

# LISTA 0

## ZAD. 3. ALGORYTM SORTOWANIA BĄBELKOWE.

```

bubble sort(T[0, n-1])
    for i ← 0 to n do
        for j ← 1 to n-i do
            if T[j-1] > T[j] then
                pam = T[j-1]
                T[j-1] = T[j]
                T[j] = pam

```

**RÓŁKAD DANYCH:** Względem danych nie ma znaczenia, ponieważ algorytm sortowania bąbelkowego ma asymptotyczną złożoność czasową  $O(n^2)$ . Zatem aby dane ułożone lub nie, nie zmienia złożoności czasowej w przypadku sortowania bąbelkowego. Może to jednak nastąpić w przypadku insert sort (sortowanie przez wstawianie), w którym względem danych nie ma znaczenia i może spowodować zmianę asymptotycznej złożoności czasowej z  $O(n^2)$  na  $O(n)$ .

**WIELKOŚĆ REKORDÓW:** jeśli rekordy są duże to operacja przedstawiania elementów jest kosztowna. Sortowanie bąbelkowe w najgorszym wypadku dokona  $n^2$  zmian. To samo dotyczy insert sort. W przypadku select sort dokonywane jest maksymalnie  $n$  zmian.

**STABILNOŚĆ:** procedura jest stabilna gdy rekordy o jednokowych kluczach pozostają w tablicy wynikowej w takim samym względnym porządku w jakim były początkowo. Sortowanie insert sort i bubble sort są stabilne, w przeciwieństwie do select sort (przykład  $\begin{matrix} 5 & 5 & 1 & 0 \\ A & B & C & D \end{matrix}$ )

**ZAD. 4. UDOWODNIJ, ŻE ALGORYTM MNOŻENIA LICZB "PO RÓSY JSKU" JEST POPRAWNY. JAKA JEST JEGO ZŁOŻNOŚĆ CZASOWA I PAMIĘCIOWA PRZY:**

- JEDNORODNYM KRYTERIUM KOSTÓW
- LOGARYTMICZNYM KRYTERIUM KOSTÓW?

**MNOŻENIE "PO RÓSY JSKU"**

1. oblicz ciąg  $a_1, a_2, \dots, a_k$  taki, że  $a_1 = a$ ,  $a_k = 1$ ,  $a_{i+1} = \lfloor \frac{a_i}{2} \rfloor$  (dla  $i=1, \dots, k-1$ ).

2. oblicz ciąg  $b_1, b_2, \dots, b_k$  taki, że  $b_1 = b$ ,  $b_{i+1} = 2b_i$  (dla  $i=1, \dots, k-1$ )

3. oblicz

$$\sum_{i=1}^k b_i$$

ai nieparzyste

mnożenie ( $b_1 b_2$ ):

wynik = 0

while ( $b > 0$ )

if ( $b \% 2 == 1$ )

wynik += e

e \*= 2

b /= 2

return wynik

## ROZPIŚ DŁUGI

## PRZYKŁAD

3 3 2	5 1
6 6 4	2 5
1 3 2 8	1 2
2 6 5 6	0
5 3 1 2	0
1 0 6 2 4	1
1 6 9 3 2	1

} wykonujemy działanie  
z algorytmu

zauważmy, że działanie w  
prawej kolumnie to operacja  
podobna do obliczenia wartości  
liczby  $5^3$  w systemie dwójkowym.

Operacje w prawym słupku, w którym wykres-  
lony liczby parzyste (czyli takie, które  
podzielone przez 2 dają 0) to zanione  
liczby  $5^3$  w systemie zero-jedynkowy.

Operacje w lewym słupku to obliczenie  
wartości dziesiętnych kolejnych cyfr układu  
dwójkowego, pomnożonych przez liczbę (332).

