HASH TABLE

PROBLEM SET

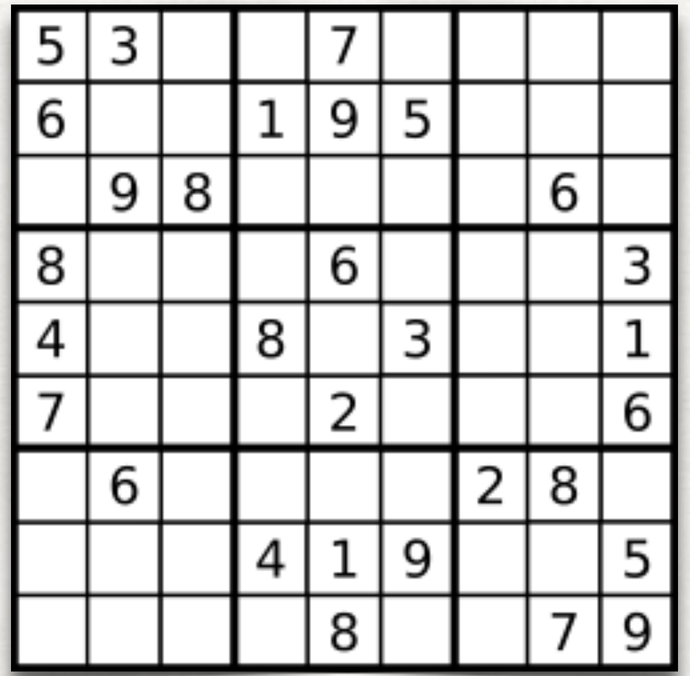
- No.36 Valid Sudoku
- No.202 Happy Number
- No.204 Count Primes
- No.205 Isomorphic Strings
- No.290 Word Pattern
- No.299 Bulls and Cows
- No.49 Group Anagrams
- No.424 Longest Repeating Character Replacement
- No. 454 4 Sum II
- No.387 First Unique Character in a String
- No.409 Longest Palindrome
- No.438 Find All Anagrams in a String
- No.447 Number of Boomerangs
- No.3 Longest Substring Without Repeating Characters
- No.187 Repeated DNA Sequences
- No.388 Longest Absolute File Path
- No.473 Matchsticks to Square
- No.554 Brick Wall

NO. 36

VALID SUDOKU

PROBLEM DESCRIPTION

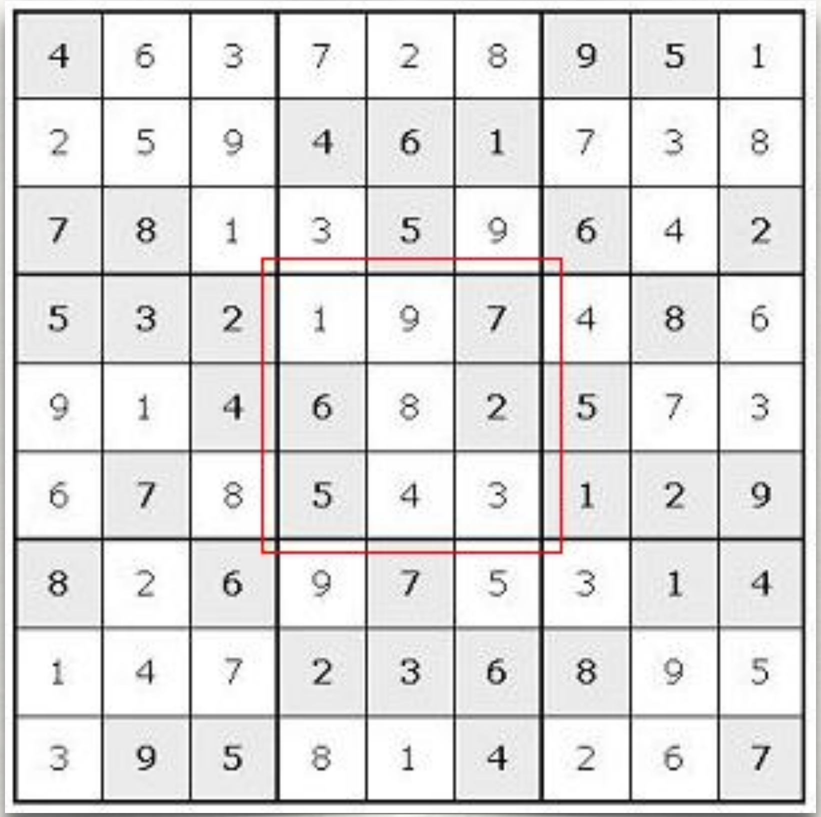
- Determine if a Sudoku is valid



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

IDEA

- Domain 1~9
- No repeated numbers per row and per column
- No repeated numbers per each 3X3 grid



