

Data Structure and Algorithms Leetcode Coding Review

Jiabin Zhang

2022 年 5 月 26 日

目录

1	Python basic knowledge	2
2	Double pointer and sliding window	2
3	Binary search	30
4	Recursion backtracking	30
5	Graph DFS	30
6	Graph BFS	30
7	Binary Tree	30
8	Stack	30
9	Heap (Priority Queue)	30
10	DP	40
11	Linked list	44
12	Sorting	44
13	Prefix sum	44
14	Array/String	50
15	Math	50
16	Special topics	53
16.0.1	Subarray	53
16.0.2	Substring	53
16.0.3	Subsequence	53

17 Probability and Statistics	53
17.1 Probability	53
17.1.0.1 Sum of two uniform distribution?	53

1 Python basic knowledge

2 Double pointer and sliding window

```

1  # Double pointer and sliding window
2
3  ## 使用条件
4  1. 滑动窗口 - 90% 的概率
5  2. 时间复杂度要求  $O(n)$  - 80% 的概率
6  3. 要求原地操作, 只可以交换使用, 不能使用额外空间, 所以空间复杂度  $O(1)$  - 80%
7  4. 有子数组 subarray, 子字符串 substring 的关键词 - 50%
8  5. 有回文问题 palindrome 关键词 - 50%
9
10 ## time complexity
11 >> 时间复杂度与最内层循环主体的 loop 执行次数有关, 与有多少重循环无关,  $O(n)$ 
12 ## space complexity
13 >> 只需要分配 2 个指针的额外内存, 所以 space 是  $O(1)$ 
14
15 ## 几种类型的双指针及相关题目
16 1. 同向: 特点是指针不回头, 全 0 子串数量 - slow, fast, 基本等价于 sliding window
17 2. 相向: two sum, three sum, left, right
18 3. 背向: 最长回文子串
19
20 ## template
21 ### 相向双指针 - partition in quicksort
22 ```python
23 class Solution:
24     def partition(self, A, start, end):
25         if start >= end:
26             return
27         left, right = start, end
28         # key point 1: pivot is the value, not the index
29         pivot = A[(start + end) // 2]
30         # key point 2: every time you compare left & right, it should be left <= right not left < right
31         while left <= right:
32             while left <= right and A[left] < pivot:
33                 left += 1
34             while left <= right and A[right] > pivot:
35                 right -= 1
36             if left <= right:
37                 A[left], A[right] = A[right], A[left]
38                 left += 1
39                 right -= 1
40         ...
41 ### 背向双指针
42 ```python

```

```

43 class Solution:
44     def partition(self, A, start, end):
45         left = position
46         right = position + 1
47         while left >=0 and right < len(s):
48             if left and right 可以停下来了:
49                 break
50             left -= 1
51             right += 1
52     ```
53     ### 同向双指针 - 快慢指针
54
55     ```python
56     class Solution:
57         def partition(self, A, start, end):
58             j = 0
59             for i in range(n):
60                 # 不满足则循环到满足搭配为止
61                 while j < n and i and j 之间不满足条件:
62                     j += 1
63                 if i 到 j 之间满足条件:
64                     处理 i 到 j 这段区间
65     ```
66
67     ## 高频题目整理
68
69     ### 相向双指针
70     - #1. Two Sum
71         > 暴力 double for loop -> hashtable -> 排序双指针 (如何排序 + index 操作需要注意); 这里要求返回下面, 如果返回值比较容易
72     - #167. Two Sum II - Input Array Is Sorted https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/
73         > 排序之后返回 index, 比 two sum 简单, 我觉得只有排序之后, 才能相向双指针, 否则没有意义
74     - #15. 3Sum https://leetcode.com/problems/3sum/
75         > 最外层 for loop 作为一个指针, 内嵌 while loop, 考虑 left, right 指针, 这个难点在于去重, 去重很多种办法, 包括 set, 还有左
76         ↳ 右移动, 因为已经排序, 所以用相邻位置的比较来去重
77     - #16. 3Sum Closest https://leetcode.com/problems/3sum-closest/
78         > 比 15 简单, 几个 edge cases 都考虑到了
79     - #259. 3Sum Smaller https://leetcode.com/problems/3sum-smaller/
80         > res += right - left # 这一步是关键, 之前没有想清楚, 为什么是 right-left, 其实就是中间的都可以
81     - #18. 4Sum https://leetcode.com/problems/4sum/
82         > 完全和 3sum 一样! 就是复杂的一点, time  $O(n^3)$ , space  $O(n)$ 
83     - #454. 4Sum II https://leetcode.com/problems/4sum-ii/
84         > hashmap, time  $O(n^2)$ , space  $O(n^2)$ 
85     - #75. Sort Colors https://leetcode.com/problems/sort-colors/
86     - #1229. Meeting Scheduler https://leetcode.com/problems/meeting-scheduler/
87     - #125. Valid Palindrome https://leetcode.com/problems/valid-palindrome/
88
89     ### 背向双指针
90     - #5. Longest Palindromic Substring https://leetcode.com/problems/longest-palindromic-substring/
91     - #408. Valid Word Abbreviation https://leetcode.com/problems/valid-word-abbreviation/
92     - #409. Longest Palindrome https://leetcode.com/problems/longest-palindrome/
93     - #680. Valid Palindrome II https://leetcode.com/problems/valid-palindrome-ii/

```

```

94
95 ### Sliding windows
96 - #3. Longest Substring Without Repeating Characters
97   ↳ https://leetcode.com/problems/longest-substring-without-repeating-characters
98 - #76. Minimum Window Substring https://leetcode.com/problems/minimum-window-substring
99 - #1004. Max Consecutive Ones III https://leetcode.com/problems/max-consecutive-ones-iii
100 - #209. Minimum Size Subarray Sum https://leetcode.com/problems/minimum-size-subarray-sum
101 - #1438. Longest Continuous Subarray With Absolute Diff Less Than or Equal to Limit
102   ↳ https://leetcode.com/problems/longest-continuous-subarray-with-absolute-diff-less-than-or-equal-to-limit
103
104 ### 新添加一些题
105 - #38. Count and Say https://leetcode.com/problems/count-and-say (用到拼接的思想, 如何双指针)
106 - #30. Substring with Concatenation of All Words https://leetcode.com/problems/substring-with-concatenation-of-all-words (sliding window 好题)
107 - #228. Summary Ranges https://leetcode.com/problems/summary-ranges/
108
109 ## 题目答案和分析
110
111 ### 1. Two Sum
112 ``python
113 # double pointer
114 class Solution:
115     def twoSum(self, nums, target):
116         nums = [(number, index) for index, number in enumerate(nums)]
117         nums.sort()
118         left, right = 0, len(nums) - 1
119         while left < right:
120             if nums[left][0] + nums[right][0] < target: # 小于 target, left 右移
121                 left += 1
122             elif nums[left][0] + nums[right][0] > target: # 大于 target, right 左移
123                 right -= 1
124             else: # 这个必须要有, 就是说 == target, 直接返回
125                 return sorted([nums[left][1], nums[right][1]]) # 是否拍需要看要求, 这个需要从小到大, 就排一下
126         return
127
128 # hashmap solution - better one!
129 class Solution:
130     def twoSum(self, nums, target):
131         if not nums: return
132         n = len(nums)
133         hashmap = {}
134         for i in range(n):
135             residual = target - nums[i]
136             if residual in hashmap:
137                 res = [hashmap[residual], i]
138                 hashmap[nums[i]] = i
139         return sorted(res)
140
141 ### 1099. Two Sum Less Than K https://leetcode.com/problems/two-sum-less-than-k/

```

```

142 Given an array nums of integers and integer k, return the maximum sum such that there exists i < j with nums[i]
    ↪ + nums[j] = sum and sum < k. If no i, j exist satisfying this equation, return -1.
143 ```
144 Example 1:
145
146 Input: nums = [34,23,1,24,75,33,54,8], k = 60
147 Output: 58
148 Explanation: We can use 34 and 24 to sum 58 which is less than 60.
149 Example 2:
150
151 Input: nums = [10,20,30], k = 15
152 Output: -1
153 Explanation: In this case it is not possible to get a pair sum less than 15.
154 ```
155 > two sum 的一个变种!
156 ```python
157 class Solution:
158     def twoSumLessThanK(self, nums: List[int], k: int) -> int:
159         # brute force 不对, 草!      time o(n^2) and space o(1)
160         res = -1 # 这个最开始就应该等于-1
161         for i in range(len(nums)):
162             for j in range(i + 1, len(nums)): # 不一样的数
163                 twosum = nums[i] + nums[j]
164                 if twosum < k:
165                     res = max(res, twosum)
166         return res
167         # double pointer - 这个思路就很清晰, time o(nlogn), space o(logn) to o(n)
168         nums.sort()
169         left, right = 0, len(nums) - 1
170         res = - 1
171         while left < right:
172             twosum = nums[left] + nums[right]
173             if twosum < k:
174                 left += 1
175                 res = max(res, twosum)
176             else:
177                 right -= 1
178         return res
179 ```
180
181 ### 167. Two Sum II - Input Array Is Sorted https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/
182 Given a 1-indexed array of integers numbers that is already sorted in non-decreasing order, find two numbers
    ↪ such that they add up to a specific target number. Let these two numbers be numbers[index1] and
    ↪ numbers[index2] where 1 <= index1 < index2 <= numbers.length. Return the indices of the two numbers, index1
    ↪ and index2, added by one as an integer array [index1, index2] of length 2. The tests are generated such that
    ↪ there is exactly one solution. You may not use the same element twice. Your solution must use only constant
    ↪ extra space.
183 ```
184 Example 1:
185
186 Input: numbers = [2,7,11,15], target = 9
187 Output: [1,2]

```

```

188 Explanation: The sum of 2 and 7 is 9. Therefore, index1 = 1, index2 = 2. We return [1, 2].
189 Example 2:
190
191 Input: numbers = [2,3,4], target = 6
192 Output: [1,3]
193 Explanation: The sum of 2 and 4 is 6. Therefore index1 = 1, index2 = 3. We return [1, 3].
194 Example 3:
195
196 Input: numbers = [-1,0], target = -1
197 Output: [1,2]
198 Explanation: The sum of -1 and 0 is -1. Therefore index1 = 1, index2 = 2. We return [1, 2].
199 ```
200 > two sum 变种
201 ```python
202 class Solution:
203     def twoSum(self, numbers: List[int], target: int) -> List[int]:
204         # 就是 two sum with double pointer but it is sorted
205         # 要求 o(1) space
206         left, right = 0, len(numbers) - 1
207         while left < right:
208             if numbers[left] + numbers[right] < target:
209                 left += 1
210             elif numbers[left] + numbers[right] > target:
211                 right -= 1
212             else:
213                 return [left + 1, right + 1]
214         return []
215 ```
216
217 ### 15. 3Sum https://leetcode.com/problems/3sum/
218 ```python
219 class Solution:
220     def threeSum(self, nums):
221         if len(nums) < 3:
222             return
223
224         n = len(nums)
225         nums.sort()
226         ans = []
227         for i in range(n):
228             left = i + 1
229             right = n - 1
230             if nums[i] > 0: # 最小值大于 0
231                 break
232             if i >= 1 and nums[i] == nums[i-1]: # 差一点, i >= 1 才行, 这种去重的题还是挺不好弄的!
233                 continue # 如果相邻的重复, 那么就 move 一下, 去掉重复的, 直接跳过这个 loop
234             # left, right = 0, n - 1 # 不是这样的, 要和 i 相关
235             while left < right:
236                 target = nums[i] + nums[left] + nums[right]
237                 if target < 0:
238                     left += 1
239                 elif target > 0:

