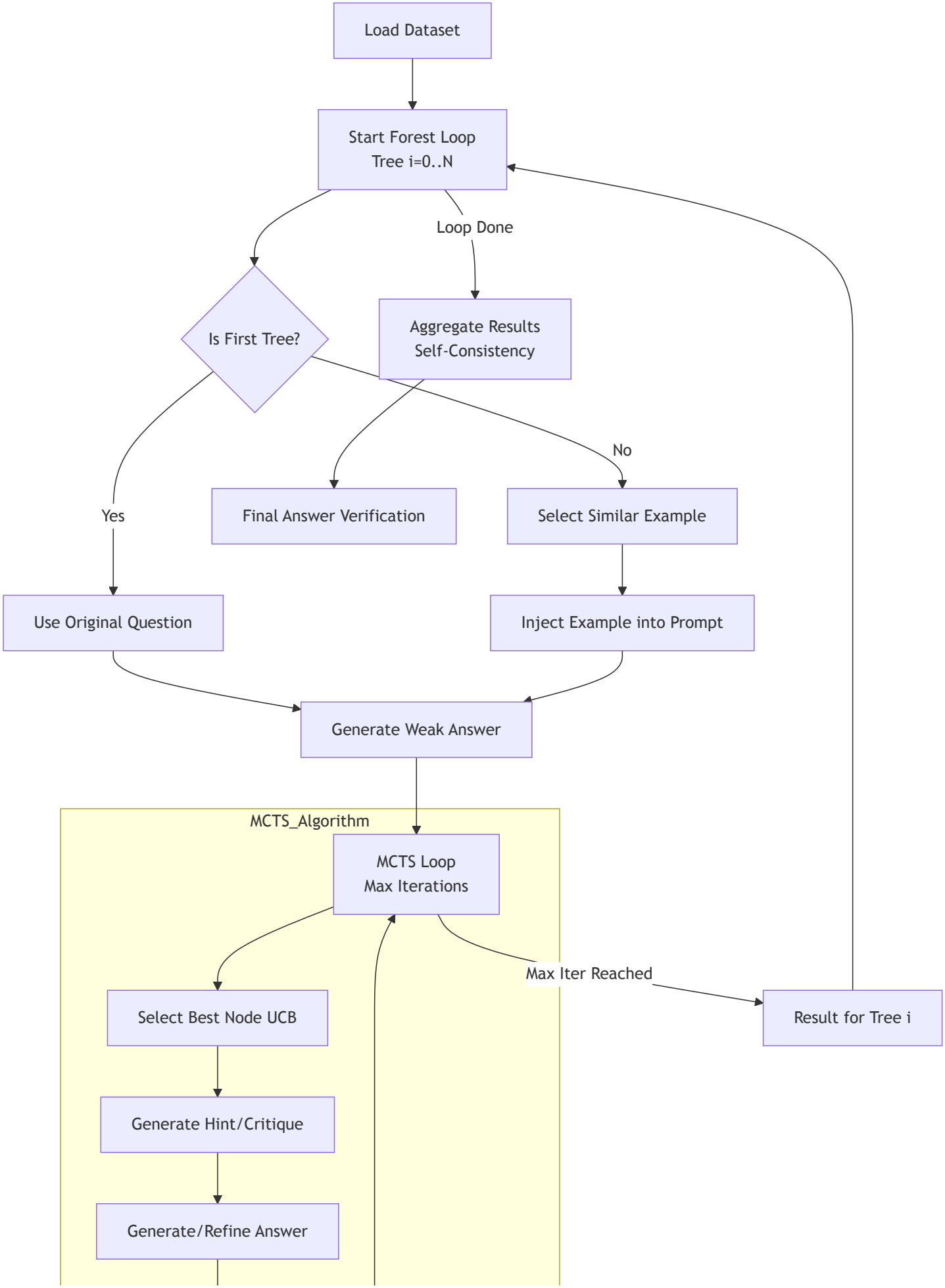
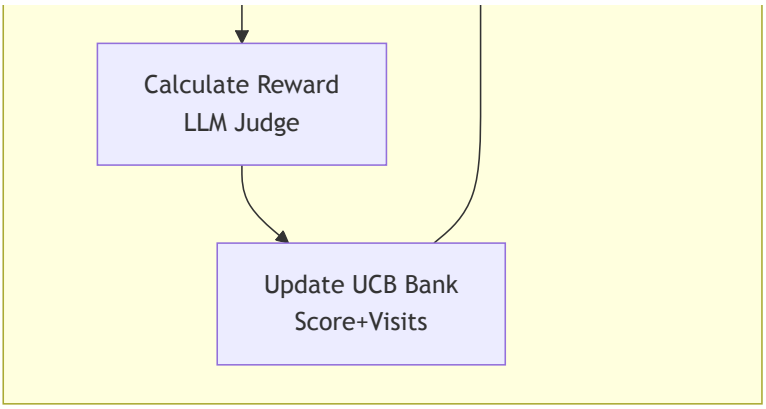
# Forest-of-Thought: Detailed Workflow

