**Date:** 26-08-2021

**Time:** 10:30 – 11:15 AM IST

**Where:** Google Meet and Shared Document for Coding

**Duration:** 45 Minutes

**Language:** JavaScript

**Write a program and explain the logic from end-to-end including the efficiency.**

Sample Input: W = {ben, bet, bent, bets, den, dent, dents, dets}

Sample output: den -> dent -> dents

Condition: Only one letter can be extra from previous word.

**Question:**

Given a set of words, find the **longest chain** of words that can be made out of those words with the following rules:

* Each word in the chain is *one letter* longer than the previous word.
* Each word in the chain differs from the *previous word* only by its last letter.

Write test case's. Also, mention the **Time** and **Space** complexity for the same.

**Constraint Examples:**   
  
den -> dent-> dents is valid (meets all constraints)   
  
den -> dew is not valid (same length: 3 characters)   
  
den -> cent is not valid (differs by 1’st char [‘d’ != ‘c’])

**Example:**

Input: {ben, bent, dew, dents, dent, bet, den}

Output: 3 ({den, dent, dents})

**Interviewer:**

den -> dent-> dents

W = {ben, bet, bent, bets, den, dent, dents, dets} // {ben, bet, bent} ----> resultArray = [ben (3) - bent (4)] ----> resultArray = [ben

ben -> bent (2)

bet -> bets (2)

den -> dent -> dents (3) # output

W : dictionary

i) only one letter can be extra from previous word

**Pranam:**

{ben, bet, dent, dents, bets, den, dets,}

function (arrInput){

//var arrInput = [];

if (arrInput.length != null){ //Check if array is empty or not

var resultArray = [];

var count = arrInput[0].length; # 3

for(var i=0;i<arrInput.length; i++){

if (arrInput[i].length > count){ //To check array element length ben - 1st

resultArray = [arrInput[i]]; // keep adding elemets if length is greater than 3

}

elseif (resultArray[i] > i+n){ bets(4) - bette (5) (6).. unless length greater

}

console.log(resultArray); //Result

}

}

}

## dent dents bets dets

## den dent dents < expected>

console.log(arrInput[ben, bet, bent, bets, den, dent, dents, dets]); //Input Array

**Explanation:**

Input: {ben, bet, dent, dents, bets, den, dets,}

//Check if array is empty or not

//To check array element length ben - 1st

// keep adding elements if length is greater than 3 bent – char count (4)

Print resultArray as output.

**Time Complexity:** O(n)