Problem Solving - Arrays

Relevel
by Unacademy

List of Problems Involved

- Sparse Array
- Shift Negative elements to the end of an array
- Reverse Array
- Reverse subarray to sort array
- Cyclically Rotate Array by 1
- Longest consecutive sequence

Sparse Array

Given a 2D array. Our task is to check if the array is sparse or not.
Sparse Matrix - A matrix is said to be sparse if the number of zeros is more than half of the total elements.

Input:

1	0	15	0
7	0	0	22
0	0	0	0
10	0	0	28

Output:

True

Sparse Array

Approach :

Since it is mentioned that if half of the elements in a given array are zero, then it will be a sparse matrix. So we need to find the total number of zeros in the array. We will compare it with total elements and if it is greater then we will print true else false

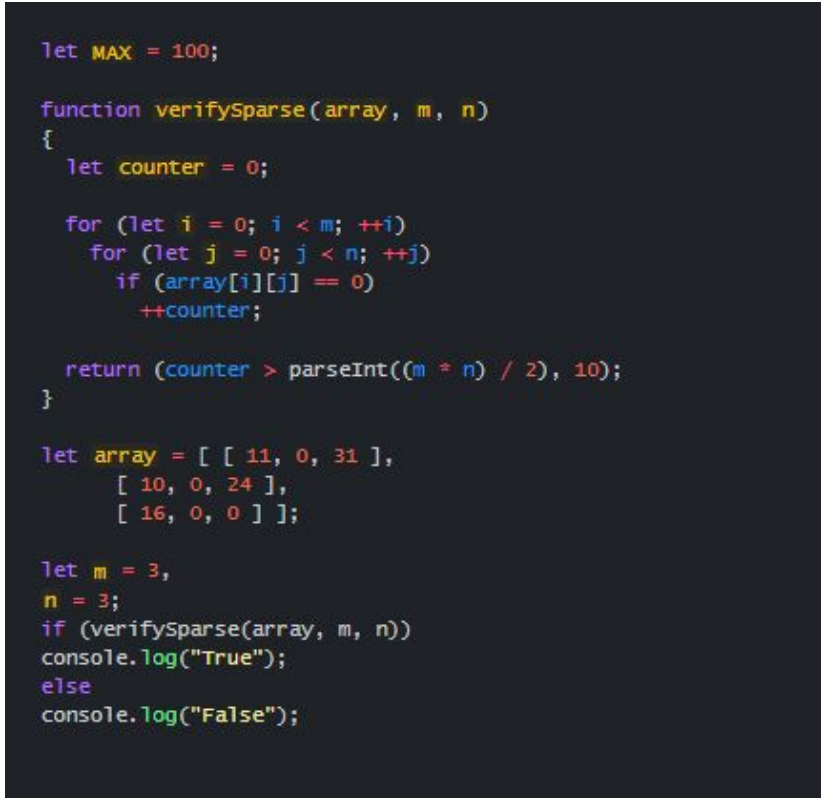
Steps -

1) Let m = number of rows and n = number of columns, counter = 0
2) Total elements = m*n
3) Iterate through the array and increment the counter if the element is 0
4) Compare the counter value with (total elements) / 2

Sparse Array

Time Complexity –
If there are m rows and n columns in the array. Then complexity will be O(m*n)
Space Complexity –
If there are m rows and n columns in the array. Then complexity will be O(1)

CodeLink-> https://jsfiddle.net/ygt9q8f3/



```
let MAX = 100;

function verifySparse(array, m, n)
{
  let counter = 0;

  for (let i = 0; i < m; ++i)
    for (let j = 0; j < n; ++j)
      if (array[i][j] == 0)
        ++counter;

  return (counter > parseInt((m * n) / 2), 10);
}

let array = [ [ 11, 0, 31 ],
      [ 10, 0, 24 ],
      [ 16, 0, 0 ] ];

let m = 3,
n = 3;
if (verifySparse(array, m, n))
console.log("True");
else
console.log("False");
```

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 }

Approach 1(With extra space) – We can use a temp array to store the values. First store all positive numbers in 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.

