Algorytmy i złożoność - ćwiczenia 28.02.2023r.

Przypomnieć sobie algorytm euklidesa (algebra z teorią liczb), zasady indukcji, wykresy podstawowych funkcji elementarnych

Zasada indukcji matematycznej (Silna)

Dane: $T(n)$ - zdanie zależne od n, $n \in \mathbb{N}$, $n_0 \in \mathbb{N}$

Jeżeli: sprawdziłem, że $T(n_0)$ jest prawda ① i $\forall_{n \geq n_0} [T(n_0), T(n_0+1), \ldots, T(n) \Rightarrow T(n+1)]$ ②, to $\forall_{n \geq n_0} T(n)$

②' $\forall_{n \geq n_0} [T(n) \Rightarrow T(n+1)]$

(Zwykła) zasada indukcji matematycznej:

Dane: $T(n)$ - zdanie zależne od n, $n \in \mathbb{N}$, $n_0 \in \mathbb{N}$

Jeżeli: $T(n_0)$ jest prawda ① i $\forall_{n \geq n_0} [T(n) \Rightarrow T(n+1)]$ ②', to $\forall_{n \geq n_0} T(n)$

$$\text{„}\underbrace{T(n_0), T(n_0+1), \ldots, T(n)}_{\text{więcej}}\text{"} \Rightarrow \text{„}\underbrace{T(n)}_{\text{mniej}}\text{"}$$

② więcej $\Rightarrow T(n+1)$
②' mniej $\Rightarrow T(n+1)$
} ②' jest silniejsze niż ②

Silna Z. I.: $\underline{① \wedge ②} \Rightarrow \forall_{n \geq n_0} T(n)$ ← słabsze założenia

Zwykła Z. I.: $\underline{① \wedge ②'} \Rightarrow \forall_{n \geq n_0} T(n)$ ← silniejsze założenia

Podsumowując: ① ∧ ② ⇒ $\forall_{n \geq n_0} T(n)$ - silniejsza ① ∧ ②' ⇒ $\forall_{n \geq n_0} T(n)$ - słabsza

② $\forall_{n \geq n_0} [\underbrace{T(n_0), T(n_0+1), \ldots, T(n)}_{\forall_{n_0 \leq k \leq n} T(k)} \Rightarrow T(n+1)]$

② $\forall_{n \geq n_0} [(\forall_{n_0 \leq k \leq n} T(k)) \Rightarrow T(n+1)]$ niech $m = n+1$, $n = m-1$. Wtedy

② $\forall_{m > n_0} [(\forall_{n_0 < k \leq m-1} T(k)) \Rightarrow T(m)]$ ② $\forall_{m > n_0} [(\forall_{n_0 \leq k < m} T(k)) \Rightarrow T(m)]$

② $\forall_{n \geq n_0} [(\forall_{n_0 \leq k < n} T(k)) \Rightarrow T(n)]$

Silna zasada indukcji matematycznej (inna wersja)

Dane: $T(n)$ - zdanie zależne od n, $n \in \mathbb{N}$, $n_0 \in \mathbb{N}$

Jeżeli: ① $T(n_0)$ jest prawda i ② $\forall_{n > n_0} [(\forall_{n_0 \leq k < n} T(k)) \Rightarrow T(n)]$,

to $\forall_{n \geq n_0} T(n)$ jest prawda.

Ciąg liczb Fibonacciego:

$F_0 = 0$, $F_1 = 1$, $\forall_{n \geq 2}$ $F_n = F_{n-1} + F_{n-2}$

n	$(F_n)^2$	$\sum_{i=0}^{n} (F_i)^2$
0	0^2	0
1	1^2	1
2	1^2	2
3	2^2	6
4	3^2	15

n	$(F_n)^2$	$\sum_{i=0}^{n} (F_i)^2$
5	5^2	40
6	6^2	104
7	7^2	273

n	$F_n \cdot F_{n+1}$
0	$0 \cdot 1 = 0$
1	$1 \cdot 1 = 1$
2	$1 \cdot 2 = 2$
3	$2 \cdot 3 = 6$
4	$3 \cdot 5 = 15$

n	$F_n \cdot F_{n+1}$
5	$5 \cdot 8 = 40$
6	$8 \cdot 13 = 104$
7	$13 \cdot 21 = 273$

Hipoteza: $T(n): \sum_{i=0} F_i^2 = F_n \cdot F_{n+1}$

$\forall_{n\geq 1}\ F_{n+1} = F_n + F_{n-1}$, $F_0 = 0$, $F_1 = 1$

Chcemy wykazać że $\forall_{n\geq 0}\ T(n)$.

Silna Z.I.

zał. ① $T(n_0)$ ② $(\forall_{k<n}\ T(k)) \Rightarrow$

Teza: $\forall_n\ T(n)$

W tym celu sprawdzimy ① i ② z silnej Z.I.. Wtedy na mocy Z.I.M.: $\forall_n\ T(n)$.

Ad. ①: $n_0 = 0$ $T(0)$: $F_0^2 = F_0 \cdot F_1$ (?)
$0^2 = 0 \cdot 1$ (?) } Prawda

Ad ②: Mamy sprawdzić że $\forall_{n\geq n_0} [\boxed{\forall_{k<n}\ T(k)} \Rightarrow T(n)]$ (?)

Niech $n > 0$. Zakładam, że $\forall_{k<n}\ T(k)$ jest prawdą. To znaczy, że $\forall_{k<n} \sum_{i=0}^{k} F_i^2 = F_k \cdot F_{k+1}$. Chcę wykazać, że $T(n)$ jest prawdą, tzn. $\sum_{i=0}^{n} F_i^2 = F_n \cdot F_{n+1}$ (?).

Założenia (indukcyjne): $n > 0$, $\forall_{k<n} \sum_{i=0}^{k} F_i^2 = F_k \cdot F_{k+1}$. ⊛

Chcę (Teza indukcyjna): $\sum_{i=0}^{n} F_i^2 = F_n \cdot F_{n+1}$ (?)

Dowód Tezy indukcyjnej:

$F_n \cdot F_{n+1} \underset{\uparrow}{=} F_n \cdot (F_n + F_{n-1}) = (F_n)^2 + F_n \cdot F_{n-1} \underset{\uparrow}{=} (F_n)^2 + F_{m+1} \cdot F_m \underset{\uparrow}{=}$

korzystamy z faktu że $n \geq 1$ Niech $m = n-1$, $m < n$ korzystamy z ⊛

$= (F_n)^2 + \sum_{i=0}^{m} (F_i)^2 = (F_n)^2 + \sum_{i=0}^{n-1} (F_i)^2 = \sum_{i=0}^{n} (F_i)^2$

Na mocy zasady indukcji matematycznej $\forall_{n\geq 0} \sum_{i=0}^{n} (F_i)^2 = F_n \cdot F_{n+1}$

Zadania (częściowo domowe):

1) Wykazać, że $\forall_{n>0\ (n_0)} [\sum_{i=0}^{n} f_i = f_{n+2} - 1]$

2) Ciąg liczb Fibonacciego o ujemnych indeksach

Ma to być taki ciąg, który spełnia: $F_{-n} + F_{-n+1} = F_{-n+2}$.

Zdefiniować: $F_{-1}, F_{-2}, \ldots$. Zaproponować (i udowodnić) związek między F_{-n}, a F_n.

✓ 3) Wykazać, że $\forall_{n\geq 1}\ F_{2n-1} = (F_{n-1})^2 + (F_n)^2$

$\forall_{n\geq 1}\ F_{2n} = F_n (2F_{n-1} + F_n)$

4) Wykazać, że $\forall_{n\geq 1} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} F_{n-1} \\ F_n \end{bmatrix}$

Ad. 4) $T(n): \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} F_{n-1} \\ F_n \end{bmatrix}$

Indukcja na n:

① Dla $n=1$ (np.): $\begin{bmatrix}0&1\\1&1\end{bmatrix}^1 \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}F_0\\F_1\end{bmatrix}$ (?)

$\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$ (?) – TAK jest to prawda

② Dla $n \geq 1$: $T(k): \begin{bmatrix}0&1\\1&1\end{bmatrix}^k \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}F_{k-1}\\F_k\end{bmatrix}$, $\underline{k<n}$ $\rightarrow 1 \leq k < n$

Teza indukcyjna: $T(n): \begin{bmatrix}0&1\\1&1\end{bmatrix}^n \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}F_{n-1}\\F_n\end{bmatrix}$

Dowód tezy indukcyjnej:

$k = n-1$

$k = n-1 > 0 \Rightarrow \underline{k \geq 1}$

$L = \begin{bmatrix}0&1\\1&1\end{bmatrix}^n \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0&1\\1&1\end{bmatrix}^{1+(n-1)} \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0&1\\1&1\end{bmatrix}^1 \cdot \begin{bmatrix}0&1\\1&1\end{bmatrix}^{n-1} \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0&1\\1&1\end{bmatrix} \cdot \left(\begin{bmatrix}0&1\\1&1\end{bmatrix}^k \cdot \begin{bmatrix}1\\0\end{bmatrix}\right) =$ z zał. ind.

$= \begin{bmatrix}0&1\\1&1\end{bmatrix} \cdot \begin{bmatrix}F_{k-1}\\F_k\end{bmatrix} = \begin{bmatrix}F_k\\F_{k-1}+F_k\end{bmatrix} = \begin{bmatrix}F_k\\F_{k+1}\end{bmatrix} = \begin{bmatrix}F_{n-1}\\F_n\end{bmatrix} = P$

Na mocy zasady indukcji matematycznej $\forall_{n \geq 1}\ T(n)$.

Algorytmy i złożoność – wykład 01.03.2023 r.

Notacja O. (Obliczeniowa)

$\sum_{i=0}^{n} i = 0+1+\ldots+n = \frac{n(n+1)}{2}$ $\quad \sum_{i=0}^{n} i^2 = 0+1+2^2+3^2+\ldots+n^2 = \frac{n(n+1)(2n+1)}{6}$

$\sum_{i=0}^{n} q^n = \frac{1-q^{n+1}}{1-q}$

A – pewien algorytm, x – dane, $|x|$ – długość danych

$T_A(x)$ – liczba kroków, który nasz algorytm A wykonuje na danych x do zatrzymania się na każdych danych.

$T_A: \mathbb{N} \rightarrow \mathbb{R}_+ = (0, +\infty)$ $\quad T_A(n) = \max\{T_A(x) : |x| = n\}$,

Definicja 1: Niech $f, g: \mathbb{N} \rightarrow \mathbb{R}_+$ (ciągi liczbowe). Mówimy że f jest klasy O od funkcji g i piszemy $f \in O(g)$ ($f = O(g)$) wtw, gdy $\exists c>0\ \exists N \in \mathbb{N}\ \forall n \in \mathbb{N}\ [n > N \Rightarrow f(n) \leq c \cdot g(n)]$

Przykład: $f_1(n) = 5n$, $f_2(n) = \frac{1}{3}n$, $g(n) = n$; $f_1 \in O(g)$, $f_2 \in O(g)$, $g \in O(g)$

Oznaczenie: $5n \in O(n)$ zamiast „$f_1 \in O(g)$, gdzie $f_1(n) = 5n$, $g(n) = n$

Obserwacja: 1) $f \in O(f)$; 2) Jeśli $\forall n\ f(n) \leq g(n)$ to $f \in O(g)$

$\langle f(n) \leq 1 \cdot f(n) \Rightarrow \exists \underset{1}{c}>0\ \exists \underset{0}{N}\ \forall n>N\ f(n) \leq c \cdot f(n)$ $\square$[1]

$f(n) \leq 1 \cdot g(n) \Rightarrow \exists \underset{1}{c}>0\ \exists \underset{0}{N}\ \forall n>N\ f(n) \leq c \cdot g(n)$ $\square$[2] $\rangle$

Oznaczenie: „$\forall! n$" – dla prawie wszystkich n „$\exists N\ \forall n>N$".

Oznaczenie: Zamiast $f \in O(g)$ piszemy $f \preccurlyeq g$.

Oznaczenie: Zamiast $f \in O(g)$ piszemy $f \preccurlyeq f$ ($f \in O(f)$)

Wniosek: Relacja $\preccurlyeq$ jest zwrotna, czyli $f \preccurlyeq f$ ($f \in O(f)$)

Stwierdzenie: Relacja $\preccurlyeq$ jest przechodnia.

Dowód: Należy wykazać że dla $f, g, h: \mathbb{N} \rightarrow \mathbb{R}_+$, jeżeli $f \preccurlyeq O(g)$ i $g \preccurlyeq O(h)$, to $f \preccurlyeq O(h)$.

$f \in O(g)$ i $g \in O(h)$, to $f \in O(h)$

Założenia: 1) $f \in O(g)$ i $g \in O(h)$, 2) $\exists c_1>0\ \exists N_1 \in \mathbb{N}\ \forall n>N_1\ f(n) \leq c_1 \cdot g(n)$

3) $\exists c_2>0\ \exists N_2 \in \mathbb{N}\ \forall n>N_2\ \ g(n) \leq c_2 \cdot h(n)$

Niech c_1, c_2, N_1, N_2 będą takie jak w 2) i 3). Rozważmy $n > \max\{N_1, N_2\}$. Wtedy $n>N_1$ i $n>N_2$. Z 2) i 3) otrzymujemy, że: $f(n) \leq c_1 \cdot g(n) \leq \overset{c_1>0}{c_1} \cdot c_2 \cdot h(n)$

$c = c_1 \cdot c_2 > 0$; $c_1, c_2 > 0 \longrightarrow f(n) \leq c \cdot h(n)$

Mamy więc $\exists \underset{c_1 \cdot c_2}{c>0}\ \exists \underset{\max\{N_1, N_2\}}{N \in \mathbb{N}}\ \forall n>N\ f(n) \leq c \cdot h(n)$. Zatem $f \in O(h)$ □

Dowód: Twierdzenie: Niech $f_1, f_2, g, h : \mathbb{N} \to \mathbb{R}_+$.

1. Jeżeli $f \in O(g)$ to $g+f \in O(g)$
2. Jeżeli $f_1 \in O(g)$, $f \in O(h)$ to $f_1 \cdot f \in O(g \cdot h)$

Obserwacja: Relacja $\preccurlyeq$ nie jest ani symetryczna ani antysymetryczna

Przykład: (kontrprzykład na antysymetrię)

$f(n) = 5n$, $g(n) = n$. Wtedy $f \in O(g)$, $g(n) \leq f(n)$, $g \in O(f)$. Mamy $f \preccurlyeq g$ i $g \preccurlyeq f$, ale $f \neq g$. Zatem $\preccurlyeq$ nie jest relacją częściowego porządku.

Przykład: (kontrprzykład na symetrię)

$f(n) = n$, $g(n) = n^2$. Wtedy $f(n) \leq g(n)$, $f \preccurlyeq g$. Pokażemy że $g \not\preccurlyeq f$.

Dla dowodu nie wprost załóżmy, że $g \preccurlyeq f$.

Z definicji 1: $\exists c>0\ \exists N \in \mathbb{N}\ \forall n>N\ \ g(n) \leq c \cdot f(n)$, weźmy to c i to N. Niech $n>N$.

$g(n) \leq c \cdot f(n) \overset{f(n)=n>0}{\Longrightarrow} \frac{g(n)}{f(n)} \leq c \Rightarrow \frac{n^2}{n} \leq c \Rightarrow n \leq c$

Otrzymaliśmy $\forall n>N\ \ n \leq c$. Wystarczy więc $n = N+c+1 > N$ $\Rightarrow n = -1 > c \Rightarrow n > N$ i $\neg(n \leq c)$ – sprzeczność. Zatem $g \not\preccurlyeq f$. □

Definicja 2: Niech f, g będzie jak wyżej. Mówimy że f jest klasy $\Theta(g)$ i piszemy $f \asymp g$ wtw, gdy $f \in O(g)$ i $g \in O(f)$.

Wnioski: Relacja $\asymp$ jest symetryczna.

Ze stwierdzenia na temat własności relacji $\preccurlyeq$ wynika, że $\asymp$ jest też zwrotna i przechodnia. Zatem relacja $\asymp$ jest relacją równoważności.

$[f]_{\asymp} = \{ g : \mathbb{N} \to \mathbb{R}_+ : f \in O(g) \wedge g \in O(f) \}$ = klasa Θ funkcji $f = \Theta(f)$

(w praktyce stosuje się terminologię „klasa O" zamiast „klasa Θ").

Funkcja logarytmiczna:

Oznaczenie: $\log_2 x = \lg x$, $\log_{10} x = \log x$, $\log_2 x = \frac{\log_{10} x}{\log_{10} 2} = \frac{1}{\log_{10} 2} \cdot \log_{10} x = \log_2 10 \cdot \log_{10} x$

$\lg x = c \cdot \log x$, $c > 0$ $(c = \log_2 10 = \lg 10)$

$\exists c > 0 \ \forall n \in \mathbb{N}_+ \ \lg n \leq c \cdot \log n$ i $\exists c_1 > 0 \ \forall n \in \mathbb{N}_+ \ \log n \leq c_1 \cdot \lg n$

Wniosek: $\lg n \asymp \log n$

Obserwacja: Niech $a, b > 1$, wtedy $\log_a n \in \Theta(\log_b n)$

Uwaga! $f \in \Theta(g) \iff g \in \Theta(f)$

Twierdzenie: Niech f, g będzie jak wyżej. $(g(n) \neq 0)$. Dowód: 1) i 3)

1) Jeżeli: $\exists g \in \mathbb{R} \setminus \{0\}$ $\lim_{n \to \infty} \frac{f(n)}{g(n)} = g$, to $f \in \Theta(g)$.

2) Jeżeli: $\lim_{n \to \infty} \frac{f(n)}{g(n)} = +\infty$, to $g \in O(f)$, ale $f \notin O(g)$.

3) Jeżeli: $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$, to $f \in O(g)$ ale $g \notin O(f)$.

Notacja: Jeśli $f \in O(g)$ ale $g \notin O(f)$, to stosujemy zapis: $f \prec g$.

Dowód: Ad. 2) $\forall A > 0 \ \forall^{\infty} n \ \frac{f(n)}{g(n)} \geq A \Rightarrow f(n) \geq A \cdot g(n) \Rightarrow g(n) \leq \frac{1}{A} \cdot f(n)$.

np. dla $A = 1$ $\forall^{\infty} n \ g(n) \leq \underset{c}{1} \cdot f(n) \Rightarrow g \in O(f)$.

Pokażemy teraz, że $f \notin O(g)$. Dla dowodu nie wprost: $f \in O(g)$.

Z definicji 1: $\exists c > 0 \ \exists N \in \mathbb{N} \ \forall n > N \ f(n) \leq c \cdot g(n)$

$\forall^{\infty} n \ \frac{f(n)}{g(n)} \leq c$, ale $\lim_{n \to \infty} \frac{f(n)}{g(n)} \leq c$ – sprzeczność, czyli $f \notin O(g)$ □.

Wniosek: $\left. \begin{array}{l} f(n) = a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + \ldots + a_1 \cdot n + a_0, \ a_k \neq 0 \\ g(n) = b_k \cdot n^k + b_{k-1} \cdot n^{k-1} + \ldots + b_1 \cdot n + b_0, \ b_k \neq 0. \end{array} \right\}$ założenia

Tezа: $f \in \Theta(g)$. □

Wniosek: Dowolny wielomian stopnia k jest klasy Θ funkcji n^k.

$a_k \cdot n^k + a_{k-1} \cdot n^{k-1} + \ldots + a_1 \cdot n + a_0 \in \Theta(n^k)$ np: $5n^2 - 10n + 17 \in \Theta(n^2)$

Algorytmy i złożoność – ćwiczenia 07.03.2023 r.

Zadanie 1: Udowodnić że 1) $F_{2n-1} = F_{n-1}^2 + F_n^2$ i 2) $F_{2n} = F_n(2F_{n-1} + F_n)$.

① Biorę $n_0 = 1$. 1) $T(n_0)$: $F_1 = F_0^2 + F_1^2$ (?) 2) $T(n_0)$: $F_2 = F_1(2 \cdot F_0 + F_1)$ (?)

$1 = 0^2 + 1^2$ (?) $1 = 1(2 \cdot 0 + 1)$ (?)

PRAWDA $1 = 1$ PRAWDA $1 = 1$

Zatem $T(n_0)$ jest prawdziwe

② Niech $n > 1$ zał. ind.: 1) $T(k)$: $F_{2k-1} = F_{k-1}^2 + F_k^2$ 2) $T(k)$: $F_{2k} = F_k(2 \cdot F_{k-1} + F_k)$

Teza ind.: 1) $T(n)$: $F_{2n-1} = F_{n-1}^2 + F_n^2$ 2) $T(n)$: $F_{2n} = F_n(2F_{n-1} + F_n)$

Dowód Tezy indukcyjnej:

1) $L = F_{2n-1} \underset{k=n-1,\ k<n,\ n=k+1}{=} F_{2k+1} = F_{2k} + F_{2k-1} \underset{F_n = F_{n-1} + F_{n-2}}{=} F_k(2F_{k-1} + F_k) + F_{k-1}^2 + F_k^2 \underset{\text{zał.}}{=}$

$= \underline{2F_k \cdot F_{k-1}} + \underline{F_k^2} + \underline{F_{k-1}^2} + F_k^2 = (F_k + F_{k-1})^2 + F_k^2 = F_{k+1}^2 + F_k^2 =$

$= F_n^2 + F_{n-1}^2 = P$

2) $L = F_{2n} \underset{n=k+1}{=} F_{2k+2} = F_{2k} + F_{2k+1} = F_{2k} + F_{2k} + F_{2k-1} = 2F_{2k} + F_{2k-1} \overset{\text{zał.}}{=}$

$= 2(F_k(2F_{k-1} + F_k)) + (F_{k-1})^2 + (F_k)^2 = \underline{4F_k \cdot F_{k-1}} + 2F_k^2 + \underline{F_{k-1}^2} + \underline{F_k^2} =$

$= (F_k + F_{k-1})^2 + 2F_k \cdot F_{k-1} + 2F_k^2 = (F_{k+1})^2 + 2F_k \cdot F_{k-1} + 2F_k^2 = (F_{k+1})^2 + 2F_k(F$

$= (F_{k+1})^2 + 2F_k \cdot F_{k-1} = F_{k+1}(2F_k + F_{k+1}) = F_n(2F_{n-1} + F_n) = P$

Z zasady indukcji matematycznej $T(n)$ jest prawdziwe.

$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$

Zadanie 3: Lista: $1, n, n^2, n^3, n^4, n^5, \log n, n \log n, n^n, n!, 2^n, 3^n, \pi^n, n^\pi$ ← funkcje $g(n)$

Zadanie 2: NWD(777, 977) do domu

a) $f(n) = 2^n + 4^n$ b) $f(n) = 3n^4 + 6n^3 - n^2$ c) $f(n) = 3n! - n^4$

d) $f(n) = n + \log n^2$ e) Uszeregujemy funkcje $g(n)$ według

Tw. 1 $f \in O(g) \Rightarrow f + g \in O(g)$

Ad. e) np: $1, n$ $g_1(n) = 1$, $g_2(n) = n$ albo można to zapisać

$f(n) = 1$, $g(n) = n$, czy $f \in O(g)$?, czyli czy

$\exists_{c>0} \exists_{N \in \mathbb{N}} \forall_{n \in \mathbb{N}} [n > N \Rightarrow \underbrace{f(n)}_{1} \le c \cdot \underbrace{g(n)}_{n}]$? – czy istnieje stała $c > 0$, dla którego dla prawie wszystkich n, $f(n)$ jest mniejsze od $c \cdot g$ lub równe

$\exists_{c>0} \exists_{N \in \mathbb{N}} \forall_{n \in \mathbb{N}} [n > N \Rightarrow 1 \le c \cdot n]$?

$1 \le c \cdot n$?

Niech $N = 1$ i $c = 1$. Wtedy $\forall_{n \in \mathbb{N}} [n > N \Rightarrow 1 \le 1 \cdot n]$ TAK, czyli $1 \in O(n)$

Czy $n \le 1$? zakładamy że $n \le 1$ czyli $\exists_{c_1>0} \exists_{N_1 \in \mathbb{N}} \forall_{n > N_1}$, $n \le c_1$ (*)

$\forall_{n > N_1}$, $n \le c_1$. Niech $N_2 = \max\{N_1, c_1\}$, więc $n = N_2 + 1$ zatem

$n > N_2 \wedge n > N_1 \wedge n > c_1$

$n > N_1 \wedge \neg(n \le c_1)$ – sprzeczne z (*)

$\forall! n$ $f(n) \le g(n) \Rightarrow f \in O(g)$ $(f \le g)$

•) n, n^2 – chcemy wykazać że $n < n^2$ (?)

