A2SV
Africa To Silicon Valley

# The 7 Steps of Highly Effective Problem Solving

A2SV
Africa To Silicon Valley

If we want to perform well in the interviews with top tech companies, we need to prepare and play with the best tactics.

A2SV
Africa To Silicon Valley

## 1. Understand the Problem

If you understand something different than what is asked, the odds are extremely low that you land on a correct solution.

Each sentence of the text reveals something about the problem. Sometimes it may not be clear after the first round of reading.

In such times, check the input/output and its explanation.

Don't try to go too fast. Don't try to cut to the chase immediately.

**If something is written there, there must be a reason for that.**

## 2. Solve the problem manually

**‘Nothing can be automated that cannot be done manually!’**

Solving a problem programmatically means telling the computer how to solve it via a language it can understand.

Computers can do things at scale, a lot faster than people can.

If you don't know how to solve the problem for small input, how can you tell a computer to solve it at scale?

Sometimes, solving it by hand reveals some patterns that are key to the programmatic solution.

Observe yourself, and understand how you do it by hand to automize.

## 3. Come up with a solution idea on the paper

What comes after solving it by hand?

The big picture and patterns are **not always visible** at first glance.

At this step, it is important to check your algorithm's **time and space complexity**, including the helper functions you use

## Rookie mistake

Skipping time and space complexity checks.

Finally, flesh out the **details** of your algorithm.

A2SV
Africa To Silicon Valley

## Bruteforce & Optimization

Despite being possibly slow, a brute force algorithm is valuable. It's a starting point for optimizations, and it helps you wrap your head around the problem.

A2SV
Africa To Silicon Valley

## Techniques for Optimization

Look for BUD (Bottlenecks, Unnecessary Work, Duplicated Work):

- Identify bottlenecks in the algorithm that slow down overall runtime.
- Look for unnecessary work and eliminate it where possible.
- Consider duplicated work and find ways to reduce redundant computations.

## 4. Play devil's advocate against your idea before falling in love with it

We love to love our ideas.

At the same time, it is better to fall in love with them after getting to know them well.

Therefore, before falling in love, play devil's advocate against them.

A2SV
Africa To Silicon Valley

Imagine the person that you like the **least**( :) ) proposes this idea, and you want to **refute** it so badly.

Ask yourself, **how can I break this idea?** Can you come up with an input that breaks it? Try to target it logically, what can be an edge case for it?

## TIP

A tip that can be useful for coming up with edge cases is going over your proposed solution and looking for assumptions you might have made.

**Time is gold, especially in the interviews.**

A2SV
Africa To Silicon Valley

## TIP

A tip that can be useful for coming up with edge cases is going over your proposed solution and looking for assumptions you might have made.

**Time is gold, especially in the interviews.**

Africa To Silicon Valley

## 5. The implementation, that is our mission.

Pseudocode is a high-level representation of a solution that outlines the steps and logic without adhering to any specific programming language syntax.

pseudocode helps in writing code faster and reduces the chances of introducing errors during implementation.

## Example: Minimum Size Subarray Sum

Given an array of positive integers nums and a positive integer target, the goal is to find the minimal length of a subarray whose sum is greater than or equal to target.

## Pseudocode solution

```
○ for every idx have a total sum
    ■ while that total sum is less than target
        ● include neighbours
    ■ if total greater or equal have the length
        ● total - current
```

Do not try to explain everything to the interviewer. At the same time, inform them about what is going on after completing each block of code.

Divide your code into blocks and using helper functions whenever you can.

A2SV
Africa To Silicon Valley

1. CHOOSE THE BEST LANGUAGE FIT
2. VARIABLE NAMING
3. USE HELPER FUNCTIONS
4. EXPLAIN CODE BLOCKS AFTER FINISHING

```python
def minSubArrayLen(self, target: int, nums: List[int]) -> int:
    total = 0
    minim = len(nums) + 1
    right = 0

    for idx, curr in enumerate(nums):
        while right < len(nums) and total < target:
            total += nums[right]
            right += 1

        if total >= target:
            minim = min(minim, right - idx)

        total -= curr

    return minim if minim != len(nums) + 1 else 0
```

A2SV
Africa To Silicon Valley

## 6. Comprehensive test cases

Africa To Silicon Valley

After implementing the solution, proceed to test with interesting cases, including edge ones, to make sure the solution works for different scenarios.

Testing is not just coming up with **obvious** edge cases like null nodes, empty arrays, or zero values. It is also not trying some random cases.

A2SV
Africa To Silicon Valley

## Each test case you try should have a motivation.

At this stage, do not just think about your algorithm. Imagine the code is written by a 3rd party. Your job is to make sure your tests save the codebase from merging a buggy or inefficient code.

## Testing for Assumptions

When developing code or solutions, it is common to make assumptions about certain conditions or expectations. By intentionally testing these assumptions, we can ensure the accuracy and reliability of our codebase.

Let's consider an example of assumption testing. Suppose we have implemented the prefix sum algorithm, assuming that all input numbers are positive.

To prevent merging a potentially flawed codebase, we must design test cases that verify this assumption. One test case would involve providing an array with negative numbers as input.

## 7. Simplify and clean up your code

Now that we have a working solution and we are confident in its accuracy and efficiency let's do some **cleanup** and **simplifications**.

Consider better variable naming, adding some comments, and possibly modularizing further. This step can be optional, depending on how much time you have left.

A2SV
Africa To Silicon Valley

## Resources

★ Gayle Laakmann McDowell. (2015). Cracking the Coding Interview, 6th Edition. CareerCup, LLC. Page Numbers: [60-81].

## Quote of the day

"Practice doesn't make perfect, perfect practice makes perfect"

*– Vince Lombardi*