

# LeetCode - Easy



- [0009 - Palindrome Number](#)
- [0013 - Roman to Integer](#)
- [0014 - Longest Common Prefix](#)
- [0020 - Valid Parentheses](#)
- [0021 - Merge Two Sorted Lists](#)
- [0026 - Remove Duplicates from Sorted Array](#)
- [0027 - Remove Element](#)
- [0028 - Find the Index of the First Occurrence in a String](#)
- [0035 - Search Insert Position](#)
- [0058 - Length of Last Word](#)
- [0066 - Plus One](#)
- [0088 - Merge Sorted Array](#)

# 0009 - Palindrome Number



 121  121 abcba 






  
 / 2



 Python Code  / 2 

```
class Solution:
    def isPalindrome(self, x: int) -> bool:
        if x < 0:
            return False

        number_str = str(x)
        result = False
        for i in range(0, int(len(number_str) / 2)): #  0 <= x <= 9  len / 2  0
            if number_str[i] == number_str[len(number_str) - 1 - i]:
                result = True
            else:
                result = False
        return result
```

 list list 

```

class Solution:
    def isPalindrome(self, x: int) -> bool:
        if x < 0:
            return False

        number_str = str(x)
        reverse_num = []
        for i in range(0, len(number_str), 1):
            reverse_num.append(number_str[len(number_str) - 1 - i])

        return "".join(reverse_num) == number_str

```

11111111

Python 11111111

- Slice

```

class Solution:
    def isPalindrome(self, x: int) -> bool:
        num_str = str(x)
        reversed_num = num_str[::-1] # Reverse slice
        return reversed_num == num_str

```

Java 111111111111

String 111111

Char Array 111111111111

StringBuilder (11111111)

):

```

class Solution {
    public boolean isPalindrome(int x) {
        String numStr = String.valueOf(x);
        char[] numChrArray = numStr.toCharArray();
        StringBuilder builder = new StringBuilder();

        for(int i = 0; i < numChrArray.length; i++){
            builder.append(numChrArray[numChrArray.length - 1 - i]);
        }

        return numStr.equals(builder.toString());
    }
}

```

111111

String.charAt() 11111111

Char Array:

```

class Solution {
    public boolean isPalindrome(int x) {
        String numStr = String.valueOf(x);
        StringBuilder reverseNum = new StringBuilder();

        for(int i = 0; i < numStr.length(); i++){
            reverseNum.append(numStr.charAt(numStr.length() - 1 - i));
        }

        return numStr.equals(reverseNum.toString());
    }
}

```



- 0
- 
- (% 10)
- 
- 
- (x 10)
- 
- 
- 



Java

```

class Solution {
    public boolean isPalindrome(int x) {
        if(x < 0){
            return false;
        }

        long temp = x;

```

```









long reversed = 0;

while(temp != 0){
    long y = temp % 10;
    reversed = reversed * 10 + y;
    temp /= 10;
}

return x == reversed;
}
}






```








-  10  10  0 
-   

-  10 



```

class Solution {
public boolean isPalindrome(int x) {
    if((x < 0) || (x != 0 && x % 10 == 0)){ //  x  0  10  0 
        return false;
    }
    long reversed = 0;

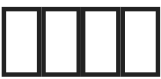
    while(x > reversed){ //  x  10  reversed loop
        long y = x % 10;
        reversed = reversed * 10 + y;
        x /= 10;
    }

    return (x == reversed) || (x == reversed / 10); // reversed / 10
    
}
}

```



# 0013 - Roman to Integer








Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000



- 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.



- 
  -  I X C 
  -  total  




```
class Solution(object):
    def romanToInt(self, s):
        """
        :type s: str
        :rtype: int
        """
        total = 0
        formar_char = "
```

```

if char == 'I':
    total += 1
elif char == 'V':
    if format_char == 'I':
        total += 3 # 11 4 111111 1 1111 XD111111
    else:
        total += 5
elif char == 'X':
    if format_char == 'I':
        total += 8
    else:
        total += 10
elif char == 'L':
    if format_char == 'X':
        total += 30
    else:
        total += 50
elif char == 'C':
    if format_char == 'X':
        total += 80
    else:
        total += 100
elif char == 'D':
    if format_char == 'C':
        total += 300
    else:
        total += 500
elif char == 'M':
    if format_char == 'C':
        total += 800
    else:
        total += 1000
format_char = char
return total

