📚 Grumpy Bookstore Owner – Tumhare Code Ki Full Explanation LeetCode:1052

🔧 Problem Recap:

- `customers[i]`: Customers coming at minute `i`
- `grumpy[i] == 1`: Owner is grumpy → customers unhappy 😡
- `grumpy[i] == 0`: Owner is happy → customers happy 😊
- Owner can use a secret technique to be **not grumpy for `minutes` minutes continuously** ⏳ (only once)

🎯 **Goal**: Maximize total satisfied customers 😄

✅ Tumhara Approach (Unique & Smart)

> "Main dekh raha hoon ki kis window me sabse zyada unsatisfied customers hai, ussi window me trick lagani hai!" 🔍💡

🔍 Step-by-Step Code Breakdown:

```
int maxSatisfied(vector<int>& customers, vector<int>& grumpy, int minutes) {
```

🧠 Start of function: `customers` and `grumpy` vectors are inputs, and `minutes` is the duration of the secret technique.

```
int n = customers.size();
int maxidx = 0;             // 🧭 Index to apply the trick
```

```
int sum = 0;              // 🤯 Current sum of unsatisfied customers in window
int maxsum = INT_MIN;     // 🔝 Max of such sum found so far
```

- We're going to find the window of length `minutes` where **most customers are lost** due to grumpiness.

🪟 Step 1: Initial Window Calculation

```
for(int i=0; i<minutes; i++){
    sum += customers[i] * grumpy[i];  // 😤 Only count if owner is grumpy
}
```

🪟 First window from `0` to `minutes-1`

📦 `sum` contains unsatisfied customers in this window.

```
int i = 1;
maxsum = sum;
int j = i + minutes - 1;
```

- Start sliding the window
- `i`: Start of the new window
- `j`: End of the window

🔁 Step 2: Sliding Window Begins

```
while(j < n){
    sum = sum + (customers[j] * grumpy[j]) - (customers[i-1] * grumpy[i-1]);
```

- Add the new element coming into the window (if grumpy)
- Subtract the old element going out of the window (if grumpy)

```
    if(maxsum < sum){
        maxsum = sum;      // 🔝 Update max if current window is better
```

```
            maxidx = i;        // 🧭 Store the starting index of the best window
        }
        i++;
        j++;
    }
```

🧪 Step 3: Apply the Trick!

```
for(int i = maxidx; i < maxidx + minutes; i++){
    grumpy[i] = 0;  // 💊 Owner becomes happy in the best window
}
```

🪄 This effectively turns **grumpy minutes to happy**, making customers in this window satisfied!

🧱 Step 4: Final Count of Happy Customers

```
int anssum = 0;
for(int i=0;i<n;i++){
    if(grumpy[i] == 0){
        anssum += customers[i];
    }
}
```

✔️ Add up all customers where `grumpy[i] == 0` → includes original happy minutes + trick-applied window.

🔚 Step 5: Return the Answer

```
return anssum;
```

🎉 Return the **maximum number of satisfied customers**!

🧠 Time & Space Complexity

Type	Value
Time Complexity	O(n) ✅
Space	O(1) ✅ (in-place grumpy update)

🔑 Summary (Emoji Style)

Step	What it Does	Emoji Summary
🪟 Initial	Calculate lost customers in first window	😤📉
🔁 Slide	Track max lost customers in any window	🪟🔄📈
💊 Trick	Apply trick to that window	✨😇
🧱 Final	Count all satisfied customers now	🧾🧠✅

🏁 Example:

```
customers = [1,0,1,2,1,1,7,5]
grumpy    = [0,1,0,1,0,1,0,1]
minutes   = 3
```

✅ Trick applied from index `5 → 7`, turning those grumpy minutes happy → Final Answer = `16`