```

```

240         right -= 1
241     else:
242         ans.append([nums[i], nums[left], nums[right]])
243         while left != right and nums[left] == nums[left + 1]: # 去重 left
244             left += 1
245         while left != right and nums[right] == nums[right - 1]: # 去重 right
246             right -= 1
247         left += 1
248         right -= 1
249     return ans
250
251
252 ### 18. 4Sum https://leetcode.com/problems/4sum/
253 ```python
254 # hashmap solution
255 class Solution:
256     def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
257         hashmap = {}
258         for i in nums1:
259             for j in nums2:
260                 if i + j in hashmap:
261                     hashmap[i + j] += 1
262                 else:
263                     hashmap[i + j] = 1
264         res = 0
265         for m in nums3:
266             for n in nums4:
267                 if 0 - m - n in hashmap:
268                     res += hashmap[0 - m - n]
269         return res
270
271 # double pointer solution
272 class Solution:
273     def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
274         # 完全和 3sum 一样! 就是复杂的一点, time o(n^3), space o(n)
275         nums.sort()
276         n = len(nums)
277         res = []
278         for i in range(n):
279             if i > 0 and nums[i] == nums[i - 1]: continue
280             for j in range(i+1, n):
281                 if j > i + 1 and nums[j] == nums[j - 1]: continue
282                 left = j + 1
283                 right = n - 1
284                 while left < right:
285                     sum_all = nums[i] + nums[j] + nums[left] + nums[right]
286                     if sum_all < target:
287                         left += 1
288                     elif sum_all > target:
289                         right -= 1
290                     else:
291                         res.append([nums[i], nums[j], nums[left], nums[right]])

```

```

292         while left < right and nums[left] == nums[left + 1]:
293             left += 1
294         while left < right and nums[right] == nums[right - 1]:
295             right -= 1
296
297         left += 1
298         right -= 1
299
300     return res # 位置错了, 在最外层, 容易忽视!
301
302
303
304     ### 187. Repeated DNA Sequences https://leetcode.com/problems/repeated-dna-sequences/
305
306     The DNA sequence is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'.
307
308     For example, "ACGAATTCG" is a DNA sequence.
309     When studying DNA, it is useful to identify repeated sequences within the DNA.
310
311     Given a string s that represents a DNA sequence, return all the 10-letter-long sequences (substrings) that occur
312     ⇨ more than once in a DNA molecule. You may return the answer in any order.
313
314     Example 1:
315     Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"
316     Output: ["AAAAACCCCC", "CCCCCAAAAA"]
317
318     Example 2:
319     Input: s = "AAAAAAAAAAAA"
320     Output: ["AAAAAAAAAA"]
321
322     ```python
323     # 不难, 应该不算典型的 sliding window!
324     class Solution:
325     def findRepeatedDnaSequences(self, s: str) -> List[str]:
326         if not s: return
327         n = len(s)
328         hashm = {}
329         res = []
330         for i in range(n - 10 + 1): # 别忘了 +1
331             curr = s[i:i + 10]
332             hashm[curr] = hashm.get(curr, 0) + 1
333             if hashm[curr] > 1:
334                 res.append(curr)
335
336         return set(res) # 这个要加 set 否则输出一样的
337         # time o(n), space o(n)
338
339
340     ### 209. Minimum Size Subarray Sum https://leetcode.com/problems/minimum-size-subarray-sum/
341
342

```



```

343 Given an array of positive integers nums and a positive integer target, return the minimal length of a
    ↳ contiguous subarray [numsl, numsl+1, ..., numsr-1, numsr] of which the sum is greater than or equal to
    ↳ target. If there is no such subarray, return 0 instead.
344
345 Example 1:
346 ```
347 Input: target = 7, nums = [2,3,1,2,4,3]
348 Output: 2
349 Explanation: The subarray [4,3] has the minimal length under the problem constraint.
350 ```
351 Example 2:
352 ```
353 Input: target = 4, nums = [1,4,4]
354 Output: 1
355 ```
356 Example 3:
357 ```
358 Input: target = 11, nums = [1,1,1,1,1,1,1,1]
359 Output: 0
360 ```
361 ```python
362 # sliding window 模板题, 不难, 涉及 subarray
363 class Solution:
364     def minSubArrayLen(self, target: int, nums: List[int]) -> int:
365         if sum(nums) < target: # 临界状态
366             return 0
367         n = len(nums)
368         slow = 0
369         res = inf
370         sum_ = 0
371         for fast in range(n):
372             sum_ += nums[fast]
373             while sum_ >= target:
374                 res = min(res, fast - slow + 1)
375                 sum_ -= nums[slow]
376                 slow += 1
377         return res
378     ```
379
380 ---
381 217, 219, 220 是连续三个 contains duplicate 比较常见
382
383 ### 217. Contains Duplicate https://leetcode.com/problems/contains-duplicate/
384 Given an integer array nums, return true if any value appears at least twice in the array, and return false if
    ↳ every element is distinct.
385 ```
386 Example 1:
387 Input: nums = [1,2,3,1]
388 Output: true
389 Example 2:
390 Input: nums = [1,2,3,4]
391 Output: false

```

```

392 Example 3:
393 Input: nums = [1,1,1,3,3,4,3,2,4,2]
394 Output: true
395 ```
396
397 ```python
398 class Solution:
399     def containsDuplicate(self, nums: List[int]) -> bool:
400         # way 1 - hashtable
401         record = set()
402         for num in nums:
403             if num not in record: # 注意是 not in 不是 is not in
404                 record.add(num)
405             else:
406                 return True
407         return False
408         # time o(n), space o(n)
409
410         # way 2 - sorting - optimal space - time o(nlogn), space o(1)
411         nums.sort()
412         print(nums)
413         n = len(nums)
414         for i in range(1, n):
415             if nums[i-1] == nums[i]: # 要搞清楚题目含义
416                 return True
417         return False
418 ```
419
420
421 ### 219. Contains Duplicate II https://leetcode.com/problems/contains-duplicate-ii/
422 Given an integer array nums and an integer k, return true if there are two distinct indices i and j in the array
423 ↪ such that nums[i] == nums[j] and abs(i - j) <= k.
424
425 Example 1:
426 Input: nums = [1,2,3,1], k = 3
427 Output: true
428
429 Example 2:
430 Input: nums = [1,0,1,1], k = 1
431 Output: true
432
433 Example 3:
434 Input: nums = [1,2,3,1,2,3], k = 2
435 Output: false
436
437 ```python
438
439 class Solution:
440     def containsNearbyDuplicate(self, nums: List[int], k: int) -> bool:

```

```

443 # brute force - 超时
444 for i in range(len(nums)):
445     for j in range(i + 1, len(nums)):
446         if nums[i] == nums[j] and abs(i-j) <= k:
447             return True
448 return False
449
450 # hashtable 还是做出来了，之前方向不对！
451 # 思路是，用 hash 找到所有相同的数，然后要注意更新，因为是一次遍历，如果之前的相等的数不 work 要 update 到当前值
452 hashm = {}
453 for i in range(len(nums)):
454     if nums[i] not in hashm:
455         hashm[nums[i]] = i
456     else:
457         dis = abs(i - hashm[nums[i]])
458         if dis <= k:
459             return True
460     else:
461         hashm[nums[i]] = i # 这一步是关键，之前没有更新 nums = [1,0,1,1], k = 1 就过不去
462         continue
463 return False
464 time o(n), space o(n)
465
466 # sliding window, 但不容易，容易顺序错了！
467 s = set()
468 for i, num in enumerate(nums):
469     # 先判断是不是有违法的窗口
470     if i > k:
471         s.remove(nums[i - k - 1])
472     # 如果没有违法的，看是否满足目标
473     if num in s:
474         return True
475     # 上面都没有满足，加入 set 来判断，这三条顺序都不能变！
476     s.add(num)
477 return False
478
479
480
481 ### 220. Contains Duplicate III https://leetcode.com/problems/contains-duplicate-iii/
482
483 Given an integer array nums and two integers k and t, return true if there are two distinct indices i and j in
484 ↪ the array such that abs(nums[i] - nums[j]) <= t and abs(i - j) <= k.
485
486 Example 1:
487 Input: nums = [1,2,3,1], k = 3, t = 0
488 Output: true
489 Example 2:
490 Input: nums = [1,0,1,1], k = 1, t = 2
491 Output: true
492 Example 3:
493 Input: nums = [1,5,9,1,5,9], k = 2, t = 3
494 Output: false

```

```

494     ``
495 > 挺难的, 不太了解, 超出了通常 sliding window 的范围! 在 219 的基础上可能做出来一些, 但排序已经 bucket 不好做
496
497     `` python
498     # brute force 过不了, 意义不大, 太 easy 了
499     # class Solution:
500     #     def containsNearbyAlmostDuplicate(self, nums: List[int], k: int, t: int) -> bool:
501     #         # # brute force - o(n^2), space o(1)
502     #         # n = len(nums)
503     #         # for i in range(n):
504     #             # for j in range(i + 1, n):
505     #                 # if abs(nums[i] - nums[j]) <= t and abs(i - j) <= k:
506     #                     # return True
507     #         # return False
508     class Solution:
509     def containsNearbyAlmostDuplicate(self, nums: List[int], k: int, t: int) -> bool:
510         from sortedcontainers import SortedSet
511         if not nums or t < 0: return False # Handle special cases
512         ss, n = SortedSet(), 0 # Create SortedSet. 'n' is the size of sortedset, max value of
513         ↪ 'n' is 'k' from input
514         for i, num in enumerate(nums):
515             ceiling_idx = ss.bisect_left(num) # index whose value is greater than or equal to 'num'
516             floor_idx = ceiling_idx - 1 # index whose value is smaller than 'num'
517             if ceiling_idx < n and abs(ss[ceiling_idx]-num) <= t: return True # check right neighbour
518             if 0 <= floor_idx and abs(ss[floor_idx]-num) <= t: return True # check left neighbour
519             ss.add(num)
520             n += 1
521             if i - k >= 0: # maintain the size of sortedset by finding & removing the earliest number in
522                 ↪ sortedset
523                 ss.remove(nums[i-k])
524                 n -= 1
525         return False
526
527     ``
528
529 ---
530
531 ❶ 系列题, 关于 longest substring distinct characters 很多类似的题目, 总结一下! 主要是 hashtable, sliding window 的结合,
532 ↪ 复杂的 case 需要 dp. upstart 考了类似的题目!
533
534 ❷ substring, subarray, subsequence 三种常见的问题, 总结一下!
535
536 ❸ Substring & string 类型的
537
538 ### 159. Longest Substring with At Most Two Distinct Characters
539 ↪ https://leetcode.com/problems/longest-substring-with-at-most-two-distinct-characters/
540
541 Given a string s, return the length of the longest substring that contains at most two distinct characters.
542 Example 1:
543 ``
544 Input: s = "eceba"

```

```

542 Output: 3
543 Explanation: The substring is "ece" which its length is 3.
544 ```
545 Example 2:
546 ```
547 Input: s = "ccaabbb"
548 Output: 5
549 Explanation: The substring is "aabbb" which its length is 5.
550 ```
551
552 ```python
553 # 经典 sliding window 模板题,
554 class Solution:
555     def lengthOfLongestSubstringTwoDistinct(self, s: str) -> int:
556         # 这个模板不错!
557         slow = 0
558         n = len(s)
559         # hashset = set() # 尽量用 set 来弄, 或者 hashmap
560         hashmap = {}
561         res = 0
562         for fast in range(n):
563             # 先去操作目标, 放进 hashmap
564             hashmap[s[fast]] = hashmap.get(s[fast], 0) + 1
565             # 先判断是否满足条件, 如果满足, 操作
566             if len(hashmap) <= 2:
567                 res = max(res, fast - slow + 1)
568             # 不满足的话, 想办法更新 slow 指针
569             while len(hashmap) > 2:
570                 head = s[slow]
571                 hashmap[head] -= 1
572                 if hashmap[head] == 0:
573                     del hashmap[head]
574                 slow += 1
575
576         return res
577 ```
578
579 ### 340. Longest Substring with At Most K Distinct Characters
580 ↳ https://leetcode.com/problems/longest-substring-with-at-most-k-distinct-characters/
581 Given a string s and an integer k, return the length of the longest substring of s that contains at most k
582 ↳ distinct characters.
583 ```
584 Example 1:
585
586 Input: s = "eceba", k = 2
587 Output: 3
588 Explanation: The substring is "ece" with length 3.
589 Example 2:
590
591 Input: s = "aa", k = 1
592 Output: 2
593 Explanation: The substring is "aa" with length 2.

