My LeetCode Submissions - @RedSerperior

Download PDF | Follow @TheShubham99 on GitHub | Star on GitHub | View Source Code

1528 Kids With the Greatest Number of Candies (link)

Description

There are `n` kids with candies. You are given an integer array `candies`, where each `candies[i]` represents the number of candies the `i`th kid has, and an integer `extraCandies`, denoting the number of extra candies that you have.

Return *a boolean array* `result` *of length* `n`*, where* `result[i]` *is* `true` *if, after giving the* `i`th *kid all the* `extraCandies`*, they will have the* ***greatest*** *number of candies among all the kids, or* `false` *otherwise*.

Note that **multiple** kids can have the **greatest** number of candies.

Example 1:

```
Input: candies = [2,3,5,1,3], extraCandies = 3
Output: [true,true,true,false,true]
Explanation: If you give all extraCandies to:
- Kid 1, they will have 2 + 3 = 5 candies, which is the greatest among the kids.
- Kid 2, they will have 3 + 3 = 6 candies, which is the greatest among the kids.
- Kid 3, they will have 5 + 3 = 8 candies, which is the greatest among the kids.
- Kid 4, they will have 1 + 3 = 4 candies, which is not the greatest among the kids.
- Kid 5, they will have 3 + 3 = 6 candies, which is the greatest among the kids.
```

Example 2:

```
Input: candies = [4,2,1,1,2], extraCandies = 1
Output: [true,false,false,false,false]
Explanation: There is only 1 extra candy.
Kid 1 will always have the greatest number of candies, even if a different kid is given t
```

Example 3:

```
Input: candies = [12,1,12], extraCandies = 10
Output: [true,false,true]
```

Constraints:

- `n == candies.length`
- `2 <= n <= 100`
- `1 <= candies[i] <= 100`
- `1 <= extraCandies <= 50`

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def kidsWithCandies(self, candies: List[int], extraCandies: int) -> List[bool]:
        listb=[]

        for i in candies:
            new=0
            j=0
            new=i+extraCandies
            while j<len(candies):
                if new<candies[j]:
                    listb.append(False)
                    break
                j+=1
            if j==len(candies):
                listb.append(True)


        return listb
```

2383 Add Two Integers (link)

Description

Given two integers `num1` and `num2`, return *the **sum** of the two integers*.

Example 1:

```
Input: num1 = 12, num2 = 5
Output: 17
Explanation: num1 is 12, num2 is 5, and their sum is 12 + 5 = 17, so 17 is returned.
```

Example 2:

```
Input: num1 = -10, num2 = 4
Output: -6
Explanation: num1 + num2 = -6, so -6 is returned.
```

Constraints:

- `-100 <= num1, num2 <= 100`

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def sum(self, num1: int, num2: int) -> int:
        return num1+num2
```

3476 Find Minimum Operations to Make All Elements Divisible by Three (link)

Description

You are given an integer array `nums`. In one operation, you can add or subtract 1 from **any** element of `nums`.

Return the **minimum** number of operations to make all elements of `nums` divisible by 3.

Example 1:

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

Output: 3

Explanation:

All array elements can be made divisible by 3 using 3 operations:

- Subtract 1 from 1.
- Add 1 to 2.
- Subtract 1 from 4.

Example 2:

Input: nums = [3,6,9]

Output: 0

Constraints:

- `1 <= nums.length <= 50`
- `1 <= nums[i] <= 50`

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def minimumOperations(self, nums: List[int]) -> int:
        count=0
        for i in nums:
            if i %3!=0:
                if (i-1) %3==0:
                    i-=1
                    count+=1
                else:
                    i+=1
                    count+=1


        return count
```

1635 Number of Good Pairs (link)

Description

Given an array of integers `nums`, return *the number of* ***good pairs***.

A pair `(i, j)` is called *good* if `nums[i] == nums[j]` and `i < j`.

Example 1:

```
Input: nums = [1,2,3,1,1,3]
Output: 4
Explanation: There are 4 good pairs (0,3), (0,4), (3,4), (2,5) 0-indexed.
```

Example 2:

```
Input: nums = [1,1,1,1]
Output: 6
Explanation: Each pair in the array are good.
```

Example 3:

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

Constraints:

- `1 <= nums.length <= 100`
- `1 <= nums[i] <= 100`

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def numIdenticalPairs(self, nums: List[int]) -> int:
        pair=0
        for i in range(0,len(nums)-1):
            for j in range(i+1,len(nums)):
                if nums[i]==nums[j] and i <j:
                    pair+=1

        return pair
```

2058 Concatenation of Array (link)

Description

Given an integer array `nums` of length `n`, you want to create an array `ans` of length `2n` where `ans[i] == nums[i]` and `ans[i + n] == nums[i]` for `0 <= i < n` (**0-indexed**).

