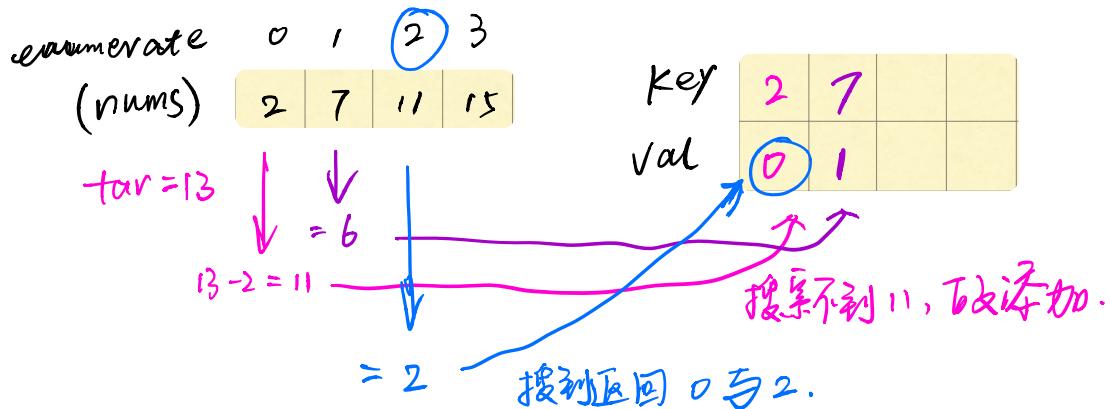


1. 两数之和 two-sum #1

Solution 1: 字典

遍历数组，对 enumerate(nums) 每一个数用 target 减去它，得到结果在字典中查询，如无则将该数下标与值添加字典，直至搜到则返回两个下标。



利用字典的 hashmap 达到时间复杂度 $O(n^2)$

Solution 2: 首尾递进查找

保持数组进行“原地排序”，设置 head, tail 两个指针，从首尾开始，两数相加，如和 > target 则尾指针前调（选一个更小的数来加），反之首指针后调。

原地排序：sorted(range(len(nums)), key=lambda k: nums[k])



因为最终返回的是字串，同时又希望 `nums` 由小到大排序执行
我们的算法，借助字串排序很好的解决了该问题

2. 最长回文子串 longest-Palindrome #5

'a' → 'a', 'aaa' → 'aaa', '' → '', 'abc' → 'a'

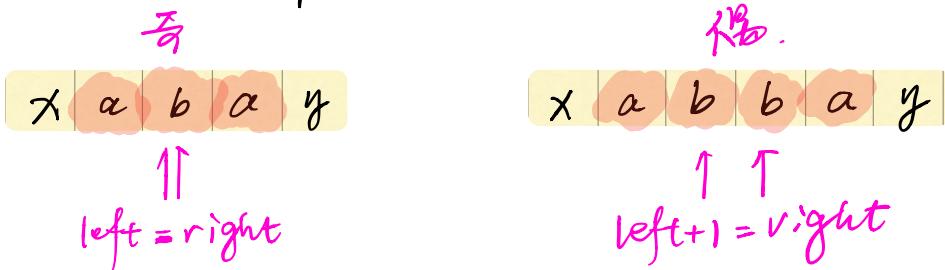
'a' ... '测适时用例.'

Solution 1：暴力循环

会超时，不作解释。

Solution 2：中心扩散。

扫描每一个字串，从右向左侧扩散，进而找出最长子串。
注意分奇偶和边界处理。



时间复杂度 $O(N^2)$ 空间复杂度 $O(1)$

Solution 3：动态规划

解决“最长子结构”问题可以考虑使用“动态规划”方法。

将 $\text{len}(s) \leq 1$ 和 > 1 分开考虑， ≤ 1 时直接 return

首先初始化 dp ，二倍数组值设置 False .

状态转移方程：

两个字母相同 (且只剩3位以内 或 向内收缩一位也相等)

if $s[j] == s[i]$ and ($i-j \leq 2$ or $\text{dp}[j+1][i-1]$):
 $\text{dp}[j][i] = \text{true}$.

时间复杂度 $O(N^2)$ 空间复杂度 $O(N^2)$ = 二维DP问题

3. 两数相加 #2

Solution 1: Listnode.

while l_1, l_2 不为空节点：

取出 l_1, l_2 (此处作判断, 有值取值, 无则取0)

相加, 处理 carry 位. 将个位存至结果表中.

循环结束判断 carry 是否还有1. 有则加一个结点.

最后 return (结果链表. next)

| | | | |
|-------|---|---|---|
| l_1 | 2 | 4 | 3 |
| l_2 | 5 | 6 | 7 |
| rc | 7 | 0 | 8 |

(c) $(7 \quad 0 \quad 8 \quad 1) \rightarrow \text{return}$

注意对于链表的操作都是以 node 为单位的.

