Reverse Integer

In [2]:
```
a = int(input('Enter the number'))

# Method 1
astr = str(a)
for i in range(len(astr)-1,-1,-1):
    print(astr[i],end='')

print()

# Method 2
ast = str(a)
print(ast[::-1])
```

```
312
312
```

Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays.

In [3]:
```
# Method 1
nums1 = [1,3]
nums2 = [2,7]
mergeList = sorted(nums1 + nums2)
s = sum(mergeList)/len(mergeList)
print(f'Median of two list is {s}')

# Method 2
import numpy as np
med = np.median(mergeList)
print(f'Median of two list using Numpy package is {med}')
```

```
Median of two list is 3.25
Median of two list using Numpy package is 2.5
```

Longest Palindromic Substring

(1) Initialize variable revs_number = 0

(2) Loop while number > 0

```
        (a) Multiply revs_number by 10 and add remainder of
   number
              divide by 10 to revs_number
                   revs_number = revs_number*10 + number%10;

        (b) Divide num by 10
```

(3) Return revs_number

In [4]:
```
# Part 1 to check if the string is palindrome or not
def checkPali(s):
    return s == s[::-1]
inString = str(input())
inString2 = checkPali(inString)
if inString2:
    print(f'{inString} is a palindrome')
else:
    print(f'{inString} is not a palindrome')

# Part 2 to check longest palindromic string
cout = 0
def recur_reverse(num):
    global cout   # We can use it out of the function
    if (num > 0):
        Reminder = num % 10
        cout = (cout * 10) + Reminder
        recur_reverse(num // 10)
    return cout
cout = recur_reverse(int(inString))
if cout == inString:
    print(f'{inString} is a logest palindrome')
else:
    print(f'{inString} is not a longest palindrome')
```

```
123321123123 is not a palindrome
123321123123 is not a longest palindrome
```

Implement atoi which converts a string to an integer.

Sample 1

Input: str = "42"

Output: 42

Sample 2

Input: str = " -42"

Output: -42

Explanation: The first non-whitespace character is '-', which is the minus sign. Then take as many numerical digits as possible, which gets 42.

Sample 3

Input: str = "4193 with words"

Output: 4193

Explanation: Conversion stops at digit '3' as the next character is not a numerical digit.

Sample 4

Input: str = "words and 987"

Output: 0

Explanation: The first non-whitespace character is 'w', which is not a numerical digit or a +/- sign. Therefore no valid conversion could be performed.

Sample 5

Input: str = "-91283472332"

Output: -2147483648

Explanation: The number "-91283472332" is out of the range of a 32-bit signed integer. Thefore INT_MIN (−231) is returned.

In [5]:
```
# Method 1
import re
string = '4193 words'
a = string.split( )
for i in a:
    if re.search('^[0-9]*$',i):
        # print(i) # To check the value
        if a.index(i) == string.index(i):
            print(f'Method 1 output is {i}')
        else:
            print('0')

# Method 2
def atoi2(string):
    return int(string)
print('Method 2 output is',atoi2('-91283472332'))

# Method 3
def atoi3(s):
    for i in s.split():
        if i.isdigit():
            out = int(i)
    return out
print(f'Method 3 output is {atoi3("Aneruth has 2 dogs.")}')
```

```
Method 1 output is 4193
Method 2 output is -91283472332
Method 3 output is 2
```

Two Sum

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.

In [6]:
```
aList = [1,2,3]
target = 5
for i in range(len(aList)): # Traverse the list
    for j in range(i+1,len(aList)): # To get the pair for the list
        tot = aList[i] + aList[j] # sum up the pair
        if tot == target: # to check if the pair sum is same as target ele
            print(f'The pair is {[i,j]}') # to print the pair of which the
            print(f'Sum of the pair is {tot}')
```

```
The pair is [1, 2]
Sum of the pair is 5
```

Add Two Numbers

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

In [7]:
```
l1 = [2,4,3]
l2 = [5,6,4]
# This is applicable only for list of same length
def find_sum_of_two_nos(l1,l2):
    st1,st2 = '',''
    for i,j in zip(l1,l2): # zips the two list
        st1 += str(i)
        st2 += str(j)

    # To change it to int
    st1_to_int = int(st1[::-1])
    st2_to_int = int(st2[::-1])
    sum_of_two_nos = st1_to_int + st2_to_int
    sum_lst = [int(q) for q in str(sum_of_two_nos)]
    return sum_lst[::-1]
find_sum_of_two_nos(l1,l2)
```

Out[7]:
```
[7, 0, 8]
```

Longest Substring Without Repeating Characters

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

Example 1:

Input: s = "abcabcbb"

Output: 3

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

Example 2:

Input: s = "bbbbb"

Output: 1

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

Example 3:

Input: s = "pwwkew"

Output: 3

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

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

Example 4:

Input: s = ""

Output: 0

In [8]:
```
def subString(string):
    new_dict = {}
    max_len,idx = 0,0

    for i in range(len(string)):
        if string[i] in new_dict and idx <= new_dict[string[i]]:
            idx = new_dict[string[i]] + 1
        else:
            max_len = max(max_len,i-idx+1)
        new_dict[string[i]] = i

    return max_len

print(subString('abcabcbb'))
print(subString('bbbbb'))
print(subString('pwwkew'))
print(subString(''))
```

```
3
1
3
0
```

sWAP cASE

You are given a string and your task is to swap cases. In other words, convert all lowercase letters to uppercase letters and vice versa.

For Example: Pythonist 2 → pYTHONIST 2

In [9]:
```
s = 'Pyhtonist 2'
s.swapcase()
```

Out[9]:
```
'pYHTONIST 2'
```

String Split and Join

Example:

> a = "this is a string"
>
> a = a.split(" ") # a is converted to a list of strings.
>
> print a

[['this', 'is', 'a', 'string']]

Sample Input

this is a string

Sample Output

this-is-a-string

In [10]:
```
s1 = 'this is a string'
s1 = '-'.join(s1.split(' '))
print(s1)
```

```
this-is-a-string
```

Shift the Index postion of even numbers to n+1th index position.

Test Case: [[0,1,2,3,4,5]] --> [[1.0.3,2,5,4]]

In [11]:
```
qList = [0,1,2,3,4,5]

# Method 1
def odd_even(x):
    odds = sorted(filter(lambda n: n % 2 == 1, x))
    evens = sorted(filter(lambda n: n % 2 == 0, x))
    pairList = zip(odds, evens)
    return [n for t in pairList for n in t]
print('Method 1',odd_even(qList))

# Method 2
eve_lst = qList[0:len(qList):2]
odd_lst = qList[1:len(qList):2]
pair = zip(odd_lst,eve_lst)
print('Method 2',[n for t in pair for n in t])

# Method 3
xz,kd = [],[]
for i in qList:
    xz.append(str(i))
separator = ''
z = separator.join(xz)
ze = z[0:len(z):2]
zo = z[1:len(z):2]
zz = zip(zo,ze)
for i in zz:
    for j in i:
        kd.append(j)
print('Method 3',kd)
```

```
Method 1 [1, 0, 3, 2, 5, 4]
Method 2 [1, 0, 3, 2, 5, 4]
Method 3 ['1', '0', '3', '2', '5', '4']
```

Pair the first element with 4 elemnt from the index postion and subsequent mapps.

Example:

Input : [['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o']]

Output : [[['a', 'e', 'i', 'm']],[['b', 'f', 'j', 'n']],[['c', 'g', 'k', 'o']],[['e', 'i', 'm']]]

In [12]:
```
long_list = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o']
split1 = long_list[0:len(long_list):4]
split2 = long_list[1:len(long_list):4]
split3 = long_list[2:len(long_list):4]
split4 = long_list[4:len(long_list):4]
[split1] + [split2] + [split3] + [split4]
```

Out[12]:
```
[['a', 'e', 'i', 'm'],
 ['b', 'f', 'j', 'n'],
 ['c', 'g', 'k', 'o'],
 ['e', 'i', 'm']]
```

Multiply Strings

Given two non-negative integers num1 and num2 represented as strings, return the product of num1 and num2, also represented as a string.

Note: You must not use any built-in BigInteger library or convert the inputs to integer directly.

Example 1:

Input: num1 = "2", num2 = "3"

Output: "6"

Example 2:

Input: num1 = "123", num2 = "456"

Output: "56088"

In [13]:
```
a = '123'
b = '456'
k = map(int,a)
q = map(int,b)
def mn(a,b):
    for i,j in zip(k,q):
        x1 = i*j
    return (str(x1))
mn(k,q)
```

Out[13]:
```
'18'
```

Palindrome in both Decimal and Binary

Given a number N. check whether a given number N is palindrome or not in it's both formates (Decimal and Binary).

Example 1:

Input: N = 7

Output: "Yes"

Explanation: 7 is palindrome in it's decimal and also in it's binary (111).So answer is "Yes".

Example 2:

Input: N = 12

Output: "No"

Explanation: 12 is not palindrome in it's decimal and also in it's binary (1100).So answer is "No".

In [14]:
```
ds = []

def binaryPali(num):
    if num >= 1:
        bno = binaryPali(num // 2)
        ds.append(num%2)
        # print(num % 2, end = '')
    return ds
# binaryPali(12)

def checkPali(func):
    if func[::-1] == func:
        print('Yes')
    else:
        print('Nope')
checkPali(binaryPali(7))
checkPali(binaryPali(12))
```

```
Yes
Nope
```

Matching Pair

Given a set of numbers from 1 to N, each number is exactly present twice so there are N pairs. In the worst-case scenario, how many numbers X should be picked and removed from the set until we find a matching pair?

Example 1:

Input: N = 1

Output: 2

Explanation: When N=1 Then there is one pair and a matching pair can be extracted in 2 Draws.

Example 2:

Input: N = 2

Output: 3

Explanation: When N=2 then there are 2 pairs, let them be {1,2,1,2} and a matching pair will be made in 3 draws.

In [15]:
```
def findPairs(lst, K):
    res = []
    while lst:
        num = lst.pop()
        diff = K - num
        if diff in lst:
            res.append((diff, num))

    res.reverse()
    return res

# Driver code
lst = [1, 2,1,2]
K = 2
print(findPairs(lst, K))
```

```
[(1, 1)]
```

Remove Duplicates from Sorted Array

Given a sorted array nums, remove the duplicates in-place such that each element appears only once and returns the new length.

Do not allocate extra space for another array, you must do this by modifying the input array in-place with O(1) extra memory.

Clarification:

Confused why the returned value is an integer but your answer is an array?

Note that the input array is passed in by reference, which means a modification to the input array will be known to the caller as well.

Example 1:

Input: nums = [[1,1,2]]

Output: 2, nums = [[1,2]]

Explanation: Your function should return length = 2, with the first two elements of nums being 1 and 2 respectively. It doesn't matter what you leave beyond the returned length.

Example 2:

Input: nums = [[0,0,1,1,1,2,2,3,3,4]]

Output: 5, nums = [[0,1,2,3,4]]

Explanation: Your function should return length = 5, with the first five elements of nums being modified to 0, 1, 2, 3, and 4 respectively. It doesn't matter what values are set beyond the returned length.

In [16]:
```python
# Method 1
def delDups(l):
    l.sort()
    dup = list(set(l))
    return len(dup)
print(f'Method 1 Output:{delDups([0,0,1,1,1,2,2,3,3,4])}')

# Method 2
def removeDuplicate(aList):
    aList.sort()
    i = 0
    for j in range(1,len(aList)):
        if aList[j] != aList[i]:
            i += 1
            aList[i] = aList[j]
    return i + 1
print(f'Method 2 Output:{removeDuplicate([0,0,1,1,1,2,2,3,3,4])}')

# Method 3
def remove_dup(nums):
    nums[:] = sorted(list(set(nums)))
    return len(nums)
print(f'Method 3 Output:{remove_dup([0,0,1,1,1,2,2,3,3,4])}')
```

```
Method 1 Output:5
Method 2 Output:5
Method 3 Output:5
```

Remove Element

Given an array nums and a value val, remove all instances of that value in-place and return the new length.

Do not allocate extra space for another array, you must do this by modifying the input array in-place with O(1) extra memory.

The order of elements can be changed. It doesn't matter what you leave beyond the new length.

Example 1:

Input: nums = [[3,2,2,3]], val = 3

Output: 2, nums = [[2,2]]

Explanation: Your function should return length = 2, with the first two elements of nums being 2. It doesn't matter what you leave beyond the returned length. For example if you return 2 with nums = [[2,2,3,3]] or nums = [[2,2,0,0]], your answer will be accepted.

Example 2:

Input: nums = [[0,1,2,2,3,0,4,2]], val = 2

Output: 5, nums = [[0,1,4,0,3]]

Explanation: Your function should return length = 5, with the first five elements of nums containing 0, 1, 3, 0, and 4. Note that the order of those five elements can be arbitrary. It doesn't matter what values are set beyond the returned length.

In [17]:
```
# Method 1
def xuz(aList,val):
    pit = [value for value in aList if value != val]
    return len(pit)

print(f'Output from method 1: {xuz([0,1,2,2,3,0,4,2],2)}')

# Method 2
def removeElement(nums, val):
    output = 0
    for j in range(len(nums)):
            if nums[j] != val:
                    nums[output] = nums[j]
                    output += 1
    return output
print(f'Output from method 2: {removeElement([0,1,2,2,3,0,4,2],2)}')
```

```
Output from method 1: 5
Output from method 2: 5
```

Yet to see this

Roman to Integer

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol Value

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

For example, 2 is written as II in Roman numeral, just two one's added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

I can be placed before V (5) and X (10) to make 4 and 9.

X can be placed before L (50) and C (100) to make 40 and 90.

C can be placed before D (500) and M (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

Example 1:

Input: s = "III"

Output: 3

Example 2:

Input: s = "IV"

Output: 4

Example 3:

Input: s = "IX"

Output: 9

Example 4:

Input: s = "LVIII"

Output: 58

Explanation: L = 50, V= 5, III = 3.

Example 5:

Input: s = "MCMXCIV"

Output: 1994

Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

In [18]:
```
def roman_to_int(s):
    rom_val = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 
    int_val = 0
    for i in range(len(s)):
        if i > 0 and rom_val[s[i]] > rom_val[s[i - 1]]:
            int_val += rom_val[s[i]] - 2 * rom_val[s[i - 1]]
        else:
            int_val += rom_val[s[i]]
    return int_val

roman_to_int('XX')
```

Out[18]: 20

Yet to see this

Minimum Index Sum of Two Lists

Suppose Andy and Doris want to choose a restaurant for dinner, and they both have a list of favorite restaurants represented by strings.

You need to help them find out their common interest with the least list index sum. If there is a choice tie between answers, output all of them with no order requirement. You could assume there always exists an answer.

Example 1:

Input: list1 = [["Shogun","Tapioca Express","Burger King","KFC"]], list2 = [["Piatti","The Grill at Torrey Pines","Hungry Hunter Steakhouse","Shogun"]]

Output: [["Shogun"]]

Explanation: The only restaurant they both like is "Shogun".

Example 2:

Input: list1 = [["Shogun","Tapioca Express","Burger King","KFC"]], list2 = [["KFC","Shogun","Burger King"]]

Output: [["Shogun"]]

Explanation: The restaurant they both like and have the least index sum is "Shogun" with index sum 1 (0+1).

Example 3:

Input: list1 = [["Shogun","Tapioca Express","Burger King","KFC"]], list2 = [["KFC","Burger King","Tapioca Express","Shogun"]]

Output: [["KFC","Burger King","Tapioca Express","Shogun"]]

Example 4:

Input: list1 = [["Shogun","Tapioca Express","Burger King","KFC"]], list2 = [["KNN","KFC","Burger King","Tapioca Express","Shogun"]]

Output: [["KFC","Burger King","Tapioca Express","Shogun"]]

Example 5:

Input: list1 = [["KFC"]], list2 = [["KFC"]]

Output: [["KFC"]]

In [19]:
```
aL = ["Shogun","Tapioca Express","Burger King","KFC"]
bL = ["KFC","Shogun","Burger King"]
xx = []
for i in aL:
    if i in bL:
        if i not in xx:
            xx.append(i)
    else:
        for j in bL:
            # if aL.index(i) == bL.index(j):
            #     x.append(i)
            if list(set(i)) == list(set(j)):
                xx.append(j)
print(xx)
```

```
['Shogun', 'Burger King', 'KFC']
```

Search Insert Position

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

Example 1:

Input: nums = [[1,3,5,6]], target = 5

Output: 2

Example 2:

Input: nums = [[1,3,5,6]], target = 2

Output: 1

Example 3:

Input: nums = [[1,3,5,6]], target = 7

Output: 4

Example 4:

Input: nums = [[1,3,5,6]], target = 0

Output: 0

Example 5:

Input: nums = [[1]], target = 0

Output: 0

In [20]:
```
def lookup(lt,tar):
    lt.sort()
    if tar in lt:
        print(lt.index(tar))
    elif tar == 0 and tar not in lt:
        print('0')
    elif tar not in lt:
        b = lt.__add__([tar])
        b.sort()
        print(b.index(tar))
lookup([1,3,5,6],5)
```

```
2
```

Yet to see this

Plus One

Given a non-empty array of decimal digits representing a non-negative integer, increment one to the integer.

The digits are stored such that the most significant digit is at the head of the list, and each element in the array contains a single digit.

You may assume the integer does not contain any leading zero, except the number 0 itself.

Example 1:

Input: digits = [[1,2,3]]

Output: [[1,2,4]]

Explanation: The array represents the integer 123.

Example 2:

Input: digits = [[4,3,2,1]]

Output: [[4,3,2,2]]

Explanation: The array represents the integer 4321.

Example 3:

Input: digits = [[0]]

Output: [[1]]

In [21]:
```
# Method 1
def plusOne(digits):
    digits[-1] = digits[-1] + 1
    return digits

# Method 2
dk = []
def inc_last(lst):
    lst[-1] = lst[-1] + 1
    return [int(x) if x.isdigit() else x for z in lst for x in str(z)]
print(f'Output from method 1 {plusOne([9])}')
print(f'Output from method 2 {inc_last([9,9])}')
```

```
Output from method 1 [10]
Output from method 2 [9, 1, 0]
```

Diagonal Traverse

Given a matrix of M x N elements (M rows, N columns), return all elements of the matrix in diagonal order.

In [22]:
```
def mat_traverse(mat):
    if not mat or not mat[0]:
        return []
    rows,cols = len(mat),len(mat[0])

    diag = [[] for _ in range(rows + cols - 1)]

    for i in range(rows):
        for j in range(cols):
            diag[i + j].append(mat[i][j])

    res = diag[0] # Since the first element starts with first value.

    for i in range(1,len(diag)):
        if i % 2 == 1:
            res.extend(diag[i])
        else:
            res.extend(diag[i][::-1])

    return res

mat_traverse([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
```

Out[22]:
```
[1, 2, 5, 9, 6, 3, 4, 7, 10, 11, 8, 12]
```

Yet to see this

Power of Two

Given an integer n, return true if it is a power of two. Otherwise, return false.

