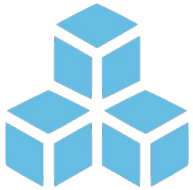
Advanced Problem Solving: Arrays and Objects

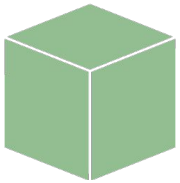
Relevel
by Unacademy

What does it take?

As we would be concentrating on hands-on advanced problem-solving in this session, please revise the concepts covered in the last 2 sessions:



Arrays



Objects

Basic Problem Solving

Classes

List of Problems Involved

- Anagram
- Print frequency of element in the string
- First non-repeating character
- Subarray with sum 0
- Subarray with sum x
- Longest consecutive sequence
- Shift Negative elements to the end of an array
- Cyclically Rotate Array by 1

Anagram

Anagram – An anagram of a string is a string that contains the same characters but the order can be different.

Example – unacademy , acadmyenu are an anagram of each other

Problem – Given an array of strings. You need to print all anagrams together. For example –

Input – {“data”, “atad”, “number”, “tada”, “adat”, “bernum”}

Output – {“data”, “atad”, “tada”, “adat”, “number”, “bernum”}

Approach 1

1) Create 2 arrays – index array and word array. Populate word array with input array and index array with respective index.

   index[]–> 0 1 2 3 4 5

   word[] -> “data” “atad” “number” “tada” “adat” “bernum”

2) Now, sort each string in a word array

   index[]–> 0 1 2 3 4 5

   word[] -> “aadt” “aadt” “bemnru” “aadt” “aadt” “bemnru”

Approach 1

3) Now, sort word array and move respective index as well in index array

index[]–> 0 1 3 4 2 5

word[] -> “aadt” “aadt” “aadt” “aadt” “bemnru” “bemnru”

4) Print final output array using index array

{“data”, “atad”, “tada”, “adat”, “number”, “bernum”}

Code Link -> https://jsfiddle.net/cpsjxfuk/

Approach 2

We will sort each string individually and then put a sorted string as key and all the words having sorted value similar to key as a value.

Input array -> {“data”, “atad”, “number”, “tada”, “adat”, “bernum”}

Map –> Key Value

“aadt” {“data”, “atad”, “tada”, “adat”}

“bemnru” {“number”, “bernum”}

Code Link -> https://jsfiddle.net/tj10Lo2b/1/

Problem 2

Given two strings. You need to find if they are anagrams of each other. For example –

Input – “silent”, “listen”

Output – Two strings are anagrams of each other

Approach – We can use one count array. Our major step will be to increment the value for characters in str1 and decrement the value for characters in str2. If all the count values in the count array will be 0, it means that the two strings are anagrams of each other.

Code Link -> https://jsfiddle.net/brL1698h/

Time Complexity – O(N)

Space Complexity – O(N)

Print frequency of element in the string

Frequency – The frequency of an element in the string is defined as the number of times that element is present in the string.

Problem – Given a string and a character as input. You need to print the frequency of the character in a given string.

For example –

Input – str = “unacademy”, c = ‘a’

Output – 2

Print frequency of element in the string

Approach 1 – We can traverse string characters one by one and check if a given character is equal to that character. If it is matching, we can increase our count.

Code Link - https://jsfiddle.net/x5r4wdgt/

Approach 2 – We can also use recursion to find the frequency

Code Link - https://jsfiddle.net/crk8217u/

Time Complexity –

If there are N characters in string str, then complexity will be O(N)

Space Complexity –

If there are N numbers in given array arr, then complexity will be O(1)

First non-repeating character

Non-repeating – A character is known as a non-repeating character when its frequency is unit in a given string.

Problem – Given a string. You need to print the first non-repeating character. For example –

Input – str = “unacademy”

Output – u

First non-repeating character

Approach – We can use HashMap and frequency concept. We can insert characters as keys in the map and their respective frequencies as values.

Steps –

1) Create a map having a character as key and frequency as value
2) Iterate the given string and check whether a given character has a unit frequency.
3) If yes, print given character as output

Code Link -> https://jsfiddle.net/9x52zmyL/

Time Complexity –

If there are N characters in string str, then complexity will be O(N)

Space Complexity –

If there are N characters in string str, then complexity will be O(N)

Subarray with sum 0

Subarray with sum 0 – A subarray in a given array having a sum equal to 0.

Problem – Given an array having positive and negative numbers. You need to find a subarray having a sum of 0. For example –

Input – {0, 2, 4, -1, -3, 4, -5, 5}

Output –

Index 3 to 5

Index 6 to 7

Subarray with sum 0

Approach - We can use hashing concept. Intuition: If we take prefix sum of an array, then every index of a prefix sum which has a value as 0, denotes the ending of a subarray starting from index 0 with sum 0

Also in the prefix sum array every index j that has a repeated value with index i denotes a subarray starting from i+1 to j with sum zero, as the prefix sum remains unchanged at j.

Steps –

1) Iterate through the array
2) Track the sum of elements in a variable
3) If sum = 0, there is a subarray from index 0 to the current index
4) Verify if sum exists in the hash table
5) If sum exists, it means that sum was part of subarray (index 0 to index i) and now the same is for subarray (index 0 to index j) and hence subarray (index i+1 to index j) must be 0
6) Add the current sum to the hash table

Code Link -> https://jsfiddle.net/0e3n1Lgb/

Subarray with sum 0

Time Complexity –

If there are N characters in string str, then complexity will be O(N)

Space Complexity –

If there are N characters in string str, then complexity will be O(N)

Subarray with sum x

Subarray with sum 0 – A subarray in a given array having a sum equal to x.

