Click Here

Its 2025 and most top tech companies are still asking LeetCode style questions in coding interviews.But getting started on LeetCode is harder than ever. With over 3,000 problems, its easy to feel overwhelmed and lost.How do you even start? Which problems should you solve? How many problems are enough for coding interviews?How much time should you spend on each problem?I will answer these questions and more in this article to make your journey smoother, and lot less less painful.For context - Ive solved more than 1,500 LeetCode problems and cleared interviews at multiple big tech companies including Amazon, Google, and Microsoft and in this article Ill share everything Ive learned to help you start and navigate LeetCode more effectively.Ill share practical tips and resources to help you save time, stay focused and build your problem-solving skills without feeling overwhelmed.You might be wondering: why is everyone doing LeetCode questions. Is it really necessary to land a Software Engineering job?The short answer is: not always.There are plenty of startups and smaller companies that focus more on your experience with specific tech stacks and the projects youve built. They might not even include LeetCode style questions in their interview process.But if your goal is to work at big tech companies like Amazon, Google, or Microsoft, you would need to practice LeetCode style questions since thats what they ask in their interviews.This is a question I get asked all the time.Many people get stuck at deciding which programming language to use.But heres the truth: It doesnt really matter. A programming language is just a tool. Once you understand the approach to solving a problem, you can implement it in any language.In my experience giving interviews, unless the job specifically requires expertise in a certain language, youll be fine using any language youre comfortable with.The point is, LeetCode isnt about syntax. Its about using the right data structures, algorithms, and your ability to think critically and solve problems.That said, if youre new to coding, I recommend starting with Python. Its beginner-friendly and has a simpler syntax.If you already know a languagestick with it. Whether its C++, Java, C#, JavaScript, TypeScript or Go, theres no need to switch.You dont need to be an expert in your chosen language, but you should know the basics like:variables and data typesloops (for and while)if-else conditionsarrays and stringsfunctions and classesand input/output operationsBeyond the basics, spend some time learning the built-in libraries for common data structures and algorithms.These libraries save time and let you focus on solving the problem rather than reimplementing data structures from scratch.For example:Python has lists, dictionaries, and sets.Java has ArrayList, HashMap, and PriorityQueue.C++ offers the STL library with vector, map, and set.Before diving into LeetCode problems, its good to familiarize yourself with the fundamentals of data structures and algorithms.Start by understanding:Big O notation and Time ComplexityBasic Data Structures like Arrays, strings, linked lists, stacks, queues, hash tables, and binary trees.and Fundamental Algorithms like Sorting techniques, binary search, and recursion.Recursion is particularly important, since many problems specially tree and graph related ones rely heavily on it.When youre familiar with these topics, it becomes much easier to recognize which concept applies to a specific problem.You dont need to dive too deeply into every topic upfront. Most of your understanding will develop naturally as you solve problems.There are plenty of resources available online to help you get started. Dont waste time searching for the perfect resourcepick one and start learning.The focus should always be on progress, not perfection.Here are some resources I personally found quite useful:For DSA Patterns: AlgoMasterIO: Its my second channel where I upload animated tutorials on DSA and LeetCode patterns.Heres a comprehensive list of DSA topics you should know for coding interviews:ShareIf youre a beginner, focus on one topic at a time to avoid feeling overwhelmed.For example: Start with arrays, the move to strings, then progress to more complex topics like linked lists, hash tables, and binary trees.This approach helps you develop a deeper understanding of each topic and teaches you how to recognize when to use a particular data structure or algorithm.Once youve learned the basics of a topic and understand how to implement it from scratch, its time to put your knowledge to the test.Solve 4-5 easy problems related to that topic on LeetCode.This will:Reinforce your learning.and Build confidence in applying the concepts.If you want to learn more about how to master a DSA topic, you can check out this article where I discuss it in more detail:If you open LeetCode, there are over 3,000 problems.That number alone can feel overwhelming. So, how do you decide which ones to solve?Start with easy problems for each topic. These are perfect for building confidence and understanding the basics of problem-solving.Once youre comfortable with the easy ones, gradually challenge yourself with slightly harder problems that push you beyond your comfort zone.Remember, real