

Programação dinâmica em dígitos

Professor: Wladimir Araújo Tavares

Considere o seguinte problema:

Calcule a soma de todos os dígitos de todos os números entre 1 e A com

$$1 < A \leq 10^{18}$$

Por exemplo, se $B = 13$, a resposta é

$$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 1 + 0 + 1 + 1 + 1 + 2 + 1 + 3 = 51$$

Observe que as restrições dos valores de entrada já impossibilita um algoritmo de força bruta que calcula a soma dos dígitos de cada número entre 1 e A.

Considere que você quer calcular a soma dos dígitos de todos os números menores ou iguais a 4578. Vamos considerar os casos em que os dois primeiros dígitos da esquerda para direita já foram escolhidos.

- Caso 1: Quando os dois elementos escolhidos coincidem com os dígitos dos números, ou seja, 45xy. Neste caso,
 - se $x \in \{0, 1, 2, 3, 4, 5, 6\}$ então $y \in \{0, 1, 2, \dots, 9\}$ (y não restrito)
 - se $x \in \{7\}$ então $y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ (y restrito)
- Caso 2: Quando o prefixo formado pelos dois primeiros elementos é menor que o prefixo formado pelos dois primeiros dígitos do nosso limite superior. Por exemplo, 42xy
 - Neste caso, $x, y \in \{0, 1, 2, \dots, 9\}$ (x,y não restritos)

Seja $N = d_n d_{n-1} \dots d_1$ o inteiro que queremos calcular a soma dos dígitos de todos os números entre 1 e N. Considere que já construímos um número no seguinte formato:

$$e_n e_{n-1} \dots e_{idx+1} ? \dots ?$$

tal que $e_n + e_{n-1} + \dots + e_{idx+1} = sum$

Vamos associar ao subproblema $dp(idx, tight, sum)$ que vai calcular a soma dos dígitos de todos os números entre 1 e N tal que

$$e_n + e_{n-1} + \dots + e_{idx+1} = sum$$

e

$$(e_n e_{n-1} \dots e_{idx+1})_{10} < (d_n d_{n-1} \dots d_{idx+1})_{10}, \text{ if } tight = false$$

ou

$$(e_n e_{n-1} \dots e_{idx+1})_{10} = (d_n d_{n-1} \dots d_{idx+1})_{10}, \text{ if } tight = true$$

O problema a ser resolvido será $dp(n, 1, 0)$

Relação de recorrência

$$dp(idx, 0, sum) = \sum_{d=1}^{d=9} dp(idx-1, 0, sum+d)$$

$$dp(idx, 1, sum) = dp(idx-1, 1, sum + d_{idx}) + \sum_{d=1}^{d=d_{idx}-1} dp(idx-1, 0, sum+d)$$

Implementação

```
#include "bits/stdc++.h"
using namespace std;
#define max_digit 20
#define max_sum max_digit*9

long long dp[max_digit][max_sum][2];

long long digitSum(int idx, int sum, int tight,
                  vector<int> &digit)
{

    if (idx == -1)
        return sum;

    if (dp[idx][sum][tight] != -1)
        return dp[idx][sum][tight];

    if (!tight){
        long long ret = 0;

        for (int i = 0; i <= 9 ; i++)
        {
```

```

        ret += digitSum(idx-1, sum + i, 0, digit);
    }
    return dp[idx][sum][0] = ret;
} else {

    long long ret = digitSum( idx-1, sum + digit[idx], 1, digit);

    for(int i = 0; i < digit[idx]; i++){

        ret += digitSum( idx-1, sum + i, 0, digit);
    }

    return dp[idx][sum][1] = ret;

}

}

vector <int >getDigits(string ss){
    vector <int> digits;
    digits.resize( ss.size() );
    int pos = ss.size()-1;
    for(int i = 0; i < ss.size(); i++){
        digits[pos--] = ss[i] - '0';
    }
    return digits;
}

int main()
{
    string a, b;
    getline(cin, a);
    vector<int> digitA;
    digitA = getDigits(a);
    memset(dp, -1, sizeof(dp));
    long long ans1 = digitSum(digitA.size()-1, 0, 1, digitA);
    cout << ans1 << endl;
    return 0;
}

```