4	6	3	7	2	8	9	5	1
2	5	9	4	6	1	7	3	8
7	8	1	3	5	9	6	4	2
5	3	2	1	9	7	4	8	6
9	1	4	6	8	2	5	7	3
6	7	8	5	4	3	1	2	9
8	2	6	9	7	5	3	1	4
1	4	7	2	3	6	8	9	5
3	9	5	8	1	4	2	6	7

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/36_Valid_Sudoku

NO. 202

HAPPY NUMBER

PROBLEM DESCRIPTION

- Write an algorithm to determine if a number is "happy".
  - A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers.
- Example: 19 is a happy number
  - 12 + 92 = 82
  - 82 + 22 = 68
  - 62 + 82 = 100
  - 12 + 02 + 02 = 1

IDEA

- Use a set() to track whether there is a loop
    - if a number already existed in set(), we don't have to evaluate

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/202_Happy_Number

NO. 204

COUNT PRIMES

PROBLEM DESCRIPTION

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

IDEA

- Sieve of Eratosthenes
- Sugar coating

LEVEARGE SQRT

```
for i in range(2, int(n ** 0.5) + 1):
        if primes[i]:
            primes[i * i: n: i] = [False] * len(primes[i * i: n: i])
```

MUL STARTS FROM I

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/204_Count_Primes

NO. 205

ISOMORPHIC STRINGS

PROBLEM DESCRIPTION

- Given two strings s and t, determine if they are isomorphic.
- Two strings are isomorphic if the characters in s can be replaced to get t.

IDEA

- Two hash tables

```
if s[i] not in charHashS and t[i] not in charHashT:
        charHashS[ s[i] ] = t[i]
        charHashT[ t[i] ] = s[i]
elif s[i] in charHashS and t[i] not in charHashT:
        return False
elif s[i] not in charHashS and t[i] in charHashT:
        return False
elif s[i] in charHashS and t[i] in charHashT:
        if charHashS[ s[i] ] != t[i]: return False
        if charHashT[ t[i] ] != s[i]: return False
```

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/205_Isomorphic_Strings

NO. 290

WORD PATTERN

PROBLEM DESCRIPTION

- Given a pattern and a string str, find if str follows the same pattern. Here follow means a full match, such that there is a bijection between a letter in pattern and a non-empty word in str.
- Examples:
  - pattern = "abba", str = "dog cat cat dog" should return true.

IDEA

- Exactly the same as 205. Isomorphic strings

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/290_Word_Pattern

NO. 299

BULLS AND COWS

PROBLEM DESCRIPTION

- Write a function to return a hint according to the secret number and friend's guess
- For example:
  - Secret number: "1807"
  - Friend's guess: “7810"
  - return “1A3B”

IDEA

- Total = sum(min(guess.count(i), secret.count(i)) for i in "0123456789")
- A = sum(itertools.imap(operator.eq, secret, guess))
- B = Total - A

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/299_Bulls_and_Cows

NO. 49

GROUP ANAGRAMS

PROBLEM DESCRIPTION

- Given an array of strings, group anagrams together.
- For example, given: ["eat", "tea", "tan", "ate", "nat", "bat"],
  - Return:
  - [
    - ["ate", "eat","tea"],
    - ["nat","tan"],
    - ["bat"]
  -]

IDEA

- Use hash table defaultdict(list)
  - key: “”.join(sorted(str))
  - value: str

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/49_Group_Anagrams

NO. 187

REPEATED DNA SEQUENCES

PROBLEM DESCRIPTION

- All DNA is composed of a series of nucleotides abbreviated as A, C, G, and T, for example: "ACGAATTCCG". When studying DNA, it is sometimes useful to identify repeated sequences within the DNA. Write a function to find all the 10-letter-long sequences (substrings) that occur more than once in a DNA molecule.

- For example,
  - Given s = “AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT",
  - Return: ["AAAAACCCCC", "CCCCCAAAAA"].

