facebook

Above & Beyond CS (ABCS)

Coding Interview Workshop Series

Workshop 1

Proactive Communication

Recap: The Goal of a Coding Interview

...is to get signal on things that we do at Facebook every day.

- How you think about and tackle hard problems and how you communicate about code.
  - Evaluate your problem-solving skills to see if you can translate thought into reasonably correct, well-structured code
- How you consider engineering tradeoffs (memory vs. time)
- Limits of what you know

ABCS Tips and Tricks

Before coding

...the first 5'

1. Communicate Proactively
2. Design Your Algorithm
3. Work the Clock

During coding

...the next 10'

4. Writing code at the whiteboard
5. Talk through your code / solution
6. Handling mistakes

"After" coding

...the last 2-3'

7. Test your code
8. Ask questions!

Before you start coding...

1. Communicate proactively
2. Design your algorithm
3. Work the clock

1. Communicate proactively

Proactive Communication

Huh? What exactly does that mean?

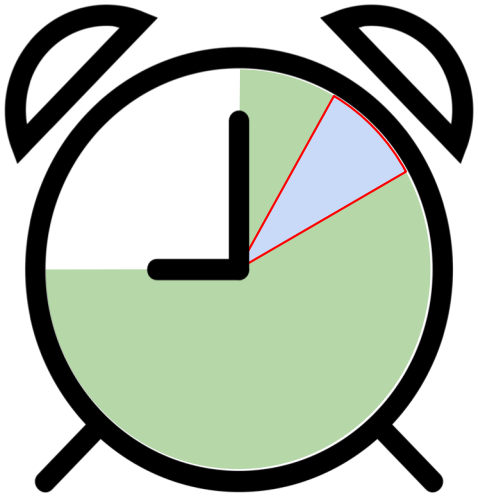
Pro • ac • tive

/pro'aktiv/

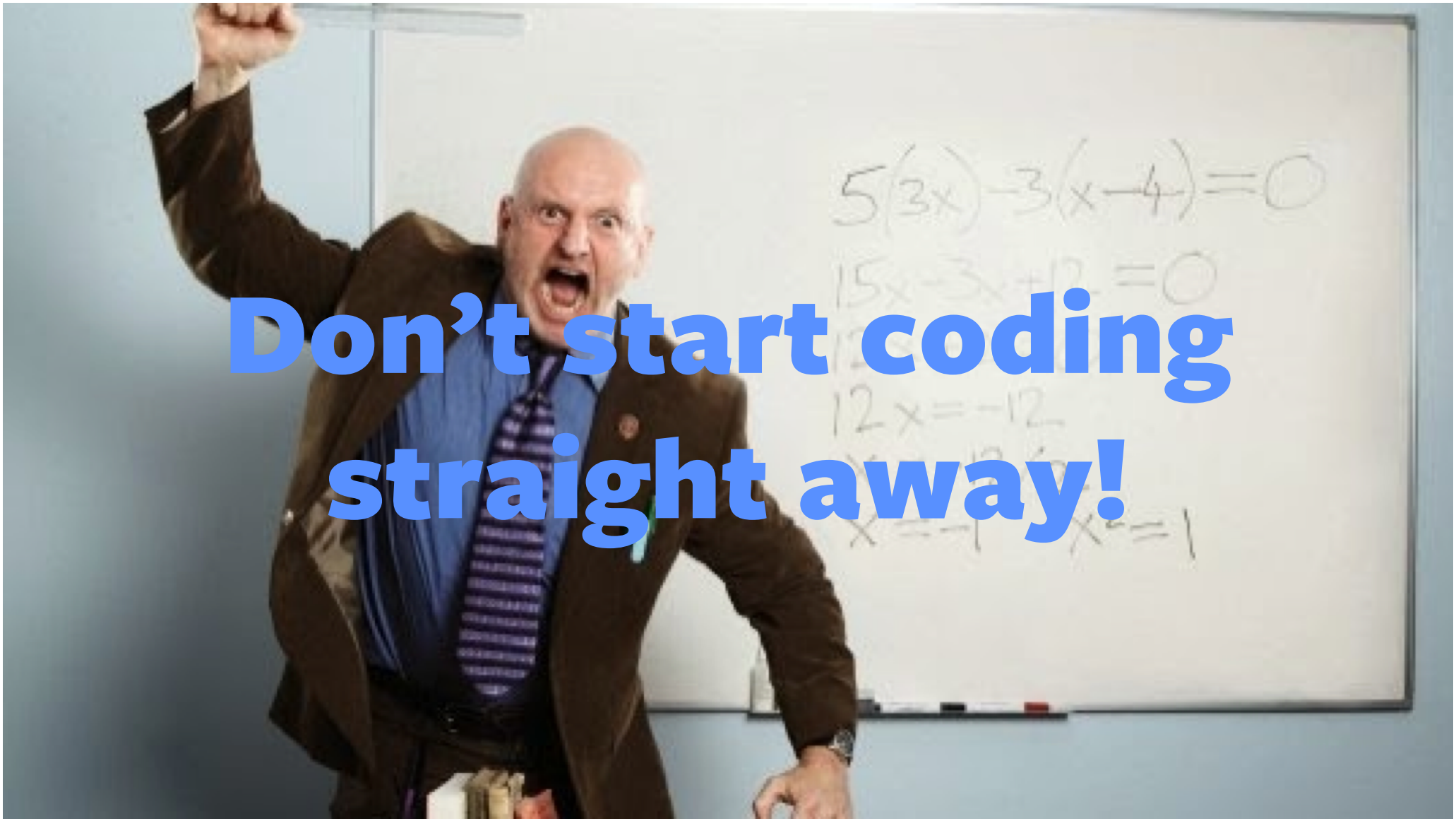
(adj) controlling a situation by causing something to happen rather than responding to it after it has happened

