Problem 393: UTF-8 Validation

Problem Information

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

Problem Description

Given an integer array

data

representing the data, return whether it is a valid

UTF-8

encoding (i.e. it translates to a sequence of valid UTF-8 encoded characters).

A character in

UTF8

can be from

1 to 4 bytes

long, subjected to the following rules:

For a

1-byte

character, the first bit is a

0

, followed by its Unicode code.

For an

n-bytes

character, the first

n

bits are all one's, the

n + 1

bit is

0

, followed by

n - 1

bytes with the most significant

2

bits being

10

.

This is how the UTF-8 encoding would work:

Number of Bytes | UTF-8 Octet Sequence | (binary)
--------------------+--- 1 | 0xxxxxxx 2 | 110xxxxx 10xxxxxx 3 | 1110xxxx 10xxxxxx 10xxxxxx 4 | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

x

denotes a bit in the binary form of a byte that may be either

0

or

1

.

Note:

The input is an array of integers. Only the

least significant 8 bits

of each integer is used to store the data. This means each integer represents only 1 byte of data.

Example 1:

Input:

data = [197,130,1]

Output:

true

Explanation:

data represents the octet sequence: 11000101 10000010 00000001. It is a valid utf-8 encoding for a 2-bytes character followed by a 1-byte character.

Example 2:

Input:

data = [235,140,4]

Output:

false

Explanation:

data represented the octet sequence: 11101011 10001100 00000100. The first 3 bits are all one's and the 4th bit is 0 means it is a 3-bytes character. The next byte is a continuation byte which starts with 10 and that's correct. But the second continuation byte does not start with 10, so it is invalid.

Constraints:

1 <= data.length <= 2 * 10

4

0 <= data[i] <= 255

Code Snippets

C++:

```
class Solution {
public:
bool validUtf8(vector<int>& data) {

}
};
```

Java:

```
class Solution {
public boolean validUtf8(int[] data) {

}
}
```

Python3:

```
class Solution:
def validUtf8(self, data: List[int]) -> bool:
```

Python:

```
class Solution(object):
def validUtf8(self, data):
"""
:type data: List[int]
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number[]} data
* @return {boolean}
*/
var validUtf8 = function(data) {

};
```

TypeScript:

```
function validUtf8(data: number[]): boolean {

};
```

C#:

```
public class Solution {
public bool ValidUtf8(int[] data) {

}
}
```

C:

```
bool validUtf8(int* data, int dataSize) {

}
```

Go:

```
func validUtf8(data []int) bool {

}
```

Kotlin:

```
class Solution {
fun validUtf8(data: IntArray): Boolean {

}
}
```

Swift:

```
class Solution {
func validUtf8(_ data: [Int]) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn valid_utf8(data: Vec<i32>) -> bool {

}
}
```

Ruby:

```
# @param {Integer[]} data
# @return {Boolean}
def valid_utf8(data)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $data
* @return Boolean
*/
function validUtf8($data) {

}
}
```

Dart:

```
class Solution {
bool validUtf8(List<int> data) {

}
}
```

Scala:

```
object Solution {
def validUtf8(data: Array[Int]): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec valid_utf8(data :: [integer]) :: boolean
def valid_utf8(data) do

end
end
```

Erlang:

```
-spec valid_utf8(Data :: [integer()]) -> boolean().
valid_utf8(Data) ->
.
```

Racket:

```
(define/contract (valid-utf8 data)
(-> (listof exact-integer?) boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: UTF-8 Validation
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
bool validUtf8(vector<int>& data) {

}
};
```

Java Solution:

```
/**
* Problem: UTF-8 Validation
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public boolean validUtf8(int[] data) {

}
}
```

Python3 Solution:

```
"""
Problem: UTF-8 Validation
Difficulty: Medium
Tags: array

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def validUtf8(self, data: List[int]) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def validUtf8(self, data):
"""
:type data: List[int]
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: UTF-8 Validation
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
```

```
* @param {number[]} data
* @return {boolean}
*/
var validUtf8 = function(data) {

};
```

TypeScript Solution:

```
/**
* Problem: UTF-8 Validation
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function validUtf8(data: number[]): boolean {

};
```

C# Solution:

```
/*
* Problem: UTF-8 Validation
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public bool ValidUtf8(int[] data) {

}
}
```

C Solution:

```
/*
* Problem: UTF-8 Validation
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

bool validUtf8(int* data, int dataSize) {

}
```

Go Solution:

```
// Problem: UTF-8 Validation
// Difficulty: Medium
// Tags: array
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func validUtf8(data []int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun validUtf8(data: IntArray): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func validUtf8(_ data: [Int]) -> Bool {
}
}
```

Rust Solution:

```
// Problem: UTF-8 Validation
// Difficulty: Medium
// Tags: array
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn valid_utf8(data: Vec<i32>) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer[]} data
# @return {Boolean}
def valid_utf8(data)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $data
* @return Boolean
*/
function validUtf8($data) {

}
}
```

Dart Solution:

```
class Solution {
bool validUtf8(List<int> data) {

}
}
```

Scala Solution:

```
object Solution {
def validUtf8(data: Array[Int]): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec valid_utf8(data :: [integer]) :: boolean
def valid_utf8(data) do

end
end
```

Erlang Solution:

```
-spec valid_utf8(Data :: [integer()]) -> boolean().
valid_utf8(Data) ->
.
```

Racket Solution:

```
(define/contract (valid-utf8 data)
(-> (listof exact-integer?) boolean?)
)
```