$664 =$	$2^0 \cdot 332$	51
	$2^1 \cdot 332$	25
		12
		8
$5312 =$	$2^4 \cdot 332$	3
$10624 =$	$2^5 \cdot 332$	1

$$16932 = 332(2^0 + 2^1 + 2^4 + 2^5) = \\ = 332 \cdot 51$$

- MEDNORODNE KRYTERIUM koszt każdej operacji maszyny RAM jest jednostkowy.

Złożoność czasowa algorytmu mnożenia "po rosyjsku" to  $O(\log n)$ , ponieważ w  
każdym poniższym działaniu mnożymy  
liczby i ewentualnie dodajemy. Działanie  
te wykonujemy  $\log n$  razy. Każde działanie  
wykonuje się w czasie  $O(1)$ , a złożoność  
pamięciowa wynosi  $O(1)$ . Koszt każdej operacji maszyny  
RAM jest jednostkowy.

LOGARITMICZNE KRYTERIUM KOSTÓW - kost operacji mnożenia RAM jest równy sumie głębi operendów.

Złożoność czasowa algorytmu to  $\log_2 n$ . Względniemy najgorszy przypadek rozmiaru danych. Wtedy ich przedstawienie  $\log_2 nm$  (liczba n'm system binarnym). Zatem złożoność czasowa  $O(\log_2 n \cdot \log_2 nm)$ . Złożoność pamięciowa to  $O(\log_2 nm)$ , bo największa możliwa liczba.

ZAD. 6 RÓWNAZ PONIŻSZY ALGORYTM, KTÓRY DLA DANEGO (WIELO)ZBIORU A LICZB CIEKAWYCH WYKŁADA PEWNĄ WARTOSĆ. TWÓJM ZADANIEM JEST NAPISANIE PROGRAMU (W PSEUDOKOPIE), MOŻLIWEJ NAJOSZCZEDNIEJSZEGO FAKTORYLLOWO, KTÓRY WYKŁADA TĄ SAMĄ WARTOSĆ.

```
while |A| > 1 do
    a ← losowy element z A
    A ← A \ {a}
    b ← losowy element z A
    A ← A \ {b}
    A ← A ∪ {a - b}
output (x mod 2), gdzie x jest elementem zbioru A
```

Wynikiem powyższego algorytmu będąte liczba  $\% 2$ . Zatem mamy dwa możliwe wyniki  $\rightarrow 1$  lub  $0$ .

W każdym obracie pętli wybieramy dwa element i usuwamy, ale dodajemy do zbioru jeden element, który jest różnicą dwóch usuniętych elementów.

Skorzystajmy z właściwości

$$(a - b) \text{ mod } 2 = (\underbrace{a \text{ mod } 2}_{\text{dla } 0 \text{ albo } 1} - \underbrace{b \text{ mod } 2}_{\text{dla } 0 \text{ albo } 1}) \text{ mod } 2$$

czyli

$$\text{jeśli } m, n \in \{0, 1\} \text{ to } (m - n) \text{ mod } 2 = m \text{ xor } n$$

Zatem  $(a - b) \text{ mod } 2 = (a \text{ mod } 2) \text{ xor } (b \text{ mod } 2)$

Dowiedzmy więc każdy element nie 0 lub 1 w zbioreczności od tego, czy odpowiadają jest on parzysty lub nieparzysty.

W algorytmie możemy wprowadzić ustalenie, że  $a, b$  będą elementem dodanym w poprzedniej pętli.

W takim momencie wiemy że  $b \bmod 2$ .