$n \le n^2$ (?) oraz $n \in O(n^2)$ (?), czyli $\exists_{c>0}$ $n \le c \cdot n^2$, $\neg(n^2 \le n)$?

Pytamy czy $n \leq n^2$? czyli czy $n \in O(n^2)$?

$\exists c>0\ \exists N\in\mathbb{N}\ \forall n\in\mathbb{N}\ [n > N \Rightarrow \underbrace{f(n)}_{n} \leq c\cdot \underbrace{g(n)}_{n^2}]$, ale $\forall n\geq 0\ n \leq n^2 \Rightarrow$

$\Rightarrow \exists \underset{1}{c>0}\ \exists \underset{0}{N\in\mathbb{N}}\ \forall n\geq N\ n \leq n^2 \cdot \underset{1}{c}$ co należało dowieść.

$\boxed{n^2 \leq c\cdot n \text{ dla } n \leq c}$

Chcemy wykazać czy $n^2 \notin O(n)$? Hipoteza: $n^2 \in O(n)$

Algorytmy i złożoność - wykład 08.03.2023r.

Oznaczenie: $f \prec g$ wtw, gdy $f \preccurlyeq g \wedge \neg(g \preccurlyeq f)$, równoważnie $f \prec g$ wtw, gdy $f \preccurlyeq g \wedge \neg(f \asymp g)$.

Obserwacje: 1) Jeżeli $\forall n\ f(n) = c$ to $f \in \Theta(1)$. Bowiem $f(n) = c \leq c\cdot 1$. $\exists c>0\ \exists \underset{0}{N}\ \forall n > N\ f(n) \leq c\cdot 1$

$[f] = f \in O(1)$, również $1 \leq \frac{1}{c}\cdot c = \frac{1}{c}\cdot f(n)$, $c_1 = \frac{1}{c} > 0$, czyli $\exists c_1>0\ \forall n\ 1 \leq c_1\cdot f(n)$ więc $1 \in \Omega(f)$ czyli $1 \asymp f$ (tej samej klasy)

Przypomnienie: $f \in O(g) \Rightarrow f+g \in O(g)$ i $f_1 \in O(g) \wedge f_2 \in O(h) \Rightarrow f_1\cdot f_2 \in O(gh)$

$n^2+1 \in O(n^2)$ (, bo $n^2+1 \leq \underset{c}{2}\cdot n^2$), $n \in O(n^2)$, $5 \in O(1)$,

$5n \in O(n^2)$ (?), $f_1 = n$, $g = n^2$, $f = 5$, $h = 1$, czyli $5n \in O(n^2)$,

$n^2+5n \in O(n^2)$, $1 \in O(n^2)$.

Przykład: czy $n^2+5n+1 \in O(n^2)$? $n^2 \in O(n^2)$ i $5n+1 \in O(n^2)$, czyli zatem $n^2+5n+1 \in O(n^2)$.

$n^2 \in O(n^2+5n+1)$, bo $n^2 \leq n^2+5n+1$ zawsze ("$\leq$ $\subseteq$ $\preccurlyeq$")

Obserwacja: $|an^2+bn+c| \in O(n^2)$

Przypomnienie: Jeżeli $\exists g\in\mathbb{R}\setminus\{0\}\ \lim_{n\to\infty}\frac{f(n)}{g(n)} = g$ to $\overbrace{f\in\Theta(g)}^{f\in O(g)\wedge g\in O(f)}$

Stąd: $\lim_{n\to\infty}\frac{|an^2+bn+c|}{n^2} = |a| \neq 0$ czyli $|an^2+bn+c| \in O(n^2)$

ale też na odwrót $n^2 \in O(|an^2+bn+c|)$. Więc $n^2 \asymp an^2+bn+c$, $a>0$

Ogólnie, $W(n) = a_0n^k + a_1n^{k-1} + \ldots + a_{k-1}n + a_k$; $a_0 > 0 \Rightarrow$

$\Rightarrow W(n) \in \Theta(n^k)$

Stwierdzenie: $W(n)$ jak wyżej oraz $P(n) = b_0n^l + b_1n^{l-1} + \ldots + b_{l-1}n + b_l$,

$b_0 > 0$. $\Rightarrow$ Jeśli $l < k$ to $P \in O(W)$ i $W \notin O(P)$, bo:

$$\frac{P(n)}{W(n)} = \frac{b_0n^l + \ldots + b_l}{a_0n^k + \ldots + a_k} = \frac{b_0 + \frac{b_1}{n} + \ldots + \frac{b_l}{n^l}}{a_0\cdot n^{k-l} + \ldots + \frac{a_k}{n^l}} \xrightarrow[n\to\infty]{k-l>0} 0$$, czyli $P \prec W$.

$l < k \Rightarrow b_0n^l + \ldots + b_l \in O(a_0n^k + \ldots + a_k)$, $b_0, a_0 > 0$

$l = k \Rightarrow b_0n^l + \ldots + b_l \in \Theta(a_0n^k + \ldots + a_k)$

$l > k \Rightarrow a_0n^k + \ldots + a_k \in O(b_0n^l + \ldots + b_l)$

Obserwacja: $1 < n < n^2 < n^3 < \dots < n^k < n^l < \dots$, gdzie $3 < k$

Obserwacja: $n^2 < 2^n$, $n^k < 2^n$, $k \in \mathbb{R}_+$

Stwierdzenie: $2^n < 3^n$, bo $\lim_{n\to\infty} \frac{2^n}{3^n} = \lim_{n\to\infty} (\frac{2}{3})^n = 0$. Stąd $2^n < 3$

Stwierdzenie: $a^n < b^n$, dla $1 < a < b$.

Wniosek: $(1,1)^n < (1,2)^n < 2^n < 3^n < \pi^n < 4^n$ ... odpowiednio go między

Stwierdzenie: $n^k < a^n$, $a > 1$, $k \in \mathbb{N}$ np: $n^{100} < (1,1)^n$

Przypomnienie: $\log_a n \times \log_b n$, $a, b > 1$

Oznaczenie: $\lg n = \log_2 n$ oraz $\log_{10} n = \log n$.

Dowód: Stwierdzenie: $1 < \lg n < n$ (oraz $n < n \lg n < n^2$) (patrz stwierdzenie przypomniane wyżej)

do domowe Zadanie: a) wykonać wykresy funkcji: $1, n, n^2, 2^n, \lg n$ w jednym układzie współrzędnych i zaobserwować ow... wione nierówności.

b) wykonać wykresy funkcji: $25, n, n^2, \frac{n^3}{100}$, na jednym ukł... współ.

c) wykonać wykresy funkcji: $1, \lg n, n, n \lg n, \frac{n^2}{10}$ na jednym ukł... współ.

Zadanie 3 (z ćwiczeń): $1 < \lg n < n < n \lg n < n^2 < n^3 < \dots < (1,1)^n < 2^n < 3^n < \dots$

Dowód: $\dots < n! < n^n$ (umieć uzasadnić) udowodnić $3^n \le n!$ lub $n! \le n^n$ dla dowolnego jakiegoś n b.

---

n – długość danych, $f(n)$ – liczba podstawowych kroków dł... czasowych które algorytm wykonuje na danych długości n.

Zakładamy że sprzęt wykonuje $1000000 = 10^6$ pojedynczych kroków na sekundę.

$f(n) \backslash n$	10	100	1000	5000
25	$25 \cdot 10^{-6}$ s	$25 \cdot 10^{-6}$ s	$25 \cdot 10^{-6}$ s	$25 \cdot 10^{-6}$ s
$\lg n$	$3{,}3 \cdot 10^{-6}$ s	$6{,}6 \cdot 10^{-6}$ s	$9{,}9 \cdot 10^{-6}$ s $\approx 10^{-5}$ s	$1{,}2 \cdot 10^{-5}$ s.
n	10^{-5} s	10^{-4} s	10^{-3} s	$5 \cdot 10^{-3}$ s
$n \lg n$	$3{,}3 \cdot 10^{-5}$ s	$6{,}6 \cdot 10^{-4}$ s	10^{-2} s	$6 \cdot 10^{-2}$ s
n^2	10^{-4} s	10^{-2} s	1 s	25 s
$2n^2+5n+10$	$3{,}5 \cdot 10^{-4}$ s	0,027 s		50 s

$\frac{n^3}{100}$	10^{-5} s	10^{-2} s $= 0{,}01$ s	10 s	27 min
2^n	10^{-7} s	$1{,}6 \cdot 10^{23}$ min $= 4 \cdot 10^{16}$ lat		$4{,}5 \cdot 10^{1497}$ lat

W tabeli znajduje się czas działania algorytmu na danych o długości n przy „złożoności obliczeniowej" $f(n)$.

Pomocnicze obliczenia: ($\approx$ jest różne od $\approx$)

$\lg \approx 3{,}3$; $2^x = 10$ $(2^3 < 10 < 2^4)$; $\lg n^2 = 2 \lg n$,

$\log_2 100 \approx 6{,}6$, $\lg 5000 = \log(5 \cdot 10^3) = \log 5 + \lg 10^3 \approx 2 + 9{,}9 \approx 12$

$n = 10 \Rightarrow 10 \cdot \lg 10 \approx 3{,}3 \cdot 10$; $3{,}3 \cdot 10 \cdot 10^{-6} = 3{,}3 \cdot 10^{-5}$ $(n \lg n)$

$n = 100 \Rightarrow 100 \cdot \lg 100 \approx 100 \cdot 6{,}6$, $100 \cdot 6{,}6 \cdot 10^{-6} = 6{,}6 \cdot 10^{-4}$ $(n \lg n)$

$\frac{n^3}{100}$ dla $n = 5000 \Rightarrow \frac{(5 \cdot 10^3)^3}{10^2} = \frac{5^3 \cdot 10^9}{10^2} = 125 \cdot 10^7$; $125 \cdot 10^7 \cdot 10^{-6} = 1250 =$
$= 1{,}25 \cdot 10^3$ (s) $\Rightarrow \frac{1250}{60} \approx 21$ min

2^n dla $n = 10 \Rightarrow 2^{10} = 1024 \approx 10^3$; $10^3 \cdot 10^{-6} = 10^{-3}$

2^n dla $n = 100 \Rightarrow 2^{100} = (2^{10})^{10} \approx (10^3)^{10} = 10^{30}$; $10^{30} \cdot 10^{-6} = 10^{24}$ s
$\frac{10^{24}}{60} = \frac{10^{23}}{6} \approx 16 \cdot 10^{22} \approx 1{,}6 \cdot 10^{23}$ min

Wnioski: Algorytmy wykonujące 2^n kroków obliczeniowych na danych długości n są niepraktyczne (tym bardziej $n!$, n^n, ...). Takie algorytmy określamy jako „nieefektywne". Złożoność obliczeniowa algorytmu, to klasa Θ funkcji wyrażającej maksymalną liczbę podstawowych „kroków" obliczeniowych, które algorytm wykonuje na danych długości n.

$z_A = f : \mathbb{N} \to \mathbb{R}^+$, $\exists g : \mathbb{N} \to \mathbb{R}^+$ $g(n) = \max\{$liczba podst. obliczeń wykonanych przez A na danych o długości $n\}$. $g \in \Theta(f)$

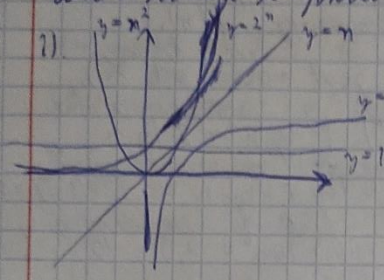
np: $g(n) = \frac{n^2}{2} + 2n + 7$, to za f bierzemy n^2.

Jeśli złożoność obliczeniowa algorytmu A wynosi $\Theta(n^k)$, dla $k \in \mathbb{N}$, to mówimy, że A jest efektywne.

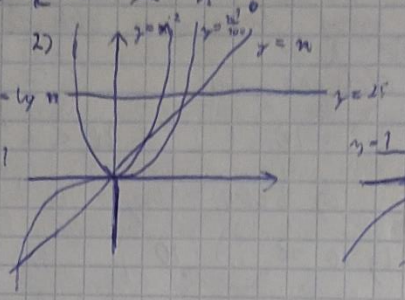
Dokładniej: gdy złożoność wynosi $g(n)$, gdzie $g \in O(n^k)$, $k \in \mathbb{N}$ to algorytm A jest efektywny. np: złożoność wynosi $n \lg n \in O(n^2)$ to algorytm jest efektywny.

Klasa P problemów to klasa wszystkich tych zagadnień algorytmicznych (problemów) do rozwiązania których istnieje algorytm o złożoności n^k, dla $k \in \mathbb{N}$ lub mniejsze wielomian stopnia k $\times$ n^k

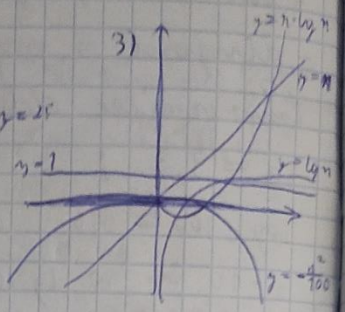
1).



2)



3)



NWD (1111, 977) = ... 1

1111 : 977 = 1 r. 134 → 977 : 134 = 7 r. 39 → 134 : 39 = 3 r. 17 →
→ 39 : 17 = 2 r. 5 → 17 : 5 = 3 r. 2 → 5 : 2 = 2 r. 1 → 2 : 1 = 2 r. 0

Algorytmy i złożoność – ćwiczenia 14. 03. 2023 r.

$1 < \lg n < n < n^2 < n^3 < n^4 < n^5 < 2^n < 3^n < \pi^n < n!$

f) $f(n) = 2^n + 4n^2$ Hipoteza: $f(n) \in O(2^n)$, $2^n \in O(f(n))$

$\underbrace{2^n + 4n^2}_{f} \geq \underbrace{2^n}_{g}$ $\underbrace{E\, c>0\ \exists N \in \mathbb{N}\ \forall n \in \mathbb{N}\ [n > N \Rightarrow g(n) \leq c \cdot f(n)]}_{g \in O f(n)}$

Jeżeli $g \leq f$, to biorąc $c = 1$ mamy że $g \in O f(n)$ (bo $g \leq f \cdot 1$) więc $2^n \in O f(n)$, ponieważ ($2^n + 4n^2 \geq 2^n$ zawsze $\Longleftrightarrow$) to $g \in O f(n)$.

$\underbrace{2^n + 4n^2}_{f} \leq \underbrace{2^n \cdot c}_{g}$ (?) $/ : 2^n$

$\frac{2^n + 4n}{2^n} \leq c$ (?)

$\frac{1 + \frac{4n^2}{2^n}}{1} \leq c$ (?)

Wtedy $c = 2$. Wtedy $4n^2 \leq 2^n\ \forall! n$

Uzasadnimy to za chwilę.

$2^n + 4n^2 \leq 2^n + 2^n \Rightarrow 2^n + 4n^2 \leq 2 \cdot 2^n$

Zatem $\exists c > 0 : \forall! n\ 2^n + 4n^2 \leq 2^n \cdot c$

Z tego wynika że $f(n) \in O(2^n)$. Podsumowując $f(n) \in \Theta(2^n)$.

$4n^2 \leq 2^n$ (?) $/ : 4 \Rightarrow n^2 \leq \frac{2^n}{2^2}$ (?) $\xrightarrow{/ : 2^n} \lim_{n \to \infty} \frac{n^2}{2^n} = 0$ Z tego wnioskujemy że $\forall! n\ [\frac{n^2}{2^n} \leq \frac{1}{4}] \Rightarrow 4n^2 \leq 2^n, \forall! n$

Lemat: $\forall! n\ 4n^2 \leq 2^n$ Dowód lematu:

$\lim_{n \to \infty} \frac{n^2}{2^n} = 0$ z twierdzenia o granicy: $\forall \varepsilon > 0\ \exists N\ \forall n > N\ [\frac{n^2}{2^n} < \varepsilon]$.

Wtedy $\varepsilon = \frac{1}{4}$. Zatem $\forall! n\ [\frac{n^2}{2^n} \leq \frac{1}{4}]$, więc wynika z tego że $\forall! n\ 4n^2 \leq 2^n$ □

z lematu wynika że $\forall^{!} n \quad 2^n + 4n^2 \leq 2 + 2^n$. zatem

$\exists c>0 : \forall^{!} n \quad 2^n + 4n^2 \leq 2^n \cdot c$ więc $f(n) \in O(2^n)$. Podsumowując, $f(n) \in \Theta(2^n)$

c) $F(n) = 2^n + 4n^n$. Chcemy wykazać że $F \in O(n^n)$.

Niech $f(n) = 2^n$, $g(n) = 4n^n$. Niech $i(n) = 4$ oraz $h(n) = n^n$.

Teraz wykażemy że $g \in O(n^n)$. $i \in O(1) \wedge h \in O(n^n) \overset{TW.}{\Longrightarrow} 4 \cdot n^n \in O(n^n \cdot 1)$.

Zatem $4n^n \in O(n^n)$. Teraz wykażemy, że $F \in O(n^n)$, tzn

pokażemy, że: $\exists c>0 \ \exists N \in \mathbb{N} \ \forall n > N : F(n) \leq c \cdot n^n$, czyli $2^n \leq c \cdot n^n$.

Weźmy $c = 100$, które $c > 0$ oraz $N = 1$. Wtedy zachodzi:

$\forall n \geq 1 : 2^n \leq 100 \cdot n^n \Longrightarrow f \in O(n^n)$.

$(f \in O(n^n) \wedge g \in O(n^n)) \overset{TW}{\Longrightarrow} f + g \in O(n^n)$ zatem $F(n) \in O(n^n)$.

$2^n + 4n^n \in O(n^n)$ mimo tego, że $2^n + 4n^n \not\leq n^n$, ale ($2^n + 4n^n \leq n^n$

oraz $n^2 \leq 2^n + 4n^n$) wynika, że $F(n) \in \Theta(n^n)$.

g) Chcemy pokazać, że $2^n < 3^n$.

$\lim_{n\to\infty} \frac{2^n}{3^n} = \lim_{n\to\infty} \left(\frac{2}{3}\right)^n = 0$, bo $\frac{2}{3} < 1$ zatem z tw. $2^n \in O(3^n)$ oraz

$3^n \notin O(2^n)$.

h) Chcemy pokazać że $\log_a n \asymp \log_b n$. Niech $a, b > 1$.

$\log_a n = \frac{\log_b n}{\log_b a}$ z tw. o logarytmach. Zatem $\frac{1}{\log_b a} = c$ - stała $c > 0$

Zatem $\log_a n = c \cdot \log_b n$.

i) Udowodnić $\lim_{n\to\infty} \frac{f(n)}{g(n)} = a \neq 0 \Rightarrow f \in \Theta(g)$. (Wniosek: przy założeniach jak wyżej $f \in \Theta(g)$.)

$\exists c>0 \ \exists N \in \mathbb{N} \ \forall n > N \quad f(n) \leq c \cdot g(n)$ (?)

—— || —— $\frac{f(n)}{g(n)} \leq c$ (?), ale wiemy z założenia, że

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = a \neq 0$. Weźmy dowolne $c > a$. Wtedy $\forall^{!} n \ a_n < c$,

(gdzie $a_n = \frac{f(n)}{g(n)}$) więc $\forall^{!} n \ \frac{f(n)}{g(n)} < c$. Zatem $\forall^{!} n \ f(n) \leq c \cdot g(n)$.

Wniosek: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = a$, $a \neq 0$ to $g \in O(f)$, wtedy $\lim_{n\to\infty} \frac{g(n)}{f(n)} = \frac{1}{a} \neq 0$.

j) $f(n) = 3n^4 + 6n^3 - n^2$. Hipoteza: $f \in \Theta(n^4)$.

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = a \neq 0 \Rightarrow g \in O(f)$ i $f \in O(g)$. Niech $g(n) = n^4$.

Wtedy $\lim_{n\to\infty} \frac{3n^4 + 6n^3 - n^2}{n^4} = 3 \neq 0$. Zatem z twierdzenia wynika

że $n^4 \in O(f)$ i $f \in O(n^4)$. Podsumowując $f \in \Theta(n^4)$.

II sposób: $6n^3 - n^2 < 3n^4$, $\forall^{!} n$. Więc wnioskujemy z definicji

, że $6n^3 - n^2 \in O(3n^4)$. $f+g$

Wniosek: $\overbrace{6n^3 - n^2 + 3n^4}^{f+g} \in O(3n^4)$, zatem $\exists c>0 : 3n^4 \leq c \cdot$ (takim c jest np. 3) więc $3n^4 \in O(n^4)$, czyli $\underline{6n^3 - n^2 + 3n^4 \in O(n^4)}$.

$n^4 \leq \underbrace{3n^4 + 6n^3 - n^2}_{f}, \forall! n \Rightarrow \underline{n^4 \in O(f)}$. Zatem $f \in \Theta(n^4)$.

Zadanie: Zapisać liczbę 1111 jako $1111 = c_1 \cdot 10^3 + c_2 \cdot 10^2 + c_3 \cdot 10^1 + c_4 \cdot 10^0$ (Zj) oraz liczbę $297 = c_1 \cdot 10^2 + c_2 \cdot 10^1 + c_3 \cdot 10^0$ i zrobić to tylko z potęgami 2, czyli $(297)_{(10)} = \sum_{i=0}^{8} c_i \cdot 2^i$, $c_i = 0,1$

Algorytmy i złożoność - wykład 15.03.2023 r.

Zależność rekurencyjna rzędu k, gdzie $k \in \mathbb{N} \setminus \{0\}$, to wyrażenie postaci: (R) $a_n = \varphi(a_{n-1}, a_{n-2}, \dots, a_{n-k})$, dla $n \geq$ gdzie φ jest pewną funkcją k-argumentów, czyli $\varphi: \mathbb{R}^k \to$

Gdy dane są, dodatkowo, warunki początkowe

(P): $a_0 = x_0, \dots, a_{k-1} = x_{k-1}$, gdzie $x_0, \dots, x_{k-1} \in \mathbb{R}$, to układ (R), (P) będziemy nazywać zależnością rekurencyjną rzędu k z warunkami początkowymi. Wtedy ciąg (a_n) jest zdefiniowany jednoznacznie.

Przykład 1 (ciąg geometryczny):

$$\begin{cases} a_0 = x_0 \\ a_n = q \cdot a_{n-1}, \text{ dla } n \geq 1 \end{cases}$$ ← (zależność rekurencyjna rzędu 1 z warunkiem początkowym)

$\varphi(x) = q \cdot x$

Przykład 2 (ciąg Fibonacciego):

$$\begin{cases} a_0 = 1 \\ a_1 = 1 \\ a_n = a_{n-1} + a_{n-2}, \ n \geq 2 \end{cases}$$ $\varphi(x,y) = x+y$ ← (zależność rekurencyjna rzędu 2 z warunkami początkowymi).

I Liniowe zależności rekurencyjne rzędu k

Zależność rekurencyjna: $a_n = c_1 \cdot a_{n-1} + c_2 \cdot a_{n-2} + \dots + c_k \cdot a_{n-k} + f$ gdzie $f: \mathbb{N} \to \mathbb{N}$ nazywamy zależnością rek. liniową.

Gdy $c_k \neq 0$, to mówimy że jest to zależ. rek. liniowa rzędu k.

Gdy $f(n) \equiv 0$, to $a_n = c_1 \cdot a_{n-1} + \dots + c_k \cdot a_{n-k} + f(n)$ nazywamy jednorodną i ma ona postać (Zj) $a_n = c_1 \cdot a_{n-1} + \dots + c_k \cdot$

W przykładzie 1 oraz 2 mówimy o jednorodnych zależ. rek.

Gdy $f(n)$ nie jest stale równa 0, to $a_n = c_1 \cdot a_{n-1} + \ldots + c_k \cdot a_{n-k} + f(n)$ nazywamy niejednorodną zależ. rek. liniową.

II Rozwiązywanie jednorodnych zależności rekurencyjnych liniowych.

