Strings in DSA: Patterns, Problems & Interview Mastery

Subtitle: Learn how to solve string problems using patterns that frequently appear in coding interviews.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Strings are one of the most frequently tested topics in coding interviews. Many problems that seem complex are actually simple once you understand common string patterns. This guide focuses on practical techniques and problem-solving approaches to master strings.

Step 1: What is a String?

A string is a **sequence of characters**.

- Stored as an array of characters internally
- Indexed starting from **0**

Example:

```
let str = "hello";

console.log(str[0]); // h
```

Step 2: Basic String Operations

1. Traversing a String

```
for (let i = 0; i < str.length; i++) {

  console.log(str[i]);

}
```

2. Reverse a String

```
let reversed = str.split('').reverse().join('');
```

3. Convert Case

```
str.toUpperCase();
```

```
str.toLowerCase();
```

4. Check Substring

```
str.includes("ell"); // true
```

Step 3: Important String Patterns

1. Two Pointers (Palindrome Check)

Used to compare characters from both ends

```
function isPalindrome(str) {
  let left = 0;
  let right = str.length - 1;

  while (left < right) {
    if (str[left] !== str[right]) return false;
    left++;
    right--;
  }
  return true;
}
```

2. Frequency Count (Hash Map)

Used to count characters

```
let map = {};

for (let char of str) {
  map[char] = (map[char] || 0) + 1;
}
```

3. Sliding Window (Substrings)

Used for substring problems

```
let set = new Set();

let left = 0;


for (let right = 0; right < str.length; right++) {
  while (set.has(str[right])) {
    set.delete(str[left]);
    left++;
  }
  set.add(str[right]);
}
```

Step 4: Important Problems You Must Practice

1. Check Palindrome

Already covered using two pointers.

2. Count Vowels

```
let count = 0;

let vowels = "aeiou";


for (let char of str.toLowerCase()) {
  if (vowels.includes(char)) count++;
}
```

3. Find First Non-Repeating Character

```
let map = {};


for (let char of str) {
```

```
  map[char] = (map[char] || 0) + 1;
}

for (let char of str) {
  if (map[char] === 1) {
    console.log(char);
    break;
  }
}
```

4. Longest Substring Without Repeating Characters

```
let set = new Set();
let left = 0;
let maxLength = 0;

for (let right = 0; right < str.length; right++) {
  while (set.has(str[right])) {
    set.delete(str[left]);
    left++;
  }
  set.add(str[right]);
  maxLength = Math.max(maxLength, right - left + 1);
}
```

Step 5: Common Mistakes

- Ignoring **case sensitivity**
- Not handling **empty strings**
- Using inefficient approaches instead of patterns
- Forgetting edge cases like **single character strings**

Step 6: Practice Plan for Strings

Day 1:

- Basics + traversal
- 5 easy problems

Day 2–3:

- Palindrome + frequency count
- 8–10 problems

Day 4–5:

- Sliding window
- 10–12 problems

Day 6–7:

- Mixed problems + revision
- 15 problems

Mini Exercises

1. Reverse words in a sentence
2. Check if two strings are **anagrams**
3. Count frequency of each character
4. Find longest palindrome substring (try basic version)

Key Takeaways

- Strings are **high-frequency interview topics**
- Master patterns like **two pointers, hash maps, sliding window**
- Focus on **logic building**, not memorization
- Practice a variety of problems consistently
- Handle **edge cases carefully**

Visit haas.dev for more DSA guides, coding problems, and interview preparation resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app