ASD

1) Struktury danych
2) Techniki algorytmiczne
3) Analiza algorytmów: "dowodzenie" poprawności, szacowanie złożoności.
4) Klasyczne problemy algorytmiczne: sortowanie, alg. grafowe, słowniki.
5) Klasyczne algorytmy.

Jak analizować algorytmy?

- analiza zużycia zasobów - czas i pamięć
- wnioskowanie - metoda niezmienników.

Def Niech A to program w jakimś ustalonym modelu obliczeń.

Niech $w \in \Sigma^*$ skończone słowo nad alfabetem Σ.
w - jest wejściem naszego programu.

(2)

$|w|$ - długość słowa w

Niech $C_A^T(w)$ = ilość elementarnych operacji jakie A wykonuje na wejściu w.

Co to są elem. op.?

Wg. instrukcje procesora.

Złożoność czasowa A

$C_A^T : \mathbb{N}^+ \to \mathbb{N}^+$, gdzie $\mathbb{N}^+ \cong \mathbb{N} \setminus \{0\}$.

$$C_A^T(n) = \max \{ C_A^T(w) : w \in \Sigma^n \},$$

$|w| = n$.

Σ - alfabet, nad którym tworzymy wejścia do A.

np. $\Sigma = \{0,1\}$, Σ - kody ASCII.
To jest złożoność pesymistyczna.

Złożoś pamięciowa

$C_A^S(w)$ = ilość pamięci wykorzystywana przez A na wejściu w

---

$C_A^S : \mathbb{N}^+ \to \mathbb{N}^+$

$C_A^S(n) = \max\{C_A^S(w) : w \in \Sigma^n\}$

Asymptotyczne klasy wzrostu funkcji.

$f : \mathbb{N}^+ \to \mathbb{N}^+$

$O(f) = \{g : \mathbb{N}^+ \to \mathbb{N}^+ : \exists C>0\ \exists N\ \forall n \geq N\ \ g(n) \leq C \cdot f(n)\}$

$o(f) = \{g : \mathbb{N}^+ \to \mathbb{N}^+ : \forall \varepsilon>0\ \exists N\ \forall n \geq N\ \ g(n) \leq \varepsilon \cdot f(n)\}$

$\Theta(f) = \{g : \text{---}'\text{---} : O(f) = O(g)\}$

Fakt

$g \in O(f)$ wtw $\exists C > 0 \ \forall n \geq 1 \quad g(n) \leq C \cdot f(n) + C$.

Fakt

$\forall f : \mathbb{N}^+ \to \mathbb{N}^+ \quad f \in O(f) \setminus o(f)$.

Notacja. $g = O(f)$, $g = o(f)$, $g = \Theta(f)$

$h(n) = n^2 + o(n)$

$h'(n) = n^2 + O(1/n)$, $\lim (h'(n) - n^2) = 0$

$\lim \frac{h(n)}{n}$ $\lim_{n \to \infty} \frac{h(n)}{n^2} = \lim \frac{n^2 + o(n)}{n^2}$

$= \lim \left(1 + \frac{o(n)}{n^2}\right)$, $\frac{o(n)}{n^2} \to 0$

ale $\lim (h(n) - n^2)$ może być różna ∞ lecz $o(n)$ to już.

Złożoność alg.	Rozmiar problemu który w ciągu 1 sek	1 min.	1 godz.
n	1000	$6 \cdot 10^4 = 60 \cdot 1000$	
$n \log(n)$	140	≈ 4890	
n^2	31	244	
n^3	10	39	
2^n	9	15	

Złożoność alg.	rozmiar zadania który w ciągu dzisiaj	to samo ale dla 10 razy szybszych komputerów
n	s_1	$10 s_1$
$n \log n$	s_2	$\approx 10 \cdot s_2$, dla dużych s_2.
n^2	s_3	$3.16 \cdot (s_3)$
n^3	s_4	$2.15 \cdot s_4$

$$\frac{n^3}{2^n}$$

S_4

S_5

$2.16 \cdot (S_3)$

$2.15 \cdot S_4$

$S_5 + 3.3$

Metody analizy algorytmów.

Metoda niezmienników.