Specifically, `ans` is the **concatenation** of two `nums` arrays.

Return *the array* `ans`.

Example 1:

```
Input: nums = [1,2,1]
Output: [1,2,1,1,2,1]
Explanation: The array ans is formed as follows:
- ans = [nums[0],nums[1],nums[2],nums[0],nums[1],nums[2]]
- ans = [1,2,1,1,2,1]
```

Example 2:

```
Input: nums = [1,3,2,1]
Output: [1,3,2,1,1,3,2,1]
Explanation: The array ans is formed as follows:
- ans = [nums[0],nums[1],nums[2],nums[3],nums[0],nums[1],nums[2],nums[3]]
- ans = [1,3,2,1,1,3,2,1]
```

Constraints:

- `n == nums.length`
- `1 <= n <= 1000`
- `1 <= nums[i] <= 1000`

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def getConcatenation(self, nums: List[int]) -> List[int]:
        return nums+nums
```

2556 Convert the Temperature (link)

Description

You are given a non-negative floating point number rounded to two decimal places `celsius`, that denotes the **temperature in Celsius**.

You should convert Celsius into **Kelvin** and **Fahrenheit** and return it as an array `ans = [kelvin, fahrenheit]`.

Return *the array* `ans`. Answers within 10^{-5} of the actual answer will be accepted.

Note that:

- `Kelvin = Celsius + 273.15`
- `Fahrenheit = Celsius * 1.80 + 32.00`

Example 1:

```
Input: celsius = 36.50
Output: [309.65000,97.70000]
Explanation: Temperature at 36.50 Celsius converted in Kelvin is 309.65 and converted in
```

Example 2:

```
Input: celsius = 122.11
Output: [395.26000,251.79800]
Explanation: Temperature at 122.11 Celsius converted in Kelvin is 395.26 and converted ir
```

Constraints:

- `0 <= celsius <= 1000`

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def convertTemperature(self, celsius: float) -> List[float]:
        kelvin=celsius+273.15
        fahrenheit= celsius *1.80+32.00
        ans = [kelvin,fahrenheit]
        return ans
```

2048 Build Array from Permutation (link)

Description

Given a **zero-based permutation** `nums` (**0-indexed**), build an array `ans` of the **same length** where `ans[i] = nums[nums[i]]` for each `0 <= i < nums.length` and return it.

A **zero-based permutation** `nums` is an array of **distinct** integers from `0` to `nums.length - 1` (**inclusive**).

Example 1:

```
Input: nums = [0,2,1,5,3,4]
Output: [0,1,2,4,5,3]
Explanation: The array ans is built as follows:
ans = [nums[nums[0]], nums[nums[1]], nums[nums[2]], nums[nums[3]], nums[nums[4]], nums[nu
    = [nums[0], nums[2], nums[1], nums[5], nums[3], nums[4]]
    = [0,1,2,4,5,3]
```

Example 2:

```
Input: nums = [5,0,1,2,3,4]
Output: [4,5,0,1,2,3]
Explanation: The array ans is built as follows:
ans = [nums[nums[0]], nums[nums[1]], nums[nums[2]], nums[nums[3]], nums[nums[4]], nums[nu
    = [nums[5], nums[0], nums[1], nums[2], nums[3], nums[4]]
    = [4,5,0,1,2,3]
```

Constraints:

- `1 <= nums.length <= 1000`
- `0 <= nums[i] < nums.length`
- The elements in `nums` are **distinct**.

Follow-up: Can you solve it without using an extra space (i.e., `O(1)` memory)?

