Full Stack Development Interview Guide

Introductory Questions

Questions like ` Tell me about yourself ` are there in every interview. here's how we need to approach.

```
-> Connect personal strengths with supporting examples.
-> Be yourself, but be the best of yourself.
-> Avoid summarising your resume word for word.
-> Mention past experiences and proven successes.
-> Align your current job responsibilities to the role.
-> Avoid mentioning personal information related to your marital status, children, political or religious views.
-> Highlight your personality.
-> Avoid rushing into deeper conversations about the role and company.
-> Connect your skills to the job description.
-> Briefly mention about hobbies, intellectual development and community involvement.
```

Here's a better answer, don't copy make your own version.

```
"Hi, My name is ABC. I've recently done my Bachelors in Technology with Computer Science stream from ABC college.
As you'd have known – I am here for the role of Software developer. Coming to my skills I know  javascript and c++.
I had 2 internships as well in this particular domain where along with core technical things I also
 learned organizational ethics and discipline. Moreover, I also did 2 successful individual projects in web development in my college tenur
To be honest, my short-term goal is to get a job in a reputed company like _company_name_ , where I can use my skills and knowledge to deli
value-added result and in long term, the goal would be to achieve a good position in the particular company to scale up organization's
 production & growth rate and at the same time for the betterment of my personal career growth as well. Other than that, if I talk about my
strengths – I can say that I'm a quick learner, team player, adaptable and creative person – and I do guess that these strengths of
 mine are perfectly suitable for this job role. In my free time, I enjoy spending time with my family, reading, playing outdoor
sports, and sometimes cooking.

thanks for this opportunity."
```

DSA Questions

Many times candidates get rejected even after solving questions in interview. So while approaching a DSA question in interview one need to know that

DSA interviews are not judged only if you solved.

The first thing to understand is what are the interviewers interested in so that we can address them accordingly. The interviewer usually evaluates you on the following evaluation criteria (EC)

[EC1] — Ability to understand the question

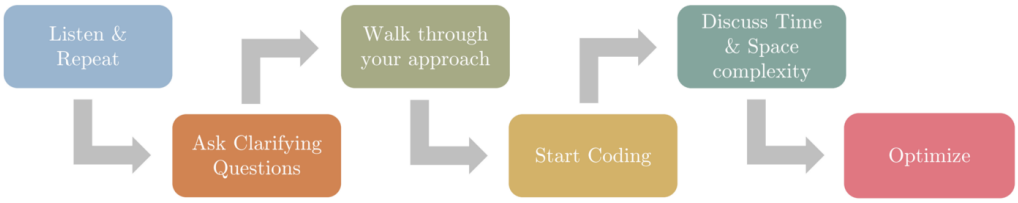
[EC2] — Ability to gather information when needed

[EC3] — Your thought process when presented with a problem

[EC4] — Ability to code in an organized/modular manner

[EC5] — Your coding skills and ability to assess the run-time constraints

[EC6] — Ability to realize improvements in your work



Here's how few tips to ace DSA interview

1. Listen to the question:

The interviewer will explain the question and go through a toy example to help you understand the problem. While the interviewer explains it, note down the important points on a separate sheet of paper, that you think are key takeaways.

Target: [EC1]

2. Talk about your understanding of the question:

After listening to the question, DO NOT jump in and start coding (even if you know exactly what the problem is and how to solve it). Instead, repeat the question and confirm your understanding. Ask clarifying questions such as

1. What are the Input/Output data type limitations?
2. Is there any restriction on the Input size/length?
3. What happens if the input is invalid?
4. Corner/special cases: Problem-specific questions such as what happens when you see a non-digit character in a string where the two digit-strings needed to be multiplied? (Do you consider it as invalid input, or do you ignore the non-digit character?)

