[**Task Scheduler**](https://leetcode.com/problems/task-scheduler/)

You are given an array of CPU tasks, each represented by letters A to Z, and a cooling time, n. Each cycle or interval allows the completion of one task. Tasks can be completed in any order, but there's a constraint: identical tasks must be separated by at least n intervals due to cooling time.​Return the minimum number of intervals required to complete all tasks.]Example 1:Input: tasks = ["A","A","A","B","B","B"], n = 2Output: 8Explanation: A possible sequence is: A -> B -> idle -> A -> B -> idle -> A -> B.After completing task A, you must wait two cycles before doing A again. The same applies to task B. In the 3rd interval, neither A nor B can be done, so you idle. By the 4th cycle, you can do A again as 2 intervals have passed.Example 2:Input: tasks = ["A","C","A","B","D","B"], n = 1Output: 6Explanation: A possible sequence is: A -> B -> C -> D -> A -> B.With a cooling interval of 1, you can repeat a task after just one other task.

Constraints:1 <= tasks.length <= 104tasks[i] is an uppercase English letter.0 <= n <= 100

Solution:

class Solution {

public:

    int cnt[26], maxcnt = 0, e = 0;

    int leastInterval(vector<char>& tasks, int n) {

        for (char c : tasks) cnt[c-'A']++;

        for (int i = 0; i < 26; i++) maxcnt = max(maxcnt, cnt[i]);

        for (int i = 0; i < 26; i++)

            if (cnt[i] == maxcnt) e++;

        return max(int(tasks.size()), (maxcnt-1)\*(n+1) + e);

    }

};