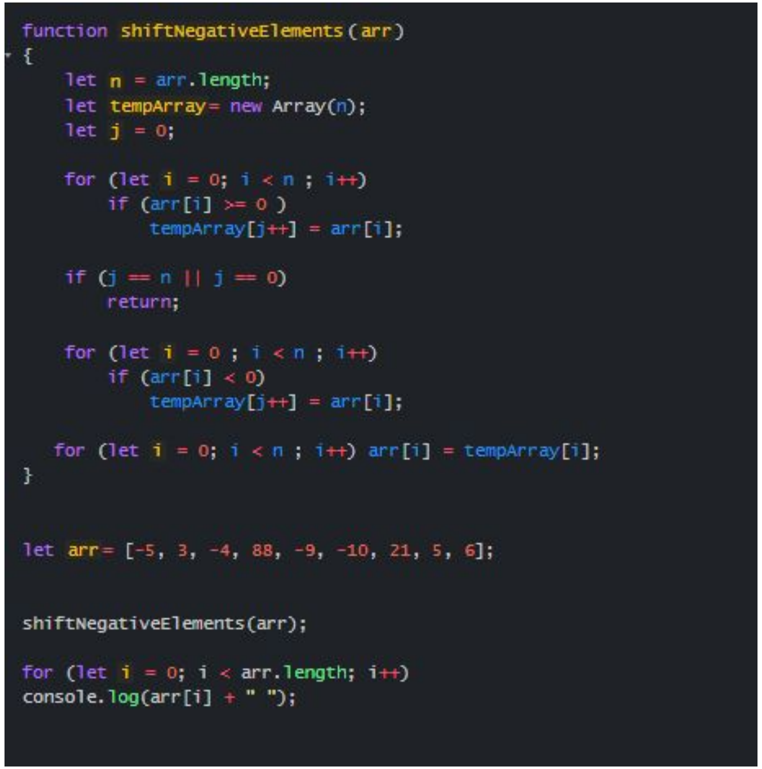
Time Complexity –

If there are N numbers in the 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

Code Link - https://jsfiddle.net/8kLfqtw6/



```
function shiftNegativeElements(arr)
{
    let n = arr.length;
    let tempArray= new Array(n);
    let j = 0;

    for (let i = 0; i < n ; i++)
        if (arr[i] >= 0 )
            tempArray[j++] = arr[i];

    if (j == n || j == 0)
        return;

    for (let i = 0 ; i < n ; i++)
        if (arr[i] < 0)
            tempArray[j++] = arr[i];

   for (let i = 0; i < n ; i++) arr[i] = tempArray[i];
}


let arr= [-5, 3, -4, 88, -9, -10, 21, 5, 6];


shiftNegativeElements(arr);

for (let i = 0; i < arr.length; i++)
console.log(arr[i] + " ");
```

Relevel
by Unacademy

Shift Negative elements to the end of an array

Approach 2(Without extra space) –

We don't need to use any temp array in this approach.

Intuition - Our main task is to re-arrange the numbers in the given array. We can use a two-pointer approach where we can increment or decrement both pointers based on the condition. Let's have a look on all the steps -

Steps -

1) Initialize left pointer to 0 and right pointer to last index of array
2) If left and right pointers are negative, increment left pointer
3) If left pointer is positive and right pointer is negative, swap the elements and increment left pointer and decrement right pointer
4) If left and right pointers are positive, decrement the right pointer
5) Repeat above steps till left <= right pointer

Time Complexity –

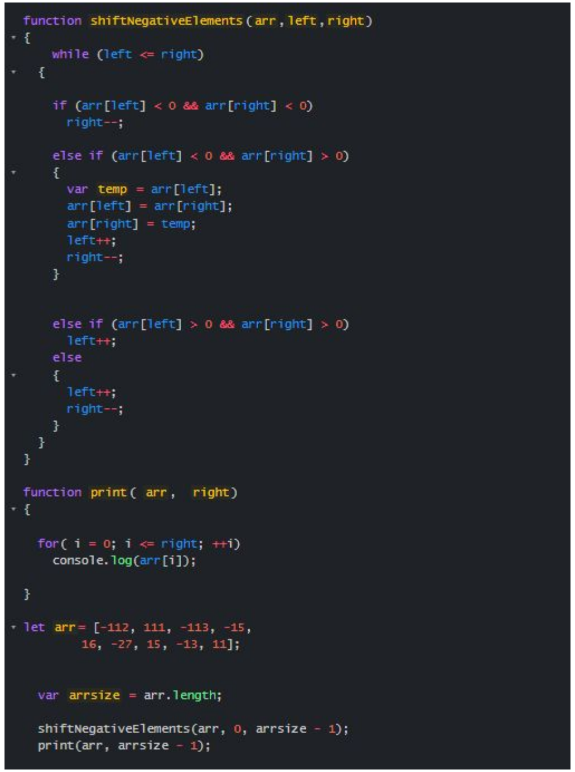
If there are N numbers in the given array arr, then complexity will be O(N)

