While folks are joining

Get you laptops ready and login to www.crio.do.
We will be coding away in the session!

Crio Foundation Series: DSA

Session 9

What’s for this session?

- Two Pointer Pattern
- Sliding Window Pattern
- Solve problems
  - Find pair with given sum in sorted array
  - Merge two sorted arrays

Approach to problem solving

- **Milestone 1**: Understand the problem clearly
  - **Ask questions** & clarify the problem statement clearly.
  - **Take an example or two** to confirm your understanding of the input/output & extend it to test cases
- **Milestone 2:** Finalize approach & execution plan
  - Understand what type of problem you are solving, **reduce to known patterns or concepts**
  - **Brainstorm multiple approaches** to solve the problem and **pick one**
  - **Get to a point where you can explain your approach to a 10 year old**
  - Take a stab at the high level logic & **write it down - Pseudocode**
  - Try to offload processing to **functions** & keeping your main code small.
- **Milestone 3: Code** by expanding your pseudocode
  - Make sure you name the variables, functions clearly.
  - Avoid constants in your code, go for generic functions, you can use examples for your thinking.
  - Use **libraries** as much as possible
- **Milestone 4:** Prove to the interviewer that your code works with unit tests
  - Make sure you check boundary conditions
- Time & storage complexity
- Suggest optimizations

Two Pointer Pattern

- Is an optimization to solve certain array or linked list traversal problems.
- Used when the solution needs multiple traversals to find or organize data, according to specified constraints.
- Reduces the number of nested loops needed, **reducing Time Complexity**.
- The idea is to position **2 pointers which can traverse** at the same or different speed and in the same or opposite directions to solve the problem.
- Let's understand this with examples.

Takeaways

- **When to use this method?**
  - When the problem involves arrays or linked lists and the goal is to find or organize data according to some criteria
  - When there is a need to have better time complexity than bruteforce traversal since this method reduces the need for nested traversals

- **How to use this method?**
  - Start with two pointers positioned as necessary to solve the problem and start traversing in the direction needed
  - First pointer at the first position (or middle)
  - Second pointer at the last position (or second or middle or k-th or first itself)

- **Additional takeaways**
  - If array is unsorted, sort it first if needed. The method will be faster in spite of the additional complexity sorting brings in
  - The pointers can traverse in different directions (e.g. Check for palindrome where pointers move towards each other from either end).
  - The pointers can also move at different speeds (e.g. Linked List cycle - which we'll cover under linked lists)
  - These pointers can point to 2 separate arrays or lists (e.g. Merge 2 sorted arrays)

Frequently asked problems

- Reverse a string in place
- Move all 0s in an integer array to one end maintaining order of other elements
- In an array of integers, find 2 numbers that add up to a given value
- In an array of integers, find 3 numbers that add up to a given value (or add up to 0)
- Merge 2 sorted arrays
- Confirm if a given string is a palindrome
- Find (or Remove) duplicates from sorted array
- Trapping rain water
- Container with most water
- Sort 3 Colors in constant space or Dutch National Flag problem

Example Problems

- Two Sum
  - Sum to 0 or Sum to a given number
  - Sorted or Unsorted Array
- Three Sum
  - Builds on top of Two Sum

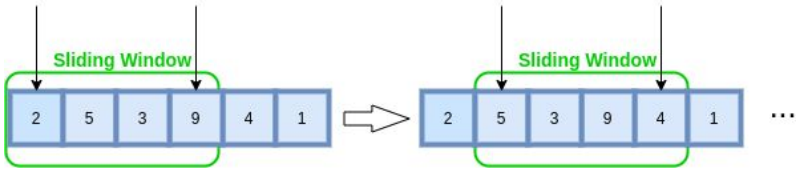
Activity 1 - Find pair with given sum in sorted array

5 minute break

Activity 2 - Merge Two Sorted Arrays

Sliding Window Pattern

- A Sliding Window is used to solve problems where we need to operate on a **contiguous subarray (or sublist)** of a larger array (or linked list).
- It is applicable in cases where we need to
    - Find **longest or shortest subarray/substring** meeting a particular criteria (E.g. Smallest substring with X unique characters or Longest stretch of days when stock price did not decrease). These are variable size windows.
    - Find a **window of fixed size** with contents meeting some criteria (E.g. Subarray of size N with largest sum)
- It **involves 2 pointers**. One indicating the beginning of the window and the second indicating the end.
- The idea is to **create a window containing a subarray and slide the window** along as we traverse through the array to find the answer.
- The window size can change by **adding new elements in the end** or **removing elements from the beginning** as we traverse.
- Let's understand this with some examples.



Takeaways

- **When to use this method?**
  - When we see a contiguous subarray problem
  - When goal is to find the longest, shortest or fixed size window in the given array, string or linked list
  - When we need to reduce the Time Complexity by avoiding recalculation of values which are already in the window as we move through the array (e.g. Calculation of average of X consecutive values in an array)

- **How to use this method?**
  - Start with a window of size 1 or a predefined size K (e.g. Subarray of size K with the maximum sum)
  - Add elements at the end and/or remove elements at the beginning of the window as we traverse along the array, string or linked list

Other frequently asked problems

- Longest substring without repeating characters
- Maximum sum subarray of fixed size K
- Given a string and a pattern, find the smallest substring which has all the characters of the pattern
- Smallest (or largest) subarray that adds up to a given sum (or zero)
- Given a string and a pattern, find out if the string contains any permutation of the pattern
    - (slide the window along looking for the permutations of the pattern in the window)
- Given a string and a set of words of the same length, find all substrings that are a concatenation of all the words exactly once
    - (similar to permutation above, but with words instead of letters)

Questions?

Take home exercises

- Remove duplicates such that each element occurs at most twice
- Find pair with given sum in an unsorted array

To be solved before the next session.

Feedback

Thank you for joining in today.

We’d love to hear your thoughts and feedback.

Thank you