Amazon ও Google এ চাকরির প্রস্তুতি পর্ব

ইন্ডাস্ট্রিতে চাকরি করার কোন পূর্ব পরিকল্পনা ছিলনা। আমাকে মোটামুটি শূন্য থেকে শুরু করতে হয়েছে। আমার ৬ মাস সময় ছিল। প্রথমে ইন্টারভিউর সব ধাপ সম্পর্কে বলি।

০। Resume: ভালো Resume তৈরী করা চাকরি খোঁজার প্রথম কাজ। এখন Resume কয় পাতা হবে, আমার Recommendation থাকবে ১ পাতা। আমি প্রথমে ২ পাতার Resume তৈরী করেছিলাম, পরে ১ পাতায় নিয়ে আসি। Google, Amazon এ এই ১ পাতার Resume দিয়েই ইন্টারভিউ কল পেয়েছি। আমি এই সাইট থেকে https://app.zety.com/user/cv template ব্যবহার করেছি। Resume তে বিভিন্ন Programming Skills রেটিং আকারে না দিয়ে Expert, Familiar, Advanced এইভাবে দিলে ভালো। No Typo, No Buzz Word। Resume তে বিভিন্ন প্রজেক্ট এর সাথে GitHub লিংকও দেয়া উচিত। বিশেষত যদি, Mid-Range Software Company তে চাকরির জন্য আবেদন করা হয়। Resume তে ছবি, জন্ম তারিখ, রেফারেন্স অবশ্যই দেয়া যাবে না। রেফারেন্স On Request এ দিতে হয়। Recruiter রা সাধারনত গড়ে ৬ সেকেন্ড সময় নিয়ে Resume দেখে [https://medium.com/free-code-camp/writing-a-killer-software-engineering-resume-b11c91ef699d], কাজেই এই ৬ সেকেন্ডে যাতে নিজের সম্পর্কে আগ্রহী করে তোলা যায়।

১। ইন্টারভিউ কলঃ দ্বিতীয় কাজ হচ্ছে ইন্টারভিউর কল আসার ব্যবস্থা করা। এক্ষেত্রে LinkedIn থেকে সরাসরি চাকরিতে ইন্টারভিউর জন্য আবেদন করা যায়। LinkedIn এ Premium Account ও একটা ভালো উপায়, বিভিন্ন কোম্পানির Hiring Manager / Recruiter কে সরাসরি মেসেজ দেয়া যায়। সবচেয়ে ভালো উপায় হল referral ব্যবহার করা। কোম্পানিতে কাজ করে এমন কাউকে দিয়ে চাকরির জন্য আবেদন করা। Referral এর জন্য খুব ভালো ভাবে পরিচিত লাগবে, এইটা দরকারি না, Undergrad/Grad School এ পরিচিত বা সিনিয়র/জুনিয়র কিছু একটা হলেই হবে। আমি LinkedIn এ খুঁজে খুঁজে Stony Brook University বা Bangladesh বা SUST এর কেউ হলে নিচের মত মেসেজ দিতাম। অনেকেই হয়ত ব্যস্ততার কারণে উত্তর নাও দিতে পারে, অথবা Referral এর সুযোগ নাও থাকতে পারে, এটাও মাথায় রাখতে হবে।

Subject: Fellow ‘B’ Grad Looking for Software Engineer job at ‘X’

Dear ‘Y’,

I am ‘Z’, currently pursuing ‘A’ at ‘B’. I have seen you have completed your ‘A’ from ’B’. I will be graduating this month, and now actively seeking for job. There are several jobs opening at ‘X’. I have attached my resume, if you think my profile is fit for them, please let me know.

Thanks in advance.

২। ইন্টারভিউ Recruiter: প্রথমে সাধারণত Recruiter রা ফোন দেয়। ওরা মুলত বর্তমান পড়ালেখা বা চাকরির অবস্থা, কোন বিষয়ে কাজ করতে আগ্রহী, কবে ইন্টারভিউ দিতে পারবে এই সব কথা বলে। এখানে Recruiter রাই ইন্টারভিউ থেকে শুরু করে Job Offer, Salary Negotiation সব কিছু করে থাকে। ইন্টারভিউ কবে দিব, ফলাফল কি, Onsite ইন্টারভিউ হলে আসা-যাওয়া, থাকা-খাওয়া মোটামুটি সবই Recruiter রাই ব্যবস্থা করে দেয়। ইন্টারভিউতে প্রশ্ন কেমন হতে পারে, কি কি বিষয়ে পড়তে হবে, গুরুতপূর্ন অনেক Resource ও দিয়ে থাকে। কোন বিষয়ে প্রশ্ন থাকলে Recruiter কে

বলে জেনে নেয়া ভালো। যেমন আমাকে একটা ইন্টারভিউতে প্রথমে Machine Learning Engineer পোস্টের জন্য Recruiter আবেদন করতে বলেছিল, কিন্তু আমি কথা বলে পরিবর্তন করে Software Engineer পোস্টের জন্য ব্যবস্থা করতে বলি।

৩। Online Round: অনেক কোম্পানি প্রথমে Online Round নেয়। এইটা Recruiter মেইলে লিংক দিয়ে ৭/১০ দিন সময়ের ভিতরে করতে বলে। সাধারণত ১/২ ঘন্টা সময়ে ২/৫ টা প্রবলেম সমাধান করতে হয়। অনেক সময় সাথে কিছু MCQ প্রশ্নও থাকে।

৪। Phone Screen: সাধারণত ২/৩ টা Phone Screen ইন্টারভিউ হয়। Phone Screen ইন্টারভিউতে Google doc বা https://codeshare.io/ তে Code করতে হয়। Interviewer Skype, Hangout বা Phone এ কথা বলে। প্রথমে প্রবলেম কিভাবে সমাধান করব, উদাহরণ দিয়ে বুঝাতে হয়, Time and Space Complexity নিয়ে আলোচনা করতে হয়, যদি Interviewer সবুজ সংকেত দেয়, তবেই Code লেখা শুরু করা উচিত। Phone screen ৪৫ মিনিট থেকে ১ ঘন্টা হয়। প্রথম ৫-১০ মিনিট নিজের সম্পর্কে বলতে হয়, Interviewer ও নিজের কথা বলে। এর পরে ৩০-৪৫ মিনিট ২ টা প্রবলেম সমাধান করতে হয়। কখনও দুইটা ভিন্ন প্রবলেম থাকে, কখনও প্রথম প্রবলেমের Time and Space Complexity Optimize করতে হয়।

৫। Onsite: আগের ধাপ Successfully শেষ হলে Onsite এ ইন্টারভিউ। Onsite এ ৪/৫ রাউন্ড থাকে। প্রত্যেকটা ৪৫ মিনিট থেকে ১ ঘন্টা, কোন Break ছাড়া, তবে Lunch Break থাকতে পারে। Onsite ইন্টারভিউগুলো Phone Screen এর মত। তবে Code করতে হয় Whiteboard এ। যদিও ইদানীং Google এ Chromebook দেওয়া হয়। আমার কাছে Whiteboard ভালো, নিজের চিন্তা ভাল ভাবে ব্যাখ্যা করে যায়, Code এ কোন পরিবর্তন থাকে সহজে করা যায়।

