1271. Hexspeak

String

# **Problem Description**

Hexspeak is a whimsical representation of hexadecimal numbers where the digits '0' and '1' are replaced with the letters '0' and '1' respectively. Therefore, the valid characters in Hexspeak are only 'A', 'B', 'C', 'D', 'E', 'F', 'I', and 'O'. The task involves the following steps:

The problem provided requires converting a decimal number to a special hexadecimal representation known as "Hexspeak."

1. Take the decimal number as a string and convert it to a hexadecimal string.

3. Check if the resulting string only contains the Hexspeak characters mentioned above.

2. Change all the '0's to '0's and all the '1's to 'l's in the hexadecimal string.

- If the string after performing the conversion contains only the valid Hexspeak characters, it is considered a valid Hexspeak

representation. If any other character is present, the function should return the string "ERROR."

The intuition behind the solution is straightforward. First, we convert the given decimal number to its hexadecimal equivalent

## using the built-in functions of Python. Then we apply string manipulation techniques to interchange '0's and '1's with '0's and 'I's

as per the Hexspeak rules. We use the hex function to convert the decimal number to hexadecimal then slice the 0x prefix provided by Python's hex conversion. To ensure Hexspeak format, we convert the hexadecimal string to uppercase. Following that, we replace '0's with '0's

and '1's with 'I's. We make sure that the transformed string complies with the Hexspeak standard by comparing each character to

an approved set of characters 'ABCDEFIO'. The check involves iterating through each character of the transformed string and verifying it exists in the set of valid Hexspeak characters. If all characters are valid, the transformed string is returned. If any character doesn't match the valid set, we know that it isn't a correct Hexspeak representation, and we return "ERROR."

Solution Approach The implementation of the solution can be broken down into a few steps involving algorithms, data structures, and programming

## equivalent. Since the hex() function returns a string prefixed with "0x" representing that the number is hexadecimal, we slice

rules:

patterns:

the result to omit the "0x" part using array slicing ([2:]). String Manipulation: We carry out a series of string manipulations to convert the hexadecimal number to follow Hexspeak

Hexadecimal Conversion: We use Python's built-in hex() function, which converts a decimal number to its hexadecimal

• Replace all occurrences of '0' with '0' and '1' with 'l' using the .replace('0', '0').replace('1', 'I') method chain. Validation Against Hexspeak Specification: To ensure the conversion is valid as per Hexspeak rules, we check each character of the transformed string:

Convert to uppercase with \*upper() method to align with Hexspeak's requirement that all letters must be uppercase.

• We introduce a set s containing the allowed Hexspeak characters - ABCDEFIO.

If any character is not found in the set s, the string "ERROR" is returned.

required operations in the Python standard library.

number into a Hexspeak representation.

Conditional Output: The result of the validation checks decides the output: If all characters are in the set s (that is, they're all Hexspeak valid characters), the transformed string t is returned as the result.

combination with the all() function, which checks if all elements of the iteration meet the condition (c in s for c in t).

• We then iterate over each character in the transformed string t to confirm if it exists in the set s. This is done using a list comprehension in

- The choice of data structures and algorithms is suitable for the given task, with an emphasis on simplicity and readability. String operations and a set are well-suited for this conversion and validation task due to their efficiency and the direct support for the
- **Example Walkthrough**

Let's take an example where the decimal number is 257. We will walk through the solution approach step by step to convert this

1. Hexadecimal Conversion

First, we convert the decimal number 257 into hexadecimal. We use Python's hex() function:  $hex_number = hex(257)$ 

## To convert hex\_number to Hexspeak, we make the following string manipulations:

2. String Manipulation

hex\_number = hex\_number[2:]

So, hex\_number will now be "101".

• We convert the string to uppercase.

• We replace '0' with '0' and '1' with 'I'.

