Bianca Gauthier

Reflection

In this project, I built an iterative squaring bot that relies on GPT-4.1 to perform repeated multiplications instead of using Python's built-in math operations. While the task appears simple, the experiment reveals how quickly a language model stops behaving like a calculator and starts behaving like a guesser.

I can't say exactly why it is falling; however, I have noticed a pattern. When the numbers remain small, the bot performs very well, like I tried the input (3,5) and it correctly computed all iterations, even to its high value of 1,853,020,188,851,841. However, the moment you start going higher than at least 6 or 7 iterations, then that's where it makes errors. When it has very high number values, it makes errors, and that's where it's more obvious what is stomping it.

One of the first errors is that it didnt label all of the rest of the numbers the “e+57”; this automatically affected later numbers. In the (8,20) test, failure begins at iteration 6. Instead of returning 6.277101735386681e+57, the model outputs only 6.277101735386681, stripping away the exponent entirely. From that point forward, the errors compound: GPT produces small floating-point numbers where massive scientific-notation values should exist.

A similar breakdown happens with (2,10). GPT remains accurate until iteration 7, then suddenly drops the exponent again, returning values like 3.402823669209385 instead of 3.402823669209385e+38. Each iteration after that exaggerates the mismatch between believable-sounding output and actual mathematical truth.