Rozwiązać oznacza przedstawić wzór jawny na a_n, czyli $a_n = A(n)$. (wzór jawny = explicit formula)

(Z_j) $a_n = c_1 a_{n-1} + \ldots + c_k a_{n-k}$.

Obserwacja 1: Jeśli $p_n = p(n)$ jest jakimś rozwiązaniem (Z_j), $x \in \mathbb{R}$, to $k \cdot p_n$ też jest rozwiązaniem (Z_j). Tzn:

jeśli $\forall n$ $p_n = c_1 p_{n-1} + \ldots + c_k p_{n-k}$, to $x p_n = c_1 x p_{n-1} + \ldots + c_k x p_{n-k}$.

czyli $g_n = x p_n$, $g_n = c_1 g_{n-1} + \ldots + c_k g_{n-k}$.

Obserwacja 2: Jeśli g_n i p_n są dwoma rozwiązaniami (Z_j), to $r_n = p_n + g_n$ także jest rozwiązaniem (Z_j).

Obserwacja 3: $\underline{0}_n = 0$ - ciąg stale równy zero, $\underline{0}_n$ jest rozwiązaniem (Z_j).

$\langle \{p_n : p_n$ jest rozw. $(Z_j)\}, +, \circ, \underline{0}_n \rangle$ - przestrzeń wektorowa

Sk. wymiar: k W przykładzie 1: $\begin{cases} a_0 \\ a_n = q a_{n-1}, n \geq 1. \end{cases}$

Wzór jawny: $a_n = q^n \cdot a_0$ $(a_0 = q^0 a_0,\ a_1 = q^1 a_1, \ldots)$

Poszukajmy ciągu geometrycznego spełniającego (Z_j).

$a_n = a_0 q^n$ podstawiamy do (Z_j): $a_0 q^n = c_1 a_0 q^{n-1} + c_2 a_1 q^{n-2} + \ldots + c_i a_0 q^{n-i} + \ldots + c_k a_0 q^{n-k}$. Dzielimy przez $a_0 q^{n-k}$ oraz zakładamy że $a_0 \neq 0$ i $q \neq 0$. Zatem: $q^k = c_1 q^{k-1} + c_2 q^{k-2} + \ldots + c_i q^{k-i} + \ldots + \underbrace{c_k q^0}_{c_k}$

METODA:

1) Tworzymy równanie charakterystyczne (Z_j):

$x^k = c_1 x^{k-1} + c_2 x^{k-2} + \ldots + c_k$

2) Rozwiązujemy (CH) w $\mathbb{C}$. Otrzymujemy k pierwiastków $x_1, x_2, \ldots, x_k$. (mogą się powtarzać). Każdy ciąg postaci $p_n = x_i^n$ jest rozwiązaniem (Z_j). Z obserwacji 1 i 2, każda kombinacja liniowa: $d_1 \cdot x_1^n + d_2 x_2^n + \ldots + d_k x_k^n$ też jest rozwiązaniem (Z_j).

3) Jeżeli $x_1, \ldots, x_k$ są parami różne to każde rozwiązanie (Z_j) da się przedstawić jako $d_1 x_1^n + \ldots + d_k x_k^n = p(n)$, $d_i \in \mathbb{R}$

jeśli x_1 jest l-krotnym pierwiastkiem (CH), to jako bazowe rozwiązania bierzemy: $x_1^n, n x_1^n, n^2 x_1^n, \dots, n^{l-1} x_1^n$.

wtedy: $p(n) = d_1 x_1^n + d_2 n x_1^n + \dots + d_l n^{l-1} x_1^n + \dots$, gdzie $d_1, \dots, d_k$ - dowolne liczby rzeczywiste

4) aby rozwiązać zależności jednorodne z warunkami początkowymi (P), znajduje $d_1, \dots, d_k$ korzystając z

Przykład:

$$\begin{cases} a_0 = 2 \\ a_n = 3 \cdot a_{n-1}, n \geq 1 \end{cases}$$

$k=1$ (CH): $x^1 = 3$

Rozwiązanie $x_1 = 3$

ROZj: $p_n = d_1 x_1^n$

$p_n = d_1 \cdot 3^n$

$a_0 = 2$, $2 = d_1 \cdot 3^0$

$2 = d_1$

Rozw: $a_n = 2 \cdot 3^n$

Przykład:

$$\begin{cases} a_0 = 1 \\ a_n = -5 a_{n-1} - 6 a_{n-2}, n \geq 2 \end{cases}$$

$k=2$ (CH): $x^2 = -5x - 6 \Rightarrow$

$\Rightarrow x^2 + 5x + 6 = 0 \Rightarrow (x+2)(x+3) = 0$

$x_1 = -2$, $x_2 = -3$

ROZj: $a_n = d_1(-2)^n + d_2(-3)^n$

(P): $a_0 = 1 \Rightarrow 1 = d_1(-2)^0 + d_2(-3)^0 \Rightarrow 1 = d_1 + d_2$

$2 = d_1 \cdot (-2)^1 + d_2(-3)^1 \Rightarrow 2 = -2d_1 - 3d_2$

$$\begin{cases} d_1 + d_2 = 1 \quad |\cdot 2 \\ -2d_1 - 3d_2 = 2 \end{cases} \Rightarrow \begin{cases} 2d_1 + 2d_2 = 2 \\ -2d_1 - 3d_2 = 2 \end{cases} \Bigg| + \Rightarrow \begin{matrix} -d_2 = 4 \\ d_2 = -4 \end{matrix} \Rightarrow \begin{matrix} 1 = d_1 + d_2 \\ d_1 = 5 \end{matrix}$$

Rozw: $a_n = 5 \cdot (-2)^n - 4(-3)^n$ Spr. $a_0 = 1$, $a_1 = 2$, $a_2 = -16$

III Ciąg liczb Fibonacciego:

(P) $a_0 = 1$, $a_1 = 1$ (Zj) $a_n = a_{n-1} + a_{n-2}$, $n \geq 2$, $k = 2$

(CH) $x^2 = x + 1 \Rightarrow x^2 - x - 1 = 0 \Rightarrow x_1 = \frac{1-\sqrt{5}}{2}$ $x_2 = \frac{1+\sqrt{5}}{2}$

ROZj: $p_n = d_1 x_1^n + d_2 x_2^n$

(P) $a_0 = 1 \Rightarrow 1 = d_1 x_1^0 + d_2 x_2^0$, $a_1 = 1 \Rightarrow 1 = d_1 x_1^1 + d_2 x_2^1$

$1 = d_1 + d_2 \quad |\cdot x_1$

$$\begin{cases} x_1 = d_1 x_1 + d_2 x_1 \\ 1 = d_1 x_1 + d_2 x_2 \end{cases}$$

$1 - x_1 = d_2(x_2 - x_1)$

$d_2 = \frac{1 - x_1}{x_2 - x_1}$, $x_2 \neq x_1$

$d_1 = 1 - d_2$

wzór jawny na n-ty wyraz ciągu Fibonacciego $FIB_n = a_n$ z tego zadania:

$FIB_n = \frac{\sqrt{5}-1}{2\sqrt{5}} \cdot \left(\frac{1-\sqrt{5}}{2}\right)^n + \frac{1+\sqrt{5}}{2\sqrt{5}} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n$

$FIB_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^{n+1} + \left(-\frac{1}{\sqrt{5}}\right)\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}$

$$\boxed{FIB_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}}$$

$d_2 = \frac{1-\frac{1+\sqrt5}{2}}{\frac{1+\sqrt5}{2}-\frac{1+\sqrt5}{2}} = \frac{2-(1-\sqrt5)}{1+\sqrt5-(1-\sqrt5)} = \frac{1+\sqrt5}{2\sqrt5}$ $\quad n=0,\ n=1,\ n=2$

$d_1 = \frac{-(1+\sqrt5)}{2\sqrt5}+1 = \frac{2\sqrt5-(1+\sqrt5)}{2\sqrt5} = \frac{\sqrt5-1}{2\sqrt5}$

Rozważmy liczbę: $\left|\frac{1-\sqrt5}{2}\right| = \frac{\sqrt5-1}{2}$. $2<\sqrt5<3$ więc $\frac13<\frac{1}{\sqrt5}<\frac12$.

$1<\sqrt5-1<2$ więc $\frac{\sqrt5-1}{2}<1$, zatem $\left|\frac{1-\sqrt5}{2}\right|<1$, czyli $\left|\frac{1}{\sqrt5}\cdot\frac{1-\sqrt5}{2}\right|<\frac12$.

$\left|\frac{1}{\sqrt5}\cdot\left(\frac{1+\sqrt5}{2}\right)^{n+1} - FIB_n\right| = \left|\frac{1}{\sqrt5}\cdot\left(\frac{1-\sqrt5}{2}\right)^{n}\right| < \frac12$

Wniosek: n-ty wyraz ciągu Fibonacciego jest liczbą całkowitą najbliższą liczbie $\frac{1}{\sqrt5}\left(\frac{1+\sqrt5}{2}\right)^{n+1}$, czyli

$$\boxed{\frac{1}{\sqrt5}\left(\frac{1+\sqrt5}{2}\right)^{n+1} \approx FIB_n}.$$

$1111 = \overset{c_1}{1}\cdot 10^3 + \overset{c_2}{1}\cdot 10^2 + \overset{c_3}{1}\cdot 10^1 + \overset{c_4}{1}\cdot 10^0$ $\qquad 297 = \overset{c_1}{2}\cdot 10^2 + \overset{c_2}{9}\cdot 10^1 + \overset{c_3}{7}\cdot 10^0$

$1111 = 1\cdot 2^{10} + 0\cdot 2^9 + 0\cdot 2^8 + 0\cdot 2^7 + 1\cdot 2^6 + 0\cdot 2^5 + 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0$

$297 = 1\cdot 2^8 + 0\cdot 2^7 + 0\cdot 2^6 + 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0$ Szybsze!

Algorytmy i złożoność – ćwiczenia 27.03.2023r.

1) $a_n = a_{n-1} + 6a_{n-2}$, $n \geq 2$, $a_0 = 1$, $a_1 = 3$ – zależność rekurencyjna stopnia 2 (, bo odwołuje się do 2 poprzednich wyrazów i ma 2 warunki początkowe) liniowa.

Ogólny wzór na n-ty wyraz: $x^2 = x + 6 \Rightarrow x^2 - x - 6 = 0 \Rightarrow$

$\Rightarrow (x-3)(x+2) = 0 \Rightarrow x_1 = 3 \wedge x_2 = -2$

$a_n = d_1 x_1^n + d_2 x_2^n$ ← rozw. ogólne (nie biorąc warunków początkowych)

$a_0 = 1 \Rightarrow 1 = d_1 x_1^0 + d_2 x_2^0 \Rightarrow \begin{cases} 1 = d_1 + d_2 & /\cdot 2 \\ 3 = 3d_1 - 2d_2 \end{cases} \Rightarrow \begin{cases} 2 = 2d_1 + 2d_2 \\ 3 = 3d_1 - 2d_2 \end{cases} + \Rightarrow$

$a_1 = 3 \Rightarrow 3 = d_1 x_1^1 + d_2 x_2^1 \Rightarrow$

$\Rightarrow \begin{cases} 5 = 5d_1 \\ 2 = 2d_1 + 2d_2 \end{cases} \Rightarrow \begin{cases} d_1 = 1 \\ 2 = 2\cdot 1 + 2d_2 \end{cases} \Rightarrow \begin{cases} d_1 = 1 \\ d_2 = 0 \end{cases}$

Rozwiązanie szczególne: $a_n = 3^n$

Sprawdzenie: Krok podstawowy: $a_0 = 1$ Ok, $a_1 = 3^1$ Ok

Krok indukcyjny: zał. indukcyjne: $a_{n-1} = 3^{n-1}$, $a_{n-2} = 3^{n-2}$

Teza indukcyjna: $a_n = 3^n$ (?)

Ze wzoru danego w treści zadania: $a_n = a_{n-1} + 6\cdot a_{n-2} \overset{\text{zał. ind.}}{=} 3^{n-1} + 6\cdot 3^{n-2} = 3^{n-1} + 2\cdot 3^{n-1} = 3\cdot 3^{n-1} = 3^n$ □. Na mocy zasady indukcji matematycznej $\forall n \geq 0\ a_n = 3^n$.

2) $a_n = 6a_{n-1} - 9a_{n-2}$, $n \geq 0$ $\quad a_0 = -5$, $a_1 = 3$

Wzór ogólny: $x^2 = 6x - 9 \Rightarrow x^2 - 6x + 9 = 0 \Rightarrow (x-3)^2 = 0$

ŹLE

$p_n = d_1 x_1^n + d_2 x_2^n \Rightarrow p_n = d_1 \cdot 3^n + d_2 \cdot 3^n \Rightarrow x_1 = 3, x_2 = 3$

$\Rightarrow p_n = 3^n(d_1 + d_2) \Rightarrow p_n = 3^n \cdot f$

$a_0 = -5 \Rightarrow -5 = 3^0 \cdot f \Rightarrow f = -5$

$a_1 = 3 \Rightarrow 3 = 3^1 \cdot f \Rightarrow f = 1$ – sprzeczność

Brakuje wyr.: $a_2 = 6a_1 - 9a_0 = 45 + 18 =$

$p_n = 3^n$ – pierwsze szczególne rozw. x_1^n

$q_n = n \cdot 3^n$ – drugie szczególne rozw. x_2^n } Baza przestrzeni

Ogólne rozwiązanie: $d_1 \cdot 3^n + d_2 \cdot n \cdot 3^n = p_n$ – kombinacja lin.

$a_0 = -5 = d_1 \cdot 3^0 + d_2 \cdot 0 \cdot 3^0 \Rightarrow d_1 = p_0 \Rightarrow \underline{-5 = d_1}$

$a_1 = 3 = d_1 \cdot 3^1 + d_2 \cdot 1 \cdot 3^1 \Rightarrow 3d_1 + 3d_2 = 3 \Rightarrow -15 + 3d_2 = 3 \Rightarrow \underline{d_2 = 6}$

Rozwiązanie szczególne: $a_n = -5 \cdot 3^n + 6 \cdot n \cdot 3^n$ dla $\forall n \geq 0$

Sprawdzenie: $a_0 = -5 \cdot 3^0 + 0 = -5$, $a_1 = -5 \cdot 3^1 + 6 \cdot 1 \cdot 3^1 = 3$

Krok indukcyjny: Niech $n \geq 2$.

zał. ind.: $a_{n-1} = -5 \cdot 3^{n-1} + 6(n-1) \cdot 3^{n-1}$, $a_{n-2} = -5 \cdot 3^{n-2} + 6(n-2) \cdot 3^{n-2}$

Teza indukcyjna: $a_n = -5 \cdot 3^n + 6n \cdot 3^n$

Dowód T. I.: $a_n = 6 \cdot a_{n-1} - 9a_{n-2} \overset{z.I}{=} 6 \cdot (-5 \cdot 3^{n-1} + 6(n-1) \cdot 3^{n-1}) -$
$- 9(-5 \cdot 3^{n-2} + 6(n-2) \cdot 3^{n-2}) = -30 \cdot 3^{n-1} + 36n \cdot 3^{n-1} - 36 \cdot 3^{n-1} + 45 \cdot 3^{n-1} - 18n \cdot 3^{n-1}$
$+ 36 \cdot 3^{n-1} = 3^{n-1}(-30 - 36 + 45 + 36) + n \cdot 3^{n-1}(36 - 18) = -15 \cdot 3^{n-1} + 18n \cdot 3^{n-1} =$
$= -5 \cdot 3 \cdot 3^{n-1} + 6 \cdot 3 \cdot 3^{n-1} = -5 \cdot 3^n + 6n \cdot 3^n$ D

na mocy z. I. M. $\forall n \geq 0$ $a_n = -5 \cdot 3^n + 6n \cdot 3^n$

Wieże Hanoi: $r_1 = 1$ (1 krążek), $r_2 = 3$ (2 krążki), $r_3 = 7$ (3 krążki), ... $r_4 = ?$

Przełożenie $n = 3$ górnych krążków na słupek środkowy używając III jako pomocniczy

Potem przekładamy ostatni największy krążek na III słupek

$r_{n+1} = 2r_n + 1$

Na końcu przekładamy wszystkie krążki z środkowego na III słupek używając I jako pomocniczy.

$r_4 = 7 + 1 + 7 = 2 \cdot 7 + 1 = 15$ więc $r_{n+1} = 2r_n + 1$

Algorytmy i złożoność - wykład 22.03.2023r

Szacowanie złożoności algorytmu Euklidesa w zależności od liczby pojedynczych dzieleń.

Dane: $a, b > 0$, $a, b \in \mathbb{N}$, $a > b$.

Ile rachunków dzieleń należy wykonać, by przeprowadzić algorytm Euklidesa dla tych danych.

$a = q_1 \cdot b + r_1$, $0 \le r_1 < b$

$b = q_2 \cdot r_1 + r_2$, $0 \le r_2 < r_1$

$r_1 = q_3 \cdot r_2 + r_3$, $0 \le r_3 < r_2$

$\vdots$

$r_{k-2} = q_k \cdot r_{k-1} + r_k$ $0 \le r_k < r_{k-1}$

$r_{k-1} = q_{k+1} \cdot r_k$ $(r_{k+1} = 0)$

$NWD(a,b) = GCD(a,b) = r_k$

Szacujemy k.

Obs. 1: $q_{k+1} \ge 1$, bo $r_{k-1} > r_k$ oraz $r_{k-1} = q_{k+1} \cdot r_k$ (więc $q_{k+1} \ge 2$)

Obserwacja 2: $\forall i$ $q_i \ge 1$ (bo $r_k > 0$)

$r_k \ge 1 \Rightarrow r_{k-1} \ge 2 \cdot r_k \ge 2 \Rightarrow r_{k-2} = q_k \cdot r_{k-1} + r_k \ge \underline{r_{k-1} + r_k} \ge 2 + 1 = 3$

$r_{k-3} = q_{k-1} \cdot r_{k-2} + r_{k-1} \ge 1 \cdot r_{k-2} + r_{k-1} \ge \underline{r_{k-2} + r_{k-1}} \ge 3 + 2 = 5$

$r_1 \ge r_2 + r_3$

$b \ge r_1 + r_2$

$b \ge F_{k+1}$ $\quad l = k+1 \Rightarrow b \ge F_l$

$F_l = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{l+1} - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{l+1} \in \mathbb{Z}$

Wniosek: $\frac{1}{\sqrt{5}}\left(\frac{\sqrt{5}-1}{2}\right)^{l+1}$ - mały

Liczba Fibonacciego F_l jest najbliższą liczbą całkowitą do liczby $\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{l+1}$

$F_l \approx \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{l+1} \Rightarrow F_l \ge \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{l+1} - 1 \Rightarrow b \ge \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{l+1} - 1 \Rightarrow$

$\Rightarrow \sqrt{5}(1+b) \ge \left(\frac{1+\sqrt{5}}{2}\right)^{l+1}$ $\qquad \underbrace{A \cdot b}_{\sqrt{5} \cdot b} + \underbrace{B}_{\sqrt{5}} \ge \left(\frac{1+\sqrt{5}}{2}\right)^{l+2}$, $A, B > 0$

Niech $C = \frac{1+\sqrt{5}}{2} > 1$. Zatem $\lg C > 0$, więc

$\lg(A \cdot b + B) \ge (l+1)\lg(C) \Rightarrow (l+1)\underbrace{\lg(C)}_{\text{stała}} \le \lg(A \cdot b + B) \Rightarrow$

$\Rightarrow l \le \lg(\underbrace{A \cdot b + B}_{\Theta(b)}) - \lg C \Rightarrow l \in O(\lg(b))$, bo $A \cdot n + B \in \Theta(n)$

Liczba kroków (tzn. dzieleń) w algorytmie Euklidesa jest funkcją klasy $O(\lg(b))$. Algorytm Euklidesa ma złożoność $O(\lg b)$, gdzie b jest mniejszą z liczb których NWD obliczamy.

II Rozwiązywanie liniowych zależności rekurencyjnych rzędu k niekoniecznie jednorodnych.

(Z_n) $a_n = c_1 \cdot a_{n-1} + \ldots + c_k a_{n-k} + f(n)$, $c_k \neq 0$. Wtedy rozważamy (Z_j) $a_n = c_1 \cdot a_{n-1} + \ldots + c_k \cdot a_{n-k}$. Znajdujemy ogólne rozwiązanie (Z_j). Niech $p(n)$ będzie ROZ_j.

Przypuśćmy, że znamy jakieś rozwiązanie (Z_n). Czyli $g(n) = c_1 \cdot g(n-1) + \ldots + c_k \cdot g(n-k) + f(n)$.

Wtedy $a_n = p(n) + g(n)$, $n \geq 0$ jest rozwiązaniem (Z_n). Aby to sprawdzić podstawiamy a_n do (Z_n) i sprawdzamy czy (Z_n) jest spełnione. Ma być (Z_n)

$p(n) + g(n) \overset{?}{=} c_1(p(n-1) + g(n-1)) + c_2(p(n-2) + g(n-2)) + \ldots$

$\ldots + c_{n-k}(p(n-k) + g(n-k)) + f(n)$. Ale

$p_n = c_1 p(n-1) + c_2 p(n-2) + \ldots + c_{n-k} p(n-k) + c_1 g(n-1) + \ldots + c_{n-k} g(n-k)$

$= p(n) + g(n) = a_n$ $\qquad ROZ_p = ROZ_j + \underbrace{RSZ_N}_{\text{rozwiązanie szczególne}}$ (jakieś rozw.

Jak można poszukiwać RSZ_n?

Jeżeli $f(n)$ jest pewnej postaci to poszukujemy RSZ_n w takiej samej postaci. np: $f(n)$ – stała, to poszukujemy $g(n)$ jako stałej. Przypuszczamy że $g(n) = G$. Podstawiamy $g(n)$ do (Z_n) i to pozwala obliczyć G.

np: $f(n)$ – liniowa, przypuszczamy że $g(n) = G \cdot n + H$

np: $f(n)$ – wielomian stopnia l, to przypuszczamy że $g(n) = G_0 \cdot n^l + G_1 \cdot n^{l-1} + \ldots + G_{l-1} \cdot n + G_l$

np: $f(n) = G^n = 2^{a \cdot n}$ lub $f(n) = 2^{a \cdot n + b}$

$\log_2 G = a \Rightarrow (2^a)^b = G \Rightarrow G^n = 2^{(a \cdot b) \cdot n}$ w roli $g(n) = 2^{G \cdot n}$

np: $f(n) = W(n) \cdot 2^{an+b}$ $\qquad g(n) = Q(n) \cdot 2^{G \cdot n + H}$, gdzie $Q(n)$ – wielomian ogólny tego samego stopnia co W, G, H – ogólne stałe. Podstawiając przewidywane $g(n)$ do (Z_n) otrzymujemy wartości stałych potrzebnych po to by $g(n)$ rozwiązało (Z_n).

III Metoda rozwiązania (Z_n) z warunkami początkowymi

1. Tworzymy i rozwiązujemy (Z_j). Otrzymujemy (ROZ_j) i p
2. Szukamy jakiegoś szczególnego (RZ_n), przewiduje $g(n)$

w postaci podobnej do postaci $f(n)$ i wyznaczając współczynniki w $g(n)$ z (Z_n).

3. $a_n = p(n) + g(n)$ – (ROZ_n) (występują tam nieznane współczynniki z 1.)

