Leetcode(python) 队列中等题

621任务调度器

题目：给定一个用字符数组表示的 CPU 需要执行的任务列表。其中包含使用大写的 A - Z 字母表示的26 种不同种类的任务。任务可以以任意顺序执行，并且每个任务都可以在 1 个单位时间内执行完。CPU 在任何一个单位时间内都可以执行一个任务，或者在待命状态。

然而，两个**相同种类**的任务之间必须有长度为**n** 的冷却时间，因此至少有连续 n 个单位时间内 CPU 在执行不同的任务，或者在待命状态。

你需要计算完成所有任务所需要的**最短时间**。

**示例 1：**

**输入:** tasks = ["A","A","A","B","B","B"], n = 2

**输出:** 8

**执行顺序:** A -> B -> (待命) -> A -> B -> (待命) -> A -> B.

**注：**

1. 任务的总个数为 [1, 10000]。
2. n 的取值范围为 [0, 100]。

思路：只要知道出现最多的那个（或几个）task就行了，其他的任务往缝隙里面塞。

比如：

如给定：AAABBCD，n=2。那么我们满足个数最多的任务所需的数量，即可以满足任务间隔要求，即：AXXAXXA；（其中，X表示需要填充任务或者idle的间隔）

如果有两种或两种以上的任务具有相同的最多的任务数，如：AAAABBBBCCDE，n=3。那么我们将具有相同个数的任务A和B视为一个任务对，最终满足要求的分配为：ABXXABXXABXXAB，剩余的任务在不违背要求间隔的情况下穿插进间隔位置即可，空缺位置补idle。

由上面的分析我们可以得到最终需要最少的任务时间：（最多任务数-1）\*（n + 1） + （相同最多任务的任务个数）。

有上面的例子来说就是：(num(A)-1) \* (3+1) + (2)。

其中，（最多任务数-1）\*（n + 1）代表的是ABXXABXXABXX，（相同最多任务的任务个数）代表的是最后的AB.

代码：

1. **class** Solution(object):
2. **def** leastInterval(self, tasks, n):
3. """
4. :type tasks: List[str]
5. :type n: int
6. :rtype: int
7. """
8. count=collections.Counter(tasks)
9. most=count.most\_common()[0][1]
10. num\_most=len([i **for** i, v **in** count.items() **if** v==most])
11. time=(most-1)\*(n+1)+num\_most
12. **return** max(time,len(tasks))

641设计循环双端队列

题目：

设计实现双端队列。  
你的实现需要支持以下操作：

* MyCircularDeque(k)：构造函数,双端队列的大小为k。
* insertFront()：将一个元素添加到双端队列头部。 如果操作成功返回 true。
* insertLast()：将一个元素添加到双端队列尾部。如果操作成功返回 true。
* deleteFront()：从双端队列头部删除一个元素。 如果操作成功返回 true。
* deleteLast()：从双端队列尾部删除一个元素。如果操作成功返回 true。
* getFront()：从双端队列头部获得一个元素。如果双端队列为空，返回 -1。
* getRear()：获得双端队列的最后一个元素。 如果双端队列为空，返回 -1。
* isEmpty()：检查双端队列是否为空。
* isFull()：检查双端队列是否满了。

**示例：**

MyCircularDeque circularDeque = new MycircularDeque(3); // 设置容量大小为3

circularDeque.insertLast(1); // 返回 true

circularDeque.insertLast(2); // 返回 true

circularDeque.insertFront(3); // 返回 true

circularDeque.insertFront(4); // 已经满了，返回 false

circularDeque.getRear(); // 返回 2

circularDeque.isFull(); // 返回 true

circularDeque.deleteLast(); // 返回 true

circularDeque.insertFront(4); // 返回 true

circularDeque.getFront(); // 返回 4

**提示：**

* 所有值的范围为 [1, 1000]
* 操作次数的范围为 [1, 1000]
* 请不要使用内置的双端队列库。

思路：

代码：

1. **class** MyCircularDeque(object):
3. **def** \_\_init\_\_(self, k):
4. """
5. Initialize your data structure here. Set the size of the deque to be k.
6. :type k: int
7. """
8. self.queue=[]
9. self.size=k
11. **def** insertFront(self, value):
12. """
13. Adds an item at the front of Deque. Return true if the operation is successful.
14. :type value: int
15. :rtype: bool
16. """
17. **if** **not** self.isFull():
18. self.queue.insert(0,value)
19. **return** True
20. **else**:
21. **return** False
23. **def** insertLast(self, value):
24. """
25. Adds an item at the rear of Deque. Return true if the operation is successful.
26. :type value: int
27. :rtype: bool
28. """
29. **if** **not** self.isFull():
30. self.queue.append(value)
31. **return** True
32. **else**:
33. **return** False
35. **def** deleteFront(self):
36. """
37. Deletes an item from the front of Deque. Return true if the operation is successful.
38. :rtype: bool
39. """
40. **if** **not** self.isEmpty():
41. self.queue.pop(0)
42. **return** True
43. **else**:
44. **return** False
46. **def** deleteLast(self):
47. """
48. Deletes an item from the rear of Deque. Return true if the operation is successful.
49. :rtype: bool
50. """
51. **if** **not** self.isEmpty():
52. self.queue.pop()
53. **return** True
54. **else**:
55. **return** False
57. **def** getFront(self):
58. """
59. Get the front item from the deque.
60. :rtype: int
61. """
62. **if** self.isEmpty():
63. **return** -1
64. **else**:
65. **return** self.queue[0]
67. **def** getRear(self):
68. """
69. Get the last item from the deque.
70. :rtype: int
71. """
72. **if** self.isEmpty():
73. **return** -1
74. **else**:
75. **return** self.queue[-1]
77. **def** isEmpty(self):
78. """
79. Checks whether the circular deque is empty or not.
80. :rtype: bool
81. """
82. **return** len(self.queue)==0
84. **def** isFull(self):
85. """
86. Checks whether the circular deque is full or not.
87. :rtype: bool
88. """
89. **return** self.size==len(self.queue)

92. # Your MyCircularDeque object will be instantiated and called as such:
93. # obj = MyCircularDeque(k)
94. # param\_1 = obj.insertFront(value)
95. # param\_2 = obj.insertLast(value)
96. # param\_3 = obj.deleteFront()
97. # param\_4 = obj.deleteLast()
98. # param\_5 = obj.getFront()
99. # param\_6 = obj.getRear()
100. # param\_7 = obj.isEmpty()
101. # param\_8 = obj.isFull()