Basic Problem Solving: Arrays and Objects

Relevel
by Unacademy

Pre-Requisites

- JS Arrays and Objects.
- Array functions in JS

Introduction

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

- Functions
- Basic Problem Solving
- Arrays
- Objects

We will be using JSfiddle as our coding playground. Please show a small demo on the look and feel of the same (Link for your reference - https://jsfiddle.net/).
There are no prerequisite installations for its usage.

List of Concepts Involved

- Array in JS
- Sorting array with different techniques like:
  - Pointers
- Filters in array

Problem Solving

1. Sort array of 0 1
2. Sort array of 0 1 2
3. Product of Array Except Self
4. Check for Duplicates
5. Rain water trapping
6. Find element that occurs once in the array where rest of the element appear twice
7. Minimum Absolute Difference
8. Best time to buy the stock

Sort Array of 0,1

Given an array having 0,1 as input.We need to write a function that sorts the array in ascending order without using inbuilt sort function
Expected time complexity :O(n)
Expected space complexity:O(1)

Example:
Input: {0, 1, 1,0, 0, 1}
Output: {0, 0, 0, 1, 1, 1}

Input:{0,0,1,1,0,1,0}
Output:{0,0,0,0,1,1,1}

Sort Array of 0,1

Approach 1:

Brute-force algorithm using double traversal

In this approach we will be counting the numbers of 0's and 1's in the input array and store it in a variable and then traverse through the array again and using those counts, we will assign zeroes and ones in the array and finally return the array.

Sort Array of 0,1

Implementation: https://jsfiddle.net/qodmL56y/

```
var input = [1,1,1,0,0,1,1,0,0,0,1,0,1,0,1];
var size = input.length;
// function to sort an array containing 0,1, and return the sorted ar
ray
function sortZeroOne(input, size) {
  let countZero = 0; // to count number of zeroes
  let countOne = 0; // to count number of ones
  let i = 0;// looping thorugh array
  while (i < size) {
    if(input[i] == 0)
      countZero++;
    if(input[i] == 1)
      countOne++;
    i++;
  }
  i = 0;
  while (i < countZero) {
    input[i] = 0;
    i++;
  }
  while (i < countOne + countZero) {
    input[i] = 1;
    i++;
  }
  return input;
}
var output = sortZeroOne(input, size);
console.log(output);
```

Sort Array of 0,1

Approach 2:

We divide the array into three parts .Part one containing the zeros,second part containing the values which can be zero or one and the last part containing one.If the element in second part is zero will reduce the size of second part,if the element is one will move it to the third part and reduce the size of second part.

Sort Array of 0,1

Algorithm

1)We will have two indices mid=0,end=N and there are three parts

(0,mid):the elements here will be 0

(mid,end):the elements here can be 0,1

(end,N):the elements here will be 1

Where N = size of the array

2)We will be traversing the array from start to end until mid is less than end

3)If the mid element is 0,increment mid by one

4)If the mid element is 1,will swap with the element at index end and decrement the value of end by one

6)We will continue doing this till the mid value is less than the end.

Sort Array of 0,1

Implementation:

Code: https://jsfiddle.net/041zoLwu/1/

```
var input = [1,1,1,0,0,1,1,0,0,0,1,0,1,0,1];
var size = input.length;
// function to sort an array containing 0,1, and return the sorted arr
ay
function sortZeroOne(input, size) {
  let mid = 0;
  let end = size-1;
  let swap = 0; //variable for swapping
  while (mid <= end) {
    if (input[mid] == 0) {
      mid++;
    } else {
      swap = input[end];
      input[end] = input[mid];
      input[mid] = swap;
      end--;
    }
  }
  return input;
}

var output = sortZeroOne(input, size);
console.log(output);
```

Sort Array of 0,1

Complexity Analysis:

Time Complexity :O(n)

Only one iteration of the array is made

Space Complexity:O(1)

No extra space is required

Sort Array of 0,1,2

Given an array having 0,1,2.We need to write a function that sorts the array in ascending order without using inbuilt sort function
Expected time complexity :O(n)
Expected space complexity:O(1)