এই মোটামুটি ইন্টারভিউ প্রক্রিয়া।

Phone Screen ও Onsite ইন্টারভিউগুলোতে তিন ধরনের প্রশ্ন থাকে, ক) Data Structure and Algorithm, খ) System Design আর গ) Behavioral।

ক) **Data Structure and Algorithm:** ইন্টারভিউর জন্য সবচেয়ে গুরুত্বপূর্ণ হল Data Structure and Algorithm Problem. এই ধরনের প্রবলেম Phone Screen এ আসে এবং Onsite এ সিনিয়র পজিশন না হলে ১টা System Design আর বাকি সব Data Structure and Algorithm। ইন্টারভিউতে খুব ছোট প্রশ্ন থাকে, এবং Code ১০/১৫ লাইন হয়, এটা অবশ্য Python এর জন্য, আর Code অবশ্যই Executable হতে হবে। NO PSCUDOCODE. একটা প্রশ্নঃ n সংখ্যক পজিটিভ sorted সংখ্যা আছে, এর মধ্যে সবচেয়ে ছোট যেই পজিটিভ সংখ্যা এই লিস্টে নাই, সেটা বের করতে হবে। Input: [1, 2 ,3 4, 6], Output: 5. একটা উপায় হল, ১ থেকে চেক করা, কোন সংখ্যা নাই, Time Complexity O(n^2). আবার Sum of N integer এর সুত্র দিয়েও করা যায়, Time Complexity O(n). কিন্তু Optimal Time Complexity O(log n). Binary Search করা। এখন প্রথমেই Binary Search এর কথা না বলে, অন্য উপায় বলে, পরে Binary Search বলা ভালো। আর Code অবশ্যই Interviewer বলার পরে লিখতে হবে। এখন যদি Binary Search এর আইডিয়া না আসে? খুবই স্বাভাবিক, ইন্টারভিউর সময় মাথা ঠিক রাখা খুবই কঠিন। সেক্ষেত্রে Interviewer হয়ত Hints দিবে, যেমন বলতে পারে, Data Sorted, sorted থাকলে কি করা যায়? Interviewer এর hints ধরতে পারাও Credit. এমন না যে Hints নেয়া মানে RED FLAG. তবে hints ধরতে না পারলে RED FLAG. ও আচ্ছা, অনেক প্রশ্ন

করতে হবে প্রথমে Interviewer কে। যেমন, Interviewer প্রথমে শুধু n সংখ্যক সংখ্যা দিয়ে শুরু করতে পারে। কি ধরনের সংখ্যা, Duplicate number, sorted number, empty list এইগুলা প্রশ্ন করে Clear করতে হবে। প্রশ্ন না করাও RED FLAG. Code করতে হবে Production Ready, মানে যত ছোট ভাগে Function দিয়ে করা যায়, Code reusable, modular হওয়া উচিৎ। এত কিছু ইন্টারভিউর সময় কি মাথায় থাকে? বেশী বেশী ইন্টারভিউ প্র্যাকটিস করা লাগবে, Pramp https://www.pramp.com/dashboard#/, সাইটে দিতে হবে, পরিচিত যারা ইন্ডাস্ট্রিতে কাজ করেন, তাদের কাছে দিতে হবে।

খ) **System Design:** এই বিষয়ে আমার কোন **Prior Knowledge** ছিলনা। ইন্টারভিউর প্রস্তুতি থেকেই শেখা। System Design এ Interviewer খুবই Vague প্রশ্ন করে। Interviewer পালটা প্রশ্ন করে requirements clear করতে হয়। যেমনঃ প্রশ্ন করতে পারে WhatsApp এর মত messenger ডিজাইন করতে। এখন প্রশ্ন করে নিশ্চিত হওয়া লাগবে কি কি থাকবে ডিজাইনে। এমন না যে হুবহু WhatsApp এর মত, কিছু assumption থাকবে, যেগুলো Interviewer কে বলে clear করতে হবে। যেমন, প্রথমে শুধু টেক্সট মেসেজ ডিজাইন দিয়ে শুরু করা যেতে পারে। পরে ছবি ও ভিডিও যুক্ত করা হবে। গ্রুপ মেসেজ আপাতত বাদ। ধরে নিলাম, WhatsApp মোট ৫০০ মিলিয়ন user, Daily Active User ১০০ মিলিয়ন, সবাই রোজ ৫০ টা মেসেজ দেয়, একটা মেসেজ এ ১০০ টা অক্ষর থাকে। এখন হিসাব করতে হবে, এই ডাটা ১০ বছরের জন্য কত Storage লাগবে, Bandwidth আর message traffic কত হবে। পরে, এই মেসেজ কি ধরনের NoSQL ডাটাবেজে রাখতে হবে, Hadoop, Casandra, MongoDB, etc pros and cons বলতে হবে। Data Sharding কিভাবে হবে, Caching কিভাবে হবে, Load Balancer কোথায় কোথায় লাগবে, Fault Tolerant কিভাবে নিশ্চিত করা যায়, এগুলো আলোচনা করতে হবে।

গ) **Behavioral বা Cultural Fit:** শুধু Coding Skill থাকলেই হবে না, প্রার্থীর Cultural Fit কি না Amazon এ এটা খুবই গুরুত্বপূর্ণ। Amazon এ আমাকে প্রতি রাউন্ডে প্রথম ১৫/২০ মিনিট Behavioral প্রশ্ন করেছে। Amazon এর Leadership Principle বলে ১৪ টা Principal আছে। ওদের উদ্দেশ্য হচ্ছে যে সব প্রার্থীর Behavioral প্রশ্নের উত্তরে এইসব Leadership Principle এর ব্যবহার আছে কি না দেখা। তাই Behavioral এর উত্তরে আগে থেকে এইসব Leadership Principle দেখে উত্তরে এইগুলা যুক্ত করে উত্তর দিতে হবে। Google এও একটা Round ছিল Googlyness and Leadership.

Onsite আর Phone Screen ইন্টারভিউর শেষে Interviewer কেও প্রশ্ন করার সুযোগ থাকে এবং অবশ্যই Interviewer কে প্রশ্ন করতে হবে। যেমন কিছু কমন প্রশ্ন হতে পারে, Interviewer কি ধরনের কাজ করে, আমি জয়েন করলে কি কাজ করতে হতে পারে, প্রতিদিনের রুটিন কাজ কি কি করতে হয় বা Interviewer এর পরিচিতি জানার সময় কিছু Improvised প্রশ্ন করা যেতে পারে। আমার Amazon এর একজন Interviewer Security and Privacy নিয়ে কাজ করত, আমার প্রশ্ন ছিল, Amazon এ Privacy সংক্রান্ত ইস্যু অনেক কম শোনা যাওয়ার কারন কি?

