Problem 838: Push Dominoes

Problem Information

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

Problem Description

There are

n

dominoes in a line, and we place each domino vertically upright. In the beginning, we simultaneously push some of the dominoes either to the left or to the right.

After each second, each domino that is falling to the left pushes the adjacent domino on the left. Similarly, the dominoes falling to the right push their adjacent dominoes standing on the right.

When a vertical domino has dominoes falling on it from both sides, it stays still due to the balance of the forces.

For the purposes of this question, we will consider that a falling domino expends no additional force to a falling or already fallen domino.

You are given a string

dominoes

representing the initial state where:

dominoes[i] = 'L'

, if the

i

th

domino has been pushed to the left,

dominoes[i] = 'R'

, if the

i

th

domino has been pushed to the right, and

dominoes[i] = '.'

, if the

i

th

domino has not been pushed.

Return

a string representing the final state

.

Example 1:

Input:

dominoes = "RR.L"

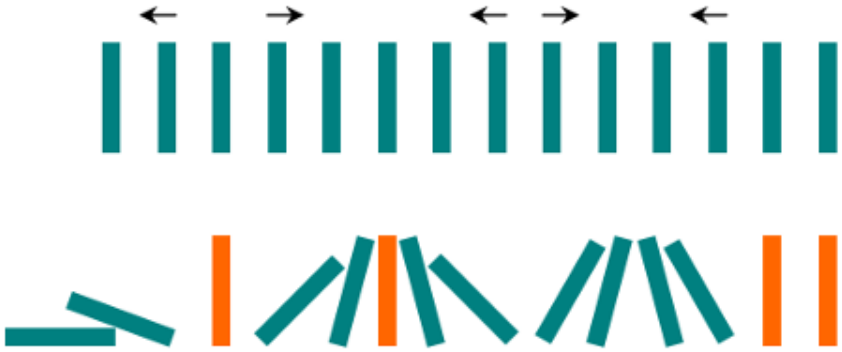
Output:

"RR.L"

Explanation:

The first domino expends no additional force on the second domino.

Example 2:



Input:

dominoes = ".L.R...LR..L.."

Output:

"LL.RR.LLRRLL.."

Constraints:

n == dominoes.length

1 <= n <= 10

5

dominoes[i]

is either

'L'

,

'R'

, or

'.'

.

Code Snippets

C++:

```
class Solution {
public:
string pushDominoes(string dominoes) {

}
};
```

Java:

```
class Solution {
public String pushDominoes(String dominoes) {

}
}
```

Python3:

```
class Solution:
def pushDominoes(self, dominoes: str) -> str:
```

Python:

```
class Solution(object):
def pushDominoes(self, dominoes):
```

```
"""
:type dominoes: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} dominoes
* @return {string}
*/
var pushDominoes = function(dominoes) {

};
```

TypeScript:

```
function pushDominoes(dominoes: string): string {

};
```

C#:

```
public class Solution {
public string PushDominoes(string dominoes) {

}
}
```

C:

```
char* pushDominoes(char* dominoes) {

}
```

Go:

```
func pushDominoes(dominoes string) string {

}
```

Kotlin:

```
class Solution {
fun pushDominoes(dominoes: String): String {

}
}
```

Swift:

```
class Solution {
func pushDominoes(_ dominoes: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn push_dominoes(dominoes: String) -> String {

}
}
```

Ruby:

```
# @param {String} dominoes
# @return {String}
def push_dominoes(dominoes)

end
```

PHP:

```
class Solution {

/**
* @param String $dominoes
* @return String
*/
function pushDominoes($dominoes) {

}
}
```

Dart:

```
class Solution {
String pushDominoes(String dominoes) {

}
}
```

Scala:

```
object Solution {
def pushDominoes(dominoes: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec push_dominoes(dominoes :: String.t) :: String.t
def push_dominoes(dominoes) do

end
end
```

Erlang:

```
-spec push_dominoes(Dominoes :: unicode:unicode_binary()) ->
unicode:unicode_binary().
push_dominoes(Dominoes) ->
.
```

Racket:

```
(define/contract (push-dominoes dominoes)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Push Dominoes
* Difficulty: Medium
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
string pushDominoes(string dominoes) {

}
};
```

Java Solution:

```
/**
* Problem: Push Dominoes
* Difficulty: Medium
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public String pushDominoes(String dominoes) {

}
}
```

Python3 Solution:

```
"""
Problem: Push Dominoes
Difficulty: Medium
Tags: array, string, dp
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

```
class Solution(object):
def pushDominoes(self, dominoes):
"""
:type dominoes: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Push Dominoes
* Difficulty: Medium
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} dominoes
* @return {string}
*/
var pushDominoes = function(dominoes) {

};
```

TypeScript Solution:

```
/**
* Problem: Push Dominoes
* Difficulty: Medium
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function pushDominoes(dominoes: string): string {

};
```

C# Solution:

```
/*
* Problem: Push Dominoes
* Difficulty: Medium
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public string PushDominoes(string dominoes) {

}
}
```

C Solution:

```
/*
* Problem: Push Dominoes
* Difficulty: Medium
* Tags: array, string, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

char* pushDominoes(char* dominoes) {

}
```

Go Solution:

```
// Problem: Push Dominoes
// Difficulty: Medium
// Tags: array, string, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func pushDominoes(dominoes string) string {

}
```

Kotlin Solution:

```
class Solution {
fun pushDominoes(dominoes: String): String {

}
}
```

Swift Solution:

```
class Solution {
func pushDominoes(_ dominoes: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Push Dominoes
// Difficulty: Medium
// Tags: array, string, dp
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn push_dominoes(dominoes: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} dominoes
# @return {String}
def push_dominoes(dominoes)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $dominoes
* @return String
*/
function pushDominoes($dominoes) {

}
}
```

Dart Solution:

```
class Solution {
String pushDominoes(String dominoes) {

}
}
```

Scala Solution:

```
object Solution {
def pushDominoes(dominoes: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec push_dominoes(dominoes :: String.t) :: String.t
def push_dominoes(dominoes) do

end
end
```

Erlang Solution:

```
-spec push_dominoes(Dominoes :: unicode:unicode_binary()) ->
unicode:unicode_binary().
push_dominoes(Dominoes) ->
.
```

Racket Solution:

```
(define/contract (push-dominoes dominoes)
(-> string? string?)
)
```