Rabbits in Forest

In a forest, each rabbit has some color. Some subset of rabbits (possibly all of them) tell you how many other rabbits have the same color as them. Those `answers` are placed in an array.

Return the minimum number of rabbits that could be in the forest.

```
Examples:
Input: answers = [1, 1, 2]
Output: 5
Explanation:
The two rabbits that answered "1" could both be the same color, say red.
The rabbit than answered "2" can't be red or the answers would be inconsistent.
Say the rabbit that answered "2" was blue.
Then there should be 2 other blue rabbits in the forest that didn't answer into the a
rray.
The smallest possible number of rabbits in the forest is therefore 5: 3 that answered
 plus 2 that didn't.

Input: answers = [10, 10, 10]
Output: 11

Input: answers = []
Output: 0
```

Note:

1. `answers` will have length at most `1000`.
2. Each `answers[i]` will be an integer in the range `[0, 999]`.

Solution 1

```
class Solution {
    public int numRabbits(int[] answers) {
        int res = 0;
        Map<Integer,Integer> map = new HashMap<>();
        for(int answer : answers){
            map.put(answer,map.getOrDefault(answer,0)+1);
        }
        for(Integer n : map.keySet()){
            int group = map.get(n)/(n+1);
            res += map.get(n)%(n+1) != 0 ? (group+1)*(n+1) : group*(n+1);
        }
        return res;
    }
}
```

written by bzdjsm original link here

Solution 2

If `x+1` rabbits have same color, then we get `x+1` rabbits who all answer `x`.
now n rabbits answer x.
If `n%(x+1)==0`, we need `n/(x+1)` groups of `x+1` rabbits.
If `n%(x+1)!=0`, we need `n/(x+1) + 1` groups of `x+1` rabbits.

the number of groups is `math.ceil(n/(x+1))` and it equals to `(n+i)/(i+1)`, which is more elegant.

C++:

```
int numRabbits(vector<int>& answers) {
        map<int, int> c;
        for (int i : answers) c[i]++;
        int res = 0;
        for (auto i : c) res += (i.second + i.first) / (i.first + 1) * (i.first + 1
);
        return res;
    }
```

Java

```
public int numRabbits(int[] answers) {
        Map<Integer, Integer> m = new HashMap<>();
        for (int i : answers) m.put(i, m.getOrDefault(i, 0) + 1);
        int res = 0;
        for (int i : m.keySet()) res += (m.get(i) + i) / (i + 1) * (i + 1);
        return res;
    }
```

Python

```
def numRabbits(self, answers):
        c = collections.Counter(answers)
        return sum((c[i] + i) / (i + 1) * (i + 1) for i in c)
```

written by lee215 original link here

Solution 3

```
public int numRabbits(int[] answers) {
    HashMap<Integer, Double> map = new HashMap();
    for (int num : answers) {
        map.put(num, map.getOrDefault(num, 0.0) + 1);
    }
    int res = 0;
    for (int key : map.keySet()) {
       res += (key + 1) * Math.ceil(map.get(key) / (key + 1));
    }
    return res;
}
```

written by coolth original link here