Problem 2390: Removing Stars From a String

Problem Information

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

Problem Description

You are given a string

s

, which contains stars

*

.

In one operation, you can:

Choose a star in

s

.

Remove the closest

non-star

character to its

left

, as well as remove the star itself.

Return

the string after

all

stars have been removed

.

Note:

The input will be generated such that the operation is always possible.

It can be shown that the resulting string will always be unique.

Example 1:

Input:

s = "leet**cod*e"

Output:

"lecoe"

Explanation:

Performing the removals from left to right: - The closest character to the 1

st

star is 't' in "lee

t

**cod*e". s becomes "lee*cod*e". - The closest character to the 2 nd

star is 'e' in "le

e

*cod*e". s becomes "lecod*e". - The closest character to the 3

rd

star is 'd' in "leco

d

*e". s becomes "lecoe". There are no more stars, so we return "lecoe".

Example 2:

Input:

s = "erase*****"

Output:

""

Explanation:

The entire string is removed, so we return an empty string.

Constraints:

1 <= s.length <= 10

5

s

consists of lowercase English letters and stars

*

.

The operation above can be performed on

s

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

```
class Solution(object):
def removeStars(self, s):
```

```
"""
:type s: str
:rtype: str
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
char* removeStars(char* s) {

}
```

Go:

```
func removeStars(s string) string {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (remove-stars s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Removing Stars From a String
* Difficulty: Medium
* Tags: string, stack
*
* 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 removeStars(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Removing Stars From a String
* Difficulty: Medium
* Tags: string, stack
*
* 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 removeStars(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Removing Stars From a String
Difficulty: Medium
Tags: string, stack
```

```
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 removeStars(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Removing Stars From a String
* Difficulty: Medium
* Tags: string, stack
*
* 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 removeStars = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Removing Stars From a String
* Difficulty: Medium
* Tags: string, stack
*
* 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 removeStars(s: string): string {


};
```

C# Solution:

```
/*
* Problem: Removing Stars From a String
* Difficulty: Medium
* Tags: string, stack
*
* 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 RemoveStars(string s) {


}
}
```

C Solution:

```
/*
* Problem: Removing Stars From a String
* Difficulty: Medium
* Tags: string, stack
*
* 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* removeStars(char* s) {

}
```

Go Solution:

```
// Problem: Removing Stars From a String
// Difficulty: Medium
// Tags: string, stack
//
// 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 removeStars(s string) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Removing Stars From a String
// Difficulty: Medium
// Tags: string, stack
```

```
//
// 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 remove_stars(s: String) -> String {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (remove-stars s)
(-> string? string?)
)
```