Most of the time the interviewer does not give you all the required information. One of the things that the interviewer is looking for in a candidate is ***being able to ask the right questions to gather all the necessary information***

Target: [EC1], [EC2]

3. Discuss your approach:

Discuss your approach to the problem and ask the interviewer if he/she agrees with it. Talk about the data structure you prefer to use and the reason behind it. Discuss the pseudo-code with the interviewer.

Target: [EC3]

4. Start coding:

Always ask the interviewer before you actually start coding. Define useful functions and explain as you write the code. ***The most important thing while you code is to think out loud so the interviewer can evaluate your thought process.*** With each line of the code, say out loud why you are using it and how this choice will impact the code output. For example, while writing a for loop, say

"Now we will define a for loop to iterate over the xyz list one element at a time so that we can process each element according to the desired output [or problem-specific reason]"

In case you are stuck somewhere, the interviewer will drop in subtle hints. Make sure you pay attention to those hints.

Target: [EC4], [EC5]

5. Discuss the time and space complexity

Discuss the time and space complexity of your code in terms of Big O for your approach. This useful resource can be a good starting point for beginners. Try to break down your code into chunks, discuss the time complexity, and then talk about the overall time complexity of the code.

Target: [EC5]

6. Optimise the approach (if possible or the interviewer suggests)

After discussing the complexity of your code, the interviewer can ask you to improve it, if your approach is not already optimised. The interviewer will drop by hints highlighting the chunk of code that can be improved. Make sure you pay attention to that.

Target: [EC5], [EC6]

Few additional tips -

> **-> try not modify the provided data. (if alternative solution has same complexity do that)**
> **-> do not declare global variables**

> **-> if repeated code ⇒ make function**
> **-> if using repeated datas structure like graph or segment tree ⇒ make a class and use objects.**
> **-> make test-cases and cross verify with interviewer if you got the problem statement.**
> **-> speak out brute fore first, and then further optimise.**
> **-> function/variable name should proper and self explaining.**
> **-> think of edge cases, and dry run on it.**

Important Data Structures

> **- Arrays**
> **- Strings**
> **- Linked List**
> **- Hash maps**
> **- Stack**
> **- Queue**
> **- Trees**
> **- Graphs**

Important Algorithms/Techniques

> **- Sorting algorithms**
> **- DFS and BFS**
> **- Recursion**
> **- Binary Search**
> **- Sliding Window and Two Pointers**
> **- Dynamic programming**

Theoretical question / Core Subject question

Most of core subject based questions are definition’s, difference between two like questions. For answering these questions you must know these.

Few example questions :

- OOP vs Procedural
- What is Polymorphism
- Explain Normalisation

So, it’s pretty much clear that to answer these questions we must have knowledge about them.

Now if you know, here’s how you should answer

→ Make sure you understand the question. Ask the interviewer a question if you have any doubt in the question.
→ Once you get the question, answer to the point, be crisp and brief in your answer.
→ Don't give a bookish answer. Complement it with an example.

if you don’t know

→ Incase you don't know answer, Don’t keep quiet, keep the conversation going, admit that you don’t know and tell what you know about that topic instead
(it will ensure the interviewer that you have knowledge about that topic except that one question)

→ Don't pretend that you know everything. You may say that "Sir, I don't know the answer but will surely look into it."

rather than bluffing the answer. Don't answer illogical, rather be true to whatever you answer.

Must know topics / concepts

Object Oriented Programming (OOP)

> **What is OOP ?**
> **OOP vs Procedural**
> **Object vs Class**
> **Features of OOP**
> **Encapsulation**
> **Abstraction**
> **Polymorphism**
> **Inheritance**
> **Compile time polymorphism vs runtime polymorphism**
> **Types of inheritance**
> **Constructors vs methods**
> **This pointer**
> **Destructors**
> **Abstract class vs Interfaces**
> **Friends function**
> **Overriding**
> **Object Oriented Design of Parking System**

