

Report

The LLM normally can work well in bug fix, here's how I use LLM to facilitate bug fix in code and the pros and cons when I am using the LLM.

Usage of LLM

Prompts: Faulty Code + give me the bug brief in this code and point the bug position and give the fix version there's only one line wrong.

Basically, I use the above prompts to let the LLM identify the problem and fix it. And after LLM generate the code, I will put bug fix code into leetcode and test if the generating code is correct.

Pros and Cons

Pros: In four out of five examples, the LLM accurately diagnosed the root causes of errors and corrected the faulty code snippets in a single attempt. It effectively grasps the intended functionality of the code while pinpointing the precise locations of issues. LLMs demonstrate good performance in bug fixing. For such short code snippets in the examples, LLMs can resolve these bugs very easily.

Cons: In the remaining one questions where the LLM did not get the correct answer on the first try, it failed to generate results that matched expectations because it did not properly understand the instruction to "modify only one line of code".

In the "robot-collisions" example the LLM insists that the if branch in Pic.1 should be put outside the while loop which didn't follow the instruction. It moved more than one line of code in Pic.1.

Bug position (the one wrong line)

- Inside the collision while-loop:
if (shouldAddCurrent) {
processingStack.push(current);
}

This push is placed inside the while and should instead occur after the while finishes.

Pic.1 the LLM's bug position answer

Even after reiterating that only one error location could be moved, the LLM still

insisted that the entire if branch needed to be moved in Pic.2.

```
    }  
    // <-- moved this single line here (push only once after all collisions)  
    if (shouldAddCurrent) {  
        processingStack.push(current);  
    }  
}
```

Pic.2. LLM's second answer

After that, I just point out the possible solution, ask LLM not to move the if branch, after thinking, LLM got it right in Pic.3.

can we just change the if (shouldAddCurrent) condition inside rather than move the if branch

gpt-5-mini | Reasoning | Multi-Modal 今天 20:18

• -- Free credit: 7.39032; Paid credit: 0.0 --

Yes — you can fix it by changing only that condition. Briefly:

- Bug: the code pushes the surviving left-moving robot onto the stack immediately inside the collision while-loop, which prevents it from colliding with earlier right-moving robots.
- Bug position: the single line inside the while-loop:
if (shouldAddCurrent) { processingStack.push(current); }
- Fix (one-line change): only push current when it truly has no more right-moving robots to fight — i.e., when the stack is empty or the top is a left-moving robot.

Pic.3. LLM's right answer

The first answer LLM gave out is also right after I put it on Leetcode and it can pass all testcase. The only issue is that this answer does not meet the problem's requirement of modifying only one line of code.

Thus, we can have the conclusion that LLM can work well in many circumstance, however, under certain constraints, LLMs cannot work well on their own and still rely on some human guidance to ensure complete correctness.