Lista 3, Zadanie 4

Problem

Definiujemy

$$G_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \\ G_{n-1} + G_{n-2} + G_{n-3} & \text{if } n \geq 3 \end{cases}$$

Jest to ciąg generowany rekursywnie na podstawie ziarna *(seed-a)* w postaci trzech pierwszych wyrazów.

Concept (bottom-up)

Analogicznie jest w przypadku ciągu Fibonacciego, który definiujemy następująco:

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{oth.} \end{cases}$$

Jednakże możemy zapisać ciąg Fibonacciego w sposób macierzowy:

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} F_{n+2} & F_{n+1} \\ F_{n+1} & F_n \end{bmatrix}$$

Dla naszego problemu G_n również musimy znaleźć odpowiednią macierz, którą po podniesieniu do odpowiedniej potęgi uzyskamy dostęp do n-tego wyrazu ciągu G.

Rozwiązanie

Musimy znaleźć taką macierz A, że:

$$\begin{bmatrix} G_n \\ G_{n-1} \\ G_{n-2} \end{bmatrix} = A \times \begin{bmatrix} G_{n-1} \\ G_{n-2} \\ G_{n-3} \end{bmatrix}$$

Czyli, żeby otrzymać rekurencję $G_n = G_{n-1} + G_{n-2} + G_{n-3}$ pierwszy wiersz A musi wynosić $A_1 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$.
W przypadku wierszy A_2 oraz A_3 wystarczy *„przepisać"* wartości G_{n-1} oraz G_{n-2}. Więc wystarczy zrobić prostą mapę bitową przenoszącą te wartości do macierzy po prawej stronie. Dlatego też:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^2 = \begin{bmatrix} 2 & 2 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^3 = \begin{bmatrix} 4 & 3 & 2 \\ 2 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

i ogólnie

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^n = \begin{bmatrix} G_n & G_{n-1} + G_{n-2} & G_{n-1} \\ G_{n-1} & G_{n-2} + G_{n-3} & G_{n-2} \\ G_{n-2} & G_{n-3} + G_{n-4} & G_{n-3} \end{bmatrix}$$

przy czym, żeby nie mieć problemu z określeniem liczb dla ujemnych indeksów (np. co to jest G_{n-4} kiedy mamy $n = 3$?) ustalimy macierz początkową

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^3 = \begin{bmatrix} 4 & 3 & 2 \\ 2 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} G_4 & G_3 + G_2 & G_3 \\ G_3 & G_2 + G_1 & G_2 \\ G_2 & G_1 + G_0 & G_1 \end{bmatrix}$$

którą będziemy przemnażać przez macierz A celem uzyskania kolejnych wyrazów ciągu G_n.

W takim układzie mamy na samym początku już policzone wyrazy od G_0 do G_4 a następne wyrazy będziemy uzyskiwali przez mnożenie macierzy X przez A.

Teraz, problemem jest osiągnięcie złożoności obliczeniowej $T(n) = O(\lg n)$ przy przemnażaniu macierzy X przez A.
Należy zauważyć, że w ogólnym przypadku nie ma przemienności w mnożeniu macierzy. Tutaj jednak, operujemy na potęgach tak naprawdę jednej macierzy co daje nam przemienność.

Przy przemnażaniu naszej *„macierzy atomowej”* A nie należy się też martwić o złożoność, ponieważ za każdym razem jest to macierz o 3 kolumnach i 3 wierszach.

Można użyć algorytmu analogicznego do liczenia potęg liczb naturalnych w czasie $O(\lg n)$.

Nasz algorytm działałby następująco:

1. Jeśli $n \in \{0, \ldots, 4\}$ zwróć zapisaną wartość.
2. W przeciwnym wypadku oblicz $n - 4$ potęgę macierzy A przy pomocy poniższej funkcji `power_matrix`, pomnóż wynik przez X i go zwróć.

Liczenie potęgi macierzy w czasie $O(\lg n)$

Zdefiniujemy funkcję `PowerMatrix`(A, k):

1. `output =` $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
2. `while` $k > 0$:
    1. `if` $2 \not| k$:
        1. `output *=` A
    2. $k = \lfloor \frac{k}{2} \rfloor$
    3. $A = A^2$
3. `return output`