growth happens when you constantly challenge yourself.Dont worry about hard problems in the beginning.Most coding interviews focus on medium-level problems, so thats where you should spend the majority of your time.LeetCode also provides curated lists like the Top 100 Liked Problems and Top Interview 150 Problems.These lists are excellent, and I highly recommend solving every problem on them.Theres no magic number, but from my experience, 300 well-chosen problems is the sweet spot.However, its not about solving just any 300 problems. Focus on high-quality problems that cover the most topics and patterns.To make it easier for you, Ive created a free resource: a curated list of the Top 300 LeetCode Problems organized into 60 topics and patterns that are most commonly asked in coding interviews.You can find it on algomaster.io. Just head to the practice page. You will see a structured list of problems that you can follow from top to bottom.Here youll find:Resources to help you learn the topics.Ability to track your progress and mark problems for future revision.Links to GitHub and YouTube solutions.and ability to filter problems based on pattern, difficulty level and status.Dont rush through problems just to increase your count. Instead of focusing on the quantity of problems you solve, focus on what you learn from each problem.Its far better to deeply understand and solve 50 problems than to superficially solve 500 problems.After solving a problem, ask yourself:Why does this solution work?Whats the one key insight that made everything else easier?Also, try to make it fun. There is a fun in learning to solve coding challenges. Its not just about getting a job.The key to getting better at LeetCode is learning as many patterns as possible.Instead of focusing on individual questions, focus on identifying underlying patterns that connect similar problems.On LeetCode, youll come across multiple problems that follow a similar patterns. Once you solve one of them, you can apply the same approach to solve others.For example: After learning the monotonic stack pattern, I was able to solve over 10 similar problems.I made a video covering the 15 most important LeetCode patterns I learned after solving over 1,500 problems. You can check it out later.On algomaster.io, Ive categorized problems by pattern. This makes it easy to focus on one pattern at a time. By going through the list, you will get to know all the important patterns for coding interviews.The first time, focus on getting the big picture. What is the problem asking you to do?On the second read, pay attention to specific constraints and conditions. These small details often provide clues for optimizing your solution.Carefully go through the examples provided in the problem.Walk through the input and output step by step to understand how the problem works. I have found that, many times simply walking through a few examples helped me figure out the solution.Many problems specially the ones related to trees and graphs are much easier to understand when you draw them out.Try to take examples that cover different scenarios and input sizes. Think about any edge cases that might come up.Dont expect to come up with the most optimal solution right away.First see, if you can solve the problems using a brute force approach. While it might not be efficient, it gives you a baseline to improve upon.Once you have a brute force solution, focus on optimizing it.Here are few things to consider:Leverage Unused Information: Look for details you might have missed. For instance, if the problem states that the array is sorted, consider leveraging this to use binary search or a two-pointer approach.Precompute Information: If certain calculations are repeated multiple times, consider precomputing them. Use techniques like prefix sums or frequency counts to avoid redundant calculations. Hash tables are widely used in interview questions and should be at the top of your mind.Make Time vs. Space Tradeoffs: Sometimes, using additional memory (e.g., hash tables) can speed up your solution.Data Structure Brainstorm: Run through the popular data structures and try to apply each one to the problem at hand.Avoid overcomplicating your solution. At every stage ask yourself, "Is there a simpler way to do this?". This will not only make your solution easier to understand but it will also reduce the chances of errors during coding.Develop the habit of analyzing the time and space complexity of every problem you submit.When youre just starting out, even easy problems can take a while to solve. Thats completely normaldont get discouraged if you cant come up with a solution right away.Your main goal in the beginning should be to focus on learning and understanding the problem deeply. Real progress happens when you take the time to think, make mistakes and refine your approach.But, some problems involve specific tricks or patterns that are hard to figure out unless youve seen them before.Spending hours on such problems without progress can be counterproductive.Heres a good rule of thumb:Give yourself 30 to 60 minutes of focused effort for each problem.If youre still stuck, its okay to look at hints or solutions.Read the official solution and and try to understand the top voted solutions on