

Minimum Characters to Make Palindrome

Reverse + KMP = min insertions ■

● HARD

■ Hook

Find the minimum characters to insert at the front of string s to make it a palindrome. This is identical to Shortest Palindrome (LC 214). Answer = $n - (\text{longest palindromic prefix length})$.

■ Core Intuition

The longer the palindromic prefix, the fewer characters we need to prepend. Find longest palindromic prefix using KMP (same technique as Shortest Palindrome). Min chars = $n - \text{lps_last}$.

■ Visual

```
s = 'abcd'
rev = 'dcba'
combined = 'abcd#dcba'
lps = [0,0,0,0,0,0,0,0,1] ← lps[last]=1? let's check:
Actually: 'a' at start matches 'a' at end → lps[last]=1
Longest palindromic prefix = 'a' (length 1)
Min insertions = 4 - 1 = 3 (prepend 'dcb')
Result: 'dcbabcd' ✓
s = 'abacaba' → already a palindrome → 0 insertions
```

■ Code

```
public int minCharsToMakePalindrome(String s) {
    String rev = new StringBuilder(s).reverse().toString();
    String combined = s + '#' + rev;
    int[] lps = buildLPS(combined);
    return s.length() - lps[combined.length()-1];
}
// Same buildLPS as in KMP topic
// Answer = n - palPrefixLen
// To reconstruct: prepend reverse of s.substring(palPrefixLen)
```

■ TIME

$O(n)$

■ SPACE

$O(n)$

■ Patterns & Connections

This family of problems (Shortest Palindrome, Longest Happy Prefix, Min Chars for Palindrome) all reduce to: **find the longest prefix-suffix match** = KMP LPS. Recognizing this family makes all three problems trivial once you master KMP.

- ✓ min insertions = $n - (\text{longest palindromic prefix length})$
- ✓ Find longest palindromic prefix via KMP: combine $s + \# + \text{reverse}(s)$
- ✓ $\text{lps}[\text{last}] = \text{longest palindromic prefix length}$
- ✓ To reconstruct: prepend $\text{reverse}(s[\text{palLen}..n-1])$
- ✓ Time: $O(n)$ | Space: $O(n)$

■ MENTAL MODEL — Remember This Forever

Minimum insertions to make a palindrome = n minus the longest palindromic prefix. The longer the head-start palindrome you have, the less work remains. KMP finds this head-start in $O(n)$. Pattern: anything involving 'prefix that is also suffix' → KMP LPS.