Example:
Input: {0, 1, 2, 2, 1, 2}
Output: {0, 1, 1, 2, 2, 2}

Input:{2,2,1,0,0,1,2,2}
Output:{0,0,1,1,2,2,2,2}

Sort Array of 0,1,2

Approach 1:

Brute-force algorithm using double traversal

In this approach we will traverse through the given array count the numbers of 0's and 1's and 2's or deduce count of 2 by subtracting number of 1's and 0's from length of array and store it in a variable and then traverse through the array again and using those counts stored, we will assign same amount of zeroes, ones and twos in the array using a pointer and finally return the array.

Sort Array of 0,1,2

Implementation:

https://jsfiddle.net/hxt4qv7o/

```
var input = [1, 1, 1, 2, 0, 0, 0];
var size = input.length;
// function to sort an array containing 0,1,2 and return the sorted array
function sortZeroOneTwo(input, size) {
  let countZero = 0; // to count number of zeroes
  let countOne = 0; // to count number of ones
  let countTwo = 0; // to count number of twos
  let i = 0;// looping thorugh array
  while (i < size) {
    if(input[i] == 0)
      countZero++;
    if(input[i] == 1)
      countOne++;
    if(input[i] == 2)
      countTwo++;
    i++;
  }
  i = 0;
  while (i < countZero) {
    input[i] = 0;
    i++;
  }
  while (i < countOne + countZero) {
    input[i] = 1;
    i++;
  }
  while (i < countTwo + countOne + countZero) {
    input[i] = 2;
    i++;
  }
  return input;
}
var output = sortZeroOneTwo(input, size);
console.log(output);
```

Sort Array of 0,1,2

Approach 2:

First will divide the array into four parts .Part one containing the zeros,second part containing the ones,third part containing the values which can be zero,one or two and the last part containing two.If the element in third part is zero will move it to the first part and reduce the size of third part,if the element is one will reduce only the size of third part and if the element is two will move it to the fourth part and reduce the size of third part.

Sort Array of 0,1,2

Algorithm

1)We will have three indices start=0,mid=0,end=N and there are four parts
(0,start):the elements here will be 0
(start,mid):the elements here will be 1
(mid,end):the elements here can be 0,1,2
(end,N):the elements here will be 2
Where N = size of the array
2)We will be traversing the array from start to end until mid is less than end
3)If the mid element is 0 will swap element at start index with mid and increment both mid and start by one
4)If the element is 1, increment the mid by one.
5)If the element is 2,will swap with the element at index end and decrement the value of end by one
6)We will continue doing this till the mid value is less than the end.

Sort Array of 0,1,2

Implementation:

Code: https://jsfiddle.net/041zoLwu/

```
var input = [1, 1, 1, 2, 0, 0, 0];
var size = input.length;
// function to sort an array containing 0,1,2 and return the sorted array
function sortZeroOneTwo(input, size) {
  let start = 0;
  let mid = 0;
  let end = size - 1;
  let swap = 0; //variable for swapping
  while (mid <= end) {
    if (input[mid] == 0) {
      swap = input[start];
      input[start] = input[mid];
      input[mid] = swap;
      mid++;
      start++;
    } else if (input[mid] == 1) {
      mid++;
    } else {
      swap = input[end];
      input[end] = input[mid];
      input[mid] = swap;
      end--;
    }
  }
  return input;
}

var output = sortZeroOneTwo(input, size);
console.log(output);
```

Sort Array of 0,1,2

Complexity Analysis:

Time Complexity :O(n)

Only one iteration of the array is made

Space Complexity:O(1)

No extra space is required

Product of Array Except Self

Given an integer array nums, return an array answer such that answer[i] is equal to the product of all the elements of nums except nums[i].

The product of any prefix or suffix of nums is guaranteed to fit in a 32-bit integer.

You must write an algorithm that runs in O(n) time and without using the division operation.

