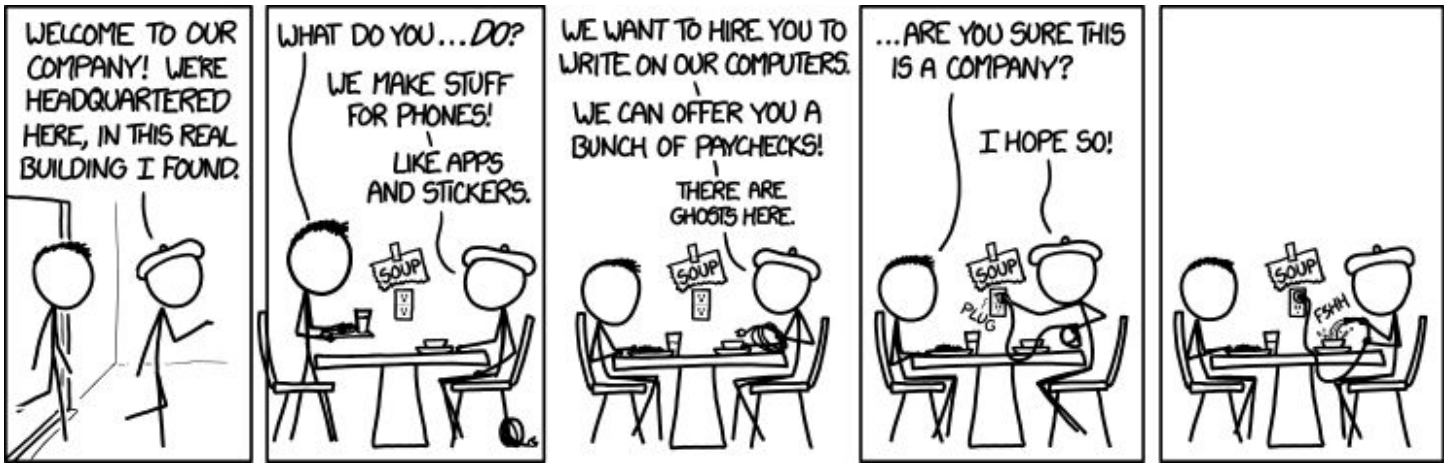
# Interview Tips




## Credit

The information in this document was given to me by my friend, Veronica Gunn, along the way. Networking with awesome CS people  is important!

## General Comments

I know we all don't like to think so, but interviewing is part luck. There are several factors that are completely out of your control. The interviewer might be in a bad mood, you might be sick, the phone connection might be bad, you might be asked a question you just don't happen to know, the list goes on. You can't control these things, and it's better to acknowledge that to start with.

Your interviewer is another human being. This person is not some programming god who is here to make you feel inadequate.

Interviews are a game. You're solving a puzzle and getting to know another engineer. It's fun, so you may as well have fun with it.

## Phone Interviews

Your interviewer will call you, usually. Don't panic if the call is a couple minutes late. If you still haven't received a call or email ten minutes after your scheduled interview time, email your recruiter. I send something like this:

> Hi (Recruiter's name),
>
> > I'm very excited to speak with (interviewer's name if known or "a team member" if not known)! According to the email I received I am scheduled to interview today at (time, including time zone) at (your phone number, including area code). I haven't received a call yet and wanted to confirm that this information was correct.
>
> Thank you,

(Your name)

Make sure you're in a quiet area with good internet and cell reception.

Politely tell the interviewer immediately if you can't hear them, if there's an echo, or if there's any other auditory factor that would prevent you from succeeding. The sooner you say something, the less it sounds like an excuse and the earlier in the interview it gets corrected. "Struggle through" isn't the answer here.

## On Campus

Be careful with your scheduling! Remember that you will have to walk to wherever interviews are being held, and that might not be a convenient location. If you have class directly before, make sure you know where you're going and how to get there quickly.

*Dress:* You should dress differently depending on what company you are interviewing with. Remember that you can always ask your recruiter what kind of dress is expected! Here is what I generally wear for each category, and which categories requested this kind of dress:

**Business Casual** (RetailMeNot, Union Pacific, Verizon)

- A dress that is about knee length and covers my shoulders, flat black shoes, light makeup, hair half up half down
- A pencil skirt, a blouse, heels, light makeup, hair half up half down

**Business Professional** (Banks, gas and oil companies):

- A pencil skirt, a blouse or a button up shirt, a jacket, makeup, hair up, appropriate jewelry (I like to wear pearls, but it depends on what else you're wearing)
    - Could wear slacks instead of a pencil skirt
- A professional dress, a jacket if that dress doesn't cover my shoulders, makeup, hair up, appropriate jewelry

**Casual:**

- Jeans and a blouse. I usually don't wear makeup because it makes me uncomfortable, but apply whatever makeup you typically wear on a daily basis. I usually wear flats or nice sneakers.
- Jeans and a t-shirt. I wouldn't advise wearing a t-shirt with a company logo on it, either that company or another company.

