Stanford Computer Forum
Technical Interview Workshop

Christian Yongwhan Lim

12pm PT, April 25, 2024

Christian Yongwhan Lim

Education

Stanford University

Part-time Jobs

Full-time Job

Google Research

TWO SIGMA

Workshops

Stanford ENGINEERING | Stanford Computer Forum

Coach/Judge

icpc International Collegiate Programming Contest

icpc.foundation

advancing the art and sport of competitive programming

https://www.yongwhan.io

Christian Yongwhan Lim

- Christian and Grace Consulting **Owner**;
- Columbia **Adjunct** (Associate in CS);
- Columbia ICPC **Head Coach**;
- ICPC **Internship Manager**;
- ICPC World Finals CLI Symposium **Co-lead**;
- ICPC Curriculum Committee **Co-lead**;
- ICPC North America **Leadership** Team;
- ICPC North America Championship **Operations**;
- ICPC North America Programming Camp **Trainer**;
- ICPC NAQ and Regionals **Judge**;

https://www.yongwhan.io

Overview

- Part I: Data Structures Interview
- Part II: Competitive Programming
- Part III: Behavioral Interview
- Part IV: System Design Interview
- Part V: Machine Learning Interview (ML Engineer/Data Scientist)

Part I: Data Structures Interview

Interview Types

- **Technical Interview**
  - Tests technical skill-sets required for a job.
- Behavioral Interview
  - Tests soft skills (e.g., effective communication, conflict resolution, etc)

Technical Interview

- Recruiter Call
- **0-1 Online Coding Challenge**
  - automated screening with 2-3 questions.
- **2-3 Technical Phone Screens**
  - first technical conversation with human.
- **4-7 Interviews in Onsite**
  - similar to phone screening but more in-depth; you may get probed on your claimed expertise.
- 0-5 Fit Calls & Negotiation

Interview Topics

- **Data Structures and Algorithms**
- System Design

Interview Topics

- **Fundamentals**
  - Arrays and Linked Lists
  - Binary Trees
  - Heaps
  - Sorting

Interview Topics

- **Important**
  - Stacks and Queues
  - Hash Tables
  - Binary Search Trees
  - Searching
  - Recursion

Interview Topics

- **Real Differentiators (Tech vs Quant)**
  - **Strings**: Knuth Morris Pratt (KMP); Rabin Karp / String Hashing; Suffix Array; Suffix Automaton;
  - **Dynamic Programming**: 1D; 2D; Interval; Tree;
  - **Greedy Algorithms** and Invariants: Matroid;
  - **Graphs**: Shortest Path; Lowest Common Ancestor; Flow / Matching; Minimum Spanning Tree;

Interview Topics

- **Real Differentiators (Tech vs Quant)**
  - **Strings**: Knuth Morris Pratt (KMP); Rabin Karp / String Hashing; Suffix Array; Suffix Automaton;
  - **Dynamic Programming**: 1D; 2D; Interval; Tree;
  - **Greedy Algorithms** and Invariants: Matroid;
  - **Graphs**: Shortest Path; Lowest Common Ancestor; Flow / Matching; Minimum Spanning Tree;
    - BFS; DFS; Dijkstra; Bellman-Ford; Floyd-Warshall;
    - Ford-Fulkerson/Edmond-Karp; Dinic;
    - Prim; Kruskal (DSU);

Sample Problem

- **A minimum number of insertions to make a string a palindrome.**

Sample Problem

- **A minimum number of insertions to make a string a palindrome.**

- **Constraint**
  - string length is at most 5000
  - each character is from ‘a’ to ‘z’

Sample Problem

- **A minimum number of insertions to make a string a palindrome.**

- **Constraint**
  - string length is at most 5000
  - each character is from ‘a’ to ‘z’

Any Idea?