LeetCode discussion forum.Dont just move on after viewing the solution. Try to grasp why the solution works.Rewrite the solution from scratch without looking at the code. Writing it yourself helps you internalize the approach and ensures you truly understand it.If you couldnt solve the problem on your first attempt, mark it for revision and revisit it after a few weeks. Revisiting problems helps reinforce your understanding and ensures long-term retention.One of the most common mistakesone that I made myselfis memorizing solutions to difficult problems and moving on.At first, it might seem like a shortcut, but in reality, its a big mistake for three main reasons:Youll quickly forget memorized solutions. Without understanding the logic behind them, they wont stick in your memory.Your problem-solving skills wont improve. Memorization skips the critical thinking process, which is the real value of practicing coding problems.Youll struggle in interviews. Most interviewers ask variants of problems or follow-up questions, and without a deep understanding, you wont be able to adapt.Instead of memorizing, focus on understanding the solution, even if it takes hours.Break it down step by step and ask yourself:Why does this approach work?Whats the key insight that simplifies the problem?How can I apply this approach to similar problems?This effort pays off in the long term. The deeper your understanding, the more confident youll feel explaining your thought process during interviews.Solving a problem during an interview is very different from solving it from the comfort of your home.Thats why its a good idea to time-box yourself and practice in a timed environment after you have learned the basics and solved 100-150 problems.Simply turn on a timer while practicing and try to complete the problem within that time.For easy problems, aim to complete them in 10-15 minutes.For medium problems, set a timer for up to 30 minutes.For hard problems, allow yourself up to an hour.To take your practice a step further:Participate in LeetCode contests.Try virtual contests by simulating past LeetCode contests.Its okay if you struggle to solve even one problem initiallythis is normal. The goal is to gain experience solving problems within a time limit.When I started participating in contests, I could only solve 1-2 problems. But with consistent practice, I improved to solving 3 problems, and eventually, I was able to solve all of them in some contests.Getting good at LeetCode isnt just about learning new concepts or solving problemsits also about retaining that knowledge over time.Thats why you should regularly revisit concepts and problems youve already solved, especially the ones you found challenging.Try to solve those problems again without looking at the solution.By practicing consistently and revisiting old problems, youll make it much easier to retain what youve learned over time.I discuss more about how to effectively revise LeetCode problems in this article, so make sure to check it out: Getting good at LeetCode takes time.Some topics might take weeks or even months to master, and thats okay. Be patient with yourself and remember: Its completely normal to feel stuck or frustrated when working on a tough problem or grasping a complex topic.If a problem feels too hard, take a break, then come back to it with a fresh perspective.The more you practice, the better you will get at solving LeetCode problems.Thank you for reading!If you found it valuable, hit a like and consider subscribing for more such content every week.If you have any questions or suggestions, leave a comment.This post is public so feel free to share it.ShareP.S. If youre finding this newsletter helpful and want to get even more value, consider becoming a paid subscriber.As a paid subscriber, you'll receive an exclusive deep dive every week, access to a comprehensive system design learning resource , and other premium perks.Get full access to AlgoMasterThere are group discounts, gift options, and referral bonuses available.Checkout my Youtube channel for more in-depth content.Follow me on LinkedIn, X and Medium to stay updated.Checkout my GitHub repositories for free interview preparation resources.I hope you have a lovely day!See you soon,Ashish I want to tell you a story about how I started from the level of "can't solve even 1 easy problem out of 10" to the level of "can solve every other medium problem" and went through several coding sessions at companies like Meta, Booking, Careem, Avito... It all started at the end of 2022 when I firmly decided that I wanted to land a job at a FAANG company. However, as you probably know, this requires knowledge of algorithms and data structures, which I lacked because in regular work, you don't need heap, tree, disjoint set, and other data structures. So, I had to start almost from scratch. It was very challenging since I didn't know where to begin, how to approach problem-solving, what was important, and what wasn't. I also had questions like how many problems to solve to feel confident and if I could even progress to medium-level problems. Solving even easy problems was puzzling, and below, I'll try to help you with this. The first thing that comes to mind when you want to start solving LeetCode problems is to