Given an integer array data representing the data, return whether it is a valid **UTF-8** encoding.

A character in **UTF8** can be from **1 to 4 bytes** long, subjected to the following rules:

1. For a **1-byte** character, the first bit is a 0, followed by its Unicode code.
2. 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:

Char. number range | UTF-8 octet sequence

(hexadecimal) | (binary)

--------------------+---------------------------------------------

0000 0000-0000 007F | 0xxxxxxx

0000 0080-0000 07FF | 110xxxxx 10xxxxxx

0000 0800-0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx

0001 0000-0010 FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

**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 \* 104
* 0 <= data[i] <= 255