Zatem

$$(a-b) \bmod 2 = (a \bmod 2) \text{ xor } (b \bmod 2) = \\ = (a \bmod 2) \text{ xor } b$$

PROGRAM:

$b \leftarrow$  bieżący element  
 $A \leftarrow A \setminus b$   
 $b \leftarrow b \bmod 2$

while  $|A| > 0$

$a \leftarrow$  bieżący element  
 $A \leftarrow A \setminus a$   
 $b \leftarrow (a-b) \bmod 2$

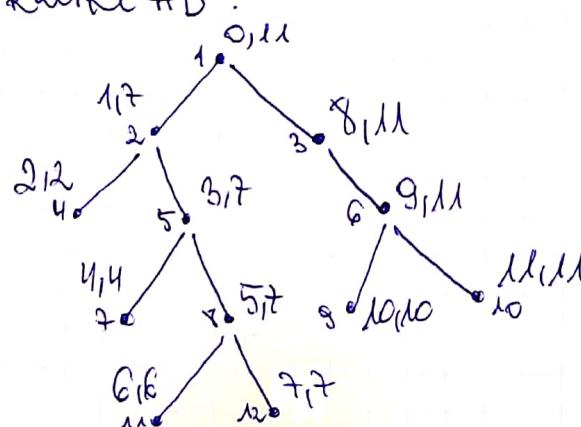
return  $b$

KONCOWY  
JEDNAK  
NIE KORZYSTAM

ZAD. 4 WŁÓŻ ALGORYTM, KTÓRY DLA DRzewA  
 $T = (V, E)$  ORAZ LISTY PAR WIERCHOTOKÓW  
 $(v_i, u_i)$  ( $i = 1, \dots, m$ ), SPRAWDZA, CZY  $v_i$  LEŻY  
NA SCIEZCE Z  $u_i$  DO KORZENIA. PRZYGMIJ  
ZE DRZEWO ZADANE JEST JAKO LISTA  $m-1$   
KRAWĘDZI  $(p_i, o_i)$ , TAKI CH, ŻE  $p_i$  JEST OJCEM  
 $o_i$  W DRZEWIE.

Predstawimy drzewo za pomocą macierzy  
sąsiedztwa. Aby określić czy wierzchołek  $u$   
znajduje się na ścieżce wierzchołka  $v$  do  
korzenia, każdemu wierzchołkowi nadajmy  
dwie dodatkowe parametry - czas wejścia oraz  
czas wyjścia.

PRZESKIAD:



1	2,13
2	4,15
3	6
4	
5	7,18
6	8,10
7	
8	11,12
9	
10	
11	
12	

Jeśli  $v$  jest potomkiem  $u$  to (czas wejścia  
 $v >$  czas wejścia  $u$ ) oraz (czas wyjścia  $v >$   
czas wyjścia  $u$ )

Dokonajmy zmiany reprezentacji na  
reprezentacje w postaci listy sąsiedstwa.

Chang-Mep ( lista par (ojciec, dziecko));  
for pera in lista per (ojciec, dziecko):  
• dodaj dziecko do listy ojca  
• zmień wartość w tablicy pod  
indeksem dziecko na 1

tablica  
wygenerowana

żeromii, kalka, dawemny, sie z nimi  
który wiez-  
chotek jest  
koreniem

c2cs = 0

time (wierzchołek, c2cs)

- oznacz wierzchołek w tablicy jako  
odwiedzony
  - c2cs\_wjścia = c2cs → dla wierzchołka
- for v in lista sąsiedstwa wierzchołka:  
jeśli v jest nieodwiedzony  
c2cs++
- time (v, c2cs)
- c2cs\_wyjścia = c2cs → dla  
wierzchołka

on-the-path (v, u)

jeśli (c2cs\_wjścia u > c2cs\_wyjście v) ||  
(c2cs\_wyjście u ≤ c2cs\_wjście v)  
zwrot true  
zwrot false

koren = 0

D = chang-Mep(D)

time (koren)

ZAD. 5 POKAZ W JAKI SPOSOB ALGORYTM  
"MACIERZOWY" OBLCZANIA N-TEJ LICZBY  
FIBONACCIEGO MOŻNA UOGÓLNIĆ NA INNE CIĄGI,  
W KTÓRICH KOLEJNE ELEMENTY DEFINIWANE  
SĄ UNIOWA KOMBINACJA SKONCZONEJ LICZBY  
ELEMENTÓW WCIĘŚNIEJSZYCH. NASTĘPNIE UOGÓLNIJ  
SWOJE ROZWIAZANIE NA PRZYPADEK, W KTÓRYM  
N-TY ELEMENT CIĄGU DEFINIWANY JEST JAKO SUMA  
KOMBINACJI UNIOWEJ SKONCZONEJ LICZBY ELEMENTÓW  
WCIĘŚNIEJSZYCH DRAZ WIELOMIANU ZMIENNIĘ N.

