§ 9 Klasy P i NP

"Praktyczne" algorytmy pracują w czasie wielomianowym:

Def:

(i) $P := \bigcup_{i \geq 1} DTIME(n^i)$

$NP := \bigcup_{i \geq 1} NTIME(n^i)$

(ii) $PSPACE := \bigcup_{i \geq 1} DSPACE(n^i)$

$NSPACE := \bigcup_{i \geq 1} NSPACE(n^i)$

Twierdzenie

$$P \subseteq NP \subseteq PSPACE = NSPACE$$

Dowód:

$NSPACE(n^i) \subseteq DSPACE(n^{2i})$

Przykłady:

(i) Niech $L \in RL$.

wtedy $L \in DTIME(n) \subseteq P$

(ii) Niech $L \in CF$

Niech $L = L(G)$ dla G w postaci normalnej Chomsky'ego.

$WP_G(x) \in P$:

Niech $w = w_1 \dots w_n \in \Sigma^*$. Tworzymy tabelę T, która w pozycji i,j zawiera $A \in N$, takie że $A \overset{*}{\Rightarrow} w_j \dots w_{j+i-1}$, czyli A generuje podsłowo w długości i zaczynając w j-tym literze.
Więc $A \in T_{i,j}$ jeżeli istnieją reguła $A \to BC$ oraz $1 \leq k \leq i-1$, takie że

$$B \overset{*}{\Rightarrow} w_j \dots w_k \quad \text{oraz} \quad C \overset{*}{\Rightarrow} w_{k+1} \dots w_{j+i-1}$$

tzn. $B \in T_{k,j}$ oraz $C \in T_{i-k,j+k}$.
Tabelę więc możemy wypełnić wiersz po wierszu (programowanie dynamicznie). Jeżeli dla reguły $A \to BC$ istnieje $k < m$, takie że $B \in T_{k,j}$ oraz $C \in T_{m-k,j+k}$, to piszemy A do $T_{m,j}$.
$w = w_1 \dots w_n \in L(G)$ wtw. $S \in T_{n,1}$.

Rozmiar tabelu jest ograniczony przez n^2.
Czas potrzebny do wypełnienia jednej pozycje w tabeli jest proporcjonalny do n,
czyli $w \in L(G)$ można sprawdzić w czasie ograniczonym przez n^3.

(Algorytm CYK,
Cook, Younger, Kasami)

Przykład:

$S \to SC \mid XC \mid AZ \mid AS$

$X \to YB \mid AB$

$Y \to AX$

$Z \to BT \mid BC$

$T \to ZC$

$A \to a$

$B \to b$

$C \to c$

$w = aabbcc$

$i \backslash j$	1	2	3	4	5	6
1	A	A	B	B	C	C
2	$\emptyset$	X	$\emptyset$	Z	$\emptyset$	
3	Y	$\emptyset$	$\emptyset$	T		
4	X	$\emptyset$	Z			
5	S	S				
6	S					

czyli $w \in L(G)$, bo $S \in T_{6,1}$

wypełnienie $T_{4,3}$:

$m = 4$

$Z \to BT$ i $B \in T_{1,3}$, $T \in T_{3,4}$

czyli dla $k=1$, $j=3$ mamy $B \in T_{k,j}$, $C \in T_{m-k,j+k}$

(iii) Niech A będzie formułą boolowską w koniunkcyjnej postaci normalnej, czyli

$$A = C_1 \wedge \ldots \wedge C_m$$

gdzie każda klauzula C_i jest alternatywą literałów. Literał jest zmienną lub negacją zmiennej.

Przykład:

$$A = (x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_4)$$

Problem SAT

Czy A jest spełnialna?

Jeżeli A zawiera n zmienne, $x_1, \ldots x_n$, to istnieją 2^n kombinacje wartości T i F dla $x_1, \ldots x_n$.

Maszyna Turinga generuje niedeterministycznie taką kombinację i sprawdza, czy w klauzule jest conajmniej jeden T. Jeżeli jest, to M akceptuje, czyli A jest spełnialna.

