title: **Technical Interviews & ADHD**
num_comments: 30
num_up_votes: 90
upvote_ratio: 1.0

#### Post Text ####

Hi all! I've been programming for a little over 5 years and am self-taught. I recently got diagnosed with ADHD and General Anxiety Disorder and that's been helpful to have a bit more understanding of why things have felt really hard this whole time and why it feels like I haven't retained a lot of the information that I learn in the moment. However, I'm still trying to get setup with medication and am just starting to try out some suggestions for how to work with my ADHD better.   I'm currently employed but am interested in another role at a company that has a mission I really care about. However, I'm holding back from applying because I'm stressed out about having to do a technical interview where you code with someone watching you. I draw a blank in situations like this, forget what I do know, scramble to google things and don't perform well. My current and previous job had take-home assignments that I could do on my own and then talk through at a panel and those went really well. I'm considering asking this potential company if that would be an option but I'm not sure if that's going be looked down on and I don't know if it'd backfire to even mention my ADHD? I want to show my competency but I know that in that scenario of being on the spot with someone I don't know, I'm not going to do as well but I could happily talk them through it once I've done the work. Any thoughts or experience with this out there? Thanks all!

#### Comments ####

###### Unknown_User
ID: REDACTED! ~(o.o)~ <3, Upvotes: 34
I have been struggling with this problem as well.

I recently made it through technical interview and just got my official offer letter at an awesome company.

What I did was when practicing leetcode problems or even just general problems I'm dealing with I try to break down the problems in pictures of all the objects I'm dealing with it.

Using this method it helped me visual the problem I'm trying to solve and then solve it on paper first with the pictures/objects I've created.

Once I've solved the problem on paper it's a lot easier for me to structure my thoughts and my code. It also helps you show the company how you think in a very effective way which is more important than the rest of it.

Just my 2 cents on the matter and it worked for me

> ###### Commenter_2
> ID: REDACTED! ~(o.o)~ <3, Upvotes: 25
> > It also helps you show the company how you think in a very effective way which is more important than the rest of it.
>
> I can't stress this enough. If I had an applicant that solved the entire problem on paper before even writing one bit of code, that's halfway there. If they get stuck on the implementation details, I'll often even offer things to help them along preemptively. I don't want someone that's memorized a language's spec. I want someone that can approach, deconstruct, and solve problems. If you forget how to write some syntax, that doesn't matter. Hell, I forget syntax all the time.

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 8
Gah, that's really helpful to hear. I think I get so focused on the syntax or doing something the right way and that feels more important than just demonstrating critical thinking and approach. Good reminder to shift this perspective a bit to focus on what's more important.

###### Unknown_User
ID: REDACTED! ~(o.o)~ <3, Upvotes: 6
Absolutely, which in my past experience always talking out my ideas and showing process through pictures has not only helped me collect my thoughts into simple ideas but also helps me then translate it into syntax very easily.

I always say if your first step in solving a problem is writing code then you've already lost.

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 7
Congrats on the offer! That seems like a great suggestion to pull the brain back from drawing the blank and kind of pushes it to move forward. And I agree that being able to do this would demonstrate a lot to the company about how you work rather than the specifics of the code itself. Thanks for sharing:)

###### Unknown_User
ID: REDACTED! ~(o.o)~ <3, Upvotes: 5
Thank you!

Good luck and PM if you have any questions or want to go deeper into certain examples I've used in the past that helped me train my brain into this method.

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
Awesome, thanks!

###### Commenter_4
ID: REDACTED! ~(o.o)~ <3, Upvotes: 3
great comment

###### Commenter_2
ID: REDACTED! ~(o.o)~ <3, Upvotes: 17
I've been on both sides of the equation here, and my advice is to take it slow and not to be afraid to talk through things with your interviewer. In my experience, technical interviews aren't as much about your ability to pull code out of thin air, as much as your approach to solving problems. If you can't remember something, just say that. "I don't remember how to do this. I'm going to look it up." and then go look it up.

Also, **don't feel like you have to fill all the silence**. If you're stuck, take a moment to collect your thoughts, and let the interviewer sit there in silence while you do so. If it's helpful to you to talk through it with them (rubber duck), do so, but if you catch yourself babbling, rein it in and try to work through the problem in your head or on paper.

