

Problem 3723: Maximize Sum of Squares of Digits

Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given two

positive

integers

num

and

sum

.

A positive integer

n

is

good

if it satisfies both of the following:

The number of digits in

n

is

exactly

num

.

The sum of digits in

n

is

exactly

sum

.

The

score

of a

good

integer

n

is the sum of the squares of digits in

n

.

Return a

string

denoting the

good

integer

n

that achieves the

maximum

score

. If there are multiple possible integers, return the

maximum

one. If no such integer exists, return an empty string.

Example 1:

Input:

num = 2, sum = 3

Output:

"30"

Explanation:

There are 3 good integers: 12, 21, and 30.

The score of 12 is

1

2

+ 2

2

= 5

.

The score of 21 is

2

2

+ 1

2

= 5

.

The score of 30 is

3

2

+ 0

2

= 9

.

The maximum score is 9, which is achieved by the good integer 30. Therefore, the answer is

"30"

.

Example 2:

Input:

num = 2, sum = 17

Output:

"98"

Explanation:

There are 2 good integers: 89 and 98.

The score of 89 is

8

2

+ 9

2

= 145

.

The score of 98 is

9

2

+ 8

2

= 145

.

The maximum score is 145. The maximum good integer that achieves this score is 98.
Therefore, the answer is

"98"

.

Example 3:

Input:

num = 1, sum = 10

Output:

""

Explanation:

There are no integers that have exactly 1 digit and whose digits sum to 10. Therefore, the
answer is

""

.

Constraints:

1 <= num <= 2 * 10

5

1 <= sum <= 2 * 10

6

Code Snippets

C++:

```
class Solution {  
public:  
    string maxSumOfSquares(int num, int sum) {  
  
    }  
};
```

Java:

```
class Solution {  
    public String maxSumOfSquares(int num, int sum) {  
  
    }  
}
```

Python3:

```
class Solution:  
    def maxSumOfSquares(self, num: int, sum: int) -> str:
```

Python:

```
class Solution(object):  
    def maxSumOfSquares(self, num, sum):  
        """  
        :type num: int  
        :type sum: int  
        :rtype: str
```

```
"""
```

JavaScript:

```
/**
 * @param {number} num
 * @param {number} sum
 * @return {string}
 */
var maxSumOfSquares = function(num, sum) {

};
```

TypeScript:

```
function maxSumOfSquares(num: number, sum: number): string {

};
```

C#:

```
public class Solution {
    public string MaxSumOfSquares(int num, int sum) {

    }
}
```

C:

```
char* maxSumOfSquares(int num, int sum) {

}
```

Go:

```
func maxSumOfSquares(num int, sum int) string {

}
```

Kotlin:

```
class Solution {  
    fun maxSumOfSquares(num: Int, sum: Int): String {  
  
    }  
}
```

Swift:

```
class Solution {  
    func maxSumOfSquares(_ num: Int, _ sum: Int) -> String {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn max_sum_of_squares(num: i32, sum: i32) -> String {  
  
    }  
}
```

Ruby:

```
# @param {Integer} num  
# @param {Integer} sum  
# @return {String}  
def max_sum_of_squares(num, sum)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer $num  
     * @param Integer $sum  
     * @return String  
     */  
    function maxSumOfSquares($num, $sum) {  
  
    }  
}
```

```
}
```

Dart:

```
class Solution {  
  String maxSumOfSquares(int num, int sum) {  
  
  }  
}
```

Scala:

```
object Solution {  
  def maxSumOfSquares(num: Int, sum: Int): String = {  
  
  }  
}
```

Elixir:

```
defmodule Solution do  
  @spec max_sum_of_squares(num :: integer, sum :: integer) :: String.t  
  def max_sum_of_squares(num, sum) do  
  
  end  
end
```

Erlang:

```
-spec max_sum_of_squares(Num :: integer(), Sum :: integer()) ->  
  unicode:unicode_binary().  
max_sum_of_squares(Num, Sum) ->  
  .
```

Racket:

```
(define/contract (max-sum-of-squares num sum)  
  (-> exact-integer? exact-integer? string?)  
  )
```