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def buildArray(self, nums: List[int]) -> List[int]:
        ans=[]
        for i in range(0,len(nums)):
            ans.append(nums[nums[i]])
        return ans
```

3194 Find Words Containing Character (link)

Description

You are given a **0-indexed** array of strings `words` and a character `x`.

Return *an **array of indices** representing the words that contain the character* `x`.

Note that the returned array may be in **any** order.

Example 1:

```
Input: words = ["leet","code"], x = "e"
Output: [0,1]
Explanation: "e" occurs in both words: "leet", and "code". Hence, we return indices 0 and
```

Example 2:

```
Input: words = ["abc","bcd","aaaa","cbc"], x = "a"
Output: [0,2]
Explanation: "a" occurs in "abc", and "aaaa". Hence, we return indices 0 and 2.
```

Example 3:

```
Input: words = ["abc","bcd","aaaa","cbc"], x = "z"
Output: []
Explanation: "z" does not occur in any of the words. Hence, we return an empty array.
```

Constraints:

- `1 <= words.length <= 50`
- `1 <= words[i].length <= 50`
- `x` is a lowercase English letter.
- `words[i]` consists only of lowercase English letters.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def findWordsContaining(self, words: List[str], x: str) -> List[int]:
        l=[]
        for i in range(0,len(words)):
            if x in words[i]:
                l.append(i)
        return l
```

3172 Divisible and Non-divisible Sums Difference (link)

Description

You are given positive integers `n` and `m`.

Define two integers as follows:

- `num1`: The sum of all integers in the range `[1, n]` (both **inclusive**) that are **not divisible** by `m`.
- `num2`: The sum of all integers in the range `[1, n]` (both **inclusive**) that are **divisible** by `m`.

Return *the integer* `num1 - num2`.

Example 1:

```
Input: n = 10, m = 3
Output: 19
Explanation: In the given example:
- Integers in the range [1, 10] that are not divisible by 3 are [1,2,4,5,7,8,10], num1 is
- Integers in the range [1, 10] that are divisible by 3 are [3,6,9], num2 is the sum of t
We return 37 - 18 = 19 as the answer.
```

Example 2:

```
Input: n = 5, m = 6
Output: 15
Explanation: In the given example:
- Integers in the range [1, 5] that are not divisible by 6 are [1,2,3,4,5], num1 is the s
- Integers in the range [1, 5] that are divisible by 6 are [], num2 is the sum of those i
We return 15 - 0 = 15 as the answer.
```

Example 3:

```
Input: n = 5, m = 1
Output: -15
Explanation: In the given example:
- Integers in the range [1, 5] that are not divisible by 1 are [], num1 is the sum of tho
- Integers in the range [1, 5] that are divisible by 1 are [1,2,3,4,5], num2 is the sum o
We return 0 - 15 = -15 as the answer.
```

Constraints:

- `1 <= n, m <= 1000`

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def differenceOfSums(self, n: int, m: int) -> int:
        sum1=0
        sum2=0
        for i in range(1,n+1):
            if i%m != 0:
                sum1+=i
            else:
                sum2+=i

        return sum1-sum2
```

2812 Find the Maximum Achievable Number (link)

Description

Given two integers, `num` and `t`. A **number** `x` is **achievable** if it can become equal to `num` after applying the following operation **at most** `t` times:

- Increase or decrease `x` by `1`, and *simultaneously* increase or decrease `num` by `1`.

Return the **maximum** possible value of `x`.

Example 1:

Input: num = 4, t = 1

Output: 6

Explanation:

Apply the following operation once to make the maximum achievable number equal to `num`:

- Decrease the maximum achievable number by 1, and increase `num` by 1.

Example 2:

Input: num = 3, t = 2

Output: 7

Explanation:

Apply the following operation twice to make the maximum achievable number equal to `num`:

- Decrease the maximum achievable number by 1, and increase `num` by 1.

Constraints:

- `1 <= num, t <= 50`

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def theMaximumAchievableX(self, num: int, t: int) -> int:
        new=num+t+t
        return new
```

782 Jewels and Stones (link)

Description

You're given strings `jewels` representing the types of stones that are jewels, and `stones` representing the stones you have. Each character in `stones` is a type of stone you have. You want to know how many of the stones you have are also jewels.

Letters are case sensitive, so `"a"` is considered a different type of stone from `"A"`.

