# Obliczalność i złożoność

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. zimowy 2025/26

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





# Złożoność pamięciowa





## Model maszyny Turinga

- model maszyny do celów studiowania złożoności pamięciowej:
  - taśma wejściowa, na której znajduje się ciąg $\$w\$$, gdzie $w$ jest słowem wejściowym, a $\$$ jest specjalnym symbolem $\$ \notin \Sigma$ do zaznaczania końca słowa; zawartość taśmy nie zmienia się
  - taśmy robocze.
- definicja: maszyna Turinga działa w pamięci ograniczonej przez funkcję $S(n)$, jeżeli dla słowa wejściowego $w$ żadne obliczenie maszyny na $w$ nie odwiedza więcej niż $S(|w|)$ komórek na żadnej taśmie roboczej.
  – komórki taśmy wejściowej **nie** liczą się do użytej pamięci
  – będziemy zakładać, że $S(n) \geq \log n$





## Konfiguracja maszyny Turinga

- konfiguracja maszyny Turinga z taśmą wejściową i $k$ taśmami roboczymi:

$$(q, i, \gamma_1, j_1, \ldots, \gamma_k, j_k)$$

  gdzie
  - $q$ jest stanem maszyny
  - $i$ jest pozycją głowicy na taśmie wejściowej
  - $\gamma_\ell$ jest zawartością taśmy $\ell$
  - $j_\ell$ jest pozycją głowicy na taśmie $\ell$
- konfiguracja nie zawiera zawartości taśmy wejściowej, ponieważ zawartość się nie zmienia w czasie obliczenia

## Liczba konfiguracji

- liczba konfiguracji maszyny na wejściu $w$ z pamięcią ograniczoną przez funkcję $S(n) \geq \log n$ jest ograniczona przez

$$d^{S(|w|)}$$

  dla pewnej stałej $d$ zależnej od maszyny
- różnych konfiguracji $(q, i, \gamma_1, j_1, \ldots, \gamma_k, j_k)$ jest

$$|Q| \cdot (|w| + 2) \cdot (|\Gamma|^{S(|w|)})^k \cdot (S(|w|))^k$$

- przy założeniu, że $S(n) \geq \log n$ liczbę tę można ograniczyć przez $d^{S(|w|)}$ dla $d = |Q| \cdot 2 \cdot |\Gamma|^{2k}$
  – obliczenia dłuższe muszą prowadzić do powtórzenia się konfiguracji, powstającą pętlę można opuścić





## Klasa złożoności pamięciowej

- def.: język $L$ należy do klasy złożoności pamięciowej $SPACE(f(n))$, $f : \mathbb{N} \to \mathbb{N}$, jeżeli istnieje deterministyczna maszyna Turinga, z taśmą wejściową i taśmami roboczymi, która rozpoznaje $L$ i działa w pamięci ograniczonej przez $f(n)$





## Przykład

- dla języka

$$L = \{u \# u^R \mid u \in \{a, b\}^*\}$$

  istnieje maszyna akceptującą $L$ działającą w pamięci $S(n) = O(\log n)$, czyli $L \in SPACE(\log n)$
- *nie* jest nią maszyna, która zapisze na taśmie pomocniczej pierwszą część słowa a potem sprawdza czy druga część jest odwróceniem
  – złożoność czasowa $O(n)$ ale złożoność pamięciowa również $O(n)$
- pamięć logarytmiczna jest wystarczająca dla maszyny, która po kolei
  - zapisuje $i$-ty symbol słowa wejściowego
  - zapisuje numer $i$, potrzebuje $\log(|w|)$ bitów
  - odczytuje $i$-ty symbol od końca słowa wejściowego i porównuje z zapisanym
  - wraca do początku taśmy roboczej, którą będzie nadpisywać i przechodzi do kolejnego symbolu





## Mnożenie pamięci przez stałą

- twierdzenie:
  dla każdej deterministycznej (lub niedeterministycznej) maszyny Turinga $M$ z pamięcią ograniczoną przez funkcję $S(n)$ oraz stałej $c > 0$ istnieje deterministyczna (odpowiednio niedeterministyczna) maszyna Turinga $N$ z pamięcią ograniczoną przez funkcję $c \cdot S(n)$.
- uzasadnienie:
  jeden symbol taśmy roboczej nowej maszyny $N$ reprezentuje $k$ kolejnych symboli z taśmy oryginalnej maszyny $M$, $k > \frac{1}{c}$
  obliczenie maszyny $M$ jest symulowane krok po kroku; każdy krok $M$ przez jeden krok $N$, $N$ pamięta pozycję głowicy $M$ w takiej $k$-tce symboli
- oczywiście liczba komórek pamięci zmniejsza się tylko dlatego, że ich wielkość odpowiednio się zwiększa