Sample Solution

```
int minInsertions(string &s) {
    int n = s.size();
    vector<vector<int>> dp(n, vector<int>(n,0));
    for (int i = 1; i < n; i++)
        for (int j = 0, k = i; k < n; j++, k++)
            dp[j][k] = (s[j]==s[k]) ?
                            dp[j+1][k-1] :
                            min(dp[j][k-1],dp[j+1][k])+1;
    return dp[0][n-1];
}
```

Interview Preparation Resources (Tech)

- **LeetCode**: Solve all four weekly/biweekly problems in **60 minutes**!
  - 3 + 6 + 12 + 24 (+15 buffer)

- **CodeForces**: Get to **1800+** rating
  - Clear 4 questions out of 6!

Interview Preparation Resources (Quant)

- **LeetCode**: Solve all four weekly/biweekly problems in **20 minutes**!
  - 1 + 2 + 4 + 8 (+5 buffer)

- **CodeForces**: Get to **2200+** rating
  - Clear 5 questions out of 6 **fast**!

Interview Preparation Resources

- **Tech**: *Elements of Programming Interview*

- **Quant**: *Competitive Programming 4*

Part II: Competitive Programming

Programming Zealots @Discord

- Break into **CodeForces** rating of **2200+** as fast as you can!

- Join the discord server!

  bit.ly/programming-zealot

Programming Zealots @CodeForces

- Also, join CodeForces group!

bit.ly/cf-zealots

Success Pathways

- **Programming Zealots @ CodeForces**
- **800 - 2100 (A - N)**
  - **For those who are just starting**
  - To gain some experiences with an explicit goal to enjoy the process of solving new problems;
  - To make it to the ICPC North America Championship (NAC)!

Success Pathways

- **Programming Zealots @ CodeForces**
- **800 - 2100 (A - N)**
  - **For those who are just starting**
  - To gain some experiences with an explicit goal to enjoy the process of solving new problems;
  - To make it to the ICPC North America Championship (NAC)!
- **2200 - 3500 (O - ZB)**
  - **For those who are more serious**
  - To make it to the ICPC World Finals (and potentially winning a medal)!

Practice Strategy

- If your goal is to get to a rating of **X**, you should practice on problems that are **X + 300** typically, with a spread of 100. So, picking problems within the range of:

  {X + 200, X + 300, X + 400}

  would be sensible!

Practice Strategy

- If your goal is to get to a rating of **X**, you should practice on problems that are **X + 300** typically, with a spread of 100. So, picking problems within the range of:

  {X + 200, X + 300, X + 400}

  would be sensible!

- So, if you want to target becoming a **red (grandmaster)**, which has a lower-bound of 2400, you should aim to solving {2600, 2700, 2800}.
- **(Eventual) Target**: You should focus on solving it for 30 minutes or less!

Practice Strategy (con't)

- You should focus on solving each problem for **30 minutes or less**; if you cannot, you should consider solving a problem with a lower rating.
- You should aim to solve **~5 problems** each day within this range to expect a rank up within six months.

Practice Strategy (con't)

- You should focus on solving each problem for **30 minutes or less**; if you cannot, you should consider solving a problem with a lower rating.
- You should aim to solve **~5 problems** each day within this range to expect a rank up within six months.
- If you cannot solve a problem, here is a sample recipe you can follow:
  - Look at editorial for **hints**, and try to solve the problem.
  - Look at editorial for **full solutions**, and try to solve the problem.
  - Look at **accepted code**, and try to solve the problem.
  - Make sure you **revisit after two weeks** and see if you can solve it.

Popular Training Resources

- **U ICPC**: https://u.icpc.global/training/

- **CP Algorithms**: https://cp-algorithms.com/
- **USACO Guide**: https://usaco.guide/

- **Kattis**: https://open.kattis.com/
- **CSES**: https://cses.fi/problemset/
- **solved.ac**: https://solved.ac/

Stanford ICPC

- Join the Stanford ICPC discord server!

bit.ly/stanford-icpc

CP Trainer's Guide to ICPC World Finals @Stanford ICPC

- There is a special workshop tomorrow!
  - Date: Friday, April 26 (**Tomorrow**!)
  - Time: **6pm PT**
  - Location: Gates Computer Science, **Room 403** (Fujitsu)
  - Great for learning about **how to advance** to the ICPC World Finals.