Example 1:

```
Input: jewels = "aA", stones = "aAAbbbb"
Output: 3
```

Example 2:

```
Input: jewels = "z", stones = "ZZ"
Output: 0
```

Constraints:

- `1 <= jewels.length, stones.length <= 50`
- `jewels` and `stones` consist of only English letters.
- All the characters of `jewels` are **unique**.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def numJewelsInStones(self, jewels: str, stones: str) -> int:
        count = 0
        for i in stones:
            if i in jewels:
                count+=1

        return count
```

387 First Unique Character in a String (link)

Description

Given a string `s`, find the **first** non-repeating character in it and return its index. If it **does not** exist, return `-1`.

Example 1:

Input: s = "leetcode"

Output: 0

Explanation:

The character `'l'` at index 0 is the first character that does not occur at any other index.

Example 2:

Input: s = "loveleetcode"

Output: 2

Example 3:

Input: s = "aabb"

Output: -1

Constraints:

- `1 <= s.length <= 10^5`
- `s` consists of only lowercase English letters.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def firstUniqChar(self, s: str) -> int:
        for i in s:
            if s.rindex(i) ==s.index(i):
                return s.index(i)
        return -1
```

1205 Defanging an IP Address (link)

Description

Given a valid (IPv4) IP `address`, return a defanged version of that IP address.

A *defanged IP address* replaces every period `"."` with `"[.]"`.

Example 1:

```
Input: address = "1.1.1.1"
Output: "1[.]1[.]1[.]1"
```

Example 2:

```
Input: address = "255.100.50.0"
Output: "255[.]100[.]50[.]0"
```

Constraints:

- The given `address` is a valid IPv4 address.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def defangIPaddr(self, address: str) -> str:
        s=""
        for i in address:
            if i == ".":
                s = address.replace(".","[.]")
        return s
```

3379 Score of a String (link)

Description

You are given a string `s`. The **score** of a string is defined as the sum of the absolute difference between the **ASCII** values of adjacent characters.

Return the **score** of `s`.

Example 1:

Input: s = "hello"

Output: 13

Explanation:

The **ASCII** values of the characters in `s` are: `'h' = 104`, `'e' = 101`, `'l' = 108`, `'o' = 111`. So, the score of `s` would be `|104 - 101| + |101 - 108| + |108 - 108| + |108 - 111| = 3 + 7 + 0 + 3 = 13`.

Example 2:

Input: s = "zaz"

Output: 50

Explanation:

The **ASCII** values of the characters in `s` are: `'z' = 122`, `'a' = 97`. So, the score of `s` would be `|122 - 97| + |97 - 122| = 25 + 25 = 50`.

Constraints:

- `2 <= s.length <= 100`
- `s` consists only of lowercase English letters.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def scoreOfString(self, s: str) -> int:
        sum=0
        for i in range(0,len(s)-1):
            v=ord(s[i])
            for j in range(i+1,len(s)):
                u=ord(s[j])
                z=abs(v-u)
                break
            sum=sum+z
        return sum
```

283 Move Zeroes (link)

Description

Given an integer array `nums`, move all `0`'s to the end of it while maintaining the relative order of the non-zero elements.

Note that you must do this in-place without making a copy of the array.

Example 1:

```
Input: nums = [0,1,0,3,12]
Output: [1,3,12,0,0]
```

Example 2:

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

Constraints:

- `1 <= nums.length <= 10^4`
- `-2^31 <= nums[i] <= 2^31 - 1`

Follow up: Could you minimize the total number of operations done?

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def moveZeroes(self, nums: List[int]) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        count=0
        for i in range(0,len(nums)):
            if nums[i] == 0:
                count+=1
        for i in range(0,count):
            nums.remove(0)
            nums.append(0)
```

268 Missing Number (link)

Description

Given an array `nums` containing `n` distinct numbers in the range `[0, n]`, return *the only number in the range that is missing from the array.*

Example 1:

Input: nums = [3,0,1]

Output: 2

Explanation:

`n = 3` since there are 3 numbers, so all numbers are in the range `[0,3]`. 2 is the missing number in the range since it does not appear in `nums`.

Example 2:

Input: nums = [0,1]

Output: 2

Explanation:

`n = 2` since there are 2 numbers, so all numbers are in the range `[0,2]`. 2 is the missing number in the range since it does not appear in `nums`.

Example 3:

Input: nums = [9,6,4,2,3,5,7,0,1]

Output: 8

Explanation:

`n = 9` since there are 9 numbers, so all numbers are in the range `[0,9]`. 8 is the missing number in the range since it does not appear in `nums`.

Constraints:

- `n == nums.length`
- `1 <= n <= 10^4`
- `0 <= nums[i] <= n`
- All the numbers of `nums` are **unique**.

Follow up: Could you implement a solution using only `O(1)` extra space complexity and `O(n)` runtime complexity?

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def missingNumber(self, nums: List[int]) -> int:
        s=sorted(nums)
        if s[0]!=0:
            return int(0)
        else:
            for i in range(1,len(s)):
                if s[i-1] != s[i]-1:
                    if s[i] >0:
                        return s[i]-1

        return len(s)
```

217 Contains Duplicate (link)

Description

Given an integer array `nums`, return `true` if any value appears **at least twice** in the array, and return `false` if every element is distinct.