This document traces the complete execution flow of the MCTS algorithm, showing each step, function call, file location, and the exact prompts sent to the LLM.

# Overview


Load Dataset
Start Forest Loop
Tree i=0..N
Loop Done
Is First Tree?
Aggregate Results
Self-Consistency
Yes
No
Final Answer Verification
Select Similar Example
Use Original Question
Inject Example into Prompt
Generate Weak Answer
MCTS_Algorithm
MCTS Loop
Max Iterations
Max Iter Reached
Select Best Node UCB
Result for Tree i
Generate Hint/Critique
Generate/Refine Answer




# Step-by-Step Execution

## Step 1: Load Dataset

**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Line ~628)

```
dataset = mcts_load_data(args)
# Loads: datasets/gsm8k/test.parquet
```

**Example Problem**:

```
{
  "question": "Janet's ducks lay 16 eggs per day...",
  "answer": "#### 18"
}
```

## Step 2: The Forest Loop (Input Diversity)

**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Inside [run](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py#249-311) method of [Monte_Carlo_Forest](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf_stop_noearly.py#314-600), Lines ~355-365)

The system runs **multiple trees** ( `tree_nums` , e.g., 5) for the *same* problem to explore different reasoning paths.

1. **Tree 0**: Uses the original question only.

2. **Tree 1+**: Injects a "similar" solved example from the bank to guide the model's reasoning style (Input Diversity).

```
# run_with_mcf.py
for t in range(self.tree_nums):
    if t > 0:
        # Find closest example from utils/examples.py
        # Uses TF-IDF + Cosine Similarity
        case = self.learning_cases[query_index]
        # Modifies prompt: "Question: [Example] Answer: [Example] ... Question: [Target]"
        query = ...

    # Run independent MCTS for this tree
    monte_carlo_tree(...)
```

**Helper Function**: [get_similarity_question](file:///Users/admin/Documents/Forest-of-Thought/utils/examples.py#217-234)
**File**: [utils/examples.py](file:///Users/admin/Documents/Forest-of-Thought/utils/examples.py) (Lines 217-234)

- **Logic**: Uses TF-IDF and Cosine Similarity to find the "closest" example in the bank.
- **Key Behavior**: Even if no good match exists, it *always* returns the mathematical nearest neighbor to ensure the prompt is perturbed, adding diversity.

## Step 3: Generate Initial "Weak" Answer

**Function**: [get_weak_answer()](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py#62-68)
**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Lines 64-67)

**Prompt Sent to LLM**:

```
Question: Janet's ducks lay 16 eggs per day...

The response should begin with [reasoning process]...[Verification]... and end with "[Fin
#### [answer]"

Let's think step by step.
```

**LLM Response (Example)**:

```
[reasoning process]
Janet starts with 16 eggs...
9 × $2 = $18

[Verification]
9 × 2 = 18 ✓

[Final Answer] The answer is 18
#### 18
```

**Score**: [cal_reward()](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf_stop_noearly.py#35-50) is called immediately after to score this initial node (see Step 4.4).