Metoda "macierzowa" obliczanie n-tej liczby  
fibonacciego:

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} f_i \\ f_{i+1} \end{bmatrix} = \begin{bmatrix} f_{i+1} \\ f_{i+2} \end{bmatrix}$$

$$\text{Stąd } \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{n-1} \cdot \begin{bmatrix} f_0 \\ f_1 \end{bmatrix} = \begin{bmatrix} f_{n-1} \\ f_n \end{bmatrix}$$

Szukamy macierzy A takiej, że:

$$A \cdot \begin{bmatrix} \alpha_k \\ \alpha_{k+1} \\ \vdots \\ \alpha_{k+l-1} \end{bmatrix} = \begin{bmatrix} \alpha_{k+1} \\ \alpha_{k+2} \\ \vdots \\ \alpha_{k+l} \end{bmatrix}$$

więc według poleceńie

$$\alpha_{k+l} = d_0 \alpha_k + d_1 \alpha_{k+1} + \dots + d_l \alpha_{k+l}$$

Zatem można powiedzieć, że macierz A jest postaci:

$$A = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ d_0 & d_1 & d_2 & \cdots & d_l \end{bmatrix}$$

$$\begin{bmatrix} \alpha_k \\ \alpha_{k+1} \\ \vdots \\ \alpha_{k+l-1} \\ \alpha_{k+l} \\ \alpha_{k+2} \\ \vdots \\ 0 \\ \vdots \\ 1 \\ d_0 \alpha_k + d_1 \alpha_{k+1} + d_2 \alpha_{k+2} + \dots + d_l \alpha_{k+l-1} \end{bmatrix} \rightarrow \alpha_{k+l}$$

Uwagdujmy przypadek, w którym u -ty element zależy również od jokiego  $\alpha_u$ .

Wtedy  $\alpha_{k+l} = d_0 \alpha_k + d_1 \alpha_{k+1} + \dots + d_l \alpha_{k+l-1} + W(n)$

gdzie  $W(n) = b_0 n^0 + b_1 n^1 + b_2 n^2 + \dots + b_z n^z$

Teraz szukamy macierzy A takiej, że:

$$(n+1)^k = \sum_{i=0}^k \binom{k}{i} n^i$$

$A \cdot \begin{bmatrix} n^1 \\ n^2 \\ \alpha_k \\ \alpha_{k+1} \\ \vdots \\ \alpha_{k+l-1} \end{bmatrix} = \begin{bmatrix} (n+1)^1 \\ (n+1)^2 \\ \alpha_{k+1} \\ \alpha_{k+2} \\ \vdots \\ \alpha_{k+l} \end{bmatrix}$

$A = \begin{bmatrix} \binom{0}{0} & \binom{0}{1} & \cdots & \binom{0}{l} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \binom{1}{0} & \binom{1}{1} & \cdots & \binom{1}{l} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \binom{l}{0} & \binom{l}{1} & \cdots & \binom{l}{l} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ b_0 & b_1 & \cdots & b_2 & b_3 & \cdots & b_l & \cdots & d_m \end{bmatrix}$

# YAD 8. UŁÓŻ ALGORYTM DLA NASTĘPUJĄCEGO PROBLEMU:

## PROBLEM:

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

wynik: wartość współczynnika przy  $x^2$   
 (względem modulo  $m$ ) wielomianu  
 $\underbrace{\dots ((x-2)^2 - 2)^2 \dots - 2^2}_{n \text{ razy}}$

CW 1.10.15.2 ZASTOSOWANIE METODY UŻYTEJ  
 W SŁUŻBOWYM ALGORYTMIE OBLCZANIA N-TEJ  
 LICZBY FIBONACCIEGO DO ROZWIĄZANIA TEGO  
 PROBLEMU?

