String

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

✔ 13 ~~Roman to Integer (/~~

```
Input: s = "PAYPALISHIRING", numRows = 4
Output: "PINALSIGYAHRPI"
Explanation:

P     I    N
A   L S  I G
Y A   H R
P     I
```

```
for ch in s:
    temp[ind].append(ch)
    if ind == numRows - 1 :
        sign = -1
    if ind == 0:
        sign = 1
    ind += sign
```

6 ~~ZigZag Conversion (/~~ Dec 29, 2018 6:26 PM, can skip

```
Input: "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
```

```
m = {}
start = 0
res = 0
for i in range(len(s)):
    if s[i] in m:
        start = max(start, m[s[i]]+1)
    m[s[i]] = i
    res = max(res, i - start+1)
return res
```

You can try the sliding window solution, em, not skip
Dec 29, 2018 6:29 PM

✔ 3 ~~Longest Substring Without Repeating Characters~~ ~~substring-without-repeating-characters)~~

~~12~~ ~~Integer to Roman~~

```
String M[] = {"", "M", "MM", "MMM"};
String C[] = {"", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"};
String X[] = {"", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"};
String I[] = {"", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"};
return M[num/1000] + C[(num%1000)/100] + X[(num%100)/10] + I[num%10];
```

Can skip
Dec 29, 2018 6:32 PM

Do 273 instead.

Redo

5 Longest Palindromic Substring

```
Input: "babad"
Output: "bab"
Note: "aba" is also a valid answer.
```

extend

38 Count and Say (/problems/count-and-say) Google 10

```
1.     1
2.     11
3.     21
4.     1211
5.     111221
6.     312211
7.     13112221
8.     1113213211
9.     31131211131221
10.    13211311123113112211
```

```
"""
1.     1
2.     11
3.     21
4.     1211
5.     111221
"""

prev = self. countAndSay(n-1)
count = 1
cur = prev[0]
res = ""
for i in range(1, len(prev)):
    if prev[i] == cur:
        count += 1
    else:
        res = res + "{}{}".format(count, cur)
        cur = prev[i]
        count = 1

res = res + "{}{}".format(count, cur)
return res
```

Almost bug free, can skip once or twice.

```
Input: "the sky is blue",
Output: "blue is sky the".
```

✔ 151 Reverse Words in a String (/problems/reverse-words-in-a-string)

67 Add Binary (/problems/add-binary)

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

```
if len(a) < len(b):
    return self.addBinary(b,a)
# assure len(a) >= len(b)

res = ""
extra = 0
i = len(a) - 1
j = len(b) - 1
while j >= 0:
    sum_ = extra + (int)(a[i]) + (int)(b[j])
    extra = sum_//2
    res = (str)(sum_%2) + res
    i -= 1
    j -= 1

while i >= 0:
    sum_ = extra + (int)(a[i])
    extra = sum_//2
    res = (str)(sum_%2) + res
    i-=1
if extra != 0:
    res = (str)(extra) + res
return res
```

8 ~~String to Integer (atoi)~~

1. The first char might be - or +, set it to sign
2. Num = num * 10 + cur until not digits,
3. If larger than integer.max, during 2, return max int
Or min int depend on sign. Basically can skip.

```
Input: "42"
Output: 42
```

```
Input: "   -42"
Output: -42
Explanation: The first n
             Then take a
```

```
Input: "4193 with words"
Output: 4193
Explanation: Conversion
```

```
Input: "words and 987"
Output: 0
Explanation: The first n
             digit or a
```

```
Input: "-91283472332"
Output: -2147483648
Explanation: The number
```

273 Integer to English Words

```
Input: 123
Output: "One Hundred Twenty Three"
```

```
Input: 12345
Output: "Twelve Thousand Three Hundred Forty Five"
```

```
Input: 1234567
Output: "One Million Two Hundred Thirty Four Thousand Five Hundred Sixty Seven"
```

```
Input: 1234567891
Output: "One Billion Two Hundred Thirty Four Million Five Hundred Sixty Seven T
```

```
String[] LESS_THAN_20 = {"", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"
String[] TENS = {"", "Ten", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ni
String[] THOUSANDS = {"", "Thousand", "Million", "Billion"};
```

1,234,567,891

```
int[] pos = new int[m + n];

for(int i = m - 1; i >= 0; i--) {
    for(int j = n - 1; j >= 0; j--) {
        int mul = (num1.charAt(i) - '0') * (num2.charAt(j) - '0');
        int p1 = i + j, p2 = i + j + 1;
        int sum = mul + pos[p2];

        pos[p1] += sum / 10;
        pos[p2] = (sum) % 10;
    }
}
```

43 Multiply Strings

Can skip.

if the chars are coming in as a data stream.

