1736. Latest Time by Replacing Hidden Digits

Easy Greedy String Leetcode Link

Problem Description

The given problem presents a scenario where we have a string representation of a time in the format `hh:mm`. Some of the digits in this string are unknown and are represented by the character `?`. Our task is to determine the latest possible valid time that can be formed by replacing these unknown characters with digits.

The constraints are that the time must be valid and fall within the 24-hour clock range from `00:00` to `23:59`. This means that `hh` should be between `00` and `23` and `mm` should be between `00` and `59`.

The problem requires us to find the maximum possible hour and minute combination that can be achieved by substituting the `?` with appropriate digits.

Intuition

The intuition behind the solution is to replace each `?` with the largest possible digit that will still result in a valid time. Since we want the latest valid time, we prioritize the leftmost unknown digits first, as they have a larger impact on the time value.

For the hours (`hh`):

- If the first digit is unknown (`?`), the largest value it can take depends on the second digit. If the second digit is 4 or greater, the first digit can only be `1` otherwise it could lead to an invalid time (e.g., `?4` can't be `24`, but `14` is valid). If the second digit is less than 4 or unknown, the first digit can be `2`, as we can achieve times up to `23:xx`.
- If the second digit of the hours is unknown, its value depends on the first digit. If the first digit is `2`, then the second digit can be at most `3` (making `23:xx` the latest valid hour). If the first digit is not `2`, the second digit can be `9` (making it `19:xx`, `09:xx`, etc.).

For the minutes (`mm`):

- If the third digit (minute's tens place) is unknown, it can always be a `5` as the maximum valid value for minutes' tens place is `59`.
- If the fourth digit (minute's ones place) is unknown, it can always be `9` since it's the largest valid digit for the ones place of minutes.

By following this logic, we systematically replace unknown digits with the maximum possible values that retain the time's validity. This approach ensures that the resulting time is the latest possible valid time.

Solution Approach

The solution to the problem uses a straightforward, greedy algorithm that iteratively checks each character of the input string. Since strings in Python are immutable, we first convert the time string into a list of characters, which allows us to modify individual characters directly.

The algorithm checks each position where a '?' may appear and replaces it with the highest possible digit that would still form a valid time. Here is a step-by-step breakdown of the implementation:

- **Hours (First position, `hh:??:??`):** If the first character is a '?', then we use a conditional statement to determine its value. If the second character is between '4' and '9', then the first character must be '1' since '2' followed by '4' or higher would not be a valid hour. Otherwise, if the second character is '0' to '3' or '?', the first character can be '2' to maintain the largest possible valid hour such as '20', '21', '22', or '23'.

- **Hours (Second position, `h?:??:??`):** For the second character, if it is a '?', we check the value of the first character. If the first digit is '2', the second character can at most be '3' to not exceed '23'. If the first character is '0', '1', or '?', the second character can be replaced by '9', producing the latest possible time within the valid range, like '09', '19', or '29' (where '29' would be corrected to '19' since the first digit would have been already set to '1' in the first step).

- **Minutes (Third position, `hh:??:??`):** If the third character (ten minute's place) is a '?', we can always set it to '5', since it's the largest digit that can be placed in the tens place of the minutes part of a time, giving us the latest valid minutes such as '50', '51', ... '59'.

- **Minutes (Fourth position, `hh:mm:?`):** Similarly, if the last character (one minute's place) is a '?', we replace it with '9', which is the largest possible digit for the one minute's place, while ensuring that the time remains valid.

After these conditional replacements, we join the list of characters back into a string and return the result, which represents the latest valid time that can be obtained by filling in the unknown digits.

Here is the code chunk that executes the described approach:

```python
class Solution:
    def maximumTime(self, time: str) -> str:
        t = list(time)  # Convert the string to a list for mutability.
        if t[0] == '?':
            t[0] = '1' if '4' <= t[1] <= '9' else '2'
        if t[1] == '?':
            t[1] = '3' if t[0] == '2' else '9'
        if t[3] == '?':
            t[3] = '5'
        if t[4] == '?':
            t[4] = '9'
        return ''.join(t)  # Join the list back into a string.
```

This approach assumes that the given string always has the format `hh:mm` and contains no more than two '?' characters for the hours and two '?' characters for the minutes. The conditional logic ensures that only valid times are considered, respecting the 24-hour format restrictions.

Example Walkthrough

Given the string `1?:2?`, we'll walk through how the solution approach can be applied to find the latest valid time.

1. Starting with the first character (the tens place of the hour), we see that it is already determined and is the digit `1`. Since it is not a `?`, we leave it as is.

2. The second character (the ones place of the hour) is a `?`. Since the first character is `1`, which is less than `2`, we can replace the `?` with the largest possible digit for the ones place of a valid hour, which is `9`. Now the string looks like `19:??:??`.

3. Next, we look at the third character, which is the tens place of the minutes `2`. Since this digit is already defined, there's no `?` to replace, we leave it as it is.

4. The fourth character is `?`, which represents the ones place of the minutes. Since there are no restrictions other than it being a valid digit (0-9), we replace it with the largest possible digit, `9`. Now we have `19:29`.

By following the solution approach, the final string `19:29` is the latest valid time that can be obtained by filling in the unknown digits from the original input `1?:2?`.

