按键持续时间最长的键

LeetCode 设计了一款新式键盘，正在测试其可用性。测试人员将会点击一系列键（总计 n 个），每次一个。

给你一个长度为 n 的字符串 keysPressed ，其中 keysPressed[i] 表示测试序列中第 i 个被按下的键。releaseTimes 是一个升序排列的列表，其中 releaseTimes[i] 表示松开第 i 个键的时间。字符串和数组的 **下标都从 0 开始** 。第 0 个键在时间为 0 时被按下，接下来每个键都 **恰好** 在前一个键松开时被按下。

测试人员想要找出按键 **持续时间最长** 的键。第 i 次按键的持续时间为 releaseTimes[i] - releaseTimes[i - 1] ，第 0 次按键的持续时间为 releaseTimes[0] 。

注意，测试期间，同一个键可以在不同时刻被多次按下，而每次的持续时间都可能不同。

请返回按键 **持续时间最长** 的键，如果有多个这样的键，则返回 **按字母顺序排列最大** 的那个键。

示例 1：

输入： releaseTimes = [9,29,49,50], keysPressed = "cbcd"

输出： "c"

解释： 按键顺序和持续时间如下：

按下 'c' ，持续时间 9（时间 0 按下，时间 9 松开）

按下 'b' ，持续时间 29 - 9 = 20（松开上一个键的时间 9 按下，时间 29 松开）

按下 'c' ，持续时间 49 - 29 = 20（松开上一个键的时间 29 按下，时间 49 松开）

按下 'd' ，持续时间 50 - 49 = 1（松开上一个键的时间 49 按下，时间 50 松开）

按键持续时间最长的键是 'b' 和 'c'（第二次按下时），持续时间都是 20

'c' 按字母顺序排列比 'b' 大，所以答案是 'c'

示例 2：

输入： releaseTimes = [12,23,36,46,62], keysPressed = "spuda"

输出： "a"

解释： 按键顺序和持续时间如下：

按下 's' ，持续时间 12

按下 'p' ，持续时间 23 - 12 = 11

按下 'u' ，持续时间 36 - 23 = 13

按下 'd' ，持续时间 46 - 36 = 10

按下 'a' ，持续时间 62 - 46 = 16

按键持续时间最长的键是 'a' ，持续时间 16

```
class Solution:
    def slowestKey(self, releaseTimes: List[int], keysPressed: str) -> str:
        #标记键序号
        flagkey=0
        #标记最大时间
        maxtime=releaseTimes[0]
        for i in range(1,len(releaseTimes)):
            #如果时间差有更大值 则更新 maxtime 和对应序号 flagkey
            #如果时间差相等 则按照字母排序更新 flagkey
            if releaseTimes[i]-releaseTimes[i-
1]>maxtime or releaseTimes[i]-releaseTimes[i-
1]==maxtime and keysPressed[i]>keysPressed[flagkey]:
                flagkey=i
                maxtime=releaseTimes[i]-releaseTimes[i-1]
        return keysPressed[flagkey]
```