Proactive Communication

Is how you make the first 5 minutes count.



- Proactive communication is any communication that takes place before you start coding
- Intended to help you get what you need, get organized, and set the tone so that you can show what you know
- Don’t let the first 5min impede the next 30min


Don’t start coding
straight away!

Proactive Communication

Don’t start coding straight away!

- Don’t assume the question is easy
  - Repeat the question and rephrase it in your own words; the interviewer will clarify if necessary
  - Assume all information that is given is necessary to solve the problem
- Ask A LOT of questions
  - Make sure you fully understand what you’re being asked to return
  - Resolve any areas of ambiguity; clarify the scope and intention of the problem
  - Validate or state assumptions
  - If there are edge cases (inputs that could break your solution), how should they be handled?

For example:

You are asked to design an algorithm to sort a list. What questions might you ask?

For example:

You are asked to design an algorithm to sort a list. What questions might you ask?

- What sort of list? An array? A linked list?
- What does the array hold? Numbers? Characters? Strings?
- Are the numbers integers?
- Where did the numbers come from? Are they IDs? Values of something?
- How many numbers are there?

Other Common Questions to Ask

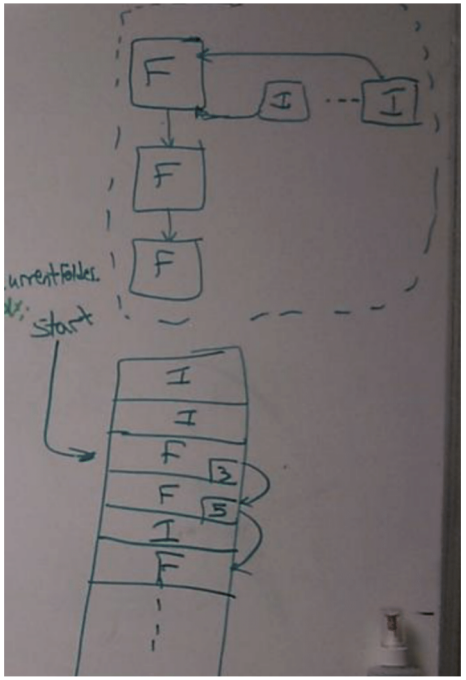
These are just a few to get you started.

- How big is the size of the input?
- How big is the range of values?
- What kind of values are they? Are there negative numbers? Floating points? Will there be empty inputs?
- Are there duplicates within the input?
- What are some extreme cases of the input?
- How is the input stored? If you are given a dictionary of words, is it a list of strings or a trie?

Other Forms of Proactive Communication

Remember, communication isn't always verbal.

- Be visual
  - Use diagrams
  - Illustrate the problem and your high-level solution approach



Thank you!

Workshop 2
[INSERT REGION DATE/TIME]

Complete Pre-Work

- ✓ Review screencast
- ✓ Solve HackerRank problems
- ✓ Be prepared to walkthrough your submitted code