CS 393 TA Guide

By Drew Wilson

Mock Interviews

This is where you'll spend the majority of your time as a TA for this class. You'll be conducting mock interviews with students, giving them feedback, and helping them improve their interviewing skills.

You are qualified to do this job

Some TAs have reached out to me before concerned that they're not "interviewing the right way," or that they're not "qualified" to be conducting mock interviews. I want to assure you that you are. You don't need to be an expert in the field to help students improve their interviewing skills. You just need to be able to ask good questions, give good feedback, and be encouraging. In real interviews, I've had all kinds of interviewers with different personalities and interview styles. Some were very technical, some were very casual, some were very formal. The students need to be prepared for all kinds of interviewers, so it's good for them to get a variety of interview experiences. So, don't worry about "doing it right." Know that you bring value to the students just by being there and helping them practice.

Choosing a problem

I try to ask the students questions that relate to a recent topic we've covered in class. As I've done Leetcode problems, I've added them to an organized list I can use as a "question bank." I try to have ~3 questions prepared for each topic so that I can be sure to ask the student a question they haven't seen before. I suggest that as you go through Leetcode problems, you create your own list of problems you're comfortable with and can ask the students.

Conducting the interview

Traditionally, the TAs in this class have used Calendly to schedule mock interviews. I've typically done mine in person in the TMCB basement (back of room 1066) with a whiteboard, because I like to give the students a variety of interview experiences. However, most TAs have done their online over Zoom, which is also a great option since that's becoming more common in the industry, and gives the students more flexibility. It's good to have a mix of interview styles available. When doing interviews over Zoom, there are two main ways to give the student a coding problem: 1) send them a link to a Leetcode problem and have them share their screen as they code, or 2) use a shared code editor like CodeShare and explain the problem verbally. Again, the students need to be prepared for any kind of interview, so the variety is good.

A typical interview might go like this:

- 2-3 minutes to introduce yourself and get to know the student or catch up since the last time you met
- 2-3 minutes to explain the problem, give examples, and answer any questions the student has about the problem
- 20 minutes for the student to solve the problem
- 5 minutes at the end to discuss the solution and give feedback

A few assorted tips:

- Be encouraging. The students are nervous, and they need to know that you're on their side and want them to succeed.
- Get to know them. Ask them about their background, their interests, their career goals, etc. This will help you give more personalized feedback and help them feel more comfortable.
- I like to not share the constraints of the problem (how large input can get, etc.) until the student asks for them. This is more realistic to a real interview, where you have to ask clarifying questions to get all the information you need, and it's a good habit for the students to get into early.
- If the student is stuck, you can give them hints or ask them questions to help them get unstuck. You can also ask them to explain their thought process to you, which can help them see where they're going wrong.
- If the student finishes the problem quickly, you can ask them to optimize it, or ask them a follow-up question like adding additional constraints or complexities. This gets easier as you do more problems and get a feel for what kinds of follow-up questions are appropriate.
- **Have the student think out loud**. This is the most important part of the interview. You need to know what the student is thinking so you can help them when they get stuck. If they're silent, you can't help them. If they're talking, you can guide them in the right direction. The same applies to a real interview. If you're silent, the interviewer doesn't get to learn about the way you think, which is often more important than the solution itself. While they're coding, have them explain their thought process and why they're making the decisions they're making, what their code is doing, etc.

Giving feedback

Focus on the positives. What did the students do well? Let them know so they can get into the habit of doing it consistently. Then, give them constructive feedback on what they can improve. Be specific. Don't just say "you need to improve your problem-solving skills." Say "you need to practice breaking down the problem into smaller parts before you start coding." This gives them something concrete to work on. If you can, give them a specific example of a time they did something well or poorly. This helps them see what you're talking about and gives them a concrete example to work from.

Most common points of feedback I give:

- **Clarifying questions**: Make sure you understand the problem before you start coding. Ask clarifying questions if you need to.
- **Thinking out loud**: Make sure you're explaining your thought process to the interviewer. This helps them help you when you get stuck.
- **Edge cases**: Make sure you're considering edge cases and constraints of the problem.
- **Optimization**: Once you have a working solution, think about how you can make it better. Can you make it faster? Use less memory? Handle more complex inputs?
- **Communication**: Make sure you're communicating clearly with the interviewer. If you're stuck, let them know. If you have an idea, share it. If you're not sure about something, ask.