```





if-else      yandev...XD  
Map      Map      iterator      for

```
class Solution(object):
    def romanToInt(self, s):
        """
        :type s: str
        :rtype: int
        """
        total = 0

        roman_map = {
            'I': 1,
            'V': 5,
            'X': 10,
            'L': 50,
            'C': 100,
            'D': 500,
            'M': 1000
        }

        for i in range(0, len(s)):
            if i != 0 and roman_map[s[i - 1]] < roman_map[s[i]]:
                # 
                total = total + roman_map[s[i]] - (roman_map[s[i - 1]] * 2)
            else:
                total += roman_map[s[i]]

        return total
```

str.replace() ...

Java

```
class Solution {
    public int romanToInt(String s) {
        HashMap<Character, Integer> romanMap = new HashMap<>();
        romanMap.put('I', 1);
        romanMap.put('V', 5);
        romanMap.put('X', 10);
```

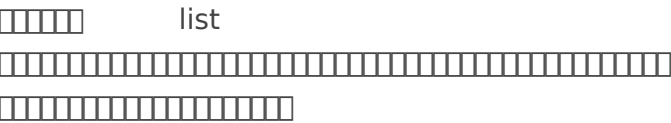
```
romanMap.put('L', 50);
romanMap.put('C', 100);
romanMap.put('D', 500);
romanMap.put('M', 1000);

int total = 0;
for(int count = 0; count < s.length(); count++){
    if(count != 0 && romanMap.get(s.charAt(count)) > romanMap.get(s.charAt(count - 1))){
        total -= romanMap.get(s.charAt(count - 1)) * 2;
    }

    total += romanMap.get(s.charAt(count));
}

return total;
}
}
```

# 0014 - Longest Common Prefix



```
class Solution(object):
    def longestCommonPrefix(self, strs):
        """
        :type strs: List[str]
        :rtype: str
        """

        result = ""

        # [flower, flow, flight, dog]
        sorted_list = sorted(strs)
```

```
first = sorted_list[0]
last = sorted_list[-1]
```

```
for i in range(0, min(len(first), len(last))):
    if first[i] != last[i]:
        return result
    result += first[i]
```

```
return result
```

|||||

Python ||||

sorted() || Java |||

Array.sort() ||

||| Java code:

```
class Solution {
    public String longestCommonPrefix(String[] strs) {
        ArrayList<String> arraylist = new ArrayList<>(Arrays.asList(strs));
        arraylist.sort(Comparator.naturalOrder());

        StringBuilder samePart = new StringBuilder();

        String first = arraylist.getFirst();
        String last = arraylist.getLast();
        int count = 0;

        while(count < Math.min(first.length(), last.length())){
            if(first.charAt(count) == last.charAt(count)){
                samePart.append(first.charAt(count));
            } else {
                return samePart.toString();
            }
            count++;
        }

        return samePart.toString();
    }
}
```

# 0020 - Valid Parentheses



□□□□□□□□      '(', ')', '{', '}', '[', and ']' □□

□□□□□□      "()[]{}" □□□□□□□□      "{}()" □□□□□□□□      :Kappa:



□□      Stack □□□□□□□□      Stack □□□□□□□□□□□□□□□□      Stack  
□□□□□□□□      Stack □□□□□□□□

```
class Solution(object):
    def isValid(self, s):
        """
        :type s: str
        :rtype: bool
        """
        if len(s) < 2:
            return False

        barket_stack = []

        for ch in s:
            if len(barket_stack) == 0:
                barket_stack.append(ch)
                continue

            temp = barket_stack.pop()

            if temp == '(' and ch == ')':
                continue
            if temp == '[' and ch == ']':
```

```

        continue
    if temp == '{' and ch == '}':
        continue
    barket_stack.append(temp)
    barket_stack.append(ch)

return len(barket_stack) == 0