Operating System

> **Process vs Program**
> **Context Switching**
> **Process table and Process control block**
> **CPU Scheduling Algorithms**
> **Preamptive vs non preamptive scheduling**
> **Conveys effect and Belady's anamoly**
> **Race condition vs critical section**
> **Semaphores vs mutex**
> **Inter process communication**
> **Deadlock (Detection, recovery, prevention, avoidance)**
> **Bankers algorithm**
> **Threads vs Process**
> **Multiprogramming vs Multitasking vs Multi-threading**
> **Fork() and exec() system calls**
> **Memory allocation (contiguous vs non contiguous)**
> **Static vs dynamic partitioning**
> **Segmentation vs Paging**
> **Memory management unit, paging and Page Table**
> **Demand paging, virtual paging and Page fault**

- **Thrashing**
- **Page replacement algorithms**
- **File allocation methods**
- **Disk scheduling algorithms**
- **Spooling**
- **Starvation, Ageing**
- **System Calls and Interrupts**

Computer Networks

- **Network topology**
- **Criteria to check network reliability**
- **What is Bandwidth**
- **What is Gateway**
- **Node and Link**
- **DNS**
- **Network Interface Card**
- **Subnetting**
- **Types of Networks**
- **Communication vs Transmission**
- **TCP vs UDP**
- **What is POP3**
- **IP vs MAC**
- **Encoder vs Decoder**
- **Unicast, Multicast, Any-cast and Broadcast**
- **Private IP vs Public IP**
- **OSI model and all its layers**
- **TCP/IP vs OSI model**
- **Multiplexing**
- **firewall**
- **What is Modem**
- **Ping and Trace Route**
- **FTP and Anonymous FTP**
- **Domain vs Workgroup**

DBMS (Do prepare for SQL queries)

- **Advantages of DBMS**
- **DBMS vs RDBMS**
- **Disadvantages of File Processing System**
- **Entity**
- **ER Model**
- **Relationships (one-one, many-one etc)**
- **DDL vs DML**

> **OLAP vs OLTP**
> **Constraints (different types of keys)**
> **Functional Dependency**
> **Normalisation**
> **1NF , 2NF , 3NF and BCNF**
> **Atomicity**
> **Concurrency control**
> **Data Abstraction and its levels**
> **Transactions and its phases**
> **Serializable vs Non Serializable vs Conflict Serializable**
> **Exclusive lock vs Shared Lock**
> **Inner Joins, Left Joins, Right Joins and Full Join**
> **Raid Technology**
> **Aggregate functions**
> **Stored Procedures**
> **Triggers**
> **Applications of Triggers**
> **Indexing**
> **Different types of indexing**
> **Sharding**
> **Data warehousing**
> **SQL vs NOsql**

Miscellaneous

> **Vertical Scaling vs Horizontal Scaling**
> **CAP theorem**
> **CIA Triad**
> **Forward Proxy vs Reverse Proxy**
> **Load Balancers**
> **Consistant Hashing**
> **Virtual machines vs Container vs Bare metal**
> **System Design of Pastebin**
> **System Design of URL shortener**

Core JavaScript Questions/ Topics

> **Is JavaScript Single Threaded ?**
> **What makes JavaScript Asynchronous ?**
> **Explain Event Loop**
> **CallStack, Callback Queue and Microtask Queue**
> **What is WebApi in javaScript ?**
> **Explain how a code in javaScript executes (ex : expalin each step of execution of settimeout(console.log(100), 100)**

What is Execution Context
Explain Hoisting
Explain this keyword
What is Temporal Dead Zone
Block Scope Shadowing
Closures, properties and Uses
Constructors
SetTimeout
Garbage Collection
Function and Types
First Class Functions
Callbacks
Higher Order Functions
Map , Filter ,Reduce

React Js Questions/ Topics

Virtual DOM
What are porps
What is JSX , How it works
Use State
Prop Drilling
What are hooks
Rules to Use Hooks
How useEffect Works
Why do React Hooks make use of refs?
Custom Hooks
useMemo()
Lazy Loading
Methods to pass data between components
Higher order components
Component Lifecycle
How to pass data between sibling components using React router?

