**Valid Palindrome II**

**Approach 1: Brute force**

1. At each index, try removing the letter and check if resulting string forms a palindrome
2. If yes, return True, because we could form a palindrome after removing a letter, otherwise False

Time -> O(n ^ n)

Space -> O(n)

**Approach 2: Efficient**

1. Start comparing letter from start to end, if a mismatch found, form 2 string, in one string omit the left char and in the other right char, if any of these return that the string is a palindrome, we return True
2. If at the end nothing returned, string already a palindrome and therefore reaches end of loop, return True

Time -> O(n)

Space -> O(1)

**Code**:

def validPalindrome(self, s: str) -> bool:

n = len(s)

start = 0

end = len(s)-1

while start < end:

if s[start] !=s[end]:

# skip last letter

s1 = s[start:end]

# skip start letter

s2 = s[start+1: end+1]

return self.palin(s1) or self.palin(s2)

start += 1

end -= 1

return True