepared sandwiches. She ate 1 sandwich a...	Let's solve the problem step by step.\n\n1. **...	10	10	10	True
1197	algorithmic_prompting	Joseph and his friends watched	Let's solve the problem step by	4	20	4	False

	method	two movies in h question	step.\n\n1. ** model_response	true_answer	predicted_numeric	true_numeric	correct
1198	algorithmic_prompting	Every 4 weeks, Helen hand washes her silk pill...	To determine how much time Helen spends hand w...	390	390	390	True
1199	algorithmic_prompting	On our last vacation, I bought 4 times as many...	Let's solve the problem step by step.\n\n1. **...	330	330	330	True

1200 rows × 7 columns

In [ ]:

```python
# prompt: Make a very consice table which only shows Question as Q1 or Q2 or Q3 and then the propmting techniqe, Expected answer, actual answer just the number from the reponsce

import pandas as pd

# Load the results CSV
df = pd.read_csv("/content/gsm8k_test_results.csv")

# Helper: Extract numeric answers
def extract_numeric_answer(text):
    if pd.isna(text):
        return None
    numbers = re.findall(r"\d+(?:\.\d+)?", str(text))
    return numbers[-1] if numbers else None

# Apply numeric extraction
df["predicted_numeric"] = df["model_response"].apply(lambda x: extract_numeric_answer(str(x)))
df["true_numeric"] = df["true_answer"].apply(lambda x: extract_numeric_answer(str(x)))

# Create a concise table
concise_table = pd.DataFrame()
concise_table["Question"] = ["Q" + str(i+1) for i in range(len(df))]
concise_table["Prompting Technique"] = df["method"]
concise_table["Expected Answer"] = df["true_numeric"]
concise_table["Actual Answer"] = df["predicted_numeric"]

concise_table
```

Out[ ]:

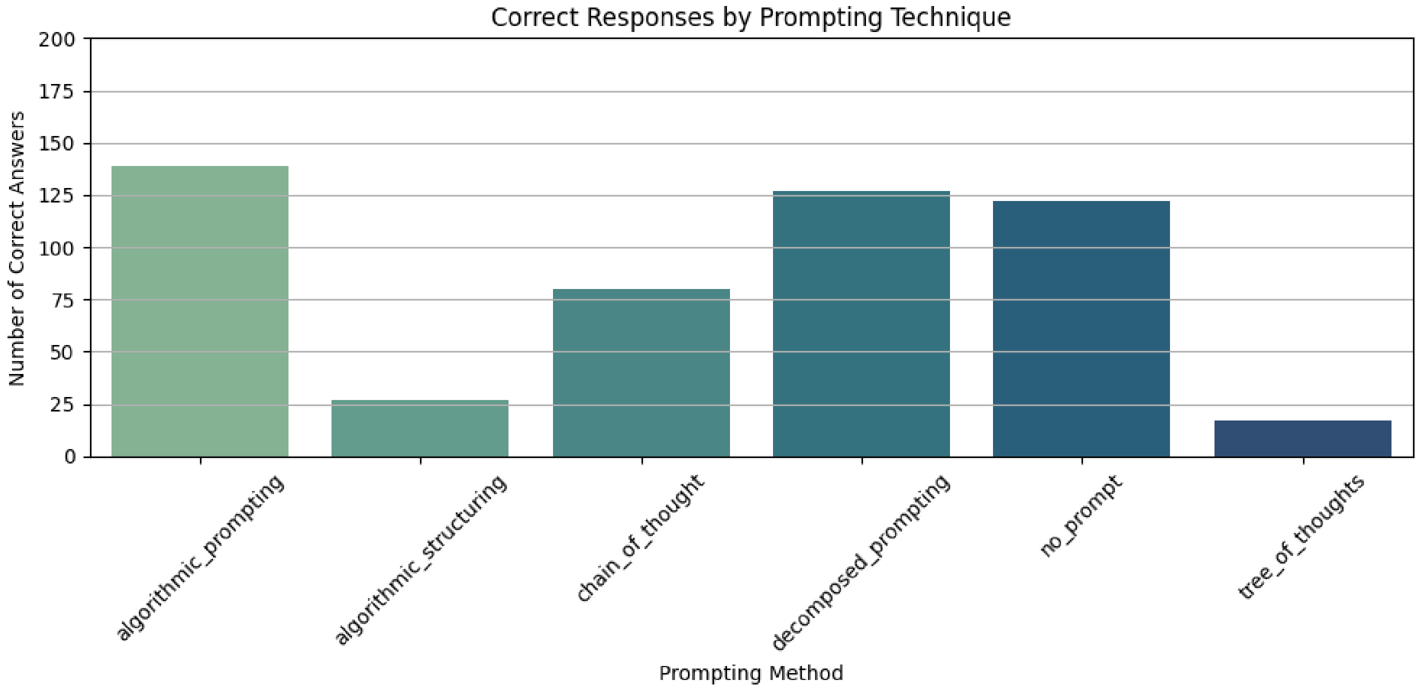
	Question	Prompting Technique	Expected Answer	Actual Answer
0	Q1	no_prompt	75	75
1	Q2	no_prompt	16	16
2	Q3	no_prompt	15	15
3	Q4	no_prompt	196	2
4	Q5	no_prompt	7	4
...	...	...	...	...
1195	Q1196	algorithmic_prompting	132	132
1196	Q1197	algorithmic_prompting	10	10
1197	Q1198	algorithmic_prompting	4	20
1198	Q1199	algorithmic_prompting	390	390
1199	Q1200	algorithmic_prompting	330	330

1200 rows × 4 columns

In [ ]:

```python
# ============================================
# Accuracy Comparison (Absolute Count Scale)
# ============================================

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re

# Load the results CSV (adjust if needed)
df = pd.read_csv("/content/gsm8k_test_results.csv")

# Helper: Extract final number from response
def extract_numeric_answer(text):
    if pd.isna(text):
        return None
    numbers = re.findall(r"\d+(?:\.\d+)?", str(text))
    return numbers[-1] if numbers else None

# Extract predicted and true numeric answers
df["predicted_numeric"] = df["model_response"].apply(lambda x: extract_numeric_answer(st
r(x)))
df["true_numeric"] = df["true_answer"].apply(lambda x: extract_numeric_answer(str(x)))
df["correct"] = df.apply(lambda row: str(row["predicted_numeric"]) == str(row["true_nume
ric"]), axis=1)

# Count correct responses per method
correct_counts = df.groupby("method")["correct"].sum().reset_index()
correct_counts.columns = ["Prompting Method", "Correct Count"]

# Plot
plt.figure(figsize=(10, 5))
sns.barplot(data=correct_counts, x="Prompting Method", y="Correct Count", palette="crest
")
plt.title("Correct Responses by Prompting Technique")
plt.ylabel("Number of Correct Answers")
plt.xticks(rotation=45)
plt.ylim(0, 200)  # since  tested 20 questions per method
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

```
<ipython-input-357-68453f3982a9>:31: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. A
ssign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=correct_counts, x="Prompting Method", y="Correct Count", palette="cres
t")
```



In [ ]: