CS 1290

Midterm

Jose Lujan

15 Problems from leetcode

Level medium

**Problem 1:** Find the duplicate number

**Description:** Given an array *nums* containing *n* + 1 integers where each integer is between 1 and *n*(inclusive), prove that at least one duplicate number must exist. Assume that there is only one duplicate number, find the duplicate one.

**Example:**

**Example 1:**

**Input:** [1,3,4,2,2]

**Output:** 2

**Example 2:**

**Input:** [3,1,3,4,2]

**Output:** 3

**Questions I will ask:**

* Can the array have negative numbers?
* Is the array a 2-d array?
* Does the length of the array has a maximum length?

**Using IDEAL:**

To get a better understanding of the problem, I read the problem. At first I thought, what if we have more than one duplicate. Once I read the problem again I notice that it says, that there should only be one duplicate. What I want to get from the array, it will be one number that it appears more than once. The easiest way to solve the problem will be to create a second array to store the integer that has been seen before, another solution could be to have a counter and if the counter gets equal to two, then that will be the output.

**Dukes 7 steps:**

I used dukes 7 steps, to get a better understanding of how I could solve the problem, In the first step is to try small instances, so I had an array which had at the most three integers, and then decreased until the array was empty. I did this to get the edge cases. In this problem we could create a new array that will store the integer that have a duplicate.

**Problem 2:** House Robber lll

**Description:** The thief has found himself a new place for his thievery again. There is only one entrance to this area, called the "root." Besides the root, each house has one and only one parent house. After a tour, the smart thief realized that "all houses in this place forms a binary tree". It will automatically contact the police if two directly-linked houses were broken into on the same night.

Determine the maximum amount of money the thief can rob tonight without alerting the police.

**Example:**

**Example 1:**

**Input:** [3,2,3,null,3,null,1]

3

/ \

2 3

\ \

3 1

**Output:** 7

**Explanation:** Maximum amount of money the thief can rob = 3 + 3 + 1 = **7**.

**Questions I will ask:**

* Does the fact that all the houses in the place form a binary tree, help to decide which houses to rob, like if the parent has two childs its better to always skip the parent to go to all the children from that parent node?

**Using IDEAL:**

In this problem what we get its an actual binary tree, but the input is represented as an array, in which index 0 is the root, what we want to get from the binary three is basically to know which pattern will give us the maximum number by adding the amount each node has inside. The first thing will be to create the edge case, in this problem would be to return a 0 if the binary three is empty. To solve this problem I will try a recursive method that will add the childs and compare them to the parent if its greater it will go to the next root.

**Dukes 7 steps:**

To try solving this problem we know for sure that it will return a 0 if its empty and it will return the item in the root if it has no children. If it has children we will have to add the items from each child and compare them to the parent.

**Problem 3: Minimum Path Sum:**

Description: Given a m x n grid filled with non-negative numbers, find a path from top left to bottom right which minimizes the sum of all numbers along its path.

**Note:** You can only move either down or right at any point in time.

**Example:**

**Input:**

[

  [1,3,1],

[1,5,1],

[4,2,1]

]

**Output:** 7

**Explanation:** Because the path 1→3→1→1→1 minimizes the sum.

**Questions I will ask:**

* We know for sure that it can’t have negative numbers, but can the array have 0’s?
* Is the array only filled with numbers from 0-9, or can it have a two digit number?

**Using IDEAL:**

To understand better the problem I looked at the example, that way I knew I had a 2d array with arrays, and every array had three integers inside. The arrays weren’t sorted, so that means that the “min” integer in each array could be at any index from [0] to len-1. To solve this problem the best thing to do is to compare all the integers from the first array and keep the path that will return the minimum sum from all the arrays.

**Using Dukes 7 steps:**

The first thing we need to do is to check if the grid is empty. I the grid is not empty we have to go from one cell and compare it to the sum of the current value of i and j. once it has the minimum number it will continue adding up, until it has passed through every cell in the grid, which will return the result from the addition.

Problem 4: Group Anagrams #49

Description: Given an array of strings, group anagrams together.

**Example:**

**Input:** ["eat", "tea", "tan", "ate", "nat", "bat"],

**Output:**

[

["ate","eat","tea"],

["nat","tan"],

["bat"]

]

**Note:**

* All inputs will be in lowercase.
* The order of your output does not matter.

**Questions I will ask:**

* Can we assume that all the strings have the same length?
* What should we do with the strings that only have one anagram?

**Ideal:**

For this problem, basically we just have some random words, and we need to see how many anagrams does each word has, comparing all the string from the array, once we know which array is an anagram with another word. We will group them together. Once we have all the groups we can put each group into a new array. For the solutions we can just compare the amount of characters that each word has, in the example we know that all the words have the same length, but if we were to get words that have different length. In this problem if we were looking at edge case, it would be like if there is a word with a certain amount of characters, but that’s the only word with that size in the whole array then that word would be by itself in a different array.

