Zadanie 4 - Lista 3

Treść zadania 4. Zdefiniujmy liczbę

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

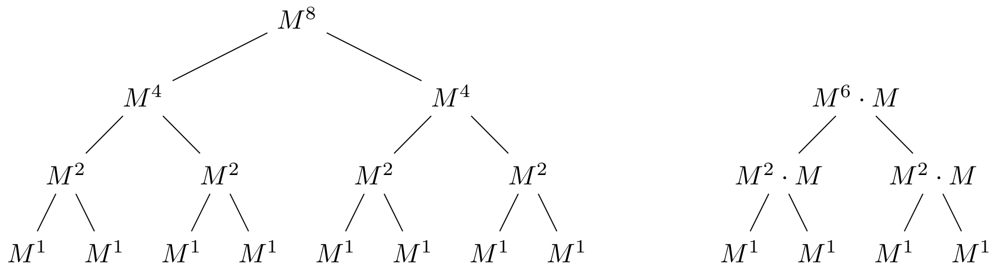
Zaprojektuj macierzowy algorytm wyliczania liczby G_n w czasie $O(logn)$. Udowodnij poprawność działania zaproponowanego algorytmu.

Rozwiązanie.
Wykorzystamy algorytm dziel i zwyciężaj z podejściem macierzowym.
Pseudokod algorytmu:

g(n)
  if $n = 0$
    then return 0
  if $n = 1$
    then return 1
  $X \leftarrow power(n-1)$
  return $X[1][1]$

power(n)
  if $n = 1$
    then return $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$
  $A \leftarrow power(\lfloor \frac{n}{2} \rfloor)$
  $A \leftarrow A \times A$
  if n nieparzysta
    then $A \leftarrow A \times \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$
  return A

Aby otrzymać G(n) potrzeba podnieść macierz M do potęgi $n-1$. Dla początkowych $n \in \{0, 1\}$ zwracamy ich wartość, dla dalszych wykonujemy funkcję $power(n-1)$. Funkcja *power* działa rekurencyjnie. Jeśli dostanie $n = 1$ zwróci naszą macierz M, w innym przypadku wykonuje się rekurencyjnie dla argumentu $\lfloor \frac{n}{2} \rfloor$. Następnie podnosi otrzymaną macierz do kwadratu i gdy n było nieparzyste mnoży jescze raz przez naszą macierz M. Poniżej przykład drzewka rekurencyjnego dla parzystego $n = 8$ oraz nieparzystego $n = 7$.



Nasza macierz $M = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$ została zaprojektowana tak, aby w lewym górnym rogu $M[1][1]$ znajdował się wynik dla $G(n)$ przy podniesieniu jej do $n-1$ dla $n \geqslant 2$.

W górnym wierszu znajdują się kolejno 3 po sobie następujące liczby $\begin{bmatrix} A_2 & A_1 & A_0 \\ - & - & - \\ - & - & - \end{bmatrix}$ natomiast w pierwszej kolumnie są same jedynki $\begin{bmatrix} 1 & - & - \\ 1 & - & - \\ 1 & - & - \end{bmatrix}$ a więc kwadrat i każda następna potęga będzie dawała sumę górnego wiersza w $M[1][1]$.

$$\begin{bmatrix} A_2 & A_1 & A_0 \\ - & - & - \\ - & - & - \end{bmatrix} \times \begin{bmatrix} 1 & - & - \\ 1 & - & - \\ 1 & - & - \end{bmatrix} = \begin{bmatrix} A_2 + A_1 + A_0 & - & - \\ - & - & - \\ - & - & - \end{bmatrix} = \begin{bmatrix} A_3 & - & - \\ - & - & - \\ - & - & - \end{bmatrix}$$

Teraz potrzebujemy, aby nasz górny wiersz wyglądał tak $\begin{bmatrix} A_3 & A_2 & A_1 \\ - & - & - \\ - & - & - \end{bmatrix}$ czyli przy każdym podnoszeniu wartość w $M[1][2]$ musi przejść do $M[1][3]$, a także wartość w $M[1][1]$ do $M[1][2]$.

$$\begin{bmatrix} A_2 & A_1 & A_0 \\ - & - & - \\ - & - & - \end{bmatrix} \times \begin{bmatrix} - & 1 & 0 \\ - & 0 & 1 \\ - & 0 & 0 \end{bmatrix} = \begin{bmatrix} - & A_2 \cdot 1 + A_1 \cdot 0 + A_0 \cdot 0 & A_2 \cdot 0 + A_1 \cdot 1 + A_0 \cdot 0 \\ - & - & - \\ - & - & - \end{bmatrix} = \begin{bmatrix} - & A_2 & A_1 \\ - & - & - \\ - & - & - \end{bmatrix}$$

Czyli nasza macierz musi wyglądać tak $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$

Czas działania algorytmu:

W wariancie pesymistycznym, gdy $n \geqslant 3$ to złożoność czasowa wynosi $\Theta(\log n)$. A więc ogólnie złożoność algorytmu wynosi $O(\log n)$, ponieważ:

$T(n) = O(\log n)$, czyli $\forall n \geqslant n_0 : T(n) \leqslant c \cdot \log n$

$T(n) = \Theta(\log n)$, czyli $\forall n \geqslant n_0 : c_1 \cdot \log n \leqslant T(n) \leqslant c_2 \cdot \log n$

Więc pozostaje nam udowodnić, że $T(n) = \Theta(\log n)$ dla przypadku pesymistycznego.

W obliczaniu czasu działania algorytmu różnica między $\frac{n}{2}$, a $\left\lfloor \frac{n}{2} \right\rfloor$ nie wpływa istotnie na asymptotyczne zachowanie rekurencji.

Czas wykonania się algorytmu składa się z sumy czasu wykonania $g(n)$ oraz $power(n)$ i czasu algorytmu dla $power(\frac{n}{2})$.

Czas wykonania się $g(n)$ oraz $power(n)$ wynosi $\Theta(1)$, ponieważ nie zależy od liczby n.

Zatem:

$$T(n) = T(\frac{n}{2}) + \Theta(1)$$

Skorzystamy z twierdzenia o rekurencji uniwersalnej.
Mając rekurencję $T(n) = aT(\frac{n}{b}) + f(n)$ i $f(n) = \Theta(n^{\log_b a})$, to $T(n) = \Theta(n^{\log_b a}) \log n)$.
Czyli w naszym przypadku:

$$f(n) = \Theta(n^{\log_2 1}) = \Theta(1)$$
$$T(n) = \Theta(n^{\log_2 1}) \log n) = \Theta(\log n)$$