Coding Strategy

30-minute execution plan

Minute	Action
0–3	Read both problems and choose easier one
3–15	Code easier problem
15–18	Test easy problem
18–28	Code second problem
28–30	Final edge-case testing

How to choose easier problem

Pick the problem that:

- has simple input/output,
- uses array/string/number basics,
- can be solved without advanced data structures,
- has clear examples.

Avoid first if:

- it needs graph/DP,
- input format is confusing,
- constraints require optimization you cannot identify quickly.

Input-output discipline

Read the problem statement carefully for:

- number of test cases,
- array size,
- indexing,
- output format,
- whether result should be printed or returned,
- case sensitivity.

Testing method

Always test:

1. Smallest input
2. Normal input
3. Duplicate values
4. Negative values if allowed
5. Already sorted input
6. Reverse sorted input
7. All same values

If stuck

Use brute force if constraints are small or unclear. A correct $O(n^2)$ solution may pass many placement tests if `n` is small.

If constraints are large:

- sorting often reduces complexity,
- hashing often helps with frequency/search,
- two pointers often helps sorted arrays/strings.

Output formatting

If expected output is:

```
1 2 3
```

Do not print:

```
[1, 2, 3]
```

or extra labels like:

```
Result is 1 2 3
```

Only print what is asked.