Use linked hash map is also ok, one pass not two pass, but both time and space are same, can skip.

✔ ~~387~~ ~~First Unique Character in a String~~ s = "leetcode" return 0. s = "loveleetcode", return 2. ~~er-in-a-~~

S = "ADOBECODEBANC", T = "ABC" : "BANC"

✔ 76 Minimum Window Substring (/problems/minimu ng)

336 Palindrome Pairs (/problems/palindrome-pairs)

```
Input: ["abcd","dcba","lls","s","sssll"]
Output: [[0,1],[1,0],[3,2],[2,4]]
Explanation: The palindromes are ["dcbaabcd","abcddcba","slls","llssssll"]
```

65 Valid Number (/problems/valid-number)

"0" => true
" 0.1 " => true
"abc" => false
"1 a" => false
"2e10" => true
" -90e3 " => true
" 1e" => false
"e3" => false
" 6e-1" => true
" 99e2.5 " => false
"53.5e93" => true
" --6 " => false
"-+3" => false
"95a54e53" => false

32 Longest Valid Parentheses

```
Input: ")()())"
Output: 4
Explanation: The longest valid parentheses substring is "()()"
```

647 Palindromic Substrings

```
Input: "aaa"
Output: 6
Explanation: Six palindromic strings: "a", "a", "a", "aa", "aa", "aaa".
```

类似 lc5 extend的思路。can skip

~~125~~ ~~Valid Palindrome~~

```
Input: "A man, a plan, a canal: Panama"
Output: true
```

1. lower() 2. Skip if not char or number until I == j.

30 Substring with Concatenation of All Words ith-concatenation-of-all-words) Facebook 3

```
Input:
  s = "barfoothefoobarman",
  words = ["foo","bar"]
Output: [0,9]
```

165 Compare Version Numbers

```
Input: version1 = "7.5.2.4", version2 = "7.5.3"
Output: -1
```

If `version1 > version2` return `1;` if `version1 < version2` return `-1;` otherwise return `0` .

Split with dot and compare each number by iterate according to the short length,

✔ 91 Decode Ways

```
Input: "226"
Output: 3
Explanation: It could be decoded as "BZ" (2 26), "VF" (22 6), or "BBF" (2 2 6).
```

06 is not okay, so for 206, you cannot divide it into 2 and 06. 0 is not okay.

Not bug free

✔ 681 Next Closest Time xt-closest-time) Google 52

```
Input: "23:59"
Output: "22:22"
Explanation: The
```

Not bug free, cannot skip

~~609~~ ~~Find Duplicate File in System~~ (/problems/find-duplicate

214 Shortest Palindrome (/problems/shortest-palindrome)

Given a string **s**, you are allowed to convert it to a palindrome by adding characters in front of it.

```
Input: "aacecaaa"
Output: "aaacecaaa"
```

Example 2:

```
Input: "abcd"
Output: "dcbabcd"
```

```
n = len(s)
j=0
for i in range(n-1,-1,-1):

    if s[i] == s[j]:
        j += 1

if j == n:
    return s

rev = s[j:n]
print(j,rev)
rev = rev[::-1]

return rev + self.shortestPalindrome(s[0:j]) + s[j:];
```

```
aaaaaaaab - caaaaaaaaa
aaaaaaaa - b
b - aaaaaaaa - b
aaaaaaaaac - baaaaaaaab - caaaaaaaaa
```

87 Scramble String (/problems/scramble-string)

71 Simplify Path (/problems/simplify-path)

"/a//b////c/d//./..//.." , => "/a/b/c"

Can skip 71, relative easy, if "/" temp = "" restart, if "." Skip, if "..", Pop stack,

126 Word Ladder II (/problems/word-ladder-ii)

68 Text Justification (/proble

That trick again, -3//2 = -2 while int(-3/2) = -1

227 Basic Calculator II (/prob

696 Count Binary Substrings

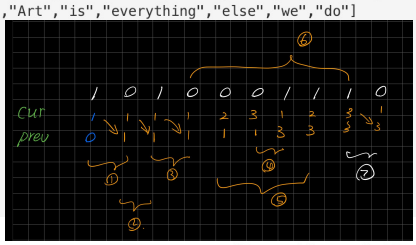
770 Basic Calculator IV (/problems/basic-calculator-iv)

564 Find the Closest Palindrome (/problems/find-the-closest-palind

What about 10,

~~606 Construct String from Binary Tree (/problems/construct-string-from-binary-tree)~~ Skip

```
Input:
words = ["Science","is","what","we","understand","well","enough","to","explain",
         "to","a","computer.","Art","is","everything","else","we","do"]
maxWidth = 20
Output:
[
  "Science  is  what we",
  "understand      well",
  "enough to explain to",
  "a  computer.  Art is",
  "everything  else  we",
  "do                  "
]
```



Given an integer n, find the closest integer (not including itself), which is a palindrome.