প্রস্তুতি Resources:

১। Cracking the Coding Interview http://www.crackingthecodinginterview.com/ দিয়ে আমার প্রস্তুতির শুরু। এই বইয়ে ইন্টারভিউর সব কিছুরই একটা ধারনা পাওয়া যায়। যদিও এই বইয়ের প্রশ্ন খুব কমন, তাই এগুলো ইন্টারভিউতে আসেনা, তবে ইন্টারভিউতে আসলে কেউ যদি না পারে, তাহলে ধরে নেয়া, Homework করা হয় নাই। MUST READ BOOK.

২। Elements of Programming Interview, অনেকেই বলছেন এই বই Google / Facebook এর জন্য ভালো।

৩। LeetCode Premium Account https://leetcode.com/problemset/all/ : Premium Account এর সুবিধা হল এইটায় কোম্পানি tag থাকে, বড় কোম্পানির আলাদা section আছে, Mock ইন্টারভিউ দেয়া যায়। অন্তত পক্ষে ২ মাসের জন্য হলেও LeetCode Premium Account নিয়ে Code করা উচিৎ। USA তে চাকরি পেতে হলে LeetCode MUST.

৪। Geeksforgeeks https://www.geeksforgeeks.org/: এই সাইটে মুলত আমি বিভিন্ন প্রশ্নের ভিন্ন ভিন্ন উত্তর জানার জন্য দেখেছি।

৫। Back to Back SWE https://www.youtube.com/channel/UCmJz2DV1a3yfgrR7GqRtUUA: এই চ্যানেল LeetCode এর বিভিন্ন সমাধান এর ব্যাখ্যা দেয়া আছে। আমার কাছে খুব ভালো লেগেছে।

৬। Algorithm Every Day https://www.youtube.com/channel/UCx-kFfzekMbhODaBss-ZnsA: এই চ্যানেল LeetCode এর বিভিন্ন সমাধান এর ব্যাখ্যা দেয়া আছে।

৭। Grokking The System Design Interview https://www.educative.io/collection/5668639101419520/5649050225344512: System Design এর জন্য এইটা MUST READ। যদিও paid course, কিন্তু System Design এর জন্য এই কোর্সের কোন বিকল্প নেই।

৮। System Design https://www.youtube.com/channel/UCRPMAqdtSgd0Ipeef7iFsKw: এই চ্যানেল System Design এর জন্য MUST. System Design প্রশ্ন এর জন্য resource খুবই কম।

৯। Dan Crator https://www.youtube.com/channel/UCw0uQHve23oMWgQcTTpgQsQ: এই চ্যানেল যদিও Amazon specific Behavioral Question, তবে যে কোন Company তে প্রযোজ্য হবে।

১০। https://www.pramp.com/dashboard#/, ইন্টারভিউ প্র্যাকটিস এর জন্য MUST. আমি প্রায় ৩০টার মত ইন্টারভিউ দিয়েছি এইখানে। এই সাইটে অন্য যারা ইন্টারভিউ দিবে তারা আমার ইন্টারভিউ নেয়, পরে আমি ওদের। প্রশ্ন সাইট থেকে দেয়া হয়, সাথে সমাধান এবং hints ও দেয়া থাকে।

১১। https://github.com/donnemartin/system-design-primer, এই লিংকেও System Design এর ভালো Content আছে।

১২। https://medium.com/@scarletinked/are-you-the-leader-were-looking-for-interviewing-at-amazon-8301d787815d, Amazon LP প্রশ্নের জন্য খুবই ভালো।

১ম মাসঃ LeetCode Easy Problem

প্রথমে আমি Cracking The Coding Interview দিয়ে পড়া শুরু করি। এই বইয়ের Data Structures and Algorithm সম্পর্কিত সব চ্যাপ্টার প্রথমে শেষ করি। এতে বেসিক আইডিয়াগুলো Revised হয়। প্রথমবার এই বইয়ের প্রবলেম সমাধান করার সময় Naïve Approach এ করেছি। যেই প্রবলেম সমাধান করতে পারি নাই, ঐ গুলো আপাতত বাদ দিয়ে

যাই। এরপর LeetCode করা শুরু। Research এর জন্য Code করা হলেও আমি অনেক দিন থেকে Data Structures and Algorithm সম্পর্কিত Code করা বন্ধ ছিল। তাই প্রথম মাসে টার্গেট ছিল LeetCode এ শুধু Easy প্রবলেম, প্রতিদিন ১০ টা করে, ৩০০ টা করব। LeetCode এ প্রবলেমগুলা খুব ছোট থাকে।

Easy:

https://leetcode.com/problems/find-all-numbers-disappeared-in-an-array/

https://leetcode.com/problems/single-number/, https://leetcode.com/problems/single-number-ii/, https://leetcode.com/problems/single-number-iii/

https://leetcode.com/problems/two-sum/

https://leetcode.com/problems/longest-common-prefix/

https://leetcode.com/problems/majority-element/

আমি একটা প্রবলেম এ ১৫/২০ মিনিটের বেশী সময় দিতাম না। এর মধ্যে না হলে সমাধান দেখে ফেলতাম। এছাড়া Stack, Queue, HashMap, Linked List, Binary Tree, Set, Disjoint Set, Tries, বিভিন্ন ধরনের Sorting, Binary Search, Heap এইগুলার Scratch থেকে Implementation + Python এর Library দিয়ে ব্যবহার করলে এইগুলার সব ধরনের Time Complexity যেমন Insert, Delete, Search https://wiki.python.org/moin/TimeComplexity এইগুলা এই মাসে দেখেছি। Interviewer রা Code Production Ready দেখতে পছন্দ করেন, আমার যেহেতু Industry Experience নাই, তাই এই লিংক থেকে https://pep8.org/ Python এর Coding Standard Review দিয়েছি।

Time and Space Complexity:

যে কোন প্রবলেমের Time and Space Complexity Analysis অবশ্যই করতে হবে। এইটা ভালো করে শিখতে হবে, বিশেষ করে Recursive প্রবলেমের Time and Space Complexity Analysis। খুব Accurate না হলেও কাছাকাছি যাতে বের করা যায়। LeetCode এ মোটামুটি সব প্রবলেমের Time and Space Complexity Analysis Discussion এ থাকে।

২য় মাসঃ Data Structure: Linked List, Binary Tree, HashMap, Heap

Algorithm থেকে Data Structure বেশী গুরুত্বপূর্ণ। Algorithm এ সর্বোচ্চ DFS / BFS আসে। Dijkstra, Bellman Ford বা Prims Algorithm, এইগুলা কপাল খুব খুব খারাপ না হলে আসার কথা না। এই মাসে LeetCode এ Linked List, Binary Tree, HashMap, Heap এর প্রায় সব প্রবলেম Hard ছাড়া সমাধান করেছি। LeetCode ছাড়াও Geeksforgeeks থেকেও আরও LinkedList, Binary Tree এর প্রবলেম সমাধান করেছি। LinkedList, Binary Tree দিয়ে মোটামুটি ২০০ বেশী এর প্রবলেম সমাধান করেছি। মনে হতে পারে, Binary Tree দিয়ে কি এত প্রবলেম আছে? Binary Tree Left View, Right View, Top View, Bottom View, Boundary View, Depth Order, Level Order, Vertical Order, Zigzag Order, Inorder, Preorder, Postorder শুধু Tree Traversal দিয়েই ১১ টা প্রশ্ন।