Generowanie jest możliwe w czasie wielomianowym (w n); sprawdzanie jest możliwe w czasie wielomianowym (nawet deterministyczni

czyli $SAT \in NP$

Twierdzenie $L \subseteq \Sigma^*$

$L \in NP$ wtw. istnieją wielomian p oraz predykat R, takie że

(i) R jest obliczalny w czasie wielomianowym

(ii) $x \in L$ wtw. istnieje y, takie że

$|y| \leq p(x)$ oraz $R(x,y)$.

(y nazywa się certyfikat)

Dowód:

"$\Leftarrow$" Maszyna Turinga niedeterministycznie generuje y, takie że $|y| \leq p(|w|)$. Symulując maszynę Turinga M' dla R sprawdza, czy $R(w,y)$.

"$\Rightarrow$" Niech M_L będzie maszyną Turinga czasowo ograniczoną przez wielomian p, która akceptuje L. Alternatywy w obliczeniach M_L są ograniczone przez pewien m; więc obliczenie długości k maszyny M_L można opisać przez słowo γ długości k (nad skończonym alfabetem).

$R(w, \gamma) :\Leftrightarrow \gamma$ opisuje akceptujące obliczenie M_L dla w

wtedy R obliczalny w czasie wielomianowym

oraz $w \in L \Leftrightarrow M_L$ akceptuje w

$\Leftrightarrow \exists y : |y| \leq p(|w|) \wedge R(w,y)$

Twierdzenie intuicyjnie oznacza, że $L \in NP$, jeżeli dla danego certyfikatu można deterministycznie w czasie wielomianowym sprawdzić, czy należy do L (czy jest "rozwiązaniem problemu")

9.2 NP-zupełność

Def: $L, L' \subseteq \Sigma^*$

L redukuje się na L', jeżeli istnieje funkcja f, taka że

(i) f jest obliczalna w czasie wielomianowym przez deterministyczną maszynę Turinga

(ii) $w \in L$ wtw. $f(w) \in L'$ dla wszystkich w

piszemy $L \leq_p L'$.

Twierdzenie

$L, K \subseteq \Sigma^*$

(i) $L \leq_p K$ oraz $K \in P$, to $L \in P$.

(ii) $L \leq_p K$ oraz $K \in NP$, to $L \in NP$.

czyli, jeżeli $L \leq_p K$, to L nie jest trudniejszy niż K.

Def: $L \subseteq \Sigma^*$

(i) L jest NP-trudny, jeżeli $K \leq_p L$ dla wszystkich $K \in NP$.

(ii) L jest NP-zupełny, jeżeli $L \in NP$ oraz L jest NP-trudny.

Problemy NP-zupełne są najtrudniesze w klasie NP.

Twierdzenie

Jeżeli istnieje $L \subseteq \Sigma^*$, takie że

(i) L jest NP-zupełny

(ii) $L \in P$,

to $P = NP$.

Jak identyfikować problemy NP-zupełne?

jasne: $L \leq_p K$ i L NP-zupełny,

to K NP-zupełny.

ale: czy wogóle istnieje problem NP-zupełny?

Ograniczony problem stop

$$S_{NP} := \{ x\#k\#w\#a^m \mid x, k \geq 0,\ w \in \Sigma^*,$$

w jest zaakceptowana przez niedeterministyczną maszynę Turinga M_x używając conajwyżej $|w|^k$ kroków, gdzie $m \geq |w|^k$ $\}$

Twierdzenie

S_{NP} jest NP-zupełny.

Dowód:

Maszyna M akceptująca S_{NP} dla wyjścia $x\#k\#w\#a^m$ sprawdza, czy $m \geq |w|^k$. Jeżeli tak, to M simuluje niedeterministyczną maszynę M_x dla $|w|^k$ kroków, i akceptuje, jeżeli M_x po $|w|^k$ krokow akceptował.

stąd: $S_{NP} \in NP$

Niech $L \in NP$ i M_L niedeterministyczna maszyna Turinga czasowo ograniczona przez wielomian p, która akceptuje L, czyli istnieje k_L, taka że $x \in L$ jest zaakceptowane przez obliczenie mając conajwyżej $|x|^{k_L}$ kroków. Niech x_L będzie indeksem maszyny M_L.

$$f(w) := x_L \# k_L \# w \# a^m, \text{ gdzie } m = |w|^{k_L}$$

redukuje L na S_{NP} w czasie wielomianowym.