Example 1:

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

Output: true

Explanation:

The element 1 occurs at the indices 0 and 3.

Example 2:

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

Output: false

Explanation:

All elements are distinct.

Example 3:

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

Output: true

Constraints:

- `1 <= nums.length <= 10^5`
- `-10^9 <= nums[i] <= 10^9`

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def containsDuplicate(self, nums: List[int]) -> bool:
        counts = {}
        for i in nums:
            if i in counts:
                return True
            else:
                counts[i]=1

        return False
```

263 Ugly Number (link)

Description

An **ugly number** is a *positive* integer which does not have a prime factor other than 2, 3, and 5.

Given an integer `n`, return `true` *if* `n` *is an* ***ugly number***.

Example 1:

```
Input: n = 6
Output: true
Explanation: 6 = 2 × 3
```

Example 2:

```
Input: n = 1
Output: true
Explanation: 1 has no prime factors.
```

Example 3:

```
Input: n = 14
Output: false
Explanation: 14 is not ugly since it includes the prime factor 7.
```

Constraints:

- -2^{31} <= n <= 2^{31} - 1

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def isUgly(self, n: int) -> bool:
        if n<=0:
            return False

        while n%2==0:
            n/=2
        while n%3==0:
            n/=3
        while n%5==0:
            n/=5

        return n==1
```

258 Add Digits (link)

Description

Given an integer `num`, repeatedly add all its digits until the result has only one digit, and return it.

Example 1:

```
Input: num = 38
Output: 2
Explanation: The process is
38 --> 3 + 8 --> 11
11 --> 1 + 1 --> 2
Since 2 has only one digit, return it.
```

Example 2:

```
Input: num = 0
Output: 0
```

Constraints:

- $0 <= num <= 2^{31} - 1$

Follow up: Could you do it without any loop/recursion in $O(1)$ runtime?

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def addDigits(self, num: int) -> int:
        l=list(str(num))
        s=[]
        if len(l) <= 1:
            return num
        else:
            while len(l)!=1:
                sum=0
                for i in l:
                    sum=sum+int(i)
                s.clear()
                s.append(str(sum))
                l=list(str(sum))

        return int(s[0])
```

242 Valid Anagram (link)

Description

Given two strings `s` and `t`, return `true` if `t` is an anagram of `s`, and `false` otherwise.

Example 1:

Input: s = "anagram", t = "nagaram"

Output: true

Example 2:

Input: s = "rat", t = "car"

Output: false

Constraints:

- `1 <= s.length, t.length <= 5 * 10^4`
- `s` and `t` consist of lowercase English letters.

Follow up: What if the inputs contain Unicode characters? How would you adapt your solution to such a case?

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        if len(s) != len(t):
            return False
        og=[]
        for letter in s:
            og.append(letter)

        counts={}
        counts2={}

        for i in range(0,len(t)):
            if t[i] in counts:
                counts[t[i]]+=1
            else:
                counts[t[i]]=1
        for i in range(0,len(og)):
            if og[i] in counts2:
                counts2[og[i]]+=1
            else:
                counts2[og[i]]=1

        if counts == counts2:
            return True
        else:
            return False
```

169 Majority Element (link)

Description

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
```

Constraints:

- `n == nums.length`
- `1 <= n <= 5 * 10^4`
- `-10^9 <= nums[i] <= 10^9`

Follow-up: Could you solve the problem in linear time and in `O(1)` space?

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def majorityElement(self, nums: List[int]) -> int:
        counts={}
        l=[]

        for i in range(0,len(nums)):
            l.append(nums[i])

        for i in range(0,len(l)):
            if l[i] in counts:
                counts[l[i]]+=1
            else:
                counts[l[i]]=1

        maxi = max(counts, key=counts.get)
        return maxi
```

191 Number of 1 Bits (link)

Description

Given a positive integer `n`, write a function that returns the number of set bits in its binary representation (also known as the Hamming weight).

