INTERVIEW PACKET

SDE 2_3 Backend Guide

The evaluation process:

Rounds:
Typically, the process involves 3 technical rounds including interaction with HM.

- **Round 1**: 60 mins DSA interview round Conducted online. Medium level questions asked. Typically, 1 - 2 questions in the interview.
  Questions can be around any primary data structures- Linked Lists, Trees, Arrays, Queues, Maps, Sets etc.
- **Round 2**: 60-minute System Design round. Conducted online.
  Overview of the system, entities involved, schema diagram, flow of system, interactions between services and different components etc
  Focus on implementing code for a design pattern. Secondary focus on locks, mutex, concurrent programming + Scaling- LLD + HLD
- **Round 3**: 60 min Hiring Manager round. Previous Project + Behavioral questions.
  Discuss previous work and projects. Deep dive into the decision-making process of the project, technologies used. What would have been done differently. Some behavioral questions.
  Questions about tech stack - Kafka, Redis, Microservices Vs Monolithic etc

Round-wise details:

Round 1: Problem Solving /DS Algo Round

Sample Questions:

Q1. Given an integer array nums, return the length of the longest strictly increasing subsequence.

https://leetcode.com/problems/longest-increasing-subsequence/description/

Example 1
Input: *nums = [10,9,2,5,3,7,101,18]*
Output: *4*
Explanation: *The longest increasing subsequence is [2,3,7,101], therefore the length is 4.*

Example 2
Input: *nums = [0,1,0,3,2,3]*
Output: *4*

Example 3
Input: *nums = [7,7,7,7,7,7,7]*
Output: *1*

Follow up: *Can you come up with an algorithm that runs in O(n log(n)) time complexity?*

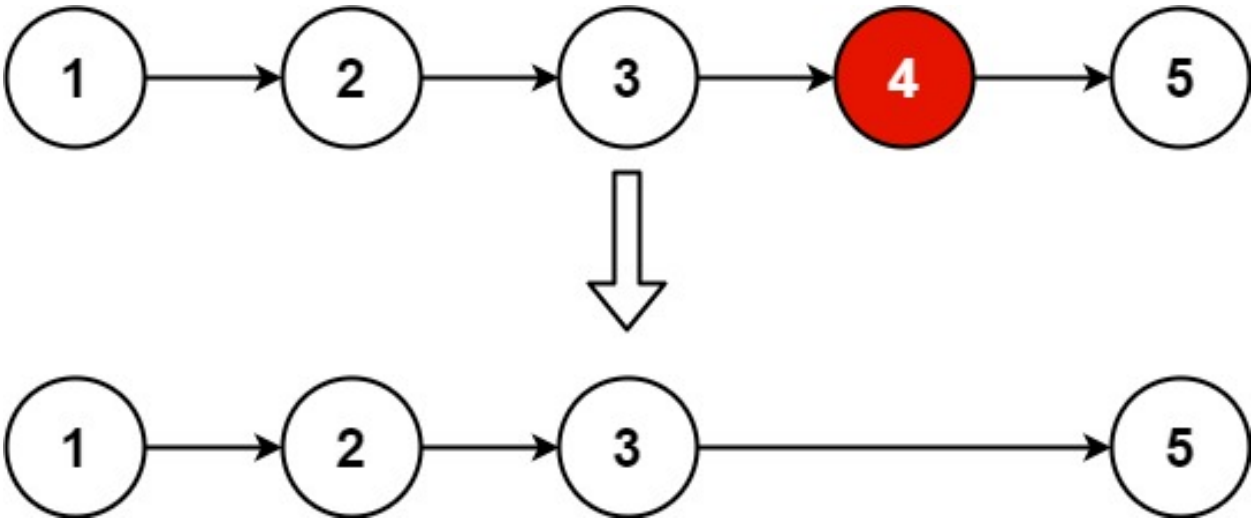
Q2. Given the head of a linked list, remove the n_{th} node from the end of the list and return its head

https://leetcode.com/problems/remove-nth-node-from-end-of-list/

.

Example 1

Input: *head = [1,2,3,4,5], n = 2*
Output: *[1,2,3,5]*



Example 2

Input: head = [1], n = 1
Output: []

Example 3

Input: head = [1,2], n = 1
Output: [1]

Follow up: Could you do this in one pass?

Sample answers:

Q1.

public int lengthOfLIS(int[] nums) {

```
    int n = nums.length;
    int[] dp = new int[n]; // dp[i] stores the length of the longest increasing subsequence ending at
index i
    int len = 0; // len stores the length of the longest increasing subsequence found so far

    for (int i = 0; i < n; i++) {
        int index = Arrays.binarySearch(dp, 0, len, nums[i]);
        if (index < 0) {
            index = -(index + 1);
        }
        dp[index] = nums[i];
        if (index == len) {
            len++;
        }
    }

    return len;
}
```

In this implementation, we use an array `dp` to store the length of the longest increasing subsequence ending at each index. We also use a variable `len` to keep track of the length of the longest increasing subsequence found so far.
We iterate through the array `nums`, and for each element `nums[i]`, we use binary search to find the index `index` where `nums[i]` should be inserted into the `dp` array to maintain its sorted order. If `nums[i]` is not found in the `dp` array, the binary search method returns a negative index, which we can convert to the correct insertion index by taking the complement and negating it. We then update `dp[index]` to `nums[i]`.
If `index` is equal to `len`, this means that we have found a new longest increasing subsequence, and we update `len` accordingly.
Finally, we return `len`, which represents the length of the longest increasing subsequence found.