Twierdzenie (Cook)

SAT jest NP-zupełny.

Dowód:

Niech $L \in NP$.

konstruujemy funkcję f redukującą L na SAT, czyli formułę A_w, taką że $w \in L \iff f(w) \in SAT$.

A_w opisuje akceptujące obliczenia niedeterministycznej maszyny Turinga M_L, która akceptuje L.

Niech $M_L = (\Sigma, Q, \delta, q_0, F)$.

Dla $w \in L$ długości n więc istnieje obliczenie

$$k_0 \to k_1 \to \dots \to k_{p(n)}$$

taka że $k_0 = q_0 b w b$ oraz $k_{p(n)}$ jest konfiguracją akceptującą.

Możliwe pozycje głowy na taśmie, to

$$-p(n), \dots -1, 0, 1 \dots p(n)$$

konfiguracje opisujemy przez zmienne:

$T_{i,j,k}$ na pozycji i w k-tym kroku jest litera a_j

$H_{i,k}$ w k-tym kroku głowa jest na pozycji i

$Q_{q,k}$ w k-tym kroku M_L jest w stanie q

klausuly

dla $-p(n) \le i \le p(n)$, $0 \le k \le p(n)$

(i) konfiguracja startowa $k_0 = q_0 b w_1 \dots w_n b$

$$T_{0,0,0}$$

$$T_{0,j,w_j}, \quad 1 \le j \le n$$

$$Q_{q_0,0}$$

$$H_{0,0}$$

(ii) kolejna konfiguracja $\delta(q_k, a_j) \ni (q_{k'}, a_{j'}, o)$

$$(H_{i,k} \wedge Q_{q_k,k} \wedge T_{i,j,k}) \longrightarrow \bigvee_{\substack{(q_{k'}, a_{j'}, o) \\ \in \delta(q_k, a_j)}} (H_{o(i),k+1} \wedge Q_{q_{k'},k+1} \wedge T_{i,j',k+1})$$

(iii) konfiguracja końcowa

$$\bigvee_{q \in F} Q_{q,p(n)}$$

(iv) Dodatkowe

$$(\neg T_{i,j,k} \vee \neg T_{i,j',k}), \quad j \ne j'$$

$$\bigvee_{j \in \Sigma} T_{i,j,k}$$

w każdym kroku k jest dokładnie jedna litera na pozycji j

$$(T_{i,j,k} \wedge T_{i,j',k}) \to H_{i,k} \quad , j \neq j'$$

"tylko głowa piszy"

$$\neg Q_{q,k} \vee \neg Q_{q',k} \quad , q \neq q'$$

$$\bigvee_{q \in Q} Q_{q,k}$$

$$\neg H_{i,k} \vee \neg H_{i',k} \quad , i \neq i'$$

$$\bigvee_{-p(n) \leq i \leq p(n)} H_{i,k}$$

Jeżeli $k_0 \to \dots k_{p(n)}$ zaakceptuje, to każda alternatywa jest spełniona, czyli A_w spełnialna.
Jeżeli A_w spełnialna, spełniona A_w opisuje akceptujące obliczenie $k_0 \to \dots \to k_{p(n)}$, czyli $w \in L$, czyli

$$\begin{aligned} w \in L &\iff w \in M_L \\ &\iff \text{istnieje } k_0 \to \dots \to k_{p(n)} \text{ zaakceptujące} \\ &\iff A_w \text{ spełnialna} \\ &\iff A_w \in SAT \end{aligned}$$

A_w można deterministycznie z w konstruować w czasie wielomianowym.

9.3 Dalsze problemy NP-zupełne