An integer n is a power of two, if there exists an integer x such that n == pow(2,x).

Example 1:

Input: n = 1

Output: true

Explanation: 2 ** 0 = 1

Example 2:

Input: n = 16

Output: true

Explanation: 2 ** 4 = 16

Example 3:

Input: n = 3

Output: false

Example 4:

Input: n = 4

Output: true

Example 5:

Input: n = 5

Output: false

In [23]:
```
def power_two(num):
    inp,out = 1,1
    while inp <= num:
        if inp == num:
            return True
        inp = 2 ** out
        out += 1
    return False

power_two(4)
```

Out[23]: True

Median of Two Sorted Arrays

Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays.

Follow up: The overall run time complexity should be O(log (m+n)).

Example 1:

Input: nums1 = [[1,3]], nums2 = [[2]]

Output: 2.00000

Explanation: merged array = [[1,2,3]] and median is 2.

Example 2:

Input: nums1 = [[1,2]], nums2 = [[3,4]]

Output: 2.50000

Explanation: merged array = [[1,2,3,4]] and median is (2 + 3) / 2 = 2.5.

Example 3:

Input: nums1 = [[0,0]], nums2 = [[0,0]]

Output: 0.00000

Example 4:

Input: nums1 = [], nums2 = [[1]]

Output: 1.00000

Example 5:

Input: nums1 = [[2]], nums2 = []

Output: 2.00000

In [24]:
```
# Method 1

def median_sorted(l1,l2):
    l1.extend(l2)
    l1.sort()
    if len(l1) % 2 ==0:
        median = (l1[len(l1) // 2 - 1] + l1[len(l1) // 2 ]) // 2.0
        print(f'Median from method 1: {median}')
    else:
        median = l1[len(l1) // 2]
        print(f'Median from method 1: {median}')
median_sorted([1,2],[3,4])

# Method 2
import numpy as np
def findMedianSortedArrays(nums1,nums2):
        mergeList = sorted(nums1 + nums2)
        MedianOfSortedArray = np.median(mergeList)
        return MedianOfSortedArray
print(f'Output from method 2: {findMedianSortedArrays([1,2],[3,4])}')

# Method 3
def med(x,y):
    x.extend(y)
    if len(x) == 1:
            median = float(x[0])
    elif len(x) % 2 != 0:
        median = float(x[len(x)//2])
    else:
        median = (x[len(x)//2]+x[(len(x)//2)-1])/2
    return median
print(f'Output from method 3: {med([1,2],[3,4])}')
```

```
Median from method 1: 2.0
Output from method 2: 2.5
Output from method 3: 2.5
```

In [25]:
```
al = [1,2,3]
al.extend([4,5])
al
```

Out[25]:
```
[1, 2, 3, 4, 5]
```

Length of Last Word

Given a string s consists of some words separated by spaces, return the length of the last word in the string. If the last word does not exist, return 0.

A word is a maximal substring consisting of non-space characters only.

Example 1:

Input: s = "Hello World"

Output: 5

Example 2:

Input: s = " "

Output: 0

In [26]:
```
def len_of_last(s) -> int:
    dum = s.split()
    if dum:
        print(len(dum[-1]))
    elif " ":
        print(0)
    else:
        print(len(s))
len_of_last(" ")
```

```
0
```

YET TO SEE THIS

First Bad Version

You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.

Suppose you have n versions [[1, 2, ..., n]] and you want to find out the first bad one, which causes all the following ones to be bad.

You are given an API bool isBadVersion(version) which returns whether version is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

Example 1:

Input: n = 5, bad = 4

Output: 4

Explanation:

call isBadVersion(3) -> false

call isBadVersion(5) -> true

call isBadVersion(4) -> true

Then 4 is the first bad version.

Example 2:

Input: n = 1, bad = 1

Output: 1

#

Search in Rotated Sorted Array

You are given an integer array nums sorted in ascending order (with distinct values), and an integer target.

Suppose that nums is rotated at some pivot unknown to you beforehand (i.e., [[0,1,2,4,5,6,7]] might become [[4,5,6,7,0,1,2]]).

If target is found in the array return its index, otherwise, return -1.

Example 1:

Input: nums = [[4,5,6,7,0,1,2]], target = 0

Output: 4

Example 2:

Input: nums = [[4,5,6,7,0,1,2]], target = 3

Output: -1

Example 3:

Input: nums = [[1]], target = 0

Output: -1

In [27]:
```
nums = [4,5,6,7,0,1,2]
target = 0
if target in nums:
    idx = nums.index(target)
    print(idx)
else:
    idx = -1
    print(idx)
```

```
4
```

Yet to see this

Container With Most Water

Given n non-negative integers a_1, a_2, ..., a_n , where each represents a point at coordinate (i, a_i). n vertical lines are drawn such that the two endpoints of the line i is at (i, a_i) and (i, 0). Find two lines, which, together with the x-axis forms a container, such that the container contains the most water.

Notice that you may not slant the container.

Example 1:

Input: height = [[1,8,6,2,5,4,8,3,7]]

Output: 49

Explanation: The above vertical lines are represented by array [[1,8,6,2,5,4,8,3,7]]. In this case, the max area of water (blue section) the container can contain is 49.

Example 2:

Input: height = [[1,1]]

Output: 1

Example 3:

Input: height = [[4,3,2,1,4]]

Output: 16

Example 4:

Input: height = [[1,2,1]]

Output: 2

In [28]:
```
# def maxArea(heights):
#     if len(heights) > 3:
#         max_value = max(heights)
#         sliced_list = heights[heights.index(max_value) + 1:]
#         out = len(sliced_list) ** 2

#     else:
#         min_list_val = max(heights)
#         sliced_list_val = heights[heights.index(min_list_val):]
#         # out =
#     return min_list_val
# maxArea([1,2,1])
  def maxArea(height):
    left, right = 0, len(height)-1
    dummy_value = 0

    while left < right:
        dummy_value = max(dummy_value, (right-left) * min(height[left], hei

        if height[left] < height[right]:
            left+=1
        else:
            right-=1

    return dummy_value
maxArea([1,1])
```

```
  File "<ipython-input-28-ea3f74861c91>", line 13
    def maxArea(height):
    ^
IndentationError: unexpected indent
```

3Sum

Given an array nums of n integers, are there elements a, b, c in nums such that a + b + c = 0? Find all unique triplets in the array which gives the sum of zero.

Notice that the solution set must not contain duplicate triplets.

Example 1:

Input: nums = [[-1,0,1,2,-1,-4]]

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

Example 2:

Input: nums = []

Output: []

Example 3:

Input: nums = [[0]]

Output: []

In [26]:
```
al = [-1,0,1,2,-1,-4]
lte = []
def three_sum(l):
    al.sort()
    xc = list(set(al))
    for i in range(len(xc)):
        for j in range(i-1,len(xc)):
            for k in range(j+1,len(xc)):
                tot = i + xc[j] + xc[k]
                if tot == 0:
                    lte.append([i,xc[j],xc[k]])
                elif len(al) <= 3:
                    return []
    return lte
three_sum(al)
```

Out[26]:
```
[[0, -1, 1], [0, 1, -1], [1, 0, -1], [2, 2, -4]]
```

Sqrt(x)

Given a non-negative integer x, compute and return the square root of x.

Since the return type is an integer, the decimal digits are truncated, and only the integer part of the result is returned.

Example 1:

Input: x = 4

Output: 2

Example 2:

Input: x = 8

Output: 2

Explanation: The square root of 8 is 2.82842..., and since the decimal part is truncated, 2 is returned.

In [27]:
```
def sqrt(x):
    return int(pow(x,1/2))
sqrt(8)
```

Out[27]: 2

Climbing Stairs

You are climbing a staircase. It takes n steps to reach the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Example 1:

Input: n = 2

Output: 2

Explanation: There are two ways to climb to the top.

1. 1 step + 1 step
2. 2 steps

Example 2:

Input: n = 3

Output: 3

Explanation: There are three ways to climb to the top.

1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step

In [73]:
```
# Method 1 recursive
def climb(x):
    if x == 1 or x ==2 or x == 3:
        z = x
    else:
        z = climb(x-1) + climb(x-2)
    return z
print(f'Output for method 1: {climb(1)}')

# Method 2
def climbStairs(n):
    if n == 1: out = 1
    first,second = 1,2
    for i in range(3,n+1):
        third = first + second
        first,second = second,third
        out = second
    return out
print(f'Output for method 2: {climbStairs(1)}')
```

```
Output for method 1: 1
Output for method 2: 1
```

Merge Sorted Array

Given two sorted integer arrays nums1 and nums2, merge nums2 into nums1 as one sorted array.

The number of elements initialized in nums1 and nums2 are m and n respectively. You may assume that nums1 has a size equal to m + n such that it has enough space to hold additional elements from nums2.