## Właściwe funkcje złożoności

- funkcja $f$ jest **właściwą** funkcją złożoności, jeżeli spełnia warunki:
  - $f$ jest niemalejąca,
  - istnieje maszyna Turinga, która dla słowa wejściowego długości $n$ zatrzymuje się po wykonaniu $O(n + f(n))$ kroków i na taśmie roboczej zapisuje $f(n)$ wyróżnionych symboli.
- przykłady funkcji właściwych $f(n) = const$, $f(n) = n$, $f(n) = \log n$ (i wiele innych)





## Zależności pomiędzy klasami złożoności

- zakładamy, że $f$ jest właściwą funkcją złożoności, zachodzą zawierania
  1. $SPACE(f(n)) \subseteq NSPACE(f(n))$,
  2. $TIME(f(n)) \subseteq NTIME(f(n))$,
  3. $NTIME(f(n)) \subseteq SPACE(f(n))$
- pierwsze dwa zawierania są oczywiste,
- maszyna niedeterministyczna w czasie $f(n)$ może wykonać co najwyżej $f(n)$ kroków
  – liczba różnych konfiguracji maszyny w czasie $f(n)$ jest ograniczona
  – aby sprawdzić, czy istnieje ścieżka akceptująca, wystarczy przeszukać drzewo konfiguracji
  – złożoność czasowa może być zdecydowanie większa, ale
  – do przeszukania wystarczy zapisywać bieżącą ścieżkę, której długość jest ograniczona przez $f(n)$





## Problem osiągalności REACHABILITY

- dla grafu $G = (V, E)$ problem osiagalności to rozstrzyganie czy w grafie istnieje droga od danego wierzchołka $x$ do $y$
- istnieje algorytm dla problemu osiągalności o czasie liniowym względem *opisu grafu* (liczby wierzchołków plus liczba krawędzi)
- jeśli odnosimy się do liczby wierzchołków, to złożoność jest kwadratowa





## $NSPACE(S(n)) \subseteq TIME(c^{S(n)})$

- tw.: jeżeli $S(n) \geq \log n$, to

$$NSPACE(S(n)) \subseteq TIME(c^{S(n)})$$

- dw.: załóżmy, że $L \in NSPACE(S(n))$, dla słowa wejściowego $w$ liczba konfiguracji maszyny $M$ akceptującej $L$ jest ograniczona przez $c_1^{S(|w|)}$ dla pewnej stałej $c_1$ (zależnej od $M$).
- sprawdzenie czy $w \in L$ można sprowadzić do rozstrzygania problemu osiągalności dla grafu konfiguracji
- graf konfiguracji ma rozmiar danych $c_1^{S(|w|)}$, czyli rozstrzyganie problemu osiągalności ma złożoność czasową $c_1^{S(|w|)\cdot 2}$
- przyjmując $c = 2 \cdot c_1$ mamy żądany wynik

## Metoda osiągalności w grafie, c.d.

- algorytm dla badania osiągalności w grafie o złożoności czasowej $O(n^2)$ można wykorzystać do badania czy $w \in L$ na dwa sposoby
  - dla słowa wejściowego $w$ i maszyny $M$ można zbudować macierz sąsiedztwa dla grafu konfiguracji $G(M, w)$, lub
  - graf konfiguracji jest wykorzystywany w sposób niejawny, w każdym kroku, gdy potrzebna jest informacja czy konfiguracje $C_1$ i $C_2$ są sąsiednie w grafie $G(M, w)$, uruchamia się procedurę, która rozstrzyga czy można przejść w jednym kroku z $C_1$ do $C_2$





## Twierdzenie Savitcha

**Twierdzenie**

*Dla każdej niedeterministycznej maszyny Turinga $M_N$ z pamięcią ograniczoną przez funkcję $S(n) \geq \log n$ istnieje równoważna jej deterministyczna maszyna Turinga $M_D$ z pamięcią ograniczoną przez $(S(n))^2$*

- czyli

$$NSPACE(S(n)) \subseteq SPACE((S(n)^2))$$





## Dowód tw. Savitcha

- można założyć, że maszyna $M_N$ ma tylko jedną konfigurację akceptującą $\beta_a$
- maszyna $M_D$ na wejściu $w$ długości $n$ stwierdzi, czy istnieje obliczenie prowadzące od konfiguracji początkowej do konfiguracji $\beta_a$
- jeżeli takie obliczenie istnieje, to istnieje także obliczenie nie dłuższe od $d^{S(n)}$, gdzie $d > 1$ jest pewną stałą zależną od maszyny
- liczby nie większe niż $d^{S(n)}$ można zapamiętać na taśmie $M_D$ w $S(n)$ komórkach