open the problems page and start solving them sequentially. However, it's not a great idea, even if you filter for easy problems, because many tasks labeled as easy are quite challenging. This can be demotivating and even lower your self-esteem. Tips:First tip - pay attention to the Acceptance rate; the higher, the better. This indicates that many people manage to solve the problem, increasing the likelihood that you can too. Additionally, you can explore the study plan and try solving problems from there:Top 100 Liked: A list of highly liked problems. If there are many likes, there's a chance it's not a convoluted problem that could lower your assessment.Top Interview 150: A list of the top 150 problems for interview preparation. You're likely to encounter these in interviews, but some might be challenging, especially early on.LeetCode 75: Described as a "Must-do problem list for interview prep." These problems are categorized into different topics; solve them following the next piece of advice.Second tip - start with the most fundamental data structures like arrays (one-dimensional/two-dimensional), set, hashmap; you'll encounter them frequently. For instance, in many problems, execution time can be improved by adding a set or hashmap (though keep in mind this may increase memory consumption). Only after mastering these should you delve into other structures like stacks, queues, linked lists, trees, and graphs.On LeetCode, there are cards providing explanations of data structures and algorithms, along with problems to reinforce these concepts - LeetCode Explore. As mentioned earlier, begin with simple cards like arrays and strings, gradually increasing difficulty. I revisited these cards several times because some concepts weren't clear initially. Don't think there's anything wrong if something is unclear; return later, and you might find you now understand it.Here are some additional resources:LeetCode Learn: Cards with explanations and problems.Neetcode Roadmap: A roadmap with topics and problems; highly recommended.Tech Interview Handbook: Create a weekly plan using various filters.LeetCode Patterns: A list of 150 popular problems categorized by topics, difficulty, and companies.Third tip - if you can't come up with an optimal solution, start with a "brute-force" solution. Once you have that, think about how to improve it.Can you use a specific data structure?Can the array be sorted?What if you use a set or hashmap?Is there a template you can apply?If nothing helps and you've already spent 20-40 minutes, confidently open the solution and study it. After doing so, close the solution and try to solve it again (this may take several iterations, which is normal). If you're still unable to solve it after multiple attempts, consider typing out the solution (I've done this several times). However, don't forget to revisit the problem later and try solving it again. It's entirely possible that you might not succeed initially, and that's okay. Just repeat the steps until you're confident.Fourth tip - go through these 14 patterns, especially focusing on two pointers, sliding window, and fast & slow pointers, as they are often encountered. Take a pattern, find problems under that pattern (on the LeetCode problems page, use the filter for tags like "Two pointers" or "Sliding window"), and solve enough problems to solidify your understanding of that topic.Fifth and most important tip - practice, practice, and practice again. In the beginning, I often felt like crying from frustration because I couldn't even solve simple problems. This is not an exaggeration. The thought of "I'm so dumb" lingered in my mind for the first few months. Even after solving 600 problems, there are still moments when self-esteem drops, especially if I can't solve a seemingly easy problem. What's important is that with each solved problem, you'll become better relative to yourself. Solving LeetCode problems is now addressed on the platform Firecode, where problems are presented one after another, starting with easy ones and revisiting previously solved problems for reinforcement. There are explanations for various topics, and after solving a problem, you can see how others approached it, which is very helpful for understanding different solution strategies. FAQ: What should I do if I can't solve a problem? If you can't solve a problem after 30-40 minutes, look at the solution, try to find a video solution for that problem. After reviewing and understanding, attempt to solve it again. If unsuccessful, repeat the process. Periodically revisit old problems, especially those that were challenging. I've sometimes solved the same problem 10-15 times or more. How many problems should I solve before applying to jobs? It varies individually. I spent six months solving a few problems every day, around 300 problems, before feeling a solid foundation. Confidence came after solving around 400-500 problems. Some may only need to solve 100-200 problems, but unfortunately, I'm not among them. Which programming language is best for problem-solving? In short - Python. I started with PHP, switched to Go, then Java. After watching a video, I tried Python and have been using it since. Python's concise code, built-in functions (like