Documentation: Excel Sheet Column Title Conversion

Problem Description

Given an integer columnNumber, the task is to convert this integer into its corresponding column title as it appears in an Excel sheet.

Example Mapping

'A' -> 1

'B' -> 2

'C' -> 3

...

'Z' -> 26

'AA' -> 27

'AB' -> 28

'AC' -> 29

...

'ZY' -> 701

...

Input

columnNumber: An integer representing the column number in Excel.

Output

A string representing the corresponding column title.

Constraints

(1 leq text{columnNumber} leq 2^{31} 1)

Example 1:

Input: columnNumber = 1

Output: "A"

Example 2:

Input: columnNumber = 28

Output: "AB"

Example 3:

Input: columnNumber = 701

Output: "ZY"

Explanation

Excel columns are labeled in a way similar to a base-26 number system, but with a key difference: there is no "zero" character in Excel's alphabet system.

- 'A' represents 1, 'B' represents 2, ..., 'Z' represents 26.
- After 'Z', the sequence continues with 'AA' (representing 27), 'AB' (representing 28), and so on.
- This pattern is similar to numbering systems, where after reaching the end character 'Z', we start over with an increment in the next significant position (similar to how '10' follows '9' in base-10).

Approach

To convert a given columnNumber to its corresponding Excel column title:

1. **Decrement the columnNumber:** Excel columns are indexed from 1 to 26 for 'A' to 'Z'. However, in a typical base system, indices start at 0. Thus, decrementing columnNumber by 1 aligns it with 0-based indexing for easier computation.

2. **Determine the current character:**

- Compute remainder = columnNumber % 26 to find the index of the current character in the alphabet.
- Convert this remainder to a character by adding it to the ASCII value of 'A' (65). This converts 0 -> 'A', 1 -> 'B', ..., 25 -> 'Z'.

3. **Update columnNumber:**

- After determining the character for the current position, update columnNumber to columnNumber // 26 to process the next higher position.
- Continue the process until columnNumber becomes 0.

4. **Build the Result String:**
   - Each character computed from the remainder is prepended to the result string, as it represents a position starting from the least significant (rightmost) character to the most significant (leftmost) character in the column title.

Detailed Step-by-Step Process

1. **Initialization:**
   - Start with an empty string to store the result.
2. **Loop through columnNumber:**
   - Decrement columnNumber by 1 to adjust for 0-based indexing.
   - Calculate the remainder using columnNumber % 26.
   - Convert the remainder to its corresponding character in the alphabet.
   - Prepend the character to the result string.
   - Update columnNumber by performing integer division columnNumber // 26.
3. Repeat until columnNumber is 0.
4. Return the result string.

Edge Cases and Considerations

1. **Single Character Output:**
   - If columnNumber is between 1 and 26, the output is simply 'A' to 'Z'.

2. **Multi-Character Output:**
   - For values beyond 26, the output requires multiple iterations through the loop to determine each character.

3. Large Inputs:

- Given the constraint of up to (2^{31} 1), ensure that the solution efficiently handles large numbers. The while loop continues until the columnNumber becomes 0, making the time complexity approximately (O(log_{26}(n))).

Conclusion

This problem involves converting a numeric representation into a string format that aligns with the Excel column naming convention. The solution makes use of modular arithmetic and character conversion using ASCII values to generate the required column titles. The approach is efficient and runs in logarithmic time relative to the size of the input.