Recursive Digit Sum Hackerrank Solution

Recursive digit sum HackerRank solution is a problem that challenges participants to understand the concept of recursion while effectively manipulating numbers. The problem is part of the HackerRank platform, which offers a wide array of coding challenges to enhance programming skills. In this article, we will explore the recursive digit sum problem, provide an in-depth explanation of the solution, and discuss the implementation in various programming languages.

Understanding the Problem Statement

The recursive digit sum problem requires you to compute a single-digit number derived from a given integer. The process involves repeatedly summing the digits of the number until you arrive at a single-digit result. The challenge becomes more interesting when you consider large numbers, as they may require multiple iterations of summation.

Problem Definition

The task can be defined as follows:

1. You are given an integer, `n`, and a radix, `k`.
2. You will first multiply `n` by `k` and then recursively sum the digits of the result.
3. The process continues until a single-digit number is obtained.

For example, if `n = 9875` and `k = 4`, the first step would be to calculate `9875 4 = 39500`. The next step is to sum the digits of `39500`, which yields `3 + 9 + 5 + 0 + 0 = 17`. Since `17` is not a single digit, we sum the digits of `17`, resulting in `1 + 7 = 8`. Therefore, the final result for this example would be `8`.

Steps to Solve the Problem

To effectively solve the recursive digit sum problem, follow these steps:

1. **Read the inputs:** Read the values of `n` and `k`.
2. **Calculate the initial product:** Multiply `n` by `k` to obtain a new number, `product`.
3. **Implement a recursive function:** Create a function that sums the digits of a number and checks if the result is a single digit.
4. **Return the result:** Once a single-digit result is achieved, return it.

Recursive Function Explanation

The recursive function is crucial in this problem. It works as follows:

- Base Case: If the number is a single digit, return that number.
- Recursive Case: If the number is greater than 9, convert the number to a string, split it into its digits, convert them back to integers, and sum them. Then call the function recursively with this sum until a single digit is reached.

Pseudocode for the Solution

Here’s a simple pseudocode representation of the recursive digit sum solution:

```
function recursive_digit_sum(n, k):
    product = n k
    return digit_sum(product)

function digit_sum(num):
    if num < 10:
        return num
    else:
        sum = 0
        while num > 0:
            sum += num % 10
            num = num // 10
        return digit_sum(sum)
```

Implementing the Solution in Different Languages

Below, we provide implementations of the recursive digit sum solution in Python and Java.

Python Implementation

```python
def recursive_digit_sum(n, k):
product = n k
return digit_sum(product)

def digit_sum(num):
if num < 10:
return num
else:
total = 0
while num > 0:
total += num % 10
num //= 10
return digit_sum(total)

Example usage
n = 9875
k = 4
result = recursive_digit_sum(n, k)
print(result) Output: 8
```

Java Implementation

```java
public class RecursiveDigitSum {
public static void main(String[] args) {
int n = 9875;
int k = 4;
int result = recursiveDigitSum(n, k);
System.out.println(result); // Output: 8
}

public static int recursiveDigitSum(int n, int k) {
int product = n k;
return digitSum(product);
}

public static int digitSum(int num) {
if (num < 10) {
```

```
return num;
} else {
int total = 0;
while (num > 0) {
total += num % 10;
num /= 10;
}
return digitSum(total);
}
}
}
```

Complexity Analysis

Understanding the complexity of the recursive digit sum solution is essential for evaluating its efficiency.

Time Complexity

The time complexity can be analyzed as follows:

- The digit summation takes O(d) time, where d is the number of digits in the number.
- Since the number of digits reduces logarithmically with each recursive call, the overall time complexity can be approximated to O(log n) for each digit summation until a single digit is reached.

Space Complexity

The space complexity is primarily due to the recursion stack, which is O(log n) due to the reduction in the number of digits with each recursive call.

Conclusion

The **recursive digit sum HackerRank solution** provides an excellent exercise in recursion and number manipulation. By following the steps outlined in this article, understanding the problem, and implementing it in various programming languages, you can enhance your coding skills and prepare for similar challenges in the future. Practicing with such problems on platforms like HackerRank not only sharpens your algorithmic thinking but also improves your problem-solving capabilities in real-world applications.

Frequently Asked Questions

What is the recursive digit sum problem in HackerRank?

The recursive digit sum problem requires calculating the repeated sum of digits of a number until a single digit is obtained, using a recursive approach.

How do you approach solving the recursive digit sum problem?

To solve the recursive digit sum problem, repeatedly sum the digits of the number until the result is a single digit. This can be done using a recursive function or an iterative approach.

What is an efficient way to implement the recursive digit sum in Python?

You can create a recursive function that takes an integer, converts it to a string to sum its digits, and calls itself with the new sum until a single digit is achieved.

What are the input constraints for the recursive digit sum problem on HackerRank?