4. Podstawiamy a_n do (P). Wyliczamy stałe. Otrzymujemy (ROZ_n) z (P).

Przykład: $(Z_n)\begin{cases} a_n = 5a_{n-1} - 2a_{n-2} + 3n^2, & n \geq 2 \\ (P)\ a_0 = 0,\ a_1 = 3 \end{cases}$

1. $(Zj) = a_n = 5a_{n-1} - 2a_{n-2},\ n \geq 2 \Rightarrow x^2 = 5x - 2 \Rightarrow x^2 - 5x + 2 = 0$

$\Rightarrow x_1, x_2 = \frac{5 \mp \sqrt{17}}{2}$ $\quad ROZj: p(n) = c_1 \cdot \left(\frac{5-\sqrt{17}}{2}\right)^n + c_2 \cdot \left(\frac{5+\sqrt{17}}{2}\right)^n$

2. $f(n) = 3n^2$ Przewidujemy: $g(n) = an^2 + bn + c$ i podstawiamy do (Z_n): $g(n) = 5 \cdot g(n-1) - 2 \cdot g(n-2) + 3n^2 \Rightarrow$

$\Rightarrow an^2 + bn + c = 5 \cdot (a(n-1)^2 + b(n-1) + c) - 2(a(n-2)^2 + b(n-2) + c) + 3n^2 \Rightarrow$

$\Rightarrow an^2 + bn + c = 5(an^2 - 2an + a + bn - b + c) - 2(an^2 - 4an + 4a + bn - 2b + c) + 3n^2 \Rightarrow$

$\Rightarrow an^2 + bn + c = (5a - 2a + 3)n^2 + (-10a + b + 8a - 2b)n + (5a - 5b + 5c - 8a + 4b - 2c)$

$\Rightarrow an^2 + bn + c = (3a + 3)n^2 + (-2a + 3b)n - 3a - b + 3c. \Rightarrow$

$$\Rightarrow \begin{cases} c = 3c - 3a - b \\ b = -2a + 3b \\ a = 3a + 3 \end{cases} \Rightarrow \begin{cases} a = -\frac{3}{2} \\ b = -2 \cdot (-\frac{3}{2}) + 3b \\ c = 3c - 3 \cdot (-\frac{3}{2}) - b \end{cases} \Rightarrow \begin{cases} a = -\frac{3}{2} \\ b = -\frac{3}{2} \\ c = 3c + \frac{9}{2} - \frac{3}{2} \end{cases} \Rightarrow \begin{cases} a = -\frac{3}{2} \\ b = -\frac{3}{2} \\ c = -\frac{9}{4} \end{cases}$$

$g(n) = -\frac{3}{2}n^2 - \frac{3}{2}n - \frac{9}{4}$

3. $ROZ_n: a_n = c_1 \cdot \left(\frac{5-\sqrt{17}}{2}\right)^n + c_2 \cdot \left(\frac{5+\sqrt{17}}{2}\right)^n + \left(-\frac{3}{2}n^2 - \frac{3}{2}n - \frac{9}{4}\right)$

4. $a_0 = 0,\ a_1 = 3$ $\quad (P)|a_0: 0 = c_1 + c_2 - \frac{9}{4} \Rightarrow c_1 + c_2 = \frac{9}{4}$

$a_1: 3 = c_1 \frac{5-\sqrt{17}}{2} + c_2 \cdot \frac{5+\sqrt{17}}{2} - \frac{3}{2} - \frac{3}{2} - \frac{9}{4} \Rightarrow 7\frac{3}{4} = c_1 \cdot \frac{5-\sqrt{17}}{2} + c_2 \cdot \frac{5+\sqrt{17}}{2} \overset{k2}{\Rightarrow}$

$c_2 = \frac{9}{4} - c_1$

$\Rightarrow 14\frac{1}{2} = 5c_1 - \sqrt{17} \cdot c_1 + 5c_2 + \sqrt{17} \cdot c_2$

$14\frac{1}{2} = 5 \cdot \frac{5}{4} - \sqrt{17}(c_1 - c_2) \Rightarrow 14\frac{1}{2} = \frac{45}{4} - \sqrt{17}(2c_1 - \frac{9}{4}) \overset{k4}{\Rightarrow}$

$\Rightarrow 58 = 45 - \sqrt{17}(8c_1 - 9) \Rightarrow 13 - 9\sqrt{17} = -8\sqrt{17} \cdot c_1 \Rightarrow$

$\Rightarrow c_1 = \frac{-13 + 9\sqrt{17}}{8\sqrt{17}} = \frac{-13\sqrt{17} + 9 \cdot 17}{8 \cdot 17} = \frac{-13\sqrt{17} + 153}{136}$

$c_2 = \frac{9}{4} + \frac{13\sqrt{17}}{136} - \frac{153}{136} \Rightarrow c_2 = \frac{153 + 13\sqrt{17}}{136}$

Odpowiedź: $a_n = \frac{-13\sqrt{17} + 153}{136}\left(\frac{5-\sqrt{17}}{2}\right)^n + \frac{153 + 13\sqrt{17}}{136} \cdot \left(\frac{5+\sqrt{17}}{2}\right)^n - \frac{3}{2}n^2 - \frac{3}{2}n - \frac{9}{4}.$

Przykład: $(Z_n)\begin{cases} r_n = 2r_{n-1} + 1 & n \geq 2 \\ (P)\ r_1 = 1 \end{cases}$ $\quad f(n) = 1$

$(Zj): r_n = 2r_{n-1}$ $\quad ROZj: p_n = c \cdot 2^n$, $q_n = b$ podstawiamy

do (Zn): $b = 2b + 1 \Rightarrow -1 = b \Rightarrow q_n = -1$

$ROZ_n = ROZj + RSZ_n$ $\quad r_n = c \cdot 2^n - 1$ wyznaczamy c z r_1

$r_1 = 1$: $1 = c \cdot 2^1 - 1 \Rightarrow 2 = c \cdot 2 \Rightarrow c = 1$

Odpowiedź: $r_n = 2^n - 1$ $\quad r_1 = 1,\ r_2 = 3,\ r_3 = 7,\ r_4 = 15$

Wzór jawny na liczbę pojedynczych ruchów, które należy wykonać, by rozwiązać zagadkę „wieży Hanoi" z n krążkami.

Np: gdy przetbadanie 1 krążka zajmuje 1 sekundę to wykonanie zagadki dla 8 krążków zajmie $r_8 = 2^8 - 1 = 255$ sekund ≈ 4 min.

Algorytmy i złożoność - ćwiczenia 28.03.2023r.

Zad. 1

a) $a_n = 5a_{n-1}$, $n \geq 1 \leftarrow (Zj)$

(Ch): $x_1 = 5$ $\quad ROZj: p_n = d_1 \cdot x_1^n \Rightarrow p_n = d_1 \cdot 5^n$

e) (Zj): $a_n = 4a_{n-1} - 4a_{n-2}$, $n \geq 2$

(Ch) $x^2 = 4x - 4 \Rightarrow x^2 - 4x + 4 = 0 \Rightarrow (x-2)^2 = 0$ $\quad x_1 = 2$ $\quad x_2 = 2$

$(ROZj)$ $p(n) = d_1 x_1^n + d_2 x_2^n \Rightarrow p(n) = d_1 2^n + d_2 \cdot n \cdot 2^n$

f) (Zj): $a_n = 4a_{n-1} + 4a_{n-2}$, $n \geq 2$

(Ch) $x^2 = 4x_1 + 4 \Rightarrow x^2 - 4x - 4 = 0$ $\quad \Delta = 16 + 16 = 32$ $\quad \sqrt{\Delta} = 4\sqrt{2}$

$x_1 = 2 - 2\sqrt{2}$ $\quad x_2 = 2 + 2\sqrt{2}$

$(ROZj)$: $p(n) = d_1 \cdot (2 - 2\sqrt{2})^n + d_2 \cdot (2 + 2\sqrt{2})^n$

g) (Zj): $a_n = 2a_{n-1} - a_{n-2}$, $n \geq 2$

(Ch): $x^2 = 2x - 1 \Rightarrow x^2 - 2x + 1 = 0 \Rightarrow (x-1)^2 = 0 \Rightarrow x_1 = 1$ $\quad x_2 = 1$

$(ROZj)$: $p(n) = d_1 \cdot x_1^n + d_2 x_2^n \Rightarrow p(n) = d_1 \cdot 1^n + d_2 \cdot n \cdot 1^n = d_1 + n \cdot d_2$

Sprawdzamy, że ~~[illegible]~~ $p(n) = d_1 + n \cdot d_2$ to $p(n)$ spełnia warunek g) dla $n \geq 2$. W tym celu $p(n-1) = d_1 + (n-1)d_2$, $p(n-2) = d_1 + (n-2)d_2$ wstawiamy do warunku g), czyli $P_g = p_g = 2(d_1 + (n-1)d_2) - (d_1 + (n-2)d_2) = 2d_1 + 2nd_2 - 2d_2 - d_1 - nd_2 + 2d_2 = d_1 + nd_2 = p(n) = L_g$

c) (2_2): $a_n = a_{n-1} - a_{n-2} + a_{n-3}$, $n \geq 3$

(Ch): $x^3 = x^2 - x + 1 \Rightarrow x^3 - x^2 + x - 1 = 0 \Rightarrow x^2(x-1) + 1(x-1) = 0 \Rightarrow$

$\Rightarrow (x^2+1)(x-1) = 0 \Rightarrow x_1 = 1$ $x^2 = -1$ – sprzeczne w $\mathbb{R}$ ale $x_2 = i$, $x_3 = -i$

ROZ$_p$: $p(n) = d_1 x_1^n + d_2 x_2^n + d_3 x_3^n \Rightarrow p(n) = d_1 \cdot 1^n + d_2 \cdot i^n + d_3 (-i)^n$

np: $n = 4$ $p(4) = d_1 \cdot 1^4 + d_2 \cdot i^4 + d_3 \cdot (-i)^4 = d_1 + d_2 + d_3$

zad. 2

a) $p(n) = 12a_{n-1} - 46a_{n-2} + 60a_{n-3} - 25a_{n-4}$ $n \geq 4$

(Ch): $x^4 = 12x^3 - 46x^2 + 60x - 25 \Rightarrow x_1 = 1$, $x_2 = 5$

$x = 5$: $625 = 12 \cdot 125 - 46 \cdot 25 + 60 \cdot 5 - 25 \overset{125}{\Rightarrow} 25 = 60 - 46 + 12 - 1$ OK

$(x-1)(x-5) = x^2 - 6x + 5$

(Ch): $(x-1)^2(x-5)^2 = 0$

$x_1 = 1$ $x_2 = 1$ $x_3 = 5$ $x_4 = 5$

$$\begin{array}{l} x^4 - 12x^3 + 46x^2 - 60x + 25 : (x^2 - 6x + 5) = x^2 - 6x + 5 \\ \underline{+ -x^4 + 6x^3 - 5x^2} \\ -6x^3 + 41x^2 - 60x + 25 \\ \underline{+ 6x^3 - 36x^2 + 30x} \\ 5x^2 - 30x + 25 \\ \underline{+ -5x^2 + 30x - 25} \\ 0 \end{array}$$

ROZ$_p$: $p(n) = d_1 x_1^n + n \cdot d_2 \cdot x_2^n + d_3 x_3^n + n \cdot d_4 x_4^n$

$\boxed{p(n) = d_1 + n \cdot d_2 + d_3 \cdot 5^n + 5^n \cdot n \cdot d_4}$

(P): $a_0 = 2$ $a_1 = 1$ $a_2 = 8$ $a_3 = 28$

$$\begin{cases} 2 = d_1 + 0 + d_3 + 0 \Rightarrow d_1 = 2 - d_3 \\ 1 = d_1 + d_2 + 5d_3 + 5d_4 \\ 8 = d_1 + 2d_2 + 25d_3 + 50d_4 \\ 28 = d_1 + 3d_2 + 125d_3 + 375d_4 \end{cases} \Leftrightarrow \begin{cases} d_1 = 2 - d_3 \\ -1 = d_2 + 4d_3 + 5d_4 \Rightarrow d_2 = -1 - 4d_3 - 5d_4 \\ 6 = 2d_2 + 24d_3 + 50d_4 \\ 26 = 3d_2 + 124d_3 + 375d_4 \end{cases}$$

$$\Rightarrow \begin{cases} d_1 = 2 - d_3,\ d_2 = -1 - 4d_3 - 5d_4 \\ 8 = 16d_3 + 40d_4 \Rightarrow 2d_3 = 1 - 5d_4 \\ 29 = 72d_3 + 360d_4 \end{cases} \Leftrightarrow \begin{cases} d_1 = 2 - d_3,\ d_2 = -1 - 4d_3 - 5d_4 \\ 2d_3 = 1 - 5d_4 \\ -27 = 80d_4 \Rightarrow d_4 = -\frac{27}{80} \end{cases} \ldots$$

$d_4 = \frac{-27}{80}$ $d_3 = \frac{43}{32}$ $d_2 = \frac{-75}{16}$ $d_1 = \frac{-11}{16}$

$p(n) = -\frac{11}{16} - n \cdot \frac{75}{16} + \frac{43}{32} \cdot 5^n - \frac{27}{80} \cdot n \cdot 5^n$

zad domowe
$a_n = 16a_{n-1} - 9$
$a_0 = 7$

b) $a_n = -2a_{n-1} + 15a_{n-2} + 24$, $n \geq 2$ (P): $a_0 = 1$, $a_1 = -1$

I etap: (Ch): $x^2 = -2x + 15 \Rightarrow (x-3)(x+5) = 0 \Rightarrow x_1 = 3$ $x_2 = -5$

$\boxed{p(n) = d_1 \cdot 3^n + d_2 \cdot (-5)^n}$

I etap: $F_n = 24$ – f. stała więc przewidywane roz. Zn $q(n) = c$

$q(n) = -2q(n) + 15q(n) + 24 \Rightarrow c = -2c + 15c + 24 \Rightarrow c = -2 = q(n)$ ← RSZ$_n$

II etap: ROZ$_n$: $a_n = p(n) + q(n) \Rightarrow a_n = d_1 \cdot 3^n + d_2 \cdot (-5)^n - 2$

III etap: $a_0 = 1 = d_1 \cdot 3^0 + d_2 \cdot (-5)^0 - 2$ $a_1 = -1 = d_1 \cdot 3^1 + d_2 \cdot (-5)^1 - 2$

$$\begin{cases} 3 = d_1 + d_2 \\ 1 = 3d_1 - 5d_2 \end{cases} \Rightarrow \begin{cases} d_1 = 3 - d_2 \\ 1 = 9 - 3d_2 - 5d_2 \end{cases} \Rightarrow \begin{cases} d_2 = 1 \\ d_1 = 2 \end{cases}$$

Rozwiązanie ogólne: $a_n = 2 \cdot 3^n + (-5)^n - 2, \ n \geq 0$

(2_n): $a_n = 5a_{n-1} - 6a_{n-2} + 3n, \ n \geq 2$ (P) $a_0 = 2 \quad a_1 = 14$

I etap: $x^2 = 5x - 6 \Rightarrow (x-2)(x-3) = 0 \Rightarrow x_1 = 2 \ x_2 = 3$

$\underline{p_n = b_1 \cdot 2^n + b_2 \cdot 3^n}$

II etap: $f(n) = 3n$ - przewidujemy że $g(n) = a \cdot n + b$

$g_n = 5 \cdot g_{n-1} - 6g_{n-2} + 3n \Rightarrow an + b = 5 \cdot (a(n-1) + b) - 6(a(n-2) + b) + 3n$

$\Rightarrow an + b = (5a - 6a + 3)n + (-5a + 5b + 12a - 6b)$

$$\begin{cases} a = -a + 3 \\ b = 7a - b \end{cases} \Leftrightarrow \begin{cases} 2a = 3 \\ 2b = 7a \end{cases} \Leftrightarrow \begin{cases} a = \frac{3}{2} \\ b = \frac{21}{4} \end{cases} \quad RS2_n: \ g(n) = \frac{3}{2} \cdot n + \frac{21}{4}$$

III etap: $RO2_n$: $a_n = p(n) + g(n) \Rightarrow \underline{a_n = b_1 \cdot 2^n + b_2 \cdot 3^n + \frac{3}{2}n + \frac{21}{4}}$

IV etap: $a_0 = 2 \quad a_1 = 14$

$$\begin{cases} 2 = b_1 + b_2 + \frac{21}{4} \\ 14 = 2b_1 + 3b_2 + \frac{3}{2} + \frac{21}{4} \end{cases} \Leftrightarrow \begin{cases} b_1 + b_2 = -\frac{13}{4} \\ 2b_1 + 3b_2 = \frac{29}{4} \end{cases} \Leftrightarrow \begin{cases} \frac{29}{4} = 2b_1 + 3b_2 \\ -\frac{26}{4} = 2b_1 + 2b_2 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} b_2 = \frac{55}{4} \\ b_1 = -17 \end{cases}$$ Rozwiązanie: $a_n = -17 \cdot 2^n + \frac{55}{4} \cdot 3^n + \frac{3}{2}n + \frac{27}{4}$

(2_n): $a_n = 16a_{n-1} - 9$, (P): $a_0 = 7$

I etap: (Ch): $x_1 = 16 \qquad p(n) = d_1 \cdot x_1^n = d_1 \cdot 16^n$

II etap: $f(n) = 16$ przewidujemy że $g(n) = c$

103 $g_n = 16g_{n-1} - 9 \Rightarrow c = 16c - 9 \Rightarrow -15c = -9 \Rightarrow c = \frac{9}{15} = g(n)$

III etap: $RO2_n$: $a_n = p(n) + g(n) \Rightarrow a_n = d_1 \cdot 16^n + \frac{9}{15}$

IV etap: $a_0 = 7 = d_1 \cdot 16^0 + \frac{9}{15} \Rightarrow d_1 = 6\frac{6}{15}$

Rozwiązanie ogólne: $a_n = \frac{96}{15} \cdot 16^n + \frac{9}{15}$

Algorytmy i złożoność - ćwiczenia 04.04.2023 r.

1. (Z_n): $a_n = -6a_{n-1} - 9a_{n-2} + n^2 + 3n, \ n \geq 2$ (P) $a_0 = \frac{179}{128}, a_1 = -\frac{21}{128}$

I etap: (Ch): $x^2 = -6x - 9 \Rightarrow x^2 + 6x + 9 = 0 \Rightarrow (x+3)^2 = 0 \Rightarrow x_1 = -3$

$\underline{p(n) = d_1 \cdot (-3)^n + d_2 \cdot n \cdot (-3)^n}$

II etap: $f(n) = n^2 + 3n$, przewidujemy że $g(n) = an^2 + bn + c$

$g_n = -6g_{n-1} - 9g_{n-2} + n^2 + 3n \Rightarrow an^2 + bn + c = -6(a(n-1)^2 + b(n-1) + c) -$

$-9(a(n-2)^2 + b(n-2) + c) + n^2 + 3n \Rightarrow an^2 + bn + c = -6an^2 + 12an - 6a - 6bn + 6b$

$-9an^2+36an-36a-9bn+18b-9c+n^2+3n \Rightarrow an^2+bn+c = -3an^2+48an-42a-15bn+24b-15c+n^2+3n \Rightarrow an^2+bn+c = (-15a+1)n^2+(48a-15b+3)n-42a+24b-15c$

$$\begin{cases} a=-15a+1 \Rightarrow a=\frac{1}{16} \\ b=48a-15b+3 \\ c=-42a+24b-15c \end{cases} \Rightarrow \begin{cases} a=\frac{1}{16} \\ b=-15b+6 \Rightarrow b=\frac{6}{16} \\ c=-\frac{42}{16}+24b-15c \end{cases} \Rightarrow \begin{cases} a=\frac{1}{16} \\ b=\frac{6}{16} \\ c=-\frac{42}{16}+9-15c \end{cases} \Rightarrow \begin{cases} a=\frac{1}{16} \\ b=\frac{6}{16} \\ c=\frac{57}{8\cdot 16} \end{cases}$$

Rozwiązanie: $g_n = \frac{1}{16}n^2+\frac{3}{8}n+\frac{57}{128}$

II etap: ROZ_n: $a_n = p(n)+g(n) \Rightarrow a_n = d_1\cdot(-3)^n + d_2\cdot n\cdot(-3)^n + \frac{1}{16}n^2+\frac{3}{8}n+\frac{57}{128}$

III etap: $a_0 = \frac{179}{128} = d_1+\frac{57}{128} \Rightarrow d_1 = \frac{128}{128} = 1$

$a_1 = -\frac{21}{128} = -3\cdot 1 - 3d_2 + \frac{1}{16}+\frac{3}{8}+\frac{57}{128} \Rightarrow 3d_2 = -3+\underbrace{\frac{1}{16}+\frac{3}{8}}_{\frac{3}{16}}+\underbrace{\frac{57}{128}+\frac{21}{128}}_{\frac{78}{128}} \Rightarrow d_2 = -\frac{2}{3}$

Odpowiedź: $a_n = (-3)^n - \frac{2}{3}n\cdot(-3)^n + \frac{1}{16}n^2+\frac{3}{8}n+\frac{57}{128}$

i. (Z_n): $a_n = 4\cdot a_{n-1}+3n+2^n$, $n\geq 1$ (P): $a_0 = 4$

I etap: (Ch): $x_1 = 4$ $\quad p(n) = d_1\cdot x_1^n = d_1\cdot 4^n$

$(an+b)\cdot c\cdot 2^n = (\underbrace{ca}_{a_1}n+\underbrace{cb}_{b_1})\cdot 2^n$

II etap: $f(n) = 3n+2^n$ – przewidujemy że $g(n) = (an+b)\cdot 2^n$, bo

$g_n = 4\cdot g_{n-1}+3n\cdot 2^n \Rightarrow (an+b)\cdot 2^n = 4\cdot(a(n-1)+b)\cdot 2^{n-1}+3n\cdot 2^n \Rightarrow$

$\Rightarrow (an+b)\cdot 2^n = 2\cdot(an-a+b)\cdot 2^n+3n\cdot 2^n \Rightarrow (an+b)\cdot 2^n = (2an+3n-2a+2b)\cdot 2^n \Rightarrow$

$\Rightarrow (an+b)\cdot 2^n = ((2a+3)n+(2b-2a))\cdot 2^n$

$$\begin{cases} a=2a+3 \\ b=2b-2a \end{cases} \Rightarrow \begin{cases} a=-3 \\ -b=6 \end{cases} \Rightarrow \begin{cases} a=-3 \\ b=-6 \end{cases} \quad g_n = (-3n-6)\cdot 2^n$$

III etap: ROZ_n: $a_n = p(n)+g(n) \Rightarrow a_n = d_1\cdot 4^n+(-3n-6)\cdot 2^n \Rightarrow$

$\Rightarrow a_n = d_1\cdot 4^n - 3(n+2)\cdot 2^n$

IV etap: $a_0 = 4 = d_1-6 \Rightarrow d_1 = 10$

Odpowiedź: $a_n = 10\cdot 4^n - 3(n+2)\cdot 2^n$

Przykładowy sprawdzian:

1. Rozwiązać zależność rekurencyjną np: $a_n = 2a_{n-1}+3a_{n-2}+5^n$, $n\geq 2$ 4
$a_0 = -2$, $a_1 = 1$ (Mogą być wyższego stopnia i pierwiastki zespolone)
2. Rozważmy zależność rekurencyjną (Z_j): $a_n = ra_{n-1}+5a_{n-2}$, $n\geq 2$. 2
Niech x_1 będzie podwójnym pierwiastkiem równ. charakterystycznego. Wykazać, że $a_n = d\cdot n\cdot x_1^n$ jest rozw. (Z_j). 6
Wskazówka: $r = 2x_1$, $5 = -x_1^2$ (może być też o algorytmie

Euklidesa lub o wyprowadzeniu wzoru jawnego ciągu Fibonacci

3. Przedstawić 2759 w systemie o podstawie 2.

$2759_{(10)} = 2\cdot 10^3 + 7\cdot 10^2 + 5\cdot 10^1 + 9\cdot 10^0$

$(1001011)_2 = 1\cdot 2^6 + 0\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 32+4+1 =$

$37 = \underline{1} + \underbrace{36}_{18\cdot 2}$, $18 = \underline{0} + \underbrace{18}_{9\cdot 2}$, $9 = \underline{1} + \underbrace{8}_{4\cdot 2}$, $4 = \underline{0} + \underbrace{4}_{2\cdot 2}$, $2 = \underline{0} + \underbrace{2}_{1\cdot 2}$, $1 =$

5. $(2759)_{10} = (?)_2$

$2759 = \underline{1} + 2\cdot \underbrace{1079}_{1\cdot 2^0}$

$1079 = \underline{1} + 2\cdot \underbrace{539}_{1\cdot 2^1}$

$539 = \underline{1} + 2\cdot \underbrace{269}_{1\cdot 2^2}$

$269 = \underline{1} + 2\cdot \underbrace{134}_{1\cdot 2^3}$

$134 = \underline{0} + 2\cdot \underbrace{67}_{1\cdot 2^4}$

$67 = \underline{1} + 2\cdot \underbrace{33}_{1\cdot 2^5}$

$33 = \underline{1} + 2\cdot \underbrace{16}_{1\cdot 2^6}$

$16 = \underline{0} + 2\cdot \underbrace{8}_{0\cdot 2^7}$

$8 = \underline{0} + 2\cdot 4$

$4 = \underline{0} + 2\cdot 2$

$2 = \underline{0} + 2\cdot 1$ $\quad 0\cdot 2^{10}$

$1 = \underline{1} + 2\cdot 0$ $\quad 0\cdot 2^{11}$

$2759_{(10)} = (100001101111)_2$

Algorytmy i złożoność - wykład 05.04.2023 r.