Example 1:
Input: nums = [1,2,3,4]
Output: [24,12,8,6]

Example 2:
Input: nums = [-1,1,0,-3,3]
Output: [0,0,9,0,0]

Constraints:

- 2 <= nums.length <= 105
- -30 <= nums[i] <= 30
- The product of any prefix or suffix of nums is guaranteed to fit in a 32-bit integer.

Product of Array Except Self

Approach:

We loop through the input array twice where first time we will be storing the product to all the left elements of the ith element using a variable and next the next loop we will iterate in reverse order to store the product of element of the right side of the jth element of the array.

Code: https://jsfiddle.net/7uam3w8L/

```
/**
 * @param {number[]} nums
 * @return {number[]}
 */
function productExceptSelf (nums) {
    let output = nums.map(n => 1);
    let product = 1;

    // Multiply from the left
    for (let i = 0; i < nums.length; i++) {
        output[i] = output[i] * product;
        product = product * nums[i];

    }

    product = 1;

    // Multiply from the right
    for(let j = nums.length - 1; j >= 0; j--) {
        output[j] = output[j] * product;
        product = product * nums[j];
    }

    return output;
}

const arr = [1,2,3,4];
console.log(productExceptSelf(arr));
```

Contains Duplicate

Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct.

Example 1:
Input: nums = [1,2,3,1]
Output: true

Example 2:
Input: nums = [1,2,3,4]
Output: false

Example 3:
Input: nums = [1,1,1,3,3,4,3,2,4,2]
Output: true

Constraints:

- 1 <= nums.length <= 105
- -109 <= nums[i] <= 109

Contains Duplicate

Approach:

Here we are going to us an object to keep a track of visited numbers while traversing through the array, an object with key as the number and boolean value: true or false which will help us in determining the visited element by simple accessing the object value using key. If we get any existing element from the object we will return true and if the loops terminates it means it didn't had any duplicates so we will return false.

Code: https://jsfiddle.net/96ft82xm/

```
function containsDuplicate(nums) {
    const visitedNums = {}; // {7: true, 2: true}

    for(let i = 0; i < nums.length; i++) {
        const num = nums[i];

        if(visitedNums[num]) {
            return true;
        } else {
            visitedNums[num] = true;
        }
    }

    return false;
};


const arr1 = [1,2,3,4];
const arr2 = [1,1,1,3,3,4,3,2,4,2];
console.log(containsDuplicate(arr1));//returns false
console.log(containsDuplicate(arr2));//returns true
```

Rain water trapping

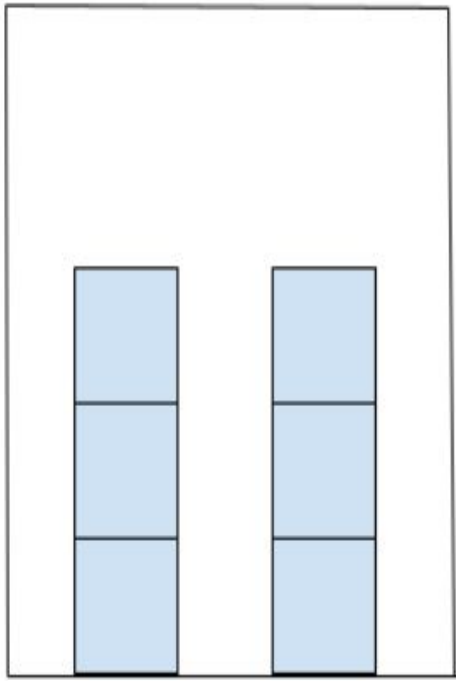
Given an array of non negative integers representing the height of each block where the width of each block is 1,we need to find out maximum rain water that can be trapped using blocks.

Example:
Input:{3,0,3}
Output:3 as three units of water can be stored between this two blocks.

