SP 2023 recursion

Question 1

Problem statement

In a hidden chamber deep within an ancient castle, you stumble upon a peculiar treasure chest. This isn't just any chest; it's known as the "Square Root Stash." Legend has it that it's filled with precious gems and artifacts from centuries past, but it's protected by a magical lock unlike any other.

The Square Root Stash lock doesn't require a traditional key. Instead, it demands unique "Square Gems" to be placed in precise order. Square Gems are special stones carved into perfect squares - numbers like 1, 4, 9, and 16. The arrangement of these Square Gems must sum up to a secret number, denoted as `n`, for the lock to release its riches.

But here's where it gets intriguing - the lock doesn't just require any combination of Square Gems; it demands the most efficient one! You must use the fewest possible Square Gems to unlock the stash. Each Square Gem is enchanted with ancient powers, so using more of them than necessary would disrupt the balance and trigger protective spells, sealing the chest forever.

So, your quest begins. Armed with your knowledge of perfect squares, you need to figure out the optimal combination of Square Gems to unlock the Square Root Stash for any given `n`. For example, if the secret number `n` is 12, you must determine the minimum number of Square Gems required and which ones to use. After careful calculations, you discover that you need exactly 3 Square Gems, and these are the numbers 4, 4, and 4. Placing them in the lock, you hear a satisfying click, and the chest slowly creaks open, revealing the treasures within.

As you venture deeper into the castle, you find more chambers and more Square Root Stashes, each with its unique secret number `n`. Your challenge is to unlock them all using the fewest Square Gems possible, decoding the mysteries of the past one treasure at a time.

Class Definition

Class: `Question1`

Method: `minSquareGems`

The class `Question1` includes the method `minSquareGems` which accepts one `int` parameter and returns an `int`.

```
//java
public int minSquareGems(int n)

//cpp
public int minSquareGems(int n)

#python
def minSquareGems(n: int) -> int
```

Constraints

- The value `n` ranges between `1` and `100000 (10^5)`, inclusive.

Examples

Example 0

```
Question1 q1 = new Question1();
System.out.println(q1.minSquareGems(12)); // Returns: 3
```

We can use 3 square keys that has value `4` to open the chest.

Example 1

```
Question1 q1 = new Question1();
System.out.println(q1.minSquareGems(13)); // Returns: 2
```

We can use square keys `4` and `9`, to open the chest.

Example 2

```
Question1 q1 = new Question1();
System.out.println(q1.minSquareGems(9999)); // Returns: 4
```

We can use square keys `9801`, `196` and 2 square keys that has value `1`, to open the chest.

Test Cases

Input	Expected Output
12	3
13	2
9999	4
169	1
135	4
30	
28	
30000	
70000	
100000	
125	
79	
50	
75	
785	
9800	

Please note that the expected outputs for some test cases are not provided.

Question 2

Problem Statement

In a quaint little workshop at the heart of a magical kingdom, you discover an ancient notepad. This notepad, however, isn't your ordinary stationery. It's a creation of the great wizard Archibald, renowned for his intriguing and puzzling inventions.

Upon opening the notepad, you notice that there's only one character written on it, a bold "**A**". But this isn't just any "A"; it's enchanted with the power of replication. You've been bestowed with two mystical operations to interact with this remarkable "A":

Operation 1 - Copy All: With a simple touch, you can copy all the characters on the notepad, but remember, partial copies are strictly forbidden. It's an all-or-nothing kind of magic.

Operation 2 - Paste: This operation allows you to paste the characters you copied during the last "Copy All" operation. It's like a glimpse into the past.

Now, here's the twist: you're on a quest to recreate the sacred symbol "A" a specific number of times, determined by an integer `n`. Your mission is to find the shortest sequence of operations that will conjure "A" exactly `n` times on the notepad.

You understand that every operation counts, so you must use your powers of magic and deduction to minimise them. Armed with this ancient notepad and your wits, you embark on a journey through enchanted forests, mystical caves, and enigmatic dungeons, uncovering hidden secrets and solving riddles that will test your puzzle-solving prowess. Your goal? To master these arcane operations and achieve the impossible - creating 'A' the exact number of times your quest demands.

Class Definition

Class: `Question2`

Method: `minOperationsCount`

The class `Question2` includes the method `minOperationsCount` which accepts one `int` parameter and returns an `int`.

```
//java
public int minOperationsCount(int n)
```

```
//cpp
public int minOperationsCount(int n)

#python
def minOperationsCount(n: int) -> int
```

Constraints

- The value `n` ranges between `1` and `10000 (10^4)`, inclusive.

Examples

Example 0

```
Question2 q2 = new Question2();
System.out.println(q2.minOperationsCount(3)); // Returns: 3
```

Initially, we have one character 'A'.

In step 1, we use Copy All operation.

In step 2, we use Paste operation to get 'AA'.

In step 3, we use Paste operation to get 'AAA'.

Example 1

```
Question2 q2 = new Question2();
System.out.println(q2.minOperationsCount(1)); // Returns: 0
```

The "A" character is already there, no need to do anything else.

Example 2

```
Question2 q2 = new Question2();
System.out.println(q2.minOperationsCount(6)); // Returns: 5
```

Initially, we have one character ‘A’.

In step 1, we use Copy All operation.

In step 2, we use Paste operation to get ‘AA’.

In step 3, we use Copy All operation.

In step 4, we use Paste operation to get ‘AAAA’.

In step 5, we use Paste operation to get ‘AAAAAA’.

Test Cases

Input	Expected Output
3	3
1	0
6	5
15	8
20	9
100	
27	
98	
932	
728	
302	
37	
1000	
10000	

Input	Expected Output
3003	
3204	

Please note that the expected outputs for some test cases are not provided.