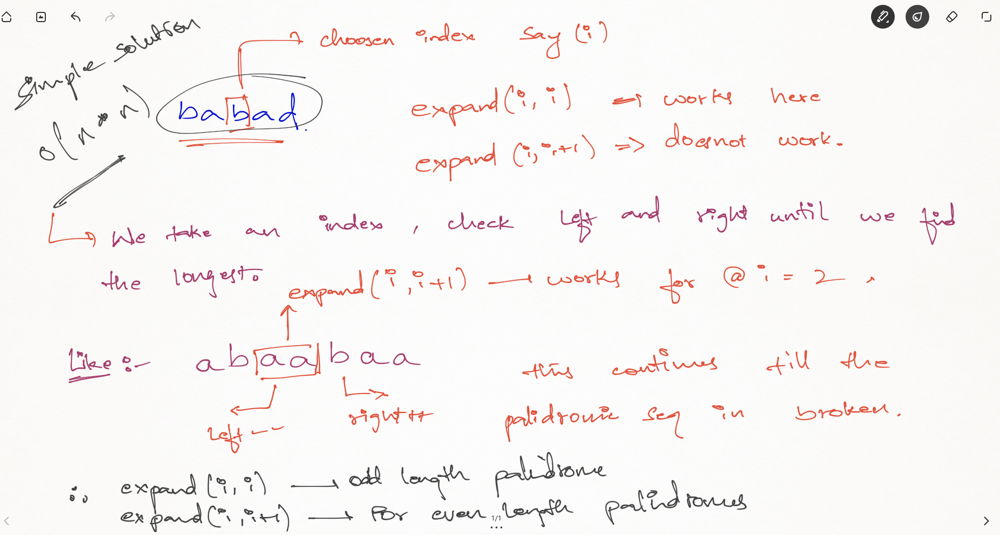
A diagram of a string

Description automatically generated

The only tricky thing is that you have to be sure the correct substring is returned from extendPalindrome (). The loop ends only after expanding the range by 1 on both sides, so you have to remove those in the final returned string.

Here we basically choose each centre and try expanding from each and every specific node thus we call extendPalindrome (i,i) and extendPalindrome (i,i+1) and later we apply two pointers on each node to find the longest palindrome.



In fact, we could solve it in *O*(*n*2) time using only constant space.

We observe that a palindrome mirrors around its center. Therefore, a palindrome can be expanded from its center, and there are only 2*n*−1 such centers.

You might be asking why there are 2*n*−1 but not *n* centers? The reason is the center of a palindrome can be in between two letters. Such palindromes have even number of letters (such as "abba") and its center are between the two 'b's. Hence, n centers for odd and n-1 centers for even. Total = n + (n-1) = 2n-1.

**Time and Space Complexity:**

**Time Complexity:** *O*(*n*2) as there is two recursion calls which are applied as two pointers so here Complexity would be*O*(*n*2).

**Space Complexity:** *O*(*n*) which is nothing but the storage consumed in this process.