Tech Interviews are still broken

I was at a friend's place on the 4th of July and they asked if I wanted to watch them complete a coding challenge for a prestigious AI bootcamp.

Surprised by their definition of "hanging out", I nonetheless said yes. I was curious for a few reasons:

1. It had been a while since I had even looked at a coding challenge. With the advent of AI, I was curious if the style of coding challenges had changed at all.
2. I wanted to see how my friends went about solving coding challenges, because I sucked at them.

I asked them if my presence would be distracting for the test. They said it would, but it didn't matter. They weren't that interested in this bootcamp. Besides, the test was going to be on "general python skills", so it probably wouldn't be particularly difficult.

Anyway, they jumped onto their computer, pulled up a seat for me and tippy-tapped and clicky-clacked to the email containing the test link.

A 1 hour timer started off in the top right corner of the screen. 5 coding challenges popped up on screen.

They were, in order:

1. <a LeetCode Easy problem>
2. <a LeetCode Medium problem>
3. <a LeetCode Medium problem>
4. <a LeetCode Medium problem>
5. <a LeetCode Hard problem>

My friend was off to the races. They breezed through the first problem, verbally exploring potential solutions, fingers zipping across the keyboard and generally making great time. They started off on the second... and that's when it all went to shit.

They struggled hard with the remaining problems. They found solutions that didn't cover every test case. They scribbled on their notepad. They attempted to divine a DP solution out of thin air. With only 15 minutes left, my friend decided to ask an AI chatbot before time ran out.

When the timer finally expired, we vented about how terrible that went.

As I reflect on that experience, a few thoughts (and intense feelings) bubbled to the surface:

1. To my dismay, there continues to be a whole sub-category of the tech industry for interviewing engineers in these asinine ways. LeetCode, HackerRank, CodeSignal, etc continue to thrive when I really thought they were a passing fad.
2. Success in interviews continues to be defined as: "Grind as many of these problems as you can until you start seeing the patterns in your sleep." It often feels like you have to have memorized the optimal solution to pass these tests.

3. These mentally taxing challenges still have nothing to do with the actual bootcamp/job that the applicant is applying for. In other words, gatekeepers are using irrelevant tests to filter out more applicants than strictly necessary.
4. Experience in the industry is irrelevant to doing well in these interviews. Everyone I know who has ever left their job or been laid off has said they have had to “get back into LeetCode”. The implication being that our jobs did not require skills that we learnt for interviews.
5. AI can solve these problems with graceful ease. But instead of embracing AI and giving up on these pointless tests, the gatekeepers have doubled down on the strategy: keeping the same tests, but trying to detect AI-assisted cheating.
6. I’m grateful and lucky to still have a well-paying job in this industry, cause I’d be in big trouble if I had to grind LeetCode to get in anywhere. I don’t do well in these kinds of tests.

I think that if there were ever a time for algorithm-heavy tech interviews to end, it is now, in the age of AI and for a few reasons:

Most software engineers don’t solve these kinds of problems

We don’t.

We really don’t.

At least I don’t and the people I know don’t.

Sure, yes, I’m generalizing. If I spend more time, I can collect data and make a better case. For now, it’s just my opinion.

As engineers, we’re asked to solve all kinds of problems, not just algorithmic ones. In fact, we often have to solve problems along constraints where an O(log n) algorithm does nothing for us. For example, if your boss wants the feature to land by tomorrow, a 10 line brute-force algorithm can often be the best way to make that happen.

AI can find the correction solution

Using AI tools in software engineering is an acceptable thing to do now. I’m not a fan of how carelessly AI is used by people, but I can’t deny, it seems adept at solving these kinds of coding problems. As I understand it, it is most likely because it has seen these kinds of problems before, because they fit a pattern.

The tech industry as a whole is moving towards a model where we exporting some of our thinking to AI using tools like Claude Code and Gemini CLI. So why would you restrict the use of these tools in the interview?

Assuming that we’ve all studied the principles and done the hard work “without calculators” at our various universities and colleges, a job interview seems like the point at which you should be allowed to use every advantage you have at your disposal?

AI can be used to make better tests

Since there are so many people vying for the top jobs and so few people to screen them properly, an easy way out is to make a really hard test that very few engineers can pass.

I think AI can do a better job here!

If you recall, the goal of my friend's bootcamp test was to determine "general python ability". If the intentions are really that innocent, this is easily fixed by using AI tools to build and run the test.

This is fairly easy to prove. On a whim, I just asked Gemini to "Design a technical interview to test general python skills. The interview will be used to determine if the candidate can attend a bootcamp about AI."

It did a fantastic job of designing a test with that intention in mind! It had a conversational section where I'd ask questions about python concepts and libraries. It had a section for a practical coding challenge. It even had a part where I'd ask the candidate to explain their code and do an informal code review on it.

I like that! It's not perfect obviously, but it's much closer to an ideal interview, if you ask me.

The system expects false-negatives, but false-positives are also common!

There's plenty of engineers who can pass LeetCode tests, but are in fact terrible engineers to work with.

They can be the absolute best at algorithmic problem solving, but they can also be terrible at many of the other **untested** skills required of our job: documentation, testing, code health, comments, foresight, initiative, independence, scheduling, prioritization, socialization.

This is actually a hidden assumption of tech interviews. I've heard engineers confidently tell their friends that they have the job in the bag after just passing the technical rounds. And they indeed do. Because every other part of the interview process has been sidelined.

Even without AI, there are better options

Thankfully, a few companies are already finding good alternatives to this interview process, but those approaches have yet to gain traction.

One alternative is to do a "work trial", where you start working for the company for about a week or so and at the end of that week, the engineers you collaborated with give feedback about whether you should be hired or not.

Another alternative is to give more practical problems that are closer to the work that the engineer would actually be doing.

A common criticism of these approaches is the amount of time that is now invested in the interview process, which matters when there's thousands of applicants, made worse by the barrage of layoffs at big tech companies.

I hope we can move to an interview model that is better than what most companies use today. I'm personally invested in such a change, because it would mean that whenever I want to change my job, I don't have to mindlessly practice a bunch of algorithmic coding questions that I will never use once I pass the test.