Interview Experience & Advice

# Companies and Roles

* Microsoft – Software Engineer
* Microsoft – Technical Evangelist (Intern & FT)
* Google – Software Engineer
* TopHat – Full Stack Developer
* ParseHub – Full Stack Developer
* Scotiabank – Data Scientist

General Interview Process

Generally, interviews follow the structure of one casual phone call, two slightly more technical phone calls, and then onsite interviews. The first phone call is filled with behavioural questions. These usually consist of pretty open ended tech-oriented questions that provide a good place for you to plug in any of your projects or work experience. If you can’t find a way to plug in related experience, just do your best to answer honestly. Never try to make up an answer and sound smart, because they’ll detect it from 1000km away. The technical interviews vary between companies. Google is known to have you code solutions to programming problems in a Google doc, while Microsoft only ever asked me more abstract problems that I just explained answers to over the phone.

At the end of every interview, you should always have questions to ask. If any come up on the spot that you’re genuinely curious about, ask away, but also prepare a few just in case. The interviewers really can’t explain the entirety of the job to you in the time they’re given, so it’s only expected that you want to know more about it. Good questions to ask include “What will be a measure of success for the candidate you hire?”, “What do you do on an average day?”, “How large is the team I will be working with?”, “What percentage of my time will be spent programming?”, “What will the learning curve for this role look like?”, and so on. There’s tons of specific questions you could ask based on the company, but these are pretty general and still do the trick.

Onsite Interviews

Onsite interviews are very consistent across the board for tech companies like Microsoft, Google, Amazon, Facebook, and Apple. You are placed in morning or afternoon slot where you have 4 interviews scheduled. Each one is about 45 minutes, and you have a 15-minute break in between to use the bathroom and get a drink. A good sign that you’re doing well is when interviewers pick up on what you say as though they’ve heard it before. This is because in that 15-minute break period, all the interviewers write feedback into a spreadsheet, and your next interviewer reads it before you come in.

Behavioural Questions

With more typical behavioural questions it is best to approach them with a Situation, Action, Result method (SAR). For example, “Tell me about a time you experienced conflict in a leadership role” is a really common question. First you define the Situation, “My teammates and I disagreed on the division of labour”, then you explain your chosen Action, “so I decided to draw up a diagram that can more evenly assign tasks based on people’s strengths and preferences, which we voted on”, followed by the result, “and everybody was satisfied, which put less stress on our collaboration and allowed us to get the top mark in the class.”

By using the SAR approach, you give the interviewer a brief story that doesn’t take too much time but still provides them all the information they need to understand what happened. This lets them ask more if they’re curious, or move on if they are satisfied.

Technical Questions

This is the part most people are concerned about. This is where *Cracking the Coding Interview* comes in. **You need to be comfortable with the fact that you will be asked a question you do not know the answer to.** If you can come to grips with this fact and avoid getting nervous, then the rest will follow. Remember that the interview is a conversation. In technical questions, a very good approach is to let the interviewer know what you think a solution might be before you code anything, be it on a whiteboard, Google doc, etc. If you gauge their reaction, it will usually let you know if what you’re doing is incorrect. They want you to succeed, so they generally push you away from wrong solutions.

It is also a perfectly fine approach to start with a suboptimal solution and improve afterwards. A solution that is O(n2) is better than no solution, even if optimal is O(n). Be sure to announce to the interviewer that this is your intention, and that **you are aware of the fact that it is suboptimal.**

# Microsoft – Software Engineer

I had one casual phone call as well as one technical phone call before I was invited to do onsite interviews. The casual phone call was very much an organic conversation. It didn’t seem like they had many set questions they really needed to ask, but rather just looked at my resume and wanted to know more about certain projects. Behavioural questions I got asked include “How do you keep up to date with technology?”, “Tell me about a time you tried to do something that didn’t work.”, “What’s your favourite Microsoft technology, and how could it be improved?” It definitely helps to know recent news regarding the company you’re interviewing with, or at least a bit about their technology used.

Questions I got asked revolved around objected oriented programming concepts (inheritance, polymorphism) as well as normal programming problems. The programming ones ranged in difficulty (I was asked to reverse a string, as well as FizzBuzz) as well as more complex problems involving recursion and hash tables.

# Microsoft – Technical Evangelist