任务最优调度

题目描述：

给定一个正整型数组表示待系统执行的任务列表，数组的每一个元素代表一个任务，元素的值表示该任务的类型。请计算执行完所有任务所需的最短时间。任务执行规则如下：

1、任务可以按任意顺序执行，且每个任务执行耗时间均为 1 个时间单位。

2、两个同类型的任务之间必须有长度为 N 个单位的冷却时间，比如：N 为 2 时，在时间 K 执行了类型 3 的任务，那么 K+1 和 K+2 两个时间不能执行类型 3 任务。

3、系统在任何一个单位时间内都可以执行一个任务，或者等待状态。

说明：数组最大长度为 1000,数组最大值 1000.

输入描述：

第一行记录一个用半角逗号分隔的数组，数组长度不超过 1000，数组元素的值不超过 1000

第二行记录任务冷却时间，N 为正整数，N<=100。

输出描述：

输出为执行完所有任务所需的最短时间。

补充说明

示例 1

输入：

```
2,2,2,3
2
```

输出：

```
7
```

说明：

时间 1：执行类型 2 任务。
时间 2：执行类型 3 的任务（因为冷却时间为 2，所以时间 2 不能执行类型 2 的任务）。
时间 3：系统等待（仍然在类型 2 的冷却时间）。
时间 4：执行类型 2 任务。
时间 5：系统等待。
时间 6：系统等待。
时间 7：执行类型 2 任务。
因此总共耗时 7。

```
#include<bits/stdc++.h>
#include<iostream>
using   namespace std;
bool camp(pair<int,int>a,pair<int,int>b){
      return a.first>b.first;
}
```

```
int main(){
    vector<int> arr;
    string   s;
    cin>>s;
    s+=',';
    int M;
    cin>>M;
    int num=0;
    for(auto j:s){
        if(j==',')
            arr.push_back(num),num=0;
        else
            num=num*10+j-'0';
    }
    map<int,int> cnt;
    for(int t:arr)
        cnt[t]++;
    vector<pair<int,int>>tasks;
    for(auto &kv:cnt)

        tasks.push_back(make_pair(kv.second,0));
    sort(tasks.begin(),tasks.end(),camp);
    int total=arr.size();
    int ans=0;
    while(total>0){
        ans++;
        bool flag=true;
        for(auto &t:tasks){
            if(flag && t.first > 0 && t.second==0){
                flag=false;
                t.first--;
                total--;
                t.second=M;
            }
            else if(t.second>0){
                t.second--;
            }
        }


    }
    cout<<ans<<endl;
```

}