Space Complexity –

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

Shift Negative elements to the end of an array

Code Link - https://onlinegdb.com/nhT0Zmhsb



```
function shiftNegativeElements(arr,left,right)
{
    while (left <= right)
  {

    if (arr[left] < 0 && arr[right] < 0)
      right--;

    else if (arr[left] < 0 && arr[right] > 0)
    {
      var temp = arr[left];
      arr[left] = arr[right];
      arr[right] = temp;
      left++;
      right--;
    }


    else if (arr[left] > 0 && arr[right] > 0)
      left++;
    else
    {
      left++;
      right--;
    }
  }
}

function print( arr,  right)
{

  for( i = 0; i <= right; ++i)
    console.log(arr[i]);

}

let arr= [-112, 111, -113, -15,
        16, -27, 15, -13, 11];


  var arrsize = arr.length;

  shiftNegativeElements(arr, 0, arrsize - 1);
  print(arr, arrsize - 1);
```

Reverse Array

Problem – Given an array of numbers. You need to reverse the array. For example –
Input – [1,2,3,4,5]
Output – [5,4,3,2,1]

Approach 1 – We can use a swapping operation to get the desired output. Let's see each step –

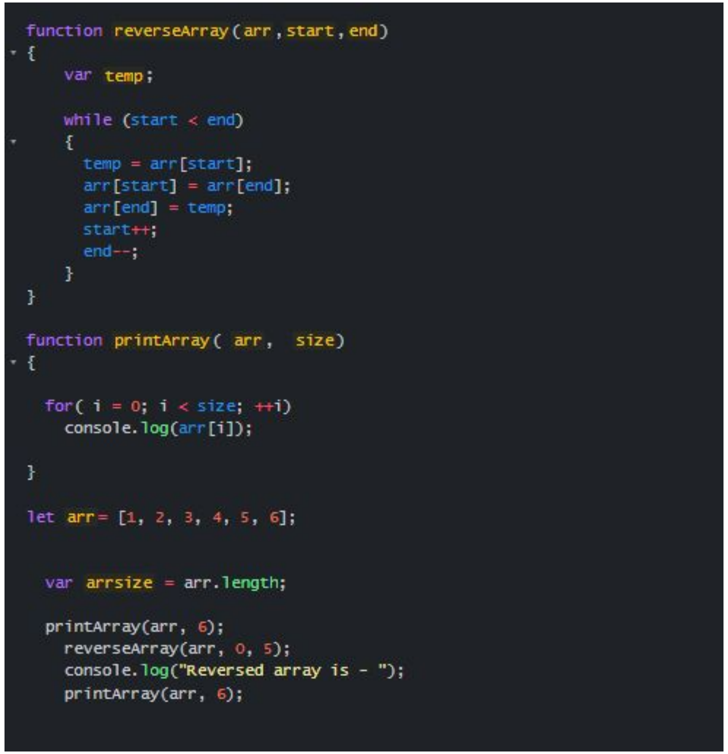
1) Initialize two pointers - start=0, end = n-1 where n = number of elements in array
2) Swap elements present at start and end
3) Start = start+1, end = end -1
4) Repeat step1 till start <= end

Reverse Array

Time Complexity –

If there are N elements in array. Then complexity will be O(N)

Code Link -> https://jsfiddle.net/f9snv853/



```
function reverseArray(arr,start,end)
{
    var temp;

    while (start < end)
    {
      temp = arr[start];
      arr[start] = arr[end];
      arr[end] = temp;
      start++;
      end--;
    }
}

function printArray( arr,  size)
{

  for( i = 0; i < size; ++i)
    console.log(arr[i]);

}

let arr= [1, 2, 3, 4, 5, 6];


  var arrsize = arr.length;

  printArray(arr, 6);
    reverseArray(arr, 0, 5);
    console.log("Reversed array is - ");
    printArray(arr, 6);
```

Reverse Array

Approach 2 – We can use recursion to get the desired output. Let’s see each step –