Twierdzenie o rekurencji uniwersalnej:

Zakładamy, że $a, b \geq 1$, $f: \mathbb{N} \to \mathbb{R}_+$, $T: \mathbb{N}_+ \to \mathbb{R}_+$

Jeżeli $T(n) = a\cdot T(\frac{n}{b}) + f(n)$, dla $n>0$, gdzie $\frac{n}{b}$ jest interpretowane jako $\lceil \frac{n}{b} \rceil$ lub $\lfloor \frac{n}{b} \rfloor$, to

1. Jeżeli $f(n) \in O(n^{(\log_b a)-\varepsilon})$ dla pewnej stałej $\varepsilon > 0$, to $T(n) \in \Theta(n^{\log_b a})$.
2. Jeżeli $f \in \Theta(n^{\log_b a})$, to $T(n) \in \Theta(n^{\log_b a} \lg n)$.
3. Jeżeli $f \in \Omega(n^{\log_b a + \varepsilon})$, dla pewnej stałej $\varepsilon > 0$ oraz $a\cdot f(\frac{n}{b}) \leq c\cdot f(n)$, dla pewnej stałej $c>1$ i prawie wszystkich, to $T(n) \in \Theta(f(n))$.

Wyjaśnienie oznaczeń: $x \in \mathbb{R}$ $\lfloor x \rfloor$ - cecha z x, floor of x

$\lfloor x \rfloor = \max\{n \in \mathbb{Z} : n \leq x\}$ np: $\lfloor 3 \rfloor = 3$, $\lfloor 2{,}8 \rfloor = 2$

$\lceil x \rceil = \min\{n \in \mathbb{Z} : x \leq n\}$ np: $\lceil 2{,}9 \rceil = 3$, $\lceil 3 \rceil = 3$, $\lceil 3{,}01 \rceil = 4$

Dla $f, g: \mathbb{N} \to \mathbb{R}_+$ oznaczenie $f \in \Omega(g) \iff g \in O(f)$

$\frac{f}{g} \to 0 \Rightarrow g < f$, $g \in O(f)$, $f \notin O(g)$, $f \in \Omega(g)$, $f \in O(g)$

Przykład 1: $T(n) = 9\cdot T(\frac{n}{3}) + n$ $\quad a=9, b=3, f(n)=n$

$\log_b a = \log_3 9 = 2$. $n^{\log_b a} = n$?

$f \in O(n^{2-\varepsilon})$? $n \in O(n^{2-\varepsilon})$? np. $\varepsilon = 1$, TAK z twierdzenia o rekurencji uniwersalnej $T(n) \in \Theta(n^2)$.

Przykład 2: $T(n) = T(\frac{2n}{3}) + 1$, $a = 1$, $b = \frac{3}{2}$, $f(n) = 1$

$\log_b a = 0$ $f(n) \in \Theta(n^0 = 1)$ $1 \in \Theta(1) = \Theta(n^{\log_b a})$

Z tw. o rek. uniw.: $T(n) \in \Theta(\lg n)$

Przykład 3: $T(n) = 3T(\frac{n}{4}) + n\lg n$, $a = 3$, $b = 4$, $f(n) = n\lg n$

$\log_b a = \log_4 3 = \frac{1}{2}\log_2 3 = \frac{1}{2}\lg 3$ $n^{\log_b a} = n^{\frac{\lg 3}{2}}$ $\underbrace{\lg 2}_{1} < \lg 3 < \underbrace{\lg 4}_{2}$

$\exists \varepsilon > 0$; $\frac{\lg 3}{2} + \varepsilon = 1$ $\frac{1}{2} < \frac{\lg 3}{2} < 1$

$n^{\frac{\lg 3}{2} + \varepsilon} = n$, $n < n\lg n$, $n \in O(n\lg n)$, $n\lg n \in \Omega(n)$

$f(n) \in \Omega(n^{\log_b a + \varepsilon})$ $\exists c > 0$: $a \cdot f(\frac{n}{b}) \le c \cdot f(n)$?

$3 \cdot f(\frac{n}{4}) = 3 \cdot (\frac{n}{4} \cdot \lg \frac{n}{4}) = \frac{3n}{4}(\lg n - \lg 4) = \frac{3n}{4}(\lg n - 2) < \frac{3}{4} n\lg n$ $c < 1$

zatem dla $c = \frac{3}{4} < 1$ $a \cdot f(\frac{n}{b}) \le c \cdot f(n)$ OK

Z tw. o rek. uniw. $T(n) \in \Theta(n\lg n)$.

Przykład 4: $T(n) = 2 \cdot T(\frac{n}{2}) + n\lg n$ $a = 2$, $b = 2$, $f(n) = n\lg n$

$\log_b a = 1$ $n^{\log_b a} = n^1 = n$ $f(n)$? $n^{\log_b a}$? $n\lg n \in \Omega(n)$

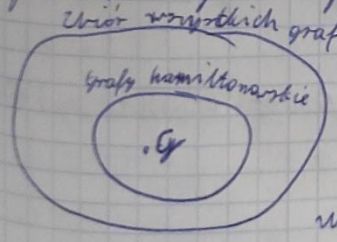
Czy $\exists \varepsilon > 0$ $n\lg n \in \Omega(n^{\log_b a + \varepsilon})$? Nie bo gdyby TAK, to

$\frac{n^{\log_b a + \varepsilon}}{n\lg n} = \frac{n^{1+\varepsilon}}{n\lg n} = \frac{n^{\varepsilon}}{\lg n} \xrightarrow{?} 0$, ale takiego ε nie ma.

W tym przykładzie tw. o rek. uniw. się nie stosuje.

Pojęcie algorytmu

Przykład: Czy dany graf jest hamiltonowski?



Intuicyjnie: Algorytm to pewna procedura, które dla właściwych danych wejściowych (input) produkuje dane wyjściowe (output, wynik) spełniające zadane wcześniej oczekiwane.

Precyzyjna definicja algorytmu – lata 30-te XX w. m. in. Turing

Definicja: Alfabet – niepusty zbiór symboli, u nas skończony

niepusty zbiór symboli np: $\Sigma = \{0, 1\}$, Σ^* – zbiór słów

Definicja: Słowem nad alfabetem Σ nazywamy dowolny skończony (dopuszczamy pusty) ciąg symboli ze zbioru Σ.

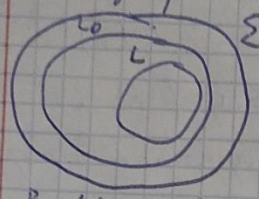
Uwaga 1: Ciąg $(\sigma_1, \sigma_2, \dots, \sigma_n)$ zapisujemy jako $\sigma_1\sigma_2 \dots \sigma_n$ (string)

Uwaga 2: Symbolem λ (lub ε) oznaczamy ciąg pusty.

Uwaga 3: Konkatenacja słów $\sigma_1 \dots \sigma_n$ i $\tilde{\sigma}_1 \dots \tilde{\sigma}_m$, to słowo $\sigma_1 \dots \sigma_n \tilde{\sigma}_1 \dots \tilde{\sigma}_m$. $w_1 \triangle w_2 \equiv \sigma_1 \dots \sigma_n \tilde{\sigma}_1 \dots \tilde{\sigma}_m$, gdzie $w_1 \equiv \sigma_1 \dots \sigma_n$, $w_2 \equiv \tilde{\sigma}_1 \dots \tilde{\sigma}_m$. Działanie $\triangle$ jest łączne i ma el. neutralny: λ

Uwaga 4: Σ^* = zbiór wszystkich słów nad Σ

Definicja: Językiem nad alfabetem Σ (alfabet) nazywamy dowolny podzbiór zbioru Σ^*.



L_0 – zbiór dopuszczalnych kodów

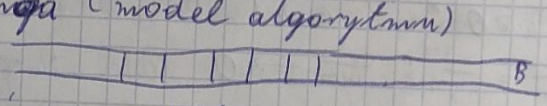
L – zbiór kodów akceptowalnych w danym problemie

Problem decyzyjny to pewien język.

Pytamy, czy dany dopuszczalny kod (egzemplarz problemu) należy do języka, tzn czy jest „zaakceptowany" jako należący do języka.

Maszyna Turinga (model algorytmu)

Obrazowo:



- Nieskończona taśma, na której tylko skończona liczba „komórek" jest zapisana. Pozostałe są puste
- Alfabet Σ, w którym zapisujemy kody (słowa) wejściowe i wyjściowe.
- Alfabet pomocniczy Γ (tzn. alfabet taśmowy) $\Sigma \subseteq \Gamma$
- Symbol $B \in \Gamma$, $B \notin \Sigma$
- Zbiór stanów maszyny: Q
- q_0 – wyróżniony stan, zwany początkowym
- $F \subseteq Q$: zbiór stanów końcowych (akceptujących)
- $\delta: (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, P\}$ – funkcja przejścia

$(q, \gamma) \xrightarrow{\delta} (\tilde{q}, \tilde{\gamma}, D)$ Interpretacja: Będąc w stanie q i czytając na taśmie symbol γ: 1) przejdź do stanu $\tilde{q}$, 2) wpisz $\tilde{\gamma}$ zamiast γ 3) przesuń głowicę czytającą w

komórka D np: $\boxed{1|0|}$ $\delta(q_0,1) = (q_1, X, P)$

X - symbol z Γ, którego mogło nie być w Σ $\boxed{1|1|0|} \xrightarrow{\delta} \boxed{X|0|}$ q_1

- głowica czytająco - pisząca, która może się przesuwać o 1 komórkę w Prawo lub Lewo (P lub L) i marać, wypisywać symbole i czytać.

$\not\to$ - oznaczenie na funkcję częściową, to znaczy np: $f: A \not\to B \iff$ istnieje podzbiór $D \subseteq A$ taki że f nie jest funkcją z D do B. W A mogą istnieć elementy (poza D), dla których f nie jest określona.

Algorytmy i złożoność - ćwiczenia 18.04.2023 r.

1. Przedstawić liczbę 117 w systemach o podstawach a) 5 i b) 3.

a) $117_{10} = \ldots\ldots_5$ $\quad 117 = 23 \cdot 5 + \underline{2}$, $\quad 23 = 4 \cdot 5 + \underline{3}$,

$4 = 0 \cdot 5 + \underline{4}$, zatem $(117)_{10} = (432)_5$

b) $117_{10} = \ldots\ldots_3$ $\quad 117 = 39 \cdot 3 + \underline{0}$, $\quad 39 = 13 \cdot 3 + 0$,

$13 = 4 \cdot 3 + \underline{1}$, $\quad 4 = 1 \cdot 3 + \underline{1}$, $\quad 1 = 3 \cdot 0 + \underline{1} \Rightarrow (117)_{10} (11100)_3$

Układy o podstawie 2, 4, 8 ✓, ale o podstawie 16 mamy cyfry: $0, 1, \ldots, 8, 9, \underset{10}{A}, \underset{11}{B}, \underset{12}{C}, \underset{13}{D}, \underset{14}{E}, \underset{15}{F}$

2. Przedstawić liczbę 21469 w systemach o podstawach a) 8, b) 16

a) $21469_{10} = \ldots\ldots_8$ $\quad 21469 = 2683 \cdot 8 + \underline{5}$,

$2683 = 335 \cdot 8 + \underline{3}$, $\quad 335 = 41 \cdot 8 + \underline{7}$,

$41 = 5 \cdot 8 + \underline{1}$, $\quad 5 = 8 \cdot 0 + \underline{5} \Rightarrow (51735)_8$

b) $21469_{10} = \ldots\ldots_{16}$ $\quad 21469 = 1341 \cdot 16 + \underline{13}$,

$1341 = 83 \cdot 16 + \underline{13}$, $\quad 83 = 5 \cdot 16 + \underline{3}$, $\quad 5 = 0 \cdot 16 + \underline{5}$

$(21469)_{10} = (53DD)_{16}$

3. Przedstawić liczbę 4034 w systemach 2, 4, 8, 16.

$4034 = 2 \cdot 2017 + 0$, $2017 = 2 \cdot 1008 + 1$, $1008 = 2 \cdot 504 + 0$,

$504 = 2 \cdot 252 + 0$, $252 = 2 \cdot 126 + 0$, $126 = 2 \cdot 63 + 0$, $63 = 2 \cdot 31 + 1$,

$31 = 2 \cdot 15 + 1$, $15 = 2 \cdot 7 + 1$, $7 = 2 \cdot 3 + 1$, $3 = 2 \cdot 1 + 1$, $1 = 2 \cdot 0 + 1$

$(4034)_{10} = (111111000010)_2$ $\quad (4034)_{10} = \ldots\ldots_4$

$4034 = 4 \cdot 1008 + 2$, $1008 = 4 \cdot 252 + 0$, $252 = 4 \cdot 63 + 0$,

$63 = 4 \cdot 15 + 3$, $15 = 4 \cdot 3 + 3$, $3 = 4 \cdot 0 + 3$

$(4034)_{10} = (333002)_4$

$(\underbrace{11}_{3},\underbrace{11}_{3},\underbrace{11}_{3},\underbrace{00}_{0},\underbrace{00}_{0},\underbrace{10}_{2})_2 \rightarrow (10)_2 = 0 \cdot 2^0 + 1 \cdot 2^1 = (2)_{10} = (2)_4$

$(00)_2 = 0$, $(01)_2 = 1$, $(10)_2 = 2$, $(11)_2 = 3$

$(111111000010)_2 = (333002)_4$

$(\underbrace{111}_{7},\underbrace{111}_{7},\underbrace{000}_{0},\underbrace{010}_{2})_2 \rightarrow (010)_2 = 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 = (2)_{10} = (2)_8$

$(100)_2 = 4$, $(101)_2 = 5$, $(110)_2 = 6$, $(111)_2 = 7$

$(111111000010)_2 = (7702)_8$

Spr. $(7702)_8 = 2 \cdot 8^0 + 0 \cdot 8^1 + 7 \cdot 8^2 + 7 \cdot 8^3 = 2 + 7 \cdot 64 + 7 \cdot 512 =$

$= 3584 + 448 + 2 = (4034)_{10}$

$(\underbrace{1111}_{F},\underbrace{1100}_{C},\underbrace{0010}_{2})_2 \rightarrow 0010 = 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 = (2)_{10}$

$(1000)_2 = 8$, $(1001)_2 = 9$, $(1010)_2 = A$, $(1011)_2 = B$, $(1100)_2 =$

$(1101)_2 = D$, $(1110)_2 = E$, $(1111)_2 = F$

$(111111000010)_2 = (FC2)_{16}$

~~$(7702)_8 \rightarrow (010)_8 = 2 \cdot 8^0 + 0 \cdot 8^1 = 2_{10} =$~~

~~$(77)_8 = 7 \cdot 8^0 + 7 \cdot 8^1 = 7 + 56 = 63_{10} =$~~

4. Przedstawić liczbę $(185178)_{10}$ w systemach 2, 4, 8, 16.

$185178 = 2 \cdot 92589 + \underline{0}$, $92589 = 2 \cdot 46294 + \underline{1}$, $46294 = 2 \cdot 23147 + \underline{0}$,

$23147 = 2 \cdot 11573 + \underline{1}$, $11573 = 2 \cdot 5786 + \underline{1}$, $5786 = 2 \cdot 2893 + \underline{0}$,

$2893 = 2 \cdot 1446 + \underline{1}$, $1446 = 2 \cdot 723 + \underline{0}$, $723 = 2 \cdot 361 + \underline{1}$,

$361 = 2 \cdot 180 + \underline{1}$, $180 = 2 \cdot 90 + \underline{0}$, $90 = 2 \cdot 45 + \underline{0}$, $45 = 2 \cdot 22 + \underline{1}$,

$22 = 2 \cdot 11 + \underline{0}$, $11 = 2 \cdot 5 + \underline{1}$, $5 = 2 \cdot 2 + \underline{1}$, $2 = 2 \cdot 1 + \underline{0}$, $1 = 0 \cdot 2 + \underline{1}$

$(185178)_{10} = (101101001101011010)_2$

Zad. Dom.

$5748 \rightarrow (\ )_2$ $\quad (\underbrace{101}_{5},\underbrace{101}_{5},\underbrace{001}_{1},\underbrace{101}_{5},\underbrace{011}_{3},\underbrace{010}_{2})_2 = (551532)_8$

$(\ )_4$

$(\ )_8$ $\quad (\underbrace{10}_{2},\underbrace{1101}_{D},\underbrace{0011}_{3},\underbrace{0101}_{5},\underbrace{1010}_{A})_2 = (2D35A)_{16}$

$(\ )_{16}$

Ile kroków obliczeniowych trzeba zrobić aby obliczyć mnożenie n liczb?

Twierdzenie o rekurencji uniwersalnej:

$T(n) = a \cdot T(\frac{n}{b}) + f(n)$ dla $n > 0$, $a, b \geq 1$,

1) Jeżeli: $f(n) \in O(n^{(\log_b a) - \varepsilon})$ dla pewnej stałej $\varepsilon > 0$ to $T(n) \in \Theta(n^{\log_b a})$

2) jeżeli $f \in \Theta(n^{\log_b a})$, to $T(n) \in \Theta(n^{\log_b a} \lg n)$

3) jeżeli $f \in \Omega(n^{\log_b a + \varepsilon})$ dla $\varepsilon > 0$ oraz $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$, dla pewnej stałej $c < 1$ i prawie wszystkich n to $T(n) \in \Theta(f(n))$

0 n cyfrowych

5. a) $T(n) = 4 \cdot T(\frac{n}{2}) + n$, b) $T(n) = 4 \cdot T(\frac{n}{2}) + n^2$, c) $T(n) = 4 \cdot T(\frac{n}{2}) + n^3$

rozkłamy $T(n) \in \Theta(?)$.

a) $a = 4$, $b = 2$, $f(n) = n$ $\quad \log_b a = \log_2 4 = 2$

$n^{\log_b a} = n^{\log_2 4} = n^2$ $\quad n \in O(n^{2+\varepsilon})$ - czy jest takie ε że jest to prawda? tak np: $\varepsilon = 1$ wtedy $n \in O(n^1)$. Zatem na podstawie tw. o rek. uniw. $T(n) \in \Theta(n^{\log_b a}) \Rightarrow T(n) \in \Theta(n^2)$

b) $a = 4$, $b = 2$, $f(n) = n^2$ $\quad \log_b a = \log_2 4 = 2$

$n^{\log_b a} = n^2$ czy $n \in \Theta(n^2)$? TAK jest to prawda. Zatem z tw. o rek. uniw. $T(n) \in \Theta(n^2 \cdot \lg n)$

c) $a = 4$, $b = 2$, $f(n) = n^3$ $\quad \log_b a = 2$ $\quad n^{\log_b a} = n^2$

czy $n^3 \in \Omega(n^{2+\varepsilon})$? to znaczy czy $f \in \Omega(g) \Leftrightarrow g \in O(f)$?

czy $n^{2+\varepsilon} \in O(n^3)$? TAK np: dla $\varepsilon = 1$

czy $a \cdot f(\frac{n}{b}) \leq c \cdot f(n)$ (?), $c < 1$ to znaczy czy

$4 \cdot (\frac{n}{2})^3 \leq c \cdot n^3$ (?), czyli czy $\frac{1}{2} n^3 \leq c \cdot n^3$ (?) TAK, dla $c = \frac{1}{2} < 1$

Zatem na podstawie tw. o rek. uniw. $T(n) \in \Theta(n^3)$

zad. dom. $T(n) = T(\frac{n}{2}) + \Theta(1)$ Rozwiązać (Odp: $T(n) = \Theta(\lg n)$)

6. Przedstawić liczbę 5748 w systemach 2, 4, 8, 16.

$5748 = 2 \cdot 2874 + \underline{0}$, $2874 = 2 \cdot 1437 + \underline{0}$, $1437 = 2 \cdot 718 + \underline{1}$,

$718 = 2 \cdot 359 + \underline{0}$, $359 = 2 \cdot 179 + \underline{1}$, $179 = 2 \cdot 89 + \underline{1}$, $89 = 2 \cdot 44 + \underline{1}$,

$44 = 2 \cdot 22 + \underline{0}$, $22 = 2 \cdot 11 + \underline{0}$, $11 = 2 \cdot 5 + \underline{1}$, $5 = 2 \cdot 2 + \underline{1}$, $2 = 2 \cdot 1 + \underline{0}$,

$1 = 2 \cdot 0 + \underline{1}$ $\quad (5748)_{10} = (1011001110100)_2$

$(\underset{1}{1}\underset{1}{0}\underset{2}{1}\underset{1}{1}\underset{1}{0}\underset{3}{0}\underset{1}{1}\underset{1}{1}\underset{0}{0}\underset{0}{0})_2 = (11213310)_4$

$(\underset{1}{1}\underset{3}{0}\underset{1}{1}\underset{1}{0}\underset{1}{0}\underset{6}{1}\underset{1}{1}\underset{4}{0}\underset{0}{0})_2 = (13164)_8$ $\quad (\underset{1}{1}\underset{6}{0}\underset{1}{1}\underset{7}{0}\underset{0}{0}\underset{1}{1}\underset{4}{0}\underset{0}{0})_2 = (1674)_{16}$

Algorytmy i złożoność - 19.04.2023 wykład

Σ - ustalony alfabet, Σ^* - zbiór słów nad alfabetem Σ

$L \subseteq \Sigma^*$ - język $\quad$ Problem decyzyjny: Pytanie czy dla danego $w \in \Sigma^*$ $w \in L$. $\quad$ Skrótowo: Problem decyzyjny $\equiv$ język.

Definicja: Maszyną Turinga nazywamy układ

$M = \langle \Sigma, Q, q_0, \Gamma, B, \delta, F \rangle$, gdzie: Σ - alfabet,
Q - zbiór, elementy Q nazywamy stanami,
$q_0 \in Q$ - q_0 nazywamy stanem początkowym,
Γ - zbiór symboli, zwany alfabetem taśmowym, $\Sigma \subseteq \Gamma$,
$B \in \Gamma \setminus \Sigma$ - symbol pustej klatki (blank),
$F \subseteq Q$, $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, P\}$ - zwana funkcją przejścia (transition function).

Definicja: Konfiguracją maszyny Turinga (m.T.) M nazywamy trójką (w, q, i), takie że:
$w \in \Gamma^*$; $w \equiv \sigma_1, \dots, \sigma_n$; $i \in \{0, \dots, n+1\}$, $q \in Q$.

Interpretacja: Maszyna znajdując się w konfiguracji $C = (w, q, i)$ jest w stanie q, na taśmie znajduje się $B \dots B\, w\, B \dots B$, a głowica czyta symbol σ_i.

Konfiguracja: $\boxed{\sigma_1 \mid \sigma_2 \mid \dots \mid \sigma_i \mid \dots \mid \sigma_n}$ (w, q, i) $w \equiv \sigma_1 \dots \sigma_n \equiv B\sigma_1 \dots \sigma_n B$

„Przejście" między konfiguracją C_1 a C_2 jest możliwe tylko zgodnie z funkcją δ, to znaczy: $C_1 \vdash C_2$
Z konfiguracji C_1 można przejść do konfiguracji C_2 wtw. $C_1 = (w_1, q^1, i)$, $C_2 = (w_2, q^2, j)$ $w_1 \equiv \sigma_1 \dots \sigma_n \equiv B\sigma_1 \dots \sigma_n$, $w_2 \equiv \tilde{\sigma}_1 \dots \tilde{\sigma}_m$ oraz jeżeli $\delta(q^1, \sigma_i) = (q^2, \gamma, P)$ to $w_2 = \sigma_1 \dots \sigma_{i-1} \gamma \sigma_{i+1} \dots \sigma_n$, $j = i+1$; a jeżeli $\delta(q^1, \sigma_i) = (q^2, \gamma, L)$, to $w_2 = \sigma_1 \dots \sigma_{i-1} \gamma \sigma_{i+1} \dots \sigma_n$, $j = i-1$.

