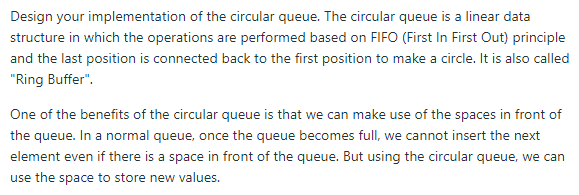
流程：确认即“面试官你看有问题吗，没问题我们就到下一步”

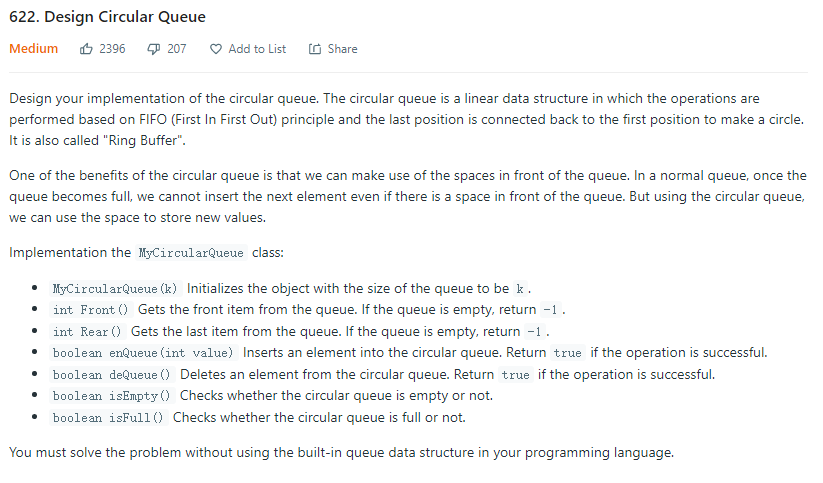
1. first ask for clarification，确认产品理解的细节



又比如“Follow up是如果用户想要删除时间区间A到B之间的浏览记录该如何实现”

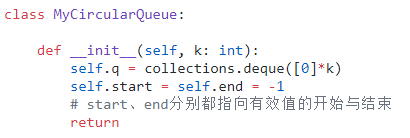
1. A、B什么类型？timestamp；2. 前端能保证输入有效，比如A < B？可以；3. 实际生活中用户不会精确到秒，颗粒度怎样？15min一个间隔。

1. 确认解决问题的数据结构与函数签名要求



1. 确认内部成员与方法的时空间复杂度
2. 确认实现成员使用的内置数据类型、初始化值

上图中最后一行，可以使用列表、队列等，不过指定了用队列



[Design Circular Queue](https://leetcode.com/problems/design-circular-queue/)

下面这道题，我有两个犹豫的点：

1. 没有起始确认start、end的初始值，默认写0导致之后写着发现不对
2. deQueue函数中，队列里没对象的等价条件 并不是 (start+1) %k>end或者(start+1) >end

class MyCircularQueue:  
  
 def \_\_init\_\_(self, k: int):  
 self.q = collections.deque([0] \* k)  
 self.start = self.end = -1  
 # start、end分别都指向有效值的开始与结束  
 return  
  
 def enQueue(self, value: int) -> bool:  
 if self.isFull():  
 return False  
  
 self.end = (self.end + 1) % len(self.q)  
 self.q[self.end] = value  
 if self.start == -1: self.start = 0  
 return True  
  
 def deQueue(self) -> bool:  
 if self.isEmpty():  
 return False  
 if self.start == self.end:  
 self.start = self.end = -1  
 else:  
 self.start = (self.start + 1) % len(self.q)  
 return True  
  
 def Front(self) -> int:  
 if self.start == -1:  
 return -1  
 return self.q[self.start]  
  
 def Rear(self) -> int:  
 if self.start == -1:  
 return -1  
 return self.q[self.end]  
  
 def isEmpty(self) -> bool:  
 return self.start == -1  
  
 def isFull(self) -> bool:  
 return (self.end + 1) % len(self.q) == self.start