Binary Tree:

https://leetcode.com/problems/validate-binary-search-tree/

https://leetcode.com/problems/vertical-order-traversal-of-a-binary-tree/

https://leetcode.com/problems/boundary-of-binary-tree/

https://leetcode.com/problems/verify-preorder-sequence-in-binary-search-tree/

https://leetcode.com/problems/binary-tree-maximum-path-sum/

Linked List:

https://leetcode.com/problems/merge-two-sorted-lists/

https://leetcode.com/problems/linked-list-cycle/

https://leetcode.com/problems/swap-nodes-in-pairs/

https://leetcode.com/problems/reverse-nodes-in-k-group/

https://leetcode.com/problems/flatten-a-multilevel-doubly-linked-list/

Heap:

Google এ খুব কমন প্রবলেম মনে হইছে আমার কাছে। কোন প্রবলেম এ k-most টার্ম থাকলেই ধরে নিতে হবে এইটা সম্ভবত Heap এর প্রবলেম। এছাড়াও যদি কোন প্রবলেম O(n) complexity তে সমাধান করার পর Interviewer আরো Optimization করতে বলে বা logarithm Complexity তে সমাধান চায়, তাহলেও more likely এটা Heap দিয়ে সমাধান করা যাবে।

https://leetcode.com/problems/merge-k-sorted-lists/

https://leetcode.com/problems/find-median-from-data-stream/

https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/

https://leetcode.com/problems/k-closest-points-to-origin/

https://leetcode.com/problems/path-with-maximum-minimum-value/

HashMap:

Python এ set, dict, Counter, defaultdict, OrderedDict এইগুলোর Basic ভালো করা জানতে হবে, কোনটার সাথে কি পার্থক্য, Search(), Insert(), Delete() এর Time and Space Complexity।

https://leetcode.com/problems/design-hashmap/

https://leetcode.com/problems/design-hashset/

https://leetcode.com/problems/max-points-on-a-line/

https://leetcode.com/problems/number-of-distinct-islands/

https://leetcode.com/problems/longest-substring-without-repeating-characters/

৩য় মাসঃ More Data Structure: Stack, Two Pointer, Sliding Window, Binary Search, Tries, DFS, BFS

এই মাসেও Data Structure প্রবলেম সমাধান করেছি। Stack দিয়ে অনেক Hard প্রবলেম সমাধান করা যায়। যেমনঃ

https://leetcode.com/problems/largest-rectangle-in-histogram/

https://leetcode.com/problems/remove-duplicate-letters/

https://leetcode.com/problems/next-greater-element-i/

https://leetcode.com/problems/next-greater-element-ii/

https://leetcode.com/problems/daily-temperatures/

Next largest, smallest বা parenthesis সম্পর্কিত কোন প্রবলেম আসলে প্রথমেই Stack মাথায় আসত। যদিও প্রথম প্রথম এই প্যাটার্ন প্রবলেম ধরতে পারাটা কঠিন। এর জন্য প্রবলেম সমাধান করে অন্যদের সমাধান দেখাও দরকার, যাতে আইডিয়া পাওয়া যায়।

Two Pointer and Sliding Window:

এই দুইটা আগে জানা ছিল না, এই দুইটাও খুব কাজের জিনিষ। কোন প্রবলেমে sub-string নিয়ে কাজ করা লাগলে সাধারনত Two Pointer বা Sliding Window মাথায় আসত।

https://leetcode.com/problems/longest-substring-without-repeating-characters/

https://leetcode.com/problems/3sum/

https://leetcode.com/problems/trapping-rain-water/

https://leetcode.com/problems/longest-substring-with-at-most-two-distinct-characters/

https://leetcode.com/problems/sliding-window-maximum/

Binary Search:

Binary Search দিয়েও অনেক Hard প্রবলেম সমাধান করা যায়। এখানেও প্রথম প্রথম প্যাটার্নটা ধরতে পারা কঠিন।

https://leetcode.com/problems/split-array-largest-sum/

https://leetcode.com/problems/median-of-two-sorted-arrays/

https://leetcode.com/problems/russian-doll-envelopes/

https://leetcode.com/problems/count-of-smaller-numbers-after-self/

https://leetcode.com/problems/search-in-rotated-sorted-array/

Tries:

Tries মুলত Time Complexity Optimize কাজে লাগে। DFS / BFS অনেক প্রবলেমের Naïve Solution হয়ত আছে, কিন্তু ইন্টারভিউতে Tries দিয়ে সমাধান করতে হয়।

https://leetcode.com/problems/implement-magic-dictionary/

https://leetcode.com/problems/word-search-ii/

https://leetcode.com/problems/word-squares/

https://leetcode.com/problems/longest-word-in-dictionary/

https://leetcode.com/problems/concatenated-words/

DFS:

DFS আমি বেশী গুরুত্ব দিয়ে করেছি। Dynamic Programming এর Tabulation বা Bottom Up আমি ভালো পারি না, তাই DFS + Memoization বা Top Down নিয়ে বেশী কাজ করেছি।

https://leetcode.com/problems/android-unlock-patterns/

https://leetcode.com/problems/cracking-the-safe/

https://leetcode.com/problems/swim-in-rising-water/

https://leetcode.com/problems/robot-room-cleaner/

https://leetcode.com/problems/distribute-coins-in-binary-tree/

BFS:

BFS দিয়ে করলে সাধারনত Time Complexity বের করা DFS থেকে সহজ। তাই, যেই প্রবলেম DFS ও BFS দুইটা দিয়েই করা যায়, তখন আমি BFS দিয়ে করতাম।

https://leetcode.com/problems/escape-a-large-maze/

https://leetcode.com/problems/shortest-path-to-get-all-keys/

https://leetcode.com/problems/trapping-rain-water-ii/

https://leetcode.com/problems/01-matrix/

https://leetcode.com/problems/open-the-lock/

৪র্থ মাসঃ System Design, Dynamic Programming, Leadership Principle and More Practice

Leadership Principle: আমাজনের Data Structures and Algorithm প্রশ্ন Google / Facebook থেকে সাধারনত সহজ হয়। Amazon এ Leadership Principle এ অনেক বেশী গুরুত্ব দেয়। আমার প্রথম টার্গেট ছিল Amazon, তাই Leadership Principle এ অনেক সময় দিয়েছি। এই লিংকে দুইটায় https://medium.com/@scarletinked/are-you-the-leader-were-looking-for-interviewing-at-amazon-8301d787815d ও https://kraftshala.com/what-questions-to-expect-in-amazon-interview/ মোটামুটি বিস্তারিত আছে। এইটা থেকে আইডিয়া নিয়ে আমি নিজের মত করে প্রশ্নের উত্তর তৈরী করেছি। অবশ্যই সব উত্তরে যাতে ২/৩ টা Leadership Principle থাকে, এইটা মাথায় রাখতে হবে।

