BIT MANIPULATION

PROBLEM SET

- No.136 Single Number
- No.137 Single Number II
- No.190 Reverse Bits
- No.191 Number of 1 Bits
- No.201 Bitwise AND of Numbers Range
- No.231 Power of Two
- No.260 Single Number III
- No.268 Missing Number
- No.318 Maximum Product of Word Lengths
- No.342 Power of Four
- No.371 Sum of Two Integers
- No.389 Find the Difference
- No.393 UTF-8 Validation
- No.401 Binary Watch
- No.421 Maximum XOR of Two Numbers in an Array
- No.461 Hamming Distance
- No.462 Minimum Moves to Equal Array Elements II
- No.477 Total Hamming Distance

NO. 136

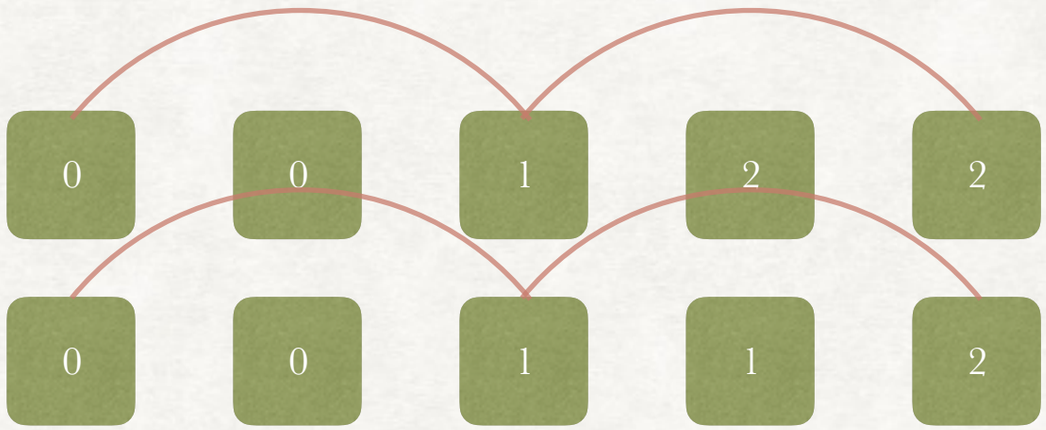
SINGLE NUMBER

PROBLEM DESCRIPTION

- Given an array of integers, every element appears twice except for one. Find that single one.

IDEAS

- Traverse this list with step = 2
  - E.g. for i in range(0, len(nums), 2)



- Compare element[i] and element[i+1]

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/136_Single_Number

NO. 137

SINGLE NUMBER

II

PROBLEM DESCRIPTION

- Given an array of integers, every element appears three times except for one, which appears exactly once. Find that single one.

IDEAS

- Sorting
- Traverse this list one by one
  - if incoming element is the same as the previous element
    - reference count += 1
  - if incoming element is different from the previous element
    - if reference count < 3 -> found missing number
    - if reference count == 3 -> update reference count to 1

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/137_Single_Number_II

NO. 260

SINGLE NUMBER

III

PROBLEM DESCRIPTION

- Given an array of numbers nums, in which exactly two elements appear only once and all the other elements appear exactly twice. Find the two elements that appear only once.
- For example:
- Given nums = [1, 2, 1, 3, 2, 5], return [3, 5].

IDEAS

- Sorting
- Use two variables “missingOne” and “missingTwo” to track the status
- for num in nums:
  - if num == missingOne, reset missingOne
  - if num == missingTwo, reset missingTwo
  - if missingOne is empty, missingOne = num
  - if missingTwo is empty, missingTwo = num
  - if num != missingOne or num != missingTwo, we got the answer

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/260_Single_Number_III

NO. 190

REVERSE BITS

PROBLEM DESCRIPTION

- Reverse bits of a given 32 bits unsigned integer.
- For example,
  - Given input 43261596
    - 00000010100101000001111010011100
  - Return 964176192
    - 00111001011110000010100101000000

IDEAS

- Use bin(%d) to convert to binary string
- Slice binaryStr[2:] to remove leading “0b”
- Formalize the binaryStr with leading ‘0’
  - “%032d” %(int(str(bin(n))[2:])
- Reverse the formalized string
  - binStr[::-1]
- Cast to 2 based binary back to decimal
  - int(reversedStr, 2)

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/190_Reverse_Bits

NO. 191

NUMBER OF 1 BITS

PROBLEM DESCRIPTION

- Write a function that takes an unsigned integer and returns the number of ‘1' bits it has (also known as the Hamming weight).
- For example, the 32-bit integer ‘11' has binary representation 00000000000000000000000000001011, so the function should return 3.