More answer reference- https://leetcode.com/problems/longest-increasing-subsequence/solutions/3175790/java-medium/

Q2.

```
class ListNode {
    int val;
    ListNode next;
    ListNode(int x) { val = x; }
}

public ListNode removeNthFromEnd(ListNode head, int n) {
    ListNode fast = head;
    ListNode slow = head;
```

```
    // Move fast pointer n nodes ahead of the slow pointer
    for (int i = 0; i < n; i++) {
        fast = fast.next;
    }

    // If the fast pointer is null, we need to remove the first node
    if (fast == null) {
        return head.next;
    }

    // Move both pointers until the fast pointer reaches the end of the list
    while (fast.next != null) {
        fast = fast.next;
        slow = slow.next;
    }

    // Remove the nth node from the end of the list
    slow.next = slow.next.next;

    return head;
}
```

In this implementation, we use two pointers, a `fast` pointer and a `slow` pointer. We move the `fast` pointer `n` nodes ahead of the `slow` pointer, and then move both pointers until the `fast` pointer reaches the end of the list. At this point, the `slow` pointer will be pointing to the nth node from the end, and we can remove it by setting its `next` pointer to the node after it. If we need to remove the first node, we can simply return the second node.

More answer reference- https://leetcode.com/problems/remove-nth-node-from-end-of-list/solutions/3081594/java-two-pointer-simple-solution-beats-100-0ms-94-memory/

Why candidates fail:

- Directly start coding instead of discussing the approach with the interviewer.
- Deciding which data structures to use on the go, instead of deciding at first.
- Not enough practice in a time constraint environment.
- Start solving the wrong problem statement. Take 3-5 mins to understand the problem statement. There is no rush.
- Start writing code before thinking of the best structure of the code. Take time to structure your code before starting to code.

Tips:

- Address the issues mentioned in the section above.
- Be prepared with leetcode problem solving practice for at least 1-2 months consistently.
- Steps
    - Understand the problem
    - Decide the logic and data structures to be used

- Decide on storage of input data, traversal, manipulation/transformation/calculations if any, accessing the output and return format
- Dry the the logic and sample inputs
- Think of edge cases and try to cover
- Look over complexities and try to optimize

Round 2: System Design Round

LLD + HLD Combined (Mostly no coding involved)
Open ended discussion

Sample Questions:

Q1. Designing a wallet system for an e-commerce site.
    Credit
    Debit
    Account
Scale, transactional properties - ACID, SOLID etc
APIs,
DB Schema

Sample answers:

1. User table: This table stores information about users, including their ID, name, email address, password, and other relevant details.
2. Wallet table: This table stores information about wallets, including the wallet ID, user ID associated with the wallet, and wallet balance.
3. Payment transaction table: This table stores information about payment transactions made by users, including the transaction ID, user ID, payment method, payment amount, and transaction date.
4. Order table: This table stores information about orders placed by users, including the order ID, user ID, order amount, and order date.
5. Wallet transaction table: This table stores information about transactions made using the wallet, including the transaction ID, user ID, transaction type (debit/credit), transaction amount, transaction date, and the associated order ID (if applicable).

This schema allows us to track all the relevant information about users, wallets, payment transactions, and orders, and enables us to process transactions made using

the wallet and update the wallet balance accordingly. It also allows us to generate reports and analytics to gain insights into user behavior and transaction patterns.

Why candidates fail:

- Don't gather the requirements earlier. What are the features expected from system are not discussed with the interviewer, which leads to back and forth in later discussions. Don't use/ fail to mention the names of design patterns used. Aren't aware of concurrency constructs required to code. Mostly need to be aware of concurrent data structures.
- They go too wide - trying to design entire e-commerce site transactions instead of focusing on what the interviewer wants to focus on.
- They only answer the question asked and then expect the interviewer to poke further.
- Not considering failure cases. Assume only the happy case.

Tips:

- Start the interview by having a discussion on different requirements from the question. Jot down and clarify those requirements and do a handshake on those with the interviewer. Suggest different features wrt system to demonstrate your understanding about those and also to demonstrate how you may collaborate in a real work environment.
- Drive the discussion. Actively state what you are thinking - pros and cons of the approach you are suggesting.
    - This does not mean that you should continuously be talking. Think and then speak. But drive the discussion. Interviewer shouldn't have to keep poking questions to keep the discussion moving.
- Actively listen to the hints that the interviewer might be giving.
- Clarify the requirements of the question -> Quickly estimate the scale for which you are building -> Understand what design goals and then jump into the actual design. First 3 sections should not take more than 5-10 mins.

Round 3: Hiring Manager Round (Behavioral + Technical)

Sample Questions:

1. How would you describe your experiences with backend tech stack used at previous projects

2. How do you handle conflict or difficult situations in the workplace?

Technical

1. Microservices Vs Monolithic

2. Kafka implementation- topics, clusters,
3. Concurrent programming
4. NoSQL DBs
5. Microservice Design Patterns
6. Frameworks used including test case frameworks

Why candidates fail:

1. Going too wide for short answers.
2. Not understanding what the HM is trying to judge by a particular question.

Tips:

1. Hiring manager is trying to judge the candidate on overall fitment including technical capabilities he/she is bringing to the organization.
2. Research about all technologies you have mentioned on your resume. If you mention Kafka, you will be questioned on the internals of Kafka, or how Kafka can be used in a certain scenario.
3. Think about healthy disagreements with peers, managers and reportees (if applicable). Exact arguments pro and against the point made. How did you arrive at a consensus?
4. Actively think about how you have grown in the last 1 year, last 2 years, last 6 months. What are the new things you have learnt - both skills wise, and behavior wise.