**Problem Explanation**

The template is an array of integers, and we are given several strings. We need to check if each string matches the template under the following conditions:

1. **Length Matching**: The length of the string must be equal to the size of the template array.
2. **Consistent Mapping**: Each integer in the template array should map to a unique character in the string, and each character in the string should map to a unique integer in the template. This means two different integers cannot map to the same character, and two different characters cannot map to the same integer.

In simpler terms, for the array and the string to "match":

* The array [3, 5, 2, 1, 3] should map consistently to the string "abfda", meaning 3 can only map to a, 5 can only map to b, and so on. If any inconsistency arises (e.g., if 3 tries to map to both a and b), the string is not valid.

**Logic**

1. **Start Mapping**:
   * For each character in the string, check if it and the corresponding integer are already mapped.
   * If neither is mapped, create a new consistent mapping.
2. **Check Consistency**:
   * If a character is already mapped to an integer but the current integer is different, or if an integer is already mapped to a different character, the mapping is inconsistent, and the string doesn't match the template.
3. **Result**:
   * If all characters and integers map consistently, the string is valid for the template.
   * If there's any inconsistency, the string is not valid.

### Example with Visualization

Let's take a specific example for clarity:

* **Template Array a[]:** [3, 5, 2, 1, 3]
* **String s:** "abfda"

We need to check if this string matches the template. Here's what happens in the loop:

| **Index (i)** | **Character (s[i])** | **Integer (a[i])** | **Mapping Check (Step-by-step)** |
| --- | --- | --- | --- |
| **0** | 'a' | 3 | Neither 'a' nor 3 have been mapped yet. So, map 'a' -> 3 and 3 -> 'a'. |
| **1** | 'b' | 5 | Neither 'b' nor 5 have been mapped yet. So, map 'b' -> 5 and 5 -> 'b'. |
| **2** | 'f' | 2 | Neither 'f' nor 2 have been mapped yet. So, map 'f' -> 2 and 2 -> 'f'. |
| **3** | 'd' | 1 | Neither 'd' nor 1 have been mapped yet. So, map 'd' -> 1 and 1 -> 'd'. |
| **4** | 'a' | 3 | Now, we check if 'a' is already mapped to 3. Yes, it is ('a' -> 3 and 3 -> 'a'), so the mapping is consistent. |

Since all characters and integers have been mapped consistently, the result for this string is **YES**.

### Another Example (with Inconsistency)

Now, let's take an example where the mapping fails:

* **Template Array a[]:** [3, 5, 2, 1, 3]
* **String s:** "afbfda"

Here, at index 4, the mapping will fail. Let's break it down step by step:

| **Index (i)** | **Character (s[i])** | **Integer (a[i])** | **Mapping Check (Step-by-step)** |
| --- | --- | --- | --- |
| **0** | 'a' | 3 | Map 'a' -> 3 and 3 -> 'a'. |
| **1** | 'f' | 5 | Map 'f' -> 5 and 5 -> 'f'. |
| **2** | 'b' | 2 | Map 'b' -> 2 and 2 -> 'b'. |
| **3** | 'f' | 1 | Now, 'f' is already mapped to 5, but here it's trying to map to 1, which causes a conflict. The program sets ok = false and breaks out of the loop. |

Since there's an inconsistency, the result for this string is **NO**.

### Mapping Explanation

* **First if condition**: ump1.find(s[i]) == ump1.end() && ump2.find(a[i]) == ump2.end()
  + This checks if the character s[i] has **never been mapped** to any integer and if the integer a[i] has **never been mapped** to any character.
  + If both are unseen, they are mapped to each other (i.e., the character is mapped to the integer and vice versa).
* **Second else if condition**:
  + (ump1.find(s[i]) != ump1.end() && ump1[s[i]] != a[i])
    - This checks if the character s[i] has already been mapped to an integer **but** the mapping doesn't match the current integer a[i]. If there's a mismatch, we set ok = false.
  + (ump2.find(a[i]) != ump2.end() && ump2[a[i]] != s[i])
    - Similarly, it checks if the integer a[i] has already been mapped to a character **but** the mapping doesn't match the current character s[i]. Again, if there's a mismatch, we set ok = false