IDEAS

- Use Python built-in function
  - bin(n).count('1')
- Formal algorithm
- For every iteration
  - n & (n-1) -> get the rest value after removing tailing '1'

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/191_Number_of_1_Bits

NO. 201

BITWISE AND OF NUMBERS RANGE

PROBLEM DESCRIPTION

- Given a range [m, n] where 0 <= m <= n <= 2147483647, return the bitwise AND of all numbers in this range, inclusive.
- For example, given the range [5, 7], you should return 4.

IDEAS

- Brute-force approach leads to TLE
- Actually we don’t have to traverse this range one by one
  - Take [1, 4] into consideration

- 1 : 0001
- 2 : 0010
- 3 : 0011
- 4 : 1000

n & (n-1) =
4 & 3
= 0

從 tail 一次拔一個 1, 看最後剩多少？

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/201_Bitwise_AND_of_Numbers_Range

NO. 231

POWER OF TWO

PROBLEM DESCRIPTION

- Given an integer, write a function to determine if it is a power of two.

IDEAS

- Consider the sequence of power of 2
  - 1 : 0b00001
  - 2 : 0b00010
  - 4 : 0b00100
  - 8 : 0b01000
  - 16 : 0b10000

拔掉一個1 就變 0

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/231_Power_of_Two

NO. 342

POWER OF FOUR

PROBLEM DESCRIPTION

- Given an integer (signed 32 bits), write a function to check whether it is a power of 4.
- Example:
  - Given num = 16, return true. Given num = 5, return false.

IDEAS

- Consider the sequence of power of 4
  - 1 : 0b00001
  - 4 : 0b00100　　拔掉一個1 就變 0
  - 16 : 0b10000　　唯一的 1 出現在奇數位

SOLUTION

- https://github.com/Brady31027/leetcode/blob/master/342_Power_of_Four/power_of_four.py

NO. 268

MISSING NUMBER

PROBLEM DESCRIPTION

- Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, find the one that is missing from the array.
- For example
  - Given nums = [0, 1, 3] return 2.

IDEAS

- Given [0, 1, 3]
- Consider the expected sum
  - Ideally it will be 0+1+2+3 = 6
    - shortcut formula /* 等差級數和 */
      - n * (n + 1) / 2 = 3 * 4 / 2 = 6
- Calculate the actual sum : sum(nums)
- The diff is the ans

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/268_Missing_Number

NO. 318

MAXIMUM PRODUCT OF WORD LENGTHS

PROBLEM DESCRIPTION

- Given a string array words, find the maximum value of length(word[i]) * length(word[j]) where the two words do not share common letters. You may assume that each word will contain only lower case letters. If no such two words exist, return 0.

- Example 1:
  - Given ["abcw", "baz", "foo", "bar", "xtfn", "abcdef"]
  - Return 16
  - The two words can be "abcw", "xtfn".

IDEAS

- Since two words can not exist common characters
  - Convert every word to a specific number
  - if number(a) & number(b) > 0, then there is a common char
- Mapping function
  - core : 1 << (ord(c) - 97)
- Build up a hash table to represent each word
  - wordHash[mappingNumber] = len(word)
    - Update the value if hash conflicting happened (if necessary)

SOLUTION

- https://github.com/Brady31027/leetcode/blob/master/318_Maximum_Product_of_Word_Lengths/maximum_product.py

NO. 371

SUM OF TWO INTEGERS

PROBLEM DESCRIPTION

- Calculate the sum of two integers a and b, but you are not allowed to use the operator + and -.
- Example:
  - Given a = 1 and b = 2, return 3.

IDEAS

- Define the boundary
  - MAX = 0x7FFFFFFF
- Current digits
  - a ^ b
    - Calculate all current digits in one shot
      - current = (a ^ b) & 0xFFFFFFFF
- Carry digits
  - a & b
    - Calculate all carry digits in on shot
      - carry = (a & b) & 0xFFFFFFFF
- Special case handling: What if the sum-up value is greater than MAX? -> It's a negative value

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/371_Sum_of_Two_Integers

NO. 389

FIND THE DIFFERENCE

PROBLEM DESCRIPTION

- Given two strings s and t which consist of only lowercase letters. String t is generated by random shuffling string s and then add one more letter at a random position. Find the letter that was added in t.

- Example:
  - Input: s = “abcd”, t = "abcde"
  - Output: e
  - Explanation:
    - 'e' is the letter that was added.

