<- Back to Blog

How to effectively conduct programming interviews

How to reliably understand if a candidate is a good programmer, inspired by Casey Muratori

Iason P. · October 22, 2025

Contents

Introduction

Are you struggling with hiring good programmers? This guide will help you to reliably understand if a candidate is a competent programmer, inspired by Casey Muratori's interview.

TL;DR

Casey doesn't like LeetCode style of interviews, instead, he follows a drill-down approach where he

- Chooses a project the candidate has worked on.
- Asks questions with the goal of having the candidate teach him about his project.

With this approach, he claims he can **always** understand if someone is a competent programmer, and he **has never seen it fail**.

Casey’s Approach

According to Casey, two questions need to be answered in an interview

1. **If they are competent programmers**.
2. **Whether they will be productive at your company/project**.

His opinion is that the second question is much harder to answer, and he doesn't know how to reliably do it, and whether there is a way to answer it.

For the first question, he says that with the process he is following, he can always understand when someone is a good programmer.

The process he follows can be summarized in the following steps:

- When looking at someone's CV, choose a project they have worked on.
- Narrow down on something specific from this project with the goal of making them teach you that thing.
- Try to get as narrow as possible, even to implementation details.
- You don't have to choose something you are not knowledgeable about, but refrain from guiding them if you know the topic.
- Ask questions like "What did you do here?", "What other approaches did you consider?".
- Also ask questions that would change the requirements, like "what if we had to do this faster?".

If they can answer your questions, and their answers come from solid foundations, it means that they know what they are doing. If they can't, then they are not what you are looking for.

In his opinion, one of the benefits of this approach, compared to LeetCode style interviews, is that people are responding to the questions from a place of comfort, which is similar to how they actually work.

Asking someone a question like reversing a linked list could make them uncomfortable. Simply because this is something they haven't done in years. On the other hand, if they are comfortable with this question, it is probably because they have practiced, and this doesn't say a lot about their skills.

Our Take

We generally agree with Casey about LeetCode style interviews, and that they usually don't indicate how good someone is, but instead, they show how well they have prepared. Which is not very applicable to real life. You can't (always) be prepared for the constraints in a project. Besides, a hash-map is not always the answer 😅.

Benefits of LeetCode style interviews

To play devil's advocate: difficult coding questions are a way to minimize false positives, with the side effect of having a large number of false negatives. Being able to prepare and apply learned techniques is a kind of problem solving. The ability of problem solving is the most important skill a programmer should have.

Another benefit of LeetCode style interviews is that they are somehow standardized. Standardized processes help large organizations stay consistent.

In our opinion, interviews of this style are not the only way to eliminate false positives. The same result can be achieved with Casey's approach too. After all, how could a candidate fake the reasons behind their project decisions, or struggle to explain how they would improve something they have spent a lot of time building?

What is missing from Casey's approach

While we prefer this approach from LeetCode style interviews, we think something is **missing**: A way to understand how adaptable the candidate is to a new codebase.

A different set of skills is needed for jumping into a huge codebase and making contributions, from being able to explain why and how you did something you had spent a lot of time working on.

Our proposed solution to this is to have a **pair programming session** with the candidate, where they have to fix a bug in an unknown codebase. During this session you can understand:

- **The candidate's approach to new problems**.
- **How they navigate unknown codebases**.
- **How good they are at spotting mistakes**.
- **How good they are at expressing themselves and collaborating**.

To get the most from this session you could try to

1. Have the codebase relatively small to not overwhelm the candidate.
2. For them to be comfortable, let them share their screen. This way they can use their favorite tools.
3. Allow them to use AI, since it's a core part of the tools we use nowadays.

Linear’s novel approach

Another quite novel approach to tech hiring we should mention, is Linear's work trials.

At Linear, they were struggling to find the right people for their fast-paced environment following the standard interview process.

For that reason, they decided to do paid work trials, which typically last from 2 to 5 days depending on the seniority of the role. They mention that this approach is working very well for them, and they have achieved, at the time of writing, 96% retention. Everyone has to do them, from engineering to C-level candidates. During the trial the candidates work on a real project.

We also believe that this could be the best way to find the perfect fit, pairing on actual tasks should be the best indicator. Unfortunately, not everyone has the bandwidth or culture to support this.

Conclusion

In our opinion, Casey's approach to coding interviews is an improvement to the current standard. However, it can be extended to have more certainty that a candidate is a good fit.

Our proposed approach is:

- **One drill-down session on a candidate's project**.
- **One pair programming session working on a bug**.

We would love to hear your approach to coding interviews and whether you disagree with our views.

Feel free to reach out at X/twitter or email me directly at iason at gethopp dot app.

Finally, after you have made the right hire, you can use Hopp, our own OSS pair programming app, to have an even better onboarding experience. During the pairing sessions, you can guide the new hires by smoothly taking control of their computer when needed and saving them time from learning obscure commands and processes.

PS: Only Grammarly has been used on this post, I decided that I am not going to use AI when putting my views online.

interviews-process programming

• • •

GET STARTED

Ready for a better way to pair?

Imagine never losing your flow to screen sharing lag again. Picture pairing sessions that feel as smooth as coding locally. You're not just getting a tool—you're unlocking the collaboration experience that separates world-class developers from the rest.

Get started

Hopp

Low-latency code-pairing made by developers, for developers

Resources

About

Blog

Changelog

Pair Programming Encyclopedia

OSS Friends

Made in Europe, by Costa A. and Iason P.

Privacy policy