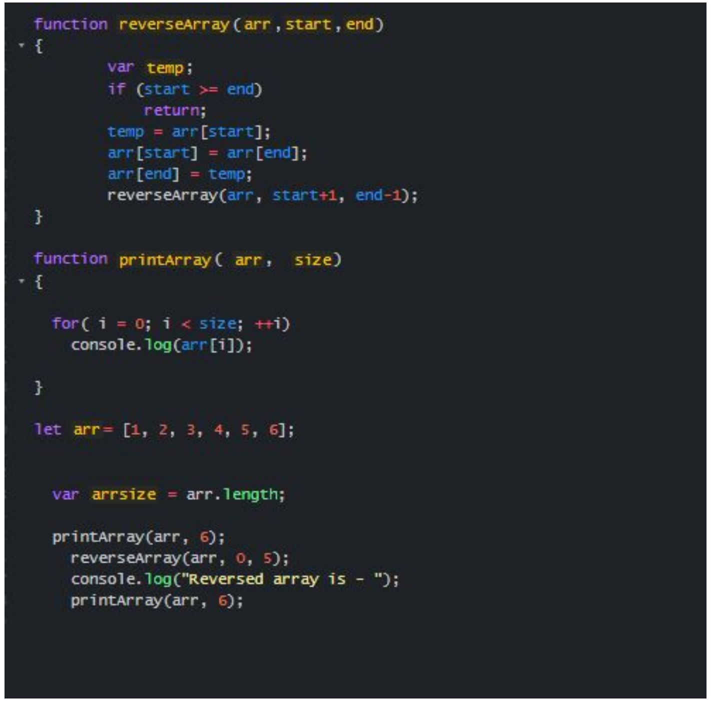
1) Initialize two pointers - start=0, end = n-1 where n = number of elements in array
2) Swap elements present at the start and end
3) Recursively call reverseArray function

Reverse Array

Time Complexity –

If there are N elements in the array. Then complexity will be O(N)

Code Link -> https://jsfiddle.net/f9snv853/1/



```
function reverseArray(arr,start,end)
{
        var temp;
        if (start >= end)
            return;
        temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        reverseArray(arr, start+1, end-1);
}

function printArray( arr,  size)
{

  for( i = 0; i < size; ++i)
    console.log(arr[i]);

}

let arr= [1, 2, 3, 4, 5, 6];


  var arrsize = arr.length;

  printArray(arr, 6);
    reverseArray(arr, 0, 5);
    console.log("Reversed array is - ");
    printArray(arr, 6);
```

Reverse Subarray to Sort Array

Problem – Given an array of numbers. You need to find if reversing a subarray can sort the array. For example –
Input – [1,2,5,4,3]
Output – true (reversing {5,4,3} will sort the array)

Approach –
Intuition - If the array has a structure like first increasing, then decreasing, and then again increasing, then we can say that after reversing the decreasing subarray, we can get our sorted array.

Let's see each step –

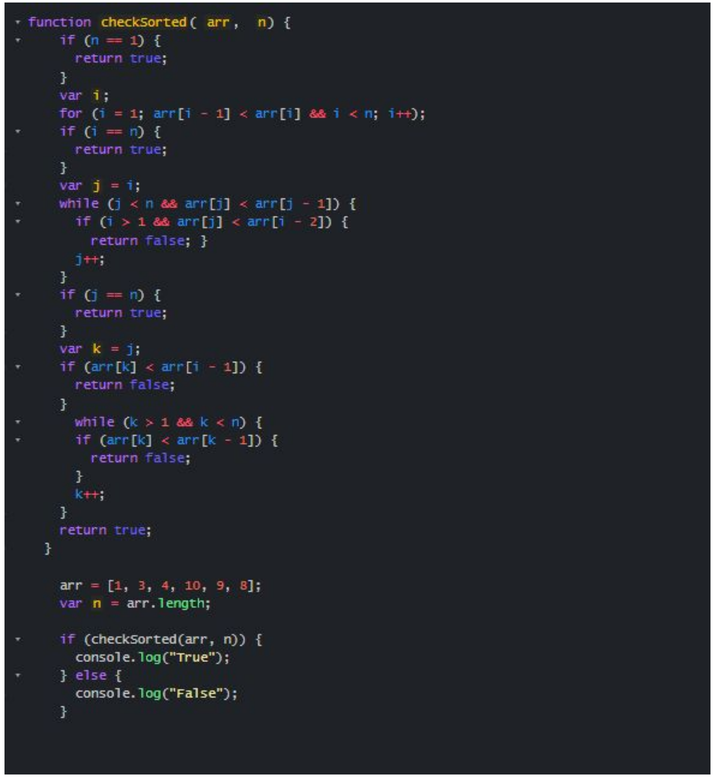
1) Find an Increasing subarray at the start
2) Find decreasing subarray
3) At last, check for an increasing subarray
4) In the third step, if we are unable to find elements, this also gives us case where we can just reverse decreasing subarray of step 2 and get sorted array