```



slice 

```
(list[-1])
```

□ □ □ □ □ □ □

if

## Python

# 0021 - Merge Two Sorted Lists



|||||

LinkedList |||||

LinkedList |||||



|||||

Timeout |||||

Code:

```
# Definition for singly-linked list.
class ListNode(object):
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution(object):
    def mergeTwoLists(self, list1, list2):
        """
        :type list1: Optional[ListNode]
        :type list2: Optional[ListNode]
        :rtype: Optional[ListNode]
        """

        head = ListNode()
        current = head

        while list1 != None and list2 != None:
            if list1.val <= list2.val:
                current.next = list1
                temp = list1.next
```

```

        current = list1
        list1 = temp
    else:
        current.next = list2
        temp = list2.next
        current = list2
        list2 = temp

if list1 != None:
    current.next = list1

if list2 != None:
    current.next = list2

return head.next

```

Diagram illustrating the structure of a linked list node (ListNode) and its pointer field:

The diagram shows a horizontal sequence of boxes representing the structure of a linked list node. The first box is labeled "ListNode" and contains four smaller boxes, representing data fields. This is followed by a long box labeled "ListNode" containing 16 smaller boxes, representing the pointer field. The sequence ends with a box labeled "ListNode" containing four smaller boxes, representing the next pointer field.

- □□ list1 □□□□□□□□                  list2□□    list □ next □□    list1
- □□□□         list1 □□ list1 □□□□□□□□                  list1.next
- □□□□         list □ next □□□    list2□□□□□                  list2 □□ list2.next
- □□□□□□□                  list1 □ list2 □□□□□□                  None (null)
- □□ list1 □ list2 □□□□      LinkedList □ Node □□□□□□□□□□□□                  None (null)  
       □□□□□□                  ListNode □□□□□□□□□□□□

LinkedList



# 0026 - Remove Duplicates from Sorted Array



K



case:

**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` being 0, 1, 2, 3, and 4 respectively. It does not matter what you leave beyond the returned  $k$  (hence they are underscores).



dupe



K

count 

Java

```
class Solution {  
    public int removeDuplicates(int[] nums) {  
        int dupeCount = 0;  
        int count = 0;  
  
        //   dupe [ ][ ][ ][ ][ ][ ][ ][ ]  
        while(count < nums.length - 1 - dupeCount){  
            if(nums[count] == nums[count + 1]){  
                // [ ][ ][ ][ ][ ][ ][ ][ ]          dupe [] + 1  
                for(int j = count; j < nums.length - 1 - dupeCount; j++){  
                    nums[j] = nums[j + 1];  
                }  
            }  
        }  
    }  
}
```

```

        dupeCount += 1;
    } else {
        count += 1;
    }
}

// 1 out-of-bound
// + 1
return count + 1;
}
}

```

...

... LeetCode

LeetCode

unique index  
 unique index

```

class Solution {
    public int removeDuplicates(int[] nums) {
        int uniqueIndex = 1;

        for(int i = 1; i < nums.length; i++){
            // uniqueIndex
            if(nums[i] != nums[i - 1]){
                nums[uniqueIndex] = nums[i];
                uniqueIndex++;
            }
        }

        return uniqueIndex;
    }
}

```

...

...

# 0027 - Remove Element



**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 containing 0, 0, 1, 3, and 4. 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).



0026


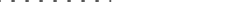


```
class Solution {
    public int removeElement(int[] nums, int val) {
        int nonTargetIndex = 0;

        for(int i = 0; i < nums.length; i++){
            if(nums[i] != val){
                nums[nonTargetIndex] = nums[i];
                nonTargetIndex += 1;
            }
        }

        return nonTargetIndex;
    }
}
```

- 

Target Index

- 
- 
- 
- 

Target Index □ Target Index

# 0028 - Find the Index of the First Occurrence in a String



index

mississippi index 4



Time out

```
class Solution {
    public int strStr(String haystack, String needle) {
        int sameCount = 0;
        int lastSuccess = 0;
        int count = 0;

        while(count < haystack.length()){
            if(sameCount == needle.length()){
                return count - sameCount;
            }

            if(haystack.charAt(count) != needle.charAt(sameCount)){
                sameCount = 0;
                if(lastSuccess != 0){
                    count = lastSuccess;
                    continue;
                }
            } else {
                sameCount++;
                count++;
            }
        }

        return -1;
    }
}
```

```

        sameCount++;
        lastSuccess = count;
    }
    count++;
}


if(sameCount == needle.length() - 1){
    return haystack.length() - sameCount;
} else {
    return -1;
}
}
}




