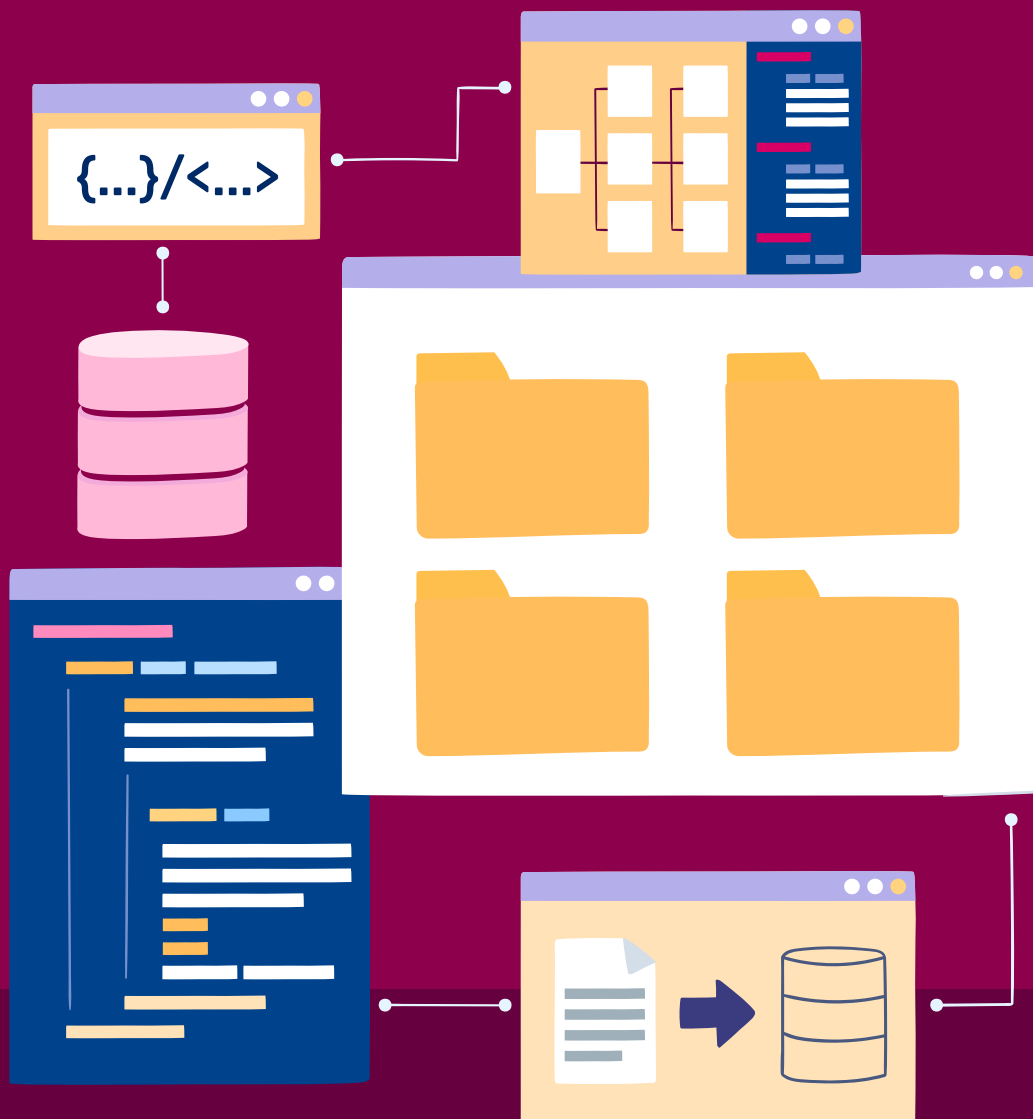


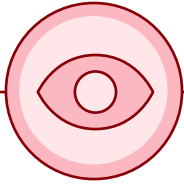
# 7 Steps for answering any \_\_\_\_\_

# DSA Questions

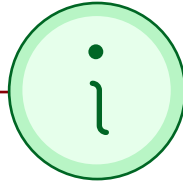


# 01

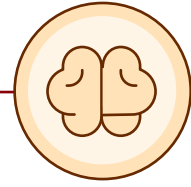
## Listen



Pay close attention to the problem description and ask clarifying questions.



