Longest Harmonious Subsequence

We define a harmonious array is an array where the difference between its maximum value and its minimum value is **exactly** 1.

Now, given an integer array, you need to find the length of its longest harmonious subsequence among all its possible subsequences.

Example 1:

```
Input: [1,3,2,2,5,2,3,7]
Output: 5
Explanation: The longest harmonious subsequence is [3,2,2,2,3].
```

Note: The length of the input array will not exceed 20,000.

Solution 1

- The idea is to keep a count of all the numbers, and eventually for each of the numbers, check if there's any adjacent number. If it's present, then add the count of both - since these two numbers form subsequence in the array.

Update : from @harkness comment, we don't need to check both +1 and -1;

```java
public int findLHS(int[] nums) {
    Map<Long, Integer> map = new HashMap<>();
    for (long num : nums) {
        map.put(num, map.getOrDefault(num, 0) + 1);
    }
    int result = 0;
    for (long key : map.keySet()) {
        if (map.containsKey(key + 1)) {
            result = Math.max(result, map.get(key + 1) + map.get(key));
        }
    }
    return result;
}
```

written by jaqenhgar original link here

Solution 2

Let `count[x]` be the number of `x`'s in our array.
Suppose our longest subsequence `B` has `min(B) = x` and `max(B) = x+1`.
Evidently, it should use all occurrences of `x` and `x+1` to maximize it's length, so `len(B) = count[x] + count[x+1]`.
Additionally, it must use `x` and `x+1` atleast once, so `count[x]` and `count[x+1]` should both be positive.

```
def findLHS(self, A):
    count = collections.Counter(A)
    ans = 0
    for x in count:
        if x+1 in count:
            ans = max(ans, count[x] + count[x+1])
    return ans
```

Alternatively, we can count values in a straightforward way using a dictionary: replacing our first line of `count = collections.Counter(A)` with:

```
count = {}
for x in A:
    count[x] = count.get(x, 0) + 1
```

written by awice original link here

Solution 3

```
class Solution {
public:
    int findLHS(vector<int>& nums) {
        map<int, int> freqs;
        for (int n : nums) {
            freqs[n]++;
        }

        int longest = 0;
        int lastNum = 0;
        int lastFreq = 0;
        for (pair<int, int> p : freqs) {
            int freq2 = 0;
            if (lastFreq && p.first == lastNum + 1) {
                freq2 = p.second + lastFreq;
            }
            longest = max(longest, freq2);
            lastNum = p.first;
            lastFreq = p.second;
        }
        return longest;
    }
};
```

written by alexander original link here