Problem 132: Palindrome Partitioning II

Problem Information

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

Problem Description

Given a string

s

, partition

s

such that every

substring

of the partition is a

palindrome

.

Return

the

minimum

cuts needed for a palindrome partitioning of s

.

Example 1:

Input:

s = "aab"

Output:

1

Explanation:

The palindrome partitioning ["aa","b"] could be produced using 1 cut.

Example 2:

Input:

s = "a"

Output:

0

Example 3:

Input:

s = "ab"

Output:

1

Constraints:

1 <= s.length <= 2000

s

consists of lowercase English letters only.

Code Snippets

C++:

```
class Solution {
public:
int minCut(string s) {

}
};
```

Java:

```
class Solution {
public int minCut(String s) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minCut(s: string): number {

};
```

C#:

```
public class Solution {
public int MinCut(string s) {

}
}
```

C:

```
int minCut(char* s) {

}
```

Go:

```
func minCut(s string) int {

}
```

Kotlin:

```
class Solution {
fun minCut(s: String): Int {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn min_cut(s: String) -> i32 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int minCut(String s) {

}
}
```

Scala:

```
object Solution {
def minCut(s: String): Int = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec min_cut(S :: unicode:unicode_binary()) -> integer().
min_cut(S) ->
.
```

Racket:

```
(define/contract (min-cut s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Palindrome Partitioning II
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

```
class Solution {
public:
int minCut(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Palindrome Partitioning II
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int minCut(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Palindrome Partitioning II
Difficulty: Hard
Tags: string, tree, dp

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

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

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Palindrome Partitioning II
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Palindrome Partitioning II
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

function minCut(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Palindrome Partitioning II
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int MinCut(string s) {

}
}
```

C Solution:

```
/*
* Problem: Palindrome Partitioning II
* Difficulty: Hard
* Tags: string, tree, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int minCut(char* s) {

}
```

Go Solution:

```
// Problem: Palindrome Partitioning II
// Difficulty: Hard
// Tags: string, tree, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func minCut(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minCut(s: String): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Palindrome Partitioning II
// Difficulty: Hard
// Tags: string, tree, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn min_cut(s: String) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int minCut(String s) {

}
}
```

Scala Solution:

```
object Solution {
def minCut(s: String): Int = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec min_cut(S :: unicode:unicode_binary()) -> integer().
min_cut(S) ->
.
```

Racket Solution:

```
(define/contract (min-cut s)
(-> string? exact-integer?)
)
```