**Coding Competency: Problem Solving**

**What is Problem Solving?**

An experienced interviewer will take on this question. This question will verify if you can deal with ambiguity and how you can think on your feet. Do you know how to adjust yourself when the interviewer requests to change the requirements. An interviewer will give you a very open-ended question (it’s not a tricky question). It’s an open-ended question with multiple and different solutions. The interviewer will add constraints, loosen the constraints, or just see how you respond when the constraints are changed. The interviewer wants you to take a **step back** to rethink the problem again.

Prior to coding, we see bar raising candidates that spend 15 minutes with the interviewer having a discussion before they go to the code in this round. Once you asked clarifying questions, removed the ambiguity, you can now start coding which should be completed within 15 minutes. In total, this coding session should be completed in 30 minutes.

Problem solving questions should assess three elements:

1) The ability for the candidate to solve a real world coding problem in multiple different ways,

2) Converting the solution(s) into working code

3) The knowledge of the candidate about time, space complexities while comparing the solutions.

**How to Pass Problem Solving Coding**

\*In the chart below, there is no absolute scoring on the criteria and they are equivalently important. It is about making decisions based on trade-offs and risks seen in the interview (and obviously the more mild strength/full strength data points the better).

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Mild Strength** | **Strength** |
| **Ability to translate the solution into code** | Good quality of solution – clean and logical code. Good / appropriate data structures and readable code. Good organization of code (class, method, etc.) | Methodical organized solution. Candidate will start to think about extensibility of the solution and can talk through it or incorporate it into their solution. |
| **Ability to identify different solutions and evaluate them** | Meets. Ability to identify the different solutions and differentiate them. Candidate can identify the best solution but does not necessarily identify all of the possible solutions | Raises. Ability to differentiate and identify different solutions in detail and compare the trade-offs and make different approaches to the problem and evaluate each. |
| **Ability to break down the problem and create it into different workable smaller components to solve** | Candidate can still break down the problem but still has to lean on the interviewer to drive the interview. Candidate can solve the problem but still requires hints. There is more of a collaboration in discussion and can expand on the hints given by the interviewer. | Candidate can systematically solve multiple layers of the problem and come up with the best solution. Candidate was able to walk through their problem solving method consistently throughout. |
| **Dealing with ambiguity** | Can identify requirements and then execute on there. May ask clarifying questions throughout and can evaluate pros and cons. | Candidate was able to ask multiple follow up questions, can handle the interviewer changing the requirements. |

**What makes a Coding: Problem Solving question different from other types of coding questions?**

1. **Multi-layer problem** - Problem should have multiple layers/level of complexity associated with it. This allows us to probe candidate’s ability to adapt their solution to accommodate different requirements.
   1. Can you break down the problem logically?
   2. Can you identify the different solutions to solve the problem and discuss the trade-offs for each solution? Can you identify the optimal solution for the problem?
   3. Can you solve the problem and translate the solution into code?
2. **Problem with multiple solutions** - There should be multiple ways to solve the problem. Candidate should be able to discuss trade-offs and characteristics of different solution options. Candidate should be able to identify optimal solution.
3. **Question statement with ambiguity** - There should be ambiguity in the question itself. This allows us to probe candidate’s ability to deal with ambiguity and gather requirements necessary to solve the problem.
4. **Equal focus on ability to solve problem and translate into code**- Focus on candidate’s ability to collect necessary requirements, break down problem and solve it in logical way. The ability of translating the solution into code is necessary.
5. **Interactive process** - Candidate should be communicating the assumptions while discussing the problem, but while communication is recommended, it is not required while solving the problem after the problem is well understood.
6. **Awareness of Conscious/Unconscious bias**.
7. **Relevance to engineering problems** - The question should be relevant to problems you would typically solve as an SDE.

**Fair/Unfair Question Traits**

Fair

1. **CS Fundamentals** - Knowledge of CS fundamentals will be required to write the solutions' code, but it should not be the cornerstone of your interview. We recommend to keep it simple (Sets, Maps, branching, ...).
2. **Contains ambiguity**
3. **Requiring specialized knowledge** - If you require any specialized knowledge, the interviewer must be able to walk them through it first.
4. **Amazon related** - Questions branded to Amazon, can be business focused.

Unfair

1. Requires memorization to solve majority of the problem.
2. Brain teasers, and/or riddles.
3. There can NOT only be one way to solve the problem.
   1. Ex: Using Amazon design checkout as an example - this could cause the candidate to think there is already an ideal way to solve this question (the way checkout currently is).
4. Cannot solve any sort of culturally-specific aspects. Should be universal only.
   1. Games can be tricky - games are often confused and different around the world. Same as sports.
   2. Avoiding things related to holidays, religious aspects
5. Avoiding niche areas.

**Sample Coding: Problem Solving questions**

***Question 1:******Product Credits***

Let’s say that Amazon surprises customers on a periodic basis where it gives each customer some credit (X amount) that can be used to buy 2 products from a curated list of products. The credit has to be used completely in one transaction and you can only buy 2 products. Come up with a solution to identify which two products would exactly match the credit.