Html Questions /Topics (Easy/ Median)

What are self closing tags
What are void elements in HTML
How to optimize website assets loading?
Define multipart form data?
What is the ‘class’ attribute in HTML?
Are the HTML tags and elements the same thing?
Section, div, article

What is character encoding
What is purpose of alt attribute on image
What is DOM
What are data- attributes good for
What is iframe used for
difference between section and div
Can a page contain multiple header and footer
What is difference between cookie, local storage, and sessionStorage
Does local storage throw error after reaches limits
What are the various formatting tags in HTML?
Is it possible to change an inline element into a block level element?

CSS questions/Topics

What is the Box model in CSS? Which CSS properties are a part of it?
What are the different types of Selectors in CSS?
What is a CSS Preprocessor? What are Sass, Less, and Stylus? Why do people use them?
What is VH/VW (viewport height/ viewport width) in CSS?
What is the difference between inline, inline-block, and block?
Difference between reset vs normalize CSS?. How do they differ?
How do you specify units in the CSS?. What are the different ways to do it?
What property is used for changing the font face?
Does margin-top or margin-bottom have an effect on inline elements?
How are the CSS selectors matched against the elements by the browser?
How is border-box different from content-box?
How is opacity specified in CSS3?
What do the following CSS selectors mean?
What are the properties of flexbox?
Explain CSS position property?
What is the grid system?

NodeJs Backend Questions /Topics

local vs global packages
Features of node.js
What is Callback Hell and what is the main cause of it?
What is Libuv.
If Node.js is single-threaded, then how does it handle concurrency?
What are the advantages of using promises instead of callbacks?
npm vs yarn
purpose of module.exports
Explain the steps how “Control Flow” controls the functions calls?
What does event-driven programming mean?
What is the package.json file?

> **What is the difference between fork() and spawn() methods in Node.js?**
> **What is a test pyramid in Node.js?**
> **What is the purpose of NODE_ENV?**
> **What is fork in node JS?**
> **How does Node.js overcome the problem of blocking of I/O operations?**
> **What is middleware?**
> **What is express ?**
> **What is a thread pool and which library handles it in Node.js ?**
> **How to measure the performance of async operations?**

Project Based Questions / Topics

Projects are one of the most important aspects while applying for a job be it in the resume shortlisting process or in the interview process.
In this video, I'll be sharing what all questions can be asked about your project and how can you answer that.

> **-> Tell the purpose of your project.**
> **-> Tell about different features of your project / motivation behind project.**
> **-> Impact / Metrics (like downloads on play-store, reviews)**
> **-> Pre-prepare expected question on project.**
> **-> Technology used (why react instead of angular etc)**
> **-> Challenging part on project (Tech challenge/ Team wise challenge)**
> **-> Flow / end to end working of any component (ex. on clicking button tell me what are the events happening in the whole project)**
> **-> Future improvements**
> **-> Deploy your project, share link**

Behavioural Questions

1. **Prepare some stories**. You should have a few relevant stories from your professional past in your pocket for a serious interview. This way, you'll be able to plan your response to highlight your positive attributes and behaviour. You also won't have to think of a story on the spot, which can be stressful.

2. **Mine the job description for the desirable skills**. Studying the job description of the position you're applying for can reveal plenty of useful information about the skills a perfect candidate should have. If the position you're applying for involves working on tight deadlines, you may want to tell a story that demonstrates your time management skills. If a role has leadership potential, you should prepare a story about a time you successfully led a project or mentored someone.

3. **Use the STAR method**. The S.T.A.R. method is a helpful way to frame and present your stories. STAR stands for situation (i.e. the challenging scenario you found yourself in), task (what the situation asked of you), action (the action or reaction you had in the situation), and results (the outcome). Structuring your stories like this allows you to hit all of your talking points without leaving anything out.

4. **Own up to past mistakes**. Interviewers will often ask about times that you have been challenged, made mistakes, or have had to overcome your weaknesses. Be honest and use these questions as opportunities to show how much you have learned from your own mistakes.