# Step 4: MCTS Iteration Loop

**Function**: [monte_carlo_tree](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf_stop_noearly.py#223-313)
**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Lines ~130-180)

**For each iteration** (0 to `max_iter-1`):

## 4.1: Selection - Pick Best Node Using UCB

**Function**: [get_best_explore_from_ucb()](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py#89-101)
**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Lines ~388-406, defined outside class or as helper)

```
# Calculate UCB for each node:
UCB = reward + 1.4 × √(log(parent_visits) / node_visits)
# Balances exploitation (high reward) and exploration (low visits)
# Returns: The best node to expand
```

## 4.2: Generate Hint/Critique

**Function**: [get_weak_hints()](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py#69-72)
**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Lines 55-57)

**Prompt Sent to LLM**:

```
Question: [Question]
Answer: [Selected Answer]

Since we have a weak Answer, could you provide me with a reflection or feedback to correc

Let's think step by step.
```

**LLM Response (Hint)**:

```
The calculation looks correct, but let me verify each step more carefully...
```

## 4.3: Generate Better Answer

**Function**: [get_better_answer()](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf_stop_noearly.py#59-62)
**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Lines 59-61)

**Prompt Sent to LLM** (includes conversation history with hint):

```
[Previous conversation history including the hint above]

Question: [Question]

Please refine your answer according to your Reflection or Feedback. The response should b

Let's think step by step.
```

**LLM Response (Refined Answer)**:

```
[reasoning process]
Let me solve this step-by-step with clear order of operations...
[Final Answer] The answer is 18
#### 18
```

### 4.4: Calculate Reward Score

**Function**: [cal_reward()](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf_stop_noearly.py#35-50)
**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Lines 35-49)

**Prompt Sent to LLM**:

```
Question: [Question]
Answer: [Refined Answer]

Analyze this Answer Strictly and Critic...
Output a score between [0,100], i.e. from 0 to 100.
```

**LLM Response**: `[Analyst]...[Score] 95/100` -> Extracted as **95**

### 4.5: Update UCB Values

**Function**: [update_ucb()](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py#105-143)
**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Lines ~408-433)

```
# Update visits and rewards for the node and its ancestors
# Recalculate UCB scores for future selections
```

## Step 5: Select Best Answer from Tree

**Function**: [get_tree_ans](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf_stop_noearly.py#212-222) / Weighted Score Logic
**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Lines 212-221)

After the loop, the absolute best answer *for this specific tree* is selected.

```
# Weighted Score = 0.5*MinReward + 0.3*Visits + 0.2*UCB
best_answer = max(answers_list, key=weighted_score)
```

## Step 6: Aggregate Results (Forest Level)

**File**: [run_with_mcf.py](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) (Inside `monte_carlo_forest.run` loop)

After running all `tree_nums` trees, the code collects all [tree_ans](file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf_stop_noearly.py#212-222) (best answer from each tree).

- *Note: In the provided code, `is_correct` logic seems to check each tree individually, but a robust Forest implementation would typically take a majority vote here.*

## Step 7: Verify Correctness

**Function**: [check()](file:///Users/admin/Documents/Forest-of-Thought/utils/utils.py#241-303)
**File**: [utils/utils.py](file:///Users/admin/Documents/Forest-of-Thought/utils/utils.py) (Lines 238-300)

**Logic**:

```
# Extract and compare canonical answers
if extract_label(answer) == extract_label(ground_truth):
    return True
```

**Console Output**:

```
18 18
correct_num=1, current_num=1
```

# Summary Table of Key Files

| Aspect | File | Key Functions |
| --- | --- | --- |
| **Main Logic** | [run_with_mcf.py] (file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py) | `Monte_Carlo_Forest.run` , [monte_carlo_tree] (file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf_stop_noearly.py#223-313), [step] (file:///Users/admin/Documents/Forest-of-Thought/run_with_mcf.py#144-148) |
| **Diversity** | [utils/examples.py] (file:///Users/admin/Documents/Forest-of-Thought/utils/examples.py) | [get_similarity_question] (file:///Users/admin/Documents/Forest-of-Thought/utils/examples.py#217-234), [get_examples] (file:///Users/admin/Documents/Forest-of-Thought/utils/examples.py#8-215) |
| **Model API** | [models/load_local_model.py] (file:///Users/admin/Documents/Forest-of-Thought/models/load_local_model.py) | [get_respond] (file:///Users/admin/Documents/Forest-of-Thought/models/load_local_model.py#70-124), [get_respond_ollama] (file:///Users/admin/Documents/Forest-of-Thought/models/load_local_model.py#125-167) |
| **Utility** | [utils/utils.py] (file:///Users/admin/Documents/Forest-of-Thought/utils/utils.py) | [check] (file:///Users/admin/Documents/Forest-of-Thought/utils/utils.py#241-303), [extract_label] (file:///Users/admin/Documents/Forest-of-Thought/utils/utils.py#202-229) |