Counter, defaultdict, divmod, lambda, etc.), and speed are advantageous during interviews. How to find time for problem-solving? I aimed to solve two problems daily, one before work and one after. On weekends, I dedicated more time, watching videos, reading articles, solving more problems, and revisiting old ones for reinforcement. Is it worth buying LeetCode premium? The most useful feature of premium is seeing which companies gave a particular problem. If preparing for a specific company, it's recommended; otherwise, the free version is sufficient. Video explanations can be found on channels like Neetcode. Is it worth buying access to educative.io? While there are many recommendations for the Grokking Coding Interview Patterns course on educative, I didn't find it very useful. Visual explanations were the only positive aspect. In my opinion, watching YouTube videos is a better alternative. Is it worth buying access to algoexpert.io? In my view, algoexpert.io is a very useful resource. Each problem comes with detailed explanations in videos lasting 30-50 minutes. They often have promotions, and you can sometimes get all courses for a year for $99. It's reasonably priced, has a pleasant website, numerous test cases, and a user-friendly editor. Results:Achieved my initial goal for solving algorithms - passing coding interviews, a crucial step in FAANG companies. Successfully went through several interviews, and most importantly, no longer fear this stage.Deepened understanding of data structures. From not knowing what a heap is, I now have a comprehensive understanding of this structure and others.Changed my perspective on code during work. Now I can identify areas to use more memory efficiently or reduce time complexity.Last but not least - brain fitness. Transitioned from "don't want to do it" to "want to solve problems every day." Useful Links: Russian version: Want to ace coding interviews and improve your problem-solving skills? Start with LeetCode. This platform offers a structured way to learn, practice, and master coding challenges. Heres how to get started effectively:Focus on Fundamentals: Learn key data structures (arrays, trees, graphs) and algorithms (sorting, recursion, dynamic programming).Start Simple: Begin with easy problems, then progress to medium and hard challenges.Use the Right Tools: Leverage LeetCode features like Explore cards, problem filters, and discussion forums.Practice Patterns: Master common coding patterns like Sliding Window, Two Pointers, and Dynamic Programming.Stay Consistent: Solve problems daily, review past solutions, and track your progress.How I would learn Leetcode if I could start overStarting on LeetCodeTo succeed on LeetCode, you need a focused approach that builds strong fundamentals. Heres how to get started effectively.Mastering Data Structures and AlgorithmsUnderstanding data structures and algorithms is key to solving problems on LeetCode. Start by mastering essential data structures like arrays, linked lists, trees, and graphs. Pair this with learning fundamental algorithms such as sorting techniques, binary search, recursion, and graph traversal.Some key areas to dive into:Working with arrays and stringsManipulating linked listsTraversing treesExploring graph search methodsChoosing the Right Programming LanguageSelecting a programming language is an important step. Python is a great choice for beginners because of its simple syntax and extensive libraries [1][5]. However, your choice should depend on your familiarity with programming, the languages demand in the job market, and its built-in support for data structures.Factors to consider:Your prior coding experiencePopularity of the language in tech rolesEase of implementing algorithms and data structures Once youve created an account, take advantage of the platforms tools like filters, Explore cards, and community forums. These features can help you streamline your learning process.FeaturePurposeProblem CategoriesFocus on specific topics to address weak areasDifficulty FiltersStart with easier challenges and gradually level upDiscussion ForumsGain insights from community solutions and tipsLeetCode Explore cards are especially helpful, offering guided learning paths that combine theory with practical problems [4].With these basics covered, youre ready to dive into solving problems and improving your skills step by step.Improving Problem-Solving SkillsImproving your problem-solving skills on LeetCode requires a clear and organized approach. By combining logical thinking with recognizing common patterns, you can sharpen your coding abilities. Lets break it down.Using a Structured Problem-Solving MethodThe first step is to carefully analyze the problem statement. Break it into smaller parts and focus on the critical aspects:StepWhat to Look ForInput AnalysisData types, size limits, edge casesOutput RequirementsExpected format, validation rulesConstraintsTime/space complexity, special conditionsTest CasesTypical inputs, edge cases, unexpected scenariosSketching out your solution beforehand can help you spot potential problems and guide your implementation process.Practicing Key Coding PatternsFamiliarize yourself with common coding patterns like Sliding Window, Two Pointers, and Dynamic Programming. These patterns are