5. Longest Palindromic Substring

```
# Find better approach
class Solution:
    def longestPalindrome(self, s: str) -> str:
        total = 0
        c= 0
        ans = ""
        for i in range(len(s)):
            for j in range(i+1,len(s)+1):
                if total < j-i and s[i:j] == s[i:j][::-1]:
                    ans = s[i:j]
                    total = j-i
        return ans
```

8. String to Integer (atoi)

```
# find optimized version
class Solution:
    def myAtoi(self, s: str) -> int:
        res = 1
        i = 0
        if not s: return 0
        while i < len(s) and s[i] == " ":
            i+=1
        if i < len(s):
            if s[i] == "+":
                i+=1
            elif s[i] == "-":
                res = -1
                i+=1
        while i< len(s) and s[i] == "0":
            i+=1

        temp = ""
        while i < len(s):
            if not ("0" <= s[i] <= "9"):
                break
            temp += s[i]
            i+=1
        if temp:
            res *= int(temp)
        else:
            return 0
        if res in range(-2**31,2**31-1+1):
            return res
        elif res < -2**31:
            return -2**31
        else:
            return 2**31-1
        return 0
```

13. Roman to Integer

```
class Solution:
    def romanToInt(self, s: str) -> int:
        hashmap = {
            "I"     :        1,
 "V"             :5,
"X"             :10,
"L"           :50,
"C"             :100,
"D"             :500,
"M"             :1000,
        }
        total = 0
        for i in range(len(s)):
            cur = hashmap.get(s[i])
            if i != len(s)-1 and  cur < hashmap.get(s[i+1]) :
                total -= cur
            else:
                total += cur
        return total
```

14. Longest Common Prefix

```
class Solution:
    def longestCommonPrefix(self, strs: List[str]) -> str:
        n = len(strs)
        m = float("inf")
        for arr in strs:
            m = min(m,len(arr))
        ans = ""
        for i in range(m):
            ele = strs[0][i]
            for j in range(n):
                if ele != strs[j][i]:
                    return ans
            else:
                ans += ele
        return ans
```

151. Reverse Words in a String

```
class Solution:
    def reverseWords(self, s: str) -> str:
        result = []
        temp = ""
        n =len(s)-1
        for i in range(len(s)):
            if s[i]!=" ":
                temp += s[i]
            elif s[i] == " " and temp !="":
                result.append(temp)
                temp=""
            if i == n and temp != "":
                result.append(temp)
        return " ".join(result[::-1])
```

205. Isomorphic Strings

```
class Solution:
    def isIsomorphic(self, s: str, t: str) -> bool:
        char_index_s = {}
        char_index_t = {}

        for i in range(len(s)):
            if s[i] not in char_index_s:
                char_index_s[s[i]] = i

            if t[i] not in char_index_t:
                char_index_t[t[i]] = i

            if char_index_s[s[i]] != char_index_t[t[i]]:
                return False

        return True
```

242. Valid Anagram

```
from collections import Counter

class Solution:
    def isAnagram(self, s: str, t: str) -> bool:
        cnt_s = Counter(s)
        cnt_t = Counter(t)
        if len(s) != len(t):
            return False
        for key,val in cnt_s.items():
            if cnt_s.get(key) != cnt_t.get(key):
                return False
        return True
```

451. Sort Characters By Frequency

```
# Use heap for better time complexity
from collections import Counter
class Solution:
    def frequencySort(self, s: str) -> str:
        cnt_s = Counter(s)
        res = ""
        for key,val in sorted(cnt_s.items(),key = lambda x:x[1],reverse = True):
            res+= key*val
        return res
```

796. Rotate String

```
class Solution:
    def rotateString(self, s: str, goal: str) -> bool:
        if len(s) != len(goal):return False
        for i in range(len(goal)):
            if goal == s[i:]+s[:i]:
                return True
        return False
```

1021. Remove Outermost Parentheses

```
class Solution:
    def removeOuterParentheses(self, s: str) -> str:
        ans = ""
        c = 0
        ind = 0
        for i in range(len(s)):
            if s[i] == "(":
                c+=1
            else:
                c-=1
                if c==0:
                    ans += s[ind+1:i]
                    ind = i+1
        return ans
```

1539. Kth Missing Positive Number

1614. Maximum Nesting Depth of the Parentheses

```
class Solution:
    def maxDepth(self, s: str) -> int:
        total = 0
        c = 0
        for i in s:
            if i == "(":
                c+=1
            elif i == ")":
                c-=1
            total = max(total,c)
        return total
```

1781. Sum of Beauty of All Substrings

```
class Solution:
    def beautySum(self, s: str) -> int:
        beutysum = 0
        n = len(s)

        for i in range(n):
            freq = defaultdict(int)
            for j in range(i,n):
                freq[s[j]]+=1
                max_freq = max(freq.values())
                min_freq = min(freq.values())
                beutysum+=(max_freq-min_freq)
        return beutysum
```

1903. Largest Odd Number in String

```
import sys
sys.set_int_max_str_digits(55555555)
class Solution:
    def largestOddNumber(self, nums: str) -> str:
        ans = ""
        lenn = 0
        for i in range(len(nums)-1,-1,-1):
            if int(nums[i]) % 2 == 1:
                return nums[:i+1]
        return ans
```