**Brief Explanation:**

In whatever way we apply rotations (left rotation by some x or right rotation by some y or left rotation by some x followed by right rotation by some y...) total number of possible strings we can get is <= N (N when all chars are different, but might be < N when duplicate chars. Consider “aaa” there will be only one string “aaa” after any rotation). So if given string is a rotation of any palindrome then that palindrome will be one of the N strings. We can try all rotations and find if it is a palindrome or not. Finding if a given string is palindrome or not is O(N). So, overall time complexity will be O(N) \* O(N) = O(N ^ 2).

**Detailed Explanation:**

In whatever way we apply rotations (left rotation by some x or right rotation by some y or left rotation by some x followed by right rotation by some y...) total number of possible strings we can get is N (N when all chars are different, but might be < N when duplicate chars. Consider “aaa” there will be only one string “aaa” after any rotation).

Initially we might think that there are many ways to apply rotations then number of strings we get will also be more, but no! Let’s take one example, “abcd” now if we do left rotation by 1 then it will give “bcda”, if we do right rotation by 3 then also it will give “bcda” and if we do left rotation by 3 and then do right rotation by 2 then also it will give “bcda”. So eventually number of different strings we can get after rotation will be no more than N.

Applying left shift/right shift by 0 to N - 1 times will generate all possible strings. Consider “abc” then 1) right shift by 0 will generate“abc” 2) right shift by 1 will generate “cab” 3) right shift by 2 will give “bca”.

So if given string is a rotation of any palindrome then that palindrome will be one of the N strings. Let’s take one example, “aab”. Now total strings we get are “aab”, “aba”, “baa”. We can see that “aba” is a palindrome so “aab” is a rotated palindrome! Now consider “abc”. Total strings we get are “abc”, “bca”, “cab”. So here none of the three is a palindrome hence we can say that “abc” is not a rotated palindrome.

Finding if a given string is palindrome or not is O(N). So, overall time complexity will be O(N) \* O(N) = O(N ^ 2).

**Just for information**:

More efficient solution like O(N logN) possible using suffix array and LCP array.

Link for suffix array:

<http://www.geeksforgeeks.org/suffix-array-set-2-a-nlognlogn-algorithm/>

Link for LCP array:

<http://www.geeksforgeeks.org/%C2%AD%C2%ADkasais-algorithm-for-construction-of-lcp-array-from-suffix-array/>

Also more efficient solutions like O(N) possible using longest palindromic substring.

Link for longest palindromic substring:

<https://en.wikipedia.org/wiki/Longest_palindromic_substring>