```

```

592  ```
593  > 这个题如果难一点就是让返回所有的最长的 substring 这个要自己写一下
594
595  ```python
596  class Solution:
597      def lengthOfLongestSubstringKDistinct(self, s: str, k: int) -> int:
598          # 模板确实厉害! 清晰
599          slow, hashm = 0, {}
600          res = 0
601          for fast in range(len(s)):
602              tail = s[fast]
603              hashm[tail] = hashm.get(tail, 0) + 1
604              if len(hashm) <= k:
605                  res = max(res, fast - slow + 1)
606              while len(hashm) > k: # 这个位置就是想清楚, 不满足 if 条件, 用 if 还是 while
607                  head = s[slow]
608                  hashm[head] -= 1
609                  if hashm[head] == 0:
610                      del hashm[head]
611                  slow += 1
612          return res
613  ```
614
615  ### 3. Longest Substring Without Repeating Characters
616  ↪ https://leetcode.com/problems/longest-substring-without-repeating-characters/
617  Given a string s, find the length of the longest substring without repeating characters.
618  ```
619
620  Example 1:
621
622  Input: s = "abcabcbb"
623  Output: 3
624  Explanation: The answer is "abc", with the length of 3.
625
626  Example 2:
627
628  Input: s = "bbbbb"
629  Output: 1
630  Explanation: The answer is "b", with the length of 1.
631
632  Example 3:
633
634  Input: s = "pwwkew"
635  Output: 3
636  Explanation: The answer is "wke", with the length of 3.
637  Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.
638  ```
639  > 这个和 340 非常一致, 可以放到一起来做, 一个 follow up 是返回所有的最长的 substring! upstart 考了!
640
641  ```python
642  class Solution:
643      def lengthOfLongestSubstring(self, s: str) -> int:
644          # 这个模板还是不错的!
645          slow = 0
646          hashmap = {}

```

```

643     res = 0
644     n = len(s)
645     for fast in range(n):
646         tail = s[fast]
647         hashmap[tail] = hashmap.get(tail, 0) + 1
648         if len(hashmap) == fast - slow + 1:
649             res = max(res, fast - slow + 1)
650         while fast - slow + 1 > len(hashmap):
651             head = s[slow]
652             hashmap[head] -= 1
653             if hashmap[head] == 0:
654                 del hashmap[head]
655             slow += 1
656     return res
657
658
659 > Follow-up: return all 最长的 substring
660 Example
661 input = "fsfetwenwac"
662 output = ['sfetw', 'enwac']
663
664 ```python
665 class Solution:
666     def lengthOfLongestSubstring(self, s: str) -> int:
667         # 这个模板还是不错的!
668         slow = 0
669         hashmap = {}
670         max_len = 0
671         n = len(s)
672         res_list = []
673         for fast in range(n):
674             tail = s[fast]
675             hashmap[tail] = hashmap.get(tail, 0) + 1
676             if len(hashmap) == fast - slow + 1:
677                 # 这段是核心记录所有 list 的办法! 其实不难
678                 print(fast - slow + 1, max_len)
679                 if fast - slow + 1 > max_len:
680                     res_list = [] # 这个就是通过 [] 来不断 update
681                     res_list.append((slow, fast))
682                     print(res_list)
683                     max_len = max(max_len, fast - slow + 1)
684                 elif fast - slow + 1 == max_len:
685                     res_list.append((slow, fast))
686
687             while fast - slow + 1 > len(hashmap):
688                 head = s[slow]
689                 hashmap[head] -= 1
690                 if hashmap[head] == 0:
691                     del hashmap[head]
692                 slow += 1
693         # 用 tuple 记录 slow and fast 位置, 然后最后一起输出
694         output = []

```

```

695     print(res_list)
696     for i, j in res_list:
697         output.append(s[i:j+1])
698
699     return output
700
701
702 > 395 算是 substring, 但不算典型的 sliding window
703
704 ### 395. Longest Substring with At Least K Repeating Characters
705 ↪ https://leetcode.com/problems/longest-substring-with-at-least-k-repeating-characters/
706 Given a string s and an integer k, return the length of the longest substring of s such that the frequency of
707 ↪ each character in this substring is greater than or equal to k.
708
709 Example 1:
710 Input: s = "aaabb", k = 3
711 Output: 3
712 Explanation: The longest substring is "aaa", as 'a' is repeated 3 times.
713 Example 2:
714 Input: s = "ababbc", k = 2
715 Output: 5
716 Explanation: The longest substring is "ababb", as 'a' is repeated 2 times and 'b' is repeated 3 times.
717
718 > 这个用 brute force 可以, 但是 sliding window 很难写, 关键是如何控制 window 里面的最小值, 递归的方法不好想, 不具有通用性! 虽然
719 ↪ 看上去和 340 挺像, 但实际上完全不一样!!!!
720
721 ```python
722 class Solution:
723     def longestSubstring(self, s: str, k: int) -> int:
724         # 这个题用正常的 sliding window 很难做!
725         # brute force 要想出来 double for loop - time o(n^2), space o(1)
726         n = len(s)
727         res = 0
728         for i in range(n):
729             for j in range(i + 1, n + 1): # 注意这个地方要 n+1
730                 hashmap = Counter(s[i:j]) # 这个也要注意
731                 if min(hashmap.values()) >= k: # 这个是可以操作的
732                     res = max(res, j - i)
733         return res
734
735 # # 递归的解法有点秀!
736 # class Solution(object):
737 #     def longestSubstring(self, s, k):
738 #         if len(s) < k:
739 #             return 0
740 #         for c in set(s):
741 #             if s.count(c) < k:
742 #                 return max(self.longestSubstring(t, k) for t in s.split(c))
743 #         return len(s)
744 ```

```



```

744
745 ### 424. Longest Repeating Character Replacement
746 ↪ https://leetcode.com/problems/longest-repeating-character-replacement/
747 You are given a string s and an integer k. You can choose any character of the string and change it to any other
748 ↪ uppercase English character. You can perform this operation at most k times. Return the length of the
749 ↪ longest substring containing the same letter you can get after performing the above operations.
750
751 Example 1:
752 Input: s = "ABAB", k = 2
753 Output: 4
754 Explanation: Replace the two 'A's with two 'B's or vice versa.
755
756 Example 2:
757 Input: s = "AABABBA", k = 1
758 Output: 4
759 Explanation: Replace the one 'A' in the middle with 'B' and form "AABBBBA".
760 The substring "BBBB" has the longest repeating letters, which is 4.
761
762 > 不算 substring, 但是 sliding windows 相关, 有一类题就是可以替换操作!
763 ```python
764 class Solution:
765     def characterReplacement(self, s: str, k: int) -> int:
766         # 应用模板不错的一个题
767         slow, res, max_freq, hashm = 0, 0, 0, {}
768         for fast in range(len(s)):
769             tail = s[fast]
770             hashm[tail] = hashm.get(tail, 0) + 1
771             max_freq = max(max_freq, hashm[tail]) # 这是关键, 统计 frequency, 和 01 问题的区别, 当时给定 1 了
772             if fast - slow + 1 <= max_freq + k: # 这步必须是 <= 之前也有类似的问题
773                 res = max(res, fast - slow + 1)
774             while fast - slow + 1 > max_freq + k:
775                 head = s[slow]
776                 hashm[head] -= 1
777                 if hashm[head] == 0:
778                     del hashm[head]
779                 slow += 1
780         return res
781
782 ### 438. Find All Anagrams in a String https://leetcode.com/problems/find-all-anagrams-in-a-string/
783 Given two strings s and p, return an array of all the start indices of p's anagrams in s. You may return the
784 ↪ answer in any order.
785 An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using
786 ↪ all the original letters exactly once.
787
788 Example 1:
789 Input: s = "cbaebabacd", p = "abc"
790 Output: [0,6]
791 Explanation:
792 The substring with start index = 0 is "cba", which is an anagram of "abc".

```

```

791 The substring with start index = 6 is "bac", which is an anagram of "abc".
792 Example 2:
793
794 Input: s = "abab", p = "ab"
795 Output: [0,1,2]
796 Explanation:
797 The substring with start index = 0 is "ab", which is an anagram of "ab".
798 The substring with start index = 1 is "ba", which is an anagram of "ab".
799 The substring with start index = 2 is "ab", which is an anagram of "ab".
800 ...
801 > sliding window 模板题
802 ```python
803 class Solution:
804     def findAnagrams(self, s: str, p: str) -> List[int]:
805         # 模板确实不错!
806         res = []
807         slow = 0
808         hash_s = {}
809         hash_p = {}
810         for char in p:
811             hash_p[char] = hash_p.get(char, 0) + 1
812         for fast in range(len(s)):
813             hash_s[s[fast]] = hash_s.get(s[fast], 0) + 1
814             if hash_s == hash_p:
815                 res.append(slow)
816             if fast >= len(p) - 1:
817                 head = s[slow]
818                 hash_s[head] -= 1
819                 if hash_s[head] == 0:
820                     del hash_s[head]
821                 slow += 1
822         return res
823 ...
824
825 ### 567. Permutation in String https://leetcode.com/problems/permutation-in-string/
826 Given two strings s1 and s2, return true if s2 contains a permutation of s1, or false otherwise. In other words,
827 ↪ return true if one of s1's permutations is the substring of s2.
828 ...
829 Example 1:
830
831 Input: s1 = "ab", s2 = "eidbaooo"
832 Output: true
833 Explanation: s2 contains one permutation of s1 ("ba").
834 Example 2:
835
836 Input: s1 = "ab", s2 = "eidboao"
837 Output: false
838 ...
839 > 和 438 很像主席细节
840 ```python
841 class Solution:
842     def checkInclusion(self, s1: str, s2: str) -> bool:

```

```

842     # sliding window 的题, 和 438 非常像, 本身也算 permutation
843     hashes1 = {}
844     hashes2 = {}
845     for char in s1:
846         hashes1[char] = hashes1.get(char, 0) + 1
847
848     slow = 0
849     for fast in range(len(s2)):
850         hashes2[s2[fast]] = hashes2.get(s2[fast], 0) + 1
851         if hashes2 == hashes1:
852             return True
853         if fast >= len(s1) - 1:
854             head = s2[slow] # 注意细节, 不是一味的背模板!
855             hashes2[head] -= 1
856             if hashes2[head] == 0:
857                 del hashes2[head]
858             slow += 1
859
860     return False
861
862
863
864     ### 系列题, max consecutive ones 1,2,3
865
866     ### 485. Max Consecutive Ones https://leetcode.com/problems/max-consecutive-ones/
867
868     Given a binary array nums, return the maximum number of consecutive 1's in the array.
869     ...
870     Example 1:
871
872     Input: nums = [1,1,0,1,1,1]
873     Output: 3
874     Explanation: The first two digits or the last three digits are consecutive 1s. The maximum number of consecutive
875     ↪ 1s is 3.
876     Example 2:
877
878     Input: nums = [1,0,1,1,0,1]
879     Output: 2
880     ...
881     ```python
882     class Solution:
883         def findMaxConsecutiveOnes(self, nums: List[int]) -> int:
884             temp = 0
885             res = 0
886             for i in nums:
887                 if i == 1:
888                     temp += 1
889                 else:
890                     temp = 0
891             res = max(res, temp)
892
893     return res

```

```

893     ``
894
895
896     ### 487. Max Consecutive Ones II https://leetcode.com/problems/max-consecutive-ones-ii/
897     Given a binary array nums, return the maximum number of consecutive 1's in the array if you can flip at most one
898     ↪ 0.
899     ``
900
901     Example 1:
902
903     Input: nums = [1,0,1,1,0]
904     Output: 4
905     Explanation: Flip the first zero will get the maximum number of consecutive 1s. After flipping, the maximum
906     ↪ number of consecutive 1s is 4.
907
908     Example 2:
909
910     Input: nums = [1,0,1,1,0,1]
911     Output: 4
912     ``
913
914     > 比 485 复杂但基本也是模板
915
916     ``python
917     class Solution:
918         def findMaxConsecutiveOnes(self, nums: List[int]) -> int:
919             # get(key) 方法在 key (键) 不在字典中时, 可以返回默认值 None 或者设置的默认值
920             # dict[key] 在 key (键) 不在字典中时, 会触发 KeyError 异常。
921             slow = 0
922             num_zero = 0
923             hashm = {}
924             res = 0
925
926             for fast in range(len(nums)):
927                 tail = nums[fast]
928                 hashm[tail] = hashm.get(tail, 0) + 1
929                 if hashm.get(0, 0) <= 1: # 如果直接 call dict[key] 就会报错, 因为没有 0, 可能
930                     res = max(res, fast - slow + 1)
931
932                 while hashm.get(0, 0) > 1: # 如果直接 call dict[key] 就会报错, 因为没有 0, 可能
933                     head = nums[slow]
934                     hashm[head] -= 1
935                     if hashm[head] == 0:
936                         del hashm[head]
937
938                 slow += 1
939             return res
940     ``
941
942     ### 1004. Max Consecutive Ones III https://leetcode.com/problems/max-consecutive-ones-iii/
943
944     Given a binary array nums and an integer k, return the maximum number of consecutive 1's in the array if you can
945     ↪ flip at most k 0's.
946     ``
947
948     Example 1:
949
950