$\sigma_1 \sigma_2 \dots \underset{\uparrow q^1}{\sigma_i} \dots \sigma_n \xrightarrow[\delta(\dots) = (\dots, P)]{} \boxed{\sigma_{i-1} \mid \gamma \mid \underset{\uparrow q^2}{\sigma_{i+1}}}$ | $\sigma_1 \dots \sigma_i \dots \sigma_n \xrightarrow[\delta(\dots) = (\dots, L)]{} \boxed{\sigma_{i-1} \mid \gamma \mid \underset{\uparrow q^2}{\sigma_{i+1}}}$

Obliczenie M.T. M na słowie $w \equiv \sigma_1 \dots \sigma_n$, to ciąg konfiguracji: $C_1 C_1 \dots C_m$, takich że: $C_0 \vdash C_1 \vdash \dots \vdash C_m$;
$C_0 = (w, q_0, 1)$ - (maszyna na swojej taśmie ma $\dots B \dots B\, w\, B \dots$ czyta symbol σ_1, znajduje się w stanie początkowym q_0);
$C_m = (\tilde{w}, q, j)$, gdzie $q \in Q$; funkcja δ nie jest określona dla $(\tilde{\sigma}_j, q)$, gdzie $\tilde{\sigma}_j$ jest symbolem na pozycji j w słowie $\tilde{w}$.
Gdy $q \in F$, to mówimy, że obliczenie akceptuje słowo w.
Język L_M to zbiór wszystkich słów akceptowalnych

przez M. Mówimy go przyjętym akceptowalnym (rozpoznawanym) przez M.

Definicja: Język L nazywamy rozstrzygającym wtw, gdy istnieje maszyna Turinga M taka że $L = L_M$ i M zatrzymuje się na każdym słowie wejściowym.

Przykład: M: $Q = \{q_0, q_1\}$, $F = \{q_0\}$, $\Sigma = \{0\}$, $\Gamma = \{0, X, Y, B\}$

δ	0	X	Y	B
q_0	X, q_1, P			B, q_0, L
q_1	Y, q_1, P			

$\underset{q_0}{0}000 \vdash X\underset{q_1}{0}00 \vdash XY\underset{q_0}{0}0 \vdash XYX\underset{q_1}{0} \vdash$

$\vdash XYXY\underset{q_0}{B} \vdash XYX\underset{q_a}{Y}$ akceptuje

$\boxed{B|\dots|B} \vdash \boxed{B|B|\dots|B}$ akceptuje

$\underset{q_0}{0}00 \vdash X\underset{q_1}{0}0 \vdash XY\underset{q_0}{0} \vdash XYX\underset{q_1}{B}$ zatrzymuje się nie akceptując

$L_M = \{w \in \{0\}^* :$ długość w jest liczbą parzystą$\}$

$L_1 = \{0\dots01\dots1 : n \in \mathbb{N}\}$ skrót $\underbrace{0\dots0}_{n} = 0^n$ $L_1 = \{0^n1^n : n \in \mathbb{N}\}$

$L_2 = \{0^n1^m2^{n+m} : m, n \in \mathbb{N}\}$ $0^n1^m2^p$ akceptowane $\Leftrightarrow p = n+m$

Dla L_1:

	0	1	X	Y	B
q_0	(q_1, X, P)	–	–	(q_3, Y, P)	
q_1	$q_1, 0, P$	q_2, Y, L		q_1, Y, P	
q_2	$q_2, 0, L$		q_0, X, P	q_2, Y, L	
q_3				q_3, Y, P	q_4, B, P
q_4	–	–	–	–	

$\overset{q_0}{0}011 \vdash X\overset{q_1}{0}11 \vdash X0\overset{q_1}{1}1 \vdash X0\overset{q_2}{Y}1 \vdash$

$\vdash X\overset{q_2}{0}Y1 \vdash X\overset{q_0}{0}Y1 \vdash XX\overset{q_1}{Y}1 \vdash$

$\vdash XXY\overset{q_1}{1} \vdash XXY\overset{q_2}{Y} \vdash XX\overset{q_2}{Y}Y \vdash$

$\vdash XX\overset{q_0}{Y}Y \vdash XXY\overset{q_3}{Y} \vdash XXYY\overset{q_4}{B}$

$F = \{q_4\}$, q_4 – akceptuje

$\overset{q_0}{0}11 \vdash X\overset{q_1}{1}1 \vdash X\overset{q_2}{Y}1 \vdash X\overset{q_0}{Y}1 \vdash XY\overset{q_3}{1}$

zatrzymuje się nie akceptując

Zatem 011 nie jest akceptowalne.

Definicja: $f: \Sigma^* \to \Sigma^*$ jest nazywana funkcją obliczalną wtw, gdy istnieje m. T. M, zatrzymująca się na każdym wejściu, taka że dla dowolnego słowa $w \in \Sigma^*$, obliczenie maszyny M na słowie w kończy się tak, że na taśmie napisane jest słowo $v = f(w)$.

Ćw. Algorytmy i złożoność – ćwiczenia 25.04.2023 r.

$B0\underbrace{11\dots1}_{n \text{ jedynek – koduje liczbę } n}0B$

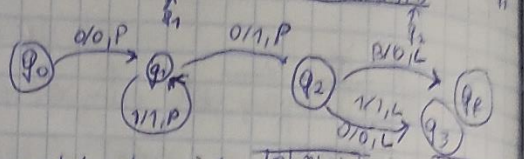
1. Zaprogramować maszynę Turinga, która przekształca ciąg postaci

n jedynek $0 1^n 0$ na $0 1^{n+1} 0$. (dodaje „1" do zako-dowanej liczby naturalnej n).

Σ - alfabet wejściowy $\Sigma = \{0, 1\}$, F - lista stanów końcowych

Q \ P	0	1	B
q_0	$q_1, 0, P$		
q_1	$q_2, 1, P$	$q_1, 1, P$	
q_2	$q_3, 0, L$	$q_3, 1, L$	$q_f, 0, L$
q_3			
q_f			

np: $\boxed{0|1|1|0}$ $\xrightarrow{\text{mamy dodać}}$ $\boxed{0|1|1|0}$ $\mapsto$ $\boxed{0|1|1|0}$ $\mapsto$ $\boxed{0|1|1|0}$ $\mapsto$ $\boxed{0|1|1|1}$ $\mapsto$ $\boxed{0|1|1|1}$



Przypadek gdy jest $\boxed{0|1^n|0|1}$ $\mapsto$ $\boxed{0|1^n|1|1}$ $\mapsto$ $\boxed{0|1|1|1}$

$Q = \{q_0, q_1, q_2, q_3, q_f\}$ $F = \{q_f\}$ $\Gamma = \Sigma \cup \{B\}$ $\delta \to$ tabel.

$\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, P\}$

Przetestować naszą maszynę Turinga na słowach:

a) 010 b) 00 c) 110 d) 01B10

ad. a) $\boxed{0|1|0}$ $\mapsto$ $\boxed{0|1|0}$ $\mapsto$ $\boxed{0|1|0}$ $\mapsto$ $\boxed{0|1|1|B}$ $\to$ $\boxed{0|1|1|0}$ akceptuje

ad. b) $\boxed{0|0}$ $\mapsto$ $\boxed{0|0}$ $\to$ $\boxed{0|1|B}$ $\mapsto$ $\boxed{0|1|0}$ akceptuje

ad. c) $\boxed{1|1|0|B}$ odrzuca (nie akceptuje)

ad. d) $\boxed{0|1|B|1|0}$ $\mapsto$ $\boxed{0|1|B|1|0}$ $\to$ $\boxed{0|1|B|1|0}$ odrzuca

2. Zaprojektować maszynę Turinga, która akceptuje dokładnie ciągi postaci $0 1^n 2^m 3^{n+m} 0$, $\Sigma = \{0, 1, 2, 3\}$

np: 011 222 333330 - maszyna powinna akceptować

Q \ P	0	1	2	3	B	X	Y
q_0	$q_1, 0, P$						
q_1	$q_f, 0, L$	q_2, X, P	q_4, X, P				q_1, Y, P
q_2		$q_2, 1, P$	$q_2, 2, P$	q_3, Y, L			
q_3		$q_3, 1, L$	$q_3, 2, L$			q_1, X, P	q_3, Y, L
q_4			$q_4, 2, P$	q_3, Y, L			q_4, Y, P
q_f							

$\overset{XX}{0 1 1} 222 \overset{YY}{333330} \mapsto^* 0XX\underset{q_1}{2}22YY333 \mapsto 0XXX\underset{q_4}{2}2YY333 \mapsto^*$

$\to^* 0XXX\underset{q_3}{2}2YYY330 \mapsto$

2. Dom.

a) 0110

b) 01010

zaprojektować M.T. która akceptuje tylko ciągi które są postaci $01^n01^{\dots}10$ ($2 \cdot n$)

zaprojektować M.T. która dla ciągu 01^n0 obliczy $01^{2n}0$

np. $00 \mapsto^* 00$

$010 \mapsto^* 0110$

$0110 \mapsto^* 011110$

012330

$0\,1222333330 \mapsto 0\,1222333330 \mapsto 0X1222B33330 \mapsto {}^{*}0X1222333330 \mapsto {}^{*}0X1222Y33330$

$\mapsto {}^{*}0X1222Y33330 \mapsto 0X1222Y33330 \mapsto 0XX222Y33330 \mapsto {}^{*}0XX222Y33330 \mapsto$

$\mapsto 0XX222YB3330 \mapsto 0XX222YYB330 \mapsto {}^{*}0XX222YY3330 \mapsto 0XX222YY3330 \mapsto$

$\mapsto 0XXX22YY3330 \mapsto {}^{*}0XXX22YY3330 \mapsto 0XXX22YYY330 \mapsto {}^{*}0XXX22YYY330 \mapsto$

$\mapsto 0XXX22YYY330 \mapsto 0XXXX2YYY330 \mapsto {}^{*}0XXXX2YYY330 \mapsto 0XXXX2YYYY30 \mapsto$

$\mapsto {}^{*}0XXXX2YYYY30 \mapsto$

$\mapsto 0XXXX2YYYY30 \mapsto 0XXXXXYYYY30 \mapsto {}^{*}0XXXXXYYYY30 \mapsto 0XXXXXYYYYY0 \mapsto$

$\mapsto {}^{*}0XXXXXYYYYY0 \mapsto 0XXXXXYYYYY0 \mapsto 0XXXXXYYYYY0 \mapsto {}^{*}0XXXXXYYYYY0 \mapsto$

$\mapsto 0XXXXXYYYYY0$ akceptuje q_f - stan końcowy

Algorytmy i złożoność - wykład 26.04.2023r.

Dodawanie dwóch liczb n-cyfrowych. Ile potrzeba dodawań dwóch cyfr, by dodać 2 liczby n-cyfrowe?

$(a_1 a_2 \ldots | a_{n-1} | a_n)_b$ - zapis liczby b_1 w układzie a o podstawie b

$(b_1 b_2 \ldots | b_{n-1} | b_n)_b$ - zapis liczby b_2 —||—

c_{n-1} - rezultat dodawania dwóch cyfr na pozycji n-1

$c_n = (a_n + b_n) \bmod b$ (jedno dodawanie) może być cyfra przeniesienia d_{n-1}

Inaczej: $2(n-1)$ dodawań + 1 (pozycja n)

Odp: Dodawanie dwóch liczb n-cyfrowych wymaga maksymalnie $2n+1$ dodawań pojedynczych cyfr ($\Theta(n)$)

Ile pojedynczych dodawań wymaga (maksymalnie) dodawanie m liczb n-cyfrowych?

$a_1^1 a_2^1 \ldots a_n^1$
$a_1^2 a_2^2 \ldots a_n^2$ } $b_1^1 \ldots b_n^1$ $2n-1$ dodawań. Rezultat może mieć $n+1$ cyfr.
pierwsze dodawanie

$a_1^m a_2^m \ldots a_n^m$

$b_0^1 \ldots b_n^1$
$a_1^3 \ldots a_n^3$ } $(n+1)$-cyfrowa liczba, drugie dodawanie
$b_1^2 b_0^2 \ldots b_n^2$ Drugie dodawanie wymaga: $2(n+1)-1$ pojedynczych dodawań, $2n+1$

Trzecie dodawanie liczb $(n+2)$-cyfrowych wymaga $2(n+2)-1$ pojedynczych dodawań, $2n+3$

$b_1^{k-1} \ldots b_n^{k-1}$ } k-te dodawanie

$0 \ldots 0 a_1^k \ldots a_n^k$ } Rezultat k-tego dodawania wymaga: $n+k$ cyfr, k-te dodawanie to dodawanie $n+k-1$-cyfrowej liczby do liczby n-cyfrowej. W k-tym dodawaniu wykonujemy $2(n+k-1)-1$ pojedynczych dodawań $= 2n+2k-3$. Łącznie mamy $m-1$ dodawań liczb kolejno: n-cyfrowej, $n+1$-cyfrowej, ..., $n+(m-2)$ cyfrowej.

Dodawanie pojedynczych cyfr wynosi: $(2n-1)+(2n+1)+(2n+3)+\ldots+(2(n+m-2)-1) = *$. Jest to suma ciągu arytmetycznego, gdzie 1-wyraz: $2n-1$, różnica: 2, l. wyrazów: $m-1$. Zatem:

$* = (m-1)\cdot(2n-1) + 2\cdot\frac{(m-1)(m-2)}{2} = (m-1)(2n-1) + (m-1)(m-2) = (m-1)[2n+m-3]$. Jeżeli $m > n$ to otrzymujemy funk. kwadratową klasy $\Theta(m^2)$. Jeśli $m \leq n$ to otrzymujemy funkcji kwadratową klasy $\Theta(n^2)$.

3. Znajdowanie wartości wielomianu

Dany: $a_0 + a_1 x + \ldots + a_n x^n$; $a_n \neq 0$, x_0

Cel: Obliczyć wartość.

Zmienna pomocnicza: P.

Krok 1: $P = 1$

Krok 2: Dla $i = 1, \ldots, n$ zastąpi P liczbą $P \cdot x_0$, niech $x_i = P \cdot x_0$. Obserwacja: $x_i = x_0$

Krok 3: Niech $S = a_0$

Krok 4: Dla $i = 1, \ldots, n$ zastąpi S przez $S + a_i x_i$

Krok 5: Wypisz S.

4. Dane: taki jak wyżej: $a_0 + a_1 x + \ldots + a_n x^n$, $a_n \neq 0$, x_0

Cel: Taki jak wyżej: znaleźć wartość

Krok 1: $S = a_n$

Krok 2: Dla $i = 1, \ldots, n$ zastąpi S przez $a_{n-i} + S \cdot x_0$

Krok 3: Wypisz S.

Przykład: wielomian $W(x) = -7 + 2x + 4x^2 - 3x^3$, $x_0 = 5$

1. $S = -3$ 2. $i = 1$: $S = a_{n-1} + (\text{stare } S) \cdot 5 = 4 + (-3) \cdot 5 = -11$

3. $i = 2$. $S = (a_n - 2) + (\text{stare } S) \cdot 5 = 2 + (-11) \cdot 5 = -53$

4. $i = 3$: $S = (-7) + (-53) \cdot 5 = -266$ 5. Wypisujemy -266.

Ogólnie: $2 \cdot n$ – złożoność tego algorytmu

Algorytmy sortowania

I Algorytm sortowania bąbelkowego (Bubble sort)

Dane: Lista, której elementami są liczby $L = [a_1, \dots, a_n]$

Cel: $\tilde{L} = [\tilde{a}_1, \dots, \tilde{a}_n]$ takie że wyrazy $a_1, \dots, a_n$ są identyczne z wyrazami $\tilde{a}_1, \dots, \tilde{a}_n$. $\tilde{a}_1 \le \tilde{a}_2 \le \dots \le \tilde{a}_n$.

Krok 1: Jeżeli $n = 1$, to wypisz a_1 i zakończ.

Krok 2: Dla $k = n-1$ do 1, wykonaj:

Krok 2a: Dla $i = 1, \dots, k$, jeśli $a_i > a_{i+1}$, to zamień miejscami a_i z a_{i+1}.

Przykład: $L = [\underset{a_1}{4}, \underset{a_2}{0}, \underset{a_3}{3}, \underset{a_4}{2}]$, $n = 1$, dla $k = 3, 2, 1$, wykonamy krok 2a

•) $k = 3$: (2a): $i = 1, 2, 3$ jeśli $i = 1$, czy $a_i > a_{i+1}$? $[0, 4, 3, 2]$

czyli czy $a_1 > a_2$? TAK. Zatem nowe $a_1 = 0$, nowe $a_2 = 4$.

jeśli $i = 2$, czy $a_i > a_{i+1}$?, czyli czy $a_2 > a_3$? czy $4 > 3$?

TAK. Zatem nowe $a_2 = 3$, nowe $a_3 = 4$ $[0, 3, 4, 2]$

jeśli $i = 3$, czy $a_i > a_{i+1}$?, czy $a_3 > a_4$?, czy $4 > 2$?

TAK. Zatem nowe $a_3 = 2$, nowe $a_4 = 4$ $[0, 3, 2, 4]$.

$[4, 0, 3, 2] \to [0, 4, 3, 2] \to [0, 3, 4, 2] \to [0, 3, 2, 4]$

• $k = 2$ Dla $i = 1, 2$ wykonaj: jeśli $a_i > a_{i+1}$ to zamień.

jeśli $i = 1$; czy $a_1 > a_2$?, czy $0 > 3$? NIE

jeśli $i = 2$, czy $a_2 > a_3$?, czy $4 > 2$? TAK. Zatem nowe $a_2 = 3$,

nowe $a_3 = 2$ $[0, 2, 3, 4]$ $[0, 3, 2, 4] \to [0, 2, 3, 4]$

• $k = 1$ Dla $i = 1$ czy $a_i > a_{i+1}$?, czy $a_1 > a_2$? NIE

Output (wypisujemy): $[0, 2, 3, 4]$

Złożoność obliczeniowa: Ile porównań w zależności od długości listy? Gdy $k = n-1$ to $n-1$ porównań, gdy $k = n-2$, to $n-2$ porównań

Łącznie: $(n-1)+(n-2)+\ldots+1$ porównań $= \frac{(n-1)n}{2}$.
Złożoność obliczeniowa algorytmu sortowania bąbelkowego jest funkcją klasy $\Theta(n^2)$. W skrócie złożoność wynosi $\Theta(n^2)$.

II Sortowanie przez scalenie (Merge-sort).

Dane: $L_1=[a_1,\ldots,a_n]$, $L_2=[b_1,\ldots,b_m]$, L_1, L_2 są posortowane tzn: $a_1 \le a_2 \le \ldots \le a_n$ i $b_1 \le b_2 \le \ldots \le b_m$.

Cel: Utworzenie listy L, posortowanej, zawierającej wszystkie wyrazy $a_1,\ldots,a_n,b_1,\ldots,b_m$.

Krok 1: Jeśli L_1 jest pusta to $L=L_2$ i zakończ. Jeśli L_2 jest pusta to $L=L_1$ i zakończ. W przeciwnym wypadku połóż $L=[\,]$ i zrealizuj Krok 2.

Krok 2: Jeżeli $a_1 \le b_1$ to dołącz a_1 do L. Jeśli reszta listy L_1 jest pusta to do listy L dołącz L_2. W przeciwnym wypadku nowa lista $L_1=[\underset{\text{nowe } a_1}{a_2}, \underset{\text{nowe } a_2}{a_3},\ldots,\underset{\text{nowe } a_{n-1}}{a_n}]$.
Nowe a_i = stare a_{i+1} dla $i=1,\ldots,n-1$. Powtórz Krok 2 w roli n biorąc $n-1$.

Jeżeli $a_1 > b_1$ to dołącz b_1 do L. Jeśli reszta L_2 jest pusta to dopisz do listy L całą listę L_1. W przeciwnym wypadku nowa lista $L_2=[b_2,b_3,\ldots,b_{m-1}]$. Nowe b_i = stare b_{i+1}.
Powtórz Krok 2 dla $i=1,\ldots,m-1$ w roli m biorąc $m-1$.

Przykład: $L_1=[5,8]$, $L_2=[7,8]$, $L=[\,]$,
$a_1 \le b_1$? TAK $\Rightarrow L=[a_1]=[5]$, $L_1=[8]$,
$8 \le 7$? NIE $\Rightarrow L=[5,7]$, $L_1=[8]$, $L_2=[8]$
$8 \le 8$? TAK $L=[5,7,8]$ $L_1=[\,]$, $L_2=[8]$
$L=[5,7,8,8]$

Algorytm sort. bąbelkowego: $[3,7,1,4]$
Scalić $[3,5,8]$ i $[1,7,8]$

Przykład: $L=[3,7,1,4]$, $n=1$, dla $k=3,2,1$ wykonujemy krok
•) $k=3$: $i=1,2,3$ Czy $a_1 > a_2$? NIE. Czy $a_2 > a_3$? TAK. zatem nowe $a_2=1$, nowe $a_3=7$. Czy $a_3 > a_4$? TAK. Zatem nowe $a_3=4$, nowe $a_4=7$ $[3,1,4,7]$
•) $k=2$: $i=1,2$ Czy $a_1 > a_2$? TAK. Zatem nowe $a_1=1$, nowe $a_2=$

Czy $a_2 \geq a_3$? NIE. [1,3,4,7]

$k=1$: $i=1$ Czy $a_1 > a_2$? NIE Output. [1,3,4,7]

Przykład: $L_1=[3,5,8]$, $L_2=[1,7,8]$, $L=[\,]$,

Czy $a_1 \leq b_1$? NIE => $L=[b_1]=[1]$, $L_2=[7,8]$

Czy $3 \leq 7$? TAK => $L=[1,3]$, $L_1=[5,8]$

Czy $5 \leq 7$? TAK => $L=[1,3,5]$, $L_1=[8]$

Czy $8 \leq 7$? NIE => $L=[1,3,5,7]$ $L_2=[8]$

Czy $8 \leq 8$? TAK => $L=[1,3,5,7,8]$ $L_1=[\,]$

L_1 - puste dodajemy elementy z L_2 do L => $[1,3,5,7,8,8]=L$

Zaprojektować M.T. która odejmuje 1 od liczby 01^n0.

δ \ P	0	1	B
q_0	$q_1, 0, P$		
q_1	$q_2, 0, L$	$q_1, 1, P$	
q_2		$q_f, 0, P$	
q_f			

np: $\underset{q_0}{0}1110 \to 0\underset{q_1}{1}110 \to 01\underset{q_2}{1}10 \to$

$\to 0111\underset{q_2}{0} \to 011\underset{q_1}{1}0 \to 01\underset{q_2}{1}10 \to$

$\to 0110\underset{q_f}{0}$ akceptuje

$\Sigma=\{0,1\}$, $Q=\{q_0,q_1,q_2,q_f\}$

$F=\{q_f\}$ $\Gamma=\{0,1,B\}$

Algorytmy i złożoność - ćwiczenie 16.05.2023r.

