Problem 3008: Find Beautiful Indices in the Given Array II

Problem Information

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

Problem Description

You are given a

0-indexed

string

s

, a string

a

, a string

b

, and an integer

k

.

An index

i

is

beautiful

if:

0 <= i <= s.length - a.length

s[i..(i + a.length - 1)] == a

There exists an index

j

such that:

0 <= j <= s.length - b.length

s[j..(j + b.length - 1)] == b

|j - i| <= k

Return

the array that contains beautiful indices in

sorted order from smallest to largest

.

Example 1:

Input:

s = "isawsquirrelnearmysquirrelhouseohmy", a = "my", b = "squirrel", k = 15

Output:

[16,33]

Explanation:

There are 2 beautiful indices: [16,33]. - The index 16 is beautiful as s[16..17] == "my" and there exists an index 4 with s[4..11] == "squirrel" and |16 - 4| <= 15. - The index 33 is beautiful as s[33..34] == "my" and there exists an index 18 with s[18..25] == "squirrel" and |33 - 18| <= 15. Thus we return [16,33] as the result.

Example 2:

Input:

s = "abcd", a = "a", b = "a", k = 4

Output:

[0]

Explanation:

There is 1 beautiful index: [0]. - The index 0 is beautiful as s[0..0] == "a" and there exists an index 0 with s[0..0] == "a" and |0 - 0| <= 4. Thus we return [0] as the result.

Constraints:

1 <= k <= s.length <= 5 * 10

5

1 <= a.length, b.length <= 5 * 10

5

s

,

a

, and

b

contain only lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
vector<int> beautifulIndices(string s, string a, string b, int k) {

}
};
```

Java:

```
class Solution {
public List<Integer> beautifulIndices(String s, String a, String b, int k) {

}
}
```

Python3:

```
class Solution:
def beautifulIndices(self, s: str, a: str, b: str, k: int) -> List[int]:
```

Python:

```
class Solution(object):
def beautifulIndices(self, s, a, b, k):
"""
:type s: str
:type a: str
:type b: str
:type k: int
:rtype: List[int]
```

```
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {string} a
* @param {string} b
* @param {number} k
* @return {number[]}
*/
var beautifulIndices = function(s, a, b, k) {

};
```

TypeScript:

```
function beautifulIndices(s: string, a: string, b: string, k: number):
number[] {

};
```

C#:

```
public class Solution {
public IList<int> BeautifulIndices(string s, string a, string b, int k) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* beautifulIndices(char* s, char* a, char* b, int k, int* returnSize) {

}
```

Go:

```
func beautifulIndices(s string, a string, b string, k int) []int {

}
```

Kotlin:

```
class Solution {
fun beautifulIndices(s: String, a: String, b: String, k: Int): List<Int> {

}
}
```

Swift:

```
class Solution {
func beautifulIndices(_ s: String, _ a: String, _ b: String, _ k: Int) ->
[Int] {

}
}
```

Rust:

```
impl Solution {
pub fn beautiful_indices(s: String, a: String, b: String, k: i32) -> Vec<i32>
{

}
}
```

Ruby:

```
# @param {String} s
# @param {String} a
# @param {String} b
# @param {Integer} k
# @return {Integer[]}
def beautiful_indices(s, a, b, k)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param String $a
* @param String $b
* @param Integer $k
* @return Integer[]
*/
function beautifulIndices($s, $a, $b, $k) {

}
}
```

Dart:

```
class Solution {
List<int> beautifulIndices(String s, String a, String b, int k) {

}
}
```

Scala:

```
object Solution {
def beautifulIndices(s: String, a: String, b: String, k: Int): List[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec beautiful_indices(s :: String.t, a :: String.t, b :: String.t, k ::
integer) :: [integer]
def beautiful_indices(s, a, b, k) do

end
end
```

Erlang:

```
-spec beautiful_indices(S :: unicode:unicode_binary(), A ::
unicode:unicode_binary(), B :: unicode:unicode_binary(), K :: integer()) ->
[integer()].
beautiful_indices(S, A, B, K) ->
.
```

Racket:

```
(define/contract (beautiful-indices s a b k)
(-> string? string? string? exact-integer? (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find Beautiful Indices in the Given Array II
* Difficulty: Hard
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
vector<int> beautifulIndices(string s, string a, string b, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Find Beautiful Indices in the Given Array II
* Difficulty: Hard
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public List<Integer> beautifulIndices(String s, String a, String b, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find Beautiful Indices in the Given Array II
Difficulty: Hard
Tags: array, string, hash, sort, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def beautifulIndices(self, s: str, a: str, b: str, k: int) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def beautifulIndices(self, s, a, b, k):
"""
:type s: str
:type a: str
:type b: str
:type k: int
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Find Beautiful Indices in the Given Array II
* Difficulty: Hard
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} s
* @param {string} a
* @param {string} b
* @param {number} k
* @return {number[]}
*/
var beautifulIndices = function(s, a, b, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find Beautiful Indices in the Given Array II
* Difficulty: Hard
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function beautifulIndices(s: string, a: string, b: string, k: number):
number[] {

};
```

C# Solution:

```
/*
* Problem: Find Beautiful Indices in the Given Array II
* Difficulty: Hard
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public IList<int> BeautifulIndices(string s, string a, string b, int k) {

}
}
```

C Solution:

```
/*
* Problem: Find Beautiful Indices in the Given Array II
* Difficulty: Hard
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* beautifulIndices(char* s, char* a, char* b, int k, int* returnSize) {

}
```

Go Solution:

```
// Problem: Find Beautiful Indices in the Given Array II
// Difficulty: Hard
// Tags: array, string, hash, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func beautifulIndices(s string, a string, b string, k int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun beautifulIndices(s: String, a: String, b: String, k: Int): List<Int> {

}
}
```

Swift Solution:

```
class Solution {
func beautifulIndices(_ s: String, _ a: String, _ b: String, _ k: Int) ->
[Int] {

}
}
```

Rust Solution:

```
// Problem: Find Beautiful Indices in the Given Array II
// Difficulty: Hard
// Tags: array, string, hash, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn beautiful_indices(s: String, a: String, b: String, k: i32) -> Vec<i32>
{

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {String} a
# @param {String} b
# @param {Integer} k
# @return {Integer[]}
def beautiful_indices(s, a, b, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param String $a
* @param String $b
* @param Integer $k
* @return Integer[]
*/
function beautifulIndices($s, $a, $b, $k) {

}
}
```

Dart Solution:

```
class Solution {
List<int> beautifulIndices(String s, String a, String b, int k) {

}
}
```

Scala Solution:

```
object Solution {
def beautifulIndices(s: String, a: String, b: String, k: Int): List[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec beautiful_indices(s :: String.t, a :: String.t, b :: String.t, k ::
integer) :: [integer]
def beautiful_indices(s, a, b, k) do

end
end
```

Erlang Solution:

```
-spec beautiful_indices(S :: unicode:unicode_binary(), A ::
unicode:unicode_binary(), B :: unicode:unicode_binary(), K :: integer()) ->
[integer()].
beautiful_indices(S, A, B, K) ->
.
```

Racket Solution:

```
(define/contract (beautiful-indices s a b k)
(-> string? string? string? exact-integer? (listof exact-integer?))
)
```