Competitive
Programming and
Mathematics
Society

Technical Interview - workshop

CPMSoc

Table of contents

Welcome

- Join our subcom!
- Mathematics workshops will run every even-numbered week (4, 6, 8, ...)
- Programming ones will run every other week (3, 5, 7, ...)
- Slides will be uploaded on website (unswcpmsoc.com)

Workshop format

- Technical interviews
- Format
- Preparation
- Technique
- Demonstration interview(s)

Interview

- What does it look like?

Interview

- What does it look like?
- Regular interview/icebreaker

Interview

- What does it look like?
- Regular interview/icebreaker
  - Introducing yourself
  - Experience
  - Personal projects/portfolio
  - Common interview questions

Interview

- What does it look like?
- Regular interview/icebreaker
  - Introducing yourself
  - Experience
  - Personal projects/portfolio
  - Common interview questions
- Technical interview

Interview

- What does it look like?
- Regular interview/icebreaker
  - Introducing yourself
  - Experience
  - Personal projects/portfolio
  - Common interview questions
- Technical interview
  - Not scary at all!

Technical interview

- Step 1: You are presented with a problem

Technical interview

- Step 1: You are presented with a problem
- Step 2: Think through the problem

Technical interview

- Step 1: You are presented with a problem
- Step 2: Think through the problem
- Step 3: Code a solution

Technical interview

- Step 1: You are presented with a problem
- Step 2: Think through the problem
- Step 3: Code a solution
- Step 4: ISSUES!

Technical interview

- Step 1: You are presented with a problem
- Step 2: Think through the problem
- Step 3: Code a solution
- Step 4: ISSUES!
- Step 5: ???

Technical interview

- Step 1: You are presented with a problem
- Step 2: Think through the problem
- Step 3: Code a solution
- Step 4: ISSUES!
- Step 5: ???
- Step 6: profit :-)

Preparation

- Preparation is KEY for less stress

Preparation

- Preparation is KEY for less stress
- Coding language

Preparation

- Preparation is KEY for less stress
- Coding language
- Data structures

Preparation

- Preparation is KEY for less stress
- Coding language
- Data structures
- Run times

Toolbox - data structures

- Arrays
- Linked lists
- Graphs
- Hash table
- Cache/memoization

Toolbox - algorithms

- Binary search
- Graph search algorithms
- Greedy algorithms
- Recursion/divide and conquer
- Dynamic programming
- String algorithms
- Sorting and searching

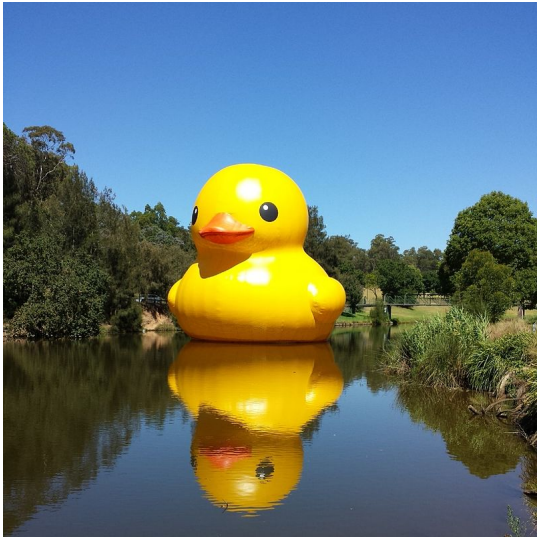
Approaching the problem

- Ask questions about the problem
- Check input and output case
- Using multiple approaches/perspectives
- Drawing diagrams is very useful for thinking about a problem
- Think top down, about the general case/overall problem, instead of specific cases (save that until later)
- If you've seen the question before, let the interviewer know

Working through the problem

- Step 5: ???

Working through the problem



Working through the problem

- Explain each step of your thought process
- Purpose of code
- THINK OUT LOUD!

Coding tips

- Explain the solution before starting to code

Coding tips

- Explain the solution before starting to code
- Use internal/intrinsic documentation if needed, use good style

Coding tips

- Explain the solution before starting to code
- Use internal/intrinsic documentation if needed, use good style
- Solution format is writing a function with inputs, and outputs correct answer

Coding tips

- Explain the solution before starting to code
- Use internal/intrinsic documentation if needed, use good style
- Solution format is writing a function with inputs, and outputs correct answer
- Debugging - use rubber-duck debugging or even output statements for program state

Coding tips

- Explain the solution before starting to code
- Use internal/intrinsic documentation if needed, use good style
- Solution format is writing a function with inputs, and outputs correct answer
- Debugging - use rubber-duck debugging or even output statements for program state
- Reason through the code/errors, and explain why any changes have been made (no guess-and-check)

Coding tips

- Explain the solution before starting to code
- Use internal/intrinsic documentation if needed, use good style
- Solution format is writing a function with inputs, and outputs correct answer
- Debugging - use rubber-duck debugging or even output statements for program state
- Reason through the code/errors, and explain why any changes have been made (no guess-and-check)
- Sometimes, running code will be impossible, which means verbal communicating/running through the code manually works

General tips

- Algorithm and solution will not be overly complicated

General tips

- Algorithm and solution will not be overly complicated
- Time complexity will also not be overly complex (mostly only requires class of time complexity)

General tips

- Algorithm and solution will not be overly complicated
- Time complexity will also not be overly complex (mostly only requires class of time complexity)
- If you need the interviewer's help, just ask

General tips

- Algorithm and solution will not be overly complicated
- Time complexity will also not be overly complex (mostly only requires class of time complexity)
- If you need the interviewer's help, just ask
- First, get A solution, then optimise

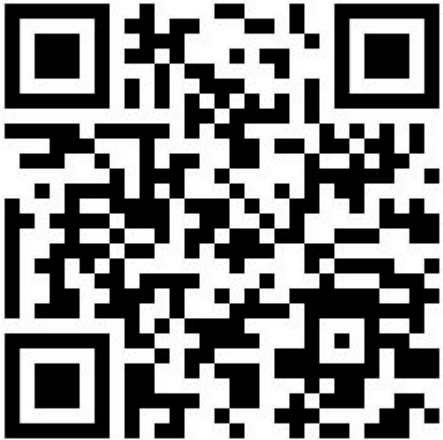
General tips

- Algorithm and solution will not be overly complicated
- Time complexity will also not be overly complex (mostly only requires class of time complexity)
- If you need the interviewer's help, just ask
- First, get A solution, then optimise
- ALWAYS communicate your thought process

Mock interview

Attendance form :D

CPMSOC



Further events

Please join us for:

- Social session tomorrow
- Math workshop next week
- Programming workshop in two weeks