**Dukes 7 steps:**

To try this problem with smaller instances, we can try an array that only has one word. Which will be just one anagram. The next case would be an array with just two words, if the aren’t the same length, it will create two arrays. So, like in the second case we have two words

Problem 5: Longest Substring Without Repeating Characters

Given a string, find the length of the **longest substring** without repeating characters.

**Example 1:**

**Input:** "abcabcbb"

**Output:** 3

**Explanation:** The answer is "abc", with the length of 3.

**Example 2:**

**Input:** "bbbbb"

**Output:** 1

**Explanation:** The answer is "b", with the length of 1.

**Questions I will ask:**

* Does it matter whether if the characters are capital or lower case?

**Ideal:**

The problem has any size of characters, in which all the characters are repeating we will get a “1”, this is something we can see in the example #2. If the character changes throughout out all the array, then we need to see when do we see a cycle, in which the first integer we saw its repeating for the second time and if the next character were also seen before, we need to count the number of character the whole string has. Like in example #1 with input “abcabcbb” we see a cycle when a its seen for the second time, the next two characters were also seen before, so we just count the character in “abc”

**Dukes 7 steps:**

We can try solving this problem in which the array only has two characters, and it the two characters are different then there is two substrings , but each of them is of length 1. We need to try we an array that has more characters, once we see that there is a pattern, we just need to count the characters in the substrings and compare them. The substring that has the maximum amount of characters is the longest substring.

**Problem 6: Perfect Squares**

Given a positive integer *n*, find the least number of perfect square numbers (for example, 1, 4, 9, 16, ...) which sum to *n*.

**Questions I will ask:**

* For the input “n”, can it be an odd number, or we can assume that all the inputs will be even numbers?

**Ideal:**

To solve this problem we need to determine how many square number add up to the value of “n”. For the square number we have some options which are: 1,4,9,16. If we get an integer for the value like 12 we know that we can’t use 16 as a perfect square because its already greater than the number given as an input. We might use 9, but that just means that we will subtract 9 from 12 which would be 3= 1+1+1. To solve this problem it would better to use 4 as the perfect square, since it will be 12= 4+4+4

**Dukes 7 steps:**

A smaller instance of this problem could like just like using 2 as an input and since we know that we can not use a perfect square greater than the input number. That means that we will only be using 1 as many times we need it for until, it adds to 2. For an input greater, we will just compare the value of “n” with the perfect squares, once we have a perfect square less than the value of n. We will subtract the perfect square from ‘n’.

**Problem 7 Sort list:**

Description: Sort a linked list in O(n log n) time using constant space complexity.

**Example 1:**

**Input:** 4->2->1->3

**Output:** 1->2->3->4

**Example 2:**

**Input:** -1->5->3->4->0

**Output:** -1->0->3->4->5

**Questions I will ask:**

* Can it be a double linked list?

**Ideal: To solve this problem we know that the smallest number in the linked-list has to be in the first node and the number with the node with the biggest number hast to be at the end. Once we have those two values all the other numbers have to be in between, but this time sorted. A solution for this will be using two pointers.**

**Dukes 7 steps:**

If the linked-list has just two integers, we will compare and switch place(the head pointer will change place). If the problem has more than two integers we can do this using a recursive method. That will compare each node with the previous one, and if the node before the current node is greater it will change values using a temp variable.

**Problem 8: Word Search**

Given a 2D board and a word, find if the word exists in the grid.

The word can be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The same letter cell may not be used more than once.

**Example:**

board =

[

['A','B','C','E'],

['S','F','C','S'],

['A','D','E','E']

]

Given word = "**ABCCED**", return **true**.

Given word = "**SEE**", return **true**.

Given word = "**ABCB**", return **false**.

**Questions I will ask:**

* The word can more than once the same letter, except if its from the same cell?

**Ideal:**

As we can see in the example, the word can actually have more than once the same character. The only thing is that this letter that repeats can’t be from the same cell. We just want to create a word with the letters from the grid, and if we have visited a certain cell we need to skip it. Like in the example we only have “B” one time in the whole grid, so that means that if we get the word “abcb” that we have visited the same cell twice, so its breaking the rule.

**7 dukes steps:**

If the word given it only has different letters, for sure we know that it will return a true statement. The only thing is when we have a word that repeats a letter, in that case we need to know how many times does the letter appears in the grid. If that letter in specific only appears one time, we already know that it will return false.

**Problem 9: Sort colors**

Given an array with *n* objects colored red, white or blue, sort them [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm)so that objects of the same color are adjacent, with the colors in the order red, white and blue.

Here, we will use the integers 0, 1, and 2 to represent the color red, white, and blue respectively.

**Note:** You are not suppose to use the library's sort function for this problem.

**Example:**

**Input:** [2,0,2,1,1,0]

**Output:** [0,0,1,1,2,2]

**Questions I will ask:**

* What should I do, if the array has a color that is not red, while or blue?
* Will the array always have the same amount from each color?