Example 1:

Input: nums1 = [[1,2,3,0,0,0]], m = 3, nums2 = [[2,5,6]], n = 3

Output: [[1,2,2,3,5,6]]

Example 2:

Input: nums1 = [[1]], m = 1, nums2 = [], n = 0

Output: [[1]]

In [29]:
```
# Method 1
def merge_sorted(n1,n2,m,n):
    n3 = n1[:m]
    n3.extend(n2[:n])
    n3.sort()
    return n3
print(f'Output from method 1: {merge_sorted([1,2,3,0,0,0],[2,5,6],3,3)}')

# Method 2
def m(n1,n2,m,n):
    if n == 0: return n1
    n1[-(len(n2)):] = n2
    n1.sort()
    return n1
print(f'Output from methid 2: {m([1,2,3,0,0,0],[2,5,6],3,3)}')
```

```
Output from method 1: [1, 2, 2, 3, 5, 6]
Output from methid 2: [1, 2, 2, 3, 5, 6]
```

Pascal's Triangle

Given a non-negative integer numRows, generate the first numRows of Pascal's triangle.
Example:

Input: 5 Output: [

[[1]],

[[1,1]],

[[1,2,1]],

[[1,3,3,1]],

[[1,4,6,4,1]]

]

In [30]:
```
def pascal(n):
    cs = []
    for i in range(1,n + 1):
        c = 1
        for j in range(1, i+1):
            # print(c,end=' ')
            c = int(c * (i-j)/j)
            cs.append(c)
    return cs
pascal(5)
```

Out[30]:
```
[0, 1, 0, 2, 1, 0, 3, 3, 1, 0, 4, 6, 4, 1, 0]
```

In [31]:
```
# Pascal function
def printPascal(n):

        for line in range(1, n + 1):
                C = 1
                for i in range(1, line + 1):
                        # The first value in a
                        # line is always 1
                        print(C, end = " ")
                        C = int(C * (line - i) / i)
                print("")

# Driver code
n = 5
printPascal(n)
```

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Valid Palindrome

Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

Note: For the purpose of this problem, we define empty string as valid palindrome.

Example 1:

Input: "A man, a plan, a canal: Panama"

Output: true

Example 2:

Input: "race a car"

Output: false

In [30]:
```
def valid_palindrome(s):
    temp = "".join([i.lower() for i in s if i.isalnum()])
    return temp[::-1] == temp
print(f'Method 1 output: {valid_palindrome("race a car")}')
```

```
Method 1 output: False
```

Single Number

Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.

Follow up: Could you implement a solution with a linear runtime complexity and without using extra memory?

Example 1:

Input: nums = [[2,2,1]]

Output: 1

Example 2:

Input: nums = [[4,1,2,1,2]]

Output: 4

Example 3:

Input: nums = [[1]]

Output: 1

In [33]:
```
# Method 1
def single_num(lt):
    dup = []
    for i in lt:
        if i not in dup:
            dup.append(i)
        else:
            dup.remove(i)
    return dup.pop()

print(single_num([2,2,1]))

# Method 2
def single_num2(lt):
    return 2 * sum(set(lt)) - sum(lt) # Considers the set of the give list
print(single_num2([2,2,1]))
```

```
1
1
```

Single Number II

Given an integer array nums where every element appears three times except for one, which appears exactly once. Find the single element and return it.

Example 1:

Input: nums = [[2,2,3,2]] | Output: 3

Example 2:

Input: nums = [[0,1,0,1,0,1,99]] | Output: 99

In [124…

```
# Method 1
def singleNumber_2(nums):
    for i in nums:
            if nums.count(i) == 1:
                    output = i
    return output
print(f'Method 1 output: {singleNumber_2([2,2,1])}')
```

```
Method 1 output: 1
```

Two Sum II - Input array is sorted

Given an array of integers numbers that is already sorted in ascending order, find two numbers such that they add up to a specific target number.

Return the indices of the two numbers (1-indexed) as an integer array answer of size 2, where 1 <= answer[0] < answer[1] <= numbers.length.

You may assume that each input would have exactly one solution and you may not use the same element twice.

Example 1:

Input: numbers = [[2,7,11,15]], target = 9

Output: [[1,2]]

Explanation: The sum of 2 and 7 is 9. Therefore index1 = 1, index2 = 2.

Example 2:

Input: numbers = [[2,3,4]], target = 6

Output: [[1,3]]

Example 3:

Input: numbers = [[-1,0]], target = -1

Output: [[1,2]]

In [34]:
```
def two_sumII(lt,tar):
    for i in range(len(lt)):
        for j in range(i+1,len(lt)):
            tot = lt[i] + lt[j]
            if tar == tot:
                print([i+1,j+1])
                # print(c)
two_sumII([2,7,11,15],9) # Solves 18/19 cases in leet code.
```

```
[1, 2]
```

Yet to see this

XOR Operation in an Array

Given an integer n and an integer start.

Define an array nums where nums[[i]] = start + 2*i (0-indexed) and n == nums.length.

Return the bitwise XOR of all elements of nums.

Example 1:

Input: n = 5, start = 0

Output: 8

Explanation: Array nums is equal to [[0, 2, 4, 6, 8]] where (0 ^ 2 ^ 4 ^ 6 ^ 8) = 8.

Where "^" corresponds to bitwise XOR operator.

Example 2:

Input: n = 4, start = 3

Output: 8

Explanation: Array nums is equal to [[3, 5, 7, 9]] where (3 ^ 5 ^ 7 ^ 9) = 8.

Example 3:

Input: n = 1, start = 7

Output: 7

Example 4:

Input: n = 10, start = 5

Output: 2

In [35]:
```
def xor(n,start):
    pointer = 0
    for i in range(n):
        pointer ^= (start + 2*i)
    return pointer
xor(5,0)
```

Out[35]: 8

Buddy Strings

Given two strings A and B of lowercase letters, return true if you can swap two letters in A so the result is equal to B, otherwise, return false.

Swapping letters is defined as taking two indices i and j (0-indexed) such that i != j and swapping the characters at A[[i]] and A[[j]]. For example, swapping at indices 0 and 2 in "abcd" results in "cbad".

Example 1:

Input: A = "ab", B = "ba" Output: true

Explanation: You can swap A[[0]] = 'a' and A[[1]] = 'b' to get "ba", which is equal to B.

Example 2:

Input: A = "ab", B = "ab" Output: false

Explanation: The only letters you can swap are A[[0]] = 'a' and A[[1]] = 'b', which results in "ba" != B.

Example 3:

Input: A = "aa", B = "aa" Output: true

Explanation: You can swap A[[0]] = 'a' and A[[1]] = 'a' to get "aa", which is equal to B.

Example 4:

Input: A = "aaaaaaabc", B = "aaaaaaacb" Output: true

Example 5:

Input: A = "", B = "aa" Output: false

In [36]:
```
# Method 1 (20/29 Test case pass)
def buddy(A,B):
    if A[::-1] == B:
        x = True
    else:
        x = False
    return x
# print(buddy('abab','abab'))

# Method 2 (19/29 Test case pass)
def l(a,b):
    # Case 1
    if ((len(a) and len(b)) >= 4) and ((len(a) or len(b)) < 20000):
        z = []
        x = ''
        w = int(len(a)/2)
        a1 = a[:w:]
        a2 = a[len(a1):]
        for i in a2:
            z.append(i)
        eve = z[0:len(z):2]
        odd = z[1:len(z)]
        for j in zip(eve,odd):
            for k in j:
                x += k
                y = a1 + x

        if y == b:
            out = True
        else:
            out = False

    # Case 2
    elif (a == "") and (b == ""):
        out = False

    # Case 3
    else:
        if a[::-1] == b:
            out = True
        else:
            out = False
    return out
l("abab","abab")
```

Out[36]: True

Squares of a Sorted Array

Given an integer array nums sorted in non-decreasing order, return an array of the squares of each number sorted in non-decreasing order.

Example 1:

Input: nums = [[-4,-1,0,3,10]] Output: [[0,1,9,16,100]]

Explanation: After squaring, the array becomes [[16,1,0,9,100]]. After sorting, it becomes [[0,1,9,16,100]].

Example 2:

Input: nums = [[-7,-3,2,3,11]] Output: [[4,9,9,49,121]]

In [37]:
```
# Method 1
def sq_array(s):
    x = [i ** 2 for i in s]
    x.sort()
    return x
print(f'Method 1 output: {sq_array([-4,-1,0,3,10])}')

# Method 2
def sq2_array(s):
    return sorted([i*i for i in s])
print(f'Method 2 output: {sq2_array([-4,-1,0,3,10])}')
```

```
Method 1 output: [0, 1, 9, 16, 100]
Method 2 output: [0, 1, 9, 16, 100]
```

Number of 1 Bits

Write a function that takes an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight).

Note:

Note that in some languages such as Java, there is no unsigned integer type. In this case, the input will be given as a signed integer type. It should not affect your implementation, as the integer's internal binary representation is the same, whether it is signed or unsigned. In Java, the compiler represents the signed integers using 2's complement notation. Therefore, in Example 3, the input represents the signed integer. -3.

Example 1:

Input: n = 00000000000000000000000000001011 ;Output: 3

Explanation: The input binary string 00000000000000000000000000001011 has a total of three '1' bits.

Example 2:

Input: n = 00000000000000000000000010000000 ;Output: 1

Explanation: The input binary string 00000000000000000000000010000000 has a total of one '1' bit.

Example 3:

Input: n = 11111111111111111111111111111101 ;Output: 31

Explanation: The input binary string 11111111111111111111111111111101 has a total of thirty one '1' bits.