Approach: There are many approaches for this problem.Will discuss each one of them one by one

Rain water trapping

Image Representation



Rain water trapping

Approach 1:

We will traverse the array from start to end and for each element will find the highest bars on left of this element and right of this element.So the amount of water it can store will be the minimum of left and right height blocks-height of the current block

Algorithm:

1)Traverse the array from beginning to end

2)For each element iterate the array from start to present element and find the highest block in left,similarly start from current element index to end and find the highest block in right

3)The amount of water that can be current stored for current block is min(left,right)-current block height

4)We will have a variable to store the sum of water for all the elements of array

Rain water trapping

Implementation

Code: https://jsfiddle.net/p85ehvyx/

```
var input = [3,0,2,0,4]

function trapWater(input){
  let ans = 0;//max water to be stored
  for(let i=1;i<input.length-1;i++){
    let left = input[i];//highest block on left
    for(let j=0;j<i;j++){
      left = Math.max(left,input[j]);
    }
    let right = input[i];//highest block on right
    for(let j=i+1;j<input.length;j++){
      right = Math.max(right,input[j]);
    }
    ans+=Math.min(left,right)-input[i];
  }
  return ans;
}

var output = trapWater(input);
console.log(output);
```

Rain water trapping

Complexity Analysis:

Time Complexity:O(N^2)

As there is nested loop

Space Complexity:O(1)

Rain water trapping

Approach 2:

Upper mentioned approach’s time complexity can be reduced if we can precompute the left and right element for every element of the array which will require extra space but time complexity will be reduced.We will create two array for storing the left and right values and use it for computation

Rain water trapping

Implementation:

Code: https://jsfiddle.net/p85ehvyx/1/

```
var input = [3,0,2,0,4]

function trapWater(input){
  let ans = 0;//max water to be stored
  let n = input.length;
  let left = new Array(n);//precompute left highest block
  let right = new Array(n);//precompute right highest block

  left[0] = input[0];
  right[n-1] = input[n-1];
  for(let i=1;i<n;i++){
    left[i] = Math.max(input[i],left[i-1]);
  }
  for(let i=n-2;i>=0;i--){
    right[i] = Math.max(input[i],right[i+1]);
  }
  for(let i=0;i<n;i++){
    ans+=Math.min(left[i],right[i])-input[i];
  }
  return ans;
}

var output = trapWater(input);
console.log(output);
```

Rain water trapping

Complexity Analysis:

Time Complexity:O(N)

As we are traversing the array constant times

Space Complexity:O(N)

As we are having left and right array for finding highest block on left and right side

Rain water trapping

Approach 3:

We can reduce the space complexity by using two variables to store the left maximum and right maximum upto that point

Algorithm:

1)Take two pointers lo and hi ,lo pointing to the start of array and hi pointing to the end of array

2)Create two variables max_left and max_right for storing max left and max right block.

3)While lo<hi

    1)if input[lo]<input[hi] update max_left,res and increment lo by one

    2)if input[lo]>=input[hi] update max_right,res and decrement hi by one

4)Return the result

Rain water trapping

Implementation:

Code: https://jsfiddle.net/p85ehvyx/2/

```
var input = [3,0,2,0,4]

function trapWater(input){
  let ans = 0;//max water to be stored
  let n = input.length;
  let max_left = 0;
  let max_right = 0;
  let lo = 0;
  let hi = n-1;
  while(lo<hi){
    if(input[lo]<input[hi]){
      if(max_left<input[lo])
        max_left = input[lo];
      ans += max_left-input[lo];
      lo++;
    }else{
      if(max_right<input[hi])
        max_right = input[hi];
      ans+=max_right-input[hi];
      hi--;
    }
  }
  return ans;
}

var output = trapWater(input);
console.log(output);
```

Rain water trapping

Complexity Analysis:

Time Complexity:O(N)

As we are traversing the array constant times

Space Complexity:O(1)

As we are using two pointers to store the left max and right max

Find element that occurs once in the array where rest of the element appear twice

Given an array where every element appears twice leaving one of the element.We need to find the element which appears once in the array

Example:
Input:{4,3,5,6,7,7,6,5,3}
Output:4

Find element that occurs once in the array where rest of the element appear twice

Approach: There are many approaches .We will discuss some of them and implement the one with least time complexity and space complexity

First approach will be two pick one element from the array and search for occurrence of that element leaving the index of the picked element and if the picked element has a single occurence then return the picked element.
Time Complexity :O(N^2)
As we will be traversing the array twice ,first for picking the element and second time for searching the same element leaving the picked element.
Space Complexity:O(1)
No extra space is required.

