CSE 201: Data Structures

LAB 05/12/2024

Assume you are given a character array representing a list of tasks a CPU needs. Each character represents a unique task. The order of the tasks is not important. Each task is done in 1 unit of time. The CPU can either complete a task or stay idle for each time unit.

However, a non-negative integer n represents the cooldown period between two tasks; that is, there must be at least n units of time between any two tasks.

Return the least number of units of time that 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
There is at least 2 units of time between any two same tasks.

Example 2:

Input: tasks = ["A","A","A","B","B","B"], n = 0
Output: 6
Explanation: On this case any permutation of size 6 would work since n = 0.
["A","A","A","B","B","B"]
["A","B","A","B","A","B"]
["B","B","B","A","A","A"]
...
And so on.
Example 3:

Input: tasks = ["A","A","A","A","A","A","B","C","D","E","F","G"], n = 2
Output: 16
Explanation:
One possible solution is
A -> B -> C -> A -> D -> E -> A -> F -> G -> A -> idle -> idle -> A -> idle -> idle -> A

Code template

```
Class Solution{
      public int leastInterval(char[] tasks, int n) {
       //first need to map the number of times each task required to
be assigned
       //store the total time taken
       //n+1 is the CPU cycle length if n is the cool down
       //the task at the top should be assigned first
        //the task with more than one occurrence, they'll come up at
the next cycle
     //add it to the remaining task list
     //if the priority queue is empty, then all the tasks are
completed
     }

}
```