A1 — Reasoning and Action

Tasks

1. Set up an Evaluation Harness
2. Add Chain of Thought Prompting
3. Add ReAct Loop

Deliverables

- `part1.py`, `part2.py`, and `part3.py`
- `a1_report.pdf`

Details

Task 1: Evaluation Harness (4 pts)

Implement an evaluation harness that evaluates a model against a subset of GSM8K, a math reasoning benchmark. The code should take in a model name (one of the inference models available through the Argonne inference server) and a number of questions to evaluate against.

```
python part1.py —model <model> -n <num>
```

The prompt should define an expected output format for each question, and the script should parse the model output and compare it to the correct answer. The script should also count the number of output “tokens” (we will just use the number of words as a proxy for tokens), and measure the time to solve each prompt. The script should print the overall accuracy, and the average and standard deviation in time and number of tokens to solve each prompt.

In the report, include a couple sentences describing any design decisions that you made in this part of the assignment, and a table comparing at least 2 different models. You can use a relatively small sample of the dataset to keep the evaluation fast.

Task 2: Implement Chain of Thought Prompting (3 pts)

Modify the prompt from part 1 to include chain of thought prompting. Choose 10 samples from the GSM8K dataset and manually create a chain of thought to solve the questions. Next, modify the script to take another parameter <samples> that chooses how many input samples to use in the chain of thought.

```
python part2.py —model <model> -n <num> -s <samples>
```

In the report, choose 1 model and include a table comparing the performance with different numbers of samples. Perform some ablation studies changing the construction of the chain-of thought. Include a few sentences in the report commenting on how sensitive the performance was to construction of the chain of thought and selection of the samples.

Task 3: Implement a ReAct Loop (3 pts)

Modify part 1 to implement a ReAct loop. We recommend using langchain for this part of the assignment but do not require it. The logging/measuring of the output will have to change to account for the tokens used that are hidden by the langchain interface. Implement a calculator tool with simple arithmetic operations that the agent can use. The interface to this agent will be the same as part 1.

```
python part3.py —model <model> -n <num>
```

In the report, include a table comparing the original benchmark, chain of thought prompting, and a ReAct loop compare in accuracy, time, and tokens. Comment on how effective the tool was at improving the model performance and possible explanations for that behavior (i.e. Did the tool get called? Was it necessary to use the tool?).