Reverse Subarray to Sort Array

Time Complexity –

If there are N elements in the array. Then complexity will be O(N)

Code Link - https://jsfiddle.net/eL4hzq16/



```
function checkSorted( arr,  n) {
    if (n == 1) {
      return true;
    }
    var i;
    for (i = 1; arr[i - 1] < arr[i] && i < n; i++);
    if (i == n) {
      return true;
    }
    var j = i;
    while (j < n && arr[j] < arr[j - 1]) {
      if (i > 1 && arr[j] < arr[i - 2]) {
        return false; }
      j++;
    }
    if (j == n) {
      return true;
    }
    var k = j;
    if (arr[k] < arr[i - 1]) {
      return false;
    }
      while (k > 1 && k < n) {
      if (arr[k] < arr[k - 1]) {
        return false;
      }
      k++;
    }
    return true;
  }

    arr = [1, 3, 4, 10, 9, 8];
    var n = arr.length;

    if (checkSorted(arr, n)) {
      console.log("True");
    } else {
      console.log("False");
    }
```

Cyclically Rotate Array by 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 the whole array. For example –

Input – { 3, 88, 21, 5, 6}
Output – { 6,3, 88, 21, 5}

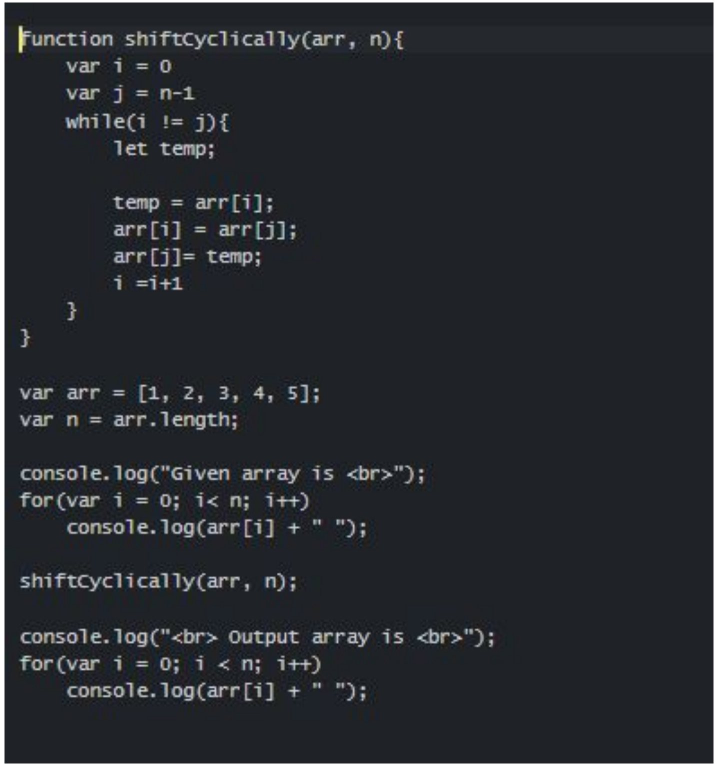
Approach – We can use two pointers like i and j which will point to the first and last elements of an array. We need to swap i and j till i is not equal to j.
Intuition - Here, we need to keep j fixed and move i to the right direction.

Cyclically Rotate Array by 1

Code Link - https://jsfiddle.net/jkpdL1h3/

Time Complexity –
If there are N numbers in the 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)



```
function shiftCyclically(arr, n){
    var i = 0
    var j = n-1
    while(i != j){
        let temp;

        temp = arr[i];
        arr[i] = arr[j];
        arr[j]= temp;
        i =i+1
    }
}

var arr = [1, 2, 3, 4, 5];
var n = arr.length;

console.log("Given array is <br>");
for(var i = 0; i< n; i++)
    console.log(arr[i] + " ");

shiftCyclically(arr, n);

console.log("<br> Output array is <br>");
for(var i = 0; i < n; i++)
    console.log(arr[i] + " ");
```

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

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.

