# JavaScript and Data Structure

## Data Structure

### Problems

#### Hashing and Array/String

* 1. Is Unique: Implement an algorithm if a string has all unique character. What if you cannot use additional data structure?

|  |
| --- |
| const checkUniqueCharacters =(text) => {  //checking ASCII character  if(text.length > 128) {  return false;  }  let hash = {}  for(item of text)  {  //If the key is already present in the hash table return false  if(hash[text[item]]){  return false  }  else {  // storing the value in hashtable  /\*hash {  text[i]: i;  }\*/  hash[text[item]] = item;  }  }  return true;  }  console.log(checkUniqueCharacters('1232'));  //Ask is it ASCII or Unicode String, assume as it is ASCII where max character can be 128  //Use Hash table, hashing also leads to collision.  //Time Complexity is O(n) , but you can say it will O(c) where c is the size of the character set (128)  //Space complexity O(1)  // Also we can improve the Time Complexity to O(n log(n)), if we sort the text, but remember it sorting can take more space. |

* 1. Check Permutation: Given two strings, write a method to decide if one is permutation of the other?

|  |
| --- |
| const checkIsPermutationofString = (str1,str2) => {  if(str1.length !== str2.length) {  return false  }  else {  if(str1 === str2) {  return true  }  return Sort(str2) === Sort(str1);  }  }  const Sort = (str) => {  const array1 = str.split('');  const sortValue = array1.sort();  return sortValue.toString();  }  // Check the length of both the string is they are not matched then it is not permutated  // Sol 1:- Sort both the string and compare    console.log(checkIsPermutationofString('abe','cab')); |

* 1. The goal of this exercise is to convert a string to a new string where each character in the new string is "(" if that character appears only once in the original string, or ")" if that character appears more than once in the original string. Ignore capitalization when determining if a character is a duplicate.The goal of this exercise is to convert a string to a new string where each character in the new string is "(" if that character appears only once in the original string, or ")" if that character appears more than once in the original string. Ignore capitalization when determining if a character is a duplicate.

|  |
| --- |
| /\*  The goal of this exercise is to convert a string to a new string where each character in the new string is '(' if that character appears only once in the original string, or ')' if that character appears more than once in the original string. Ignore capitalization when determining if a character is a duplicate.  \*/  function duplicateEncode(word){  var string = "";  var lowerCaseWord = word.toLowerCase();  for (var i = 0; i < lowerCaseWord.length; i++) {  if (lowerCaseWord.indexOf(lowerCaseWord[i]) === lowerCaseWord.lastIndexOf(lowerCaseWord[i])) {  string += "(";  } else {  string += ")";  }  }  return string;  }  duplicateEncode("din"); // "((("  duplicateEncode("recede"); // "()()()"  duplicateEncode("Success"); // ")())())"  duplicateEncode("(( @"); // "))((" |

* 1. Categorize New Member- Two dimensional array

|  |
| --- |
| /\*  The Western Suburbs Croquet Club has two categories of membership, Senior and Open. They would like your help with an application form that will tell prospective members which category they will be placed.  To be a senior, a member must be at least 55 years old and have a handicap greater than 7. In this croquet club, handicaps range from -2 to +26; the better the player the lower the handicap.  Input  Input will consist of a list of lists containing two items each. Each list contains information for a single potential member. Information consists of an integer for the person's age and an integer for the person's handicap.  Note for F#: The input will be of (int list list) which is a List>  Example Input  [[18, 20],[45, 2],[61, 12],[37, 6],[21, 21],[78, 9]]  Output  Output will consist of a list of string values (in Haskell: Open or Senior) stating whether the respective member is to be placed in the senior or open category.  Example Output  ["Open", "Open", "Senior", "Open", "Open", "Senior"]  \*/  function openOrSenior(data){  var result = []  for(var a = 0; a < data.length; a++){  if(data[a][0] >= 55 && data[a][1] > 7){  result.push('Senior')  } else {  result.push('Open')  }  }  return result  }  console.log(openOrSenior([[45, 12],[55,21],[19, -2],[104, 20]]))//['Open', 'Senior', 'Open', 'Senior']  console.log(openOrSenior([[3, 12],[55,1],[91, -2],[54, 23]]))//['Open', 'Open', 'Open', 'Open']  console.log(openOrSenior([[59, 12],[55,-1],[12, -2],[12, 12]]))//['Senior', 'Open', 'Open', 'Open']  console.log(openOrSenior([[90, 9], [54, 9], [60, 12], [21, 21], [0, 0], [90, 8], [1, 1], [55, 10], [75, 11], [90, 7]]))//[ 'Senior','Open','Senior','Open','Open','Senior','Open','Senior','Senior','Open' ]  //other solution  const openOrSenior = (data) => {  return data.map(([age,handicap]) => (age > 54 && handicap < 7 ) ? 'Senioe' : 'Open');  } |

* 1. The new "Avengers" movie has just been released! There are a lot of people at the cinema box office standing in a huge line. Each of them has a single 100, 50 or 25 dollars bill. A "Avengers" ticket costs 25 dollars.