Example 1:

Input: n = 11

Output: 3

Explanation:

The input binary string **1011** has a total of three set bits.

Example 2:

Input: n = 128

Output: 1

Explanation:

The input binary string **10000000** has a total of one set bit.

Example 3:

Input: n = 2147483645

Output: 30

Explanation:

The input binary string **11111111111111111111111111111101** has a total of thirty set bits.

Constraints:

- `1 <= n <= 2^31 - 1`

Follow up: If this function is called many times, how would you optimize it?

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def hammingWeight(self, n: int) -> int:
        v= bin(n)
        s=str(v)
        b=s[2:]
        count=0
        for i in b:
            if i=='1':
                count+=1

        return count
```

136 Single Number (link)

Description

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

You must implement a solution with a linear runtime complexity and use only constant extra space.

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

Constraints:

- `1 <= nums.length <= 3 * 10^4`
- `-3 * 10^4 <= nums[i] <= 3 * 10^4`
- Each element in the array appears twice except for one element which appears only once.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def singleNumber(self, nums: List[int]) -> int:
        a=sorted(nums)
        unique =[]
        for item in a:
            if a.count(item)==1:
                unique.append(item)

        return unique[0]
```

125 Valid Palindrome (link)

Description

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string `s`, return `true` *if it is a* ***palindrome****, or* `false` *otherwise*.

Example 1:

```
Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.
```

Example 2:

```
Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.
```

Example 3:

```
Input: s = " "
Output: true
Explanation: s is an empty string "" after removing non-alphanumeric characters.
Since an empty string reads the same forward and backward, it is a palindrome.
```

Constraints:

- `1 <= s.length <= 2 * 10^5`
- `s` consists only of printable ASCII characters.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def isPalindrome(self, s: str) -> bool:
        if s == []:
            return True
        g=[]
        for i in range(0,len(s)):
            if s[i].isalpha() == True or s[i].isnumeric()==True:
                g.append(s[i])
        h=[word.lower() for word in g]
        k=h[::-1]
        y=[]
        for i in range(min(len(h), len(k))):
            if h[i]==k[i]:
                y.append(k[i])


        if len(y)==len(g) and s!=[]:
            return True
        else:
            return False
```

67 Add Binary (link)

Description

Given two binary strings `a` and `b`, return *their sum as a binary string*.

Example 1:

```
Input: a = "11", b = "1"
Output: "100"
```

Example 2:

```
Input: a = "1010", b = "1011"
Output: "10101"
```

Constraints:

- `1 <= a.length, b.length <= 10^4`
- `a` and `b` consist only of `'0'` or `'1'` characters.
- Each string does not contain leading zeros except for the zero itself.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def addBinary(self, a: str, b: str) -> str:
        c=0
        d=0
        i=0
        e=a[::-1]
        f=b[::-1]
        while i<len(e):
            if e[i] == '1':
                c=c+(2**i)*1
            elif e[i] == '0':
                c+=0
            i+=1
        i=0
        while i<len(f):
            if f[i] == '1':
                d=d+(2**i)*1
            elif f[i] == '0':
                d+=0
            i+=1


        sum=c+d
        return bin(sum)[2:]
```

66 Plus One (link)

Description

You are given a **large integer** represented as an integer array `digits`, where each `digits[i]` is the i^{th} digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading `0`'s.

Increment the large integer by one and return *the resulting array of digits*.

Example 1:

```
Input: digits = [1,2,3]
Output: [1,2,4]
Explanation: The array represents the integer 123.
Incrementing by one gives 123 + 1 = 124.
Thus, the result should be [1,2,4].
```

Example 2:

```
Input: digits = [4,3,2,1]
Output: [4,3,2,2]
Explanation: The array represents the integer 4321.
Incrementing by one gives 4321 + 1 = 4322.
Thus, the result should be [4,3,2,2].
```

Example 3:

```
Input: digits = [9]
Output: [1,0]
Explanation: The array represents the integer 9.
Incrementing by one gives 9 + 1 = 10.
Thus, the result should be [1,0].
```

Constraints:

