Problem 1202: Smallest String With Swaps

Problem Information

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

Problem Description

You are given a string

s

, and an array of pairs of indices in the string

pairs

where

pairs[i] = [a, b]

indicates 2 indices(0-indexed) of the string.

You can swap the characters at any pair of indices in the given

pairs

any number of times

.

Return the lexicographically smallest string that

s

can be changed to after using the swaps.

Example 1:

Input:

s = "dcab", pairs = [[0,3],[1,2]]

Output:

"bacd"

Explaination:

Swap s[0] and s[3], s = "bcad" Swap s[1] and s[2], s = "bacd"

Example 2:

Input:

s = "dcab", pairs = [[0,3],[1,2],[0,2]]

Output:

"abcd"

Explaination:

Swap s[0] and s[3], s = "bcad" Swap s[0] and s[2], s = "acbd" Swap s[1] and s[2], s = "abcd"

Example 3:

Input:

s = "cba", pairs = [[0,1],[1,2]]

Output:

"abc"

Explaination:

Swap s[0] and s[1], s = "bca" Swap s[1] and s[2], s = "bac" Swap s[0] and s[1], s = "abc"

Constraints:

1 <= s.length <= 10^5

0 <= pairs.length <= 10^5

0 <= pairs[i][0], pairs[i][1] < s.length

s

only contains lower case English letters.

Code Snippets

C++:

```
class Solution {
public:
string smallestStringWithSwaps(string s, vector<vector<int>>& pairs) {

}
};
```

Java:

```
class Solution {
public String smallestStringWithSwaps(String s, List<List<Integer>> pairs) {

}
}
```

Python3:

```
class Solution:
def smallestStringWithSwaps(self, s: str, pairs: List[List[int]]) -> str:
```

Python:

```
class Solution(object):
def smallestStringWithSwaps(self, s, pairs):
"""
:type s: str
:type pairs: List[List[int]]
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {number[][]} pairs
* @return {string}
*/
var smallestStringWithSwaps = function(s, pairs) {

};
```

TypeScript:

```
function smallestStringWithSwaps(s: string, pairs: number[][]): string {

};
```

C#:

```
public class Solution {
public string SmallestStringWithSwaps(string s, IList<IList<int>> pairs) {

}
}
```

C:

```
char* smallestStringWithSwaps(char* s, int** pairs, int pairsSize, int*
pairsColSize) {

}
```

Go:

```
func smallestStringWithSwaps(s string, pairs [][]int) string {

}
```

Kotlin:

```
class Solution {
fun smallestStringWithSwaps(s: String, pairs: List<List<Int>>): String {

}
}
```

Swift:

```
class Solution {
func smallestStringWithSwaps(_ s: String, _ pairs: [[Int]]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn smallest_string_with_swaps(s: String, pairs: Vec<Vec<i32>>) -> String
{

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer[][]} pairs
# @return {String}
def smallest_string_with_swaps(s, pairs)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer[][] $pairs
* @return String
*/
function smallestStringWithSwaps($s, $pairs) {

}
}
```

Dart:

```
class Solution {
String smallestStringWithSwaps(String s, List<List<int>> pairs) {

}
}
```

Scala:

```
object Solution {
def smallestStringWithSwaps(s: String, pairs: List[List[Int]]): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec smallest_string_with_swaps(s :: String.t, pairs :: [[integer]]) ::
String.t
def smallest_string_with_swaps(s, pairs) do

end
end
```

Erlang:

```
-spec smallest_string_with_swaps(S :: unicode:unicode_binary(), Pairs ::
[[integer()]]) -> unicode:unicode_binary().
smallest_string_with_swaps(S, Pairs) ->
```

```
.
```

Racket:

```
(define/contract (smallest-string-with-swaps s pairs)
(-> string? (listof (listof exact-integer?)) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Smallest String With Swaps
* Difficulty: Medium
* Tags: array, string, graph, 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:
string smallestStringWithSwaps(string s, vector<vector<int>>& pairs) {

}
};
```

Java Solution:

```
/**
* Problem: Smallest String With Swaps
* Difficulty: Medium
* Tags: array, string, graph, 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 String smallestStringWithSwaps(String s, List<List<Integer>> pairs) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest String With Swaps
Difficulty: Medium
Tags: array, string, graph, 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 smallestStringWithSwaps(self, s: str, pairs: List[List[int]]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def smallestStringWithSwaps(self, s, pairs):
"""
:type s: str
:type pairs: List[List[int]]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Smallest String With Swaps
* Difficulty: Medium
* Tags: array, string, graph, 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 {number[][]} pairs
* @return {string}
*/
var smallestStringWithSwaps = function(s, pairs) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest String With Swaps
* Difficulty: Medium
* Tags: array, string, graph, 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 smallestStringWithSwaps(s: string, pairs: number[][]): string {

};
```

C# Solution:

```
/*
* Problem: Smallest String With Swaps
* Difficulty: Medium
* Tags: array, string, graph, 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 string SmallestStringWithSwaps(string s, IList<IList<int>> pairs) {

}
}
```

C Solution:

```
/*
* Problem: Smallest String With Swaps
* Difficulty: Medium
* Tags: array, string, graph, 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
*/

char* smallestStringWithSwaps(char* s, int** pairs, int pairsSize, int*
pairsColSize) {

}
```

Go Solution:

```
// Problem: Smallest String With Swaps
// Difficulty: Medium
// Tags: array, string, graph, 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 smallestStringWithSwaps(s string, pairs [][]int) string {

}
```

Kotlin Solution:

```
class Solution {
fun smallestStringWithSwaps(s: String, pairs: List<List<Int>>): String {

}
}
```

Swift Solution:

```
class Solution {
func smallestStringWithSwaps(_ s: String, _ pairs: [[Int]]) -> String {

}
}
```

Rust Solution:

```
// Problem: Smallest String With Swaps
// Difficulty: Medium
// Tags: array, string, graph, 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 smallest_string_with_swaps(s: String, pairs: Vec<Vec<i32>>) -> String
{

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer[][]} pairs
# @return {String}
def smallest_string_with_swaps(s, pairs)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer[][] $pairs
* @return String
*/
function smallestStringWithSwaps($s, $pairs) {

}
}
```

Dart Solution:

```
class Solution {
String smallestStringWithSwaps(String s, List<List<int>> pairs) {

}
}
```

Scala Solution:

```
object Solution {
def smallestStringWithSwaps(s: String, pairs: List[List[Int]]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec smallest_string_with_swaps(s :: String.t, pairs :: [[integer]]) ::
String.t
def smallest_string_with_swaps(s, pairs) do

end
end
```

Erlang Solution:

```
-spec smallest_string_with_swaps(S :: unicode:unicode_binary(), Pairs ::
[[integer()]]) -> unicode:unicode_binary().
```

```
smallest_string_with_swaps(S, Pairs) ->
.
```

Racket Solution:

```
(define/contract (smallest-string-with-swaps s pairs)
(-> string? (listof (listof exact-integer?)) string?)
)
```