**Ideal:**

The options in this problems as an output is just one, which is [0,1,2]. The only thing that will change is if we have a 1 before a 0 or a 2 before a 0. To solve this problem we can have an array that searches for 0 and if “i” is greater than 0, then the next operation will be to move it at the end.

**7 dukes steps:**

We can have an array with just two colors, any of the three and If the second color is smaller than the first one it will get placed into a temp variable. Once we increment the number of colors, we just need to keep doing the same operation like if the next value from the current is greater, if it’s the same it can be kept in the same place.

**Problem 10: Permutations**

Given a collection of **distinct** integers, return all possible permutations.

**Example:**

**Input:** [1,2,3]

**Output:**

[

[1,2,3],

[1,3,2],

[2,1,3],

[2,3,1],

[3,1,2],

[3,2,1]

]

**Questions I will ask:**

* Do all permutations have to be in order?

**Ideal:**

If the problem has an array with three elements inside, it can only have elements = (elements)\*(elements-1). In the example given. The input array has 3 elements, so if we use the formula we can only have six different arrays that contain the same elements, but in different order.

**Dukes 7 steps:**

If the problem is has less than 3 elements it will only return two arrays, because we can only have item1,item2 and the other way around: item2,item1. When the problem increases and it has more elements we need to use the formula at least to know how many arrays we will get as an output, and make sure that the elements in the arrays don’t repeat the same order.

**Problem 11: coin change**

You are given coins of different denominations and a total amount of money *amount*. Write a function to compute the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return -1.

**Example 1:**

**Input:** coins = [1, 2, 5], amount = 11

**Output:** 3

**Explanation:** 11 = 5 + 5 + 1

**Example 2:**

**Input:** coins = [2], amount = 3

**Output:** -1

**Note**:  
You may assume that you have an infinite number of each kind of coin.

**Questions I will ask:**

* Can we use any amount of times the same coin, or only two times?
* If there is no need to return change, what should I return?

**Ideal:**

First thing is to know what is the amount of change we need to give in return , and which coins do we have in that moment. Like in example #2 : we had to give 3 as the change, but the only coin value available has a value of 2, if we give two of those it will be greater than what we have to return and if we give just one coin it will be less than the change we have to return. If we have to give a certain amount of change, but we just have one coin available we know for sure that it will return a “-1”. Because either we will be giving more or less than the amount needed.

**7 dukes steps:**

If the coin available its greater than the change amount, it will return negative or if the amount of coins available are just one most likely it will also return a negative value. If the coins available are more than one and the coin value is less than the change we have to return, the next step will be to subtract the change amount from the coin available that is less than that number(change amount). If the result is the same as the coin value it means we will return two coins of the same value, if not we have to add a coin that is the same value as the result from the subtraction of less than that number.

**Problem 12: Search a 2d Matrix ll**

Write an efficient algorithm that searches for a value in an *m* x *n* matrix. This matrix has the following properties:

* Integers in each row are sorted in ascending from left to right.
* Integers in each column are sorted in ascending from top to bottom.

**Questions I will ask:**

* Can a number be duplicated?
* Does the matrix has negative numbers?

**Ideal:**

**7 duke steps:**

**Example:**

Consider the following matrix:

[

[1, 4, 7, 11, 15],

[2, 5, 8, 12, 19],

[3, 6, 9, 16, 22],

[10, 13, 14, 17, 24],

[18, 21, 23, 26, 30]

]

Given target = 5, return true.

Given target = 20, return false.

**Problem 13:Linked List Cycle ll:**

Given a linked list, return the node where the cycle begins. If there is no cycle, return null.

**Note:** Do not modify the linked list.

**Questions I will ask:**

* What happens if the item from the first and last node are the same?

**Ideal:**

**7 duke steps:**

Problem 14:Binary Tree Inorder Traversal

Given a binary tree, return the inorder traversal of its nodes' values.

**Example:**

**Input:** [1,null,2,3]

1

\

2

/

3

**Output:** [1,3,2]

**Follow up:** Recursive solution is trivial, could you do it iteratively?

**Questions I will ask:**

* None:

**Ideal:**

**7 duke steps:**

Even though this question only asked for iterative inorder traversal, I tried to figure out all three type of binary tree traversal.

problem 15: Jump Game

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

Determine if you are able to reach the last index.

**Example 1:**

**Input:** [2,3,1,1,4]

**Output:** true

**Explanation:** Jump 1 step from index 0 to 1, then 3 steps to the last index.

**Example 2:**

**Input:** [3,2,1,0,4]

**Output:** false

**Explanation:** You will always arrive at index 3 no matter what. Its maximum

  jump length is 0, which makes it impossible to reach the last index.

**Questions I will ask:**

* Does the array has repeating integers?

**Ideal:**

**7 duke steps:**