Niech  $W_0(x) = x$  i  $W_k(x) = (W_{k-1}(x) - 2)^2$

Interesuje nas tylko to co stoi przy  $x^2$ .  
 Zatem interesuje nas jedynie koniunkt wielomianu  $\rightarrow c_k x^2 + b_k x + c_k$

$$\text{dla } k=0 \quad W_0(x) = x \quad \rightarrow a_0=0 \quad b_0=1 \quad c_0=0$$

$$W_1(x) = (x-2)^2 = x^2 - 4x + 4 \quad \rightarrow \begin{matrix} a_1=1 \\ b_1=-4 \\ c_1=4 \end{matrix}$$

Yak otrzymać kolejny współczynnik przy  $x^2$ :

$$(a_k x^2 + b_k x + c_k - 2)^2 = a_k^2 x^4 + 2a_k b_k x^3 +$$

$$(2a_k c_k - 4a_k + b_k^2) x^2 + (2b_k c_k - 4b_k) x + (c_k^2 - 4c_k + 4)$$

Spróbujemy na  $c_{k+1}$

$$a_{k+1} = a_k^2 - 4 \cdot c_k + 4 = (c_k - 2)^2$$

$$c_1 = 0 \quad \rightarrow (0-2)^2 = 4$$

$$c_2 = ? \quad \rightarrow (4-2)^2 = 4$$

Zatem  $a_k = 4$ .

Spróbujemy na  $b_{k+1}$

$$b_{k+1} = 2b_k \cdot c_k - 4b_k = 8b_k - 4b_k = 4b_k$$

$$\begin{matrix} b_0 = 1 \\ b_1 = -4 \\ b_k = -4^k \end{matrix}$$

$$8a_k - 4c_k + 4^{2k} = 4a_k + 16^k$$

Spróbujemy na  $a_{k+1}$

$$a_{k+1} = 2a_k c_k - 4a_k + b_k^2 = 8a_k - 4a_k + (-4^k)^2$$

szukamy A ktore :

$$A \cdot \begin{bmatrix} 0^k \\ 16^k \end{bmatrix} = \begin{bmatrix} 0^{k+1} \\ 16^{k+1} \end{bmatrix}$$

Zatem

$$A = \begin{bmatrix} 4 & 1 \\ 0 & 16 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 4 & 1 \\ 0 & 16 \end{bmatrix}}_n \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0^n \\ 16^n \end{bmatrix}$$

$$\begin{bmatrix} x & ay \\ z & q \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ 0 & 16 \end{bmatrix}$$

$$\begin{bmatrix} x & ay \\ z & q \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \cdot x + 1 \cdot ay \\ 0 \cdot z + 1 \cdot q \end{bmatrix} \Rightarrow ax = ay$$

w takim razie wystarczy obliczyć

$$A^n = \begin{bmatrix} 4 & 1 \\ 0 & 16 \end{bmatrix}^n = \frac{4^{n-1}}{3} \begin{bmatrix} 3 \cdot 4^{n-1} & 1 \\ 0 & 3 \cdot 4^{n-1} \end{bmatrix}$$

Matrix Power  $[4^{n-1}, 1; 0, 3 \cdot 4^{n-1}]$

$$\frac{4^{n-1}}{3} (4^n - 1)$$

skokowe potegowanie

pot(x, m)

jeżeli  $m=1$   
return 1

jeżeli  $m \% 2 == 1$

return  $x \cdot \text{pot}(x, m-1)$

jeżeli  $m \% 2 == 0$   
return  $\text{pot}(x, m/2) * * 2$