1	2	3	4	5	6	7	8	9	10	Σ
-	-	-	+	-	+	+	+	-	+	???

ZADANIE 4.

Udowodnij, że algorytm mnożenia liczb "po rosyjsku" jest poprawny. Jaka jest jego złożoność czasowa i pamięciowa przy:

1. jednorodnym kryterium kosztów,

2. logarytmicznym kryterium kostów?

Dowodzik:

Ustalmy dowolne $b \in \mathbb{N}$. Pokażemy przez indukcję, że dla dowolnego $a \in \mathbb{N}$ wynik algorytmu jest równy ab. Przypadek bazowy, czyli $a = 1$ jest trywialny.

Załóżmy teraz, że dla dowolnego $a' \leq n$ algorytm działa i niech $a = n + 1$. Rozważmy dwa przypadki:

1. $(n + 1)$ jest nieparzyste. Wtedy $a_1 = (n + 1)$ jest nieparzyste, wpp. do $a'_1 = n$. Dalej, $a_2 = \lfloor \frac{n+1}{2} \rfloor = \lfloor \frac{n}{2} \rfloor = a'_2$, czyli od drugiego miejsca ciąg a_i dla $n + 1$ jest taki sam jak dla n, więc:

$$\sum_{i=1, a_i \text{ np}}^{k} b_i = b_1 + \sum_{i=2, a'_i \text{ np}} b_i = b_1 + n \cdot b = b + bn = b(n+1)$$

2. $(n + 1)$ jest parzyste. Wtedy $a_1 = (n + 1)$ jest parzyste, więc b_1 nie zostanie użyte w sumie. Natomiast $\lfloor \frac{n+1}{2} \rfloor = \lfloor \frac{n}{2} \rfloor + 1 = a'_2 + 1$, czyli od trzeciego indeksu a_i jest taki sam jak a'_i

$$\sum_{i=1, a_i \text{ np}}^{k} b_i = a_2 b_2 + \sum_{i=3, a_i \text{ np}}^{k} b_i = (a'_2 + 1) 2b + \sum_{i=3}^{k} b_i = 2b + \sum_{i=2, a'_i \text{ np}}^{k} b_i = 2b + \sum_{i=1}^{k} b_i - b_1 = b + \sum_{i=1}^{k} b_i = b + nb$$

Złożoność:

1. Będziemy obliczać wyrazy ciągu a_i $\log_2(a)$ razy, za każdym razem będziemy dzielić, sprawdzać podzielność, mnożyć b i ewentualnie dodawać, co jest mniej więcej stałą liczbą operacji.

2. ?????

ZADANIE 5.

Oszacuj z dokładnością do Θ złożoność poniższego fragmentu programu:

```
res <- 0
for i <- 1 to n do
    j <- i
    while (j jest parzyste) j <- j/2
    res <- res + j
```

ZADANIE 6.

Pokaż, w jaki sposób algorytm "macierzowy" obliczania n-tej liczby Fibonacciego można uogólnić na inne ciągi, w których kolejne elementy definiowane są liniową kombinacją skończonej liczby elementów wcześniejszych. Następnie uogólnij swoje rozwiązanie na przypadek, w którym n-ty element ciągu definiowany jest jako suma kombinacji liniowej skończonej liczby elementów wcześniejszych oraz wielomianu zmiennej n.

Pierwsza część jest dość prosta. Rozważamy ciąg zdefiniowany rekurencyjnie

$$a_n = \alpha_1 a_{n-1} + \alpha_2 a_{n-2} + \ldots + \alpha_k a_{n-k}.$$

Popatrzmy na macierz

$$A = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \ldots & \alpha_{k-1} & \alpha_k \\ 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 1 & 0 \end{bmatrix}$$

Mnożąc A^{n-1} przez wektor

$$\begin{bmatrix} a_{k-1} \\ a_{k-2} \\ \dots \\ a_0 \end{bmatrix}$$

dostajemy wektor zawierający a_n na pierwszym miejscu oraz wszystkie poprzednie miejsca na pozostałych miejscach.

Teraz co się dzieje dla ciągu zawierającego wielomian zmiennej n?

Przyjrzyjmy się ciągowi zdefiniowanemu rekurencyjnie ze szczyptą wielomianu:

$$a_n = \sum_{i=1}^{k} \alpha_i a_{n-i} + \sum_{i=0}^{m} \beta_i n^i$$

Mogę zacząć od tego, że

$$\begin{bmatrix} a_n \\ a_{n-1} \\ a_{n-2} \\ \dots \\ a_{n-k+1} \end{bmatrix} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_{k-1} & \alpha_k \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} a_{n-1} \\ a_{n-2} \\ a_{n-3} \\ \dots \\ a_{n-k} \end{bmatrix} + \begin{bmatrix} \sum \beta_i n^i \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

Czy jest coś więcej, co mogę o tym cudeńku powiedzieć?