- `1 <= digits.length <= 100`
- `0 <= digits[i] <= 9`
- `digits` does not contain any leading `0`'s.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def plusOne(self, digits: List[int]) -> List[int]:
        new = ""
        b=[]
        for i in digits:
            b.append(str(i))
        new = ''.join(b)
        inte = int(new)
        inte +=1
        digits.clear()
        new=str(inte)
        for i in range(0, len(new)):
            a=int(new[i])
            digits.append(a)

        return digits
```

35 Search Insert Position (link)

Description

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.

You must write an algorithm with `O(log n)` runtime complexity.

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
```

Constraints:

- `1 <= nums.length <= 10^4`
- `-10^4 <= nums[i] <= 10^4`
- `nums` contains **distinct** values sorted in **ascending** order.
- `-10^4 <= target <= 10^4`

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def searchInsert(self, nums: List[int], target: int) -> int:
        for i in range(len(nums)):
            if target == nums[i]:
                return i
            if i>0:
                if target == (nums[i-1])+1:
                    return i
                elif (target!=nums[i-1]) and (target < (nums[i]-1)):
                    return i-1
            elif i==0 and target < nums[0]:
                return 0

        return len(nums)
```

28 Find the Index of the First Occurrence in a String (link)

Description

Given two strings `needle` and `haystack`, return the index of the first occurrence of `needle` in `haystack`, or `-1` if `needle` is not part of `haystack`.

Example 1:

```
Input: haystack = "sadbutsad", needle = "sad"
Output: 0
Explanation: "sad" occurs at index 0 and 6.
The first occurrence is at index 0, so we return 0.
```

Example 2:

```
Input: haystack = "leetcode", needle = "leeto"
Output: -1
Explanation: "leeto" did not occur in "leetcode", so we return -1.
```

Constraints:

- `1 <= haystack.length, needle.length <= 10^4`
- `haystack` and `needle` consist of only lowercase English characters.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:

        i=0
        while i <= len(haystack) -len(needle):
            y=0
            while y < len(needle) and haystack[i+y] == needle[y]:
                y+=1
                if y==len(needle):
                    return i
            i+=1

        return -1
```

58 Length of Last Word (link)

Description

Given a string `s` consisting of words and spaces, return *the length of the **last** word in the string.*

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

Example 1:

```
Input: s = "Hello World"
Output: 5
Explanation: The last word is "World" with length 5.
```

Example 2:

```
Input: s = "   fly me   to   the moon  "
Output: 4
Explanation: The last word is "moon" with length 4.
```

Example 3:

```
Input: s = "luffy is still joyboy"
Output: 6
Explanation: The last word is "joyboy" with length 6.
```

Constraints:

- `1 <= s.length <= 10^4`
- `s` consists of only English letters and spaces `' '`.
- There will be at least one word in `s`.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def lengthOfLastWord(self, s: str) -> int:
        k=[]
        a=s[::-1]
        i=0

        while i< len(a):
            if a[i] ==" ":
                if k!=[]:
                   break
                i+=1
            else:
                k.append(a[i])
                i+=1

        return len(k)
```

27 Remove Element (link)

Description

Given an integer array `nums` and an integer `val`, remove all occurrences of `val` in `nums` **in-place**. The order of the elements may be changed. Then return *the number of elements in* `nums` *which are not equal to* `val`.

Consider the number of elements in `nums` which are not equal to `val` be `k`, to get accepted, you need to do the following things:

- Change the array `nums` such that the first `k` elements of `nums` contain the elements which are not equal to `val`. The remaining elements of `nums` are not important as well as the size of `nums`.
- Return `k`.

Custom Judge:

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int val = ...; // Value to remove
int[] expectedNums = [...]; // The expected answer with correct length.
                            // It is sorted with no values equaling val.

int k = removeElement(nums, val); // Calls your implementation

assert k == expectedNums.length;
sort(nums, 0, k); // Sort the first k elements of nums
for (int i = 0; i < actualLength; i++) {
    assert nums[i] == expectedNums[i];
}
```

If all assertions pass, then your solution will be **accepted**.