It's important to know that as an interviewer, we will stay silent as long as possible just to see how you work through something, how long you'll struggle, whether you ask for help. This isn't meant to intimidate or anything, it's just really to see your process and how it unfolds. In every interview I've done, when the applicant gets _really stuck_, and stops making forward progress, I would give them a nudge by asking a question that hopefully clues them in the direction they missed.

Remember, the goal is to find someone that can work independently, at the level of the position they're applying for. That means if you're interviewing someone for a junior position, you fully expect that they'll get lost in the weeds, stuck on various parts of the problem, and need a decent amount of help. That's why it's a junior position.

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 3
Great advice - especially about the silence bit. I often get focused on filling that and therefore, not thinking about the problem in front of me that needs my attention. Similarly, I think it can be easy to get so focused on coming up with the "right" solution as the goal that you forego spending your time and attention and adequately assessing and working through the problem.

###### Commenter_2
ID: REDACTED! ~(o.o)~ <3, Upvotes: 4
> it can be easy to get so focused on coming up with the "right" solution

Something that can really elevate you is actually talking through _why_ your solution is the right solution. Every, and I do mean **every** solution in software has a lot of tradeoffs. There is no solution that's correct for every context. If you, as a candidate, can speak to that and actually talk through the nuances of a particular solutions "correctness", and the pros and cons and tradeoffs it makes, I think that can really set you apart.

My point is: if you **do** focus on the "right" solution, turn that into talking through what that means with the interviewer.

An interesting side effect of ADHD is that I tend to go off on wild tangents. I've had technical interviews that went completely off the rails because of conversations like that. We may not have finished the original problem, but the point of the interview isn't to, you know, implement something. It's to figure out if the company and the candidate are good fits for one another. Mind, it's also important to temper that since it might frustrate the other person if they're wanting to stay focused on the topic, but it's definitely worth feeling it out.

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
That's a good reminder that the why/process can be more valuable than the what/content.

###### Commenter_5
ID: REDACTED! ~(o.o)~ <3, Upvotes: 16
I had a technical interview at a FAANG and my mind went completely blank, which panicked me, which made my mind race in circles to recover the thread of what I was thinking. I stopped, took a deep breath, said 'I just lost my train of thought, let's see...' and spoke out loud my thoughts from the top (the problem) and got myself back to a solution. I ended up getting the job!

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 5
Woot woot! That's great you got the job and even more so that you were able to catch yourself, share your thinking out loud and get back on track. A success story :)

###### Commenter_6
ID: REDACTED! ~(o.o)~ <3, Upvotes: 12
Following because I have one of these tomorrow and I have the same issue. I can pump out features on my own,

but don’t ask me to type my name if you’re watching.

###### Commenter_7
ID: REDACTED! ~(o.o)~ <3, Upvotes: 3
If it’s tomorrow it might be too late to ask if you can make it a take home. Is it only the technical tomorrow or is it a bunch of different interviews?

I think for us ADHDers, the working style of the team and company is just as important as the type of work. If we function better when pair programming or when working alone. Under pressure? With or without deadlines? In Scrum or Kanban? Etc etc.

I feel like you can learn a lot about whether it will be a good working environment if you can ask for what will work best for you (and in turn give them a better work product) and see if they have a way or are willing to accommodate.

I haven’t found a way to deal with nerves when people are watching, so I’m sorry I can’t help you there. :(

###### Commenter_8
ID: REDACTED! ~(o.o)~ <3, Upvotes: 9
I'm following this thread because I'm not that good with on the fly technical questions, let alone coding under pressure!

###### Unknown_User
ID: REDACTED! ~(o.o)~ <3, Upvotes: 6
[deleted]

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
I do think getting comfortable with "failing" is key because that seems like it will help you realize not getting something right can be the key to truly understanding it and it's also necessary to really grow. You've got to try stuff and get it wrong before you can get it right....that's just way easier to say than believe and do sometimes :)

###### Commenter_7
ID: REDACTED! ~(o.o)~ <3, Upvotes: 3
Do you know for sure that the technical interview at this potential employer doesn’t already offer a take-home assignment? It sounds like you haven’t applied yet so you might not know for sure?