ZADANIE 7.

Rozważ poniższy algorytm, który dla danego (wielo)zbioru A liczb całkowitych wylicza pewną wartość. Twoim zadaniem jest napisanie programu (w pseudokodzie), możliwie najoszczędniejszego pamięciowo, który wylicza tę samą wartość.

```
while |A| > 1 do
    a <- losowy element z A
    A <- A \ {a}
    b <- losowy element z A
    A <- A \ {b}
    A <- A u {a-b}

output (x mod 2), gdzie x jest elementem ze zbioru A
```

Zauważmy, że wynik zależy od ilości elementów nieparzystych. To znaczy, jeżeli elementów nieparzystych jest parzyście wiele, to dostaniemy 0, wpp. dostaniemy 1, gdyż te zbędne jedynki nieparzystości nie zniosą się do końca.

Nie musimy więc trzymać całego A w pamięci przez cały czas, a wystarczy wczytywać je element po elemencie, sprawdzać jego podzielność i odpowiednio modyfikować aktualną wartość wyniku:

```
ret = 0
while A ma niewczytane elementy:
    a <- kolejny element A
    if a % 2
        ret = (ret + 1) % 2

output ret
```

ZADANIE 8.

Ułóż algorytm, który dla drzewa $T = (V, E)$ *oraz listy par wierzchołków* $\{v_i, u_i\}$ $(i = 1, \dots, m)$ *sprawdza, czy* v_i *leży na ścieżce z* u_i *do korzenia. Przyjmij, że drzewo zadane jest jako lista* $(n - 1)$ *krawędzi* (p_i, a_i) *takich, że* p_i *jest ojcem* a_i *w drzewie.*

Koncepcja jest taka, że idziemy od u_i w górę, jeśli trafimy na v_i, piszemy TAK, wpp gdy jesteśmy na korzeniu, piszemy NIE

ZADANIE 10.

Ułóż algorytm dla następującego problemu:

PROBLEM

dane: $n, m \in \mathbb{N}$

wynik: wartość współczynnika przy x^2 *(wzięta modulo* m*) wielomiany* $(\ldots((x-2)^2-2)^2\ldots-2)^2$*, gdzie nawiasów ogółem jest* n.

Czy widzisz zastosowanie metody użytej w szybkim algorytmie obliczania n*-tej liczby Fibonacciego do rozwiązania tego problemu?*

Tak naprawdę wystarczy, że będę trzymać to, co się dzieje przy wyrazie stałym, wyrazie z x i wyrazie z x^2.

Wyraz stały w tym cudeńku to zawsze będzie 4, bo z tego nawiasu w środku zawsze mamy 4 na końcu, odejmujemy 2 i podnosimy do kwadratu wielomian z 2 jako wyrazem stałym. To widać.

Wyraz przy x to będą kolejne potęgi 4 na minusie? Można to pokazać przy pomocy ciągu rekurencyjnego a_n:

$$a_1 = -4$$

$$a_n = 4a_{n-1} = -4^n$$

bo

$$\begin{aligned} p_n(x) &= (\ldots((x-2)^2-2)^2\ldots-2)^2 = 4 + a_n x + \ldots. \\ (p_n(x)-2)^2 = p_n(x)^2 - 4p_n(x) + 4 &= (4 + a_n x + \ldots)^2 - 4(4 + a_n x + \ldots) + 4 = \\ &= 8a_n x - 4a_n x + \ldots = 4a_n x \end{aligned}$$

To teraz pozostaje mi napisać rekurencję na b_n, czyli wyraz przy x^2.

$$\begin{aligned} p_n(x) &= (\ldots((x-2)^2-2)^2\ldots)^2 = 4 - 4^n x + b_n x^2 + \ldots \\ (p_n(x)-2)^2 = p_n(x)^2 - 4p_n(x) + 4 &= (4 - 4^n x + b_n x^2 + \ldots)^2 - 4(4 - 4^n x + b_n x^2 + \ldots) + 4 = \\ &= 8b_n x^2 + 4^{2n} x^2 - 4b_n x^2 + \ldots \end{aligned}$$

Czyli

$$b_1 = 1$$

$$b_n = 4b_n + 16^{n-1}$$