Longest Consecutive Sequence

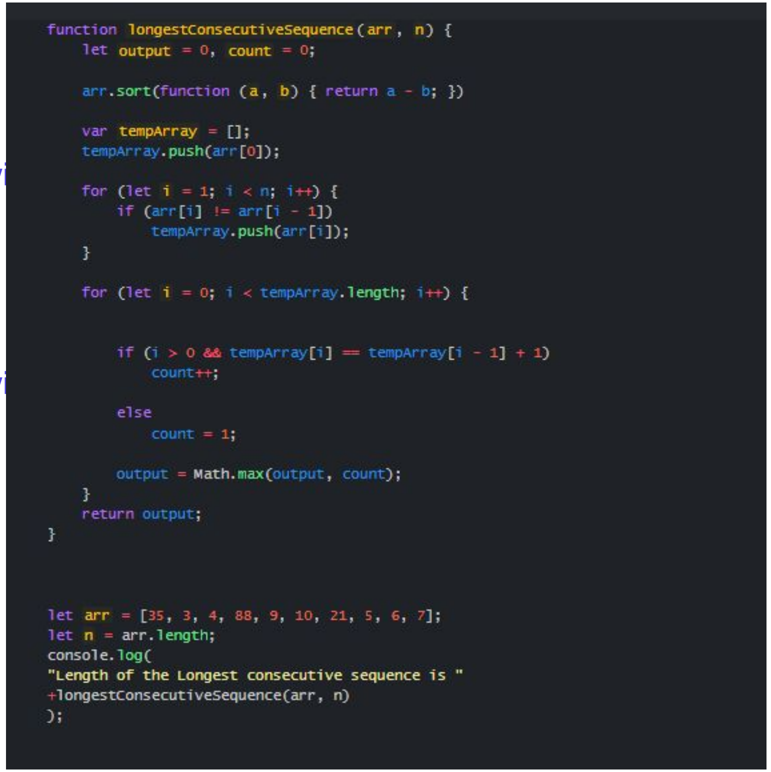
Code Link - https://jsfiddle.net/hp5cgrLb/

Time Complexity –

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

Space Complexity –

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



```
function longestConsecutiveSequence(arr, n) {
    let output = 0, count = 0;

    arr.sort(function (a, b) { return a - b; })

    var tempArray = [];
    tempArray.push(arr[0]);

    for (let i = 1; i < n; i++) {
        if (arr[i] != arr[i - 1])
            tempArray.push(arr[i]);
    }

    for (let i = 0; i < tempArray.length; i++) {


        if (i > 0 && tempArray[i] == tempArray[i - 1] + 1)
            count++;

        else
            count = 1;

        output = Math.max(output, count);
    }
    return output;
}



let arr = [35, 3, 4, 88, 9, 10, 21, 5, 6, 7];
let n = arr.length;
console.log(
"Length of the Longest consecutive sequence is "
+longestConsecutiveSequence(arr, n)
);
```

MCQ Questions

1) What will be the output array when we rotate this array by 2 in right direction?

Input -> [2,3,4,5,6,7,8]

A) 4,5,6,7,8,3,2 [Correct Answer]

B) 3,4,5,6,7,8,2

C) 5,6,7,2,3,4,8

D) None

2) When we rotate an array using its elements one by one, it is an in-place algorithm

A) True [Correct Answer]

B) False

3) Which statement is true about Arrays?

A) Arrays are immutable

B) Arrays are not linear data structure

C) Arrays are used to store similar datatypes {Correct Answer]

D) None

4) Which statement is not an advantage forArrays?

A) Stack can be implemented using Arrays
B) Queue can be implemented using Arrays
C) Memory inefficient [Correct Answer]
D) Indexing

5) When we delete an element from the index which is not present in the array, which condition is that?

A) Overflow
B) Underflow [Correct Answer]
C) Garbage
D) None

Practice Questions

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

Input Array -> [0,1,1,0,1,0,0,1]

Output -> [0,0,0,0,1,1,1,1]

2) Given 2D Array. A Magic square is a 2D array having n rows and n columns which will be having all elements as distinct and the sum of row, column or diagonal is equal to the same number.

You need to check whether that array is a magic square or not.

THANK YOU