After this step, hexspeak\_number is "IOI".

characters 'A', 'B', 'C', 'D', 'E', 'F', 'I', and 'O':

s = {'A', 'B', 'C', 'D', 'E', 'F', 'I', '0'}

Performing these operations, we have:

We iterate over each character in <a href="hexspeak\_number">hexspeak\_number</a> and check if all of them are in the set <a href="hexspeak\_number">s:</a>

Since 'I' and 'O' are valid Hexspeak characters, is\_valid\_hexspeak will be True.

hexspeak\_number = hex\_number.upper().replace('0', '0').replace('1', 'I')

This returns 0x101. We slice off the 0x part to obtain just the hexadecimal representation:

Now, we need to check if <a href="hexspeak\_number">hexspeak\_number</a> contains only valid Hexspeak characters. We create a set <a href="mailto:state-allowed">state-allowed</a>

3. Validation Against Hexspeak Specification

is\_valid\_hexspeak = all(c in s for c in hexspeak\_number)

4. Conditional Output Based on the validity check, we decide the output. Since is\_valid\_hexspeak is True, we return the hexspeak\_number:

result = hexspeak\_number

if is valid hexspeak:

result = "ERROR"

Through these steps, we have successfully converted a decimal number into the Hexspeak representation following the outlined solution approach.

# convert to uppercase, replace '0' with '0', and '1' with 'I'.

# If they are, return the hexadecimal string, else return 'ERROR'

// Define a constant set containing valid HexSpeak characters.

stringstream stream; // create a stringstream object

// If the character is '0', replace with '0'

// If the character is '1', replace with 'I'

return hexString; // return the converted string

let hexString: string = parseInt(numStr).toString(16);

// If the character is '0', replace with '0'

// If the character is '1', replace with 'I'

hexString[i] = char.toUpperCase();

// Return the converted hex string in hexspeak

// Function to convert a decimal number represented as a string to "hexspeak".

// Loop over each character of the hex string to apply transformations

// If the character is a digit between '2' and '9', return "ERROR".

// Convert lowercase letters to uppercase (for the range 'a' to 'f')

characters, and checking whether all characters belong to a set of allowed characters.

hexString = hexString.join(''); // Convert array back to string

hexString = hexString.split(''); // Convert string to array for mutation

// Hexspeak is a representation of hexadecimal values where only the letters

if (hexString[i] == '0')

function toHexspeak(numStr: string): string {

let char = hexString[i];

return "ERROR";

hexString[i] = '0';

hexString[i] = 'I';

else if (char === '1') {

**if** (char === '0') {

for (let i = 0; i < hexString.length; ++i) {</pre>

if (char >= '2' && char <= '9') {</pre>

else

hexString[i] = '0';

else if (hexString[i] == '1')

hexString[i] = 'I';

string hexString = stream.str(); // get the hex string

.replace("0", "0")

.replace("1", "I");

// Check each character of the string to ensure it's valid HexSpeak.

hex\_value = hex(int(num))[2:].upper().replace('0', '0').replace('1', 'I')

if all(character in valid\_hexspeak\_characters for character in hex\_value):

# Check if all characters in the hexadecimal string are in the set of valid characters.

private static final Set<Character> VALID\_CHARS = Set.of('A', 'B', 'C', 'D', 'E', 'F', 'I', '0');

stream << hex << stol(num); // convert the string number to a long and output as hexadecimal</pre>

hexString[i] = toupper(hexString[i]); // using the standard toupper function for clarity

for (int i = 0; i < hexString.size(); ++i) { // iterate over each character</pre>

// If the character is a digit between '2' and '9', return "ERROR".

// Convert lowercase letters to uppercase (for the range 'a' to 'f')

// 'A' to 'F' and the digits '0' and '1' are allowed. '0' is replaced with '0' and '1' with 'I'.

// Convert the input string to a number in base 10, and then to a hexadecimal string

if (hexString[i] >= '2' && hexString[i] <= '9') return "ERROR";</pre>

In this case, the output result will be "IOI", which is the correct Hexspeak representation of the decimal number 257.