5. **Take your time**. Relax and give yourself plenty of time to fully explain each of your stories. It can be easy to accidentally rush through a story, but try to avoid doing that. Make sure you clearly articulate every part of the situation you are describing.

6. **Stay positive**. Focus on the favorable aspects of your experiences. When you're explaining a time that you were faced with a challenging situation, try not to linger on the frustration you may have felt. Quickly move on to what action you took to mitigate the mistake, emphasizing your problem-solving abilities.

Some common behavioural Questions.

> **Describe a time when you disagreed with a team member. How did you resolve the problem?**
> **Tell me about a time when you failed.**
> **Give me an example of when you had to assume leadership for a team.**
> **What is the most difficult/ challenging situation you've ever had to resolved in the workplace?**
> **Tell me about a time when you disagreed with a supervisor.**
> **Describe how you used your problem-solving skills to benefit a team or company.**
> **How do you approach problems? What's your process?**
> **Are you better at working with a team or working on your own?**
> **What do you do if you disagree if another team member?**
> **Tell me about a time when you tried something risky and failed.**
> **Tell me about a time when you worked well under pressure.**
> **Describe a time when you faced a block at work and how you solved it.**

How to answer questions during interview

Theory Questions

What is the purpose of module.exports?

This is used to expose functions of a particular module or file to be used elsewhere in the project. This can be used to encapsulate all similar functions in a file which further improves the project structure.

For example, you have a file for all utils functions with util to get solutions in a different programming language of a problem statement

```
const getSolutionInJavaScript = () => {}
module.exports = { getSolutionInJavaScript }

Thus using module.exports we can use these functions in some other file:

const { getSolutionInJavaScript} = require("./utils")
```

Explain with code if code editor provided, other wise explain by some real life example

Coding Question Approach Example

Two Sum Problem (Frequently asked in interviews)

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order

1 - read the question and clarify if there's any doubt ask, gather required information
2 - Try to form some test cases (edge cases) ans clarify the output.
3 - Speak out brute force approach
we can run two forloops (n*n) and add. each numbers to check if it sums to target. if yes return.
4 - Try to reduce repetations,
in above approach we are searching for other number traversing whole array (O(n)).
So we can come up with
Sorting (O(1) extra space and O(n*log(n)) time) :
1 - sorting the array and then using binary search to find another number
2 - using two pointers method.
HashMap (O(n) extra space and O(n) time) :
1- Store each elements in hashmap and search for its complement in the map.
5 -> After discussing about the space and time complexity. start to code.
keep in mind the variable naming and code quality matters.
Also speak out loud each piece of logical code you write.

```
   vector<int> twoSum(vector<int>& nums, int target){
   map<int,int> complement ;
         for(int i = 0 ; i < nums.size(); i ++) {
      if(complement.find(nums[i]) != mp.end()) {
   return{i, mp[nums[i]]};  }
     complement[target - nums[i]] = i ;
        }
     return {} ;
      }
```

6-> apply test-cases/ edge-cases you came up with in step-2 to verify if code if working.
7-> If not, debug by printing.

Container With Most Water (Medium)

Problem Statement

You are given an integer array `height` of length `n`. There are `n` vertical lines drawn such that the two endpoints of the `ith` line are `(i, 0)` and `(i, height[i])`.

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return *the maximum amount of water a container can store*.

Notice that you may not slant the container.

Follow first two steps, from previous question.

3 - starting with the brute force, we can simply do O(n*n) approach, taking two pointers

```
for(int i = 0 ; i < height.size(); i ++ ) {
      for(int j = i+1 ; j < height.size() ; j ++ ) {
                  mx = max((min(height[i],height[j])*(j-i+1), mx);
```

4 - This will simply give you correct answer but we can clearly see the repeatitions here.