Solutions

C++ Solution:

```
/*
 * Problem: Maximize Sum of Squares of Digits
 * Difficulty: Medium
 * Tags: string, greedy, math
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public:
    string maxSumOfSquares(int num, int sum) {

    }
};
```

Java Solution:

```
/**
 * Problem: Maximize Sum of Squares of Digits
 * Difficulty: Medium
 * Tags: string, greedy, math
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
    public String maxSumOfSquares(int num, int sum) {

    }
}
```

Python3 Solution:

```
"""
Problem: Maximize Sum of Squares of Digits
```

Difficulty: Medium

Tags: string, greedy, math

Approach: String manipulation with hash map or two pointers

Time Complexity: $O(n)$ or $O(n \log n)$

Space Complexity: $O(1)$ to $O(n)$ depending on approach

"""

```
class Solution:
```

```
def maxSumOfSquares(self, num: int, sum: int) -> str:
```

```
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
```

```
def maxSumOfSquares(self, num, sum):
```

```
"""
```

```
:type num: int
```

```
:type sum: int
```

```
:rtype: str
```

```
"""
```

JavaScript Solution:

```
/**
```

```
 * Problem: Maximize Sum of Squares of Digits
```

```
 * Difficulty: Medium
```

```
 * Tags: string, greedy, math
```

```
 *
```

```
 * Approach: String manipulation with hash map or two pointers
```

```
 * Time Complexity:  $O(n)$  or  $O(n \log n)$ 
```

```
 * Space Complexity:  $O(1)$  to  $O(n)$  depending on approach
```

```
 */
```

```
/**
```

```
 * @param {number} num
```

```
 * @param {number} sum
```

```
 * @return {string}
```

```
 */
```

```
var maxSumOfSquares = function(num, sum) {
```

```
};
```

TypeScript Solution:

```
/**
 * Problem: Maximize Sum of Squares of Digits
 * Difficulty: Medium
 * Tags: string, greedy, math
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

function maxSumOfSquares(num: number, sum: number): string {

};
```

C# Solution:

```
/*
 * Problem: Maximize Sum of Squares of Digits
 * Difficulty: Medium
 * Tags: string, greedy, math
 *
 * Approach: String manipulation with hash map or two pointers
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

public class Solution {
    public string MaxSumOfSquares(int num, int sum) {

    }
}
```

C Solution:

```
/*
 * Problem: Maximize Sum of Squares of Digits
```

```

* Difficulty: Medium
* Tags: string, greedy, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* maxSumOfSquares(int num, int sum) {

}

```

Go Solution:

```

// Problem: Maximize Sum of Squares of Digits
// Difficulty: Medium
// Tags: string, greedy, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func maxSumOfSquares(num int, sum int) string {

}

```

Kotlin Solution:

```

class Solution {
    fun maxSumOfSquares(num: Int, sum: Int): String {

    }
}

```

Swift Solution:

```

class Solution {
    func maxSumOfSquares(_ num: Int, _ sum: Int) -> String {

    }
}

```

Rust Solution:

```
// Problem: Maximize Sum of Squares of Digits
// Difficulty: Medium
// Tags: string, greedy, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
    pub fn max_sum_of_squares(num: i32, sum: i32) -> String {

    }
}
```

Ruby Solution:

```
# @param {Integer} num
# @param {Integer} sum
# @return {String}
def max_sum_of_squares(num, sum)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer $num
     * @param Integer $sum
     * @return String
     */
    function maxSumOfSquares($num, $sum) {

    }

}
```

Dart Solution:

```

class Solution {
  String maxSumOfSquares(int num, int sum) {

  }
}

```

Scala Solution:

```

object Solution {
  def maxSumOfSquares(num: Int, sum: Int): String = {

  }
}

```

Elixir Solution:

```

defmodule Solution do
  @spec max_sum_of_squares(num :: integer, sum :: integer) :: String.t
  def max_sum_of_squares(num, sum) do

  end
end

```

Erlang Solution:

```

-spec max_sum_of_squares(Num :: integer(), Sum :: integer()) ->
  unicode:unicode_binary().
max_sum_of_squares(Num, Sum) ->
  .

```

Racket Solution:

```

(define/contract (max-sum-of-squares num sum)
  (-> exact-integer? exact-integer? string?)
  )

```