問題 005

解説（P3）

キーワード：ダブリング的に高速化できる場合もある

解法（小課題 3）

実は、$O(B^2 \log N)$ でも解くことができます。以下のことを考えてみましょう。

- $dp[2][?]$ の値を $dp[1][?]$ から $O(N^2)$ で求められないか？
- $dp[4][?]$ の値を $dp[2][?]$ から $O(N^2)$ で求められないか？
- $dp[8][?]$ の値を $dp[4][?]$ から $O(N^2)$ で求められないか？

実は求めることができます。実際、以下のような遷移になるのです。

$$dp[2][(10^1 \times j + k) \ mod \ B] += dp[1][j] \times dp[1][k]$$
$$dp[4][(10^2 \times j + k) \ mod \ B] += dp[2][j] \times dp[2][k]$$
$$dp[8][(10^4 \times j + k) \ mod \ B] += dp[4][j] \times dp[4][k]$$
$$\vdots$$

イメージで考えると、例えば "31415926" の上 4 桁「3141」を 10000 倍して 5926 足したら 31415926 になる、という感じです。例えば $B = 17$ の場合を考えてみましょう。実際に、$3141 \ mod \ 17 = 13$, $5926 \ mod \ 17 = 10$ ですが、$(13 \times 10^4 + 10) \ mod \ 17 = 11$ であり、これは $31415926 \ mod \ 17$ と一致します。

このように、$10^1, 10^2, 10^4, 10^8, 10^{16}, \ldots$ を B で割った余りを前計算することで、以下のような感じの実装で $O(B^2 \log N)$ で解けます。（※）

```
for (int i = 0; i < 60; i++) {
    for (int j = 0; j < B; j++) {
        for (int k = 0; k < B; k++) {
            int nex = (j * power10[i] + k) % B;
            DP[i + 1][nex] += DP[i][j] * DP[i][k];
            DP[i + 1][nex] %= mod;
        }
    }
}
```

補足説明として、なお、遷移部分を高速フーリエ変換（FFT）で高速化することで $O(B \log B \log N)$ で解くことができます。mod の値が 998244353 のような特殊な値ではないので一工夫必要ですが、それでも例えば 3 種類くらい mod を用意して中国剰余定理で復元する方法などで上手くいきます。

結果的に、ダブリングのような手法で全体計算量 $O(B^2 \log N)$ でこの問題を解くことができ、小課題 3 まで正解することができました。

なお、似たようなアイデアとしてきたまさ法という手法があります。興味のある人は是非調べてみましょう。

【類題】

- s8pc #6 F – Random Shuffles
- TDPC T – フィボナッチ

※power10[i] は $10^{2^i} \ mod \ B$ のことを指します。これで元々の dp[1][?], dp[2][?], dp[4][?], … が求まりました。あとは繰り返し二乗法の要領で計算するだけです。

※サンプルコードは GitHub 上にアップロードしています。是非ご活用ください。