I often hold back from doing things that I want to do because of a reason that _might_ be true but I realize that it often isn’t (or at least it isn’t as bad as I think it will be). I find it common among ADHD peeps like us to avoid starting or doing based on a future fear.

Applying for the company will increase the amount of information you have available to decide if you even need to ask for a take-home assignment. Not applying will all but guarantee that you don’t get a job there.

There’s also some alternatives/middle ground to gather more information before you apply if you’re not ready to take the leap.

Have you checked Glassdoor to see if the company is listed and if anyone has reviewed their interview process?

Do you know anyone who works there? Can you ask your contacts/social media contacts, or if you have a

LinkedIn account, see if any 1st or 2nd level contacts currently work there or have recently worked there?

Might be a good way to gather information.

You can also always phrase your questions to others about how the process goes, so that you don’t feel like your only choice is to ask permission of the hiring manager during the interview process, which can feel like a time of higher stakes. Bring down the stakes of the question by changing the question and who you ask.

Hope something above helps you!

OP
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
Thanks so much for this! I definitely agree that I tend to not do something because of a number of "might be true" reasons. I did check Glassdoor and there weren't any reviews about software engineer interviews but I do have a contact there that I asked and they let me know that it was a live coding assessment with one of the senior engineers. I have been in touch with the HR contact as well and she said she'd be happy to talk with me about any questions before applying, so I have an option to bring that up with her. Just not sure if asking for a take-home will be viewed negatively. However, to your initial point, the "worst case scenario" is that they say no and I just have to give the live exercise a shot.

###### Commenter_9
ID: REDACTED! ~(o.o)~ <3, Upvotes: 4
If you can't do technical interviews, don't.
Them: "Could you do a technical problem here in front of us?"
You: "I don't perform well in technical interviews, my skill set is in writing clean efficient code and overcoming challenging problems. This takes time and thought and the first approach may not always be the best solution, nor would I ever commit to putting a 30 minute solution into production. If you would like to learn more about how I program effectively, i would be glad to discuss the tools and skills I use with you to give you a better understanding."

Any programmer we've ever hired that performs well in a technical interview performs TERRIBLY in the team, poor personal skills, "big brain syndrome" and generally stubbornness and pushing lots of bad things and ignoring team standards and policies.   The more reserved programmers who are less trigger happy function much better FOR THE TEAM.   I'm not saying one is better than the other, but they have their places. It depends on the environment.

Here's my other piece of advice, don't let them run the interview, they're going to have questions, but fire back and take over. Have more questions than they do and grill them.   Let them know you're serious and you mean business, you don't NEED this job and you're only willing to accept it if it is a better move and less stress for you, make it clear. The kinds of questions I ask.

Is there a company computer?
If not, is there an equipment stipend?

Is it mac, windows, or linux environment?

Is the language mandated? (python, c#)

How long has the current team been in place? What's the length of time of the shortest member? When/Why did the last dev leave the team?

What SCM do they use? Are tests automated? What's the merge review process like?

Is it a bug-driven environment? New features requests? Is help desk tickets involved?

How is team communication? How often are meetings? Are there morning standups? Is there a scrum board? Are there reports or reviews of employees? Are the deployments blue/green? Who is in charge of the deploys? How many people can do deploys?

Who controls the infrastructure?
Are there raises? Performance compensation? How is PTO? Paternity/Maternity leave? What's the PTO policy for requesting time off? HOw many days in advance? Sick time ? How many days in advance? Doctors note required?

There's about a million more questions, but think about everything you like and hate about your job and ask it. Think about your "dream job" and then figure out why it's your dream job and ask about that.

Beyond that, make sure you get along with the team.   Is it standard to receive communication outside of normal business hours? What are business hours?

Basically, the thing you need to drill into you're head, they are not interviewing you. You are interviewing. Remember, you did not put out your resume on the internet shouting HIRE ME I NEED A JOB, they put out a job listing shouting "OMG WE NEED SOMEONE PLEASE ANYONE". They are desperate, they need and employee, you don''t need a job. You are in control, they are not interviewing you, you are interviewing them. Once you figure that out and live by that, you will ace every interview and get any job.