The input typically consists of two integers, n (the number to compute the sum for) and k (the number of times to concatenate n). The constraints can vary, but n can be very large, requiring careful handling.

Can you explain how to handle large numbers in the recursive digit sum problem?

For large numbers, you can compute the digit sum using modular arithmetic to avoid overflow and ensure efficient calculations without needing to handle the entire number as a string.

What is the expected output of the recursive digit sum function?

The expected output is a single digit integer that represents the final recursive digit sum of the given number after processing.

How does the recursive digit sum relate to digital roots?

The recursive digit sum is essentially the same as finding the digital root of a number, which can be calculated using the formula (n - 1) % 9 + 1 for n > 0.

What are common pitfalls to avoid when implementing the recursive digit sum?

Common pitfalls include not handling large numbers correctly, incorrect base cases in recursion, and failing to account for edge cases like zero or negative numbers.

Recursive Digit Sum Hackerrank Solution

git clone --recursive □ git clone --recurse-submodules□□□□
Jun 8, 2015 · □□□□□CSDN□□□□git clone --recursive □ git clone --recurse-submodules□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

linux□□codeblock □□□□make: *** [all-recursive] □□ 1
Apr 9, 2012 · □□□□□CSDN□□□□linux□□codeblock □□□□make: *** [all-recursive] □□ 1□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

makefile□□□process_begin: CreateProcess failed-CSDN□□
Oct 1, 2008 · □□□□□CSDN□□□□makefile□□□process_begin: CreateProcess failed□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

□□□□□□Leaving directory□□□ - CSDN□□
Jul 7, 2010 · □□□□□CSDN□□□□□□□□□□Leaving directory□□□□□□□□□□□□□□□□□□□Linux/Unix□□□□□□□□□□□□□ ...

□□gcc-4.7.1□□□□□□□□□□□make□□-CSDN□□
Sep 23, 2012 · □□□□□CSDN□□□□□□gcc-4.7.1□□□□□□□□□□□make□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

oralce□□ ORA-00604: error occurred at recursive SQL level 1
Sep 16, 2013 · □□□□□CSDN□□□□oralce□□ ORA-00604: error occurred at recursive SQL level 1□□□□□□□□□□□□□□Oracle □□□□□□ ...

std :: experimental :: filesystem :: recursive_directory_iterator□□□ ...
□□□□□CSDN□□□□std :: experimental :: filesystem :: recursive_directory_iterator□□ ...

MySQL With Recursive□□□ - CSDN□□
Apr 16, 2022 · □□□□□CSDN□□□□MySQL With Recursive□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

make:* [install -recursive]Error 1□□□□□□□-CSDN□□**
Feb 22, 2011 · □□□□□CSDN□□□□make:*** [install -recursive]Error 1□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ ...

k210□□□□□ - CSDN□□
Jul 29, 2020 · □□□□□CSDN□□□□k210□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

git clone --recursive □ git clone --recurse-submodules□□□□
Jun 8, 2015 · □□□□□CSDN□□□□git clone --recursive □ git clone --recurse-submodules□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

linux□□codeblock □□□□make: * [all-recursive] □□ 1**
Apr 9, 2012 · □□□□□CSDN□□□□linux□□codeblock □□□□make: *** [all-recursive] □□ 1□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

makefile□□□process_begin: CreateProcess failed-CSDN□□
Oct 1, 2008 · □□□□□CSDN□□□□makefile□□□process_begin: CreateProcess failed□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

□□□□□□Leaving directory□□□ - CSDN□□
Jul 7, 2010 · □□□□□CSDN□□□□□□□□□Leaving directory□□□□□□□□□□□□□□□□□□Linux/Unix□□□□□□□□□□□□□ ...

[□□gcc-4.7.1□□□□□□□□□□make□□-CSDN□□]
Sep 23, 2012 · □□□□□CSDN□□□□□□gcc-4.7.1□□□□□□□□□□make□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

oralce□□ ORA-00604: error occurred at recursive SQL level 1
Sep 16, 2013 · □□□□□CSDN□□□□oralce□□ ORA-00604: error occurred at recursive SQL level 1□□□□□□□□□□□□□□Oracle □□□□□□ ...

std :: experimental :: filesystem :: recursive_directory_iterator□□□ ...
□□□□□CSDN□□□□std :: experimental :: filesystem :: recursive_directory_iterator□□□CSDN□□□

MySQL With Recursive□□□ - CSDN□□
Apr 16, 2022 · □□□□□CSDN□□□□MySQL With Recursive□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

make:* [install -recursive]Error 1□□□□□□□-CSDN□□**
Feb 22, 2011 · □□□□□CSDN□□□□make:*** [install -recursive]Error 1□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ ...

k210□□□□□□ - CSDN□□
Jul 29, 2020 · □□□□□CSDN□□□□k210□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□CSDN□□□

Master the recursive digit sum problem with our comprehensive HackerRank solution. Discover how to efficiently solve challenges and enhance your coding skills!

Back to Home