## Input Example:

s = "leetcode"

Expected Output: 0 // 'l' is the first non-repeating character

# Approach 1: **Frequency Array**

int[] freq = new int[26];

## Step 1: Count character frequencies

|  |  |  |  |
| --- | --- | --- | --- |
| Index | Character | freq[char - 'a'] Before | After |
| 0 | 'l' | 0 | 1 |
| 1 | 'e' | 0 | 1 |
| 2 | 'e' | 1 | 2 |
| 3 | 't' | 0 | 1 |
| 4 | 'c' | 0 | 1 |
| 5 | 'o' | 0 | 1 |
| 6 | 'd' | 0 | 1 |
| 7 | 'e' | 2 | 3 |

## Step 2: Find first char with count 1

| **Index** | **Char** | **freq[char - 'a']** | **Action** |
| --- | --- | --- | --- |
| 0 | 'l' | 1 | ✅ Return 0 |
|  |  |  |  |

# Using FreqArray:

class Solution {

    public int firstUniqChar(String s) {

        int[] freq = new int[26];

        //count freq

        for(int i=0; i<s.length(); i++){

            freq[s.charAt(i) -'a']++;

        }

        //first the 1st char with count=1

        for(int i=0; i<s.length(); i++){

            if(freq[s.charAt(i) -'a'] == 1)

                return i;

        }

        return -1;

    }

}

# Approach 2: **HashMap**

Map<Character, Integer> freq = new HashMap<>();

## Step 1: Count character frequencies

|  |  |  |  |
| --- | --- | --- | --- |
| Index | Char | freq.get(ch) Before | After |
| 0 | 'l' | null | 1 |
| 1 | 'e' | null | 1 |
| 2 | 'e' | 1 | 2 |
| 3 | 't' | null | 1 |
| 4 | 'c' | null | 1 |
| 5 | 'o' | null | 1 |
| 6 | 'd' | null | 1 |
| 7 | 'e' | 2 | 3 |

## Step 2: Find first char with count 1

| **Index** | **Char** | **freq.get(ch)** | **Action** |
| --- | --- | --- | --- |
| 0 | 'l' | 1 | ✅ Return 0 |
|  |  |  |  |

## Comparison Summary:

|  |  |  |
| --- | --- | --- |
| Metric | Freq Array | HashMap |
| Characters Supported | Only 'a' to 'z' (lowercase) | Any character (Unicode) |
| Time Complexity | O(n) | O(n) (average case) |
| Space Complexity | O(1) — fixed size 26 | O(n) — grows with distinct chars |
| Performance | Very fast (no hashing) | Slightly slower (uses hashing) |
| Use Case | Ideal for lowercase a–z only | Needed for general strings |

## Final Output:

0 // 'l' is the first non-repeating character

# Using HashMap

class Solution {

    public int firstUniqChar(String s) {

        HashMap<Character, Integer> freq = new HashMap<>();

        //count freq

        for(int i=0; i<s.length(); i++){

            char ch = s.charAt(i);

            if(freq.get(ch) == null)

                freq.put(ch, 1);

            else

                freq.put(ch, freq.get(ch)+1);

        }

        //first the 1st char with count=1

        for(int i=0; i<s.length(); i++){

            if(freq.get(s.charAt(i)) == 1)

                return i;

        }

        return -1;

    }

}