Zapiszmy kod algorytmu:

```
WE: x,y
%α
Instr_1
%β
Instr_2
...
Instr_n
%γ
WY: z
```

α, β, γ - to formuły logiki pierwszego rzędu w języku arytmetyki teorii mnogości wykorzystujące zmienne opisujące stan programu.

niezmienniki

Cel: Jeśli sterowanie programu dotrze do danej formuły to powinna ona być prawdziwa

Niezmiennik pętli

γ - n.p.

while (WAR) {

P_i

}

γ - jest niezmiennikiem pętli jeśli

1) γ jest prawdziwa przed każdym pierwszym wykonaniem pętli (gdy sterowanie ma dopiero wejść do pętli)

2) Jeśli γ zachodzi i zachodzi WAR, to po wykonaniu P_i - programu P_i γ dalej zachodzi.

Przykład – szybkie potęgowanie

WE: $x, y \quad x \geq 0 \quad y \geq 0$

Wy: x^y

begin

$a = x; \quad b = y; \quad c = 1;$

γ - niezmiennik pętli

while ($b > 0$) {

$\gamma := x^y = a^b \cdot c \wedge b \geq 0$

Wymaga dowodu, że γ to niezmiennik

~~$t = a \cdot a;$~~ $\quad (b \bmod 2)$

~~b~~. $c = c \cdot a$

$a = a \cdot a; \quad b = \lfloor b/2 \rfloor;$ $\quad \alpha := x^y = a \cdot c$

}

α

return a ;

}

Dlaczego γ to niezmiennik pętli?

① γ prawdziwa przed pierwszym wejściem do pętli.

② Jeśli γ prawdziwa i warunek wejścia do pętli ten, to γ prawdziwa po wykonaniu pętli.

$\{\gamma \wedge WAR\}\ P\ \{\gamma\} \rightarrow \neg WAR$

$\rightarrow WAR$

$\{\gamma \wedge WAR\}\ P\ \{\gamma\} \rightarrow \neg WAR$

$\rightarrow WAR$

$\{\gamma \wedge WAR\}\ P\ \{\gamma\} \rightarrow \dots$

Ad 1 Oczywiste

~~to~~ $x = a$ i $y = b$

Ad 2.

Zał. że $x^y = a^b \cdot c \wedge b \geq 0 \wedge b > 0$

Niech a', b', c' - nowe wartości a, b, c - po wykonaniu pętli.

$a' = a \cdot a$

$b' = \frac{b}{\lfloor 2 \rfloor}$

$c' = c \cdot (a)^{b \bmod 2} = \begin{cases} c & : b \text{ - parzyste} \\ c \cdot a & : b \text{ - nieparzyste.} \end{cases}$

Teza: $a'^{b'} \cdot c' = x^y \wedge b' \geq 1$. - niezmiennik dla nowych wartości a, b.

Niech $b = 2 \cdot b' + d, \quad d \in \{0, 1\}$

$d = b \bmod 2.$

$(a')^{b'} \cdot c' = (a^2)^{\lfloor b/2 \rfloor} \cdot c \cdot (a)^d$

$= a^{2 \cdot \lfloor b/2 \rfloor} \cdot c \cdot (a)^d$

$= a^{(2 \cdot \lfloor b/2 \rfloor + d)} \cdot c$

$= a^{2b'+d} \cdot c = a^b \cdot c = x^y.$

z założenia z poprzedniej strony

$b \geq 1 \Leftarrow$ tw.

$b' = \lfloor \frac{b}{2} \rfloor \geq 1.$

W takim razie po wyjściu z pętli mamy $b = 1$ ← bo $b \geq 1$ i $\neg(b > 1)$ bo wyszliśmy z pętli

więc $a^b \cdot c = x^y$

$a^b = a \cdot a \cdot \ldots \cdot a$

$\longrightarrow a \cdot c = x^y$. □

Analiza złożoności

Ile razy wykona się pętla?

Niech $|x|$ – długość binarnego zapisu liczby x, $x \in \mathbb{N}$.

Fakt $|x| = \lceil \log_2(x+1) \rceil$.

Teza Złożoność iteracji pętli jest $\leq |y|$.

$|2x| = |x| + 1$

$|\lfloor \frac{x}{2} \rfloor| = |x| - 1$, dla $x > 1$.

$\Rightarrow$ Każdą iterację pętli zmniejszamy $|b|$ o 1.

Więc ilość iteracji nie może być większa od $|b|$, bo $|b| \geq 1$.

Ale uwaga!! Ilość iteracji pętli jest mała ale liczby obliczane w pętli są duże i działania na nich mogą być kosztowne.