프로그래머스 자바스크립트• k의 개수

태그

문제 설명
1부터 13까지의 수에서, 1은 1, 10, 11, 12, 13 이렇게 총 6번 등장합니다. 정수 i, j, k가 매개변수로 주어질 때, i부터 j까지 k가 몇 번 등장하는지 return 하도록 solution 함수를 완성해주세요.

제한사항
1 ≤ i < j ≤ 100,000
0 ≤ k ≤ 9

입출력 예

10부터 50까지 5는 15, 25, 35, 45, 50 총 5번 등장합니다. 따라서 5를 return 합니다.
입출력 예

3부터 10까지 2는 한 번도 등장하지 않으므로 0을 return 합니다.

주로 숫자의 자릿수를 따로 처리할 때 사용됩니다. 예를 들어, 숫자 123의 각 자릿수를 출력할 수 있는 코드는 다음과 같습니다.

```
const num = 123;
const digits = num.toString().split('');
for (const digit of digits) {
  console.log(digit);
}
```

```
function solution(i, j, k) {
  // Initialize a counter
  let count = 0;

  // Iterate through the numbers from i to j
  for (let num = i; num <= j; num++) {
   // 숫자를 문자열로 변환하고 문자 배열로 분할
    const charArray = num.toString().split('');

   // filter() 메서드를 사용하여 배열에 k가 나타나는 횟수를 계산합니다.
    count += charArray.filter(char => char == k).length;
  }

  // Return the count
  return count;
}
```

```
class Solution {
    public int solution(int i, int j, int k) {
        String str = "";
        for(int a = i; a <= j; a++) {
            str += a+"";
        }

        return str.length() - str.replace(k+"", "").length();
    }
}
```

```
class Solution {
    public int solution(int i, int j, int k) {
        int answer = 0;

        for (int num = i; num <= j; num++){
            int tmp = num;
            while (tmp != 0){
                if (tmp % 10 == k)
                    answer++;
                tmp /= 10;
            }
        }
        return answer;
    }
```

```
str.length()는 문자열 str의 길이를 반환합니다. 예를 들어, "abc".length()는 3을 반환합니다.
str.replace(k+"", "")는 문자열 str에서 문자열 k가 있는 부분을 찾아서 지웁니다. 예를 들어, "abcabc".
replace("a", "")는 "bcbc"를 반환합니다.
그래서 str.length() - str.replace(k+"", "").length()는 문자열 str에서 문자열 k가 들어간 횟수를
 세는 것과 같습니다. 예를 들어, "abcabc".length() - "abcabc".replace("a", "").length()는 2가
 됩니다.
```

```
이 문제를 해결하기 위해 i에서 j까지의 숫자를 반복하고 k가 몇 번 나타나는지 계산할 수 있습니다. 각 숫자를 문
자열로 변환하고 문자열에 k가 있는지 확인하기 위해 includes() 메서드를 사용하여 이를 수행할 수 있습니다. 이
것은 주어진 예제에 대한 예상 결과를 반환해야 합니다.

참고: 이 솔루션에서 toString() 메서드는 숫자를 문자열로 변환하는 데 사용되며, includes() 메서드는 문자
열에 지정된 문자가 포함되어 있는지 확인하는 데 사용됩니다. 이 솔루션은 루프를 사용하여 i에서 j까지 숫자를 반
복하고 숫자에서 k가 발견될 때마다 카운터를 증가시킵니다. 마지막에는 i에서 j까지의 범위에 k가 나타나는 횟수만
큼 카운터를 반환합니다. 이 솔루션의 시간 복잡도는 O(n)입니다. 여기서 n은 i에서 j까지의 범위에 있는 숫자의
 수입니다. 이것은 i와 j의 작은 값에 대해서는 잘 수행되지만 큰 값에 대해서는 느려질 수 있음을 의미합니다.
```

```
 import java.util.stream.Collectors;

class Solution {
    public String solution(String my_string) {
```

```
        return my_string.chars()
                .mapToObj(Character::toString)
                .distinct()
                .collect(Collectors.joining());
    }
}
```