```

```

942 Input: nums = [1,1,1,0,0,0,1,1,1,1,0], k = 2
943 Output: 6
944 Explanation: [1,1,1,0,0,1,1,1,1,1]
945 Bolded numbers were flipped from 0 to 1. The longest subarray is underlined.
946 Example 2:
947
948 Input: nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3
949 Output: 10
950 Explanation: [0,0,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1]
951 Bolded numbers were flipped from 0 to 1. The longest subarray is underlined.
952 ```
953 > 和 487 完全一样, 从 1 变成 k, 这个模板不错!
954
955 ```python
956 class Solution:
957     def longestOnes(self, nums: List[int], k: int) -> int:
958         slow = 0
959         num_zero = 0
960         hashm = {}
961         res = 0
962         for fast in range(len(nums)):
963             tail = nums[fast]
964             hashm[tail] = hashm.get(tail, 0) + 1
965             if hashm.get(0, 0) <= k: # 如果直接 call dict[key] 就会报错, 因为没有 0, 可能
966                 res = max(res, fast - slow + 1)
967
968             while hashm.get(0, 0) > k: # 如果直接 call dict[key] 就会报错, 因为没有 0, 可能
969                 head = nums[slow]
970                 hashm[head] -= 1
971                 if hashm[head] == 0:
972                     del hashm[head]
973
974                 slow += 1
975         return res
976 ```
977
978 ### 1446. Consecutive Characters https://leetcode.com/problems/consecutive-characters/
979 The power of the string is the maximum length of a non-empty substring that contains only one unique
980 ↪ character. Given a string s, return the power of s.
981
982 Example 1:
983
984 Input: s = "leetcode"
985 Output: 2
986 Explanation: The substring "ee" is of length 2 with the character 'e' only.
987
988 Example 2:
989
990 Input: s = "abbcccdddeeeedcbba"
991 Output: 5
992 Explanation: The substring "eeeee" is of length 5 with the character 'e' only.
993
994 > one pass 这种相邻的题目是一类题, substring 的这个是最简单的!

```

```

993 ```python
994 class Solution:
995     def maxPower(self, s: str) -> int:
996         res = 1 # 这个初始化是 1 不是 0
997         maxtemp = 1
998         for i in range(1, len(s)):
999             if s[i] == s[i-1]:
1000                 maxtemp += 1
1001                 res = max(res, maxtemp)
1002             else:
1003                 maxtemp = 1
1004         return res
1005         # time o(n), space o(1)
1006 ```
1007
1008 ### Subarray 题型总结!
1009 > 有一类就是和 K 结合, product less than K, summary less than K
1010
1011 ### 643. Maximum Average Subarray I https://leetcode.com/problems/maximum-average-subarray-i/
1012 You are given an integer array nums consisting of n elements, and an integer k. Find a contiguous subarray whose
1013 ↪ length is equal to k that has the maximum average value and return this value. Any answer with a calculation
1014 ↪ error less than 10-5 will be accepted.
1015
1016 Example 1:
1017
1018 Input: nums = [1,12,-5,-6,50,3], k = 4
1019 Output: 12.75000
1020 Explanation: Maximum average is (12 - 5 - 6 + 50) / 4 = 51 / 4 = 12.75
1021
1022 Example 2:
1023
1024 Input: nums = [5], k = 1
1025 Output: 5.00000
1026
1027 > subarray 经典入门题
1028 ```python
1029 class Solution:
1030     def findMaxAverage(self, nums: List[int], k: int) -> float:
1031         slow = 0
1032         n = len(nums)
1033         sum_ = 0
1034         res = -inf
1035         for fast in range(n):
1036             sum_ += nums[fast]
1037             # 不满足窗口条件, 对 slow 操作, 有时候也用 while? 因为这个是 fix 窗口, 所以用 if
1038             if fast >= k - 1:
1039                 sum_ -= nums[slow]
1040                 slow += 1
1041             # 这步可以理解! 满足条件然后操作
1042             if fast - slow + 1 == k:
1043                 res = max(res, sum_ / k)
1044         return res
1045 ```

```

```

1043
1044 ### 644. Maximum Average Subarray II https://leetcode.com/problems/maximum-average-subarray-ii/ 题目
1045 You are given an integer array nums consisting of n elements, and an integer k. Find a contiguous subarray whose
    ↳ length is greater than or equal to k that has the maximum average value and return this value. Any answer
    ↳ with a calculation error less than 10-5 will be accepted.
1046 ```
1047 Example 1:
1048
1049 Input: nums = [1,12,-5,-6,50,3], k = 4
1050 Output: 12.75000
1051 Explanation:
1052 - When the length is 4, averages are [0.5, 12.75, 10.5] and the maximum average is 12.75
1053 - When the length is 5, averages are [10.4, 10.8] and the maximum average is 10.8
1054 - When the length is 6, averages are [9.16667] and the maximum average is 9.16667
1055 The maximum average is when we choose a subarray of length 4 (i.e., the sub array [12, -5, -6, 50]) which has
    ↳ the max average 12.75, so we return 12.75
1056 Note that we do not consider the subarrays of length < 4.
1057 Example 2:
1058
1059 Input: nums = [5], k = 1
1060 Output: 5.00000
1061 ```
1062 > 虽然是 643 相似, 但这个题是二分法
1063 ```python
1064 class Solution:
1065     def findMaxAverage(self, nums: List[int], k: int) -> float:
1066         if not nums:
1067             return 0
1068         start, end = min(nums), max(nums)
1069         while end - start > 1e-5:
1070             mid = (start + end) / 2
1071             if self.check_subarray(nums, k, mid):
1072                 start = mid
1073             else:
1074                 end = mid
1075         return start
1076     def check_subarray(self, nums, k, average):
1077         prefix_sum = [0]
1078         for num in nums:
1079             prefix_sum.append(prefix_sum[-1] + num - average)
1080
1081         min_prefix_sum = 0
1082         for i in range(k, len(nums) + 1):
1083             if prefix_sum[i] - min_prefix_sum >= 0:
1084                 return True
1085             min_prefix_sum = min(min_prefix_sum, prefix_sum[i - k + 1])
1086         return False
1087 ```
1088
1089
1090 ### 713. Subarray Product Less Than K https://leetcode.com/problems/subarray-product-less-than-k/

```

1091 Given an array of integers `nums` and an integer `k`, return the number of contiguous subarrays where the product of
↳ all the elements in the subarray is strictly less than `k`.

1092 ```

1093 Example 1:

1094

1095 Input: `nums = [10,5,2,6]`, `k = 100`

1096 Output: 8

1097 Explanation: The 8 subarrays that have product less than 100 are:

1098 `[10]`, `[5]`, `[2]`, `[6]`, `[10, 5]`, `[5, 2]`, `[2, 6]`, `[5, 2, 6]`

1099 Note that `[10, 5, 2]` is not included as the product of 100 is not strictly less than `k`.

1100 Example 2:

1101

1102 Input: `nums = [1,2,3]`, `k = 0`

1103 Output: 0

1104 ```

1105 > 之前的模板要改，不能直接用，因为先判断的话，其实 `window` 并不合法，所以要在最后存结果

1106 ```python

1107 class Solution:

1108 def numSubarrayProductLessThanK(self, nums: List[int], k: int) -> int:

1109 # 这个题不错，能不能输出所有的 pair

1110 if k <= 1: return 0

1111 slow = 0

1112 prod = 1

1113 res = 0

1114 for fast in range(len(nums)):

1115 prod *= nums[fast]

1116 # if prod < k: 都是错误的，不能在这存结果，跟之前的模板不同，这个 window 不合法!

1117 # res += fast - slow + 1 # 放在这就错误了，没有更新 slow

1118 while prod >= k:

1119 prod /= nums[slow]

1120 slow += 1

1121 res += fast - slow + 1

1122 return res

1123 # time o(n), space o(1)

1124

1125 # brute force # time o(n^2) and space o(1) 会超时

1126 # res = 0

1127 # for i in range(len(nums)):

1128 # prod = 1

1129 # for j in range(i, len(nums)):

1130 # prod *= nums[j]

1131 # if prod < k:

1132 # res += 1

1133 # else:

1134 # continue

1135 # return res

1136 # time o(n^2), space o(1)

1137 ```

1138 Follow-up: 如何输出所有符合的 subarray # 如果要输出所有的 subarrays 相当于在 `nums[slow:fast]` 这个 window 里的 subset 所有

1139 ↳ 合集? 不好做!

1140


```

1141
1142
1143 面试题 Subsequence 的类型题! 很多要用 DP?
1144
1145 ### 674. Longest Continuous Increasing Subsequence
1146 ↪ https://leetcode.com/problems/longest-continuous-increasing-subsequence/
1147 Given an unsorted array of integers nums, return the length of the longest continuous increasing subsequence
1148 ↪ (i.e. subarray). The subsequence must be strictly increasing. A continuous increasing subsequence is defined
1149 ↪ by two indices l and r (l < r) such that it is [nums[l], nums[l + 1], ..., nums[r - 1], nums[r]] and for
1150 ↪ each l <= i < r, nums[i] < nums[i + 1].
1151 ```
1152 Example 1:
1153
1154 Input: nums = [1,3,5,4,7]
1155 Output: 3
1156 Explanation: The longest continuous increasing subsequence is [1,3,5] with length 3.
1157 Even though [1,3,5,7] is an increasing subsequence, it is not continuous as elements 5 and 7 are separated by
1158 ↪ element
1159 4.
1160 Example 2:
1161
1162 Input: nums = [2,2,2,2,2]
1163 Output: 1
1164 Explanation: The longest continuous increasing subsequence is [2] with length 1. Note that it must be strictly
1165 increasing.
1166 ```
1167 > 这个题挺不错, 很多种解法!
1168 ```python
1169 class Solution:
1170     def findLengthOfLCIS(self, nums: List[int]) -> int:
1171
1172         # 这个题虽然是一个 easy, 但是很多种解法, 很多不错的方法!
1173         # way 1 - 上来想到的是 simulate 也可以叫 greedy?
1174         # 这里比较的是 i + 1 和 i, 然后从 len(nums) - 1 开始的
1175         if len(nums) == 0: return 0
1176         res = 1
1177         count = 1
1178         for i in range(len(nums) - 1):
1179             if nums[i + 1] > nums[i]:
1180                 count += 1
1181             else:
1182                 count = 1
1183             res = max(res, count)
1184         return res
1185
1186         # # 这个写法也可以, 从 1 开始的, 之前写的有问题
1187         # res = 1
1188         # count = 1
1189         # for i in range(1, len(nums)):
1190             # if nums[i] > nums[i - 1]:
1191                 # count += 1
1192             # else:

```

```

1188     #         count = 1
1189     #     res = max(res, count)
1190     # return res
1191
1192
1193     # way 2 - sliding window, double pointer 这里面 sliding window 有 2 种可以做!
1194     # 方法 1 是锚钉, 像 solution 说的那种
1195     # res, anchor = 0, 0
1196     # for i in range(len(nums)):
1197     #     if i and nums[i - 1] >= nums[i]:
1198     #         anchor = i
1199     #     res = max(res, i - anchor + 1) # 在出现 nums[i - 1] >= nums[i] 之前, anchor 总是 0, 没有更新, 所以这个就是
1200     #         ↪ 记录的最大值
1201     # return res
1202
1203     # # 方法 2 是滑动 while 然后求最大长度 这个理解的不错! 复习一下 sliding windows!
1204     # if not nums: return 0
1205     # slow, fast = 0, 1
1206     # res = 1
1207     # n = len(nums)
1208     # for fast in range(n):
1209     #     while fast < n and nums[fast] > nums[fast - 1]:
1210     #         fast += 1
1211     #     res = max(res, fast - slow)
1212     #     slow = fast
1213     # return res
1214     # # time 一样是 o(n) and space is o(1)
1215
1216     # way 3 - DP 方法!
1217     # 确定 dp 的含义 dp[i] 以下标 i 为结尾的数组的连续递增子序列长度
1218     if len(nums) == 0: return 0
1219     dp = [1] * len(nums)
1220     res = 1
1221     for i in range(len(nums) - 1):
1222         if nums[i + 1] > nums[i]:
1223             dp[i + 1] = dp[i] + 1
1224             res = max(res, dp[i + 1])
1225     return res