System Design: System Design এর জন্যও প্রস্তুতি নেয়া শুরু করি এক মাসে। প্রথমে [৭,৮] এই দুইটার সব বিষয় পড়া শেষ করি। System Design এ যেই বিষয়গুলো মাথায় জানতে হবে, তা হলঃ Scalability: Horizontal vs Vertical, Load Balancing, Caching: Redis vs Memcached, Content Distribution Network, Data Partitioning: Horizontal vs Vertical, Indexing, Proxy Server, Redundancy, Replication, Deduplication, SQL vs NoSQL, CAP Theorem, Consistent Hashing, Long-Polling vs WebSockets. [৭] এ যে ডিজাইন আছে সেগুলো নিজে ডিজাইন করে কোন কোন components মিস করেছি সেগুলা দেখে নিতাম। একটা প্যাটার্নে নিজে বানিয়ে ওইটা সব সময় অনুসরণ করার চেষ্টা করতাম। প্রথমে, Storage, Bandwidth আর Traffic estimation। এরপরে Database ডিজাইন, High Level ডিজাইন। Client Side, Server, Storage, Load Balancer, Cache, Backup Storage এইগুলা একটার সাথে অন্যটার Data Flow ডিজাইন। এরপরে Data Storage কিভাবে করব, MysQL, Cassandra, Hadoop, MongoDB কোনটা কেন ভাল হবে, খারাপ হবে, Interviewer কে বার বার প্রশ্ন করে নিশ্চিত হতে হবে। এরপরে আসবে Data Sharding Technique, এখানেও Id based, User Based, Hash Based বিভিন্ন Technique এর Pros and Cons বলতে হবে। কোথায় কোথায় Load Balancer, Cache লাগবে বলতে হবে। অনেক জায়গায় connection এর জন্য http, long pooling, tcp, socket এইগুলা কোনটা কখন ভাল হবে বলতে হবে। এছাড়া আরো কিছু টার্ম যেমন, Consistent Hashing, CAP Theorem, Deduplication এইগুলা কখনও Applicable হলে Interviewer কে বলতে হবে।

Dynamic Programming: Google এ খুব বেশি Dynamic Programming প্রশ্ন আসে। আমি Tabulation বা Bottom-Up DP ভালো পারি না, তাই Top-Down, Recursion + Memoization এ বেশী জোড় দিয়েছি। আমার টার্গেট থাকত, প্রথমে শুধু Recursion দিয়ে প্রবলেম সমাধান করা, পরে Memoization যুক্ত করে Optimization করা। আবার প্রথম থেকেই Memoization চিন্তা করলে অনেক সময় আমার ঝামেলা লেগে যায়। নিচে কিছু প্রবলেম আছে, যেগুলো Bottom-Up Approach এ করা যায়, Recursion দিয়েও করা যায়, Recursion + Memoization দিয়েও হয়।

Bottom-Up DP:

https://leetcode.com/problems/regular-expression-matching/

https://leetcode.com/problems/distinct-subsequences/

https://leetcode.com/problems/burst-balloons/

https://leetcode.com/problems/decode-ways/

https://leetcode.com/problems/longest-palindromic-subsequence/

https://leetcode.com/problems/partition-to-k-equal-sum-subsets/

Top-Down DP 1D:

https://leetcode.com/problems/maximum-subarray/

https://leetcode.com/problems/maximum-product-subarray/

https://leetcode.com/problems/house-robber-ii/

https://leetcode.com/problems/minimum-cost-for-tickets/

https://leetcode.com/problems/best-time-to-buy-and-sell-stock/

Top-Down DP 2D:

https://leetcode.com/problems/maximal-rectangle/

https://leetcode.com/problems/palindromic-substrings/

https://leetcode.com/problems/longest-palindromic-substring/

https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-cooldown/

ইন্টারভিউতে সাধারনত প্রতি রাউন্ডে ২ টা প্রশ্ন করে, দ্বিতীয় প্রশ্ন বেশীর ভাগ সময়ে প্রথম প্রশ্নের Optimization করতে বলা হয়। LeetCode এ অনেক প্রবলেমেরই ২ বা তার বেশী ভার্শন আছে। এই ধরনের প্রবলেম বেশী করে সমাধান করা ভাল, এতে ইন্টারভিউতে Follow Up প্রশ্নের আইডিয়া হয়।

https://leetcode.com/problems/word-break/ and https://leetcode.com/problems/word-break-ii/

https://leetcode.com/problems/word-search-i/ and https://leetcode.com/problems/word-search-ii/

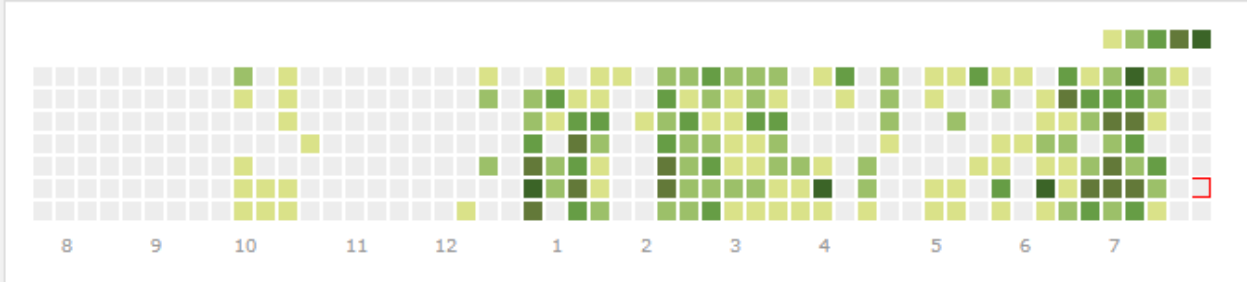
https://leetcode.com/problems/unique-paths/ and https://leetcode.com/problems/unique-paths-ii/

https://leetcode.com/problems/range-sum-query-mutable/ and https://leetcode.com/problems/range-sum-query-2d-mutable/

https://leetcode.com/problems/course-schedule/ and https://leetcode.com/problems/course-schedule-ii/

৫ম মাসঃ API Design, More Pramp, More Practice

এই মাসে কিছু অনিবার্য কারণবশত একটু Off track ছিলাম, যাই হোক পরে আবার On Track হয়েছি। প্রস্তুতিতে জোয়ার-ভাটা থাকাও স্বাভাবিক।



API Design:

