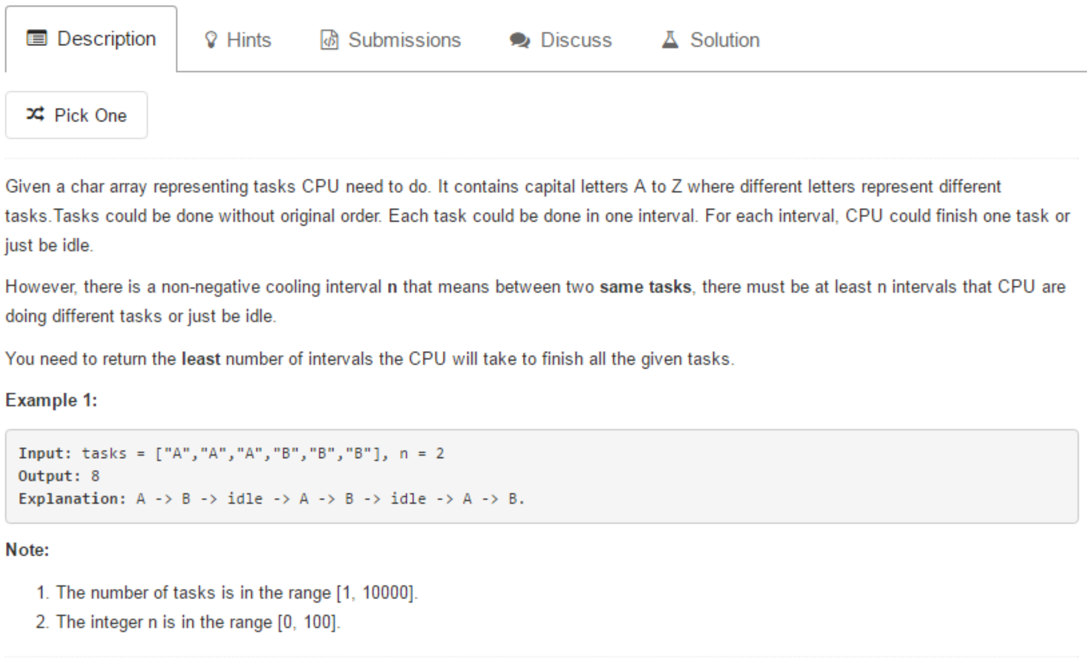
621. Task Scheduler



Given a char array representing tasks CPU need to do. It contains capital letters A to Z where different letters represent different tasks.Tasks could be done without original order. Each task could be done in one interval. For each interval, CPU could finish one task or just be idle.

However, there is a non-negative cooling interval **n** that means between two **same tasks**, there must be at least n intervals that CPU are doing different tasks or just be idle.