```

...

```

1239
1240
1241 困难 题 sliding window 困难
1242 ### 76. Minimum Window Substring https://leetcode.com/problems/minimum-window-substring/
1243 Given two strings s and t of lengths m and n respectively, return the minimum window substring of s such that
    ↳ every character in t (including duplicates) is included in the window. If there is no such substring, return
    ↳ the empty string "". The testcases will be generated such that the answer is unique. A substring is a
    ↳ contiguous sequence of characters within the string.
1244 ```
1245 Example 1:
1246
1247 Input: s = "ADOBECODEBANC", t = "ABC"
1248 Output: "BANC"
1249 Explanation: The minimum window substring "BANC" includes 'A', 'B', and 'C' from string t.
1250 Example 2:
1251
1252 Input: s = "a", t = "a"
1253 Output: "a"
1254 Explanation: The entire string s is the minimum window.
1255 Example 3:
1256
1257 Input: s = "a", t = "aa"
1258 Output: ""
1259 Explanation: Both 'a's from t must be included in the window.
1260 Since the largest window of s only has one 'a', return empty string.
1261 ```
1262 > 很复杂，不会
1263 ```python
1264 class Solution:
1265     def minWindow(self, s: str, t: str) -> str:
1266
1267         # write your code here
1268         target, source = t, s
1269
1270         if len(target) == 0 or len(source) == 0:
1271             return ''
1272
1273         m, n = len(target), len(source)
1274         target_c, sub_c = {}, {}
1275
1276         for i in range(m):
1277             target_c[target[i]] = target_c.get(target[i], 0) + 1
1278
1279         fast = 0
1280         matched_chars = 0
1281         start, substring_len = 0, float('inf')
1282
1283         for slow in range(n):
1284
1285             while fast < n and matched_chars < len(target_c):
1286                 sub_c[source[fast]] = sub_c.get(source[fast], 0) + 1
1287                 if sub_c[source[fast]] == target_c.get(source[fast], 0):

```

```

1288         matched_chars += 1
1289         fast += 1
1290
1291         if matched_chars == len(target_c):
1292             if substring_len > fast - slow:
1293                 substring_len = fast - slow
1294                 start = slow
1295
1296         sub_c[source[slow]] -= 1
1297         if sub_c[source[slow]] == target_c.get(source[slow], 0) - 1:
1298             matched_chars -= 1
1299
1300         if substring_len == float('inf') :
1301             return ''
1302
1303         return source[start : start + substring_len]
1304     ```
1305
1306     ### 239. Sliding Window Maximum https://leetcode.com/problems/sliding-window-maximum/
1307     You are given an array of integers nums, there is a sliding window of size k which is moving from the very left
1308     ↪ of the array to the very right. You can only see the k numbers in the window. Each time the sliding window
1309     ↪ moves right by one position. Return the max sliding window.
1310
1311     Example 1:
1312     Input: nums = [1,3,-1,-3,5,3,6,7], k = 3
1313     Output: [3,3,5,5,6,7]
1314     Explanation:
1315     Window position           Max
1316     -----
1317     [1  3  -1] -3  5  3  6  7      3
1318     1 [3  -1  -3] 5  3  6  7      3
1319     1  3 [-1  -3  5] 3  6  7      5
1320     1  3  -1 [-3  5  3] 6  7      5
1321     1  3  -1  -3 [5  3  6] 7      6
1322     1  3  -1  -3  5 [3  6  7]     7
1323
1324     Example 2:
1325     Input: nums = [1], k = 1
1326     Output: [1]
1327     ```
1328
1329     > hard 不会, 没有做出来, 跳过
1330     ```python
1331     class Solution:
1332         def maxSlidingWindow(self, nums: List[int], k: int) -> List[int]:
1333
1334             n = len(nums)
1335             q = collections.deque()
1336             for i in range(k):
1337                 while q and nums[i] >= nums[q[-1]]:
1338                     q.pop()
1339                 q.append(i)

```

```

1338     ans = [nums[q[0]]]
1339     for i in range(k, n):
1340         while q and nums[i] >= nums[q[-1]]:
1341             q.pop()
1342         q.append(i)
1343         while q[0] <= i - k:
1344             q.popleft()
1345         ans.append(nums[q[0]])
1346
1347     return ans
1348
1349 """
1350 ### 30. Substring with Concatenation of All Words
1351 ↪ https://leetcode.com/problems/substring-with-concatenation-of-all-words/ [hard]
1352 You are given a string s and an array of strings words of the same length. Return all starting indices of
1353 ↪ substring(s) in s that is a concatenation of each word in words exactly once, in any order, and without any
1354 ↪ intervening characters. You can return the answer in any order.
1355
1356 Example 1:
1357 """
1358 Input: s = "barfoothefoobarman", words = ["foo","bar"]
1359 Output: [0,9]
1360 Explanation: Substrings starting at index 0 and 9 are "barfoo" and "foobar" respectively.
1361 The output order does not matter, returning [9,0] is fine too.
1362 """
1363
1364 Example 2:
1365 """
1366 Input: s = "wordgoodgoodgoodbestword", words = ["word","good","best","word"]
1367 Output: []
1368 """
1369
1370 Example 3:
1371 """
1372 Input: s = "barfoofoobarthefoobarman", words = ["bar","foo","the"]
1373 Output: [6,9,12]
1374 """
1375
1376 ```python
1377 # 完全不会，太难！
1378 class Solution:
1379     def findSubstring(self, s: str, words: List[str]) -> List[int]:
1380         from collections import Counter
1381         if not s or not words: return []
1382         one_word = len(words[0])
1383         word_num = len(words)
1384         n = len(s)
1385         words = Counter(words)
1386         res = []
1387         for i in range(0, one_word):
1388             cur_cnt = 0
1389             left = i
1390             right = i
1391             cur_counter = Counter()
1392             while right + one_word <= n:

```

```

1387         w = s[right:right + one_word]
1388         right += one_word
1389         cur_Counter[w] += 1
1390         cur_cnt += 1
1391         while cur_Counter[w] > words[w]:
1392             left_w = s[left:left+one_word]
1393             left += one_word
1394             cur_Counter[left_w] -= 1
1395             cur_cnt -= 1
1396         if cur_cnt == word_num :
1397             res.append(left)
1398     return res
1399     ```

```

3 Binary search

4 Recursion backtracking

5 Graph DFS

6 Graph BFS

7 Binary Tree

8 Stack

9 Heap (Priority Queue)

```

1  # heap
2
3  ## 定义
4  1. 分为最小堆 minheap, 和最大堆, maxheap, 也就是最小元素或者最大元素在堆顶
5  2. 堆是一个完全二叉树, 但堆的底层实现一般是数组, 而不是二叉树
6  3. 孩子节点都比父亲节点大, 但是左右孩子的大小不影响
7  4. 堆不是 binary search tree
8  5. 堆的操作是从上到下, 从左到右
9
10 ## 基本操作 - 高度是  $\log n$ 
11 1. 构建堆 heapify -  $O(n)$ 
12 2. 遍历堆  $O(n \log n)$ 
13 3. add -  $O(n)$ 
14 4. remove - 理论上是  $O(\log n)$  但实际上 python 的库函数是 for loop 遍历的, 所以是  $O(n)$ 
15 5. pop -  $O(\log n)$ , push 也是  $O(\log n)$ 
16 6. min or max -  $O(1)$ 
17 7. 由于是数组操作, 选定 k, 父亲是  $k/2$ , 左孩子  $k \times 2$ , 右孩子  $k \times 2 + 1$ 
18 8. 可以结合 hashmap 去查询或者 remove 指定值

```

```

19
20
21 ## 使用条件
22 1. 找最大值或者最小值 (60%)
23 2. 找第 k 大 (pop k 次 复杂度  $O(n\log k)$ ) (50%)
24 3. 要求  $\log n$  时间对数据进行操作 (40%)
25
26 ## 堆不能解决的问题
27 1. 查询比某个数大的最小值或者最近接的值 (平衡二叉树 balanced bst 才可以解决)
28 2. 找某段区间的最大值最小值 (线段树 segmenttree 可以解决)
29 3.  $O(n)$  找第 k 大的数 (需要使用快排中的 partition 操作)
30
31
32 ## time complexity
33 ## space complexity
34
35
36 ## template
37 ```python
38
39 from heapq import heappush, heappop
40
41 class Heap:
42     def __init__(self):
43         self.minheap()
44         self.deleted_set = set()
45
46     def push(self, index, val):
47         heappush(self.minheap, (val, index))
48
49     def _lazy_deletion(self):
50         while self.minheap and self.minheap[0][1] in self.deleted_set:
51             heappop(self.minheap)
52
53     def top(self):
54         self._lazy_deletion()
55         return self.minheap[0]
56
57     def pop(self): # 移除顶端元素
58         self._lazy_deletion()
59         heappop(self.minheap)
60
61     def delete(self, index):
62         self.deleted_set.add(index)
63
64     def is_empty(self):
65         return not bool(self.minheap)
66
67 ```
68
69 ## 几种类型的双指针及相关题目
70

```

```

71 ### Example 264. Ugly Number II
72 https://leetcode.com/problems/ugly-number-ii/
73
74 An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5.
75
76 Given an integer n, return the nth ugly number.
77
78 Example 1:
79 ```
80 Input: n = 10
81 Output: 12
82 Explanation: [1, 2, 3, 4, 5, 6, 8, 9, 10, 12] is the sequence of the first 10 ugly numbers.
83 ```
84 Example 2:
85 ```
86 Input: n = 1
87 Output: 1
88 Explanation: 1 has no prime factors, therefore all of its prime factors are limited to 2, 3, and 5.
89 ```
90 >solution
91 ```python
92     # 用 heap 来做，不断的找最小值，然后 push 回去，时间复杂度  $O(n \log n)$ 
93
94     import heapq
95
96     heap = [1]
97     visited = set([1])
98
99     min_val = None
100     for i in range(n): # n 次操作，也是 nth 的最小值
101         min_val = heapq.heappop(heap)
102         for factor in [2, 3, 5]:
103             if min_val * factor not in visited:
104                 visited.add(min_val * factor)
105                 heapq.heappush(heap, min_val * factor)
106
107     return min_val
108 ```
109
110 ### 973. K Closest Points to Origin
111 https://leetcode.com/problems/k-closest-points-to-origin/
112
113 Given an array of points where points[i] = [xi, yi] represents a point on the X-Y plane and an integer k, return
114 ↪ the k closest points to the origin (0, 0).
115
116 The distance between two points on the X-Y plane is the Euclidean distance (i.e.,  $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ ).
117
118 You may return the answer in any order. The answer is guaranteed to be unique (except for the order that it is
119 ↪ in).
120
121 Example 1:
122 ```

```



```

121 Input: points = [[1,3],[-2,2]], k = 1
122 Output: [[-2,2]]
123 Explanation:
124 The distance between (1, 3) and the origin is sqrt(10).
125 The distance between (-2, 2) and the origin is sqrt(8).
126 Since sqrt(8) < sqrt(10), (-2, 2) is closer to the origin.
127 We only want the closest k = 1 points from the origin, so the answer is just [[-2,2]].
128 ```
129 Example 2:
130 ```
131 Input: points = [[3,3],[5,-1],[-2,4]], k = 2
132 Output: [[3,3],[-2,4]]
133 Explanation: The answer [[-2,4],[3,3]] would also be accepted.
134 ```
135
136 >solution
137
138 ```python
139 # # heap 最小堆 - 把所有点都放入最小堆, 然后用最小堆取出 k 个, 时间 o(nlogn) 空间 o(n) + o(k)
140 # # 遍历堆 o(nlogn) 所以时间是这个 level, 空间是因为开了 heap + res 的部分, 单独储存 res
141
142
143 # import heapq
144 # heap = []
145
146 # for point in points:
147 #     cur_dis = point[0] ** 2 + point[1] ** 2
148 #     heapq.heappush(heap, (cur_dis, point)) # 注意加进去的时候还是看 cur_dis 自动排序好了
149
150 # res = []
151 # i = 0
152 # while i < k:
153 #     _, point = heapq.heappop(heap)
154 #     res.append(point)
155 #     i += 1
156
157 # return res
158
159 # 最大堆 更优一些, 因为不需要重新开整个 heap - 时间 o(nlogk), 空间 o(k)
160 heap = []
161 for point in points:
162     cur_dis = point[0] ** 2 + point[1] ** 2
163     heapq.heappush(heap, (-cur_dis, point)) # 注意加进去的时候还是看 cur_dis 自动排序好了
164     if len(heap) > k:
165         heapq.heappop(heap)
166
167 res = []
168 i = 0
169 while i < k:
170     _, point = heapq.heappop(heap)
171     res.append(point)
172     i += 1