Python Solution

```python
class Solution:
    def maximumTime(self, time: str) -> str:
        # Convert the time string to a list of characters to modify it.
        time_chars = list(time)

        # If the first character (hour's tens place) is unknown,
        # decide its value based on the second character.
        if time_chars[0] == '?':
            # If the second character is between 4 and 9,
            # the first character can only be '1'.
            time_chars[0] = '1' if time_chars[1] >= '4' else '2'

        # If the second character (hour's ones place) is unknown,
        # decide its value based on the first character.
        if time_chars[1] == '?':
            # If the first character is '2', the latest valid
            # value for the second character is '3'.
            # Otherwise, it can go up to '9'.
            time_chars[1] = '3' if time_chars[0] == '2' else '9'

        # If the fourth character (minute's tens place) is unknown,
        # it can be at most '5' to represent valid minutes.
        if time_chars[3] == '?':
            time_chars[3] = '5'

        # If the fifth character (minute's ones place) is unknown,
        # it can be at most '9' to complete the time.
        if time_chars[4] == '?':
            time_chars[4] = '9'

        # Join the list of characters back to a string and return.
        return ''.join(time_chars)

```

Java Solution

```java
class Solution {

    // Method to find the maximum possible valid time by replacing '?' with digits
    public String maximumTime(String time) {
        //Convert the input string to a character array for easy manipulation
        char[] timeChars = time.toCharArray();

        // If the first character (hour's tens place) is '?'
        if (timeChars[0] == '?') {
            // If the second character (hour's ones place) is between 4 and 9, set first character to '1'
            // Otherwise, it's safe to set the first character to '2' as it can be 20-23 hours
            timeChars[0] = (timeChars[1] >= '4' && timeChars[1] <= '9') ? '1' : '2';
        }

        // If the second character (hour's ones place) is '?'
        if (timeChars[1] == '?') {
            // If the first character (hour's tens place) is '2', it can only be '0', '1', or '2'
            // so set second character to '3' (latest valid time), otherwise set to '9'
            timeChars[1] = (timeChars[0] == '2') ? '3' : '9';
        }

        // If the fourth character (minute's tens place) is '?', set to '5' (latest valid minute tens place)
        if (timeChars[3] == '?') {
            timeChars[3] = '5';
        }

        // If the fifth character (minute's ones place) is '?', set to '9' (latest valid minute ones place)
        if (timeChars[4] == '?') {
            timeChars[4] = '9';
        }

        // Return the new string constructed from the character array
        return new String(timeChars);
    }
}

```

C++ Solution

```cpp
class Solution {
public:
    // Function that takes a time string with unknown digits represented by '?'
    // and returns the latest valid time that can be formed by replacing the unknown digits
    string maximumTime(string time) {
        // If the first character (hours tens place) is unknown
        if (time[0] == '?') {
            // Set it to '1' if the second character is between '4' and '9' (i.e., 14-19h),
            // otherwise set it to '2' for the 20-23h range
            time[0] = (time[1] >= '4' && time[1] <= '9') ? '1' : '2';
        }

        // If the second character (hours units place) is unknown
        if (time[1] == '?') {
            // Set it to '3' if the first character is '2' (i.e., 23h),
            // otherwise set it to '9' (to make the largest possible hour in the 0-9h or 10-19h range)
            time[1] = (time[0] == '2') ? '3' : '9';
        }

        // If the third character (minutes tens place) is unknown, set it to '5'
        // because the largest digit that can occur in the tens place for minutes is '5'
        if (time[3] == '?') {
            time[3] = '5';
        }

        // If the fourth character (minutes units place) is unknown, set it to '9'
        // to form the maximum possible minutes
        if (time[4] == '?') {
            time[4] = '9';
        }

        // Return the modified time string with no '?' remaining,
        // representing the latest valid time
        return time;
    }
};

```

Typescript Solution

```typescript
/**
 * Replaces question marks in a time string with the maximum possible valid time values.
 * @param {string} time - A time string in the format "hh:mm", where "?" is used to represent unknown digits.
 * @returns {string} - The time string with unknown digits replaced by the maximum valid time values.
 */
function maximumTime(time: string): string {

    // Convert the time string into an array of characters to manipulate individual parts.
    const timeChars: string[] = Array.from(time);

    // Replace the first '?' in the hour part with the maximum possible digit.
    if (timeChars[0] === '?') {
        // If the second digit is between 4 and 9, the hour can only start with '1'.
        timeChars[0] = (timeChars[1] >= '4' && timeChars[1] <= '9') ? '1' : '2';
    }

    // Replace the second '?' in the hour part with the maximum possible digit.
    if (timeChars[1] === '?') {
        // If the first digit is '2', the second digit can only be at most '3'.
        timeChars[1] = (timeChars[0] === '2') ? '3' : '9';
    }

    // Replace the first '?' in the minutes part with '5', as it's the maximum allowed digit in that position.
    if (timeChars[3] === '?') {
        timeChars[3] = '5';
    }

    // Replace the second '?' in the minutes part with '9', as it's the maximum allowed digit in that position.
    if (timeChars[4] === '?') {
        timeChars[4] = '9';
    }

    // Join the array of characters back into a string and return the result.
    return timeChars.join('');
}

// Usage example:
// const result = maximumTime("2?:?0");
// console.log(result); // Outputs: "23:50"

```

Time and Space Complexity

The time complexity of the provided code is `O(1)` since the input `time` is always a string of a fixed length, representing the time in the format HH:MM. The operations performed include indexing, comparisons, and value assignments, all of which are done in constant time.

The space complexity of the code is also `O(1)`. The input is converted to a list `t`, but since the length of the input is fixed to 5 characters (including the separator ':' character), this also takes up a constant amount of additional space. Thus, the amount of memory used does not scale with the size of the input.