Assume that all information provided is essential



Ensure you have a clear understanding of the problem statement

# 02

## Example

1

Review any given examples but treat them with caution

2

Check if the examples are too small or special cases

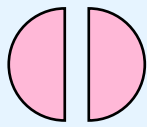
4

Examples are useful but may not cover all edge cases. Don't rely on them alone.

3

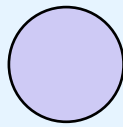
Debug the examples for correctness

# Tools of a System Thinker



Parts

VS

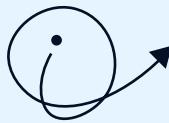


Wholes

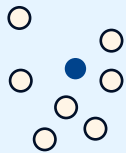


Linear

VS

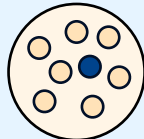


Non-Linear

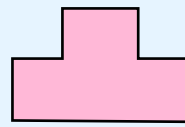


Analysis

VS

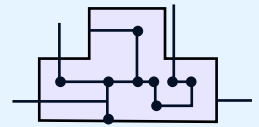


Synthesis



Structures

VS

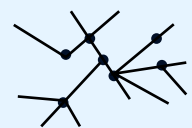


Processes



Heirarchies

VS

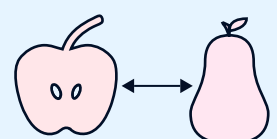


Networks



Objects

VS



Relationships

## 03

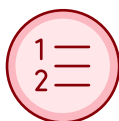
## Brute Force



Start by finding a brute-force solution



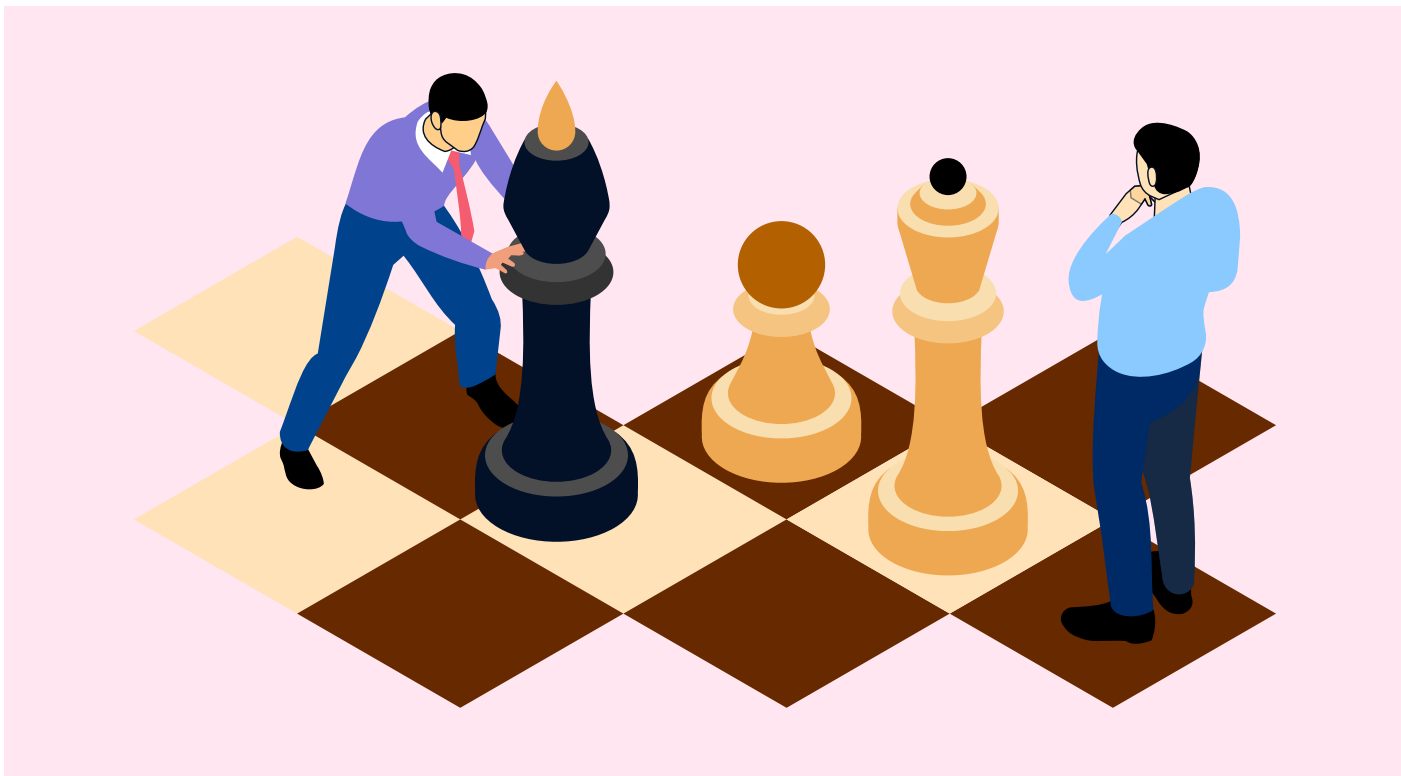
Don't worry about efficiency at this stage