In [38]:
```
def count1(s):
    cnt = 0
    for i in s: # In leet code IDE we need to change the s as bin(s) to co
        if '1' in i:
            cnt += 1
    return cnt
count1('00000000000000000000000000001011')
```

Out[38]: 3

Count Primes

Count the number of prime numbers less than a non-negative number, n.

Example 1:

Input: n = 10 Output: 4

Explanation: There are 4 prime numbers less than 10, they are 2, 3, 5, 7.

Example 2:

Input: n = 0 Output: 0

Example 3:

Input: n = 1 Output: 0

In [39]:
```
# def prime_count(n): # (Leet code 17/20 test case passed)
#     zx  = []
#     for num in range(2,n):
#         if all(num%i!=0 for i in range(2,num)):
#             zx.append(num)
#     return len(zx)
# prime_count(10)
```

Out[39]: 4

In [42]:
```
def count_prime(n):
    D = {}
    q = 2
    while q <= limit:
        if q not in D:
            yield q
            D[q * q] = [q]
        else:
            for p in D[q]:
                D.setdefault(p + q, []).append(p)
            del D[q]
        q += 1
    x = [j for j in q]
    return x
count_prime(10)
```

Out[42]: <generator object count_prime at 0x7fcf7bd20900>

Number of Segments in a String

You are given a string s, return the number of segments in the string.

A segment is defined to be a contiguous sequence of non-space characters.

Example 1:

Input: s = "Hello, my name is John"

Output: 5

Explanation: The five segments are [["Hello,", "my", "name", "is", "John"]]

Example 2:

Input: s = "Hello"

Output: 1

Example 3:

Input: s = "love live! mu'sic forever"

Output: 4

Example 4:

Input: s = ""

Output: 0

In [40]:
```
def segment(s):
    return len(s.split())
segment('Aneruth Mohanasundaram')
```

Out[40]: 2

Power of Three

Given an integer n, return true if it is a power of three. Otherwise, return false.

An integer n is a power of three, if there exists an integer x such that n == 3^x.

Example 1:

Input: n = 27

Output: true

Example 2:

Input: n = 0

Output: false

Example 3:

Input: n = 9

Output: true

Example 4:

Input: n = 45

Output: false

In [41]:
```
# Method 1 (Leet code 14059 / 21038 test cases passed.)
def power_three(x):
    if x > 1:
        if x % 3 == 0:
            out = True
    else:
        out = False

    return out
print(f'Method 1 output: {power_three(45)}')

# Method 2
from math import *
def isPowerOfThree(n):

    if (log10(n) / log10(3)) % 1 == 0:
            out = True
    else:
            out = False
    return  out

print(f'Method 2 output: {isPowerOfThree(125)}')
```

```
Method 1 output: True
Method 2 output: False
```

Yet to see this

Largest Rectangle in Histogram

Given n non-negative integers representing the histogram's bar height where the width of each bar is 1, find the area of largest rectangle in the histogram.

Example 1:

Input: heights = [[2,1,5,6,2,3]]

Output: 10

Example 2:

Input: heights = [[2,4]]

Output: 4

In [42]:
```
h = [0,9]
h.sort()
if len(h) > 1:
    first_largest,second_largest = h[-1],h[-2]
    third = first_largest - second_largest
    out = second_largest + (first_largest - third)
else:
    out = len(h)
out
```

Out[42]: 0

Binary Search

Given a sorted (in ascending order) integer array nums of n elements and a target value, write a function to search target in nums. If target exists, then return its index, otherwise return -1.

Example 1:

Input: nums = [[-1,0,3,5,9,12]], target = 9 Output: 4 Explanation: 9 exists in nums and its index is 4

Example 2:

Input: nums = [[-1,0,3,5,9,12]], target = 2 Output: -1 Explanation: 2 does not exist in nums so return -1

In [43]:
```
num12 = [-1,2,4,7,9]
target = 9
if target in num12:
    out = num12.index(target)
out
```

Out[43]: 4

Add Digits

Given a non-negative integer num, repeatedly add all its digits until the result has only one digit.

Example:

Input: 38

Output: 2

Explanation: The process is like: 3 + 8 = 11, 1 + 1 = 2. Since 2 has only one digit, return it.

In [44]:
```
# Method 1
def add_digi(n):
    x = [int(i) for i in str(n)]
    sum_x = sum(x)
    x1 = [int(i) for i in str(sum_x)]
    sum_x1 = sum(x1)
    zx = [int(i) for i in str(sum_x1)]
    return sum(zx)
print(f'Value from method 1 is {add_digi(199)}')

# Method 2
def addDigits(num: int):
    if num == 0:
        return 0
    if num % 9 == 0:
        return 9
    return num % 9
print(f'Value from method 2 is {addDigits(199)}')
```

```
Value from method 1 is 1
Value from method 2 is 1
```

Yet to see this

Super Pow

Your task is to calculate a^b mod 1337 where a is a positive integer and b is an extremely large positive integer given in the form of an array.

Example 1:

Input: a = 2, b = [[3]]

Output: 8

Example 2:

Input: a = 2, b = [[1,0]]

Output: 1024

Example 3:

Input: a = 1, b = [[4,3,3,8,5,2]]

Output: 1

Example 4:

Input: a = 2147483647, b = [[2,0,0]]

Output: 1198

In [45]:
```
# def super_pow(b:int,n:list):
#     x = [str(i) for i in n]
#     ax = ''.join(x)
#     ax_int = int(ax)
#     a = pow(b,ax_int)
#     if a > 100000:
#         out = a % 1337
#     else:
#         out = pow(b,ax_int)
#     return int(out)
# super_pow(2147483647,[2,0,0])
```

Transpose Matrix

Given a 2D integer array matrix, return the transpose of matrix.

The transpose of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.

Example 1:

Input: matrix = [[[1,2,3]],[[4,5,6]],[[7,8,9]]]

Output: [[[1,4,7]],[[2,5,8]],[[3,6,9]]]

Example 2:

Input: matrix = [[[1,2,3]],[[4,5,6]]]

Output: [[[1,4]],[[2,5]],[[3,6]]]

In [46]:
```
# Method 1 Using numpy
import numpy as np
def transpose(matrix):
    return np.transpose(matrix)
print('Output from method 1' + '\n' + f'{transpose([[1,2,3],[4,5,6],[7,8,9

# Method 2
def transpose2(matrix):
    zx = zip(*matrix)
    x = [i for i in list(zx)]
    return x
print('Output from method 2' + '\n' + f'{transpose2([[1,2,3],[4,5,6],[7,8,9
```

```
Output from method 1
[[1 4 7]
 [2 5 8]
 [3 6 9]]
Output from method 2
[(1, 4, 7), (2, 5, 8), (3, 6, 9)]
```

Reshape the Matrix

In MATLAB, there is a very useful function called 'reshape', which can reshape a matrix into a new one with different size but keep its original data.

You're given a matrix represented by a two-dimensional array, and two positive integers r and c representing the row number and column number of the wanted reshaped matrix, respectively.

The reshaped matrix need to be filled with all the elements of the original matrix in the same row-traversing order as they were.

If the 'reshape' operation with given parameters is possible and legal, output the new reshaped matrix; Otherwise, output the original matrix.

Example 1:

Input: nums = [[[1,2]], [[3,4]]] r = 1, c = 4

Output: [[1,2,3,4]]

Explanation: The row-traversing of nums is [[1,2,3,4]]. The new reshaped matrix is a 1 * 4 matrix, fill it row by row by using the previous list. Example 2:

Input: nums = [[[1,2]], [[3,4]]] r = 2, c = 4

Output: [[[1,2]], [[3,4]]]

Explanation: There is no way to reshape a 2 *2 matrix to a 2* 4 matrix. So output the original matrix.

In [47]:
```
# Method 1 using numpy
def matrix_reshape(mat,r,c):
    return np.reshape(mat,(r,c))
matrix_reshape([[1,2], [3,4]],1,4)
```

Out[47]:
```
array([[1, 2, 3, 4]])
```

Array Partition I

Given an integer array nums of 2n integers, group these integers into n pairs (a_1, b_1), (a_2, b_2), ..., (a_n, b_n) such that the sum of min(a_i,b_i) for all i is maximized. Return the maximized sum.

Example 1:

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

Explanation: All possible pairings (ignoring the ordering of elements) are:

1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3
2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3
3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4

So the maximum possible sum is 4.

Example 2:

Input: nums = [[6,2,6,5,1,2]]

Output: 9

Explanation: The optimal pairing is (2, 1), (2, 5), (6, 6). min(2, 1) + min(2, 5) + min(6, 6) = 1 + 2 + 6 = 9.

In [48]:
```
def ary_part(n):
    n.sort()
    return sum(n[::2])
ary_part([6,2,6,5,1,2])
```

Out[48]: 9

Student Attendance Record I

You are given a string representing an attendance record for a student. The record only contains the following three characters:

1. 'A' : Absent.
2. 'L' : Late.
3. 'P' : Present.

A student could be rewarded if his attendance record doesn't contain more than one 'A' (absent) or more than two continuous 'L' (late).

You need to return whether the student could be rewarded according to his attendance record.

Example 1:

Input: "PPALLP"

Output: True

Example 2:

Input: "PPALLL"

Output: False

In [49]:
```
def check_record(s): # (Leet Code 65 / 113 test cases passed.)
    c = s.strip()
    if c.count('A') > 1 or ('LLL') in s:
        out = False
    else:
        out = True
    return out
check_record("PPALLP")
```

Out[49]: True

Yet to see this

Longest Common Prefix

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

Example 1:

Input: strs = [["flower","flow","flight"]]

Output: "fl"

Example 2:

Input: strs = [["dog","racecar","car"]]

Output: ""

Explanation: There is no common prefix among the input strings.