1. Ile porównań należy wykonać, by wśród n liczb wyznaczyć liczbę największą?

np: 5, 7, 8, 1, 3, $n=5$ Możemy użyć części algorytmu na sortowanie bąbelkowe. 1) 5, 7, 8, 1, 3 ; 2) 5, 7, 8, 1, 3 ; 3) 5, 7, 1, 8, 3 ; 4) 5, 7, 1, 3, 8 i największa to $a_n=a_5=8$

Zatem potrzeba $n-1$ porównań

2. Ile pojedynczych dodawań liczb jednocyfrowych wymaga dodanie dwóch liczb n-cyfrowych?

np: 231 + 783 wtedy $\begin{array}{r} {}^{1}{}^{2}{}^{3}{}^{1} \\ +\ 7\ 8\ 3 \\ \hline 1014 \end{array}$ czyli mamy 4 dodawania

ale co najwyżej 5 dodawań dla dwóch liczb 3-cyfrowych

Dodanie dwóch liczb n-cyfrowych $(a_1 a_2 \dots a_n)$ i $(b_1 b_2 \dots b_n)$ zapisy pozycyjne liczb

Dla liczby 3 cyfrowych $\begin{array}{r} (a_1 a_2 a_3) \\ +\ (b_1 b_2 b_3) \\ \hline \end{array}$

3. 1. dodawanie: a_3+b_3; cyfra przeniesienia c

2. dodawanie: a_2+b_2; 3. dodawanie wyniku $+c$; cyfra ~~p~~ mieszcząca d; 4. dodawanie: a_1+b_1; 5. dodawanie: ~~cy~~

W ogólnym przypadku dwóch liczb n-cyfrowych jest ~~wy~~

wyżej n dodawań + $(n-1)$ dodawań = $2n-1$ dodawań

3. Ile pojedynczych dodawań (liczb 1-cyfrowych) wy~~ma~~ dodanie m liczb n-cyfrowych?

$$\left.\begin{array}{l} a_1a_2\dots a_n \\ b_1b_2\dots b_n \end{array}\right\} 2n-1 \text{ dodawań}$$

$d_0d_1d_2\dots d_n$ ← liczba $n+1$ cyfrowa $\}$ $d_0d_1\dots d_n$

$0\ c_1c_2\dots c_n$ ← liczba $n+1$ cyfrowa $\}$ $0\ c_1\dots c_n$ $\}$ $2n+1$ dodawań

Dodanie trzech liczb n-cyfrowych wymaga $(2n-1)+(2n+1)$

$= 4n$ dodawań

$$\begin{array}{l} a_1\dots a_n \\ b_1\dots b_n \end{array}\Big\} 2n-1 \text{ dodawań}$$

$$\begin{array}{l} x_0\ x_1\dots x_n \\ 0\ c_1\dots c_n \end{array}\Big\} 2n+1 \text{ dodawań}$$

$$\begin{array}{l} y_1y_0\ y_1\dots y_n \\ 0\ 0\ d_1\dots d_n \end{array}\Big\} 2n+3 \text{ dodawań}$$

Łączna liczba dodawań $\leq (2n-1)+(2n+1)+(2n+3)$ (dla 2) (dla 3) (dla ...)

$m-1$ etapów:

etap 1: $2n-1$, etap 2: $(2n-1)+2$

... etap k: $(2n-1)+\underbrace{2+\dots+2}_{k-1}$, etap $m-1$: $(2n-1)+(m-2)\cdot 2$

Jest to suma ciągu arytmetycznego $a_0=2n-1$ o $m-1$ wyrazach i o $r=2$. Suma $=(2n-1)+((2n-1)+2)+\dots+((2n-1)+2(m-2))$

$$=(m-1)(2n-1)+\sum_{i=0}^{m-2}2i=(m-1)(2n-1)+2\cdot\frac{(m-2)(m-3)}{2}=2nm-m-n+m^2-5m+6$$

Dla $m>n$: Suma $\leq 3m^2-6m+6 \to n\leq 3m^2-6m+6 \in \Theta(m^2)$

Podsumowanie: dla $m\geq n$, dodanie m liczb n-cyfrowych wymaga $\Theta(m^2)$ pojedynczych kroków (dodawań liczb 1-cyfrowych).

4. Ile dzieleń należy wykonać, aby przedstawić liczbę N w systemie o podstawie 3?

np: $(153)_{10}=(\dots)_3$ $\quad 153=3\cdot 51+0$, $51=3\cdot 17+0$,

$17=3\cdot 5+2$, $5=3\cdot 1+2$, $1=3\cdot 0+1$ $\quad (153)_{10}=(12200)_3$

$=1\cdot 3^4+2\cdot 3^3+2\cdot 3^2+0\cdot 3^1+0\cdot 3^0 < 3^5$

Liczba dzieleń = liczba cyfr w przedstawieniu N w systemie o podst. 3

Jeśli nawet liczba $2\cdot 3^4+2\cdot 3^3+\dots+2\cdot 3^0$ = to zachodzi:

$$=2\cdot\sum_{i=0}^{4}3^i=2\cdot\frac{1-q^{n+1}}{1-q}=2\cdot\frac{(1-3^5)}{-2}=3^5-1<3^5$$

$153<3^k \to$ szukamy k $\Rightarrow \log_3 153<k \Rightarrow \log_3 N<k$

$\lceil \log_3 N\rceil+1$

$O(\log_3 N + 1)$ $\quad \log_3 N + 1 \leq \log_3 N + \log_3 N = 2\log_3 N \Rightarrow \Theta(\log_3 N)$

Liczba dzieleń N przez 3 = liczba cyfr (k) w otrzymanej reprezentacji $(c_1 \ldots c_k)_3 = N$. Wtedy $N = c_1 \cdot 3^{k-1} + c_2 \cdot 3^{k-2} + \ldots + c_k \cdot 3^0 < 3^k$, zatem $\log_3 N < k$ więc liczba dzieleń jest klasy $O(\log_3 N)$.

5. Zastosować sortowanie bąbelkowe do listy [2, 3, 7, 1].

Start: [2, 3, 7, 1]

1) krok $i=3$: $[\underset{a_1}{2}, \underset{a_2}{3}, \underset{a_3}{7}, \underset{a_4}{1}] \xrightarrow[\text{NIE}]{\text{Czy } a_1 > a_2?} [\underset{a_1}{2}, \underset{a_2}{3}, \underset{a_3}{7}, \underset{a_4}{1}] \xrightarrow[\text{NIE}]{\text{Czy } a_2 > a_3?} [\underset{a_1}{2}, \underset{a_2}{3}, \underset{a_3}{7}, \underset{a_4}{1}] \xrightarrow[\text{TAK}]{\text{Czy } a_3 > a_4?} [\underset{a_1}{2}, \underset{a_2}{3}, \underset{a_3}{1}, \underset{a_4}{7}]$

2) krok $i=2$: $[\underset{a_1}{2}, \underset{a_2}{3}, \underset{a_3}{1}, \underset{a_4}{7}] \xrightarrow[\text{NIE}]{\text{Czy } a_1 > a_2?} [2, 3, 1, 7] \xrightarrow[\text{TAK}]{\text{Czy } a_2 > a_3?} [\underset{a_1}{2}, \underset{a_2}{1}, \underset{a_3}{3}, \underset{a_4}{7}]$

3) krok $i=1$: $[\underset{a_1}{2}, \underset{a_2}{1}, \underset{a_3}{3}, \underset{a_4}{7}] \xrightarrow[\text{TAK}]{\text{Czy } a_1 > a_2?} [\underset{a_1}{1}, \underset{a_2}{2}, \underset{a_3}{3}, \underset{a_4}{7}]$ Koniec: [1, 2, 3, 7]

6. Zastosować sortowanie bąbelkowe do listy [7, 5, 6, 2, 1, 3]

Start: [7, 5, 6, 2, 1, 3] krok 1) $i=5$: $[7, 5, 6, 2, 1, 3] \xrightarrow[\text{TAK}]{\text{Czy } a_1 > a_2?}$ $[5, 7, 6, 2, 1, 3] \xrightarrow[\text{TAK}]{\text{Czy } 7 > 6?} [5, 6, 7, 2, 1, 3] \xrightarrow[\text{TAK}]{\text{Czy } 7 > 2?} [5, 6, 2, 7, 1, 3] \xrightarrow[\text{TAK}]{\text{Czy } 7 > 1?}$ $[5, 6, 2, 1, 7, 3] \xrightarrow[\text{TAK}]{\text{Czy } 7 > 3?} [5, 6, 2, 1, 3, 7]$

krok 2) $i=4$: $[5, 6, 2, 1, 3, 7] \xrightarrow[\text{NIE}]{\text{Czy } 5 > 6?} [5, 6, 2, 1, 3, 7] \xrightarrow[\text{TAK}]{\text{Czy } 6 > 2?} [5, 2, 6, 1, 3, 7]$ $\xrightarrow[\text{TAK}]{\text{Czy } 6 > 1?} [5, 2, 1, 6, 3, 7] \xrightarrow[\text{TAK}]{\text{Czy } 6 > 3?} [5, 2, 1, 3, 6, 7]$

krok 3) $i=3$: $[5, 2, 1, 3, 6, 7] \xrightarrow[\text{TAK}]{\text{Czy } 5 > 2?} [2, 5, 1, 3, 6, 7] \xrightarrow[\text{TAK}]{\text{Czy } 5 > 1?} [2, 1, 5, 3, 6, 7]$ $\xrightarrow[\text{TAK}]{\text{Czy } 5 > 3?} [2, 1, 3, 5, 6, 7]$

krok 4) $i=2$: $[2, 1, 3, 5, 6, 7] \xrightarrow[\text{TAK}]{\text{Czy } 2 > 1?} [1, 2, 3, 5, 6, 7] \xrightarrow[\text{NIE}]{\text{Czy } 2 > 3?} [1, 2, 3, 5, 6, 7]$

krok 5) $i=1$: $[1, 2, 3, 5, 6, 7] \xrightarrow[\text{NIE}]{\text{Czy } 1 > 2?} [1, 2, 3, 5, 6, 7]$

Wynik: [1, 2, 3, 5, 6, 7]

7. Jaka jest złożoność obliczeniowa (ile jest porównań) algorytmu sortowania bąbelkowego.

W naszym przykładzie 6. mieliśmy $5+4+3+2+1 = 15$ porównań

W ogólnej sytuacji dla listy o n elementach mamy

$$(n-1) + (n-2) + \ldots + 1 = \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} \in \Theta(n^2)$$

Algorytmy i złożoność – wykład 14.05.2023r.

$L_1 = [3, 5]$, $L_2 = [1, 7]$, $L = []$ Scalić

I iteracja

Czy $3 \leq 1$ NIE $\rightarrow L = [1]$, $L_2 = [7]$, Czy $3 \leq 7$ TAK $\rightarrow L = [1, 3]$,

III iteracja

$L_1 = [5]$, Czy $5 \leq 7$ TAK $\rightarrow L = [1, 3, 5]$, $L_1 = []$, L_2 – puste $\rightarrow$

$\rightarrow L = [1, 3, 5, 7]$ $L_1 = []$, $L_2 = []$

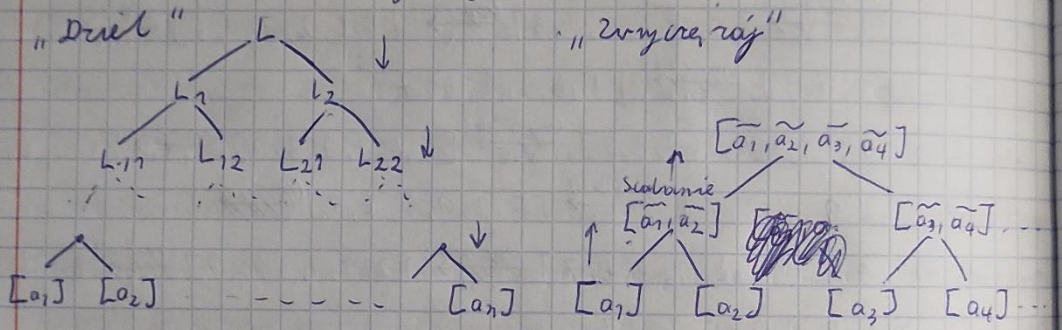
Ile czasu (kroków) zajmuje algorytm sortowania przez sc

$L_1 = l = [a_1, \dots, a_l]$, $L_2 = k = [b_1, \dots, b_k]$

W każdej iteracji jest 1 porównanie np: $\underbrace{a_1 \le b_1}_{c_1 = a_1}$ potem $\underbrace{b_1 \le a_2}_{c_2 = b_1}$ i potem $a_2 \le b_2$ i tak dalej. Zatem mam $(l+k)-1$ porównań. Jeśli $l+k = n$ to liczba kroków wynosi $n-1$. Złożoność obliczeniowa algorytmu scalania jest klasy $\Theta(n)$. (jest liniowa).

Sortowanie przez scalanie „Merge - Sort".

Dana jest lista $L = [a_1, \dots, a_n]$. Cel: Wyznaczenie $\tilde{L} = [\tilde{a}_1, \dots, \tilde{a}_n]$ posortowanej, takiej że wyrazy $\tilde{L}$ są takie same jak wyrazy listy L. Nieformalnie



Opis algorytmu: Dane: $L = [a_1, \dots, a_n]$

Krok 1: Dla $i = 1, \dots, n$ utwórz $L_i = [a_i]$

Krok 2: Jeśli $n = 1$ to zakończ i wypisz L_1. W przeciwnym razie Jeśli $\exists l \in \mathbb{Z}$ $n = 2l$, to dla $j = 1, \dots, l$ scal L_{2j-1} i L_{2j}. Powstałą listę nazwij L_j. Połóż $n = l$ <„nowe" n to l>. Powtórz krok 2. Jeśli $\exists l \in \mathbb{Z}$ $n = 2l+1$ to dla $j = 1, \dots, l$ scal L_{2j-1} i L_{2j}. Powstałą listę nazwij L_j. Połóż $L_{l+1} = L_n$. Powtórz krok 2. Połóż $n = l$.

Przykład: $L = [2, 9, 1, 4, 6, 5, 3]$

Krok 1: $n = 7$ $L_1 = [\overset{a_1}{2}]$, $L_2 = [9]$, $L_3 = [\overset{a_7}{1}]$, $L_4 = [4]$, $L_5 = [6]$, $L_6 = [5]$, $L_7 = [3]$

Krok 2: Iteracja 1. $n = 2 \cdot 3 + 1$, $l = 3$ Dla $j = 1, 2, 3$:

- $j = 1$ $L_{2j-1} = [2]$, $L_{2j} = [9]$ Scalamy: $L_1 = [2, 9]$
- $j = 2$ $L_{2j-1} = [1]$, $L_{2j} = [4]$ Scalamy $L_2 = [1, 4]$

•) $j=3$ $L_5=[6]$, $L_6=[5]$ Scalamy: $L_3=[5,6]$

•) $L_{11}=L_4=[3]$ (oraz $L_1=[2,9]$, $L_2=[7,4]$, $L_3=[5,6]$)

Iteracja 2. $n=4=2\cdot 2$, $b=2$ Dla $j=1,2$:

•) $j=1$ scalamy L_1 i L_2; $L=[\,]$ Czy $2\le 7$? NIE $L=[7]$

$L_1=[2,9]$, $L_2=[4]$, Czy $2\le 4$? TAK $L=[7,2]$, $L_1=[9]$, $L_2=[4]$

Czy $9\le 4$? NIE $L=[7,2,4]$, $L_1=[9]$, $L_2=[\,]$ $\Rightarrow L=[7,2,4,9]=L_1$.

•) $j=2$ Scalamy L_3 i L_4; $L=[\,]$ Czy $5\le 3$? NIE $L=[3]$,

$L_3=[5,6]$, $L_4=[\,]$ $\Rightarrow L=[3,5,6]=L_2$

Iteracja 3. $n=2=2\cdot 1$, $b=1$ $L_1=[7,2,4,9]$, $L_2=[3,5,6]$

•) Dla $j=1$: Czy $7\le 3$? TAK $L=[7]$, Czy $2\le 3$? TAK $L=[7,2]$

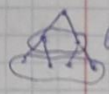
Czy $4\le 3$? NIE $L=[7,2,3]$, $L_1=[4,9]$, $L_2=[5,6]$; Czy $4\le 5$? TAK

$L=[7,2,3,4]$, Czy $9\le 5$? NIE $L=[7,2,3,4,5]$, $L_1=[9]$, $L_2=[6]$

Czy $9\le 6$? NIE $L=[7,2,3,4,5,6]$ $L_2=[\,]$, $L_1=[9]$ $\Rightarrow L=[7,2,3,4,5,6,9]$

$n=1$ zatem kończę algorytm Odp: $L=[7,2,3,4,5,6,9]$.

Złożoność algorytmu sortowania przez scalenie.



n-list zatem złożoność jest klasy $\Theta(n\cdot\lg n)$

Inny opis algorytmu sortowania przez scalenie

Dana jest lista L o długości n.

1. Podzielimy L na pół (jeśli $n=2l$, to $L_1=[a_1,\dots,a_l]$, $L_2=[a_{l+1},\dots,a_n]$. Jeśli $n=2l+1$ to $L_1=[a_1,\dots,a_{l+1}]$, $L_2=[a_{l+2},\dots,a_n]$)

Powstają dwie listy: L_1 i L_2 o długościach $\lceil\frac{n}{2}\rceil$, $\lfloor\frac{n}{2}\rfloor$

2. Jeśli L_1 ma długość 1 i L_2 ma długość 1 to scal L_1 i L_2 i zakończ.

Złożoność (inna analiza): a) Znaleźć środek listy L: c_1,

b) Sortuję L_1 i sortuję L_2 c) Sumuję L_1 i L_2 – czas $c\cdot n$

Sumarycznie: $\overbrace{T(n)}^{T(n)}=c_1+\overbrace{T(\frac{n}{2})}^{T(\frac{n}{2})}+T(\frac{n}{2})+c\cdot n=f(n)+2\cdot T(\frac{n}{2})$, gdzie

$T(n)=2\cdot T(\frac{n}{2})+f(n)$

$f(n)$ jest funkcją liniową klasy $\Theta(n)$

Twierdzenie o rekurencji uniwersalnej: $a=2$, $b=2$, $\log_2 2=1$

$n^{\log_b a}=n^1=n$, $f(n)\in\Theta(n)$, $f(n)\in\Theta(n^{\log_b a})$ z tw. o rekurencji uniwersalnej $T(n)\in\Theta(n\lg n)$.

$T(n)$ = czas dzielenia + czas wykonania algorytmu na zmienionych danych × ilość tych zmniejszonych danych + czas scalenia

$a=b=2$

$T(n) = D(n) + a \cdot T(\frac{n}{b}) + f(n)$ ..... $n^{\log_b a} = n$

$f(n) \in \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n \lg n)$

Sortowanie przez wstawianie i niezmienniki pętli.

Dane: $L = [a_1, \dots, a_n]$ Cel: posortowanie listy L

Dla $j = 2, \dots, n$, $k = a_j$ Połóż $i = j-1$ i dopóki $i > 0$ i $a_i > k$

Połóż $a_{i+1} = a_i$ nowe $i = i-1$ $a_{i+1} = k$ warunek końca pętli

Przykład: $L = [5, 3, 6]$, $n = 3$,

Dla $j=2$: $i=1$ Czy $i > 0$ i $a_1 > a_2$? $\Rightarrow a_2 = 5, i=0$

TAK $5 > 3$ TAK

$[3, 5, 6]$ Czy $i > 0$ i $a_0 > a_2$?

NIE

Dla $j=3$: $i=2$ Czy $i > 0$ i $a_2 > a_3$? $\Rightarrow$ NIE, to stawia

TAK $5 > 6$ NIE

$i = i-1 = 1$ Czy $i > 0$ i $a_1 \geq a_3$? $\Rightarrow$ NIE

TAK $3 > 6$ NIE

$L = [3, 5, 6]$

Niezmiennik pętli - jest to pewien warunek W czyli zdanie, L₁ = [taki że, jeśli przed iteracją pętli W jest prawdziwe, to przed następną iteracją pętli W też jest prawdziwe. Krok 1 Stwierdzenie: Niech W będzie pewnym zdaniem. Jeżeli W jest prawdziwy przed pierwszą iteracją pętli oraz W jest niezmiennikiem pętli to po zakończeniu realizacji wszystkich iteracji pętli, W też jest prawdziwe. Krok 2

Algorytmy i złożoność - ćwiczenia 23.05.2023r.

1. Opisać algorytm, który dla zadanej liczby n, przedstawia n w postaci jako: $n = a_k \cdot k! + \dots + a_2 \cdot 2! + a_1$, gdzie $a_1 \in \{1, 0\}$, $a_2 \in \{2, 1, 0\}$, $a_i \in \{0, 1, \dots, i\}$. Dodatkowo obliczyć złożoność tego algorytmu.

np: $n = 29 = 1 \cdot 4! + 0 \cdot 3! + 2 \cdot 2! + 1$

krok 1): $29 : 2 = 14$, $r.1$ $29 = 2 \cdot 14 + 1$

$14 : 3 = 4$, $r.2$ $29 = 2 \cdot (3 \cdot 4 + 2) + 1 = 2 \cdot 3 \cdot 4 + 4 + 1$

$n! \leq 29 < (n+1)!$

$\Downarrow$

$m = 29 - n!$

$\vdots$

$2^{k-1} \leq 29 < 2^n$

$1:4 = 1$, r. 0 $\quad 1:5 = 0$, r. 1 $\quad$ Te reszty odpowiadają współczynnikom z równania: $29 = 1\cdot 4! + 0\cdot 3! + 1\cdot 2! + 1$

Opis algorytmu: dany dana liczba n. Szukane jest $k, a_1, \dots, a_k$. Szukamy największego k, takie że $k! > 29(n)$. Dla $i = 2, \dots, k$ wykonujemy działanie $a = q\cdot i + r$, gdzie $0 \le r < i$, $a = n$. Kładziemy $a_{i-1} = r$ i $a = q$. Wypisz $a_1, \dots, a_{k-1}$.

Niech $k = 0$. Dopóki $k! \le n$, $k = k+1$. Po pętli wypisz k.

Algorytm 2: $a = n$, $i = 2$ – krok zerowy

Krok 1: Dopóki $a > 0$ wykonaj: znajdź q, r: $a = q\cdot i + r$, $0 \le r \le i$, $a_{i-1} = r$, $a = q$, $i = i+1$. Wróć do 1.

Krok 2: Wypisz: $k = i$, $a_1, \dots, a_{k-1}$

Zad 2. Zastosować algorytm scalania do scalenia list (w kolejności alfabetycznej): [anna, ewa, ola]; [dawid, tomek, wojtek]. Możemy uprościć listy: [a, e, n], [d, o, w]

Niech $L_1 = [\underset{a_1}{a}, \underset{a_2}{e}, \underset{a_3}{n}]$, $L_2 = [\underset{b_1}{d}, \underset{b_2}{o}, \underset{b_3}{w}]$, $L = [\,]$

Krok 0: $L_1 \ne [\,]$, $L \ne [\,]$, $L = [\,]$

Krok 1: Czy $a_1 \le b_1$? Czy $a \le d$? TAK $L = [a]$, $L_1 = [\underset{a_1}{e}, \underset{a_2}{n}]$

Czy $a_1 \le b_1$? Czy $e \le d$? NIE $L = [a, d]$ $L_1 = [e, n]$, $L_2 = [o, w]$

Czy $e \le o$? TAK $L = [a, d, e]$, $L_1 = [n]$, $L_2 = [o, w]$

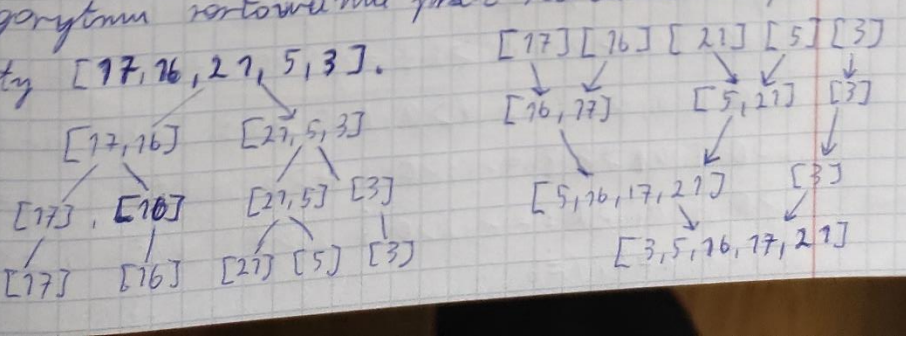
Czy $n \le o$? NIE $L = [a, d, e, o]$, $L_1 = [n]$, $L_2 = [w]$

~~$L_1 = [\,]$ – pusta lista => $L = [a, d, e, o, w]$~~

Czy $n \le w$? TAK $L = [a, d, e, o, n]$, $L_1 = [\,]$, $L_2 = [w]$

$L_1 = [\,] \Rightarrow L = [a, d, e, o, n, w]$

3. Użyć algorytmu sortowania przez scalanie do posortowania listy [17, 16, 21, 5, 3].



1. Alg. Euklidesa – przykłady omówić i zastanowić się nad niemożliwością pętli

2. Opisać alg. który dla zadanej liczby z znajdzie wszystkie liczby mniejsze $< n$. Jaka jest złożoność?

Algorytmy i złożoność - wykład 31.05.2023 r.

Zasada algorytmu sortowania przez wstawianie

Dane: $L = [a_1, \dots, a_n]$, Cel: uzyskać $\tilde{L}$, która jest posortowaną listą L.

Dla $j = 2, \dots, n$ połóż $k = a_j$, $i = j-1$.