The 'closest' is defined as absolute difference minimized between two integers.

Example 1:

```
Input: "123"
Output: "121"
```

686 Repeated String Match (/problems/repeated-string-match)

Google 52 ✔ 159 Longest Substring with At Most Two Distinct Characters (/problems/longest-substring-with-at-most-two-distinct-characters)

✔ 632 Smallest Range (/problems/smallest-range)

✔ 340 Longest Substring with At Most K Distinct Characters (/problems/longest-substring-with-at-most-k-distinct-characters)

161 One Edit Distance (/problems/one-edit-distance)

788 Rotated Digits (/problems/rotated-digits)

537 Complex Number Multiplication (/problems/complex-number-multiplication)

819 Most Common Word (/problems/most-common-word)

767 Reorganize String (/problems/reorganize-string)

761 Special Binary String (/problems/special-binary-string)

158 Read N Characters Given Read4 II - Call multiple times (/problems/read-n-characters-given-read4-ii-call-multiple-times)

459 Repeated Substring Pattern (/problems/repeated-substring-pattern)

890 Find and Replace Pattern (/problems/find-and-replace-pattern)

553 Optimal Division (/problems/optimal-division)

539 Minimum Time Difference (/problems/minimum-time-difference)

157 Read N Characters Given Read4 (/problems/read-n-characters-given-read4)

443 String Compression (/problems/string-compression)

722 Remove Comments (/problems/remove-comments)

680 Valid Palindrome II (/problems/valid-palindrome-ii)

385 Mini Parser (/problems/mini-parser)

520 Detect Capital (/problems/detect-capital)

730 Count Different Palindromic Subsequences (/problems/count-different-palindromic-subsequences)

772 Basic Calculator III (/problems/basic-calculator-iii) 🔓

249 Group Shifted Strings (/problems/group-shifted-strings) 🔓

937 Reorder Log Files (/problems/reorder-log-files)

583 Delete Operation for Two Strings (/problems/delete-operation-for-two-strings)

791 Custom Sort String (/problems/custom-sort-string)

186 Reverse Words in a String II (/problems/reverse-words-in-a-string-ii) 🔓

736 Parse Lisp Expression (/problems/parse-lisp-expression)

556 Next Greater Element III (/problems/next-greater-element-iii)

856 Score of Parentheses (/problems/score-of-parentheses)

678 Valid Parenthesis String (/problems/valid-parenthesis-string)

635 Design Log Storage System (/problems/design-log-storage-system) 🔓

434 Number of Segments in a String (/problems/number-of-segments-in-a-string)

551 Student Attendance Record I (/problems/student-attendance-record-i)

468 Validate IP Address (/problems/validate-ip-address)

893 Groups of Special-Equivalent Strings (/problems/groups-of-special-equivalent-strings)

271 Encode and Decode Strings (/problems/encode-and-decode-strings) 🔓

293 Flip Game (/problems/flip-game) 🔓

544 Output Contest Matches (/problems/output-contest-matches) 🔓

899 Orderly Queue (/problems/orderly-queue)

917 Reverse Only Letters (/problems/reverse-only-letters)

522 Longest Uncommon Subsequence II (/problems/longest-uncommon-subsequence-ii)

859 Buddy Strings (/problems/buddy-strings)

824 Goat Latin (/problems/goat-latin)

833 Find And Replace in String (/problems/find-and-replace-in-string)

616 Add Bold Tag in String (/problems/add-bold-tag-in-string) 🔓

842 Split Array into Fibonacci Sequence (/problems/split-array-into-fibonacci-sequence)

809 Expressive Words (/problems/expressive-words)

916 Word Subsets (/problems/word-subsets)

848 Shifting Letters (/problems/shifting-letters)

831 Masking Personal Information (/problems/masking-personal-information)

800 Similar RGB Color (/problems/similar-rgb-color) 🔓

936 Stamping The Sequence (/problems/stamping-the-sequence)

536 Construct Binary Tree from String (/problems/construct-binary-tree-from-string) 🔓

527 Word Abbreviation (/problems/word-abbreviation) 🔓

816 Ambiguous Coordinates (/problems/ambiguous-coordinates)

591 Tag Validator (/problems/tag-validator)

408 Valid Word Abbreviation (/problems/valid-word-abbreviation) 🔓

925 Long Pressed Name (/problems/long-pressed-name)

758 Bold Words in String (/problems/bold-words-in-string) 🔓

555 Split Concatenated Strings (/problems/split-concatenated-strings) 🔓