**Wallbreakers Training**

Week 1 - Basics of Coding And Math

Coding exercises

* Arrays:
  + [https://leetcode.com/problems/sort-array-by-parity](https://leetcode.com/problems/sort-array-by-parity/description/)

Python

class Solution:

def sortArrayByParity(self, A):

even = []

odd = []

for i in A:

if i % 2 == 0:

even.append(i)

else:

odd.append(i)

return even + odd

* + [https://leetcode.com/problems/transpose-matrix](https://leetcode.com/problems/transpose-matrix/description/)
* java

class Solution:

def transpose(self, A):

num\_Rows = len(A[0])

result = [[] for i in range(num\_Rows)]

for i in range (len(A)):

for j in range(num\_Rows):

result[j].append(A[i][j])

return result

* [https://leetcode.com/problems/flipping-an-image](https://leetcode.com/problems/flipping-an-image/description/) (python)

def flipAndInvertImage(self, A):

"""

:type A: List[List[int]]

:rtype: List[List[int]]

"""

flip\_image = []

invert\_image = []

for i in A:

i.reverse()

invert\_image = []

for j in i:

if j == 0:

j = 1

else:

j = 0

invert\_image.append(j)

flip\_image.append(invert\_image)

return flip\_image

* Basic math:
  + [https://leetcode.com/problems/self-dividing-numbers](https://leetcode.com/problems/self-dividing-numbers/description/)
  + [https://leetcode.com/problems/fizz-buzz](https://leetcode.com/problems/fizz-buzz/description/)
  + [https://leetcode.com/problems/plus-one](https://leetcode.com/problems/plus-one/description/)

class Solution:

def plusOne(self, digits: List[int]) -> List[int]:

for i in range(len(digits)-1, -1, -1):

if digits[i] < 9:

digits[i] += 1

return digits

digits[i] = 0

return [1] + digits

* + [https://leetcode.com/problems/excel-sheet-column-number](https://leetcode.com/problems/excel-sheet-column-number/description/)
  + Java)

public class Solution {

public int titleToNumber(String s) {

int sum = 0;

for(int k = 0; k < s.length(); k++) {

sum \*= 26;

sum += (s.charAt(k) - 'A' + 1);

}

return sum;

}

}

<https://leetcode.com/problems/power-of-two>

class Solution: (Python)

def isPowerOfTwo(self, n):

return (n>0) and (n & (n-1)) == 0

Simple string manipulation:

[https://leetcode.com/problems/reverse-string](https://leetcode.com/problems/reverse-string/description/" \t "_blank)

class Solution( python):

def reverseString(self, s):

"""

:type s: List[str]

:rtype: None Do not return anything, modify s in-place instead.

"""

s.reverse()

return s

* + [https://leetcode.com/problems/detect-capital](https://leetcode.com/problems/detect-capital/description/)

class Solution:

def detectCapitalUse(self, word: str) -> bool:

return word in [word.upper(), word.lower(), word.title()]

[https://leetcode.com/problems/reverse-words-in-a-string-iii](https://leetcode.com/problems/reverse-words-in-a-string-iii/description/)

class Solution:(python)

def reverseWords(self, word: str) -> str:

words = word.split(' ') # creating a list of words

reverse = [] # reversed words will be stored here

for word in words:

reverse.append(word[::-1]) # adding reversed word to the reverse list

result = ' '.join(reverse) # joining the words with a space

return result

[https://leetcode.com/problems/valid-palindrome](https://leetcode.com/problems/valid-palindrome/description/)

class Solution:(python)

def isPalindrome(self, s: str) -> bool:

s = ''.join(i for i in s if i.isalnum()).lower()

return s == s[::-1]

* + [https://leetcode.com/problems/reverse-vowels-of-a-string](https://leetcode.com/problems/reverse-vowels-of-a-string/description/)

class Solution:

def reverseVowels(self, s: str) -> str:

"""

:type s: str

:rtype: str

"""

vowels = {'a','A','e','E','i','I','o','O','u','U'}

s = list(s)

left, right = 0, len(s) - 1

while left < right:

if s[left] not in vowels:

left += 1

elif s[right] not in vowels:

right -= 1

else:

s[left], s[right] = s[right], s[left]

left += 1

right -= 1

return "".join(s)

* [https://leetcode.com/problems/longest-common-prefix](https://leetcode.com/problems/longest-common-prefix/description/)

class Solution:

def longestCommonPrefix(self, strs):

"""

:type strs: List[str]

:rtype: str

"""

if not strs:

return ""

shortest = min(strs,key=len)

for i, ch in enumerate(shortest):

for other in strs:

if other[i] != ch:

return shortest[:i]

return shortest

* Simple bitwise manipulation:

[https://leetcode.com/problems/number-complement](https://leetcode.com/problems/number-complement/description/)

class Solution:

def findComplement(self, num: int) -> int:

b = bin(num)[2:]

orList = list(b)

rList = list()

for i in orList:

if i == '0':

rList.append('1')

else:

rList.append('0')

r = ''.join(rList)

return int(r,2)

<https://leetcode.com/problems/hamming-distance/>

class Solution(python):

def hammingDistance(self, x: int, y: int) -> int:

return bin(x^y).count('1')

* + [https://leetcode.com/problems/binary-gap](https://leetcode.com/problems/binary-gap/description/)

class Solution(object):

def binaryGap(self, N):

"""

:type N: int

:rtype: int

"""

temp\_gap = 0

bin\_gap = 0

one\_detected = 0

while N:

# counter will reset when we run into 1

if N%2 == 1:

one\_detected += 1

bin\_gap = max(bin\_gap, temp\_gap)

temp\_gap = 0

# counter will start when we already encountered a digit 1

# counter will increase when the current digit is 0

elif one\_detected:

temp\_gap += 1

N //= 2

if one\_detected < 2:

return 0

else:

return bin\_gap+1

# bin\_ gap will increased by 1, because we only counted zeros between ones

# and the distance between indexes of ones will be +1

* [https://leetcode.com/problems/single-number](https://leetcode.com/problems/single-number/description/)

class Solution(object)(python):

def singleNumber(self, nums):

"""

:type nums: List[int]

:rtype: int

"""

s = set()

for num in nums:

if num in s:

s.remove(num)

else:

s.add(num)

return s.pop()

* HashMap’s and sets:

<https://leetcode.com/problems/two-sum/>

class Solution(python):

def twoSum(self, nums, target):

"""

type nums: List[int]

type target: int

return type: List[int]

"""

#initialize dictionaries

dic = {}

for i, num in enumerate(nums):

remaining = target - num

if remaining in dic:

return [dic[remaining], i]

else:

dic[num] = i

return []

* + <https://leetcode.com/problems/valid-anagram/>

class Solution:

def isAnagram(self, s: str, t: str) -> bool:

if sorted(list(t)) == sorted(list(s)): # checking if both list are equal

return True

else:

return False

* Union find:
  + <https://leetcode.com/problems/friend-circles>
  + <https://leetcode.com/problems/number-of-islands>
  + <https://leetcode.com/problems/surrounded-regions>