

Shortest Palindrome

Add chars to front — KMP on reversed string ■

● HARD

■ Hook

Given string s , find the shortest palindrome by adding characters only to the front. Insight: find the longest palindromic prefix of s . Then prepend the reverse of the remaining suffix. KMP does this in $O(n)$.

■ Core Intuition

■ The KMP Trick

Concatenate: $s + \# + \text{reverse}(s)$. Build KMP LPS array. The last value in $\text{lps}[]$ = length of longest palindromic prefix. Answer = $\text{reverse}(s[\text{lps_last}..n-1]) + s$

■ Visual — $s = \text{'aacecaaa'}$

```
s = 'aacecaaa'  rev = 'aaacecaa'
combined = 'aacecaaa#aaacecaa'
LPS last value = 7
Longest palindromic prefix length = 7 ('aacecaa')
Remaining suffix = s[7..] = 'a'
Prepend reverse('a') = 'a'
ANSWER = 'a' + 'aacecaaa' = 'aaacecaaa'
```

■ Code

```

public String shortestPalindrome(String s) {
    String rev = new StringBuilder(s).reverse().toString();
    String combined = s + '#' + rev;
    int[] lps = buildLPS(combined);
    int palLen = lps[combined.length()-1];
    return rev.substring(0, s.length()-palLen) + s;
}

int[] buildLPS(String s) {
    int n=s.length();
    int[] lps=new int[n];
    int len=0, i=1;
    while(i<n){
        if(s.charAt(i)==s.charAt(len)) lps[i++]=++len;
        else if(len!=0) len=lps[len-1];
        else lps[i++]=0;
    }
    return lps;
}

```

■ TIME

$O(n)$

■ SPACE

$O(n)$

- ✓ Find longest palindromic prefix using KMP
- ✓ Concatenate $s + \text{'\#'} + \text{reverse}(s)$, build LPS
- ✓ $\text{lps}[\text{last}] = \text{length of longest palindromic prefix}$
- ✓ Answer = reverse(remaining suffix) + s
- ✓ Time: $O(n)$ | Space: $O(n)$

■ MENTAL MODEL — Remember This Forever

The '#' separator prevents the palindromic prefix from crossing over into the reversed part. The last LPS value tells us how much of s already forms a palindrome from the start. We only need to add the reverse of what's left — minimum characters.