Zadanie przystawka

Rozważmy chyba najbardziej znany problem ze zliczaniem. Na taras wieży widokowej prowadzą schody o n stopniach. Turysta wchodzi na taras wykonując kroki co 1 lub co 2 stopnie. Na ile sposobów turysta może przejść schody?.

$$dp[i] = \underline{dp[i-1]} + \underline{dp[i-2]}$$

$$dp[0] = dp[1] = 1$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 2 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{bmatrix}$$

$$(v \cdot M) \cdot M = v \cdot M^2$$

$$M^8 = (M^4)^2$$

Obserwacja: jeżeli v_i będzie wektorem k kolejnych wartości ciągu

$v_i = \begin{bmatrix} a_{i+k-1} \\ a_{i+k-2} \\ \dots \\ a_i \end{bmatrix}$, a macierz M będzie zdefiniowana jako

$M = \begin{bmatrix} c_1 & c_2 & c_3 & \dots & c_{k-1} & c_k \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}$, to

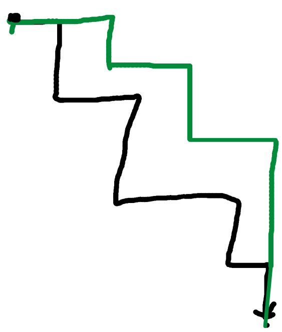
$$v_{i+1} = M \cdot v_i$$

$$a_n = 3a_{n-1} + 2a_{n-2} + 5a_{n-3}$$

$$\begin{bmatrix} 3 & 2 & 5 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} n+1 \\ n \\ n-1 \end{bmatrix}$$

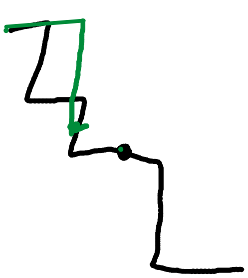
$\cancel{n-2}$

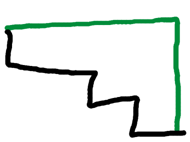
Zadanie 7. Dana jest plansza o wymiarach n na n, podzielona na kwadraty jednostkowe. W każdym polu znajduje się co najwyżej jeden diament. Plansza zadana jest jako kwadratowa tablica zer i jedynek – jedynki tam, gdzie diament jest, zero gdzie go nie ma. Wyruszamy z lewego górnego rogu planszy i wykonujemy ruchy o jedno pole wyłącznie w prawo lub w dół, a po dojściu do prawego dolnego rogu wracamy wykonując ruchy w lewo lub do góry. W drodze powrotnej możemy przechodzić przez te same lub inne pola, ale żadnego diamentu nie da się wziąć dwa razy. Obliczyć, ile najwięcej diamentów możemy zebrać.



$$dp[x_1][y_1][x_2][y_2] = dia[x_1][y_1] + (x_1 \neq x_2 \; ? \; dia[x_2][y_2] : 0)$$
$$+ \max(dp[x_1-1][y_1][x_2-1][y_2],$$
$$dp[x_1-1][y_1][x_2][y_2-1],$$
$$dp[x_1][y_1-1][x_2-1][y_2],$$
$$dp[x_1][y_1-1][x_2][y_2-1])$$

4 4





$$y_2 = x_1 + y_1 - x_2$$

$$x_1 + y_1 = x_2 + y_2$$

$$dp[x_1][y_1][x_2]$$

$$\max\big(dp[x_1][y_1-1][x_2],\ dp[x_1-1][y_1][x_2-1],\ dp[x_1-1][y_1][x_2],\ dp[x_1][y_1-1][x_2-1]\big)$$

Mamy monety o nominałach $1, k, k^2, k^3, \ldots$ (k jest stałe). Na ile sposobów możemy rozmienić nimi kwotę n?

$dp[i] = dp[i-1]$

if (i % k == 0)

$dp[i] += dp[i/k]$

$i = k \cdot q = k^2 + k + k$

$q = k + 1 + 1$

$1 \cdot k \quad k^2 \ \ldots$

$3 \cdot k$

```
int ustal(int x) {
    //upraszcza kod - zwraca reszte z dzielenia przez 3
    x += 33333333; //nie chcemy modulowac liczb ujemnych
    return x % 3;
}

DP[0][0] = 1;
for (int i = 0; i <= 9; i ++)
    DP[1][i % 3] ++;
for (int i = 2; i <= MAX_N; i ++)
    for (int j = 0; j < 3; j ++)
        for (int pierwsza = 0; pierwsza < 10; pierwsza ++)
            DP[i][j] += DP[i-1][ustal(j - pierwsza)];
```

$dp[n][j]$ - liczby n-cyfrowe dające j mod 3

1

06

4 _ _ - - -

4

0

```cpp
vector <int> wynik;
int reszta = 0;
for (int i = MAX_N; i > 0; i -) {
    for (int c = 0; c < 10; c ++) {
        long long dokoncz = DP[i - 1][ustal(3 - reszta - c)];
        if (dokoncz >= k) {
            wynik.push_back(c);
            reszta += c;
            break;
        }
        else k -= dokoncz;
    }
}
int pocz = 0;
while (wynik[pocz] == 0)
    pocz ++;
for (int i = pocz; i < wynik.size(); i ++)
    cout << wynik[i];
```

4 1 0 1

0

1

0 . . . .

9 . . . .

$15 - 4 = 11$

45

0

1

$11 - 3 = 8$

2 $8 - 3 = 5$

3 $5 - 4 = 1$

4 2

```cpp
const int D = 200;
const int S = 200;
const int M = 200;

const LL MAXV = 1e18;

LL dp[D+5][S+5][M+5];
int powBase[D+5];

int s, m, q;

void add(LL &a, LL &b) {
    a += b;
    a = min(a, MAXV);
}

void query(LL k) {
    string res;
    int sum = s;
    int mod = 0;

    if(dp[D][s][0] < k) {
        cout << "NIE\n";
        return;
    }

    for(int i = D;i > 0;--i) {
        for(int d = 0;d <= 9;++d) {
            //25 (7) - 2 = 5
            int newSum = sum - d;
            //1 - 2 = 2 (mod 3)
            int newMod = ((mod - d*powBase[i-1]) % m + m) % m; // mod - d * powBase[i-1] === x
            if(dp[i-1][newSum][newMod] < k)
                k -= dp[i-1][newSum][newMod];
            else {
                res += '0'+d;
                mod = newMod;
                sum = newSum;
                break;
            }
        }
    }

    while(res[0] == '0')
        res.erase(res.begin());

    if(res.empty())
        cout << 0 << "\n";
    else
        cout << res << "\n";
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cin >> s >> m >> q;

    powBase[0] = 1;
    for(int i = 1;i <= D;++i) {
        powBase[i] = powBase[i-1]*10 % m;
    }

    ///Zliczanie
    dp[0][0][0] = 1;
    for(int i = 0;i <= D;++i) {
        for(int j = 0;j <= s;++j) {
            for(int r = 0;r < m;++r) {
                for(int d = 0;d <= 9;++d) {
                    //25
                    add(dp[i+1][j+d][(r+d*powBase[i]) % m], dp[i][j][r]);   // [ilosc cyfr][suma cyfr][modulo m]
                }
            }
        }
    }
```

$\ldots d \underbrace{\qquad\qquad}_{i-1}$

$0 \equiv mod + d \cdot 10^{i-1} + x$

$S \underbrace{\qquad\qquad}_{d} \cdot 10^{d}$

$10^{d} \equiv 1 \pmod 3$