```

```

173         return res
174     """
175
176
177     ### Example: Lintcode 545 · Top k Largest Numbers II
178     https://www.lintcode.com/problem/545/
179
180     Description
181     Implement a data structure, provide two interfaces:
182
183     add(number). Add a new number in the data structure.
184     topk(). Return the top k largest numbers in this data structure. k is given when we create the data structure.
185
186 >solution
187 ```python
188 import heapq
189 class Solution:
190     """
191     @param: k: An integer
192     """
193     def __init__(self, k):
194         # do initialization if necessary
195         self.k = k
196         self.heap = []
197
198     """
199     @param: num: Number to be added
200     @return: nothing
201     """
202     def add(self, num):
203         # write your code here
204         heapq.heappush(self.heap, num)
205         if len(self.heap) > self.k:
206             heapq.heappop(self.heap)
207
208     """
209     @return: Top k element
210     """
211     def topk(self):
212         # write your code here
213         return sorted(self.heap, reverse=True)
214 ```
215
216
217     ### 253. Meeting Rooms II
218     https://leetcode.com/problems/meeting-rooms-ii/
219
220     Given an array of meeting time intervals intervals where intervals[i] = [starti, endi], return the minimum
221     ↪ number of conference rooms required.
222
223     Example 1:
224     """

```

```

224 Input: intervals = [[0,30],[5,10],[15,20]]
225 Output: 2
226 ```
227 Example 2:
228 ```
229 Input: intervals = [[7,10],[2,4]]
230 Output: 1
231 ```
232
233 >solution
234 https://leetcode.com/problems/meeting-rooms-ii/solution/
235
236 ```python
237 class Solution:
238     def minMeetingRooms(self, intervals: List[List[int]]) -> int:
239
240         # If there is no meeting to schedule then no room needs to be allocated.
241         if not intervals:
242             return 0
243
244         # The heap initialization
245         free_rooms = []
246
247         # Sort the meetings in increasing order of their start time.
248         intervals.sort(key= lambda x: x[0])
249
250         # Add the first meeting. We have to give a new room to the first meeting.
251         heapq.heappush(free_rooms, intervals[0][1])
252
253         # For all the remaining meeting rooms
254         for i in intervals[1:]:
255
256             # If the room due to free up the earliest is free, assign that room to this meeting.
257             if free_rooms[0] <= i[0]:
258                 heapq.heappop(free_rooms)
259
260             # If a new room is to be assigned, then also we add to the heap,
261             # If an old room is allocated, then also we have to add to the heap with updated end time.
262             heapq.heappush(free_rooms, i[1])
263
264         # The size of the heap tells us the minimum rooms required for all the meetings.
265         return len(free_rooms)
266
267 # Complexity Analysis
268
269 # Time Complexity:  $O(N\log N)$ 
270
271 # There are two major portions that take up time here. One is sorting of the array that takes  $O(N\log N)$ 
272 ↪ considering that the array consists of  $NN$  elements. Then we have the min-heap. In the worst case, all  $NN$ 
273 ↪ meetings will collide with each other. In any case we have  $NN$  add operations on the heap. In the worst case
274 ↪ we will have  $NN$  extract-min operations as well. Overall complexity being  $(N\log N)(N\log N)$  since extract-min
275 ↪ operation on a heap takes  $O(\log N)$ .

```

```

272
273 # Space Complexity:  $O(N)$  because we construct the min-heap and that can contain  $NN$  elements in the worst case as
    ↳ described above in the time complexity section. Hence, the space complexity is  $O(N)$ .
274 ```
275
276
277 ### Example: 373. Find K Pairs with Smallest Sums
278 https://leetcode.com/problems/find-k-pairs-with-smallest-sums/
279
280 You are given two integer arrays nums1 and nums2 sorted in ascending order and an integer k.
281
282 Define a pair (u, v) which consists of one element from the first array and one element from the second array.
283
284 Return the k pairs (u1, v1), (u2, v2), ..., (uk, vk) with the smallest sums.
285
286 Example 1:
287 ```
288 Input: nums1 = [1,7,11], nums2 = [2,4,6], k = 3
289 Output: [[1,2],[1,4],[1,6]]
290 Explanation: The first 3 pairs are returned from the sequence:
    ↳ [1,2],[1,4],[1,6],[7,2],[7,4],[11,2],[7,6],[11,4],[11,6]
291 ```
292 Example 2:
293 ```
294 Input: nums1 = [1,1,2], nums2 = [1,2,3], k = 2
295 Output: [[1,1],[1,1]]
296 Explanation: The first 2 pairs are returned from the sequence:
    ↳ [1,1],[1,1],[1,2],[2,1],[1,2],[2,2],[1,3],[1,3],[2,3]
297 ```
298 Example 3:
299 ```
300 Input: nums1 = [1,2], nums2 = [3], k = 3
301 Output: [[1,3],[2,3]]
302 Explanation: All possible pairs are returned from the sequence: [1,3],[2,3]
303 ```
304
305 >solution
306 ```python
307 # 用 heap 来解 - 和 973 很像, 但是是二维遍历, 然后也是用最大堆
308 import heapq
309 heap = []
310 for i in range(min(k, len(nums1))):
311     for j in range(min(k, len(nums2))):
312         if len(heap) < k:
313             heapq.heappush(heap, ((-nums1[i] - nums2[j]), i, j))
314         else:
315             if nums1[i] + nums2[j] < -heap[0][0]:
316                 heappop(heap)
317                 heappush(heap, (-nums1[i] + nums2[j]), i, j))
318 res = []
319 for _, i, j in heap:
320     res.append([nums1[i], nums2[j]])

```

```

321
322         return res
323
324         # time - klogk? 不太确定
325         # space - o(k)
326     ...
327
328     ### Example: 215. Kth Largest Element in an Array
329     https://leetcode.com/problems/kth-largest-element-in-an-array/
330     Given an integer array nums and an integer k, return the kth largest element in the array.
331
332     Note that it is the kth largest element in the sorted order, not the kth distinct element.
333
334     Example 1:
335     ...
336     Input: nums = [3,2,1,5,6,4], k = 2
337     Output: 5
338     ...
339     Example 2:
340     ...
341     Input: nums = [3,2,3,1,2,4,5,5,6], k = 4
342     Output: 4
343     ...
344
345     >solution
346     ```python
347         # 暴力干 - 但没有什么意思, python 的 sort 时间复杂度是多少? 不清楚
348         # nums.sort()
349         # return nums[-k]
350
351         # That would be an algorithm of O(NlogN) time complexity and
352         # O(1) space complexity.
353
354         # heap
355         import heapq
356         return heapq.nlargest(k, nums)[-1]
357
358         # Time complexity : O(Nlogk).
359         # Space complexity : O(k) to store the heap elements.
360     ...
361
362
363     ### Example : 692. Top K Frequent Words
364     https://leetcode.com/problems/top-k-frequent-words/
365
366     Given an array of strings words and an integer k, return the k most frequent strings.
367
368     Return the answer sorted by the frequency from highest to lowest. Sort the words with the same frequency by
    ↪ their lexicographical order.
369
370
371     Example 1:

```

```

372     ``
373 Input: words = ["i","love","leetcode","i","love","coding"], k = 2
374 Output: ["i","love"]
375 Explanation: "i" and "love" are the two most frequent words.
376 Note that "i" comes before "love" due to a lower alphabetical order.
377     ``
378 Example 2:
379     ``
380 Input: words = ["the","day","is","sunny","the","the","the","sunny","is","is"], k = 4
381 Output: ["the","is","sunny","day"]
382 Explanation: "the", "is", "sunny" and "day" are the four most frequent words, with the number of occurrence
    ↪ being 4, 3, 2 and 1 respectively.
383     ``
384
385 >Solution
386 ```python
387 class Solution:
388     def topKFrequent(self, words: List[str], k: int) -> List[str]:
389
390         # heap 很显然的解法, 但不想用这么多额外的开销
391         counts = defaultdict(lambda: 0)
392         for word in words:
393             counts[word] += 1
394
395         inverse = defaultdict(lambda: [])
396         for word, count in counts.items():
397             heappush(inverse[count], word)
398
399         res = []
400
401         for count in nlargest(k, counts.values()):
402             res.append(heapop(inverse[count]))
403
404         return res
405     ``
406
407
408 ### Example: 658. Find K Closest Elements
409 https://leetcode.com/problems/find-k-closest-elements/
410
411 Given a sorted integer array arr, two integers k and x, return the k closest integers to x in the array. The
    ↪ result should also be sorted in ascending order.
412
413 An integer a is closer to x than an integer b if:
414     ``
415     |a - x| < |b - x|, or
416     |a - x| == |b - x| and a < b
417     ``
418
419 Example 1:
420     ``
421 Input: arr = [1,2,3,4,5], k = 4, x = 3

```

```

422 Output: [1,2,3,4]
423 ```
424 Example 2:
425 ```
426 Input: arr = [1,2,3,4,5], k = 4, x = -1
427 Output: [1,2,3,4]
428 ```
429
430 >Solution
431 ```python
432 class Solution:
433     def findClosestElements(self, arr: List[int], k: int, x: int) -> List[int]:
434         # way 1 - double pointer
435         # way 2 - binary search
436
437         # way 3 - heap
438         import heapq
439         heap = []
440
441         for num in arr:
442             dis = abs(num - x)
443             heapq.heappush(heap, (dis, num))
444             # if len(heap) > k: # 用最大堆有问题，就是会忽视这个条件  $|a - x| == |b - x|$  and  $a < b$ 
445             #     heapq.heappop(heap)
446
447         res = []
448         i = 0
449         while i < k:
450             _, num = heapq.heappop(heap)
451             res.append(num)
452             i += 1
453
454         return sorted(res) # 别忘了最后要 sorted() 如果是 res.sort() 会返回 []
455 ```
456
457
458
459 ### 632. Smallest Range Covering Elements from K Lists
460 ↪ https://leetcode.com/problems/smallest-range-covering-elements-from-k-lists/
461 You have k lists of sorted integers in non-decreasing order. Find the smallest range that includes at least one
462 ↪ number from each of the k lists. We define the range [a, b] is smaller than range [c, d] if  $b - a < d - c$  or
463 ↪  $a < c$  if  $b - a == d - c$ .
464 ```
465 Example 1:
466
467 Input: nums = [[4,10,15,24,26],[0,9,12,20],[5,18,22,30]]
468 Output: [20,24]
469 Explanation:
470 List 1: [4, 10, 15, 24,26], 24 is in range [20,24].
471 List 2: [0, 9, 12, 20], 20 is in range [20,24].
472 List 3: [5, 18, 22, 30], 22 is in range [20,24].
473 Example 2:

```

```

471
472 Input: nums = [[1,2,3],[1,2,3],[1,2,3]]
473 Output: [1,1]
474 ```
475
476 ```python
477 class Solution:
478     def smallestRange(self, nums: List[List[int]]) -> List[int]:
479         # 各种暴力都超时
480         # 还有可以 hash + sliding window 就是超级麻烦, double pointer 也是超时
481
482         # 这个是 heap 的解
483         rangeLeft, rangeRight = -10**9, 10**9
484         maxValue = max(vec[0] for vec in nums)
485         priorityQueue = [(vec[0], i, 0) for i, vec in enumerate(nums)]
486         heapq.heapify(priorityQueue)
487
488         while True:
489             minValue, row, idx = heapq.heappop(priorityQueue)
490             if maxValue - minValue < rangeRight - rangeLeft:
491                 rangeLeft, rangeRight = minValue, maxValue
492             if idx == len(nums[row]) - 1:
493                 break
494             maxValue = max(maxValue, nums[row][idx + 1])
495             heapq.heappush(priorityQueue, (nums[row][idx + 1], row, idx + 1))
496
497         return [rangeLeft, rangeRight]
498     ```