frequently used in interviews and can simplify a wide range of problems.PatternWhere It HelpsSliding WindowString or array operationsTwo PointersArray manipulation, linked listsDynamic ProgrammingOptimization problemsGreedy AlgorithmsScheduling, resource allocation"Focus on fundamental data structures like arrays, sets, and hashmaps." LeetCode Solutions [4]Progressing from Easy to Hard ProblemsStart with easy problems to build a solid foundation, then gradually work your way up to medium and hard challenges. Use LeetCodes Explore cards or curated problem lists to practice systematically.DifficultyKey Topics to Focus OnEasyArray manipulations, basic string operationsMediumTree traversal, graph algorithmsHardAdvanced dynamic programming, complex optimizations"Practice regularly: Even if you cant commit a lot of time every day, try to solve at least one problem a day." LeetCode Discuss [3]If you get stuck, review solutions from others, but make sure to understand the logic behind them before moving forward. These strategies will help you consistently improve and build mastery on LeetCode.sbb-itb-f454395Strategies for Mastering LeetCodeBuilding strong coding interview skills takes more than just solving random problems. Here are some strategies to help you make steady progress on LeetCode.Maintaining a Regular Practice ScheduleSet aside at least an hour daily or 57 hours weekly to work on problems. Focus on recognizing patterns, improving solutions, and solving problems of mixed difficulty across different topics. Make it a habit to review 1015 previous problems each month to strengthen your understanding of common patterns and optimization methods."Dedicating one hour each day to solving problems can be more beneficial than solving problems for seven hours on one day." [2]Consistency is key, but it works best when paired with regular reflection and revisiting past challenges.Reviewing and Revisiting ProblemsGo back to problems youve already solved to reinforce important concepts and uncover recurring patterns. Keep a log of your solutions, approaches, and insights. Use this log for weekly or monthly reviews to deepen your understanding and refine your problem-solving techniques.Review StrategyPurposeWeekly ReviewReinforce recent concepts and patternsMonthly Deep DiveAnalyze tough problems and explore alternative methodsPattern-based ReviewGroup similar problems to strengthen pattern recognitionLearning from Solutions and DiscussionsAfter solving a problem, take time to study editorial solutions and community discussions. These resources can help you discover alternative approaches, handle edge cases, and improve optimization. Re-implementing solutions is a great way to solidify your understanding. Pay attention to the logic behind the solutions, the data structures and algorithms used, and any optimization techniques mentioned."Focus on active improvement, recognizing patterns from previous solutions, and maintaining a growth mindset." [1]Using LeetCode Features and Additional ResourcesOnce youve got a solid practice routine in place, tapping into LeetCodes tools and external resources can give you an extra edge. These options work alongside consistent practice to help you grow faster and smarter.Leveraging Curated Problem ListsCurated problem lists like Neetcode and LeetCode Patterns are great for beginners. They provide a clear path by organizing problems based on topics and difficulty levels. This makes it easier to build essential skills step by step.Collection TypeFocus AreaBenefitsNeetcode RoadmapCore AlgorithmsGuides you through key algorithm topicsLeetCode PatternsCompany-specific ProblemsPrepares you for employer-focused questionsAdding Educational Platforms to the MixLeetCode is excellent for sharpening problem-solving skills, but other platforms can help round out your learning. For instance, KodNest lets you work on hands-on projects and apply the theoretical knowledge youve gained. Plus, their job placement support can help you transition from practice to professional opportunities."Focus on active improvement, recognizing patterns from previous solutions, and maintaining a growth mindset." [1] Broaden your knowledge with resources like Cracking the Coding Interview for interview tips and strategies. You can also use tools like LeetCode Enhancer, a browser extension that helps you filter locked problems and organize your study sessions. These extras can streamline your learning process and make your practice more effective.Conclusion: Mastering LeetCodeMastering LeetCode is all about building solid skills in algorithms and problem-solving. It provides an excellent platform to sharpen the technical abilities required for coding interviews and real-world programming tasks.To succeed on LeetCode, focus on these three key areas:Building a Strong Base: Start with fundamental data structures and algorithms, then move on to more advanced topics. Pay attention to recognizing patterns and improving your solutions efficiency [4].Focused Practice: Combine learning core concepts with solving problems regularly. Short, consistent practice sessions are often more effective than cramming in long, sporadic ones [1].Learning from Review: Keep improving by revisiting solutions, exploring alternative methods, and