Problem 3517: Smallest Palindromic Rearrangement I

Problem Information

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

Problem Description

You are given a

palindromic

string

s

.

Return the

lexicographically smallest

palindromic

permutation

of

s

.

Example 1:

Input:

s = "z"

Output:

"z"

Explanation:

A string of only one character is already the lexicographically smallest palindrome.

Example 2:

Input:

s = "babab"

Output:

"abbba"

Explanation:

Rearranging

"babab"

→

"abbba"

gives the smallest lexicographic palindrome.

Example 3:

Input:

s = "daccad"

Output:

"acddca"

Explanation:

Rearranging

"daccad"

→

"acddca"

gives the smallest lexicographic palindrome.

Constraints:

1 <= s.length <= 10

5

s

consists of lowercase English letters.

s

is guaranteed to be palindromic.

Code Snippets

C++:

```
class Solution {
public:
```

```
string smallestPalindrome(string s) {

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
char* smallestPalindrome(char* s) {

}
```

Go:

```
func smallestPalindrome(s string) string {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Smallest Palindromic Rearrangement I
* Difficulty: Medium
* Tags: string, graph, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Smallest Palindromic Rearrangement I
* Difficulty: Medium
* Tags: string, graph, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String smallestPalindrome(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest Palindromic Rearrangement I
Difficulty: Medium
Tags: string, graph, sort

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Smallest Palindromic Rearrangement I
* Difficulty: Medium
* Tags: string, graph, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Smallest Palindromic Rearrangement I
* Difficulty: Medium
* Tags: string, graph, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function smallestPalindrome(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Smallest Palindromic Rearrangement I
* Difficulty: Medium
* Tags: string, graph, sort
*
```

```
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string SmallestPalindrome(string s) {

}
}
```

C Solution:

```
/*
* Problem: Smallest Palindromic Rearrangement I
* Difficulty: Medium
* Tags: string, graph, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* smallestPalindrome(char* s) {

}
```

Go Solution:

```
// Problem: Smallest Palindromic Rearrangement I
// Difficulty: Medium
// Tags: string, graph, sort
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func smallestPalindrome(s string) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Smallest Palindromic Rearrangement I
// Difficulty: Medium
// Tags: string, graph, sort
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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