Example 1:

```
Input: nums = [3,2,2,3], val = 3
Output: 2, nums = [2,2,_,_]
Explanation: Your function should return k = 2, with the first two elements of nums being
It does not matter what you leave beyond the returned k (hence they are underscores).
```

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 k = 5, with the first five elements of nums cont
```

```
Note that the five elements can be returned in any order.
It does not matter what you leave beyond the returned k (hence they are underscores).
```

Constraints:

- 0 <= nums.length <= 100
- 0 <= nums[i] <= 50
- 0 <= val <= 100

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def removeElement(self, nums: List[int], val: int) -> int:
        k=[]
        i=0
        while i < len(nums):
            if val == nums[i]:
                k.append(nums[i])
                nums.pop(i)
            else:
                i+=1
        return len(nums)
```

26 Remove Duplicates from Sorted Array (link)

Description

Given an integer array `nums` sorted in **non-decreasing order**, remove the duplicates **in-place** such that each unique element appears only **once**. The **relative order** of the elements should be kept the **same**. Then return *the number of unique elements in* `nums`.

Consider the number of unique elements of `nums` to be `k`, to get accepted, you need to do the following things:

- Change the array `nums` such that the first `k` elements of `nums` contain the unique elements in the order they were present in `nums` initially. The remaining elements of `nums` are not important as well as the size of `nums`.
- Return `k`.

Custom Judge:

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length

int k = removeDuplicates(nums); // Calls your implementation

assert k == expectedNums.length;
for (int i = 0; i < k; i++) {
    assert nums[i] == expectedNums[i];
}
```

If all assertions pass, then your solution will be **accepted**.

Example 1:

```
Input: nums = [1,1,2]
Output: 2, nums = [1,2,_]
Explanation: Your function should return k = 2, with the first two elements of nums being
It does not matter what you leave beyond the returned k (hence they are underscores).
```

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 k = 5, with the first five elements of nums beir
It does not matter what you leave beyond the returned k (hence they are underscores).
```

Constraints:

- `1 <= nums.length <= 3 * 10^4`
- `-100 <= nums[i] <= 100`
- `nums` is sorted in **non-decreasing** order.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def removeDuplicates(self, nums: List[int]) -> int:
        prev=1

        for i in range(1, len(nums)):
            if nums[i] != nums[i-1]:
                nums[prev]=nums[i]
                prev+=1

        return prev
```

13 Roman to Integer (link)

Description

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 ones 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
Explanation: III = 3.
```

Example 2:

```
Input: s = "LVIII"
Output: 58
Explanation: L = 50, V= 5, III = 3.
```

Example 3:

```
Input: s = "MCMXCIV"
Output: 1994
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.
```

Constraints:

- `1 <= s.length <= 15`
- `s` contains only the characters `('I', 'V', 'X', 'L', 'C', 'D', 'M')`.
- It is **guaranteed** that `s` is a valid roman numeral in the range `[1, 3999]`.

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def romanToInt(self, s: str) -> int:
        roman={'I' : 1,
        'V' : 5,
        'X' : 10,
        'L' : 50,
        'C' : 100,
        'D' : 500,
        'M' : 1000
        }
        result=0
        prev=0
        for i in s[::-1]:
            if roman[i] >= prev:
                result+=roman[i]
            else:
                result-=roman[i]
            prev=roman[i]
        return result
```

9 Palindrome Number (link)

Description

Given an integer `x`, return `true` *if* `x` *is a* ***palindrome****, and* `false` *otherwise*.

Example 1:

```
Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
```

Example 2:

```
Input: x = -121
Output: false
Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Ther
```

Example 3:

```
Input: x = 10
Output: false
Explanation: Reads 01 from right to left. Therefore it is not a palindrome.
```

Constraints:

- -2^{31} <= x <= 2^{31} - 1

Follow up: Could you solve it without converting the integer to a string?

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def isPalindrome(self, x: int) -> bool:
        a=str(x)
        if a=="":
            return
        if a[:] == a[::-1]:
            return True
        else:
            return False
```

1 Two Sum (link)

Description

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.

Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

Example 2:

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

Example 3:

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

Constraints:

- $2 <= nums.length <= 10^4$
- $-10^9 <= nums[i] <= 10^9$
- $-10^9 <= target <= 10^9$
- **Only one valid answer exists.**

Follow-up: Can you come up with an algorithm that is less than $O(n^2)$ time complexity?

(scroll down for solution)

Solution

Language: python3

Status: Accepted

```python
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        x=0
        while x < len(nums):
            for y in range(len(nums)):
                if x!=y:
                    if nums[x]+nums[y] == target:
                        return x,y
            x=x+1
```