One last piece of advice: Make them laugh. In my experience anytime I've got the interview team to laugh, I've gotten the job. If not, you probably don't click with the team and it's not a good fit.

And my parting wisdom: Never, ever, ever lie. If they ask you something you aren't sure about, just be honest. "I'm not familiar with that phrase, term, acronym, can you clarify it so I can understand the question better?"

And answering "I would google it" is a valid response to an approach, everyone copy/pastes from stack overflow. It's knowing what to copy/paste that is the skillset.

Also, take a notepad and pen, take notes during the interview, it mentally makes them think they've already hired you (I've gotten every job I've ever taken a notepad to and taken notes, they're usually not even good, I just write down key points and recap at the end, then ask what the next steps are and write them down before I leave.

Best of luck, you got this!

> ###### OP
> ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
> Wow...this is super helpful! Thanks for sharing all this and I'll definitely have to read through this a few times to really soak it all in. But really great questions to ask and overall perspective to have.

###### Commenter_10
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
The company i applied for just sent me a mail for “live coding interview” that would took 2 hours for the 3rd step of iterview. I just felt uggghhtt no. Just give me the assignment and let me do my own pace.

> ###### OP
> ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
> Oof, that would be a lot. I hope some of the tips others have noted are helpful for you. Good luck with it!

###### Commenter_11
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
I just went through this process and it’s remarkably easy once you understand what they’re looking for. Most engineering technicals boil down to a certain engineering problem that is presented as a run-of-the-mill problem but those problems are expected to be looked at in an abstract way. In an interview a few weeks ago I was asked to re-implement document.getElementsByClass or something like that. The problem is in a tree. The target element where you start searching would be the trunk of that tree, any child elements are the branches or if they don’t have children then they are considered leaves. They were expecting me to explain it this way then solve the problem with recursion, then make one final step to optimize. That’s the formula for the technical interview question. Most companies follow that format. They say that they are open to people without traditional backgrounds then they test you on some graph search algorithm that only people from traditional backgrounds studied in 3rd year of college. Get comfortable with breaking the problems down into parts and recognizing which part is what and what it takes to solve that problem. Also, they almost always want to see you do a brute force implementation of something then optimize it.

> ###### OP
> ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
> Great reminder that breaking down what the actual problem is and going from there is essential.

###### Unknown_User
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
[deleted]

> ###### OP
> ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
> Thanks for the suggestion - I hadn't heard of leetcode before but I'll check that out.

###### Commenter_12
ID: REDACTED! ~(o.o)~ <3, Upvotes: 2
The approach I like to take is to first prepare by learning as much as possible about common data structures and algorithms, like:

- Trees (esp binary)
- Graphs (DFS, BFS, etc.)
- Stacks
- Queues
- Heaps
- Sorting
- Linked Lists
- Array manipulation
- Recursion

First, I'll make sure I understand the question and if there are any applicable edge cases. Then I briefly discuss a brute-force solution though I might not implement it. Then if I see it right away, I'll point out where the brute force solution is duplicating effort and find a way to save some time and/or memory. If I can't see where my efforts are being duplicated or I have no idea how I can further optimize the solution, I'll try to run through several of the algorithms or data structures above and briefly think through "how will [algorithm or data structure X] help solve this problem" until I find one that clicks; I usually won't go through the whole list and may eliminate some obviously irrelevant ones on the spot. I might go through this thought process with my interviewer until I hopefully hit on one that seems promising and I'll think through how to utilize that particular data structure to solve the problem. Once I have a reasonable solution thought out I'll get the goahead from the interviewer before writing the code.

###### Commenter_13
ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
No advice to give as I'm a noob. Just wondering how people write out your code when solving it outside an IDE. Paper and pencil, white board?

###### Unknown_User
ID: REDACTED! ~(o.o)~ <3, Upvotes: 1
I've been in a technical career for around 20 years and only got diagnosed and medicated about 6 years ago. One thing I learned is that a huge number of us in co-development world that have similar neuro atypical conditions so what that means is that the very experienced interviewers in this realm, including myself, know comes with the territory

If I were you I would take advantage of your hyperfocus and try to trigger it on studying interviewing techniques. I mean if we have to "suffer" with ADHD we may as well take advantage of it.