State a naive algorithm and estimate its runtime



Avoid writing code at this point; focus on the logic



## 04 Optimize

1

Once you have a working brute-force solution, start optimizing

2

Identify **BUD**\*

3

Consider utilizing all the provided information

4

Experiment with different approaches

5

Explore **time vs. space trade-offs**

6

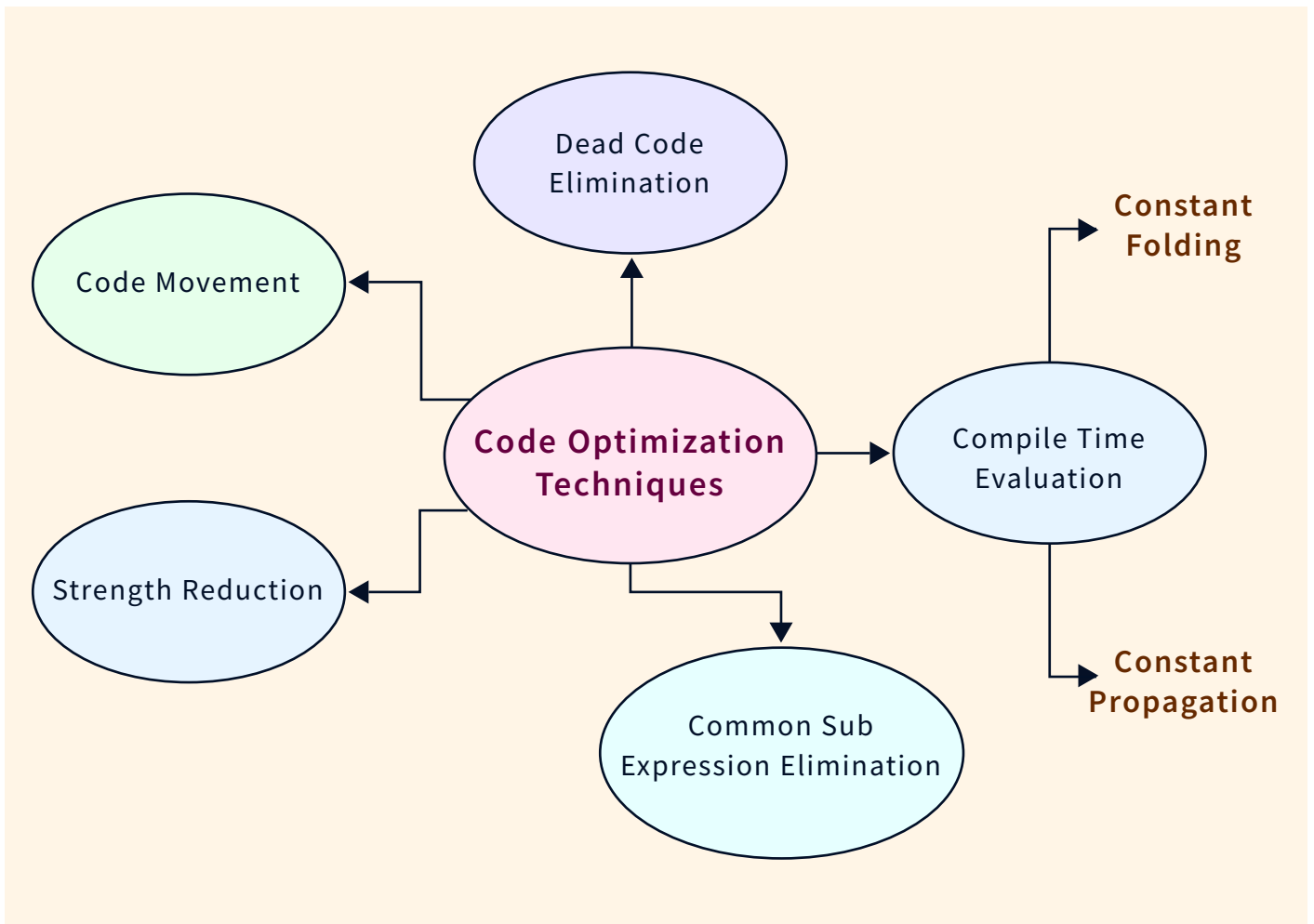
**Test and Refine** your optimization ideas