Topic Exams

Throughout the semester, we typically have a few topic exams. These are opportunities for students to pass off / demonstrate their understanding of a particular topic. For the most part, they're self-reported by students. They're done a bit differently every year but typically there is a "live" portion where students have to solve a problem with a time limit in front of a TA, and students can complete this by signing up for time on your calendar. You'll want to remind them they can book 2 back-to-back slots as these problems can often

take more than 30 minutes without your help. Students can also retake topic exams (confirm this, topic exam formats do change from year to year), so students can book time on your calendar for that as well.

Other Responsibilities as a TA

Help With Resumes, LinkedIn, and Nontechnical Interviews

A lot of students have asked me about what should go on their resume, how to format it, any resources that might exist, etc. I will usually show them my resume/LinkedIn and point them to Lane Muranaka who's a great resource for this kind of thing. I also have a list of resources I've found helpful for this kind of thing that I can share with you if you're interested.

Then we'll usually talk a little bit about that student's experience so far, what they've done with problem solving and computer science outside of class, and then use that to help them build their resume. Here are BYU's resume templates.

This approach is also really helpful because for many companies, the first interview you'll get is often a resume/experience review with a recruiter, so it's good to practice talking through your resume experience and how it relates to the job you're applying for. There are also tons of basic questions that you can prepare for that are common in these kinds of interviews, so we'll usually talk through some of those as well. A simple Google search / prior experience should be enough to help you prepare a list of 10ish questions that are asked in 80% of behavioral/fit interviews. Here are some of the most common ones I've seen:

1. Tell me about yourself
    - Very common, deceptive in its simplicity. You should have a 1-2 minute answer prepared that covers your background, your experience, and your career goals.
2. Why do you want to work here?
    - This is a great opportunity to show that you've done your research on the company and that you're excited about the work they're doing.
3. What are your strengths/weaknesses?
    - This is a classic question that you should have a good answer prepared for. For strengths, try to think of a few that are relevant to the job you're applying for. For weaknesses, try to think of something that's actually a weakness (none of that "I work too hard" stuff) but that you're actively working on improving.
4. Tell me about a time you had a conflict with a coworker
5. What's a project you're proud of / a project you've worked on outside of school / tell me about a difficult technical problem you've solved
    - Show enthusiasm for the work you've done and be prepared to talk about it in detail.
6. Tell me about a time you've led a team

Final Exam

393 has a final exam that the students must pass to pass the class. Typically, it involves solving a certain number of leetcode problems out of a set over three sessions. Typically there are ~12 questions and you need to solve 6 for 100%. Students can use 2 partial solutions (passing 50%+ of the test cases and having a good approach) to make up for up to one full solution. This part also does change slightly from year to year so be sure to confirm with the professor. Typically, the questions for each session are announced at the beginning and for their solution to count towards their final exam grade, they need to solve it in that one sitting.

As a TA, if you're available, you should come to class or help out over Zoom by administering hints via the help queue. The professor will usually set up a Google Sheets document linked to a survey that students can submit their progress to and solicit help. You can then go through the sheet and help students who are stuck.

Reassurance & Confidence Building

Everyone will learn at a different pace. Different students have different strengths and weaknesses both as interviewees and as future software engineers. It's important to remember that not being able to solve a problem as fast as someone else or or not understanding dynamic programming after several weeks of practice does not mean you're not cut out for this field. You might just need more time, more practice, or a different approach.

The same way that students have different strengths and weaknesses, different jobs have different requirements, and someone that may be unfit for one job may be a perfect fit for another. Even if the job has the same title, roles can look different at different companies. It's not important to land an offer at a top company to be successful in tech. It's much more important to find a role that fits your individual gifts and passions, helps you grow, and provides the opportunities you need to thrive.

Keep this in mind for yourself and for your students. A lot of them may express disappointment, frustration, or despair at their progress. Students have told me they feel hopeless at the job search and that this class has made them feel like they're not cut out for tech. When this happens, remind the students of the above. Computer science is a hard field. Imposter syndrome is rampant. It's normal to feel like you're not good enough. But you are. You're here, you're learning, you're growing. You're doing great. Keep going. Let's do what we can to help serve these students and reassure them that they are capable and that they can succeed.

Similarly, a lot of students will be worried about this class. Many students aren't phased at all by the workload or the technical aspects of this class, but to some, it's overwhelming. Be aware of these students and be available to help reassure them that they can succeed and that you are there to help them. A lot of the concerns come from ambiguity or unclear understanding of course/assignment expectations. Be sure to clarify these as much as possible and be available to help students understand what they need to do to succeed.If you have any questions, ask the other TAs or the professor. We're all here to help you succeed in this role. Good luck!