2182. Construct String With Repeat Limit

Medium | Topics | Companies | Hint

You are given a string `s` and an integer `repeatLimit`. Construct a new string `repeatLimitedString` using the characters of `s` such that no letter appears **more than** `repeatLimit` times **in a row**. You do **not** have to use all characters from `s`.

Return *the **lexicographically largest** `repeatLimitedString` possible*.

A string `a` is **lexicographically larger** than a string `b` if in the first position where `a` and `b` differ, string `a` has a letter that appears later in the alphabet than the corresponding letter in `b`. If the first `min(a.length, b.length)` characters do not differ, then the longer string is the lexicographically larger one.

Example 1:

```
Input: s = "cczazcc", repeatLimit = 3
Output: "zzcccac"
Explanation: We use all of the characters from s to construct the
repeatLimitedString "zzcccac".
The letter 'a' appears at most 1 time in a row.
The letter 'c' appears at most 3 times in a row.
The letter 'z' appears at most 2 times in a row.
Hence, no letter appears more than repeatLimit times in a row and the string
is a valid repeatLimitedString.
The string is the lexicographically largest repeatLimitedString possible so
we return "zzcccac".
Note that the string "zzcccca" is lexicographically larger but the letter
'c' appears more than 3 times in a row, so it is not a valid
repeatLimitedString.
```

count = Counter(s)

[(ord(c), cnt for c, cnt in count.items())]

↓ max-heap

count = [0] x 26

for c in s:

count[ord(c) - ord('a')]

+= 1

res = ''

while count:

:

```python
class Solution:
    def repeatLimitedString(self, s: str, repeatLimit: int) -> str:
        count = Counter(s)
        mh = [(-ord(c), cnt) for c, cnt in count.items()]
        heapq.heapify(mh)

        res = []
        while mh:
            c, cnt = heapq.heappop(mh)
            char = chr(-c)
            curr_cnt = min(cnt, repeatLimit)
            res.append(char * curr_cnt)

            if cnt > repeatLimit and mh:
                next_char, next_cnt = heapq.heappop(mh)
                next_curr_c = chr(-next_char)
                res.append(next_curr_c)

                if next_cnt > 1:
                    heapq.heappush(mh, (next_char, next_cnt - 1))

                heapq.heappush(mh, (c, cnt - curr_cnt))

        return ''.join(res)
```

} use max_heap

push the second largest back if there's remaining

else

the remaining will not push back, cuz there's no more second largest char and can't exceed repeatLimit.