## Dowód tw. Savitcha, c.d.

- definiujemy rekurencyjną procedurę $OBL(\alpha, \beta, k)$, która zwraca wartość *True*, jeżeli istnieje obliczenie maszyny $M_N$ prowadzące od konfiguracji $\alpha$ do konfiguracji $\beta$ w nie więcej niż $k$ krokach
- jeżeli $k = 1$, to zwróć *True*, jeżeli $\beta = \alpha$ lub $\beta$ jest osiągalna z $\alpha$ w jednym kroku
- jeżeli $k > 1$, to dla każdej konfiguracji $\gamma$ maszyny $M_N$ sprawdź, czy zachodzi $OBL(\alpha, \gamma, \lceil \frac{k}{2} \rceil)$ oraz $OBL(\gamma, \beta, \lfloor \frac{k}{2} \rfloor)$
  – jeżeli istnieje $\gamma$, dla którego oba te warunki są spełnione, to zwróć *True*

## Dowód tw. Savitcha, dokończenie

- maszyna $M_D$ wywołuje $OBL(\beta_0, \beta_a, d^{(S(|w|)})$
- liczba wywołań procedury jest ograniczona przez $\log d^{(S(|w|)} = c \cdot S(|w|)$
- pamięć maszyny $M_D$ jest ograniczona przez $c \cdot (S(|w|))^2$, gdyż każde wywołanie wymaga zapamiętania kilku konfiguracji





## PSPACE=NPSPACE

- klasa PSPACE zawiera języki akceptowane przez maszyny Turinga z pamięcią ograniczoną przez wielomiany.
- z twierdzenia Savitcha wynika, że nie jest ważne, czy w definicji klasy PSPACE weźmiemy maszyny deterministyczne czy niedeterministyczne:
  – dla każdej niedeterministycznej maszyny z pamięcią ograniczoną przez wielomian $p$ istnieje równoważna jej deterministyczna maszyna z pamięcią ograniczoną przez funkcję $p^2$, która też jest wielomianem.





## Podsumowanie

- $$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE$$

  gdzie

  $$L = SPACE(\log n),\ NL = NSPACE(\log n)$$

  zawieranie $NP \subseteq PSPACE$ wynika z symulacji obliczenia maszyny niedeterministycznej na deterministycznej – trzeba sprawdzać wszystkie wybory
- **nie wiadomo** czy zawierania są właściwe





## Równoważna charakteryzacja klasy NP przy pomocy certyfikatów

- relacja $R \subseteq \Sigma^* \times \Sigma^*$ na słowach jest **wielomianowo rozstrzygalna** jeżeli istnieje deterministyczna maszyna Turinga rozpoznająca $\{(x, y) |\ R(x, y)\}$ w czasie wielomianowym
- $R$ jest wielomianowo **zrównoważona** jeżeli istnieje liczba naturalna $k$ taka że $|y| \leq |x|^k$ dla $(x, y) \in R$
- $L \in NP$ wtedy i tylko wtedy, gdy istnieje $R$ wielomianowo rozstrzygalna, wielomianowo zrównoważona taka, że

  $$L = \{x |\ \text{istnieje}\ \ y \text{ taki, że } (x, y) \in R\}$$

## Równoważna charakteryzacja klasy NP przy pomocy certyfikatów, c.d.

- jeżeli $L \in NP$ to relację $R$ można zdefiniować następująco

  $(x, y) \in R$ jeśli $y$ jest kodem obliczenia akceptującego dla $x$

- jeśli istnieje $R$ wielomianowo rozstrzygalna, wielomianowo zrównoważona to maszyna niedeterministyczna dla wejścia $x$ zgaduje $y$ spełniające $|y| \leq |x|^k$ oraz używa wielomianowego algorytmu aby sprawdzić czy $(x, y) \in R$
  – jeśli tak, to maszyna akceptuje $x$, inaczej odrzuca
- $y$ nazywamy **certyfikatem** dla $x$





## Równoważna charakteryzacja klasy NP przy pomocy certyfikatów, przykłady

- w problemie spełnialności – certyfikatem jest wartościowanie zmiennych, przy którym formuła jest prawdziwa
- w problemie kolorowania grafów – certyfikatem jest to przyporządkowanie kolorów do wierzchołków
- w problemie ścieżki Hamiltona – certyfikatem jest ścieżka Hamiltona





## Twierdzenie Immermana-Szelepcsényi'ego

**Twierdzenie Immermana-Szelepcsényi'ego**

Dla każdej **niedeterministycznej** maszyny Turinga $M$ z pamięcią ograniczoną przez funkcję $S(n) \geq \log n$ istnieje **niedeterministyczna** maszyna Turinga $M^c$ z pamięcią ograniczoną przez tę samą funkcję $S(n)$, która akceptuje język $\Sigma^* \setminus L(M)$.

**Języki kontekstowe**

Klasa języków kontekstowych jest zamknięta ze względu na uzupełnienie.

- dw.: język kontekstowy jest akceptowany przez niedeterministyczną maszynę Turinga w pamięci liniowej.