IDEA

- Use sliding window to scan the input string
- Use set() to memorize existed pattern

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/187_Repeated_DNA_Sequences

NO. 387

FIRST UNIQUE CHARACTER IN A STRING

PROBLEM DESCRIPTION

- Given a string, find the first non-repeating character in it and return it's index. If it doesn't exist, return -1.
- Examples:
    - s = "leetcode"
    - return 0.
    - s = "loveleetcode",
    - return 2.

IDEA

- Levearge collections.Counter()
- Linear scan the input string from head and check with Counter value

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/387_First_Unique_Character_in_a_String

NO. 409

LONGEST PALINDROME

PROBLEM DESCRIPTION

- Given a string which consists of lowercase or uppercase letters, find the length of the longest palindromes that can be built with those letters. This is case sensitive, for example "Aa" is not considered a palindrome here.
- Example
    - Input:"abccccdd"
    - Output: 7
    - Explanation:
        - One longest palindrome that can be built is "dccaccd", whose length is 7.

IDEA

- Maintain a hash (char: appearing count)
    - itertools.Counter(s).iteritems()
        - Return this hash
- Count how many char appears odd times, say n
- Answer is len(s) - n + 1(if there are more than one odd items, we can put one of them in the center)

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/409_Longest_Palindrome

NO. 447

NUMBER OF BOOMERANGS

PROBLEM DESCRIPTION

- Given n points in the plane that are all pairwise distinct, a "boomerang" is a tuple of points (i, j, k) such that the distance between i and j equals the distance between i and k (the order of the tuple matters).
- Example
    - Input: [[0,0],[1,0],[2,0]]
    - Output: 2
    - Explanation:
        - The two boomerangs are [[1,0],[0,0],[2,0]] and [[1,0],[2,0],[0,0]]

IDEA

- For every point i, if there are n points which have the same distance between i and themselves, there are totally n * (n-1) combinations
- E.g. For point a, there are point b and c which have the same distance between (a, b) and (a, c). Combination includes (abc, acb) = 2 * (2-1) = 2

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/447_Number_of_Boomerangs

NO. 3

LONGEST SUBSTRING WITHOUT REPEATING CHARACTERS

PROBLEM DESCRIPTION

- Given a string, find the length of the longest substring without repeating characters.
- Examples:
- Given "abcabcbb", the answer is "abc", which the length is 3.
- Given "bbbbb", the answer is "b", with the length of 1.
- Given "pwwkew", the answer is "wke", with the length of 3. Note that the answer must be a substring, "pwke" is a subsequence and not a substring.

IDEA

- Approach1
    - Maintain a queue which only contains unique chars
- Approach2
    - Maintain a start pointer which move forwards
        - Pros: Don’t have to traverse array one by one
        - Cons: Need to take care of the pointers
            - E.g. if s[i] in tbl and **start <= tbl[s[i]]**

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/3_Longest_Substring_Without_Repeating_Characters

NO. 438

FIND ALL ANAGRAMS IN A STRING

PROBLEM DESCRIPTION

- Given a string s and a non-empty string p, find all the start indices of p's anagrams in s.
- Strings consists of lowercase English letters only and the length of both strings s and p will not be larger than 20,100.
- The order of output does not matter.

IDEA

- Sliding window + sorted string causes TLE
- Use two collections.Counter() to track s and p
- Use one collections.Counter() to maintain p
    - use count to represent the differences between the golden and the current sliding window
    - if count == 0, then we found the anagram
    - if end - start > pattern_size, the move the start pointer

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/438_Find_All_Anagrams_in_a_String

- Template:
  - https://discuss.leetcode.com/topic/30941/here-is-a-10-line-template-that-can-solve-most-substring-problems

NO. 424

LONGEST REPEATING CHARACTER REPLACEMENT

PROBLEM DESCRIPTION

- Given a string that consists of only uppercase English letters, you can replace any letter in the string with another letter at most k times. Find the length of a longest substring containing all repeating letters you can get after performing the above operations.

