Explanation :

Problem : Select a number at k as Ak and numbers that equal to Ak -1 and Ak +1 get deleted from the sequences. Getting scores of Ak

Example :

S = [1,1,2,2,3,3,4,5,6]

Select 2 : 1 and 3, *which is 2-1 and 2+1*, is now gone. You gained 2 points and left with

S = [2,4,5,6]

Select 6 : 5, which is *6-1,* is now gone. You gained 6 points and left with

S = [2,4]

Keep on going you will gained a total of 2 + 5 + 2 + 4 = 13 points

There are many ways we can approach to this but we need the best one which we gained the most score.

Approach :

Take a look at [1,1,2,2] in the sequences.

What is the best approach we can do in order to get the most score?

Yes, selecting 2, 1 is gone, and 2 will give you 2+2 = 4 points.

What about [1,1,2,2,3]?

Yes, selecting 3, 2 is gone, and 1 will give you 3+1+1 = 5

What about [1,1,2,3,3,4,5,6]?

Let's start from the very beginning.

Selecting 1 : gives 2 points, and 2 is gone(lose 2 points). [3,3,4,5,6]

Selecting 3: gives 6 points, and 4 is gone(lose 4 points). [5,6]

Selecting 6: gives 6 points, best choices here.(lose 5 points)

You gain a total of 2+6+6 = 14 points, and lose 2+4+5 = 11 points. The best approach here.

*Why? Let's see what happened when selecting 2.*

Selecting 2: gives 2 points, 1 and 3 is gone, lose(1+1+3+3 = 8 points). [4,5,6]

Selecting 4 then 6 would be the best choices here : gives 10 points, and lose 5 points.

You gain a total of 2+10 = 12 points, and lose 8+5 = 13 points. See?

So how do we implement this to code?

One of the solution here is to compared the outcome.

Which I implement like this :

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

typedef long long int ln;

ln cnt[100005] = {0};
ln max_point[100005];

int main(){
    int n,max_val = 0; cin>>n ;
    for(int i = 0;i < n; i++){
        int x;
        cin >> x;
        cnt[x]++;
        max_val = max(max_val,x);
    }
    max_point[0] = 0;
    max_point[1] = cnt[1];
    for(int i = 2;i <= max_val; i++){
        max_point[i] = max(max_point[i-1], max_point[i-2] + i*cnt[i]);
    }

    cout << max_point[max_val];
}
```

Let's take a look at sequences of [1,1,2,3,3,4,5,6]

cnt's array :

| Index (i) | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Value | 0 | 2 | 1 | 2 | 1 | 1 | 1 |

max_point's array:

| Index (i) | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| value | 0 | 2 | 2 | 8 | 8 | 13 | 14 |

Considered the green-highlighted column of max_point's array. The value of it comes from compared the previous max_point at (i-1) and ( max_point at (i-2) + i*cnt[i] ).

And the yellow-highlighted one show the unselected numbers.

So, we will able to see the best outcome between chosen number.