Obliczalność i złożoność

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. zimowy 2025/26

inf.ug.edu.pl/~amb/oblicz

Maszyny Turinga, złożoność czasowa

Przykład badania złożoności czasowej

- język $\{0^n \mid n \text{ liczba złożona}\}$
- maszyna niedeterministyczna z dwiema taśmami:
  - wpisuje na drugiej taśmie k zer, $1 < k < n$
  - przesuwa jednocześnie głowicę na obu taśmach, na drugiej cyklicznie gdy skończą się zera
  - kończy gdy skończy się pierwsza taśma
- maszyna kończy po $O(n)$ krokach
- maszyna deterministyczna wypróbowuje $O(n)$ możliwości
- czyli kończy po $O(n^2)$ krokach

Algorytm CYK

- problem: dana gramatyka bezkontekstowa w postaci Chomsky'ego (tzn. produkcje są postaci $A \to BC$ lub $A \to a$), sprawdzić czy słowo $w = w_1 \dots w_n$ jest generowane przez tę gramatykę
- maszyna deterministyczna buduje dwuwymiarową macierz CYK
  - w komórce $CYK[i,j]$ są zmienne gramatyki generujące podsłowo $w_j \dots w_{j+i-1}$, podsłowo długości i zaczynające się w miejscu j
  - dla $i = 1$ mogą być użyte jedynie produkcje typu $A \to a$
  - gdy słowa długości $< m$ zostały rozpatrzone, sprawdzamy, czy $B \in CYK[k,j]$, $C \in CYK[m-k, j+k]$ oraz $A \to BC$ jeśli tak, to $A \in CYK[m,j]$ – generuje słowo długości m w miejscu j
  - pytamy, czy $S \in CYK[n,1]$
- jest $O(n^2)$ komórek w macierzy, każdą wypełniamy kosztem $O(n)$
- złożoność $O(n^3)$

Problem osiągalności w grafie

- zapis grafu o n wierzchołkach zajmuje $O(n \cdot \log n)$ miejsca (numer jednego wierchołka to $\log n$ bitów)
- maszyna deterministyczna wielokrotnie przegląda zapis grafu, wpisuje sąsiadów danego wierchołka i sprawdza, czy jest nowy wierchołek
- czyli n iteracji, w każdej $n^2 \cdot \log n$ operacji
- złożoność liczona względem długości wejścia (nie liczby wierzchołków) jest $O(|\text{wejście}|^3)$

Problem SAT

- wejście: fomuła boolowska w postaci koniunkcyjnej (CNF)
  - tzn. koniunkcja klauzul,
  - klauzula to dysjunkcja literałów,
  - literał to zdanie atomowe lub jego negacja
- maszyna niedeterministyczna zaczyna od wyboru wartościowania dla wszystkich zdań atomowych
- sprawdzenie wartości formuły jest liniowe, sprawdzamy, czy wszystkie klauzule mają wartość *True*
- spełnialność = istnienie wartościowania dającego *True*
- czy istnieje maszyna deterministyczna sprawdzająca spełnialność w czasie dowolnie wielomianowym?
- nie wiadomo (pewnie nie, skoro do dziś nikt nie znalazł rozwiązania)

Certyfikaty

- dla problemu złożoności liczby, maszyna niedeterministyczna zgaduje dzielnik
  - złożoność sprawdzenia podzielności jest wielomianowa
- dla problemu osiągalności maszyna niedeterministyczna zgaduje ścieżkę
- dla problemu SAT maszyna niedeterimistyczna zgaduje wartościowanie zdań atomowych
  - zarówno zapis wartościowania jak i sprawdzenie mają koszt wielomianowy
- def: *certyfikat* to dowód zaakceptowania rozwiązania przez maszynę niedetermistyczną

Użycie certyfikatu

- tw: L jest akceptowany przez maszynę niedeterministyczną M w czasie wielomianowym w.t.w. gdy istnieje wielomian p i maszyna deterministyczna M_D akceptująca w czasie wielomianowym takie, że $L = \{w \mid \exists\, c.|c| \leqslant p(|w|), w\#c \in Lan(M_d)\}$
- c jest certyfikatem zgadniętym przez maszynę niedeterministyczną, jego długość musi być wielomianowa, czas zaakceptowania przez maszynę deterministyczną również wielomianowy
- dw: jeśli istnieje certyfikat, to czas działania maszyny niedeterministycznej jest wielomianowy – $q(|w| + p(|w|))$ gdzie q jest wielomianiem opisującym działanie M_d

Dowód c.d.

- zakładamy, że M akceptuje w czasie wielomianowym $q(|w|)$
- tzn. dla wejścia w istnieje obliczenie o tym czasie działania
- zapis każdej konfiguracji jest $< q(|w|)$, liczba kroków też, całość $< (q(|w|))^2$
- będzie to certyfikat dla maszyny deterministycznej, która sprawdzi, że to rzeczywiście jest ciąg obliczeń maszyny M

Koszt determinizacji

- dla każdej maszyny niedeterinistycznej M istnieje maszyna deterministyczna M_d akceptująca ten sam język
- zakładamy, że czas działania maszyny niedeterministycznej jest $T(n) \geqslant n$
- z maszyną M związana jest stała d ograniczająca liczbę wyborów w każdym kroku
- liczba konfiguracji, które maszyna deterministyczna być może musi przejrzeć, jest ograniczona przez funkcję wykładniczą
- w konkretnych przypadkach istnieją maszyny deterministyczne o mniejszej złożoności

