Problem 3556: Sum of Largest Prime Substrings

Problem Information

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

Problem Description

Given a string

s

, find the sum of the

3 largest unique

prime numbers

that can be formed using any of its

substrings

.

Return the

sum

of the three largest unique prime numbers that can be formed. If fewer than three exist, return the sum of

all

available primes. If no prime numbers can be formed, return 0.

Note:

Each prime number should be counted only

once

, even if it appears in

multiple

substrings. Additionally, when converting a substring to an integer, any leading zeros are ignored.

Example 1:

Input:

s = "12234"

Output:

1469

Explanation:

The unique prime numbers formed from the substrings of

"12234"

are 2, 3, 23, 223, and 1223.

The 3 largest primes are 1223, 223, and 23. Their sum is 1469.

Example 2:

Input:

s = "111"

Output:

11

Explanation:

The unique prime number formed from the substrings of

"111"

is 11.

Since there is only one prime number, the sum is 11.

Constraints:

1 <= s.length <= 10

s

consists of only digits.

Code Snippets

C++:

```
class Solution {
public:
long long sumOfLargestPrimes(string s) {

}
};
```

Java:

```
class Solution {
public long sumOfLargestPrimes(String s) {
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long SumOfLargestPrimes(string s) {

}
}
```

C:

```
long long sumOfLargestPrimes(char* s) {

}
```

Go:

```
func sumOfLargestPrimes(s string) int64 {

}
```

Kotlin:

```
class Solution {
fun sumOfLargestPrimes(s: String): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn sum_of_largest_primes(s: String) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def sumOfLargestPrimes(s: String): Long = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (sum-of-largest-primes s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sum of Largest Prime Substrings
* Difficulty: Medium
* Tags: string, tree, math, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
long long sumOfLargestPrimes(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Sum of Largest Prime Substrings
* Difficulty: Medium
* Tags: string, tree, math, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public long sumOfLargestPrimes(String s) {
}
}
```

Python3 Solution:

```
"""
Problem: Sum of Largest Prime Substrings
Difficulty: Medium
Tags: string, tree, math, hash, sort

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Sum of Largest Prime Substrings
* Difficulty: Medium
* Tags: string, tree, math, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Sum of Largest Prime Substrings
* Difficulty: Medium
* Tags: string, tree, math, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function sumOfLargestPrimes(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Sum of Largest Prime Substrings
* Difficulty: Medium
* Tags: string, tree, math, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public long SumOfLargestPrimes(string s) {

}
}
```

C Solution:

```
/*
* Problem: Sum of Largest Prime Substrings
* Difficulty: Medium
* Tags: string, tree, math, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

long long sumOfLargestPrimes(char* s) {

}
```

Go Solution:

```
// Problem: Sum of Largest Prime Substrings
// Difficulty: Medium
// Tags: string, tree, math, hash, sort
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func sumOfLargestPrimes(s string) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun sumOfLargestPrimes(s: String): Long {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Sum of Largest Prime Substrings
// Difficulty: Medium
// Tags: string, tree, math, hash, sort
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn sum_of_largest_primes(s: String) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def sumOfLargestPrimes(s: String): Long = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (sum-of-largest-primes s)
(-> string? exact-integer?)
)
```