In [50]:
```
s = ["flower","flow","flight"]
def foo(s):
    for i in range(len(s[0])):
        # print(s[0][i])
        c = s[0][i]
        for j in range(1,len(s)):
            if i == len(s[j]) or s[j][i] != c:
                return s[0][:i]
    return s[0]
foo(s)
```

Out[50]: 'fl'

Factorial Trailing Zeroes

Given an integer n, return the number of trailing zeroes in n!.

Follow up: Could you write a solution that works in logarithmic time complexity?

Example 1:

Input: n = 3

Output: 0

Explanation: 3! = 6, no trailing zero.

Example 2:

Input: n = 5

Output: 1

Explanation: 5! = 120, one trailing zero.

Example 3:

Input: n = 0

Output: 0

In [51]:
```
def trailingZeroes(n: int) -> int:
    fact = 1
    out = 0
    if n <= 0:
            out
    else:
            for i in range(1,n+1):
                    fact = fact * i

            for j in str(fact):
                    if '0' in j:
                            out += 1
                    else:
                            out = 0

    return out
trailingZeroes(10)
```

Out[51]: 2

Number Complement

Given a positive integer num, output its complement number. The complement strategy is to flip the bits of its binary representation.

Example 1:

Input: num = 5

Output: 2

Explanation: The binary representation of 5 is 101 (no leading zero bits), and its complement is 010. So you need to output 2.

Example 2:

Input: num = 1

Output: 0

Explanation: The binary representation of 1 is 1 (no leading zero bits), and its complement is 0. So you need to output 0.

In [52]:
```
def complement(num):
    z = bin(num).replace('0b','')
    x = int('1'*len(s),2) - num
    return int(z,2)
complement(5)
```

Out[52]: 5

Yet to see this

Subsets

Given an integer array nums of unique elements, return all possible subsets (the power set).

The solution set must not contain duplicate subsets. Return the solution in any order.

Example 1:

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

Input: nums = [[0]] Output: [[],[[0]]]

In [69]:
```
def powerset(s):
    x = len(s)
    ax = []
    for i in range(1 << x):
        out = [s[j] for j in range(x) if (i & (1 << j))] # Compares each e
        ax.append(out)
    return ax
powerset([4])
```

Out[69]:
```
[[], [4]]
```

First Missing Positive

Given an unsorted integer array nums, find the smallest missing positive integer.

Example 1:

Input: nums = [[1,2,0]] | Output: 3

Example 2:

Input: nums = [[3,4,-1,1]] | Output: 2

Example 3:

Input: nums = [[7,8,9,11,12]] | Output: 1

In [54]:
```
# Method 1
def missing_element(nums):

    if not nums:
        out = 0
    max_element = max(nums)
    value = []
    for j in range(1,max_element):
        if j not in nums:
            value.append(j)
            out = value[0]
        elif len(nums) <= 3:
            out = max_element + 1
    return out
print(f'Output for method 1: {missing_element([1,4])}')


# Method 2
def missing_element2(nu):
    max_element = max(nu)
    empty = ''
    for j in range(1,max_element):
        if j not in nu:
            empty = j
            break
    if empty == '':
        empty = max_element + 1
    elif not nu:
        empty = 0
    return empty
print(f'Output for method 2: {missing_element2([1,4])}')

# Method 3
def missing_element3(nu):
    max_e = None
    if len(nu) == 0:
        empty = 0
    for i in nu:
        if (max_e is None or i > max_e): max_e = i
    empty = ''
    for j in range(1,max_e):
        if j not in nu:
            empty = j
            break
    if empty == '':
        empty = max_e + 1
    return empty
print(f'Output for method 3: {missing_element3([1,4])}')
```

```
Output for method 1: 2
Output for method 2: 2
Output for method 3: 2
```

Yet to see this

Edit Distance

Given two strings word1 and word2, return the minimum number of operations required to convert word1 to word2.

You have the following three operations permitted on a word:

1.Insert a character

2.Delete a character

3.Replace a character

Example 1:

Input: word1 = "horse", word2 = "ros" | Output: 3

Explanation:

horse -> rorse (replace 'h' with 'r')

rorse -> rose (remove 'r')

rose -> ros (remove 'e')

Example 2:

Input: word1 = "intention", word2 = "execution" | Output: 5

Explanation:

intention -> inention (remove 't')

inention -> enention (replace 'i' with 'e')

enention -> exention (replace 'n' with 'x')

exention -> exection (replace 'n' with 'c')

exection -> execution (insert 'u')

In [82]:
```
# Method 1
def edit_dist(word1,word2): # (Leet Code 190 / 1146 test cases passed.)
    w1,w2 = set(word1),set(word2)
    out = 0
    if word1 == '' or word2 == '':
        out = len(word1) or len(word2)
    elif word1 == word2:
        out = 0
    elif (len(word1) or len(word2)) == 1:
        out = len(w1 or w2)
    else:
        out1 = w1 & w2
        out = len(out1)
    return out
print(f'Output from method 1: {edit_dist("ab","bc")}')

# Method 2
def minDistance(self, s1: str, s2: str) -> int:

    @lru_cache(maxsize=None)
    def f(i, j):
            if i == 0 and j == 0: return 0
            if i == 0 or j == 0: return i or j
            if s1[i - 1] == s2[j - 1]:
                    return f(i - 1, j - 1)
            return min(1 + f(i, j - 1), 1 + f(i - 1, j), 1 + f(i - 1, j - 1

    m, n = len(s1), len(s2)
    return f(m, n)
# print(f'Output from method 1: {minDistance("ab","bc")}')
```

```
Output from method 1: 1
```

Yet to see this

Interleaving String

Given strings s1, s2, and s3, find whether s3 is formed by an interleaving of s1 and s2.

An interleaving of two strings s and t is a configuration where they are divided into non-empty substrings such that:

s = s1 + s2 + ... + sn

t = t1 + t2 + ... + tm

|n - m| <= 1

The interleaving is s1 + t1 + s2 + t2 + s3 + t3 + ... or t1 + s1 + t2 + s2 + t3 + s3 + ...

Note: a + b is the concatenation of strings a and b.

Example 1:

Input: s1 = "aabcc", s2 = "dbbca", s3 = "aadbbcbcac" | Output: true

Example 2:

Input: s1 = "aabcc", s2 = "dbbca", s3 = "aadbbbaccc" | Output: false

Example 3:

Input: s1 = "", s2 = "", s3 = "" | Output: true

In [155…

```
# Partial Output
def find_foo(s1,s2,s3):

    if len(s1) == len(s3) and len(s2) == '':
        out = True

    if len(s1) == len(s2):
        s11,s12,s13 = s1[:2],s1[2:4],s1[4]
        s21,s22,s23 = s2[:2],s2[2:4],s2[4]
        String = s11 + s21 + s12 + s22 + s23 + s13

        if String == s3:
            out = True
        else:
            out = False
    return out

print(f'Test Case 1 : {find_foo("aabcc","dbbca","aadbbcbcac")}')
print(f'Test Case 2 : {find_foo("aabcc","dbbca","aadbbbaccc")}')
# print(f'Test Case 3 : {find_foo("","","")}')
```

```
Test Case 1 : True
Test Case 2 : False
```

Yet to see this

Spiral Matrix

Given an m x n matrix, return all elements of the matrix in spiral order.

Example 1:

Input: matrix = [[[1,2,3]],[[4,5,6]],[[7,8,9]]]

Output: [[1,2,3,6,9,8,7,4,5]]

Example 2:

Input: matrix = [[[1,2,3,4]],[[5,6,7,8]],[[9,10,11,12]]]

Output: [[1,2,3,4,8,12,11,10,9,5,6,7]]

In [31]:
```
# def spiral_matrix(matrix):
#     start_row,end_row,start_col,end_col = 0,len(matrix),0,len(matrix[0])

#     out = []
#     while start_col < end_col or start_row < end_row:

#     # Right
#         if start_row < end_row:
#             for i in range(start_col,end_col):
#                 out.extend(matrix[start_row][i])
#         start_row += 1

#     # Down
#         if start_col < end_col:
#             for i in range(start_row,end_row):
#                 out.extend(matrix[i][end_col])
#         end_col -= 1

#     # left
#         if start_row < end_row:
#             for i in range(start_col-1,end_col-1,-1):
#                 out.extend(matrix[end_row][i])
#         end_row -= 1

#     # Top
#         if start_col < end_col:
#             for i in range(end_row-1,start_row-1,-1):
#                 out.extend(matrix[i][start_col])
#         start_col += 1


#     return out

# print(spiral_matrix([[1,2,3],[4,5,6],[7,8,9]]))
```

Search a 2D Matrix

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 from left to right.

The first integer of each row is greater than the last integer of the previous row.

Example 1:

Input: matrix = [[[1,3,5,7]],[[10,11,16,20]],[[23,30,34,60]]], target = 3

Output: true

Example 2:

Input: matrix = [[[1,3,5,7]],[[10,11,16,20]],[[23,30,34,60]]], target = 13

Output: false

In [35]:
```
def lookup_matrix(matrix,target):
    empty = []
    for i in matrix:
        for j in i:
            empty.append(j)
            if target in empty:
                out = True
            else:
                out = False
    return out

lookup_matrix([[1,3,5,7],[10,11,16,20],[23,30,34,60]],13)
```

Out[35]:
```
False
```

Yet to see this

Combinations

Given two integers n and k, return all possible combinations of k numbers out of 1 ... n.