### **BUD**

**B**ottlenecks

**U**nnecessary Work

**D**uplicated Work



## 05 Walk Through

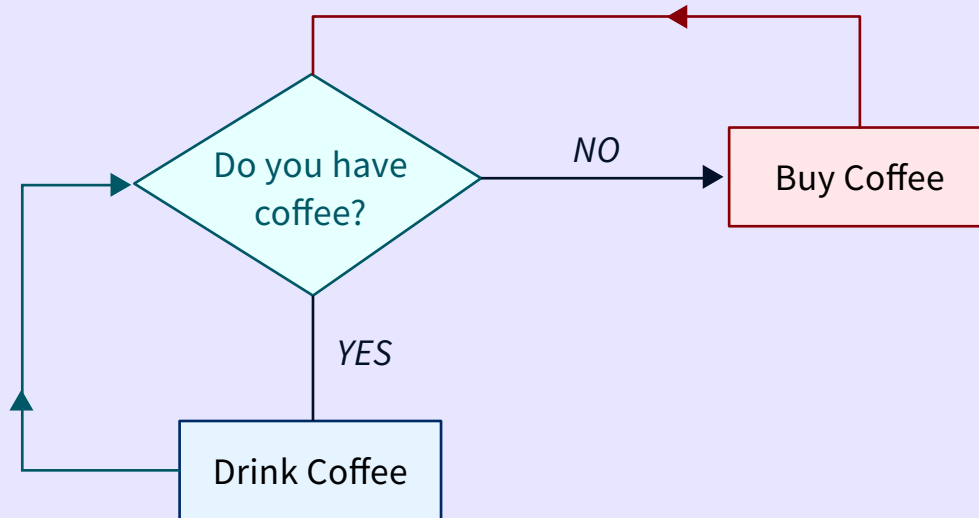
Understand your optimized solution thoroughly

Ensure you can explain each step of your approach

Reverse engineer your thought process if necessary

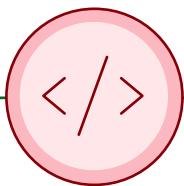
Verify that your solution is logically sound

Should I Drink Coffee? ☕

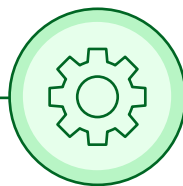


06

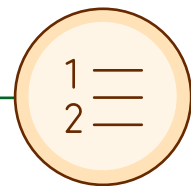
## Implement



Begin writing code,  
focusing on creating clean  
and modular code



Refactor as you go to  
maintain code quality





Implement your optimized  
solution step by step

- 1 Start with a **conceptual test**, walking through your code as if you were reviewing it.
  - 2 Test any **unusual** or **non-standard** code paths
  - 3 Pay special attention to **hotspots**, like arithmetic operations or handling null values, and variable overflows
  - 4 Begin with small test cases for **quick debugging** and **ensure correctness**
  - 5 Test with special cases and edge cases to validate **robustness**
-

# SCALER TOPICS



Unlock your potential in software development with  
**FREE COURSES** from **SCALER TOPICS**!

Register now and take the first step towards your future Success!



**PRATEEK NARANG**

**C++ for Beginners**

 5.9k enrolled  **Free**



**TARUN LUTHRA**

**Java for Beginners**

 6.8k enrolled  **Free**

That's not it. Explore 20+ Courses by clicking below

[Explore Other Courses](#)

Practice **CHALLENGES**  
and become 1% better everyday



**CIFAR-10 Image Classification Using PyTorch**  
Article

 No. Of Questions : 3

[Go to Challenge >](#)



**How to Build a Snake Game in JavaScript?**  
Article

 No. Of Questions : 3

[Go to Challenge >](#)

[Explore Other Challenges](#)