# Define a set of valid Hexspeak letters plus the replacements for 0 and 1. valid\_hexspeak\_characters = {'A', 'B', 'C', 'D', 'E', 'F', 'I', '0'} # Convert the input string 'num' to an integer, then to a hexadecimal string,

else:

class Solution {

**Python** 

Java

class Solution:

Solution Implementation

def toHexspeak(self, num: str) -> str:

return hex\_value

public String toHexspeak(String num) {

return 'ERROR'

```
// Convert the given number to a hexadecimal string in uppercase.
// Replace '0' with '0' and '1' with 'I' to adhere to HexSpeak rules.
String hexSpeak = Long.toHexString(Long.valueOf(num)).toUpperCase()
```

```
for (char c : hexSpeak.toCharArray()) {
            if (!VALID CHARS.contains(c)) {
                // If a character is not valid, return "ERROR".
                return "ERROR";
       // If all characters are valid, return the HexSpeak string.
        return hexSpeak;
C++
#include <sstream>
#include <string>
class Solution {
public:
    // Function to convert a decimal number represented as a string to "hexspeak".
    // Hexspeak is a representation of hexadecimal values where only the letters
    // 'A' to 'F' and the digits '0' and '1' are allowed. '0' is replaced with '0' and '1' with 'I'.
    string toHexspeak(string num) {
```

else {

return hexString;

class Solution:

**}**;

**TypeScript** 

```
def toHexspeak(self. num: str) -> str:
        # Define a set of valid Hexspeak letters plus the replacements for 0 and 1.
        valid_hexspeak_characters = {'A', 'B', 'C', 'D', 'E', 'F', 'I', '0'}
        # Convert the input string 'num' to an integer, then to a hexadecimal string,
        # convert to uppercase, replace '0' with '0', and '1' with 'I'.
        hex_value = hex(int(num))[2:].upper().replace('0', '0').replace('1', 'I')
        # Check if all characters in the hexadecimal string are in the set of valid characters.
        # If they are, return the hexadecimal string, else return 'ERROR'
        if all(character in valid_hexspeak_characters for character in hex_value):
            return hex_value
        else:
            return 'ERROR'
Time and Space Complexity
  The given Python code converts a decimal number to "hexspeak," which is a hex representation that only contains certain letters
  (A, B, C, D, E, F, I, O) and excludes all digits except for '0' and '1', which are replaced by '0' and 'I', respectively. If the converted
  hex contains characters not allowed in hexspeak, it returns 'ERROR'.
Time Complexity:
  The time complexity of the code is determined by several factors including converting the decimal to hexadecimal, replacing
```

#### of the number in bits. [2:], .upper(), .replace(), .replace(): The subsequent operations on the string occur in linear time relative to the size of

to N.

string K.

all(c in s for c in t): This list comprehension with containment check in a set has O(K) time complexity, as set containment check operations have constant average time complexity (0(1)), and it is done for all characters in the hex

hex(int(num)): Converting the decimal number to a hexadecimal string has a time complexity of O(N), where N is the length

the string. These have a complexity of O(K) collectively, where K is the length of the hexadecimal string which is proportional

- Since each of these steps is linear with respect to its input, and recognizing that K, the length of the hexadecimal representation, does not exceed O(log(N)) (the number of digits in a base b representation of a number is proportional to log\_b(N)), the overall
- time complexity of the function is dominated by the decimal to hexadecimal conversion and can be considered O(N). **Space Complexity:** The space complexity of the code includes the space used by the input, the set s, the intermediate string t, and the output.
- The space taken by the set s is a constant o(1) because it only includes a fixed number of characters. The intermediate string t holds the hexadecimal representation of the number, which is proportional to the number of digits
- in the input number's bit representation; this is O(log(N)). For the output, in the worst case (when t does not include any forbidden characters), the space complexity remains
- $O(\log(N))$ . Therefore, the total space complexity of the function is also dominated by the storage required for the hexadecimal string, resulting in an overall space complexity of O(log(N)).