IDEA

- Use two pointers and hash
    - start, end pointers point to string (array of char)
    - use built-in collections.Counter() as hash
- Input a char, end++
- while end - start - max(tbl.values()) > k: start++

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/424_Longest_Repeating_Character_Replacement

NO. 454

4SUM II

PROBLEM DESCRIPTION

- Given four lists A, B, C, D of integer values, compute how many tuples (i, j, k, l) there are such that A[i] + B[j] + C[k] + D[l] is zero.
- Example
    - Input: A = [1, 2], B = [-2,-1], C = [-1, 2], D = [0, 2]
    - Output: 2
    - Explanation:
        - (0, 0, 0, 1) -> A[0] + B[0] + C[0] + D[1] = 1 + (-2) + (-1) + 2 = 0
        - (1, 1, 0, 0) -> A[1] + B[1] + C[0] + D[0] = 2 + (-1) + (-1) + 0 = 0

IDEA

- Divide to 2-2 pairs
    - save sum(a, b) for a in A for b in B to table
    - count how many sum(c, d) == table(sum(a, b)) for c in C for d in D

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/454_4Sum_II

NO. 388

LONGEST ABSOLUTE FILE PATH

PROBLEM DESCRIPTION

- The string "dir\n\tsubdir1\n\tsubdir2\n\t\tfile.ext" represents:
  - dir\
  -     subdir1\
  -     subdir2\
  -         file.ext
- We are interested in finding the longest (number of characters) absolute path to a file within our file system

IDEA

- Review string.lstrip() and string.striplines()
    - Use string.striplines() to separate lines according to ‘\n’
    - Use string.lstrip() to get the base name
- Determine it’s a folder or a file
    - if ‘.’ in base_name, then it’s a file
        - maxLength = max(maxLength, tbl[depth] + len(base_name))
    - otherwise it’s a folder
        - tbl[depth + 1] = tbl[depth] + 1 + len(base_name)
        - + 1 if for ‘/’

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/388_Longest_Absolute_File_Path

NO.473

MATCHSTICKS TO SQUARE

PROBLEM DESCRIPTION

- Your input will be several matchsticks the girl has, represented with their stick length. Your output will either be true or false, to represent whether you could make one square using all the matchsticks the little match girl has.
- Example
    - Input: [1,1,2,2,2]
    - Output: true
    - Explanation: You can form a square with length 2, one side of the square came two sticks with length 1.

IDEA

- Use DFS
  - nums: sorted number with descending order
  - pos: pointer to nums
  - target: remaining edge length, init to [sideLen, sideLen, sideLend, sideLen]

```
def dfs(nums, pos, target):
    no_more_stick: return True
    foreach edge:
        if remainingEdgeLen >= nums[pos]:
            fill_with_the_longest_stick()
            if dfs(nums, pos+1, new_target): return True
            take_the_longest_stick_away()
    return False
```

SOLUTION

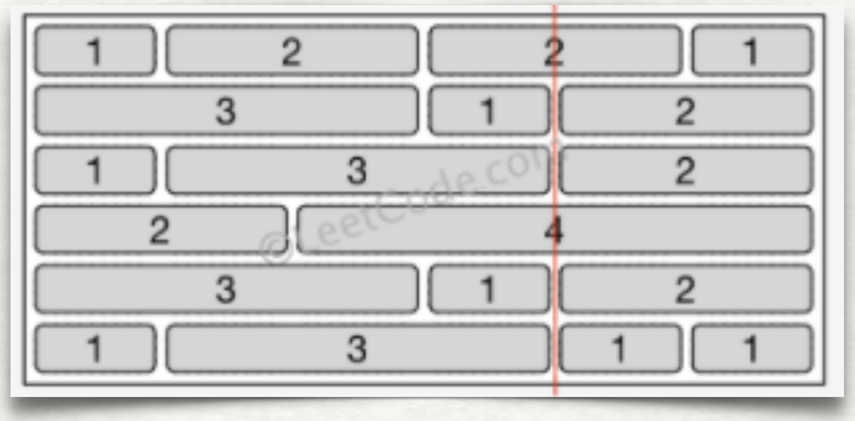
- https://github.com/Brady31027/leetcode/tree/master/473_Matchsticks_to_Square

NO.554

BRICK WALL

PROBLEM DESCRIPTION

- There is a brick wall in front of you. The wall is rectangular and has several rows of bricks. The bricks have the same height but different width. You want to draw a vertical line from the top to the bottom and cross the least bricks.



IDEA

- Accumulate brick width
  - E.g. [1,2,1] -> [1,3,4]
  - Skip the last brick or ans will be 0 for every case
    - E.g [1,3,4] -> [1, 3]
- collections.Counter() the accumulated width for every row
- Answer is len(wall) - max(Counter(wall).values())
- Special case: [[1], [1], [1]] -> Counter(wall).values() will be None
  - Adjust to max(Counter(wall).values() or [0])

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/554_Brick_Wall