```






```

class Solution {
    public int strStr(String haystack, String needle) {
        int index = 0;

        // 
        if(haystack.length() < needle.length()){
            return -1;
        }

        // 
        //  <= 
        for(index = 0; index <= haystack.length() - needle.length(); index++){
            int subCount = 0;

            // 
            for(subCount = 0; subCount < needle.length(); subCount++){
                if(haystack.charAt(index + subCount) != needle.charAt(subCount)){
                    break;
                }
            }

            //  Count  index
            if(subCount == needle.length()){
                return index;
            }
        }
    }
}

```

```
}  
}
```

```
// 
```

```
return -1;
```

```
}  
}
```

# 0035 - Search Insert Position



Index



Index

LeetCode

## 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



Binary Search



Index +

1 +1:

```
class Solution {
    public int searchInsert(int[] nums, int target) {
        // Binary search
        int left = 0;
        int right = nums.length - 1;
        int mid = 0;
```



[illegible]

# 0058 - Length of Last Word




**Input:** s = " fly me to the moon "

**Output:** 4

**Explanation:** The last word is "moon" with length 4.



index = 1   
reset

index 



```
class Solution {
    public int lengthOfLastWord(String s) {
        int count = 1;
        int index = 1;
        int spaceCount = 0;

        if(s.length() < 1){
            return 0;
        }

        while(index < s.length()){
            if(s.charAt(index - 1) == ' ' && s.charAt(index) != ' '){
                count = 0;
            }
        }
    }
}
```

```
    if(s.charAt(index) != ' '){  
        count++;  
    }  
  
    index++;  
}  
  
return count;  
}  
}
```

# 0066 - Plus One



□□□□□□□□□□ +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].



**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].



- □□□□□□ +1 □□□□□□ 10 □□□□ +1 □□□□□□□□
- □□□□□□□□□□
  - □□□□□
  - □□□□□□□□□□ -> □□□□□□□□□□

□□□□□□□□□□      ArrayList □□□□□□□□      `new int[size + 1]`  
□□□□□□□□□□□□□□□□□□□□

```
class Solution {  
    public int[] plusOne(int[] digits) {  
        int lastIndex = digits.length - 1;
```

```
int lastNum = digits[lastIndex];

lastNum += 1;

if(lastNum < 10){
    digits[lastIndex] = lastNum;
    return digits;
} else {
    digits[lastIndex] = 0;
    int carry = 1;

    for(int i = lastIndex - 1; i >= 0; i--){
        lastNum = digits[i] + carry;

        if(lastNum < 10){
            digits[i] = lastNum;
            carry = 0;
            break;
        } else {
            digits[i] = 0;
        }
    }

    if(carry > 0){
        int[] newArr = new int[digits.length + 1];
        newArr[0] = carry;

        for(int i = 1; i < newArr.length; i++){
            newArr[i] = digits[i - 1];
        }

        return newArr;
    } else {
        return digits;
    }
}
}
```



# 0088 - Merge Sorted Array



□□□□□ ... □□□□□□□□□□□□□□

□□□□□

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

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

**Explanation:** The arrays we are merging are [1,2,3] and [2,5,6].

The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.

□□□□□□□□

nums1 □□□    sort □□



## Code

□□□□□□□□□□□□□□□□□□□□□□□□

XD

```
class Solution(object):
    def merge(self, nums1, m, nums2, n):
        """
        :type nums1: List[int]
        :type m: int
        :type nums2: List[int]
        :type n: int
        :rtype: None Do not return anything, modify nums1 in-place instead.
        """

        for i in range(m, m+n):
            nums1[i] = nums2[m - i]

        # Sort: Bubble sort
        temp_index = 0
        for i in range(0, m+n - 1):
            for j in range(0, m+n - 1 - i):
```

```
if nums1[j] > nums1[j + 1]:  
    temp = nums1[j]  
    nums1[j] = nums1[j + 1]  
    nums1[j + 1] = temp
```

□□□□□□□□□□

...□