Dopóki: $i > 0$ i $a_i > k$, połóż $a_{i+1} = a_i$, $i = i-1$.

Połóż $a_{i+1} = k$.

Przykład: $L = [5, 2, 4, 7, 1, 3, 2, 6]$, $n = 8$

Pętla „dla": 1) $j = 2$: $k = a_j = a_2 = 2$, $i = 1$

Dopóki $i > 0$ i $a_i > k$? $\Rightarrow 1 > 0$ i $a_1 > 2$? $\Rightarrow 1 > 0$ i $5 > 2$?

TAK, czyli $a_2 = a_1 = 5$, nowe $i = 0$, czy $i > 0$ i ...? NIE

$a_{i+1} = k \Rightarrow a_1 = 2$, $\tilde{L} = [2, 5, 4, 7, 1, 3, 2, 6]$

2) $j = 3$, $k = a_j = a_3 = 4$, $i = 2$

Dopóki: $i > 0$ i $a_i > k$? $\Rightarrow i > 0$ i $a_2 > 4$? $\Rightarrow i > 0$ i $5 > 4$?

TAK, $a_{i+1} = a_i \Rightarrow a_3 = 5$, nowe $i = 1$.

$i > 0$ i $a_i > k$? $\Rightarrow i > 0$ i $2 > 4$, NIE, $a_{i+1} = k \Rightarrow a_2 = 4$

$\tilde{L} = [2, 4, 5, 7, 1, 3, 2, 6]$ 3) $j = 4$: $k = a_j = a_4 = 7$, $i = 3$

Dopóki: $i > 0$ i $a_3 > 7$? $\Rightarrow i > 0$ i $5 > 7$? NIE, $a_4 = 7$

4) $j = 5$, $k = 1$, $i = 4$ Dopóki: $i > 0$ i $a_4 > k$? $\Rightarrow 4 > 0$, $7 > 1$?

TAK, $a_5 = 7$, nowe $i = 3$, $i > 0$ i $a_3 > k$? $\Rightarrow 3 > 0$, $5 > 1$?

TAK, $a_4 = 5$, nowe $i = 2$, $i > 0$ i $a_2 > k$? $\Rightarrow 2 > 0$ i $4 > 1$?

TAK, $a_3 = a_2 = 4$, nowe $i = 1$, $i > 0$ i $a_1 > k$? $\Rightarrow 1 > 0$ i $2 > 1$?

TAK, $a_2 = a_1 = 2$, nowe $i = 0$, $i > 0$ i ...? NIE

$a_1 = k$, $a_1 = 1$, $\tilde{L} = [1, 2, 4, 5, 7, 3, 2, 6]$

5) $j = 6$, $k = a_j = a_6 = 3$, $i = 5$

Dopóki: $i > 0$ i $a_i > k$? $\Rightarrow 5 > 0$ i $7 > 3$? TAK, $a_6 = 7$, nowe $i = 4$

$i > 0$ i $a_i > k$? $\Rightarrow 4 > 0$ i $5 > 3$? TAK $a_5 = 5$, nowe $i = 3$

$i > 0$ i $a_i > k$? $\Rightarrow 3 > 0$ i $4 > 3$? TAK $a_4 = 4$, nowe $i = 2$

$i > 0$ i $a_i > k$? $\Rightarrow 2 > 0$ i $2 > 3$? NIE $a_3 = 3$, $\tilde{L} = [1, 2, 3, 4, 5, 7, 2, 6]$

6) $j = 7$, $k = 2$, $i = 6$

Próba: $i>0$ i $a_i>k$? => $6>0$ i $7>2$? TAK, $a_7=7$, nowe $i=5$

$i>0$ i $a_i>k$? => $5>0$ i $5>2$? TAK, $a_6=5$, nowe $i=4$

$i>0$ i $a_i>k$? => $4>0$ i $4>2$? TAK, $a_5=4$, nowe $i=3$

$i>0$ i $a_i>k$? => $3>0$ i $3>2$? TAK, $a_4=3$, nowe $i=2$

$i>0$ i $a_i>k$? => $2>0$ i $2>2$? NIE, $a_3=2$, $\tilde{L}=[1,2,2,3,4,5,7,6]$

7) $j=8$, $k=6$, $i=7$

Próba: $i>0$ i $a_i>k$? => $7>0$ i $7>6$? TAK, $a_8=7$, nowe $i=6$

$i>0$ i $a_i>k$? => $6>0$ i $5>6$? NIE, $a_7=6$, $\tilde{L}=[1,2,2,3,4,5,6,7]$

KONIEC

Złożoność algorytmu sortowania przez wstawianie

	KOSZT	
Dla $j=2,\dots,n$		$n-1$ razy
$k=a_j$	c_1	$n-1$
$i=j-1$	c_1	$\sum_{j=2}^{n} j = \frac{n(n+1)}{2}$
Próba: $i>0$ i $a_i>k$	$2c_2$	$\sum_{j=2}^{n} j = \frac{n(n+1)}{2}-1$
$a_{i+1}=a_i$	c_1	$\frac{n(n+1)}{2}-1$
$i=i-1$		$\sum_{j=2}^{n}(j-1) = \sum_{j=1}^{n-1} j = \frac{n(n-1)}{2}$
$a_{i+1}=k$	c_1	$n-1$ razy

Suma: $T(n) = 3c_1(n-1) + 2c_2\left(\frac{n(n+1)}{2}-1\right) + c_1\cdot\frac{n(n+1)}{2} + c_1\left(\frac{n(n+1)}{2}-1\right) + c_1\cdot\frac{n(n-1)}{2} = \langle c_1, c_2 - \text{stałe}\rangle \in \Theta(n^2)$

Złożoność obliczeniowa algorytmu sortowania przez wstawianie jest klasy $\Theta(n^2)$.

Poprawność algorytmu sortowania przez wstawianie.

$W(j)$ - zdanie prawdziwe: 1) na początku

2) każda iteracja pętli „dla" zachowuje prawdziwość W.

Zatem „W" jest zdaniem prawdziwym po wyjściu z pętli „dla". Prawdziwość W po zakończeniu algorytmu $\equiv$ algorytm jest poprawny.

W: Po wykonaniu „dla" dla $j\in\{2,3,\dots,n\}$, j pierwszych wyrazów listy zostało już posortowanych. $\wedge$

$\wedge$ Przed wykonaniem „dla", dla $j=2,3,\dots,n$, $j-1$ pierwszych wyrazów jest posortowanych.

$j=2$: Przed wykonaniem „dla" dla $j=2$, $[a_1]$ jest posortowane. Po wykonaniu „dla" dla $j=2$:

Było przed $[a_1, a_2]$, jest $[\tilde{a}_1, \tilde{a}_2]$

$$(a_1 = \tilde{a}_1, a_2 = \tilde{a}_2) \text{ lub } (a_1 = \tilde{a}_2, a_2 = \tilde{a}_1)$$

$a_2 = k$, $a_1 > k$? NIE $a_1 \leq k \leq a_2$ | $a_1 > a_2$, $\tilde{a}_2 > \tilde{a}_1$ TAK

Jeżeli przed wykonaniem pętli „dla" dla j $[\tilde{a}_1, \dots, \tilde{a}_{j-1}]$ są posortowane i identyczne z $[a_1, \dots, a_{j-1}]$, to po wykonaniu pętli „dla" dla j, $[\tilde{\tilde{a}}_1, \dots, \tilde{\tilde{a}}_{j-1}, \tilde{\tilde{a}}_j]$ ← posortowana i identyczna z $[a_1, \dots, a_j]$ z oryginalnej list.

Wystarczy wiedzieć, że $\tilde{\tilde{a}}_1, \dots, \tilde{\tilde{a}}_{j-1}, \tilde{\tilde{a}}_j$ to $\tilde{a}_1, \dots, \tilde{a}_{j-1}, a_j$ tyle że posortowane.

$k = a_j$. Pętla „dopóki": porównuje a_j $(=k)$ kolejno $\tilde{a}_{j-1}, \dots, \tilde{a}_i$, tak długo aż $\tilde{a}_i \leq k$. Wtedy wstawiamy k na miejsce $i+1$ – nowe $\tilde{a}_{i+1}$, k.

Nie rusza $\tilde{a}_1, \dots, \tilde{a}_i$, natomiast $\tilde{a}_{i+1}, \dots, \tilde{a}_{j-1}$ przenosi na miejsca $i+2, \dots, j$. (o 1 miejsce w prawo).

Powstaje $\tilde{a}_1, \dots, \tilde{a}_i, k, \tilde{a}_{i+1}, \dots, \tilde{a}_{j-1}$ i jest to właśnie posortowane $[\tilde{a}_1, \dots, \tilde{a}_{j-1}, k]$, czyli $[\tilde{a}_1, \dots, \tilde{a}_{j-1}, a_j]$.

W przypadku, gdy nie ma takiego i, że $k > a_i$, to ostatnie $i=0$. Wszystkie $\tilde{a}_1, \dots, \tilde{a}_{j-1}$, wtedy przesuwamy te o 1 miejsce w prawo, a na końcu $a_{i+1} = k$, czyli k jest wstawiane na miejsce a_1. Otrzymujemy $[a_j, \tilde{a}_1, \dots, \tilde{a}_{j-1}]$, czyli posortowana listę $[\tilde{a}_1, \dots, \tilde{a}_{j-1}, a_j]$.

Na końcu, po wyjściu z pętli dla $j=n$ na mocy „W" mamy: $\tilde{L} = [\tilde{a}_1, \dots, \tilde{a}_n]$ jest posortowaną listą $[a_1, \dots, a_n]$.

Algorytm jest poprawny.

$m = q \cdot n + r$ (dzieląc m przez n otrzymujemy iloraz q i resztę r) $\Rightarrow NWD(m, n) = NWD(n, m)$

Algorytmy i złożoność – ćwiczenia 06.06.2023 r.

1. Przeanalizować złożoność obliczeniową dzielenia pisemnego liczby n-cyfrowej a przez liczbę 1-cyfrową b.

Za pojedynczy krok obliczeniowy przyjmujemy:

- podzielenie jedno- lub dwu-cyfrowej liczby przez liczbę jednocyfrową, przy założeniu że rezultat jest liczbą 1-cyfrową.
- dodanie, odejęcie, pomnożenie przez siebie dwóch liczb jednocyfrowych.

Przykład: $a = 999$, $b = 5$

$$\begin{array}{r} 199 \\ 999 : 5 \\ -5 \\ \hline 49 \\ -45 \\ \hline 49 \\ -45 \\ \hline 4 \end{array}$$

1. $9 : 5 = 1$ r. 4
2. $4 \cdot 10 = 40$
3. $40 + 9 = 49$
4. $49 : 5 = 9$ r. 4
5. $4 \cdot 10 = 40$
6. $40 + 9 = 49$
7. $49 : 5 = 9$ r. 4

Niech $a = (a_1 a_2 \ldots a_n)_{10}$, $b = b$

I próba

Krok 1: Dzielimy a_1 przez b: $a_1 : b = c_1$

Krok 2: Mnożymy c_1 i b: $c_1 \cdot b$

Krok 3: Odejęcie od a_1 iloczynu $c_1 \cdot b$, daje r_1

Krok 1 + Krok 2 + Krok 3 = I etap

Etapów jest n, czyli otrzymujemy $T(n)$, gdzie n jest liczbą cyfrą $T(n) \leq 3n$.

2. Przeszukiwanie liniowe

Dana jest lista $L = [a_1, \ldots, a_n]$ oraz element x.

Cel: Czy (i gdzie) x jest na liście L?

Krok 1: Dla $i = 1, \ldots, n$, jeśli $x = a_i$, to wypisz $x = a_i$ i zakończ.

Krok 2: Wypisz „NIE" i zakończ.

BFS

przeszukiwanie grafów wszerz (breadth first search)

3. Przeszukiwanie binarne

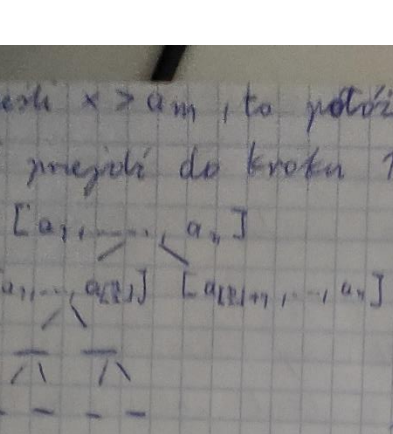
Dana jest uporządkowana lista $L = [a_1, \ldots, a_n]$ oraz element x.

Cel: Czy (i gdzie) x jest na liście L?

Krok 1: Jeśli $n = 1$, to jeśli $x = a_1$ to wypisz $x = a_1$ i zakończ. Jeśli $x \neq a_1$ to wypisz „NIE" i zakończ. Jeśli $x \leq a_m$ to połóż: $L = [a_1, \ldots, a_m]$, $m = \lfloor \frac{n}{2} \rfloor$.

Krok 2: Niech $m = \lfloor \frac{n}{2} \rfloor$. Jeśli $x \leq a_m$ to połóż: $L = [a_1, \ldots, a_m]$, nowe $n = m$ i przejdź z powrotem do kroku 1 z nową listą L i nowym n.

jeśli $x > a_m$, to połóż: $L = [a_{m+1}, \dots, a_n]$, nowe $n = n - m$ i przejdź do kroku 1 (z nową L i nowym n).



wysokość drzewa = $\lg n$

$\Downarrow$

złożoność: $\Theta(\lg n)$

4. Znaleźć liczbę 2 na liście $[1, 2, 3, 4, 5, 6, 7, 8, 9]$.

a) przeszukiwanie liniowe potrzebuje 2 porównań.

b) przeszukiwanie binarne: $[1, 2, 3, 4, 5, 6, 7, 8, 9]$, $n = 9$, $x = 2$

Krok 1: $m = \lfloor \frac{n}{2} \rfloor = \lfloor \frac{9}{2} \rfloor = \lfloor 4{,}5 \rfloor = 4$ $\quad a_4 = 4 \overset{?}{\geq} 2$ TAK

nowe $L = [1, 2, 3, 4]$, nowe $n = 4$

Krok 2: $m = \lfloor \frac{n}{2} \rfloor = \lfloor 2 \rfloor = 2$ $\quad a_2 = 2 \overset{?}{\geq} 2$ TAK

nowe $L = [1, 2]$, nowe $n = 2$

Krok 3: $m = \lfloor \frac{2}{2} \rfloor = 1$ $\quad a_1 = 1 \overset{?}{\geq} 2$ NIE $\Rightarrow$ $\overset{x}{2} > \overset{a_m}{1}$

nowe $L = [2]$, nowe $n = 2 - 1 = 1$

Krok 4: $L = [2]$, nowe $n = 1$, jeśli $x = a_1$ to wypisz x i zakończ. Czy $2 = a_1 = 2$? TAK, $2 = a_1$ i kończymy

5. Na liście $L = [1, 2, 3, 4, 5, 6, 7, 8, 9]$ i znaleźć 8.

a) przeszukiwanie liniowe potrzebuje 8 porównań.

b) przeszukiwanie binarne: $L = [1, 2, 3, 4, 5, 6, 7, 8, 9]$, $n = 9$, $x = 8$

Krok 1: $m = \lfloor \frac{9}{2} \rfloor = 4$ $\quad a_4 = 4 \geq 8$? NIE $\Rightarrow$ $8 > 4$

nowe $L = [5, 6, 7, 8, 9]$, nowe $n = 5$

Krok 2: $m = \lfloor \frac{5}{2} \rfloor = 2$, $a_2 = 6 \geq 8$? NIE $\Rightarrow$ $8 > 6$

nowe $L = [7, 8, 9]$, nowe $n = 3$

Krok 3: $m = \lfloor \frac{3}{2} \rfloor = 1$ $\quad a_1 = 7 \geq 8$? NIE $\Rightarrow$ $8 > 7$

nowe $L = [8, 9]$, nowe $n = 2$

Krok 4: $m = \lfloor \frac{2}{2} \rfloor = 1$, $a_1 = 8 \geq 8$? TAK

nowe $L = [8]$, nowe $n = 1$

Krok 5: $L = [8]$, nowe $n = 1$ Czy $8 = a_1 = 8$? TAK $8 = a_1$

KONIEC.

1. Opisać algorytm przeszukiwania ternarnego
2. W przypadku listy nieposortowanej porównać złożoność prz. liniowego z merge-sort + prz. binarne.

Algorytmy i złożoność - ~~[illegible]~~ wykład 07.06.2023r.

Algorytm przeszukiwania liniowego

Dane: $L=[a_1,\dots,a_n]$; $|L|=n$ ← długość listy; x

Oznaczenia: dla $i=1,\dots,n$: $L[i]=a_i$, $L[i,\dots,j]=\begin{cases} L[i] & \text{gdy } i=j \\ [a_i,\dots,a_j] & i\neq j \end{cases}$

1. Dla $i=1,\dots,n$, jeżeli $x=L[i]$, to wypisz „TAK" i zakończ
2. Wypisz „NIE" i zakończ.

Złożoność: $T(n)=n$, $\Theta(n)$

Algorytm przeszukiwania binarnego

Dana lista $L=[a_1,\dots,a_n]$ - posortowana; $|L|=n$; x

Oznaczenia jak wyżej.

1. Jeżeli $n=1$ to jeśli $x=L[1]$, to wypisz „TAK" i zakończ. W przeciwnym wypadku wypisz „NIE" i zakończ.
2. Niech $m=\lfloor \frac{n}{2} \rfloor$. Jeśli $x\leq a_m$, to połóż $L=L[1,\dots,m]$, $n=m$ i wróć do kroku 1. W przeciwnym wypadku połóż $L=L[m+1,\dots,n]$, $m=n-m$ i wróć do kroku 1.

Złożoność obliczeniowa: $\Theta(\lg n)$.

Grafy: $G=\langle V,E\rangle$; $V\neq\emptyset$, $E\subseteq\{\{v,w\}: v,w\in V \wedge v\neq w\}$

$\{v,w\}=\{w,v\}$, krawędź: $\{v,w\}$, V jest skończony

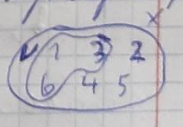
v,w - końce krawędzi

Macierz sąsiedztwa grafu.

Niech $n=|V|$ (liczba elementów zbiorów V, $\#V$ - inne oznaczenie)

Dygresja: $X=\{a_1,\dots,a_N\}$, $N\in\mathbb{N}$ ← „uniwersum"

$V\subseteq X$, $\vec{v}_V$ - wektor długości N, którego wyrazy (współ.) to 0 i 1 $\vec{v}_V(i)=\begin{cases}1, & \text{gdy } a_i\in V\\ 0, & \text{gdy } a_i\notin V\end{cases}$



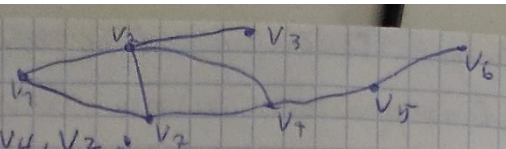
np: $X=\{1,2,3,4,5,6\}$; $N=6$ $V=\{1,3,6\}$

$V=[1,0,1,0,0,1]$

$G=\langle V,E\rangle$ jak wyżej, $V=\{v_1,\dots,v_n\}$

Macierzą sąsiedztwa grafu G nazywamy macierz $A=[a_{ij}]$, gdzie $i,j\in\{1,\dots,n\}$. Ponadto $a_{ij}=\begin{cases}0 & \text{wtw, gdy } \{v_i,v_j\}\notin E\\ 1 & \text{wtw, gdy } \{v_i,v_j\}\in E\end{cases}$

Linked list np:



$v_1 \to v_2, v_7$; $v_2 \to v_3, v_4, v_7$;

$v_3 \to$ — ; $v_4 \to v_5, v_7$; $v_5 \to v_6$; $v_6 \to$ — ; $v_7 \to$ —

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	1	0	0	0	0	1
v_2	1	0	1	1	0	0	0
v_3	0	1	0	0	0	0	0
v_4	0	1	0	0	1	0	1
v_5	0	0	0	1	0	1	0
v_6	0	0	0	0	1	0	0
v_7	1	1	0	1	0	0	0

$v_1 \to v_2, v_7$

$v_2 \to v_3, v_4, v_7$

$v_4 \to v_5, v_7$

$v_5 \to v_6$

Drzewo - graf spójny, cykliczny

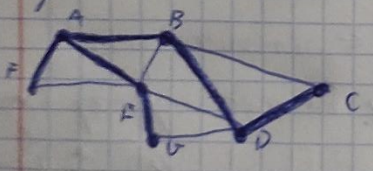
Twierdzenie: Liczba krawędzi w drzewie o n wierzchołkach wynosi n-1.

Każdy spójny graf o n wierzchołkach i n-1 krawędzi jest drzewem.

Każdy cykliczny graf o n wierzchołkach i n-1 krawędziach jest drzewem.

Drzewo spinające grafu spójnego

Dany $G = \langle V, E \rangle$ - graf spójny. Drzewem spinającym grafu G nazywamy taki podgraf T grafu G, który jest drzewem i zawiera wszystkie wierzchołki grafu G.



Jeśli $|E| = m$, $|V| = n$,

$m - (n-1)$ - liczba krawędzi odrzuconych z G by uzyskać drzewo spinające. Rząd cykliczności grafu G.

Przeszukiwanie grafów wgłąb (Depth first search; DFS)

Dany: $G = \langle V, E \rangle$, $|V| = n > 0$

Krok 1: Wybierz dowolny wierzchołek grafu G i nadaj mu etykietę „1". Przejdź do kroku 2 mając jako daną etykietkę „1".

[illegible] etykiety k i wierzchołka nie zaetykietowanego:
a) jeśli istnieje w G wierzchołek sąsiedni do wierzchołka k, który nie ma jeszcze przypisanej etykiety, nadaj mu etykietę będącą najmniejszą liczbą ze zbioru $\{1, \dots, n\}$, która jeszcze nie jest etykietą żadnego wierzchołka. Wróć do kroku 2 z tą ostatnią nadaną etykietą jako k.
b) jeśli nie istnieje w G wierzchołek sąsiedni do k, który nie miałby jeszcze etykiety, to:
- jeśli $k = 1$ to koniec.
- jeśli $k > 1$, wróć do tego wierzchołka z którego dotarłeś do k i powtórz krok 2 biorąc jako nowe k etykietę tego wierzchołka.

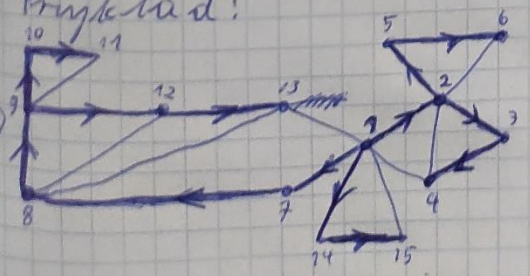
Stwierdzenie 1: Gdy algorytm zakończy działanie to dla każdego k użytego jako etykiety, istnieje w G ścieżka łącząca wierzchołek „1" z wierzchołkiem „k" prowadząca przez zaetykietowane wierzchołki.

Wniosek: Jeśli wszystkie etykiety ze zbioru $\{1, \dots, n\}$ zostały użyte, to •) wszystkie wierzchołki mają etykiety •) graf jest spójny.

Stwierdzenie 2: Jeśli G jest spójny, to DFS etykietuje każdy wierzchołek grafu G.

Stwierdzenie 3: Jeśli G jest spójny, to DFS wyznacza drzewo rozpinające grafu G.

Przykład:



k = 4, do 4 dotarłem z 3
3 nie ma niezaetykietowanego sąsiada
k = 6, do 6 dotarłem z 5
6 nie ma niezaetykietowanego sąsiada
5 nie ma niezaetykietowanego sąsiada
2 nie ma niezaetykietowanego sąsiada, 1 ma niezaetykietowanych sąsiadów. ..... Potem wracamy do 1, k = 1, kończymy

Złożoność obliczeniowa DFS:

Każda krawędź jest rozważana nie więcej niż 2 razy: raz „naprzód" i raz „wracając". Liczba krawędzi w grafie o n wierzchołkach wynosi co najwyżej $\frac{n(n-1)}{2}$.

Łącznie: liczba pojedynczych „kroków" wynosi $\leq 2 \cdot \frac{n(n-1)}{2}$, czyli $\Theta(n^2)$.