IDEAS

- Given 2 strings a and b
  - for char in b:
    - if char not in a: return char
    - if char in a: compare its appearing count

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/389_Find_the_Difference

NO. 393

UTF-8 VALIDATION

PROBLEM DESCRIPTION

- A character in UTF8 can be from 1 to 4 bytes long, subjected to the following rules:
  - For 1-byte character, the first bit is a 0, followed by its unicode code.
  - For n-bytes character, the first n-bits are all one's, the n+1 bit is 0, followed by n-1 bytes with most significant 2 bits being 10.
- Given an array of integers representing the data, return whether it is a valid utf-8 encoding

IDEAS

- Valid length -> 1 Byte, 2 Bytes, 3 Bytes, and 4 Bytes
- Use “count” to track the length
  - if count == 0, then it’s a new start UTF-8 data
  - if count > 0, then the current byte will follow the previous byte
- How to determine the initial count?
  - byte >> 5 == 0b110 -> count = 1 (actual byte = 2)
  - byte >> 4 == 0b1110 -> count = 1 (actual byte = 3)
  - …etc

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/393_UTF-8_Validation

NO. 401

BINARY WATCH

PROBLEM DESCRIPTION

- A binary watch has 4 LEDs on the top which represent the hours (0-11), and the 6 LEDs on the bottom represent the minutes (0-59).
- Given a non-negative integer n which represents the number of LEDs that are currently on, return all possible times the watch could represent.

IDEAS

- Total 10 LEDs
  - 2 ** 10 -> 1024 # not a huge number, enumerate all possibilities
  - for num in range(1024):
    - if bin(num).count('1') == specified_led_number:
      - hour = num >> 6
      - minute = num & 0x3F # 0b111111
- Remember to check the boundary
  - hour could only be [0, 12]
  - minute could only be [0, 60]

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/401_Binary_Watch

NO. 421

MAXIMUM XOR OF TWO NUMBERS IN AN ARRAY

PROBLEM DESCRIPTION

- Given a non-empty array of numbers (range = [0, 2 ** 31])
- Find the maximum result of ai XOR
- Example:
- Input: [3, 10, 5, 25, 2, 8]
- Output: 28
- Explanation: The maximum result is 5 ^ 25 = 28

IDEAS

- Use itertool.combinations leads to TLE
- Not straight-forward
  - Use bit scanning
  - if a ^ b = c, then a ^ c = b

IDEAS

- Use 4 bits for our case study
- Given nums = [3, 4, 6] # [0b0011, 0b0100, 0b0110]
- We know ans = 7 (0b0111) by calculating 3 ^ 4
- Since we want to find the maximum xor result, we scan bits from left to right
  - Enlarge mask one by one
    - 1st: 1000
    - 2nd: 1100
    - 3rd: 1110
    - 4th: 1111

IDEAS

- For each iteration, we have a corresponding mask (0b1000, 0b1100, 0b1110, or 0b1111). Thus we can get the possible answers and track them using set()
  - Record these possible answers using candidates = set()
  - candidates = set([num & mask for num in nums])
- For each iteration, the maximum number we can possibly get is (1 << bit), by taking the previous result, we can formalize it as the follows:
  - guess = ans | (1 << bit)

IDEAS

- if guess ^ one of the candidates == another one candidates
  - A ^ B = C => A ^ C = B
  - one of the candidates ^ another one of the candidates = guess

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/421_Maximum_XOR_of_Two_Numbers_in_an_Array

NO. 461

HAMMING DISTANCE

PROBLEM DESCRIPTION

- The Hamming distance between two integers is the number of positions at which the corresponding bits are different.
- Given two integers x and y, calculate the Hamming distance
- Example:
- Input: x = 1, y = 4
  - 1 = 0b001
  - 4 = 0b100
- Output: 2

IDEAS

- a XOR b
- calculate how many 1s are there in the result

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/461_Hamming_Distance

NO. 477

TOTAL HAMMING DISTANCE

PROBLEM DESCRIPTION

- Find the total Hamming distance between all pairs of the given numbers.
- Example:
- Input: 4, 14, 2
  - dis(4, 14) + dis(4, 2) + dis(14, 2) = 2 + 2 + 2 = 6
- Output: 6

IDEAS

- Use itertool.combinations leads to TLE
- Not straight-forward, simply analyze different cases
  - E.g. nums = [4, 14, 2]
    - 4 : 0b 0 1 0 0
    - 2 : 0b 0 0 1 0
    - 14 : 0b 1 1 1 0

bit scanning

digit0	digit1
# of 1: 0	# of 1: 2
# of 0: 3	# of 0: 1

digit2	digit3
# of 1: 2	# of 1: 1
# of 0: 1	# of 0: 2

dis per bit = # of 1s * # of 0s

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/477_Total_Hamming_Distance

NO. 462

MINIMUM MOVES TO EQUAL ARRAY ELEMENTS II

PROBLEM DESCRIPTION

- Given a non-empty integer array, find the minimum number of moves required to make all array elements equal, where a move is incrementing a selected element by 1 or decrementing a selected element by 1

- Example:

- Input: [1,2,3]

- Output: 2

IDEAS

- Calculate the median
- Sum up the difference between each element and the median

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/462_Minimum_Moves_to_Equal_Array_Elements_II