You may return the answer in any order.

Example 1:

Input: n = 4, k = 2 Output: [[[2,4]], [[3,4]], [[2,3]], [[1,2]], [[1,3]], [[1,4]],]

Example 2:

Input: n = 1, k = 1

Output: [[[1]]]

In [175…
```
def combinations(n,k):
    out = []
    for i in range(1,n+1):
        for j in range(1,k+2):
            if i!=j:
                if i>j:
                    out.extend([[j,i]])
            elif n == k:
                out = [[n]]
    return outo
combinations(1,1)
```

Out[175… `[[1]]`

In [141…
```
n = 4
k = 2
out = []
emp1 = [i for i in range(1,n+1)]
emp2 = [j for j in range(1,k+1)]
# pair = [[a,b] for a in emp1 for b in emp2]
# for a in range(len(emp1)):
#     for b in range(len(emp2)):
#         # print(emp1[a],emp2[b])
#         if emp1[a] != emp2[b]:
#             out.extend([[emp1[a],emp2[b]]])
# out
clist = [(i,j) for i in emp1 for j in emp2]
clist
```

Out[141… `[(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2), (4, 1), (4, 2)]`

Reverse Words in a String

Given an input string s, reverse the order of the words.

A word is defined as a sequence of non-space characters. The words in s will be separated by at least one space.

Return a string of the words in reverse order concatenated by a single space.

Note that s may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

Example 1:

Input: s = "the sky is blue"

Output: "blue is sky the"

Example 2:

Input: s = " hello world "

Output: "world hello"

Explanation: Your reversed string should not contain leading or trailing spaces.

Example 3:

Input: s = "a good example"

Output: "example good a"

Explanation: You need to reduce multiple spaces between two words to a single space in the reversed string.

Example 4:

Input: s = " Bob Loves Alice "

Output: "Alice Loves Bob"

Example 5:

Input: s = "Alice does not even like bob"

Output: "bob like even not does Alice"

In [55]:
```
def rev_string(s):
    k = s.split()
    zx = k[::-1]
    output = ' '.join(zx)
    return output
rev_string("  Bob    Loves  Alice   ")
```

Out[55]: 'Alice Loves Bob'

Divide Two Integers

Given two integers dividend and divisor, divide two integers without using multiplication, division, and mod operator.

Return the quotient after dividing dividend by divisor.

The integer division should truncate toward zero, which means losing its fractional part. For example, truncate(8.345) = 8 and truncate(-2.7335) = -2.

Example 1:

Input: dividend = 10, divisor = 3 | Output: 3

Explanation: 10/3 = truncate(3.33333..) = 3.

Example 2:

Input: dividend = 7, divisor = -3 | Output: -2 Explanation: 7/-3 = truncate(-2.33333..) = -2.

Example 3:

Input: dividend = 0, divisor = 1 | Output: 0

Example 4:

Input: dividend = 1, divisor = 1 | Output: 1

In [53]:
```
def int_div(a,b): # (Leet Code 988 / 989 test cases passed.)
    return int(a/b)

int_div(10,3)
```

Out[53]: 2

Yet to se this

Remove Duplicate Letters or Smallest-subsequence of Distinct Characters

Given a string s, remove duplicate letters so that every letter appears once and only once. You must make sure your result is the smallest in lexicographical order among all possible results.

Note: This question is the same as 1081: https://leetcode.com/problems/smallest-subsequence-of-distinct-characters/

Example 1:

Input: s = "bcabc" | Output: "abc"

Example 2:

Input: s = "cbacdcbc" | Output: "acdb"

In [72]:
```
def delDups(l):
    zx = []
    for i in l:
        if i not in zx:
            zx.append(i)
    eve_lst = zx[0:len(zx):2]
    eve = set(eve_lst)
    odd_lst = zx[1:len(zx):2]
    odd = set(odd_lst)
    ass = eve.union(odd)
    return ass
delDups("cbacdcbc")
```

Out[72]:
```
{'a', 'b', 'c', 'd'}
```

To count $ in a 2D array for a certain range.

In [31]:
```
inp = [['$','.','.','.'],['.','.','$','$'],['$','.','$','.'],['.','.','.',
def count_dollor(aList,start_pos,end_pos):
    cout = 0
    x = [j for i in aList for j in i]
    for k in x[start_pos:end_pos]:
        if '$' in k:
            cout += 1
    return cout
count_dollor(inp,2,15)
```

Out[31]:
```
4
```

Yet to see this

Find First and Last Position of Element in Sorted Array

Given an array of integers nums sorted in ascending order, find the starting and ending position of a given target value.

If target is not found in the array, return [[-1, -1]].

Example 1:

Input: nums = [[5,7,7,8,8,10]], target = 8 | Output: [[3,4]]

Example 2:

Input: nums = [[5,7,7,8,8,10]], target = 6 | Output: [[-1,-1]]

Example 3:

Input: nums = [], target = 0 | Output: [[-1,-1]]

In [44]:
```
# Method 1
def searchRange(nums, target): # (Leet Code 42 / 88 test cases passed.)
        output = []
        if len(nums) == 1 and target in nums:
                output.extend([0,0])
        elif target in nums:
                for i,j in enumerate(nums):
                        if j == target:
                                output.append(i)
        else:
                output.extend([-1,-1])

        return output
print(f'Method 1 output: {searchRange([5,7,7,8,8,10],8)}')

# Method 2
def ane(aList,target):
    for i in range(len(aList)):
        if aList[i] == target:
            left = i
            break
        else :
            left = [-1,-1]

    for j in range(len(aList)-1,-1,-1):
        if aList[j] == target:
            right = j
            break

    return [left,right]
print(f'Method 2 output: {ane([5,7,7,8,8,10],8)}')
```

```
Method 1 output: [3, 4]
Method 2 output: [3, 4]
```

Majority Element

Given an array nums of size n, return the majority element.

The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

Example 1:

Input: nums = [[3,2,3]] | Output: 3

Example 2:

Input: nums = [[2,2,1,1,1,2,2]] | Output: 2

In [61]:
```python
# Method 1
# def major_element(aList): # (Leet code 30 / 46 test cases passed.)
#     z = [i for i in aList if aList.count(i) > 1]
#     return z
# print(f'Output from method 1: {major_element([8,8,7,7,7])}')

# Method 2
def most_common(lst):
    return max(set(lst), key=lst.count)
print(f'Output from Method 2: {most_common([8,8,7,7,7])}')

# Method 3
def find(aList):
    return sorted(aList)[len(aList)//2]
print(f'Output from Method 3: {find([8,8,7,7,7])}')
```

```
Output from Method 2: 7
Output from Method 3: 7
```

Yet to see this

Minimum Number of Removals to Make Mountain Array

You may recall that an array arr is a mountain array if and only if:

arr.length >= 3

There exists some index i (0-indexed) with 0 < i < arr.length - 1 such that:

arr[[0]] < arr[[1]] < ... < arr[[i - 1]] < arr[[i]]

arr[[i]] > arr[[i + 1]] > ... > arr[[arr.length - 1]]

Given an integer array nums, return the minimum number of elements to remove to make nums a mountain array.

Example 1:

Input: nums = [[1,3,1]] | Output: 0

Explanation: The array itself is a mountain array so we do not need to remove any elements.

Example 2:

Input: nums = [[2,1,1,5,6,2,3,1]] | Output: 3 Explanation: One solution is to remove the elements at indices 0, 1, and 5, making the array nums = [1,5,6,3,1].

Example 3:

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

Example 4:

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

In [135…

```
# l = [2,1,1,5,6,2,3,1]
# for i in range(len(l)-1):
#     # print (l[i],l[i+1])
#     if l[i]!=l[i+1]:
#         b.append(l[i+1])
# org = len(l)
# org2 = len(b)
# o = org - org2
# b
```

In [197…
```
# b = [4,3,2,1,1,2,3,1]
# out = []
# middle = max(b)
# for i in range(len(b)):
#     if b[i] < b[i-1]:
#         out.extend([b[i]])
#     # elif b[i] > b[i-1]:
#     #     out.extend([b[i]])
# output = len(b) - (len(b) - len(out))
```

In [55]:
```
# b = [1,3,1]
# for i,j in enumerate(b):
#     if b[i] < b[j] and b[i] > b[j-1]:
#         print('hi')
```

Maximum Subarray

Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Example 1:

Input: nums = [[-2,1,-3,4,-1,2,1,-5,4]] | Output: 6 | Explanation: [[4,-1,2,1]] has the largest sum = 6.

Example 2:

Input: nums = [[1]] | Output: 1

Example 3:

Input: nums = [[0]] | Output: 0

Example 4:

Input: nums = [[-1]] | Output: -1

Example 5:

Input: nums = [[-100000]] | Output: -100000

In [106…
```
# Method 1
def max_sub(aList):
    if len(aList) <= 2:
        for i in aList:
            if i == aList[0]:
                out = max(aList)
            elif i:
                out = sum(aList)
    else:
        ax = list(set(aList))
        ax.sort()
        ane = ax[len(ax)//2:]
        out = sum(ane)
    return out
print(f'Method 1 output: {max_sub([3,1])}')

# Method 2
def foo(nums):
    n = len(nums)
    dp = [0] * n
    dp[0] = nums[0]

    for i in range(1,n):
        dp[i] = max(dp[i-1] + nums[i], nums[i])

    return max(dp)
print(f'Method 2 output: {foo([3,1])}')
```

```
Method 1 output: 4
Method 2 output: 4
```

Maximum Product Subarray

Given an integer array nums, find the contiguous subarray within an array (containing at least one number) which has the largest product.

