621. Task Scheduler
Medium

Given a characters array tasks, representing the tasks a CPU needs to do, where each letter represents a different task. Tasks could be done in any order. Each task is done in one unit of time. For each unit of time, the CPU could complete either one task or just be idle.

However, there is a non-negative integer n that represents the cooldown period between two same tasks (the same letter in the array), that is that there must be at least n units of time between any two same tasks.

Return the least number of units of times that the CPU will take to finish all the given tasks.

Input: tasks = ["A","A","A","A","A","A","B","C","D","E","F","G"], n = 2
Output: 16

```python
class Solution:
    def leastInterval(self, tasks: List[str], n: int) -> int:
        count = Counter(tasks)
        maxHeap = [-cnt for cnt in count.values()]
        heapq.heapify(maxHeap)
        time = 0

        q = deque() # pair of values, [-cnt, idleTime]

        while maxHeap or q:
            time += 1
            if maxHeap:
                cnt = 1 + heapq.heappop(maxHeap)
                if cnt:
                    q.append([cnt, time + n])
            if q and q[0][1] == time:
                heapq.heappush(maxHeap, q.popleft()[0])

        return time
```

n=2

```
count = {
    "A": 6,
    "B": 1,
    "C": 1,
    "D": 1,
    "E": 1,
    "F": 1,
    "G": 1
    }
```

maxHeap = [-6, -1, -1, -1, -1, -1, -1]

q = []

```
if cnt:
    q.append([cnt, time + n])
```

time	maxHeap	q
1	[-1, -1, -1, -1, -1, -1]	[[-5, 3]]
2	[-1, -1, -1, -1, -1]	[[-5, 3]]
3	[-1, -1, -1, -1]	[[-5, 3]]
`if q and q[0][1] == time: heapq.heappush(maxHeap, q.popleft()[0])` →	[-1, -1, -1, -5] / [-5, -1, -1, -1]	[]
4	[-1, -1, -1]	[[-4, 6]]
5	[-1, -1]	[[-4, 6]]
6	[-1]	[[-4, 6]]
`if q and q[0][1] == time: heapq.heappush(maxHeap, q.popleft()[0])` →	[-1, -4] / [-4, -1]	[]
7	[-1]	[[-3, 9]]
8	[]	[[-3, 9]]
9	[]	[[-3, 9]]
`if q and q[0][1] == time: heapq.heappush(maxHeap, q.popleft()[0])` →	[-3]	[]
10	[]	[[-2, 12]]
11	[]	[[-2, 12]]
12	[]	[[-2, 12]]
`if q and q[0][1] == time: heapq.heappush(maxHeap, q.popleft()[0])` →	[-2]	[]
13	[]	[[-1, 15]]
14	[]	[[-1, 15]]
15	[]	[[-1, 15]]
`if q and q[0][1] == time: heapq.heappush(maxHeap, q.popleft()[0])` →	[-1]	[]
16	[]	[]

= q[0][1]

=0 -1

```
cnt = 1 + heapq.heappop(maxHeap)
if cnt:X
    q.append([cnt, time + n])
```

nothing in q