<https://leetcode.com/problems/integer-to-roman/description/>

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol Value

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

For example, 2 is written as II in Roman numeral, just two one's added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

* I can be placed before V (5) and X (10) to make 4 and 9.
* X can be placed before L (50) and C (100) to make 40 and 90.
* C can be placed before D (500) and M (1000) to make 400 and 900.

Given an integer, convert it to a roman numeral.

**Example 1:**

Input: num = 3

Output: "III"

Explanation: 3 is represented as 3 ones.

**Example 2:**

Input: num = 58

Output: "LVIII"

Explanation: L = 50, V = 5, III = 3.

**Example 3:**

Input: num = 1994

Output: "MCMXCIV"

Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

**Constraints:**

* 1 <= num <= 3999

**Attempt 1: 2023-06-27**

**Solution 1:  Hash Table (10 min)**

**Style 1:**

class Solution {

public String intToRoman(int num) {

int[] values = {1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};

String[] symbols = {"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};

StringBuilder sb = new StringBuilder();

for(int i = 0; i < values.length; i++) {

while(num - values[i] >= 0) {

num -= values[i];

sb.append(symbols[i]);

}

}

return sb.toString();

}

}

**Style 2: In Style 1 the function is not really optimal because you will ALWAYS iterate until the end of your values array even if your number becomes zero! Which is not optimal at all.**

**In addition to that you didn't even check if the input number has a valid roman representation. You had to check that (even if the exercice assumes that the number has it already), you can do that easily with one line.**

**Here is an improvement of your function :**

class Solution {

public String intToRoman(int num) {

int[] values = {1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};

String[] symbols = {"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};

StringBuilder sb = new StringBuilder();

int i = 0;

while(num > 0) {

while(num - values[i] >= 0) {

num -= values[i];

sb.append(symbols[i]);

}

i++;

}

return sb.toString();

}

}

**Refer to**

<https://leetcode.com/problems/integer-to-roman/solutions/6310/my-java-solution-easy-to-understand/>

<https://leetcode.com/problems/integer-to-roman/solutions/6310/my-java-solution-easy-to-understand/comments/7606>