Binary Search Fix

Understanding the "Kth Missing Positive Number" Problem

Problem Statement Recap

You are given a **strictly increasing array** `vec[]` and a **positive integer** `k`. Your task is to **find the** `k`**-th missing positive integer** from the sequence.

How to Identify Missing Numbers?

A strictly increasing array means every number is greater than the previous one. However, **some numbers are missing** in between the sequence, and we need to find the `k`-th missing one.

Examples to Understand the Problem

Example 1

Input:

```plaintext
vec = [4, 7, 9, 10], k = 1
```

Missing numbers:

```plaintext
1, 2, 3, 5, 6, 8, 11, 12, ...
```

- The **1st missing number** is `1`.

Output:

```plaintext

```

```
1
```

Example 2

Input:

```plaintext
vec = [4, 7, 9, 10], k = 4
```

Missing numbers:

```plaintext
1, 2, 3, 5, 6, 8, 11, 12, ...
```

- The **4th missing number** is `5`.

Output:

```plaintext
5
```

Example 3

Input:

```plaintext
vec = [2, 3, 5, 8], k = 2
```

Missing numbers:

```plaintext
```

```
1, 4, 6, 7, 9, 10, 11, ...
```

- The **2nd missing number** is `4`.

Output:

```plaintext
4
```

Understanding the Two Key Cases

Now, let's analyze what happens when we go through the array **while searching for the `k`-th missing number**.

Case 1: When `vec[i] < k`

📌 *The current element in the array is smaller than `k`.*
Effect:

- Since this number is present, **it means one of the earlier missing numbers is accounted for.**
- The missing number **shifts forward by 1** because the current number reduces the missing count.
- So, **increase `k` by 1.**

Example for Case 1

```plaintext
vec = [2, 3, 5, 8], k = 2
```

- Initially, `k = 2` (we are looking for the 2nd missing number).
- First element `vec[0] = 2`, which is **less than** `k = 2`, so we **shift** `k` **to 3.**
- Second element `vec[1] = 3`, which is **less than** `k = 3`, so we **shift** `k` **to 4.**

- Next element `vec[2] = 5`, which is **greater than** `k = 4`, so **4 is our answer**.

Final Output:

```plaintext
4
```

Case 2: When `vec[i] > k`

📌 *The current element in the array is greater than* `k`.

Effect:

- Since `k` is missing **before this number appears**, it must be one of the missing numbers.
- **No need to shift** `k`, as `k` is still missing.
- So, **return** `k` as the answer.

Example for Case 2

```plaintext
vec = [7, 10, 12], k = 4
```

- The first number is `7`, which is **greater than** `k = 4`.
- This means `4` is still missing **and should be returned**.

Final Output:

```plaintext
4
```

Conclusion

- **When a number smaller than** `k` **is present in** `vec[]`**, we shift** `k` **forward.**

- **When a number greater than `k` appears, `k` is the missing number and should be returned.**
- This approach helps us efficiently find the `k`-th missing number without iterating through every number one by one.

Let me know if you need further clarification! 🚀