So next try to remove that. We can observe that

1. The widest container (using first and last line) is a good candidate, because of its width. Its water level is the height of the smaller one of first and last line.
2. All other containers are less wide and thus would need a higher water level in order to hold more water.
3. The smaller one of first and last line doesn't support a higher water level and can thus be safely removed from further consideration.

5 - With these we can come up with O(n) solution, code it.

```
int maxArea(vector<int>& height) {
        int n = height.size();
        int l = 0 ; int r = n -1;
        int mx = 0 ;
        while(l < r ) {
             int h = min(height[l], height[r]);
             mx  = max((r-l)*h,mx);
             if(height[l] <=  h) l ++;
             if(height[r] <= h) r -- ;
        }

        return mx ;
    }
```

Test the code with test case, and verify.

Further Explaination

Variables l and r
define the container under consideration. We initialize them to first and last line, meaning the widest container. Variable mx
will keep track of the highest amount of water we managed so far. We compute r - l
, the width of the current container, and `min(height[l], height[r])`
, the water level that this container can support. Multiply them to get how much water this container can hold, and update mx
accordingly. Next remove the smaller one of the two lines from consideration, as justified above in "Idea / Proof". Continue until there is nothing left to consider, then return the result.

Median of two Sorted Arrays (Hard)

Problem Statement

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return **the median** of the two sorted arrays.

The problems statement is very clear, even then try to repeat your understanding of the problem

→ Start with brute force, its pretty clear to see a O((m+n)log(m+n)) solution. Just merge both arrays and pick
the middle element.
That will take O(n+m) space and O((m+n)log(m+n)) time.
→ try to improve on it.
We can take two pointers i, j . i for nums1 and j for nums2 and keep on incrementing until i + j reaches (m+n)/2

```
    while(j+k<=((n+m)/2))
    {
        if(k>=m || (j<n && nums1[j]<nums2[k]))
        {
            y=x;
            x=nums1[j++];
        }
        else
        {
            y=x;
            x=nums2[k++];
        }
    }
    if((n+m)%2==0)
        return (double)(x+y)/2;
    return (double)x;
  }
```

Now here we are using O(1) space and O(n+m) time.
→ Can we improve it further ?
Yes
Both arrays are sorted, and for sorted arrays one straight hint is binary Search.

time complexity = O(log(min(n1, n2)))space complexity = O(1)

```
double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        int n1 = nums1.size();
        int n2 = nums2.size();
        // make sure first array is of smaller length i.e n1 < n2
        if(n1 > n2) return findMedianSortedArrays(nums2, nums1);
        int low = 0;
        int high = n1;
        while(low <= high)
        {
            int mid1 = (low + high) / 2;
            // works for both odd and even lengths
            int mid2 = (n1 + n2 + 1) / 2 - mid1;
            // we create two search spaces for applying binary search. left half and right half have elements
            // from both the arrays. left1 & right1 have elements from nums1 and  left2 & right2 have elements from nums2
            // left half should be smaller than right half
            // l1 is max of left1 list, l2 is max of left2 list
            // r1 is min of right1 list, r2 is min of right2 list
            int l1 = (mid1 - 1 < 0) ? INT_MIN : nums1[mid1 - 1];
            int l2 = (mid2 - 1 < 0) ? INT_MIN : nums2[mid2 - 1];
```

```
            int r1 = (mid1 == n1) ? INT_MAX : nums1[mid1];
            int r2 = (mid2 == n2) ? INT_MAX : nums2[mid2];
            // correct partioning
            if(l1 <= r2 && l2 <= r1)
            {
                // even length -> two medians -> return average of both
                if((n1 + n2) % 2 == 0) return (max(l1, l2) + min(r1, r2)) / 2.0;
                // odd length -> one median -> return it
                else return max(l1, l2);
            }
            if(l1 > r2)
            {
                high = mid1 - 1;
            }
            if(l2 > r1)
            {
                low = mid1 + 1;
            }
        }
        return 0.0;
    }
```

Go through the comments in the code for better understanding.