<https://leetcode.com/problems/smallest-string-with-a-given-numeric-value/description/>

The **numeric value** of a **lowercase character** is defined as its position (1-indexed) in the alphabet, so the numeric value of a is 1, the numeric value of b is 2, the numeric value of c is 3, and so on.

The **numeric value** of a **string** consisting of lowercase characters is defined as the sum of its characters' numeric values. For example, the numeric value of the string "abe" is equal to 1 + 2 + 5 = 8.

You are given two integers n and k. Return *the****lexicographically smallest string****with****length****equal to n and****numeric value****equal to k.*

Note that a string x is lexicographically smaller than string y if x comes before y in dictionary order, that is, either x is a prefix of y, or if i is the first position such that x[i] != y[i], then x[i] comes before y[i] in alphabetic order.

**Example 1:**

**Input:** n = 3, k = 27

**Output:** "aay"

**Explanation:** The numeric value of the string is 1 + 1 + 25 = 27, and it is the smallest string with such a value and length equal to 3.

**Example 2:**

**Input:** n = 5, k = 73

**Output:** "aaszz"

**Constraints:**

1 <= n <= 105

n <= k <= 26 \* n

**Attempt 1: 2025-09-11**

**Solution 1: Greedy + Math (10 min)**

class Solution {

    public String getSmallestString(int n, int k) {

        // Initialize all characters to 'a' (value 1)

        char[] chars = new char[n];

        for(int i = 0; i < n; i++) {

            chars[i] = 'a';

        }

        // Subtract the base value (each character at least 1)

        int remain = k - n;

        int index = n - 1;

        while(remain > 0) {

            // Add as much as possible to the last character (up to 25, since 'a' is already there)

            int add = Math.min(25, remain);

            chars[index] += add;

            remain -= add;

            index--;

        }

        return new String(chars);

    }

}

Time Complexity: O(n), we iterate through the array twice (initialization and distribution).

Space Complexity: O(n) for the character array.

**Refer to Deepseek**

I'll provide a Java solution for LeetCode 1663. "Smallest String With A Given Numeric Value", along with an explanation and complexity analysis.

**Problem Description**

The **numeric value** of a lowercase character is defined as its position in the alphabet (e.g., 'a' = 1, 'b' = 2, ..., 'z' = 26). The numeric value of a string is the sum of the numeric values of all its characters.

Given two integers n and k, return the **lexicographically smallest string** of length n with a numeric value equal to k.

**Java Solution**

class Solution {

public String getSmallestString(int n, int k) {

char[] result = new char[n];

// Initialize all characters to 'a' (value 1)

for (int i = 0; i < n; i++) {

result[i] = 'a';

}

k -= n; // Subtract the base value (each character at least 1)

int index = n - 1;

while (k > 0) {

// Add as much as possible to the last character (up to 25, since 'a' is already there)

int add = Math.min(25, k);

result[index] += add;

k -= add;

index--;

}

return new String(result);

}

}

**Explanation**

**Initialize with 'a's**:

Start by creating a character array of length n filled with 'a'. This gives each character the minimum value (1) and the total base value is n.

Subtract n from k to get the remaining value we need to distribute.

**Distribute from Right to Left**:

To form the lexicographically smallest string, we want to keep the leftmost characters as small as possible. Therefore, we add the extra value from the rightmost characters.

For each character from right to left:

Add as much as possible (up to 25, because 'a' is already 1 and we can go up to 'z'=26) to the current character.

Subtract the added value from the remaining k.

Continue until all extra value is distributed.

**Example Walkthrough**

Let n = 3, k = 27:

Initialize: ['a','a','a'] → base value = 3, remaining k = 24.

Start from the last character (index2):

Add min(25,24)=24 → character becomes 'a' + 24 = 'y' (since 'a'=1, 1+24=25 → 'y').

Now k=0.

Result: "aay" (which has value: 1+1+25=27).

But wait, is this the lexicographically smallest? Actually, we can do better by not putting all the extra at the end. However, the method above is correct because:

We want the leftmost characters to be as small as possible. By adding from the right, we ensure that the left characters remain 'a' as long as possible.

Another example: n=5, k=73:

Initialize: "aaaaa" → base=5, remaining=68.

Distribute from right:

Index4: add 25 → becomes 'z' (value 26), remaining=43.

Index3: add 25 → becomes 'z', remaining=18.

Index2: add 18 → becomes 's' (1+18=19), remaining=0.

Result: "aasz" (values: 1+1+19+26+26=73).

**Why is this lexicographically smallest?**

If we put a larger character earlier, the string would be lexicographically larger. For example, "aab" is smaller than "aba". So we keep the left ones as 'a' and put the extra weight on the right.

**Complexity Analysis**

**Time Complexity**: O(n), we iterate through the array twice (initialization and distribution).

**Space Complexity**: O(n) for the character array.

This solution efficiently builds the string by greedy distribution from the right. Let me know if you need further clarification!