Example 1:

Input: [[2,3,-2,4]] | Output: 6 | Explanation: [[2,3]] has the largest product 6. Example 2:

Input: [[-2,0,-1]] | Output: 0 | Explanation: The result cannot be 2, because [[-2,-1]] is not a subarray.

In [138…
```
def sub_prod(lt):
    out,max_val,min_val = lt[0],lt[0],lt[0]
    for i in range(1,len(lt)):
        if lt[i] < 0
            max_val,min_val = min_val,max_val
        max_val = max(lt[i],max_val * lt[i])
        min_val = min(lt[i],min_val * lt[i])
        out = max(out,max_val)
    return out
sub_prod([-2,0,-1])
```

Out[138… 0

Yet to see this

Maximum Product of Three Numbers

Given an integer array nums, find three numbers whose product is maximum and return the maximum product.

Example 1:

Input: nums = [[1,2,3]] | Output: 6

Example 2:

Input: nums = [[1,2,3,4]] | Output: 24

Example 3:

Input: nums = [[-1,-2,-3]] | Output: -6

In [164…
```
def max_prd(aList):
    n = len(aList)
    max_prod = 0
    max_prod = max(max_prod,aList[0]*aList[1]*aList[n-1])
    max_prod = max(max_prod,aList[n-3]*aList[n-2]*aList[n-1])
    return max_prod
max_prd([-1,-2,-3])
```

Out[164… 0

Contains Duplicate

Given an array of integers, find if the array contains any duplicates.

Your function should return true if any value appears at least twice in the array, and it should return false if every element is distinct.

Example 1:

Input: [[1,2,3,1]] | Output: true

Example 2:

Input: [[1,2,3,4]] | Output: false

Example 3:

Input: [[1,1,1,3,3,4,3,2,4,2]] | Output: true

In [31]:
```
# Method 1
def contains_duplicate(nums):
    # j = 0
    for i in range(len(nums)):
        for j in range(1,len(nums)):
            if nums[i] == nums[j]:
                output =True
    return output
print(f'Method 1 output: {contains_duplicate([1,1,1,3,3,4,3,2,4,2])}')
```

```
Method 1 output: True
```

Implement strStr()

Return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

Clarification:

What should we return when needle is an empty string? This is a great question to ask during an interview.

For the purpose of this problem, we will return 0 when needle is an empty string. This is consistent to C's strstr() and Java's indexOf().

Example 1:

Input: haystack = "hello", needle = "ll" | Output: 2

Example 2:

Input: haystack = "aaaaa", needle = "bba" | Output: -1

Example 3:

Input: haystack = "", needle = "" | Output: 0

In [91]:
```
# Method 1
def imp_str(str1,str2):
    if not str1:
        if not str2:
            out = 0
        else:out = -1

    for i in range(len(str1)):
        for j in range(len(str2)):
            if str2[j] in str1[i]:
                out = str1.find(str2)

    return out
print(f'Method 1 output: {imp_str("bba","ba")}')

# Method 2
def imp_str2(str1,str2):
     return str1.index(str2) if str2 in str1 else -1
print(f'Method 2 output: {imp_str2("","")}')
```

```
Method 1 output: 1
Method 2 output: 0
```

Third Maximum Number

Given integer array nums, return the third maximum number in this array. If the third maximum does not exist, return the maximum number.

Example 1:

Input: nums = [[3,2,1]] | Output: 1

Explanation: The third maximum is 1.

Example 2:

Input: nums = [[1,2]] | Output: 2

Explanation: The third maximum does not exist, so the maximum (2) is returned instead.

Example 3:

Input: nums = [[2,2,3,1]] | Output: 1

Explanation: Note that the third maximum here means the third maximum distinct number. Both numbers with value 2 are both considered as second maximum.

In [57]:
```
def thridMax(aList):
    # s = sorted(set(aList))
    # if len(s) > 3:
    #     out = s[-3]
    # else:
    #     out = s[-1]
    return aList[-3] if len(sorted(set(aList))) > 3 else aList[-1]
thridMax([5,2,2])
```

Out[57]: 2

Yet to see this

Combination Sum

Given an array of distinct integers candidates and a target integer target, return a list of all unique combinations of candidates where the chosen numbers sum to target. You may return the combinations in any order.

The same number may be chosen from candidates an unlimited number of times. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

It is guaranteed that the number of unique combinations that sum up to target is less than 150 combinations for the given input.

Example 1:

Input: candidates = [[2,3,6,7]], target = 7 | Output: [[[2,2,3]],[[7]]]

Explanation: 2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used multiple times. 7 is a candidate, and 7 = 7. These are the only two combinations.

Example 2:

Input: candidates = [[2,3,5]], target = 8 | Output: [[[2,2,2,2]],[[2,3,3]],[[3,5]]]

Example 3:

Input: candidates = [[2]], target = 1 | Output: [] Example 4:

Input: candidates = [[1]], target = 1 | Output: [[[1]]] Example 5:

Input: candidates = [[1]], target = 2 | Output: [[[1,1]]]

In [115…

```
def combinations_sum(candidates,target):
    out = []
    for i in range(0,len(candidates)):
        for j in range(i-1,len(candidates)):
            for k in range(j+1,len(candidates)):
                tot = candidates[i] + candidates[j] + candidates[k]
                if(i==j&j!=k&k!=i):
                    if tot == target:
                        out.extend([[candidates[i],candidates[j],candidates
    return out
combinations_sum([2,3,6,7],7)
```

Out[115…

```
[[2, 2, 3]]
```

Yet to see this

Letter Combinations of a Phone Number

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.

A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example 1:

Input: digits = "23" | Output: [["ad","ae","af","bd","be","bf","cd","ce","cf"]]

Example 2:

Input: digits = "" | Output: []

Example 3:

Input: digits = "2" | Output: [["a","b","c"]]

In [ ]:

```
def phone(str1):
    list1 = list(str1.strip())
    di = {"2":['a','b','c'],"3":['d','e','f'],"4":['g','h','i'],"5":['j','l
    if not di:
        val = []
    val = [""]
    for i in di:
        cal = []
        for j in val:
            for k in di[i]:
                cal.append(j+k)

    val = cal
    return val

phone('23')
```

Yet to see this

Top K Frequent Words

Given a non-empty list of words, return the k most frequent elements.

Your answer should be sorted by frequency from highest to lowest. If two words have the same frequency, then the word with the lower alphabetical order comes first.

Example 1:

Input: [["i", "love", "leetcode", "i", "love", "coding"]], k = 2

Output: [["i", "love"]]

Explanation: "i" and "love" are the two most frequent words. Note that "i" comes before "love" due to a lower alphabetical order.

Example 2:

Input: [["the", "day", "is", "sunny", "the", "the", "the", "sunny", "is", "is"]], k = 4

Output: [["the", "is", "sunny", "day"]]

Explanation: "the", "is", "sunny" and "day" are the four most frequent words, with the number of occurrence being 4, 3, 2 and 1 respectively.

In [58]:
```
ut = ["the", "day", "is", "sunny", "the", "the", "the", "sunny", "is", "is"
k = 4
ut.sort()
for i in range(len(ut)):
    for j in range(len(ut[:k])):
```

Out[58]:
```
['day', 'is', 'is', 'is', 'sunny', 'sunny', 'the', 'the', 'the', 'the']
```

Find All Numbers Disappeared in an Array

Given an array of integers where 1 ≤ a[[i]] ≤ n (n = size of array), some elements appear twice and others appear once.

Find all the elements of [[1, n]] inclusive that do not appear in this array.

Could you do it without extra space and in O(n) runtime? You may assume the returned list does not count as extra space.

Example:

Input: [[4,3,2,7,8,2,3,1]] | Output:[[5,6]]

In [40]:
```
# method 1
def missElement(aList):
    if len(aList) == 0:
        out = []
    for i in range(len(aList)):
            for j in range(i+1,len(aList)):
                if aList[i] == aList[j]:
                    out = aList[j]+ 1
                else:
                    out = [i for i in range(1,max(aList)) if i not in aLis
    return [out]
print(f'Method 1 output: {missElement([1,1])}')

# method 2
def missElement2(nums):
    return list(set(range(1,len(nums)+1))-set(nums))
print(f'Method 1 output: {missElement2([1,1])}')
```

```
Method 1 output: [2]
Method 1 output: [2]
```

Kth Missing Positive Number

Given an array arr of positive integers sorted in a strictly increasing order, and an integer k.

Find the kth positive integer that is missing from this array.

Example 1:

Input: arr = [[2,3,4,7,11]], k = 5 | Output: 9

Explanation: The missing positive integers are [[1,5,6,8,9,10,12,13,...]]. The 5th missing positive integer is 9.

Example 2:

Input: arr = [[1,2,3,4]], k = 2 | Output: 6

Explanation: The missing positive integers are [[5,6,7,...]]. The 2nd missing positive integer is 6.

In [56]:
```
# method 1
def kthMissingPositiveNumber(nums,k):
    out = [i for i in range(1,max(nums)+k+1) if i not in nums]
    return out[k-1]
print(f'Method 1 output: {kthMissingPositiveNumber([1,2,3,4,5,6],5)}')
```

```
Method 1 output: 11
```

In [ ]:
```

```