Algorithmic Complexity

Number Permutation

Intermediate - java

Create a function in this `numberPermutations` class called `makeNumber(z)` which, when given an input of a number (`z`), returns the number of all possible permutations of up to 5 digits (1 through 9 inclusive) that when added will equal `z`.

For example, if `z` is `3`, your program will find that four permutations of digits add up to that value (3, 2+1, 1+1+1, and 1+2), and thus return `4`.

You can limit `makeNumber(z)` to the use of five digits, meaning anything input above 45 (9 + 9 + 9 + 9 + 9) will return no permutations.

Repeat use of a digit is acceptable: e.g. 1+1+1 would be a valid addition of digits equalling 3.

Use of a single digit is acceptable as a permutation: e.g. 3 is itself a valid permutation of digits that add up to 3.

`makeNumber(z)` is looking for permutations, not combinations: 1+5 and 5+1 would count as two unique possible ways to add to 6, not one.

If no permutations of the digits 1 through 9 add up to the number `z`, your function should return `0`.
Here are more sample inputs and outputs:

```
input: z = 1
Output: 1
The permutation:
1

Input: z = 2
Output: 2
The permutation:
1 + 1
2

Input: z = 4
Output: 8
1 + 1 + 1 + 1
1 + 1 + 2
1 + 2 + 1
1 + 3
2 + 1 + 1
2 + 2
3 + 1
4
```

The number of permutations can be quite high.

```
Input: z = 6
Output: 31

Input: z = 12
Output: 554

Input: z = 21
Output: 3703

Input: z = 35
Output: 980

Input: z = 45
Output: 1
```

Variations of this challenge have been reported to have been asked at interviews with Facebook. If you've covered the material in Pass the Technical Interview with Java or an equivalent, you should be able to solve this challenge. If you have trouble, try refreshing your knowledge there first, particularly on recursion.