4. 整数反转 #7

Solution 1: 转字符串

字符串不能出现的情况最后对 result 判断溢出即可.

$$a[i] \begin{cases} = 0 \rightarrow \text{return } 0 \\ = \text{负号} \rightarrow a[i] \begin{cases} = 0 \rightarrow \text{负号 + 有效位反转} \\ != 0 \rightarrow \text{负号 + 效位反转} \end{cases} \\ \text{else} \rightarrow a[i] \begin{cases} = 0 \rightarrow \text{有效位反转} \\ != 0 \rightarrow \text{效位反转} \end{cases} \end{cases}$$

字符串的反转 $\text{reverse-}a = a[::-1]$

去中间 -lstrip('0') 方法

Solution 2: 列表

逻辑与 Solution 1 同理 .

`list.reverse()` 只是一个方法，没有返回值，改变原列表 .

5. 回文数 #9

Solution 1: 转 str.

return True if $st[i:i-1] == st$ else False

Solution 2: 故事半倒置.

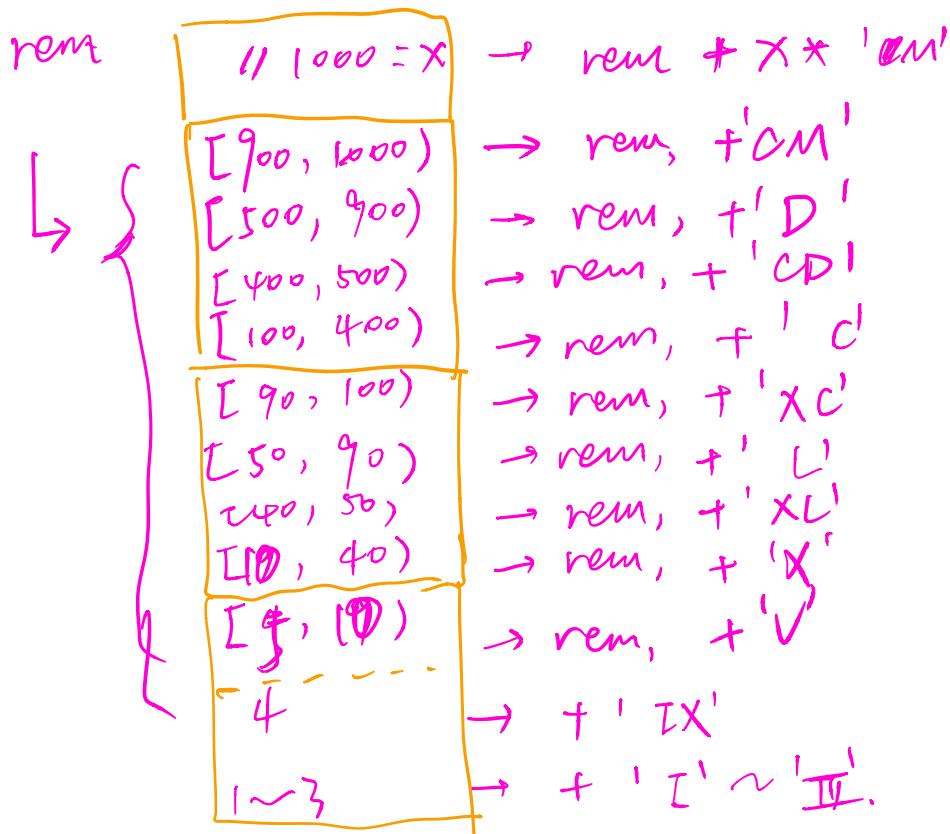
分两类，有以下特征的不可能为回文数，直接 return False： $x < 0$, x 最后一位为 0, x 只有一位。

剩余的，用除和取余方法将 x 取出并放入另一个 "int" 中，当取出的数 $>$ 剩余的即完成半倒置循环。

| x | temp | |
|---------|------|------------------|
| 1 2 1 | | |
| 1 2 | 1 | $12 > 1$ 漏矣 → |
| 1 | 1 2 | $1 < 12$ 跳出 True |
| 1 2 2 1 | 1 2 | True |
| 1 2 | 1 2 | False. |

6. 整数罗马数字 #12.

Solution): 简单.



上述逻辑不够严谨，还需加入如 $10 \sim 30$, $100 \sim 300$ 的情况考虑

7. 罗马数转换 #13

Solution 1: 字典

循环方法：
for i, n in enumerate(s):
 dict.get(s[max(i-1, 0):i+1], dict[n])

边界处理
i=0 时 $s=0 \sim 0$.
i=1 时 $s=0 \sim 1$.
i=2 时 $s=1 \sim 2$.

dict.get(key, default): 有 key 则返回 key 对应 value, 无
则返回 default 对应 value.

8. 最长公共前缀 #14.

Solution 1: ASCII 比较.

先判断是否为字符串，是则返回”，如不是，将 max, min
字符串取出逐位对比，根据 py 排序特性快速解题.

Solution 2: 水平扫描法.

从第1个子串开始，利用find()在后续子串中查找
第一个子串出现的位置，如不是0，则将第一个子串长度
-1回填查找，直到为0，再往下一个字符串查找。

$s: [flower, \leftarrow res = S[0]]$ $\begin{matrix} res \\ \| \end{matrix}$
 $flow, \rightarrow$ 在 $S[0]$ 中找 flower $\rightarrow flow \rightarrow flow \checkmark$
 $flight] \rightarrow$ 在 $S[1]$ 中找 flow $\rightarrow flo \rightarrow fl \checkmark \rightarrow return$