Problemy *NP*-zupełne

- def.: język L redukuje sie wielomianowo do języka K, $L \leqslant_p K$, jeśli istnieje funkcja f obliczalna przez deterministyczną maszynę Turinga w czasie wielomianowym taka, że $w \in L \Leftrightarrow f(w) \in K$
- tw.: jeśli $L \leqslant_p K$ oraz $K \in P$, to $L \in P$
- dw.: maszyna akceptująca problem $w \in L$ najpierw oblicza $f(w)$ a potem bada $f(w) \in K$
- tw.: jeśli $L \leqslant_p K$ oraz $K \in NP$, to $L \in NP$
- def.: język K jest NP-trudny, jeśli każdy język $L \in NP$ redukuje się wielomianowo do K
  jest NP-zupełny, jeśli jest NP-trudny i należy do NP
- tw.: jeśli $L \in P$ i L jest NP-zupełny, to $P = NP$
- nie znamy *ani jednego* przypadku takiego języka

SAT jest *NP*-trudny

- Pokażemy jak zredukować dowolny język z klasy NP do SAT.
- $M = (Q, \Sigma, \Gamma, q_0, \delta, F)$ z jedną taśmą, p – wielomian.
- Jeżeli $w \in L(M)$, to istnieje obliczenie akceptujące maszyny M na słowie w

$$\beta_0 \to \beta_1 \to \cdots \to \beta_{p(|w|)}.$$

  Zakładamy, że:
  - obliczenie ma dokładnie długość $p(n) + 1$;
  - jeżeli w i-tej konfiguracji maszyna w stanie q obserwuje symbol g w j-tej komórce, to j-ty symbol β_i jest postaci (q, g, m), gdzie m jest numerem instrukcji funkcji przejścia zastosowanej przy przejściu do β_{i+1}.
  - każda konfiguracja ma długość $p(n) + 1$ oraz zaczyna się i kończy specjalnym symbolem $\#$;
  - Γ jest alfabetem roboczym, uwzględniającym dodatkowe symbole na obserwowanej komórce.

SAT jest *NP*-trudny, cd

- Zmiennymi formuły są $P_{i,j,g}$, $0 \leqslant i,j \leqslant p(n)$, $1 \leqslant g \leqslant |\Gamma|$.
- Wartość $P_{i,j,g} = \mathit{True}$ oznacza, że w konfiguracji β_i na j-tej pozycji stoi symbol g.
- Formuła $f(w)$, w koniunktywnej postaci normalnej, będzie tak zbudowana, aby spełniały ją dokładnie te podstawienia, które opisują akceptujące obliczenie maszyny M na słowie w

SAT jest *NP*-trudny, cd

- Formuła $f(w)$ jest koniunkcją następujących klauzul i ich koniunkcji
  - $$\bigvee_{1 \leqslant g \leqslant |\Gamma|} P_{i,j,g}$$
    w konfiguracji β_i na pozycji j zapisany jest symbol $g \in \Gamma$
  - $$\bigwedge_{g \neq h} (\neg P_{i,j,g} \vee \neg P_{i,j,h}) = \neg(\bigvee_{g \neq h} P_{i,j,g} \wedge P_{i,j,h})$$
    w konfiguracji β_i na pozycji j zapisany jest najwyżej jeden symbol
  - $$P_{i,0,\#} \wedge P_{i,p(n),\#}$$
    na początku i na końcu konfiguracji β_i znajdują się symbole $\#$

SAT jest *NP*-trudny, cd

- dalsze klauzule:
  - $$P_{0,1,X_1} \vee \cdots \vee P_{0,1,X_k}$$
    symbole $X_1, \ldots X_k$ składają się z pierwszej litery słowa wejściowego w_1, stanu q_0 oraz numerów możliwych instrukcji
  - $$P_{0,j,w_j} \quad \text{dla } 2 \leqslant j \leqslant n$$
    w komórkach od 2 do n mamy resztę słowa wejściowego
  - $$P_{0,j,B} \quad \text{dla } n < j < p(n)$$
    w pozostałych komórkach na taśmie w konfiguracji początkowej β_0 jest symbol blank B
  - $$P_{p(n),1,F}$$
    w ostatniej konfiguracji $\beta_{p(n)}$ mamy (jedyny) stan akceptujący F, a maszyna zatrzymuje się z głowicą na pierwszej komórce

SAT jest *NP*-trudny, cd

- formuła Ψ_i, $1 \leqslant i \leqslant p(n)$, która wymusza, że konfiguracja β_i jest osiągalna w jednym kroku z konfiguracji β_{i-1}

$$\Psi_i = \bigwedge_{0 \leqslant j \leqslant p(n)} \bigvee_{W,X,Y,Z \in \Gamma} h(W,X,Y,Z) \wedge P_{i-1,j-1,W} \wedge P_{i-1,j,X} \wedge P_{i-1,j+1,Y} \wedge P_{i,j,Z}$$

  - j-ty symbol w konfiguracji β_i zależy tylko od trzech symboli sąsiednich o numerach $j-1$, j oraz $j+1$
  - funkcja logiczna $h(W,X,Y,Z)$, przyjmuje wartość *True*, jeżeli symbol Z może się pojawić na pozycji j pod warunkiem, że w poprzedniej konfiguracji symbole W,X,Y stoją na pozycjach sąsiednich $j-1, j, j+1$
    Z jest jednoznacznie określony, bo numer wykonywanej instrukcji jest włączony do symbolu zawierającego obserwowaną komórkę

SAT jest *NP*-zupełny

- długość formuły jest $O(p^2(n))$
- deterministyczna maszyna Turinga może ją wypisać, mając na wejściu słowo w długości n, w czasie ograniczonym przez pewien wielomian
- czyli problem $w \in L$ został zredukowany wielomianowo do $f(w) \in SAT$
- mówiliśmy wcześniej, że sam probem SAT jest *NP*
- czyli SAT jest *NP*-zupełny