Course Overview

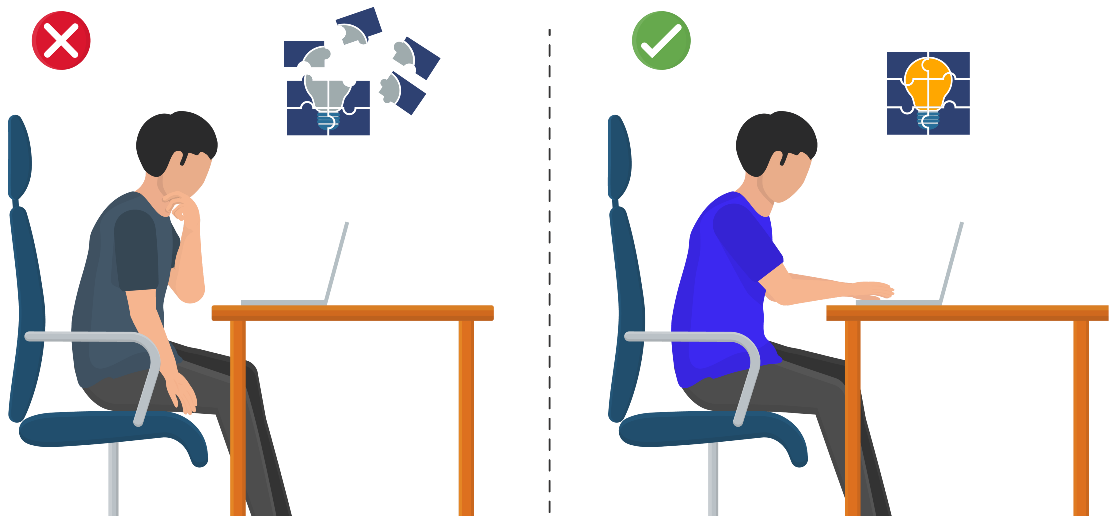
Get an overview of what this course is about.

We'll cover the following

Design patterns and why we need them

Coding interviews are hard. You never know what question you'll be asked, and there are so many of them! To make matters worse, the interviewer expects you to eventually (if not immediately) converge on an optimal solution.

The good news, though, is that there are a handful of "coding design patterns" that can help you. A coding pattern is a code blueprint that can be customized to optimally solve many related problems.



Identify the underlying pattern for different problems

You don't have to rely on rote memorization to learn the solutions to all coding interview problems. Most coding problems can be distilled into generalized versions. Mastering these design patterns gives you the tools to find the optimized approach to solve those problems. The two key things you need to know to achieve this are:

1. How each pattern works.
2. How to decide which pattern to apply to a given problem.

About this course

This course presents 175 of the most popular coding interview questions, organized as a set of 24 design patterns. Conventional wisdom recommends going through 100 or 200 difficult coding questions to prepare for a technical interview, yet without an adequate conceptual foundation, this approach is essentially just cramming for an exam. The purpose of classifying problems into design patterns is to help you build up a robust conceptual framework. This will help you to think critically about any new problem and to tackle it with the most appropriate techniques.

Each lesson drills down to focus on the core of a programming problem to present both the logic and the implementation of its recommended solution. We have included multiple interactive elements, such as puzzles and illustrations, to help build your understanding of the problem. From there, you can match it with the most suitable pattern, and construct an optimized solution.

Course Structure

Each of the 24 patterns in this course have been used to solve around 5 problems each, allowing you to understand the core of the technique and to experiment with its application to related problems.

The problems within each pattern are organized as follows:

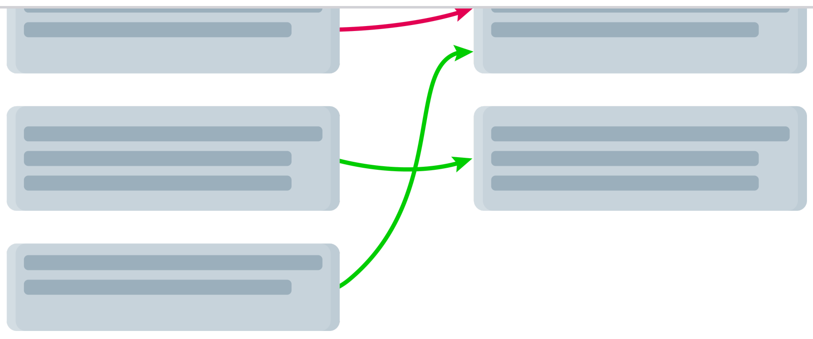
- **Category A** problems feature step-by-step solution construction so you may understand when, why, and how we design each logical building block of the solution.
- **Category B** problems showcase a deep dive into the solution, explaining every step in detail.
- **Category C** problems offer a brief overview of the solution, as well as the coded solution itself. Having guided you through three or four problems in this pattern already, this is where we start to take the training wheels off, while providing just enough support to help you analyze and understand the solution.
- **Category D** problems are designed to test your skill in applying the techniques in that pattern to solve a new problem. Help is now limited to clues and tips in order to spur your creativity.

Lastly, the **Challenge Yourself** chapter features a set of advanced interview questions commonly asked by the top tech companies.

Each problem in the course is presented as a pair of lessons. The first lesson presents the problem statement and the second presents its recommended solution. This is true for all except Category D problems and those listed in the Challenge Yourself chapter.

Strengths of this course

The course offers detailed introductions for each pattern, including a visual representation of how the pattern works, basic guidelines for using the pattern, and some real-world applications. These introductions broaden your understanding of problem-solving techniques and equip you with the necessary skills to interpret distinguishing features of the problem that can help you identify the underlying pattern.



Select the correct pattern

In the challenge lesson, we use a bottom-up approach to develop a step-by-step understanding of the problem. Illustrated examples—with color coding—help you visualize the input parameters and the corresponding output. Moreover, quizzes aid in confirming whether you've understood the problem correctly. Once you've clearly understood the problem, we provide the high-level, logical building blocks of the solution and encourage you to place them in the order required to solve the problem. This is an opportunity for you to come up with the basic outline of the solution strategy before diving into the code.

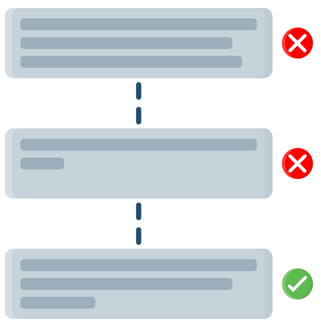


Figure out the algorithm

Lastly, a set of relevant code templates are provided for you to use at your discretion. This can make your code concise and save you the hassle of having to write the same code repeatedly, allowing you to focus on the logic of the solution.

In the solution lesson, guidelines for writing optimized code are provided by first describing the naive solution and then improving it using the techniques discussed in the pattern. A visual representation of the solution allows you to dive deep into the algorithm, tracking how variables and data structures are modified by it. The coded solution features sufficiently detailed in-line comments to help you understand th implementation of the design. Finally, the time and space complexities of the solution are stated, which provides you a useful indication of how the solution performs in the worst case. This helps you to develop the skills to effectively compare multiple solutions to the same problem, and to keep time and space complexity measures in mind when writing your own code.

A note about complexity

In this course, when reporting the space complexity of a solution, we do not count the space taken in memory by the input data or the output data. When comparing two correct solutions to a problem, the space taken up by the input and the output data will be precisely the same. Thus, counting this space does not serve the purpose of determining which solution is more space efficient.

Next →

Who Should Take This...