Find element that occurs once in the array where rest of the element appear twice

Second approach will be to create a hash table to store the array elements as key and count of the element as its value.Traverse the array again and return the element with count one
Time Complexity:O(N)
As we are traversing the array twice,once for storing in hash table and second time for checking the element with count one,so the complexity will be O(N)
Space Complexity:O(N)
As we are storing array elements in hash table so the space complexity will be O(N)

Find element that occurs once in the array where rest of the element appear twice

Best approach will be to use some properties of XOR which are as follows.

1)XOR of a number with itself returns 0

Eg: 3 XOR 3 = 0

2)XOR of a number with 0 return the number itself

Eg: 3 XOR 0 = 3

If there are elements in pairs in the array leaving one element, if we do the XOR of all the elements present in the array ,pairwise elements will return 0 and the XOR of a single occurrence element with 0 will return the single occurrence element.

Find element that occurs once in the array where rest of the element appear twice

Algorithm:

1)Traverse the array from start to end and take XOR of all the elements store it in a variable and return the value of the variable

Implementation:

Code: https://jsfiddle.net/1os7x09a/

```
var input = [3,4,5,5,4,2,2,3,1,-1,-1];

function findSingleOccurence(input){
  let ans = 0;
  //store xor of all the elements in ans
  for(let i=0;i<input.length;i++){
    ans = ans^input[i];
  }
  return ans;
}

let output = findSingleOccurence(input);
console.log(output);
```

Find element that occurs once in the array where rest of the element appear twice

Complexity Analysis:

Time Complexity:O(N)

We are traversing the array once

Space Complexity:O(1)

We are having only one variable to store the result of xor.

Minimum Absolute Difference

Given an array of integers and we need to find all the pairs in the array which have the minimum absolute difference and return the array

Example:
Input = [3,4,5,5,4,2,2,3,1,-1,-1];
Output:[[-1,-1], [2,2], [3,3], [4,4], [5,5]]

Minimum Absolute Difference

Approach:

First we need to find the minimum absolute difference and then using that we can get the element pairs.If we sort the array in ascending order minimum absolute difference can be taken by comparing the difference between the two consecutive elements.We need to traverse the sorted array and check if the absolute difference between them is equal to minimum absolute one and if its then enter this pair of elements into the result array.

Minimum Absolute Difference

Algorithm:

1)Sort the array A in ascending order and create a variable diff to store minimum absolute difference with max integer value

2)Traverse the array starting from index position one

1)If (diff>absolute(A[i]-A[i-1]) then change the value of diff to the latest consecutive difference and remove all the elements from the answer array.

2)if(diff=absolute(A[i]-A[i-1]) then insert A[i],A[i-1] in the answer array

Minimum Absolute Difference

Implementation:

Code: https://jsfiddle.net/1os7x09a/1/

```
var input = [3,4,5,5,4,2,2,3,1,-1,-1];

var minimumAbsDifference = function(arr) {
    arr.sort((a,b)=>a-b);
    let ans = [];
    let diff = Number.MAX_SAFE_INTEGER;
    for(let i=1;i<arr.length;i++){
        let abso = Math.abs(arr[i]-arr[i-1]);
        if(diff>abso){
            diff = abso;
            ans.length = 0;
        }
        if(diff==abso){
            ans.push([arr[i-1],arr[i]])
        }
    }
    return ans;
};

let output = minimumAbsDifference(input);
console.log(output);
```

Minimum Absolute Difference

Complexity Analysis:

Time complexity:

As we are sorting the array depending on the type of sort it can vary

Merge Sort:O(NLOGN)

Quick Sort:O(N^2)

Space Complexity:

Merge Sort:O(N)

Quick Sort:O(LOGN)

Best Time to Buy and Sell Stock:

You are given an array of prices where prices[i] is the price of a given stock on the ith day.

You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Best Time to Buy and Sell Stock:

Example 1:
Input: prices = [7,1,5,3,6,4]
Output: 5
Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Example 2:
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transactions are done and the max profit = 0.