API Design প্রশ্ন খুব কমন। আবার অনেক প্রবলেমে API call করে ডাটা নিতে হয়। API Design এ Interviewer কে অনেক প্রশ্ন করতে হয়। আমার Amazon + Google এর Onsite Round এর 4 out of 9 ছিল API Design প্রবলেম।

https://leetcode.com/problems/add-and-search-word-data-structure-design/

https://leetcode.com/problems/flatten-2d-vector/

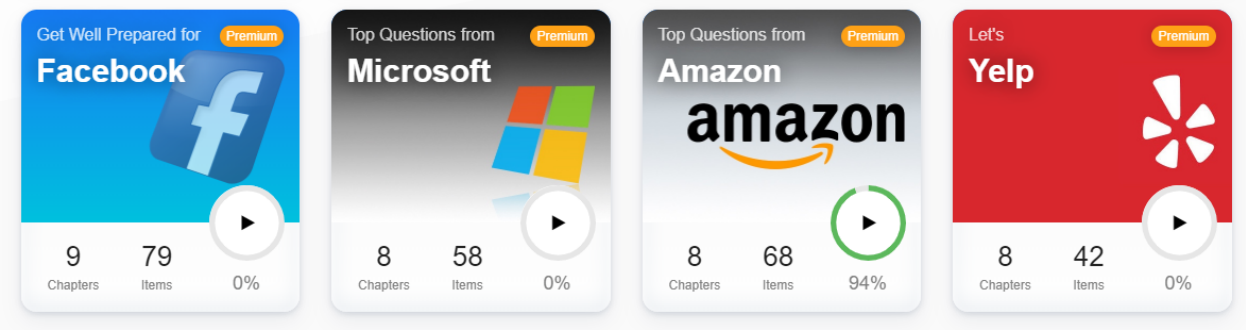
https://leetcode.com/problems/serialize-and-deserialize-binary-tree/

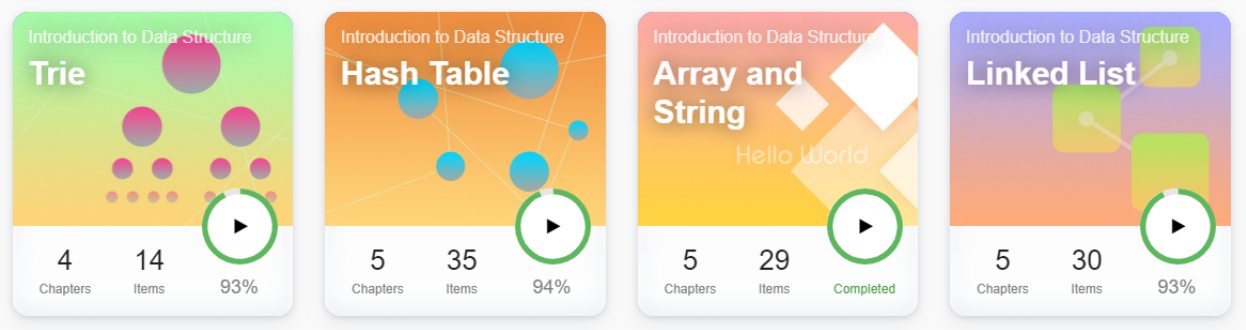
https://leetcode.com/problems/logger-rate-limiter/

https://leetcode.com/problems/design-hit-counter/

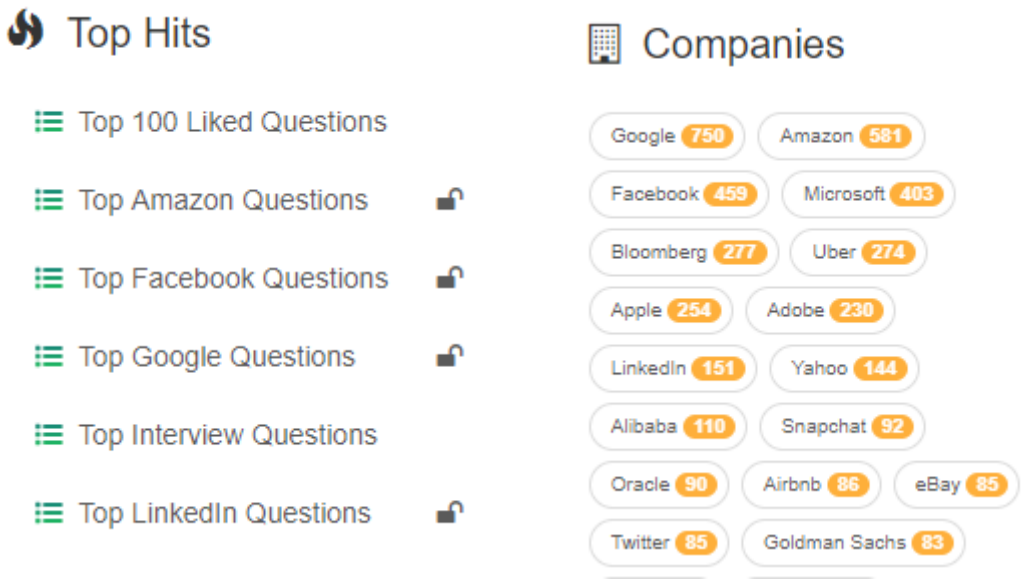
৬ষ্ঠ মাস এবং Check Local Lasting:

এই মাসে LeetCode Premium Account এর বিভিন্ন কোম্পানির জন্য আলাদা Preparation এবং বিষয় ভিত্তিক Preparation এর কোর্সগুলো করেছি।





আর Pramp এ নিয়মিত ইন্টারভিউ দিতাম। LeetCode এ বিভিন্ন কোম্পানির Top Questions প্রবলেম থেকে Amazon ও Google এর জন্য সমাধান করেছি, আগের করা থাকলেও আবার করেছি। এছাড়া Google ও Amazon tag দেয়া প্রবলেম থেকে Most Frequent গুলো করেছি।



অনেক প্রবলেম TAG করেও রাখতাম, ওইগুলা আবার Revise দিয়েছি। অনেক প্রবলেম আমি হয়ত সমাধান দেখে আগে করেছি, এখন আবার নিজে নিজে করার চেষ্টা করতাম।

LeetCode এ Discuss Section এ Interview Experience https://leetcode.com/discuss/interview-experience?currentPage=1&orderBy=hot&query= থেকে Google আর Amazon এর সব Experience পড়েছি। এমন না যে, প্রবলেম কমন পড়বে, এইটা করায় মোটামুটি আইডিয়া পাওয়া যায়, ইন্টারভিউ কেমন হতে পারে। কোন প্রবলেম Interesting মনে হলে সমাধান করেছি। Glassdoor এও অনেকে Experience Share করে, https://www.glassdoor.com/Interview/Google-Interview-Questions-E9079.htm https://www.glassdoor.com/Interview/Amazon-Interview-Questions-E6036.htm. Glassdoor থেকেও গত ২ বছরে Google / Amazon এর প্রায় সব ইন্টারভিউ পড়েছি।

Interview Experience:

আমার প্রথম Online Round ইন্টারভিউ ছিল মার্চ মাসে এবং Online Round এই Reject। এর পরে এপ্রিলে আরো দুইটা Phone Screen ইন্টারভিউ ছিল, ওইগুলোও Phone Screen এরপরে Reject। আমার টার্গেট ছিল, মে মাসের দিকে Big N গুলোতে আবেদন করব, যাতে প্রস্তুতির জন্য যথেষ্ট সময় পাওয়া যায়। জুন মাসে আরো দুইটা কোম্পানিতে Phone Screen ইন্টারভিউ ছিল। একটায় ২ টা, আরেকটায় ৩টা

