算法分析和复杂性理论

第 3 次作业

张瀚文 2201212865

1 最接近的三数之和（016）

题目描述：给你一个长度为 n 的整数数组 nums 和 一个目标值 target。请你从 nums 中选出三个整数，使它们的和与 target 最接近。返回这三个数的和。假定每组输入只存在恰好一个解。

解题思路：我们首先考虑枚举第一个元素 a，对于剩下的两个元素 b 和 c，我们希望它们的和最接近 target−a 。对于 b 和 c，如果它们在原数组中枚举的范围（既包括下标的范围，也包括元素值的范围）没有任何规律可言，那么我们还是只能使用两重循环来枚举所有的可能情况。因此，我们可以考虑对整个数组进行升序排序，这样一来：假设数组的长度为 n，我们**先枚举 a，它在数组中的位置为 i ；为了防止重复枚举，我们在位置 [i+1,n) 的范围内枚举 b 和 c**。当我们知道了 b 和 c 可以枚举的下标范围，并且知道这一范围对应的数组元素是有序（升序）的，那么我们是否可以对枚举的过程进行优化呢？借助双指针，我们就可以对枚举的过程进行优化。我们用 pb 和 pc 分别表示指向 b 和 c 的指针，初始时，pb 指向位置 i+1 ，即左边界；pc 指向位置 n−1 ，即右边界。**在每一步枚举的过程中，我们用 a+b+c 来更新答案**，并且：如果 a+b+c ≥ target ，那么就将 pc 向左移动一个位置；如果 a+b+c < target ，那么就将 pb 向右移动一个位置。

运行结果：

实验代码：

```
class Solution:
    def threeSumClosest(self, nums: List[int], target: int) -> int:
        nums.sort()
        n = len(nums)
        best = 10**7

        # 根据差值的绝对值来更新答案
```

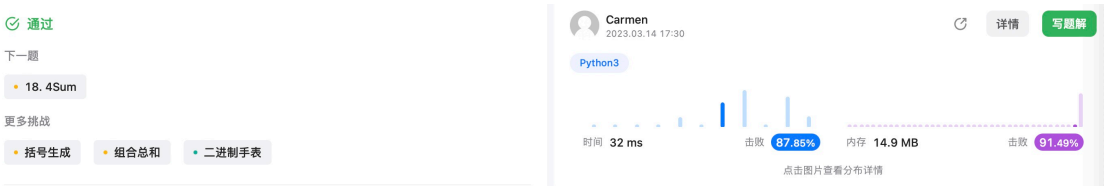
```
    def update(cur):
        nonlocal best
        if abs(cur - target) < abs(best - target):
            best = cur

    # 枚举 a
    for i in range(n):
        # 保证和上一次枚举的元素不相等
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        # 使用双指针枚举 b 和 c
        j, k = i + 1, n - 1
        while j < k:
            s = nums[i] + nums[j] + nums[k]
            # 如果和为 target 直接返回答案
            if s == target:
                return target
            update(s)
            if s > target:
                # 如果和大于 target，移动 c 对应的指针
                k0 = k - 1
                # 移动到下一个不相等的元素
                while j < k0 and nums[k0] == nums[k]:
                    k0 -= 1
                k = k0
            else:
                # 如果和小于 target，移动 b 对应的指针
                j0 = j + 1
                # 移动到下一个不相等的元素
                while j0 < k and nums[j0] == nums[j]:
                    j0 += 1
                j = j0

    return best
```

2 电话号码的字母组合（017）

题目描述：给定一个仅包含数字 2-9 的字符串，返回所有它能表示的字母组合。答案可以按任意顺序 返回。给出数字到字母的映射如下（与电话按键相同）。注意 1 不对应任何字母。

解题思路：使用队列，先将输入的 digits 中第一个数字对应的每一个字母入队，然后将出队的元素与第二个数字对应的每一个字母组合后入队...直到遍历到 digits 的结尾。最后队列中的元素就是所求结果。

运行结果：



实验代码：

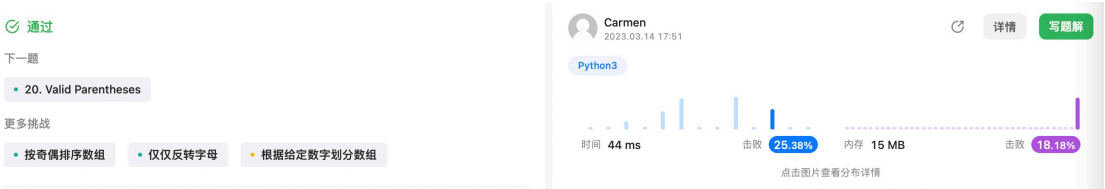
```
class Solution:
    def letterCombinations(self, digits: str) -> List[str]:
        if not digits: return []
        phone = ['abc','def','ghi','jkl','mno','pqrs','tuv','wxyz']
        queue = ['']  # 初始化队列
        for digit in digits:
            for _ in range(len(queue)):
                tmp = queue.pop(0)
                for letter in phone[ord(digit)-50]:# 这里我们不使用 int() 转换字符串，使用 ASCII 码
                    queue.append(tmp + letter)
        return queue
```

3 删除链表的倒数第 N 个结点（019）

题目描述：给你一个链表，删除链表的倒数第 n 个结点，并且返回链表的头结点。

解题思路：我们也可以在遍历链表的同时将所有节点依次入栈。根据栈「先进后出」的原则，我们弹出栈的第 n 个节点就是需要删除的节点，并且目前栈顶的节点就是待删除节点的前驱节点。这样一来，删除操作就变得十分方便了。

运行结果：



实验代码：

```
class Solution:
    def removeNthFromEnd(self, head: ListNode, n: int) -> ListNode:
        dummy = ListNode(0, head)
```

```
stack = list()
cur = dummy
while cur:
    stack.append(cur)
    cur = cur.next

for i in range(n):
    stack.pop()

prev = stack[-1]
prev.next = prev.next.next
return dummy.next
```