## Q/A

**I always get super nervous before/during an interview. How should I deal with this?**

It is very likely that in 6 months you will be the only one who remembers this interview. Not getting an offer from Google (or Microsoft or Apple or Facebook or whoever it is at this moment) is not the end all be all I quit at life scenario it can seem like. My freshman year (last year) I really psyched myself out for most of my interviews and made huge events of them. I'm happier and do better now that I've learned to calm myself down.

Acknowledge that a lot of this process is luck. There are a lot of factors that go into an interview that are completely out of your control. Was the interviewer in a bad mood? Were you sick? Could you not understand a word that was said over the phone? Did you get a question that you just didn't happen to know? All of those are things that can (and DO!) happen. When they happen it's not the end of the world. you're not a bad programmer and you're going to find a job.

Treat it like a game. I think that this is my best piece of advice. You're solving a puzzle with another engineer. It's problem solving, it's fun. After the interview you never have to think about the problem again if you don't want to. Your interviewer isn't some super human who is superior to you (and the vast majority of them do not behave as if they are).

**What challenges do you face and what tips do you have for interviewing as a girl?**
I've heard horror stories of women who faced sexism during the interview process, but can honestly say that I haven't really felt that way this semester. During my freshman year, I did deal with some down talk that *might* have been related to my gender.

If I go in obviously nervous, I am taken less seriously. When I go in happy to be there and excited to problem solve, I am taken seriously 99% of the time.

**What attitude do you try to project during an interview?**
From my experience, how I'm feeling has a lot to do with how I interview. If I'm trying to "fake it", I come across as insincere. Don't try and project any attitude. Be who you actually are, I promise you don't suck!

That being said, you have power over how you feel and how you are therefore perceived. The more I look at interviews as a game and a fun interaction, the less nervous I feel, the more excited and happy I am, the more competent I seem, and the better I perform.

**How do you prepare for a technical interview? What topics do you study?**
Ok, everyone always hates my answer to this question. I'm really involved with competitive programming and that helps me a lot with interview prep. I'm constantly challenged to solve new problems under time and pressure constraints. It's a great way for me to keep my data structures and problem solving skills fresh. The questions that I see also tend to be much harder than the ones asked during an interview.

As for topics, here's a list. All of these are things that I have actually been asked this semester. I listed them based on how frequently I was asked about/used each thing.:

- Big O/Runtime efficiency of both your code and all operations on all structures listed below
- Testing
    - How to identify potential edge cases
    - How to write test cases for those edge cases (in edition to some general test cases)

    - Why is testing important?
- Arrays/ArrayLists/2D Arrays
    - How do ArrayLists work?
- Linked Lists
    - How do they work, how do they differ from ArrayLists?
    - How to reverse and manipulate a Linked List
- Iterators (especially in regard to Linked Lists)
- Recursion
    - The basic idea, why it's useful, how to establish base cases
    - Recursive backtracking
    - How to find Big O of recursive code
- Trees
    - Binary trees
    - Binary search trees (insertion, removal, traversals)
    - Trees with more than two child nodes
    - Balanced binary search trees such as red black trees (Know they exist and the basic differences. You don't generally need to know algorithms (Generally. I'm looking at you again, Dropbox)).
    - How to traverse through a tree, both iteratively and recursively (Count the number of nodes with odd values/count the number of leaves were particularly popular this year it seems)
- Stacks/Queues
    - How to implement
    - When they are useful
- HashSets/Tables
    - How are hash sets implemented?
    - Why are they useful? (runtimes)
    - Potential issues (collisions)
    - Know when to use a HashSet in your code, and be prepared to justify this decision.
- Graphs
    - Basic graph algorithms (Such as Dijkstra's shortest path. You don't usually need to know specific implementation details)
    - Directed vs undirected
    - Weighted vs unweighted
    - The basics of how to implement/use a graph
        - Different implementations and when they are useful:
            - Adjacency Matrix (good for dense graphs)
            - Adjacency List
            - Linked node structure
- Tries
    - What a trie is, when it is useful, and how to implement one
- Sorting/searching

    - Basics of each of the sorting algorithms (insertion, selection, merge, and quick)
        - Their runtimes
        - Basic idea of how to implement
- Dynamic Programming/Memoization
    - Only ever used to solve problems. Important to know how to write clean code and how it improves efficiency.

**SUGGEST AN EDIT**