Problem – Given an array having positive and negative numbers. You need to find a number of subarrays having sum as x. For example –

Input – {2, 4, -1, -2, 4, -5, 5}, sum = 6

Output –

Index 0 to 1

Subarray with sum x

Approach - We can use hashing concept. Intuition - In our previous problem, the sum of the subarray elements was zero. In this case, the only difference is that the sum of the subarray elements should be x which can be any number.

Steps –

1) Create a HashMap that contains key= prefix sum (sum of elements up to that index) and value = index. Also, create a variable to track the current sum and sum of subarray(s)
2) Iterate through the array
3) Track the sum of elements in a variable
4) If sum = s, there is a subarray from index 0 to the current index
5) Verify if key equal to sum-s exists in the hash table, if exists, then there is subarray from (index sum-s to i)
6) Add current sum and index to HashMap

Code Link -> https://jsfiddle.net/us5k1rmn/

Subarray with sum x

Time Complexity –

If there are N characters in string str, then complexity will be O(N)

Space Complexity –

If there are N characters in string str, then complexity will be O(N)

Longest Consecutive Sequence

Longest consecutive sequence – A sequence having elements as consecutive integers. Consecutive integers can be in any order.

Problem – Given an array. You need to find the length of the longest consecutive sequence such that elements are consecutive integers. For example –

Input – {35, 3, 4, 88, 9, 10, 21, 5, 6}

Output – 4

Sequence is 3,4,5,6

Longest Consecutive Sequence

Naive Approach - We can sort the array and then find the longest consecutive subarray.

Steps –

1) Sort the array
2) Set variables count and max to 0
3) Iterate through the array
4) If the current element is not equal to (previous element + 1), set count to 1 else increment the count
5) Update max to a maximum of count and max.

Code Link -> https://jsfiddle.net/hp5cgrLb/

Time Complexity –

If there are N characters in string str, then complexity will be O(NLogN)

Space Complexity –

If there are N characters in string str, then complexity will be O(1)

Longest Consecutive Sequence

Efficient Approach - We can use hashing concept.

Steps –

1) Create a HashSet and insert all elements into HashSet

2) Iterate through the HashSet

3) Verify if arr[i] – 1 is present in the hash. If not found then this is the first element of the sequence

4) If the current element is the first element of the sequence, iterate from arr[i] + 1 till the last element and count the number of elements.

5) If count > previous count – update count

6) Add current sum and index to HashMap

Code Link -> https://jsfiddle.net/01zg89uy/

Longest Consecutive Sequence

Time Complexity –

If there are N characters in string str, then complexity will be O(N)

Space Complexity –

If there are N characters in string str, then complexity will be O(N)

Shift Negative elements to the end of an array

Problem Given an array. You need to shift negative elements to the end of an array. For example –

Input – { -5, 3, -4, 88, -9, -10, 21, 5, 6}

Output – { 3, 88, 21, 5, 6, -5, -4, -9, -10 }

Shift Negative elements to the end of an array

Approach – We can use temp array to store the values. First store all positive numbers to the temp array and then negative numbers. Then copy temp array to original array
Intuition - Our main task is to rearrange the numbers in the given array. We can use conditions on elements to check if they are positive or negative and then apply operations.
Code Link - https://jsfiddle.net/8kLfqtw6/

Time Complexity –
If there are N numbers in given array arr, then complexity will be O(N)

Space Complexity –
If there are N numbers in given array arr, then complexity will be O(N)

Shift Negative elements to the end of an array

Two-Pointer Approach – This problem can be solved by a two-pointer approach in linear time complexity and with constant space.

Steps –

1) Create 2 variables - left and right which will hold 0 and n-1 indexes of the given array
2) If the left pointer and right pointer are negative, then decrement the right pointer
3) If the left pointer is negative and the right pointer is positive, then swap the elements.
4) Else if both the pointers are positive, then increment the left pointer.
5) Repeat above steps till left <= right

Code Link - https://jsfiddle.net/427vskpc/

Time Complexity –

If there are N numbers in given array arr, then complexity will be O(N)

Space Complexity –

If there are N numbers in given array arr, then complexity will be O(1))

Cyclically Rotate Array by 1

Link for reference - https://practice.geeksforgeeks.org/problems/cyclically-rotate-an-array-by-one2614/1

Cycle Rotation – Cycle rotation is the rotation in which one rotation moves the last element of an array to the first place and shifts the remaining elements to the right.

Problem – Given an array. You need to shift all elements to the right by 1 and finally cyclically rotate whole array. For example –

Input – { 3, 88, 21, 5, 6}

Output – { 6,3, 88, 21, 5}

Cyclically Rotate Array by 1

Approach – We can use two pointers like i and j which will point to the first and last elements of an array. We need to swap i and j till i is not equal to j.

Intuition - Here, we need to keep j fixed and move i to the right direction.
Code Link - https://jsfiddle.net/jkpdL1h3/

Time Complexity –
If there are N numbers in given array arr, then complexity will be O(N)

Space Complexity –
If there are N numbers in given array arr, then complexity will be O(1)

Practice Questions

1) Find the first element having a maximum frequency in a string “unacademy”
2) Write a program to rearrange the elements such like positive numbers shift to the end and negative numbers to the right

Upcoming Class Teaser

- Intro to Recursion
- Base Cases
- Call Stack
- Type of recursion
- Fibonacci
- Factorial
- Problem Solving
- Print increasing Recursively
- Print Decreasing Recursively
- Check if the array is sorted Recursively
- Friends Pairing
- Number of binaries with no consecutive ones

Thank You!