Vasya is currently working as a clerk. He wants to sell a ticket to every single person in this line.

Can Vasya sell a ticket to each person and give the change if he initially has no money and sells the tickets strictly in the order people follow in the line?

Return YES, if Vasya can sell a ticket to each person and give the change. Otherwise return NO.

|  |
| --- |
| function tickets(peopleInLine){  // ...  let [c25, c50, c100] = [0,0,0];  for(item of peopleInLine){  if(item === 25) { c25++;}  if( item === 50) {c50++; c25--;}  if(item === 100) {c25--; c50>0?c50--:c25-=2}  if(c25<0 || c50< 0){  return 'NO';  }  }  return 'YES';  }  console.log(tickets([25, 25, 50, 100])); |

* 1. Count Vowels in a string

|  |
| --- |
| function getCount(str) {  var vowelsCount = 0;  const vowels = ['a','e','i','o','u'];  // enter your majic here  for(item of str){  for(vowel of vowels){  if(item === vowel){  vowelsCount++;  }  }  }  return vowelsCount;  }  //or  function getCount(str) {  return str.split('').filter(c => "aeiouAEIOU".includes(c)).length;  } |

* 1. Given an array of integers A sorted in non-decreasing order, return an array of the squares of each number, also in sorted non-decreasing order.

**Example 1:**

**Input:** [-4,-1,0,3,10]

**Output:** [0,1,9,16,100]

|  |
| --- |
| const sortedSquares = (A) => A.map(e => e\*e).sort((a,b) => a-b);  console.log(sortedSquares([-7,-3,2,3,11])) |

* 1. Given a m \* n matrix mat of *ones* (representing soldiers) and *zeros* (representing civilians), return the indexes of the k weakest rows in the matrix ordered from the weakest to the strongest.

A row **i** is weaker than row **j**, if the number of soldiers in row **i** is less than the number of soldiers in row **j**, or they have the same number of soldiers but **i** is less than **j**. Soldiers are **always** stand in the frontier of a row, that is, always *ones* may appear first and then *zeros*.

**Example 1:**

**Input:** mat =

[[1,1,0,0,0],

[1,1,1,1,0],

[1,0,0,0,0],

[1,1,0,0,0],

[1,1,1,1,1]],

k = 3

**Output:** [2,0,3]

**Explanation:**

The number of soldiers for each row is:

row 0 -> 2

row 1 -> 4

row 2 -> 1

row 3 -> 2

row 4 -> 5

Rows ordered from the weakest to the strongest are [2,0,3,1,4]

|  |
| --- |
| var kWeakestRows = function(mat, k) {  let storeArray= [];  let total = 0;  for(let j = 0; j<mat.length; j++){  for(let item of mat[j]){  if(item === 1){  total++;  }  }    storeArray.push([j,total]);  total = 0;  }  storeArray.sort(([i1, c1], [i2, c2]) => c1-c2 || i1 - i2);    return storeArray.map(([index]) => index).slice(0,k);    };  Or  var kWeakestRows = function(mat, k) {  return mat.map((row,i)=>[i,row.reduce((a,b) => a+b)])  .sort(([i1,c1],[i2,c2])=> c1-c2 | i1 - i2)  .map(([index])=> index).slice(0,k);  };  console.log(kWeakestRows([[1,1,0,0,0],  [1,1,1,1,0],  [1,0,0,0,0],  [1,1,0,0,0],  [1,1,1,1,1]],3)) |

#### Math Function

* 1. In this little assignment you are given a string of space separated numbers, and have to return the highest and lowest number.

|  |
| --- |
| function highAndLow(numbers){  // ...  let array1 = numbers.split(' ').sort();  return Math.max.apply(null, array1) + " " + Math.min.apply(null, array1)  }  console.log(highAndLow("4 5 29 54 4 0 -214 542 -64 1 -3 6 -6")); |

## JavaScript

1. Covert string to array?

Using split()

const array1 = str.split('');

Convert array to string

1. Using join()

const string = array1.join(‘’)

1. Find the index of a value from a string

indexOf('') and lastIndexOf('')

1. Min and max in array

|  |
| --- |
| Math.max.apply(null, array1) and Math.min.apply(null, array1) |

1. Copy one array to another without affecting the other one

let heightsSorted = [].concat(heights);