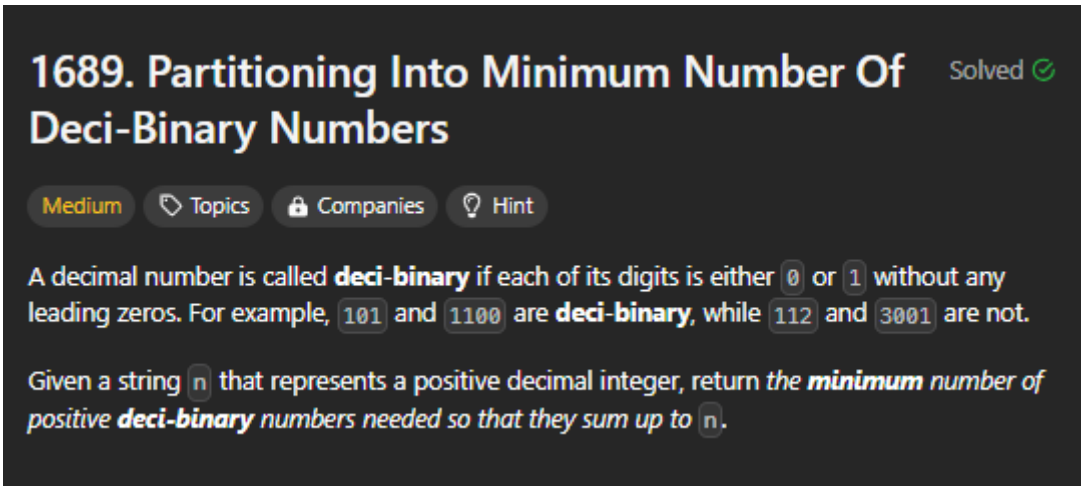
“1689. Partitioning Into Minimum Number Of Deci-Binary Numbers”

Question :



A decimal number is called deci-binary if each of its digits is either 0 or 1 without any leading zeros. For example, 101 and 1100 are deci-binary, while 112 and 3001 are not.

Given a string n that represents a positive decimal integer, return the minimum number of positive deci-binary numbers needed so that they sum up to n.

Constraints :

- 1 ≤ n.length ≤ n.length ≤ 105
- n consists of only digits.
- n does not contain any leading zeros and represents a positive integer.

Inputs :

- A string n representing a positive decimal integer.

Outputs :

- An integer representing the minimum number of deci-binary numbers required.

Example 1 :

Input:
n = "32"

Output:
3

Explanation:
The number 32 can be formed by summing 10 + 11 + 11.

Example 2 :

Input:
n = "82734"

Output:
8

Example 3 :

Input:
n = "27346209830709182346"

Output:
9

Algorithm :

1. Initialize a variable max_number to '0'.
2. Iterate through each character of the string n.
3. For every character, compare it with max_number. If it is greater, update max_number.
4. At the end of the loop, convert max_number to an integer and return it.

5. This works because the minimum number of deci-binary numbers needed is determined by the largest digit in the string.

Code :

```
int minPartitions(char* n) {
    char max_number = '0';
    for (int i = 0; n[i] != '\0'; i++) {
        if (max_number < n[i]) {
            max_number = n[i];
        }
    }
    return max_number - '0';
}
```

Time Complexity

- O(n): The algorithm iterates through the string n once, where n is the length of the string.