Amazon এর Online Round ছিল জুন মাসে, আর Onsite ইন্টারভিউ ছিল জুলাই এর ৩য় সপ্তাহে।

কোডিংঃPhone Screen এ Shared Doc আর Onsite এ Whiteboard এ Code লিখতে হয়। তাই কাগজ-কলমে Code লিখে নিজে নিজে লাইন বাই লাইন Code Execute করা প্র্যাকটিস করতে হবে। আমি প্রতিদিন ৪/৫ টা Code কাগজ-কলমে Execute করতাম, যখন মনে Code ঠিক আছে, তখন LeetCode এ লিখে Run না করেই Submit দিতাম। Accept না হলে আবার কাগজ-কলমে ভুল ঠিক করার চেষ্টা করতাম। আমি যেহেতু Whiteboard এ ক্লাস নিয়ে

অভ্যস্ত তাই, Whiteboard বেশী প্র্যাকটিস দিতে হয় নাই, ইন্টারভিউর ৭/৮ দিন আগে Whiteboard কিনে প্র্যাকটিস করেছি। Whiteboard এ প্র্যাকটিসও করাও জরুরী। অনেকে Whiteboard এ লেখা বেশী ছোট বা বড় করে ফেলে, আবার Whiteboard এর কোন জায়গা থেকে লেখা শুরু করা উচিত, এগুলো প্র্যাকটিস না দিলে হবে না।

আগেই লিখেছি, Code Reusable, Modular, Production Ready হতে হয়। Null Value, Empty String, Negative/ Zero Value এগুলো Exception Handling দিয়ে করা ভালো। Code করা শেষ হলে নিজে কিছু Test Case Generate করে Unit Test করা উচিত।

অনেক সময় বড় Code হলে Interviewer কে বল কিছু অংশ function লিখে Skip করা যেতে পারে। যেমন কোন প্রবলেমে হয়ত Tries লাগবে, সেক্ষেত্রে Tries এর Insert(), Search() শুধু function নামে লিখে, পরে implement করা যেতে পারে। তবে অবশ্যই Interviewer বলতে হবে। Variable এর নামও meaningful এবং যাতে বেশী বড় না হয়, খেয়াল রাখতে হবে। কোন প্রবলেম করার সময় প্রথম থেকেই modular করার কথা চিন্তা করা উচিত। এমন না যে, প্রথমে Rough Code করে পরে modular করব, এটা করার সময় হবে না।

আমি অনেক প্রবলেম ৮/১০ বার করেও করেছি, বিভিন্ন ভাবে। এইটা আমার কাছে একটা প্রবলেমে বিভিন্ন ভাবে দেখতে সাহায্য করেছে এবং এক কে বার হয়ত এক এক ভাবে করার চেষ্টা করতাম। অনেক প্রবলেম রিভিউ দেওয়ার সময় হয়ত মনে মনে সমধানটা চিন্তা করতাম। যদি Confident মনে হত তখন Code দেখে নিশ্চিত হয়ে নিতাম। আর Confident না হলে Code লিখে সমাধান করার চেষ্টা করতাম। এভাবে কিছু কিছু প্রবলেম এত বেশী করেছি যে, ঘুমের মধ্যে জিজ্ঞেস করলেও Code করতে পারতাম।

System Design: Grokking the System Design কোর্স করলে মোটামুটি আইডিয়া হবে System Design সম্পর্কে। এরপরে আমি যা করেছি তা হল, নিজে একটা প্যাটার্ন তৈরী করেছি, কিছু Check List। 1. Requirements Analysis, 2. Storage, 3. Bandwidth and Traffic Estimation, 4. High Level Design, 5. Component Design, 6. Database, 7. Data Transfer, 8. Data Partitioning, 9. Cache, 10. Load Balancer, 11. Fault Tolerance and Replication, 12. Deduplication. এরপরে সব ডিজাইনে এই Check List অনুসরণ করতাম। কিছু কিছু Design এ অবশ্য এগুলো ছাড়াও কিছু Component থাকে। যেমন, Uber এর মত Ride Share App ডিজাইনে Quad Tree Data Structure লাগে। System Design এ Interviewer এর সাথে খুব Engaging হতে হবে। যে কোন Component ফাইনাল করার আগে Interviewer কে Alternate অপশন ব্যাখ্যা করে কেন কোন Particular টা ব্যবহার করা ভালো বলতে হবে। যেমন messenger ডিজাইনে message কি ধরনের NoSQL এ থাকবে, Column Based, Document Based, Key-Value Based, Graph Based বলার পরে বলতে হবে কেন Column Based এ ক্ষেত্রে ভালো। Storage, Bandwidth, Traffic Estimation এ Realistic Assumption করতে হবে। Data Transfer এ Push, Pull, Hybrid অনেক উপায় আছে। Data Sharding এরও অনেক Technique আছে, যেমনঃ Instagram ডিজাইনে Photo ID, User ID, Timestamp অনেক ভাবে Data Sharding করা যায়। সবগুলো Interviewer এর সাথে আলোচনা করে একটা Select করতে হবে। Cache, Load Balancer অনেক জায়গায় বসানো যায়। কেন, কিভাবে Cache পুরো ডিজাইনকে Faster করে এবং কোন ধরনের Caching Algorithm ব্যবহার করা ভালো, এগুলো বলতে হবে। http://highscalability.com/ এই সাইটে System Design এর অনেক তথ্য আছে।

http://highscalability.com/amazon-architecture

http://highscalability.com/google-architecture

http://highscalability.com/youtube-architecture

http://highscalability.com/blog/2016/6/27/how-facebook-live-streams-to-800000-simultaneous-viewers.html

http://highscalability.com/scaling-twitter-making-twitter-10000-percent-faster

http://highscalability.com/blog/2014/2/26/the-whatsapp-architecture-facebook-bought-for-19-billion.html

http://highscalability.com/blog/2015/9/14/how-uber-scales-their-real-time-market-platform.html

http://highscalability.com/blog/2011/12/19/how-twitter-stores-250-million-tweets-a-day-using-mysql.html

https://instagram-engineering.com/what-powers-instagram-hundreds-of-instances-dozens-of-technologies-adf2e22da2ad

https://www.youtube.com/watch?v=PE4gwstWhmc

Object Oriented Design: Object Oriented Design প্রশ্ন অনেক সময় আসে। আমি এটার জন্য তেমন ভালো করে কোন প্রস্তুতি নেই নাই। Cracking The Coding Interview বইয়ে একটা অধ্যায় আছে, Object Oriented Design নিয়ে, শুধু ওইটা দেখেছি। আমাকে Google এর Recruiter বলেছিল আমাকে শুধু Data Structure and Algorithm প্রশ্ন করবে। আর Amazon ও বলেছিল শুধু Data Structure and Algorithm আর System Design থাকবে। Recruiter এর সাথে কথা বলে ইন্টারভিউতে কি ধরনের প্রশ্ন আসবে নিশ্চিত হয়ে নেয়া ভালো।