```

10 DP

```

1 # DP
2
3
4 152. Subarray DP
5
6 ### 152. Maximum Product Subarray https://leetcode.com/problems/maximum-product-subarray/
7 Given an integer array nums, find a contiguous non-empty subarray within the array that has the largest product,
8 ↪ and return the product. The test cases are generated so that the answer will fit in a 32-bit integer. A
9 ↪ subarray is a contiguous subsequence of the array.
10
11 Example 1:
12
13 Input: nums = [2,3,-2,4]
14 Output: 6
15 Explanation: [2,3] has the largest product 6.
16
17 Example 2:
18
19 Input: nums = [-2,0,-1]
20 Output: 0

```



```

18 Explanation: The result cannot be 2, because [-2,-1] is not a subarray.
19 ...
20 > 经典 DP 在 subarray 类型的! olk
21 ```python
22 class Solution:
23     def maxProduct(self, nums: List[int]) -> int:
24
25         # dp 解法和之前不一样的
26         n = len(nums)
27         dpmax = [0] * n
28         dpmin = [0] * n
29         dpmax[0] = nums[0]
30         dpmin[0] = nums[0]
31         res = nums[0]
32
33         for i in range(1, n):
34             dpmax[i] = max(dpmax[i-1] * nums[i], nums[i], dpmin[i-1]*nums[i])
35             dpmin[i] = min(dpmax[i-1] * nums[i], nums[i], dpmin[i-1]*nums[i])
36             res = max(res, dpmax[i])
37
38         return res
39 ...
40
41 ### 53. Maximum Subarray
42 Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest
43 ↪ sum and return its sum. A subarray is a contiguous part of an array.
44 ...
45 Example 1:
46
47 Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
48 Output: 6
49 Explanation: [4,-1,2,1] has the largest sum = 6.
50 Example 2:
51
52 Input: nums = [1]
53 Output: 1
54 Example 3:
55
56 Input: nums = [5,4,-1,7,8]
57 Output: 23
58 ...
59 > subarray 很多都是 DP 的, 有一部分可以 double pointer
60 ```python
61 class Solution:
62     def maxSubArray(self, nums: List[int]) -> int:
63         # dp[i] 数组的含义是以 nums[i] 结尾的连续子数组的最大和
64         # 关键是 dp[i] 转换的含义, 这里面要判断 dp[i-1] 的大小, 如果 >0, dp[i] 会更大, 但是 dp[i-1]<0, 重新开始, 因为
65         ↪ nums[i] 会更小如果加上 dp[i-1]
66
67         n = len(nums)
68         dp = [0] * n
69         dp[0] = nums[0]

```

```

68     for i in range(1, n):
69         if dp[i-1] > 0: # 不是判断 nums[i] 的正负
70             dp[i] = dp[i-1] + nums[i]
71         else:
72             dp[i] = nums[i]
73     return max(dp) # 返回所有的最大值
74
75     # brute force - 这是比较优化的方法，但超时
76     res = -inf
77     n = len(nums)
78     for i in range(n):
79         curr = 0
80         for j in range(i, n):
81             curr += nums[j]
82             res = max(res, curr)
83
84     return res
85
86
87
88 ### 718. Maximum Length of Repeated Subarray
89 Given two integer arrays nums1 and nums2, return the maximum length of a subarray that appears in both arrays.
90
91 Example 1:
92
93 Input: nums1 = [1,2,3,2,1], nums2 = [3,2,1,4,7]
94 Output: 3
95 Explanation: The repeated subarray with maximum length is [3,2,1].
96 Example 2:
97
98 Input: nums1 = [0,0,0,0,0], nums2 = [0,0,0,0,0]
99 Output: 5
100
101
102 ```python
103 class Solution:
104     def findLength(self, nums1: List[int], nums2: List[int]) -> int:
105
106         # # 动态规划可以，和之前的 maximum subsequence 一样的 dp 方程
107         # A = nums1
108         # B = nums2
109         # n, m = len(A), len(B)
110         # dp = [[0] * (m + 1) for _ in range(n + 1)]
111         # ans = 0
112         # for i in range(n - 1, -1, -1):
113         #     for j in range(m - 1, -1, -1):
114         #         dp[i][j] = dp[i + 1][j + 1] + 1 if A[i] == B[j] else 0
115         #         ans = max(ans, dp[i][j])
116         # return ans
117
118         # time o(n x m) and space o(n x m)
119

```

```

120
121     # way 2 - sliding window
122     def maxLength(addA: int, addB: int, length: int) -> int:
123         ret = k = 0
124         for i in range(length):
125             if A[addA + i] == B[addB + i]:
126                 k += 1
127                 ret = max(ret, k)
128             else:
129                 k = 0
130         return ret
131
132     A = nums1
133     B = nums2
134     n, m = len(A), len(B)
135     ret = 0
136     for i in range(n):
137         length = min(m, n - i)
138         ret = max(ret, maxLength(i, 0, length))
139     for i in range(m):
140         length = min(n, m - i)
141         ret = max(ret, maxLength(0, i, length))
142     return ret
143
144 # 作者: LeetCode-Solution
145 # 链接:
146     ↪ https://leetcode-cn.com/problems/maximum-length-of-repeated-subarray/solution/zui-chang-zhong-fu-zi-shu-zu-by-leetcode
147 # 来源: 力扣 (LeetCode)
148 # 著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。
149 ```
150
151
152 ### Stock price ###
153
154
155 ### 2110. Number of Smooth Descent Periods of a Stock
156 You are given an integer array prices representing the daily price history of a stock, where prices[i] is the
157     ↪ stock price on the ith day. A smooth descent period of a stock consists of one or more contiguous days such
158     ↪ that the price on each day is lower than the price on the preceding day by exactly 1. The first day of the
159     ↪ period is exempted from this rule. Return the number of smooth descent periods.
160
161 ```
162
163 Example 1:
164 Input: prices = [3,2,1,4]
165 Output: 7
166 Explanation: There are 7 smooth descent periods:
167 [3], [2], [1], [4], [3,2], [2,1], and [3,2,1]
168 Note that a period with one day is a smooth descent period by the definition.
169 Example 2:

```

```

168 Input: prices = [8,6,7,7]
169 Output: 4
170 Explanation: There are 4 smooth descent periods: [8], [6], [7], and [7]
171 Note that [8,6] is not a smooth descent period as 8 - 6 ≠ 1.
172 Example 3:
173
174 Input: prices = [1]
175 Output: 1
176 Explanation: There is 1 smooth descent period: [1]
177 ```
178
179 ```python
180 class Solution:
181     def getDescentPeriods(self, prices: List[int]) -> int:
182         n = len(prices)
183         res = 1 # 平滑下降阶段的总数, 初值为 dp[0]
184         prev = 1 # 上一个元素为结尾的平滑下降阶段的总数, 初值为 dp[0]
185         # 从 1 开始遍历数组, 按照递推式更新 prev 以及总数 res
186         for i in range(1, n):
187             if prices[i] == prices[i-1] - 1:
188                 prev += 1
189             else:
190                 prev = 1
191             res += prev
192         return res
193
194 # time o(n) and space o(1) 算是非常简单的 dp 了, 需要理解 dp 的定义和含义!
195 ```

```

11 Linked list

12 Sorting

13 Prefix sum

```

1 # Prefix sum
2
3 ## 很多是 subarray 的题
4
5 ### 238. Product of Array Except Self https://leetcode.com/problems/product-of-array-except-self/
6
7 Given an integer array nums, return an array answer such that answer[i] is equal to the product of all the
  ↳ elements of nums except nums[i]. The product of any prefix or suffix of nums is guaranteed to fit in a
  ↳ 32-bit integer. You must write an algorithm that runs in O(n) time and without using the division operation.
8 ```
9 Example 1:
10
11 Input: nums = [1,2,3,4]
12 Output: [24,12,8,6]

```

```

13 Example 2:
14
15 Input: nums = [-1,1,0,-3,3]
16 Output: [0,0,9,0,0]
17 ```
18
19 ```python
20 class Solution:
21     def productExceptSelf(self, nums: List[int]) -> List[int]:
22         n = len(nums)
23         ans = [0]*n
24
25         ans[0] = 1
26         for i in range(1, n):
27             ans[i] = ans[i-1]*nums[i-1]
28
29         R = 1
30         for i in reversed(range(n)):
31             ans[i] = ans[i] * R
32             R *= nums[i]
33
34         return ans
35 ```
36
37
38 ### 325. Maximum Size Subarray Sum Equals k https://leetcode.com/problems/maximum-size-subarray-sum-equals-k/
39 Given an integer array nums and an integer k, return the maximum length of a subarray that sums to k. If there
40 ↪ is not one, return 0 instead.
41 ```
42
43 Example 1:
44
45 Input: nums = [1,-1,5,-2,3], k = 3
46 Output: 4
47 Explanation: The subarray [1, -1, 5, -2] sums to 3 and is the longest.
48
49 Example 2:
50
51 Input: nums = [-2,-1,2,1], k = 1
52 Output: 2
53 Explanation: The subarray [-1, 2] sums to 1 and is the longest.
54 ```
55
56 ```python
57 class Solution:
58     def maxSubArrayLen(self, nums: List[int], k: int) -> int:
59         # presum 的题
60         hashmap = {0:-1}
61         pre_sum = 0
62         res = 0
63         for i, num in enumerate(nums):
64             pre_sum += num
65             # Check if all of the numbers seen so far sum to k. 非常有必要! 否则容易错!
66             if pre_sum == k:

```

```

64         res = i + 1
65         # If any subarray seen so far sums to k, then
66         # update the length of the longest_subarray.
67         if pre_sum - k in hashmap:
68             res = max(res, i - hashmap[pre_sum - k])
69         # Only add the current prefix_sum index pair to the
70         # map if the prefix_sum is not already in the map.
71         if pre_sum not in hashmap: # 这个判断有必要，因为可能正负都有，之前的题没有判断这个因为递增
72             hashmap[pre_sum] = hashmap.get(pre_sum, 0) + 1
73
74     return res
75
76     # time o(n), space o(n)
77     ...
78
79     """ 560. Subarray Sum Equals K
80     Given an array of integers nums and an integer k, return the total number of subarrays whose sum equals to k. A
81     ↪ subarray is a contiguous non-empty sequence of elements within an array.
82     """
83
84     Example 1:
85     Input: nums = [1,1,1], k = 2
86     Output: 2
87
88     Example 2:
89     Input: nums = [1,2,3], k = 3
90     Output: 2
91     """
92
93     """python
94     class Solution:
95         def subarraySum(self, nums: List[int], k: int) -> int:
96
97         #
98         ↪ https://leetcode-cn.com/problems/subarray-sum-equals-k/solution/qian-zhui-he-si-xiang-560-he-wei-kde-zi-shu-zu-by/
99
100         # 很高频的，前缀和的几种写法
101         # # way1 - brute force 超时, o(n^2), space o(1)
102         # res = 0
103         # for i in range(len(nums)):
104         #     for j in range(i, len(nums)):
105         #         if sum(nums[i:j+1]) == k:
106         #             res += 1
107         # return res
108
109         # prefix sum - 也是超时了 o(n^2), space o(n) 因为有 pre 的空间
110         # cnt, n = 0, len(nums)
111         # pre = [0] * (n + 1)
112         # for i in range(1, n + 1):
113         #     pre[i] = pre[i - 1] + nums[i - 1]

```

```

114 #         for i in range(1, n + 1):
115 #             for j in range(i, n + 1):
116 #                 if (pre[j] - pre[i - 1] == k): cnt += 1
117 #         return cnt
118
119 # hashmap + pre_sum 这个写法更适合我之前的习惯
120 pre, res = 0, 0
121 count = dict()
122 for num in nums:
123     pre += num
124     if pre == k: res += 1
125     res += count.get(pre-k, 0)
126     count[pre] = count.get(pre, 0) + 1 # 这个写法更好一些, 和 sliding window 一样
127
128 return res
129
130
131 #
132 # pre_sum = collections.defaultdict(int)
133 # res, cur_pre_sum = 0, 0
134 # for i in range(len(nums)):
135 #     cur_pre_sum += nums[i]
136 #     if cur_pre_sum - k in pre_sum:
137 #         res += pre_sum[cur_pre_sum]
138 #     pre_sum[cur_pre_sum] += 1
139 # return res
140
141 # # 记录 绿色 " 前缀和" (从 0 到 i 的前缀和) 的 值和出现的次数.
142 # pre_sum = collections.defaultdict(int)
143 # # 初始化 前缀和 为 0 的 子序列 出现了一次.
144 # # 对应 第一类情况, 上面的 if cur_pre_sum - k == 0 语句
145 # pre_sum[0] = 1
146 # # 记录 当前 位置的 前缀和
147 # cur_pre_sum = 0
148 # # 用于记录结果
149 # res = 0
150 # for i in range(len(nums)):
151 #     cur_pre_sum += nums[i] # 计算 当前位置的 前缀和
152 #     # cur_sum - k 是我们想找的前缀和 nums[0..j]
153 #     # 如果前面有这个前缀和, 则直接更新答案
154 #     green_sum = cur_pre_sum - k
155 #     if green_sum in pre_sum:
156 #         res += pre_sum[green_sum]
157 #     # 每次计算都将前缀和加入字典
158 #     pre_sum[cur_pre_sum] += 1
159 # return res
160
161
162 ### 974. Subarray Sums Divisible by K
163 Given an integer array nums and an integer k, return the number of non-empty subarrays that have a sum divisible
164 ↪ by k. A subarray is a contiguous part of an array.

