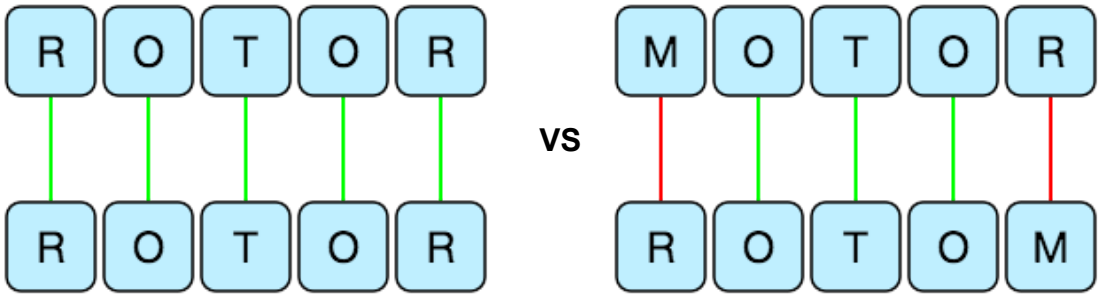
Using recursion to determine whether a word is a palindrome

A **palindrome** is a word that is spelled the same forward and backward. For example, *rotor* and *redder* are palindromes, but *motor* is not.

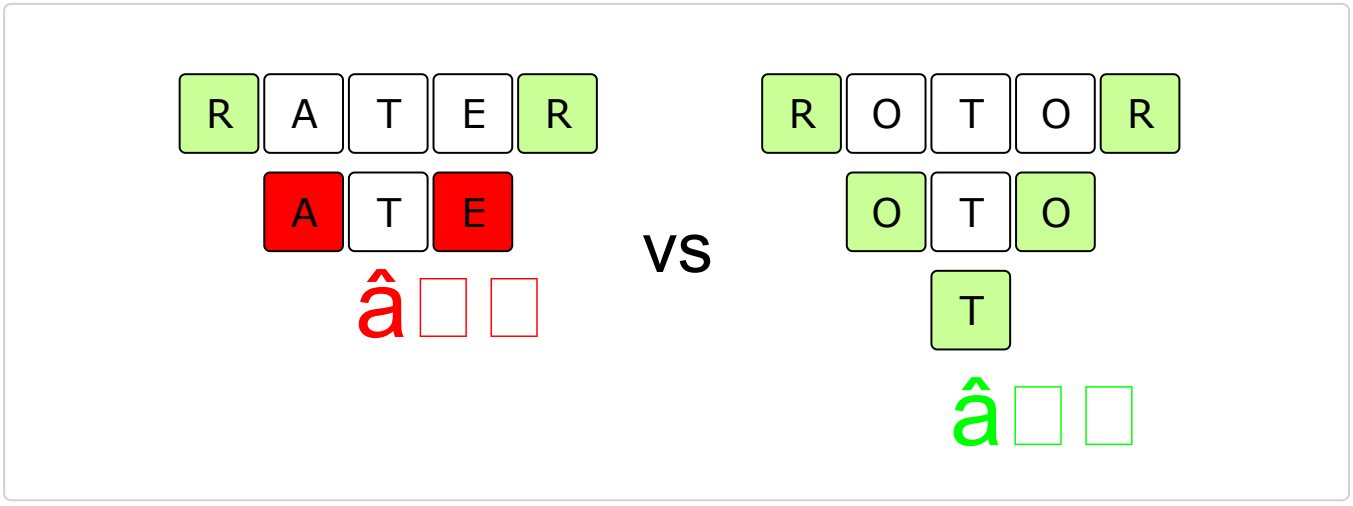


How can you use recursion to determine whether a word is a palindrome? Let's start by understanding what's a base case. Consider the word a. It's a palindrome. In fact, we don't have to think of palindromes as actual words in the English language (or whatever language you'd like to consider). We can think of a palindrome as just any sequence of letters that reads the same forward and backward, such as xyzyzyx. We call a sequence of letters a **string**. So we can say that any string containing just one letter is by default a palindrome. Now, a string can contain no letters; we call a string of zero letters an **empty string**. An empty string is also a palindrome, since it "reads" the same forward and backward. So now let's say that any string containing at most one letter is a palindrome. That's our base case: a string with exactly zero letters or one letter is a palindrome.

What if the string contains two or more letters? Here's where we'll have our recursive case. Consider the palindrome rotor. Its first and last letters are the same, and this trait must hold for any palindrome. On the other hand, if the first and last letters are not the same, as in motor, then the string cannot

possibly be a palindrome. So now we have a way to declare that a string is not a palindrome: when its first and last letters are different. We can think of this situation as another base case, since we have the answer immediately. Going back to when the first and last letters are the same, what does that tell us? The string might be a palindrome. Then again, it might not be. In the string rater, the first and last letters are the same, but the string is not a palindrome. Suppose we strip off the first and last letters, leaving the string ate. Then the first and last letters of this remaining string are not the same, and so we know that rater is not a palindrome.

So here's how we can recursively determine whether a string is a palindrome. If the first and last letters differ, then declare that the string is not a palindrome. Otherwise, strip off the first and last letters, and determine whether the string that remains—the subproblem—is a palindrome. Declare the answer for the shorter string to be the answer for the original string. Once you get down to a string with no letters or just one letter, declare it to be a palindrome. Here's a visualization of that for two words that we discussed:



How would we describe that in pseudocode?

- If the string is made of no letters or just one letter, then it is a palindrome.
- Otherwise, compare the first and last letters of the string.
    - If the first and last letters differ, then the string is not a palindrome.
    - Otherwise, the first and last letters are the same. Strip them from the string, and determine whether the string that remains is a palindrome. Take the answer for this smaller string and use it as the answer for the original string.