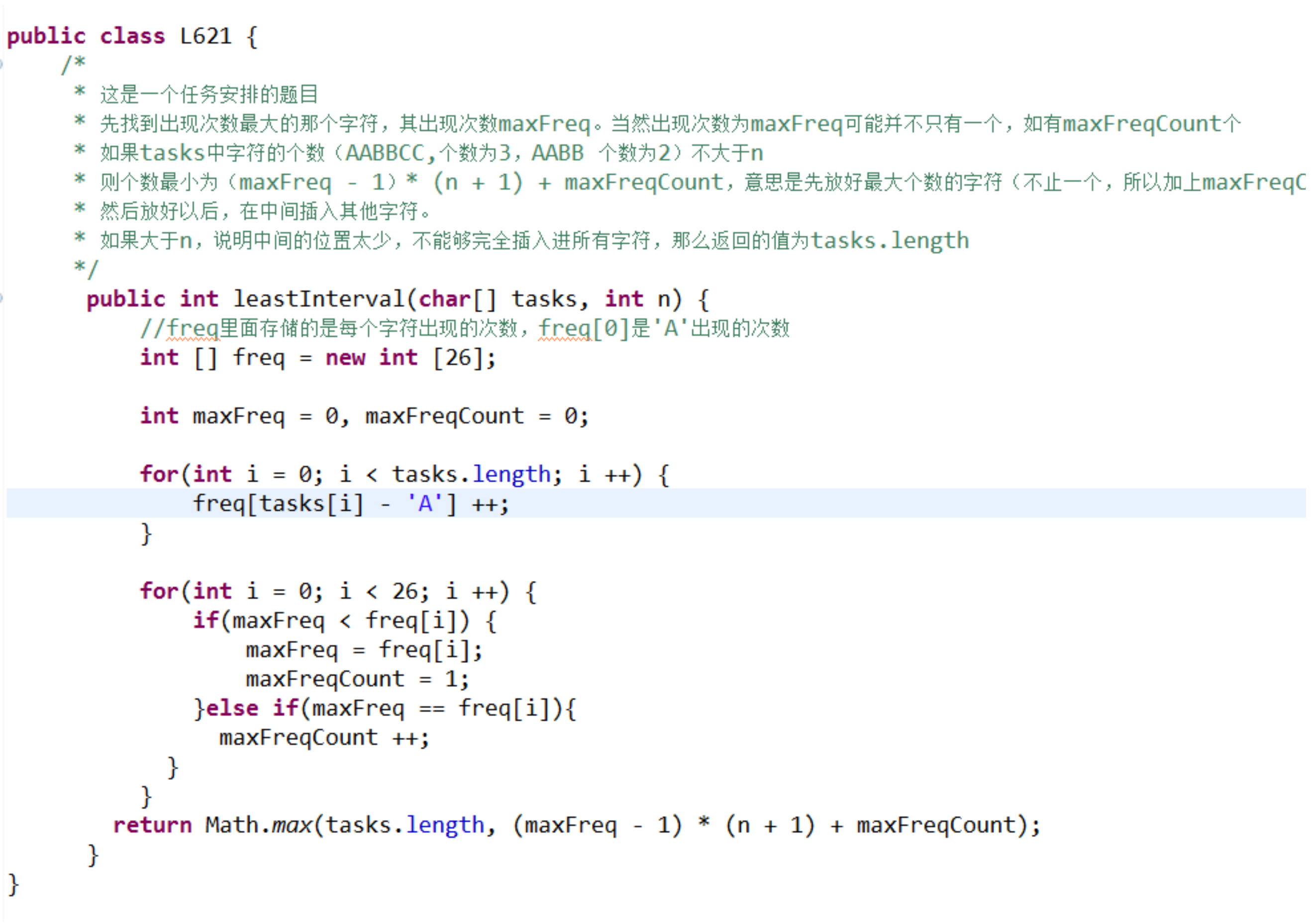
You need to return the **least** number of intervals the CPU will take to finish all the given tasks.

Example 1:

```
Input: tasks = ["A","A","A","B","B","B"], n = 2
Output: 8
Explanation: A -> B -> idle -> A -> B -> idle -> A -> B.
```

Note:

1. The number of tasks is in the range [1, 10000].
2. The integer n is in the range [0, 100].



```
public class L621 {
    /*
     * 这是一个任务安排的题目
     * 先找到出现次数最大的那个字符，其出现次数maxFreq。当然出现次数为maxFreq可能并不只有一个，如有maxFreqCount个
     * 如果tasks中字符的个数（AABBCC,个数为3，AABB 个数为2）不大于n
     * 则个数最小为（maxFreq - 1）* (n + 1) + maxFreqCount，意思是先放好最大个数的字符（不止一个，所以加上maxFreqC
     * 然后放好以后，在中间插入其他字符。
     * 如果大于n，说明中间的位置太少，不能够完全插入进所有字符，那么返回的值为tasks.length
     */
     public int leastInterval(char[] tasks, int n) {
         //freq里面存储的是每个字符出现的次数，freq[0]是'A'出现的次数
         int [] freq = new int [26];

         int maxFreq = 0, maxFreqCount = 0;

         for(int i = 0; i < tasks.length; i ++) {
             freq[tasks[i] - 'A'] ++;
         }

         for(int i = 0; i < 26; i ++) {
             if(maxFreq < freq[i]) {
                 maxFreq = freq[i];
                 maxFreqCount = 1;
             }else if(maxFreq == freq[i]){
               maxFreqCount ++;
           }
         }
       return Math.max(tasks.length, (maxFreq - 1) * (n + 1) + maxFreqCount);
     }
}
```