**NOTES for interviewer:**

* Candidate just needs to know what a transaction is and what product is - even knowing what amazon does/interacts is not needed to complete
* There are multiple ways to solve this, you could do brute force, you could do backpacking (a little dynamic programming), maybe even searching algorithm

**Follow up questions as the interviewer for L4 SDE data points:**

* Just stick with the initial stated question and attempt FUQ (Q2):
* Which two products would maximize the credit utilization (does not need to exactly match)

**Follow up questions as the interviewer for L5 SDE data points:**

* FUQ (Q1): Which two products would maximize the credit utilization (does not need to exactly match) - be able to walk through this and write some code as a bar raising SDE II
* FUQ (Q2): Generalize it for any number to buying n products (rarely there is time to discuss this or reach this point). - be able to at least explain it, if time write some code

**Follow up questions as the interviewer for L6 SDE data points:**

* SDE III data points could include FUQ (Q2):
* Generalize it for any number to buying n products (rarely there is time to discuss this or reach this point). - be able to at least explain it, if time write some code, talk through long term solutions, approach, scaling, etc.

***Question 2:******Radio Tower Signals***

Given a 2D grid having radio towers in some cells, mountains in some and the rest are empty. A radio tower signal can travel in 4 different directions recursively until it hits a mountain in that direction. What all radio towers can hear each other?

**NOTES for interviewer:**

* Mimics real life Amazon device problems

**Follow up questions as the interviewer for L4 SDE data points:**

* SDE I would talk through FUQ above, not implement. If they just solve the original question it would be mild strength and talking through the follow up question if full strength

**Follow up questions as the interviewer for L5 SDE data points:**

* FUQ: What if the tower signal intensity reduces by 1 point for every step the signal propagates and I provide that as an input to your function - they should make it through this FUQ as its not a huge refactoring twist. Decently maintainable code should be edited to accommodate this.

**Follow up questions as the interviewer for L6 SDE data points:**

* Add concept of a signal repeater, if a radio tower hits one of these, the signal degradation resets and continue on. - this may too hard to solve, get second opinion

***Question 3:******Currency Converting***

Given a file of currency conversion rates, write a function that converts one currency to another.

Sample log file entries:

{"from":"USD", "to":"EUR", "rate":1.1}

{"from":"EUR", "to":"GBP", "rate":1.2}

Means USD is worth 1.32 GBP

**NOTES for interviewer:**

* Requires the candidate to ask clarifying questions on the intention.. Sometimes they will miss the fact that you have to convert to intermediate currencies. If they miss this, direct them as an interviewer. Clarify the question is not to have the candidate find the BEST conversion, but to find A conversion.
* No domain knowledge needed, we could re-name the currencies to ABC and it would still have the same details to solve it. currency exchange is a universal topic

**Follow up questions as the interviewer for L4 SDE data points:**

* An SDE I level candidate will spend most of the time on the problem statement, and can talk through verbally the edge cases and explain them. Code to handle these edge cases not needed; code to solve the problem is still expected.

**Follow up questions as the interviewer for L5 SDE data points:**

* Get through problem and then identify 2 edge cases 1) what if there is not a conversion rate that exists and then 2) they should recognize that each currency conversion rate in the file is given 2 pieces of information. This is the cycle problem, and code for this.

**Follow up questions as the interviewer for L6 SDE data points:**

* What ideas could they have to further optimize this solution? Talking verbally through it is acceptable. Looking for ideas around scale, optimization, etc.

***Question 4:******Strings – Decreasing Costs***

Write a function to take a string and decrease all costs in the string by 15%. Example string would be "Mary spent $5.25 on books this week". Currencies can be of the form $5.25, $ 5.25, and USD 5.25.

**NOTES for interviewer:**

* Examples of using the standard regex library of each language and some common string manipulation functions in each language (split, merge, etc)
* US currency is universal, it has been a problem with anyone not understanding it, but they can ask follow up questions to understand the nuances around the currency. What may trip people up more than the currency is that different places use decimals, commas, etc. we would expect an SDE II to clarify this, and if not, then it is a follow up question outlined below.
* Questions usually seen from candidates to clarify assumptions are "what is the size of the price" "can I expect it to me always a number, decimal, two zeros", "where is price located in the string" (interviewer to say anywhere, you have to find it), ' can there be multiple prices? interviewer to say yes"

**Follow up questions as the interviewer for L4 SDE data points:**

* No follow up questions will be needed, base case solution as presented that is optimal, handling all initial edge cases. May need some hinting which would be OK

**Follow up questions as the interviewer for L5 SDE data points:**

* Follow up questions for the interviewer to ask: 1) Our solution must work for these currency formats but let us keep in mind that we want to extend the solution to work for international currencies as well. For e.g. EUR, GBP etc. (getting through this is raising the bar) 2) Some currency's use commas instead of periods, how would you handle different formats? 3) Some currencies have markers (currency symbol like dollar sign, E for EU), that come before and after, how would you check to identify this? 4) How do I parameterize the discount? What if I do not want it to always be 15%?

**Follow up questions as the interviewer for L6 SDE data points:**

* Expect an SDE III to get through all above questions above - at the very least verbally but can be done in code as many of the coding changes needed should have been caught earlier and would be easier to write out.