import java.util.stream.Collectors;는 Java 언어에서 스트림을 처리할 수 있는 유틸리티 클래스인 Collectors를 가져오는 구문, 스트림은 컬렉션(collection)의 요소들을 순차적으로 처리할 수 있는 개념입니다.
스트림을 이용하면 컬렉션의 요소들을 필터링하거나 정렬하거나 집계하거나 요소들의 값을 수정하는 등의 작업을 간단하게 처리할 수 있습니다.

스트림의 요소들 중에서 짝수만 골라서 합계를 구하는 코드

```
import java.util.List;
import java.util.stream.Collectors;

List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

int sum = numbers.stream()
                 .filter(n -> n % 2 == 0)
                 .collect(Collectors.summingInt(n -> n));

System.out.println(sum);  // 출력: 30
```

이 스트림에 Collectors.summingInt() 메서드를 적용하면 스트림의 요소들의 합계가 계산

```
import java.util.*;
class Solution {
    public String solution(String my_string) {
        String[] answer = my_string.split("");
        Set<String> set = new LinkedHashSet<String>(Arrays.asList(answer));

        return String.join("", set);
    }
}
```

Set<String> set = new LinkedHashSet<String>(Arrays.asList(answer));는 Java 언어에서 중복을 허용하지 않는 집합(set)을 생성하는 코드입니다.
Java 언어에서 집합은 중복을 허용하지 않고, 순서가 유지되지 않는 자료구조입니다. 집합은 Set 인터페이스를 구현한 객체로 생성할 수 있습니다.

LinkedHashSet 객체를 생성하고, 이 객체의 생성자로 전달한 Arrays.asList(answer)는 주어진 배열을 자바의 List 객체로 변환합니다

String.join("", set)는 Java 언어에서 집합(set)의 요소들을 연결해서 하나의 문자열로 생성하는 코드입니다.

Java 언어에서 String.join() 메서드는 주어진 구분자와 여러 개의 문자열을 연결해서 하나의 문자열로 생성합니다. 예를 들어, String.join(", ", "a", "b", "c")는 "a, b, c"를 반환합니다.

```
 class Solution {
    public String solution(String my_string) {
        String answer = "";

        for(int i=0; i<my_string.length(); i++){
            //my_string.indexOf(my_string.charAt(i));
            if(i==my_string.indexOf(my_string.charAt(i)))
                answer+=my_string.charAt(i);
        }

        return answer;
    }
}
```

```
 i == my_string.indexOf(my_string.charAt(i))는 주어진 조건식으로, 인덱스 i가 문자열 my_string에서 처음으로 나타난 위치가 자기 자신의 위치와 같은지 여부를 판단합니다.
String my_string = "abcdef";

for (int i = 0; i < my_string.length(); i++) {
  if (i == my_string.indexOf(my_string.charAt(i))) {
    System.out.println(my_string.charAt(i));
  }
}
```

문자열 my_string의 각 인덱스를 순회하며, 인덱스 i가 처음으로 나타난 위치가 자기 자신의 위치와 같은지 판단합니다. 이 조건식이 참이면 인덱스 i에 해당하는 문자가 처음으로 나타난 것이므로 출력

Java 언어에서 문자열은 불변(immutable) 객체이기 때문에, 기존의 문자열에 새로운 문자를 추가하거나 수정하거나 삭제할 수 없습니다. 그래서 새로운 문자열을 생성하고 원하는 문자열과 연결해야 합니다.

answer += my_string.charAt(i) 코드는 문자열 my_string의 인덱스 i에 해당하는 문자를 얻기 위해 my_string.charAt(i)를 사용합니다. 이 메서드는 인덱스 i에 해당하는 문자를 반환합니다. 그리고 이 문자를 문자열 변수 answer에 추가합니다. .