Problem-Solving Approach

Problem Understanding [7-10 minutes]

1. Pay attention to how the problem is presented
2. Clarify with a few ***examples*** [Good time to start thinking about corner test cases]
3. Take note of vital information [Focus on the ***properties*** and constraints]
4. Think hard about and state any ***assumptions*** you might have taken
5. Ask for clarification / missing information

Discuss and analyse Brute Force [2 minutes]

1. State the brute force solution, followed by a quick ***complexity analysis***
2. Comment on what parts of the brute force solution you think can be optimized

Ideate [5-10 minutes]

1. Think about how you would solve the problem ***on paper*** without using a computer
2. Break down problems into ***sub-problems***
3. ***Recursion*** is your friend. Try to solve the problem for a base case and build from there.
4. Brainstorm various data structures and think about if their properties can be applied here.
5. Try relating to similar problems you might have encountered before.
6. Focus on any ***unused properties*** from the question. Think about how it impacts your ideas.
7. Use ***time-space tradeoff*** to think of more ideas.
8. Verify the final idea by walking through few test cases.

Code [15 minutes]

1. If given a choice, pick the language you are most comfortable with.
2. ***Modularize*** your code
3. Substitute trivial operations with ***placeholder functions*** which you can implement later
4. Talk through your code. Some people like to talk while coding while some choose to stop after every 2-3 lines of code to communicate what they are doing next.

Test [5 minutes]

1. Always test your code with an example input. This is often known as a ***"dry run"***. The key idea is to go through each line of code and trace the exact execution for the chosen example.
2. It's a good idea to test each module / function of your code independently.

 # Suggested time breakup is based on a 45 minute interview. Keeping few minutes for asking questions to the interviewers.