Problem 214: Shortest Palindrome

Problem Information

Difficulty: Hard
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

You are given a string

s

. You can convert

s

to a

palindrome

by adding characters in front of it.

Return

the shortest palindrome you can find by performing this transformation

.

Example 1:

Input:

s = "aacecaaa"

Output:

"aaacecaaa"

Example 2:

Input:

s = "abcd"

Output:

"dcbabcd"

Constraints:

0 <= s.length <= 5 * 10

4

s

consists of lowercase English letters only.

Code Snippets

C++:

```
class Solution {
public:
string shortestPalindrome(string s) {

}
};
```

Java:

```
class Solution {
public String shortestPalindrome(String s) {
```

```
}
}
```

Python3:

```
class Solution:
def shortestPalindrome(self, s: str) -> str:
```

Python:

```
class Solution(object):
def shortestPalindrome(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {string}
*/
var shortestPalindrome = function(s) {

};
```

TypeScript:

```
function shortestPalindrome(s: string): string {

};
```

C#:

```
public class Solution {
public string ShortestPalindrome(string s) {

}
}
```

C:

```
char* shortestPalindrome(char* s) {

}
```

Go:

```
func shortestPalindrome(s string) string {

}
```

Kotlin:

```
class Solution {
fun shortestPalindrome(s: String): String {

}
}
```

Swift:

```
class Solution {
func shortestPalindrome(_ s: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn shortest_palindrome(s: String) -> String {

}
}
```

Ruby:

```
# @param {String} s
# @return {String}
def shortest_palindrome(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return String
*/
function shortestPalindrome($s) {

}
}
```

Dart:

```
class Solution {
String shortestPalindrome(String s) {

}
}
```

Scala:

```
object Solution {
def shortestPalindrome(s: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec shortest_palindrome(s :: String.t) :: String.t
def shortest_palindrome(s) do

end
end
```

Erlang:

```
-spec shortest_palindrome(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
shortest_palindrome(S) ->
```

```
.
```

Racket:

```
(define/contract (shortest-palindrome s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Shortest Palindrome
* Difficulty: Hard
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
string shortestPalindrome(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Shortest Palindrome
* Difficulty: Hard
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
class Solution {
public String shortestPalindrome(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Shortest Palindrome
Difficulty: Hard
Tags: string, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def shortestPalindrome(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def shortestPalindrome(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Shortest Palindrome
* Difficulty: Hard
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} s
* @return {string}
*/
var shortestPalindrome = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Shortest Palindrome
* Difficulty: Hard
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function shortestPalindrome(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Shortest Palindrome
* Difficulty: Hard
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
```

```
public string ShortestPalindrome(string s) {

}
}
```

C Solution:

```
/*
* Problem: Shortest Palindrome
* Difficulty: Hard
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

char* shortestPalindrome(char* s) {

}
```

Go Solution:

```
// Problem: Shortest Palindrome
// Difficulty: Hard
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func shortestPalindrome(s string) string {

}
```

Kotlin Solution:

```
class Solution {
fun shortestPalindrome(s: String): String {

}
```

```
}
```

Swift Solution:

```
class Solution {
func shortestPalindrome(_ s: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Shortest Palindrome
// Difficulty: Hard
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn shortest_palindrome(s: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {String}
def shortest_palindrome(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return String
```

```
*/
function shortestPalindrome($s) {

}
}
```

Dart Solution:

```
class Solution {
String shortestPalindrome(String s) {

}
}
```

Scala Solution:

```
object Solution {
def shortestPalindrome(s: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec shortest_palindrome(s :: String.t) :: String.t
def shortest_palindrome(s) do

end
end
```

Erlang Solution:

```
-spec shortest_palindrome(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
shortest_palindrome(S) ->
.
```

Racket Solution:

```
(define/contract (shortest-palindrome s)
(-> string? string?)
)
```