Constraints:

- 1 <= prices.length <= 105
- 0 <= prices[i] <= 104

Best Time to Buy and Sell Stock:

Approach:

To determine the max profit by selling the stock bought at day one we will keep a track on max profit we can get buy selling ith element while lopping through it and compare it with the previous one stored in last iteration starting with 0 value and similarly the cheapest price will also be tracked and will return the desired answer.

Code: https://jsfiddle.net/82jsvaz4/

```
function maxProfit(prices) {
    let maxProfit = 0;
    let cheapestPrice = prices[0];

    for ( let i =0; i< prices.length; i++) {
        const price = prices[i];

        if(price < cheapestPrice) cheapestPrice = price;

        const currentProfit = price - cheapestPrice;
        maxProfit = Math.max(currentProfit, maxProfit);
    }

    return maxProfit;
};


const arr1 = [7,6,4,3,1];
const arr2 = [7,1,5,3,6,4];

console.log(maxProfit(arr1));//return 0
console.log(maxProfit(arr2));//return 5
```

MCQs

1. **What is the output of relevel [1] of the following code?**

```
var relevel = [3,,6];
```

A. Undefined
B. Error
C. Null
D. 0

MCQs

1. **What is the output of relevel [1] of the following code?**

```
var relevel = [3,,6];
```

A. Undefined
B. Error
C. Null
D. 0

Answer: A

Array has null value when no value is entered but if value at some index is omitted it takes undefined as value

MCQs

2. What is the functionality of the pop method in an array?

A. decreases the length of array by one
B. increases the length of array by one
C. returns the deleted element
D. deletes the first element of array

MCQs

2. What is the functionality of the pop method in an array?

A. decreases the length of array by one
B. increases the length of array by one
C. returns the deleted element
D. deletes the first element of array

Answer: A

Pop function delete the last element and hence decrease the array length by one

MCQs

3. **What is the output of the following code?**

```
var one = [1,2,3];
var two = [4,5,6];
var ans = one.concat(two);
console.log(ans);
```

A. 4,5,6,1,2,3
B. 1,2,3,4,5,6
C. Error
D. Null

MCQs

3. What is the output of the following code?

```
var one = [1,2,3];
var two = [4,5,6];
var ans = one.concat(two);
console.log(ans);
```

A. 4,5,6,1,2,3
B. 1,2,3,4,5,6
C. Error
D. Null

Answer: B

Concat is a predefined function in js which is used to combine elements of two arrays

MCQs

4. What is the output of javascript code?

```
var sum = 0;
var input = [10,20,30,80];
input.forEach((elem) => {
sum+=elem;    }    );
```

A. 140
B. 100
C. 80
D. 60

MCQs

4. What is the output of javascript code?

```
var sum = 0;
var input = [10,20,30,80];
input.forEach((elem) => {
sum+=elem;    }    );
```

A. 140
B. 100
C. 80
D. 60

Answer: A

forEach work in the same way as for loop and iterates through each element of the array

MCQs

5. What is the output of the following code?

```
var a = [1,2,4,5,6];
console.log(a.slice(0,2));
```

A. [1,2]
B. [5,6]
C. [2,4]
D. [1,2,4]

MCQs

5. What is the output of the following code?

```
var a = [1,2,4,5,6];
console.log(a.slice(0,2));
```

A. [1,2]
B. [5,6]
C. [2,4]
D. [1,2,4]

Answer: A

slice is a predefined function in javascript which is used to return elements of an array from starting point to the ending point as mentioned in function argument.

Practice Questions

1. Write code for creating a new sorted array from two sorted arrays in O(n+m) time complexity where n,m are the size of the unsorted arrays.
2. Write code for finding the smallest element in an array.

Upcoming Class Teaser

- Pick problems from codeforces also for basic arrays
- Anagram
- Print frequency of elements in string
- First non repeating character
- Subarray with sum 0
- Subarray with sum x
- Longest consecutive sequence
- More problems around array and objects (for objects hashing problems can be used)

Thank You!