fine-tuning your problem-solving techniques [1] [4].Progress on LeetCode comes from steady effort and thoughtful use of its resources. Track your growth, revisit tough problems, and deepen your understanding of the core topics to make the most of your learning experience.Related posts My LeetCode ProfileLeetCode has become the go-to platform for aspiring software engineers and developers to hone their problem-solving skills and prepare for technical interviews. However, the vast array of problems can be overwhelming for newcomers. In this article, well provide a comprehensive guide on how to approach LeetCode problems step by step, from selecting the right questions to mastering your problem-solving skills.Step 1: Setting the Foundation1.1 Understand the BasicsBefore diving into LeetCode problems, ensure you have a strong grasp of fundamental data structures and algorithms, including:ArraysLinked ListsStacks and QueuesTrees and GraphsSorting and Searching AlgorithmsDynamic Programming1.2 Learn Time and Space ComplexityLeetCode categorizes problems into three levels of difficulty: Easy, Medium, and Hard. Start with Easy problems to build your confidence and gradually move on to Medium and Hard problems as you gain experience.Step 2: Selecting the Right Questions2.1 Categorize by DifficultyLeetCode categorizes problems into three levels of difficulty: Easy, Medium, and Hard. Start with Easy problems to build your confidence and gradually move on to Medium and Hard problems as you gain experience.2.2 Focus on Specific TopicsChoose problems related to specific topics you want to master, such as dynamic programming, binary search, or graph algorithms. This approach helps you deepen your knowledge in specific areas.2.3 Leverage Filters and TagsUse LeetCodes filters and tags to find problems that match your criteria. For example, you can filter by data structure or algorithm type, company-specific interview questions, or even by the most frequently asked questions in interviews.Step 3: Approach to Problem Solving3.1 Read the Problem CarefullyStart by reading the problem statement thoroughly. Understand the problem requirements, constraints, and input/output formats. Ensure you have a clear picture of what needs to be achieved.3.2 Break It DownBreak the problem into smaller subproblems. Identify any patterns or potential algorithmic approaches that can be applied. Sketch out your initial thoughts on paper or in code comments.3.3 Plan Your SolutionBefore writing code, outline your approach step by step. Consider edge cases and handle them in your plan. This will help you avoid common pitfalls and bugs during implementation.3.4 Write Code IncrementallyBegin writing your code incrementally, one step at a time. Test each part as you go, and make sure it works before moving on to the next. This approach makes debugging easier.3.5 Optimize and RefactorAfter you have a working solution, review your code for potential optimizations. Look for opportunities to improve time and space complexity. Optimization is a critical skill for technical interviews.Step 4: Test Extensively4.1 Create Test CasesDesign a set of test cases to validate your solution. Include both common cases and edge cases. LeetCode provides a platform to run your code against various inputs.4.2 Debug ThoroughlyIf your code fails any test cases, use debugging techniques to identify and fix the issues. Understand why the code is failing and make necessary adjustments.Step 5: Analyze Time and Space Complexity5.1 Time Complexity AnalysisAnalyze the time complexity of your solution. Be prepared to explain why your code runs in a particular time complexity, whether its O(n), O(log n), or any other.5.2 Space Complexity AnalysisSimilarly, analyze the space complexity of your solution. Understand the memory usage and explain the space complexity of your algorithm.Step 6: Learn from Others6.1 Participate in LeetCode DiscussionsEngage with the LeetCode community by participating in discussions. Ask questions, seek clarification, and share your knowledge. Learning from others can accelerate your growth.6.2 Follow Coding Blogs and NewslettersSubscribe to coding blogs and newsletters that regularly feature LeetCode problems and solutions. These resources often provide detailed explanations and tips for tackling specific challenges.6.3 Watch Tutorials and Coding VideosPlatforms like YouTube and online learning websites offer coding tutorials and walkthroughs of LeetCode problems. Watching these videos can help you understand various problem-solving techniques and strategies.6.4 Participate in Online Coding CommunitiesJoin coding communities and forums like Stack Overflow, Reddits/learnprogramming, or dedicated LeetCode forums. These platforms offer a wealth of knowledge, and you can benefit from the collective experience of developers worldwide.ConclusionLeetCode is a powerful platform for improving your problem-solving skills and preparing for technical interviews. By following this step-by-step guide, you can systematically approach problems, select the right questions, and enhance your ability to tackle challenging coding tasks. Remember that consistent practice and a growth mindset are key to success on LeetCode and in technical interviews.Happy coding !