- More details can be found in Stanford ICPC discord server!
  - In particular, you may submit a dinner preference there by tonight 😎

International Collegiate Programming Contest (ICPC)

- If you would like to get involved in helping out as a volunteer or an official (unpaid) intern, please reach out via **christian.lim@icpc.global** or **internship@icpc.foundation**.

International Collegiate Programming Contest (ICPC)

- If you would like to get involved in helping out as a volunteer or an official (unpaid) intern, please reach out via **christian.lim@icpc.global** or **internship@icpc.foundation**.

- **Especially great** for getting:
  - Immigration help (e.g., F-1 Post-Completion OPT)
  - Practical experiences
  - Interview and/or programming contest training
- Some Stanford students were previous interns (now at quant)! 🙂

Part III: Behavioral

Behavioral Interview

- Becoming an industry standard to have at least one session in typical software engineering interview loop.
- Wants to assess leadership potential.
- Tests soft skills (e.g., effective communication, conflict resolution, etc.)
- Open-ended: **not** about getting it right or wrong!

Sample Question

- Tell me about a time when you led a team to successfully complete a project.

Sample Answer

- Best if you led a hackathon/passion project.
- Otherwise, if you led a project as an intern, highlight it.

- Be **concise**!
- Include hard **metrics** in terms of %, $, etc.
- Provide **concrete** examples.

Resources

- *Behavioral Interview Questions and Answers* by Horatio Bird;

- *Leadership Interview Questions You'll Likely Be Asked* by Vibrant Publishers;

Part IV: System Design

System Design Interview

- Identify large components of the system (e.g., newsfeed) and describe how each component is connected.
  - Backend; Frontend; Database; ...

- Actual implementation details are **not** as important.

- Tests whether you can design an architecture using standard design patterns.

System Design Interview: EIGHT Steps

1. Clarify requirements
2. Rough estimate
3. Define system interface
4. Define data model
5. Design in high-level
6. Design in detail
7. Bottlenecks
8. Trade-offs

Resources

- *The System Design Interview*, 2nd edition by Lewis C. Lin, et. al.

- *System Design Interview* by Alex Xu

Part V: Machine Learning

Machine Learning Interview

- **Hands-on Experience** using TensorFlow/Keras/PyTorch: comfortable using data to feed into a baseline model.
- **ML Foundations** (e.g., linear regression, support vector machine, etc.)
- **Recent Trends** (reinforcement learning, deep learning architectures, etc.)

Machine Learning Interview

- **Hands-on Experience** using TensorFlow/Keras/PyTorch: comfortable using data to feed into a baseline model.
- **ML Foundations** (e.g., linear regression, support vector machine, etc.)
- **Recent Trends** (reinforcement learning, deep learning architectures, etc.)

- **In-depth knowledge** of a specialization (e.g., computer vision) can be a plus, but not required.

Sample Questions

- **Theory**: What is a difference between unsupervised learning and supervised learning?

- **Hand-on**: What are some practical ways to avoid overfitting?

- **Implementation**: Given a stock market data, predict the future stock price.

Resources

- **Textbooks**: *Deep Learning* by Ian Goodfellow, et. al.
- **Courses**: Stanford CS 229 (Machine Learning); ...
- **Tools**: PyTorch; Keras; TensorFlow; Jupyter; ...

Terse Guides

- Please take a look as needed:

  bit.ly/christian-terse-guide

Any Questions?

Contact Information

- **Email**: yongwhan@yongwhan.io

- **Personal Website**: https://www.yongwhan.io/

- **LinkedIn Profile**: https://www.linkedin.com/in/yongwhan/
  - Feel free to send me a connection request!
  - Always happy to make connections with promising students!