```

```

165 Example 1:
166
167 Input: nums = [4,5,0,-2,-3,1], k = 5
168 Output: 7
169 Explanation: There are 7 subarrays with a sum divisible by k = 5:
170 [4, 5, 0, -2, -3, 1], [5], [5, 0], [5, 0, -2, -3], [0], [0, -2, -3], [-2, -3]
171 Example 2:
172
173 Input: nums = [5], k = 9
174 Output: 0
175 ```
176 ```python
177 class Solution:
178     def subarraysDivByK(self, nums: List[int], k: int) -> int:
179         # res, cnt = 0, Counter({0:1}) # 定义哨兵节点，取余结果为 0 时，默认已经出现一次
180         res, cnt = 0, {0:1} # 改成 {} 不能直接调用，会报错，如果之前没有出现
181         pre = list(accumulate(nums, add)) # 计算前缀和数组
182
183         for i in range(len(pre)): # 遍历每个前缀和元素
184             mod = (pre[i]+k) % k # 因为可能存在负数，所以加上一个 k，再计算对 k 取余结果
185             res += cnt[mod] # 加上哈希表中存储的 mod 对应的次数，更新可行方案数，会报错
186             cnt[mod] += 1 # 更新哈希表
187             print('mod',mod)
188             print('cnt',cnt)
189             print('res',res)
190
191         return res
192
193 # # 这个题挺好的，不需要做判断，还是同余的道理，
194 # pre_sum, res = 0, 0
195 # hashmap = {} # 这样初始化，然后把 prefix sum 求和的情况在 for loop 里面判断
196
197 # for i, num in enumerate(nums):
198 #     pre_sum += num
199 #     if pre_sum % k == 0:
200 #         res += 1
201 #     reminder = pre_sum % k
202 #     res += hashmap.get(reminder,0)
203 #     hashmap[reminder] = hashmap.get(reminder, 0) + 1
204
205 # # 很多时候会考虑到判断，但是少一个解？
206 # # if reminder in hashmap:
207 # #     res += hashmap[reminder]
208 # # else:
209 # #     hashmap[reminder] = hashmap.get(reminder, 0) + 1
210
211 # return res
212 ```
213
214 ### 1590. Make Sum Divisible by P

```



```

215 Given an array of positive integers nums, remove the smallest subarray (possibly empty) such that the sum of the
    ↳ remaining elements is divisible by p. It is not allowed to remove the whole array. Return the length of the
    ↳ smallest subarray that you need to remove, or -1 if it's impossible. A subarray is defined as a contiguous
    ↳ block of elements in the array.
216 ```
217 Example 1:
218
219 Input: nums = [3,1,4,2], p = 6
220 Output: 1
221 Explanation: The sum of the elements in nums is 10, which is not divisible by 6. We can remove the subarray [4],
    ↳ and the sum of the remaining elements is 6, which is divisible by 6.
222 Example 2:
223
224 Input: nums = [6,3,5,2], p = 9
225 Output: 2
226 Explanation: We cannot remove a single element to get a sum divisible by 9. The best way is to remove the
    ↳ subarray [5,2], leaving us with [6,3] with sum 9.
227 Example 3:
228
229 Input: nums = [1,2,3], p = 3
230 Output: 0
231 Explanation: Here the sum is 6. which is already divisible by 3. Thus we do not need to remove anything.
232 ```
233
234 ```python
235 class Solution:
236     def minSubarray(self, nums: List[int], p: int) -> int:
237
238         pre = list(accumulate(nums, add))
239         mod = pre[-1] % p
240         hashT = {0:-1}
241         if mod == 0: return 0
242
243         res = len(nums)
244         for i in range(len(nums)):
245             curmod = pre[i] % p
246             tarmod = (curmod - mod + p) % p
247             if tarmod in hashT:
248                 dis = i - hashT[tarmod]
249                 res = dis if dis < res else res
250                 if res == 1 and len(nums) != 1:
251                     return 1
252             hashT[curmod] = i
253         if res == len(nums):
254             res = -1
255         return res
256 ```

```

14 Array/String

15 Math

```
1 # Math 相关题型
2
3
4 ## Randomized 随机操作
5
6 这个题以及相关题目有很多可以考的
7 1. 2个 uniform distribution 求和不是 uniform, 因为有很多重复的可能性, 比如 rand2() + rand2() 有多种可能性得到 3, 这个问题也可
   ↳ 以用数学公式推导, 他的分布是三角分布
8 2. rand_x() 生成 [1,x] 那么 (rand_x - 1) * Y + rand_y() 可以生成 rand_{xy} [1, xy] 的随机数 - 这个定理很重要
9 3. rejected sampling, 生成之后可以用 mod 操作来拒绝掉不想要的部分, 但注意效率, 可以迭代几次来提高效率
10 4. 扔骰子问题, 或者抛硬币
   ↳ https://leetcode.cn/problems/implement-rand10-using-rand7/solution/wei-rao-li-lun-yi-ge-bu-jun-yun-ying-bi-fo4ei/
11 5. 只有 2 个 Gaussian 求和是 Gaussian. 2个 Uniform 的加减乘除都不是 uniform
12 6. 2个均匀分布 求和是三角分布, 这个知乎讲得挺好的! https://www.zhihu.com/question/27060339
13
14
15 ### 470. Implement Rand10() Using Rand7() https://leetcode.com/problems/implement-rand10-using-rand7/
16
17 Given the API rand7() that generates a uniform random integer in the range [1, 7], write a function rand10()
   ↳ that generates a uniform random integer in the range [1, 10]. You can only call the API rand7(), and you
   ↳ shouldn't call any other API. Please do not use a language's built-in random API.
18
19 Each test case will have one internal argument n, the number of times that your implemented function rand10()
   ↳ will be called while testing. Note that this is not an argument passed to rand10().
20
21 Example 1:
22 ```
23 Input: n = 1
24 Output: [2]
25 ```
26 Example 2:
27 ```
28 Input: n = 2
29 Output: [2,8]
30 ```
31 Example 3:
32 ```
33 Input: n = 3
34 Output: [3,8,10]
35 ```
36 Constraints: 1 <= n <= 105
37 Follow up: What is the expected value for the number of calls to rand7() function? Could you minimize the number
   ↳ of calls to rand7()?
38
39
40 ```python
41 class Solution:
```

```

42     def rand10(self):
43         """
44         :rtype: int
45         """
46         while True:
47             base = rand7() - 1
48             new = (rand7() - 1) * 7 + rand7() # 49
49             if new <= 40:
50                 return new % 10 + 1
51
52
53 class Solution: # 解决 follow-up 的 question!
54     def rand10(self) -> int:
55         while True:
56             a = rand7()
57             b = rand7()
58             idx = (a - 1) * 7 + b
59             if idx <= 40:
60                 return 1 + (idx - 1) % 10 # 拒绝了 41-49 9 个数
61             a = idx - 40
62             b = rand7()
63             # get uniform dist from 1 - 63 - 拒绝 61,62,63 三个数
64             idx = (a - 1) * 7 + b
65             if idx <= 60:
66                 return 1 + (idx - 1) % 10
67             a = idx - 60
68             b = rand7()
69             # get uniform dist from 1 - 21 - 这样每次只拒绝 21 一个数
70             idx = (a - 1) * 7 + b
71             if idx <= 20:
72                 return 1 + (idx - 1) % 10
73
74
75
76
77 ## 进制理解 and 操作 - digit
78 基本操作不熟悉!
79 ```python
80 # 就是对进位不了解或者不熟悉
81 num = 123
82
83 print(num // 10) # 最高位 + 次高位
84 print(num // 100) # 最高位
85 print(num // 1000) # 0
86
87 print(num % 10) # 个位
88 print(num % 100) # 十位 + 个位
89 print(num % 1000) # 百分位 + 十位 + 个位
90
91 # output
92 # 12
93 # 1

```

```

94 # 0
95 # 3
96 # 23
97 # 123
98 ````
99
100 ### 400. Nth Digit https://leetcode.com/problems/nth-digit/
101
102 Given an integer n, return the nth digit of the infinite integer sequence [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
↪ ...].
103
104 Example 1:
105 ````
106 Input: n = 3
107 Output: 3
108 ````
109 Example 2:
110 ````
111 Input: n = 11
112 Output: 0
113 Explanation: The 11th digit of the sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ... is a 0, which is part of the
↪ number 10.
114 ````
115
116 ``` python
117 class Solution:
118     def totalDigits(self, length: int) -> int:
119         digits = 0
120         curCount = 9
121         for curLength in range(1, length + 1):
122             digits += curLength * curCount
123             curCount *= 10
124         return digits
125
126     def findNthDigit(self, n: int) -> int:
127         low, high = 1, 9
128         while low < high:
129             mid = (low + high) // 2
130             if self.totalDigits(mid) < n:
131                 low = mid + 1
132             else:
133                 high = mid
134         d = low # 返回属于的位数
135
136         prevDigits = self.totalDigits(d - 1) # 之前的所有数 digit 求和
137         index = n - prevDigits - 1 # 当前位数的 index, 比如 2 位数中排序
138         start = 10 ** (d - 1) # 之前有多少个数, 绝对 number, 比, 1, 2, 3, ..., 100
139         num = start + index // d # n 对应的当前的数
140         digitIndex = index % d # 判断是 d 位数中第几个, 百分位还是十分位, 个位
141         print(d, prevDigits, index, start, num, digitIndex)
142         print(num // 10 ** (d - digitIndex - 1)) # 这是关键, 返回一个 digit, 都要%10
143         return num // 10 ** (d - digitIndex - 1) % 10

```

```

144
145     # example 121
146     # print 2 9 111 10 65 1
147     # print 65
148
149 class Solution:
150     def findNthDigit(self, n: int) -> int:
151         cur, base = 1, 9
152         while n > cur * base:
153             n -= cur * base
154             cur += 1
155             base *= 10
156         print(cur,n)
157         n -= 1
158         # 数字
159         num = 10 ** (cur - 1) + n // cur
160         print(num)
161         # 数字里的第几位
162         idx = n % cur
163         print(idx)
164         return num // (10 ** (cur - 1 - idx)) % 10
165     ...

```

16 Special topics

16.0.1 Subarray

很多 prefix sum 还有 dp

16.0.2 Substring

多数是 sliding window

16.0.3 Subsequence

17 Probability and Statistics

17.1 Probability

17.1.0.1 Sum of two uniform distribution ? 这个帖子讲了 2 个 $u(0,1)$ 的加减乘除的方法, 只有 Gaussian 求和是 Gaussian 其他基本都不是.

参考文献