Leadership Principle (LP): LP প্রশ্নে সবসময় নিজের অভিজ্ঞতা বলতে হবে। যদি অভিজ্ঞতা নাও থাকে, তারপরও এই ধরনের Situation এ কি করতাম, সেটা বলতে হবে। আগে থেকে LP প্রশ্ন দেখে উত্তর তৈরী করতে হবে। অনেক সময় একই ধরনের প্রশ্ন পর পর রাউন্ডে আসতে পারে। সেক্ষেত্রে Interviewer কে বলতে হবে, আগের উত্তর দেয়া যাবে, নাকি ভিন্ন উত্তর দিতে হবে।

Job Offer: ইন্টারভিউ দেয়ার পরে Phone Screen এর ক্ষেত্রে সাধারনত ১ দিন পরেই Recruiter রা ফলাফল জানায়। Onsite এর ফলাফল অনেক কিছুর উপর নির্ভর করে। আমার Amazon ইন্টারভিউ ছিল Hiring Event, এর জন্য ৫ দিনের ভিতরে ফলাফল জানতে পেরেছি। Google এ Recruiter প্রথমে বলেছিল ২/৩ সপ্তাহ লাগতে পারে, কিন্তু যেহেতু আমার Amazon এর Job Offer ছিল, এর জন্য Google এও ১ সপ্তাহের ভিতর ফলাফল পেয়েছি। আর একটা Start up, UiPath থেকে ফলাফল পেতে প্রায় ১ মাস লেগেছে। তখন হাতে কোন Job Offer ছিল না। প্রথমে Recruiter রা জানায় যে Offer Yes / No, কিন্তু Official Offer পেতে হয়ত আরো সময় লাগে। এর মাঝে Team Matching এর ব্যাপার থাকে। ২/৩ জন Team Manager এর সাথে কথা বলে যে কোন Team Choose করা যায়। অবশ্য এটাও অনেক কিছুর উপর নির্ভর করে। আমাকে Amazon এ Team Choose এর সুযোগ দেয়া হয়েছিল, কিন্তু Google এ সময় কম থাকায় একটা মাত্র Team এর সাথে কথা বলার সুযোগ হইছে। তবে এইসব কোম্পানিতে Team Switch করা সহজ। Team Choose হওয়ার পরে সাধারনত Official Offer দিয়ে একটা সময় দেয়া হয় Offer Accept / Decline করার। অনেকের

Multiple Offer থাকলে তখন চিন্তা ভাবনা করে সিদ্ধান্ত নিতে হয় বা অনেক সময় Compensation and Benefits নিয়ে দরদামও করা যায়।

Salary Negotiation: প্রথমে একটু অদ্ভুত মনে হলেও এখানে চাকরিতে Compensation and Benefits নিয়ে Negotiation করা যায়। কোন নির্দিষ্ট Job Position এর জন্য হয়ত একটা গড় Compensation থাকে, কিন্তু তার মানে এই না যে এটাই Fixed. এই সাইটে https://www.levels.fyi/ এ বিষয়ের অনেক তথ্য আছে। প্রথম কথা হচ্ছে আগে, নিশ্চিত হতে হবে, আমাকে যে Compensation and Benefits দিচ্ছে সেটা Market Price এর সাথে সামাঞ্জস্যপূর্ন কি না। যদি না হয়, তাহলে অবশ্যই Recruiter কে জানাতে হবে। যদি মনে হয় Market Price এর সাথে সামাঞ্জস্যপূর্ন এবং হাতে আর কোন Job Offer নাই, সেক্ষেত্রে হয়ত Compensation and Benefits নিয়ে Negotiation করার সুযোগ কম। আমার হাতে ৩টা Job Offer থাকায় Compensation and Benefits নিয়ে Negotiation করার সুযোগ ছিল।

পরিশিষ্টঃ

আমি বলব চাকরির জন্য ৯৫% পরিশ্রম আর বাকি ৫% কপাল, কিন্তু ইন্টারভিউর সময় ৫% কপালের Weight ৯৫% আর ৯৫% পরিশ্রমের Weight ৫%। কারন ইন্টারভিউর দিন অনেক কিছুই হতে পারে। খুব কঠিন প্রশ্ন কিন্তু কমন পরতে পারে, আবার খুব সহয প্রশ্ন Brain Freeze হয়ে যেতে পারে, অন্য প্রার্থীদের তুলনামূলক ভালো/খারাপ হতে পারে, Interviewer এর মন মেজাজ ভালো/খারাপ হতে পারে। যেহেতু এই ৫% কপালের Weight ৯৫% এ আল্লাহ ছাড়া কারো হাত নেই, তাই বাকী ৯৫% পরিশ্রমের Weight ৫% কে যত Maximize করা যায়। আর এজন্য পরিশ্রমের বিকল্প নেই।

আমি যে প্রবলেমগুলোর লিংক দিয়েছি, এগুলো কোন Short List বা Suggestions না, আমি শুধুমাত্র প্রবলেমের বৈচিত্র্যতা বুঝানোর যে প্রবলেম আমার কাছে Interesting মনে হইছে সেগুলো দিয়েছি।

এগুলো পড়ায় অনেক Unexpected Situation সম্পর্কেও জানতে পেরেছি। যেমন একজন বলছে, Onsite এ Whiteboard এ ইন্টারভিউর সময় রুমের কোন মার্কার কাজ করেছিল না। Interviewer অন্য রুম থেকে নিয়ে আসতে গিয়ে ৫/৭ মিনিট চলে যায়, এই সময় কিন্তু দিবে না, এই ৫/৭ মিনিট পুরোটা সময় নষ্ট। মার্কার নিয়ে আমারও একটা Interesting অভিজ্ঞতা আছে, পরে বলব। ইন্টারভিউতে সময় খুবই গুরুত্বপূর্ণ, রেস্ট রুম বা পানি খেতে গেলে কিন্তু Allotted ৪৫ বা ১ ঘন্টা থেকেই সময় যাবে। ইন্টারভিউ আগে রেস্ট রুম আর পানির বোতল/চকলেট সাথে রাখা ভালো।

বিবিধ লিঙ্কঃ

1. https://hbr.org/2016/09/a-guide-to-cold-emailing, How to write cool email
2. https://www.job-hunt.org/recruiters/linkedin-recruiters2.shtml, How to add recruiter in LinkedIn
3. https://www.linkedin.com/pulse/how-connect-recruiters-linkedin-using-my-bar-strategy-anne-pryor/, How to add recruiter in LinkedIn
4. https://yangshun.github.io/tech-interview-handbook/introduction, Full preparation guide
5. http://www.codespaghetti.com/interview-success, How to success in Technical Interview
6. http://blog.gainlo.co/index.php/category/system-design-interview-questions/, System Design

7. https://leetcode.com/problems/combination-sum/discuss/16502/A-general-approach-to-backtracking-questions-in-Java-(Subsets-Permutations-Combination-Sum-Palindrome-Partitioning), LeetCode, Backtracking Problem

8. https://developers.google.com/machine-learning/glossary/, Machine Learning Glossary

9.