3186. Maximum Total Damage With Spell Casting

Solved

Medium | Topics | Companies | Hint

A magician has various spells.

You are given an array `power`, where each element represents the damage of a spell. Multiple spells can have the same damage value.

It is a known fact that if a magician decides to cast a spell with a damage of `power[i]`, they **cannot** cast any spell with a damage of `power[i] - 2`, `power[i] - 1`, `power[i] + 1`, or `power[i] + 2`.

Each spell can be cast **only once**.

Return the **maximum** possible *total damage* that a magician can cast.

Example 1:

> **Input:** `power = [1,1,3,4]`
>
> **Output:** 6
>
> **Explanation:**
>
> The maximum possible damage of 6 is produced by casting spells 0, 1, 3 with damage 1, 1, 4.

Example 2:

> **Input:** `power = [7,1,6,6]`
>
> **Output:** 13
>
> **Explanation:**
>
> The maximum possible damage of 13 is produced by casting spells 1, 2, 3 with damage 1, 6, 6.

Constraints:

- $1 <= power.length <= 10^5$
- $1 <= power[i] <= 10^9$

Python:

```
class Solution:
    def maximumTotalDamage(self, power):
```

```
        count = Counter(power)
        vec = [(-(10**9), 0)]
        for k in sorted(count.keys()):
            vec.append((k, count[k]))
        n = len(vec)
        f = [0] * n
        mx = 0
        j = 1
        for i in range(1, n):
            while j < i and vec[j][0] < vec[i][0] - 2:
                mx = max(mx, f[j])
                j += 1
            f[i] = mx + vec[i][0] * vec[i][1]
        return max(f)
```

JavaScript:

```
var maximumTotalDamage = function (power) {
    let count = new Map();
    for (let p of power) {
        count.set(p, (count.get(p) || 0) + 1);
    }
    let vec = [[-1000000000, 0]];
    let keys = Array.from(count.keys()).sort((a, b) => a - b);
    for (let k of keys) {
        vec.push([k, count.get(k)]);
    }
    let n = vec.length;
    let f = Array(n).fill(0);
    let mx = 0,
        ans = 0,
        j = 1;
    for (let i = 1; i < n; i++) {
        while (j < i && vec[j][0] < vec[i][0] - 2) {
            mx = Math.max(mx, f[j]);
            j++;
        }
        f[i] = mx + vec[i][0] * vec[i][1];
        ans = Math.max(ans, f[i]);
    }
    return ans;
};
```

Java:

```
class Solution {
```

```java
    public long maximumTotalDamage(int[] power) {
        TreeMap<Integer, Integer> count = new TreeMap<>();
        for (int p : power) {
            count.put(p, count.getOrDefault(p, 0) + 1);
        }
        List<int[]> vec = new ArrayList<>();
        vec.add(new int[] { -1000000000, 0 });
        for (Map.Entry<Integer, Integer> e : count.entrySet()) {
            vec.add(new int[] { e.getKey(), e.getValue() });
        }
        int n = vec.size();
        long[] f = new long[n];
        long mx = 0;
        long ans = 0;
        int j = 1;
        for (int i = 1; i < n; i++) {
            while (j < i && vec.get(j)[0] < vec.get(i)[0] - 2) {
                mx = Math.max(mx, f[j]);
                j++;
            }
            f[i] = mx + 1L * vec.get(i)[0] * vec.get(i)[1];
            ans = Math.max(ans, f[i]);
        }
        return ans;
    }
}
```