Teoretyczne podstawy informatyki

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. letni 2025/26

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

Maszyny Turinga – złożoność

Teoria grafów

- graf skierowany $G = (V, A)$ gdzie $A \subseteq \{(v, w) \in V \times V \mid v \neq w\}$
- krawędzie (łuki) mają kierunek „od v do w" – początek i koniec
- krawędź (v, w) to co innego niż (w, v), mogą istnieć obie niezależnie
- ścieżką jest ciąg różnych wierzchołków $v_0, \ldots, v_n$ takich, że $(v_i, v_{i+1}) \in A$, $i = 0, \ldots, n-1$
- cyklem jest ciąg $v_0, \ldots, v_n, v_0$ taki, że $v_0, v_1, \ldots, v_n$ jest ścieżką i $(v_n, v_0) \in A$
- cykl może być bardzo krótki, v,w,v jeśli $(v, w) \in A$ i $(w, v) \in A$, zaangażowane są tylko dwa wierzchołki
- graf jest silnie spójny jeśli każde dwa wierzchołki łączy ścieżka
- graf jest acykliczny jeśli nie ma cykli
- graf silnie spójny (nie punkt) ma cykle (od v do w i z powrotem)

Kodowanie grafów

- wierzchołki grafu można ponumerować, tzn. przyjmujemy $V = \{0, \ldots, n-1\}$ dla grafu o n wierzchołkach
- dla wierzchołka $k \in V$ mamy listę jego sąsiadów, zapisywaną $L_k = k : i_1, \ldots, i_{j_k}$, gdzie $(k, i_\ell) \in A$
- cały graf zapisujemy jako listę list sąsiedztw tj. $L_0; \ldots; L_{n-1}$
- czyli na taśmie wejściowej mamy zapisane ciągi liczb oraz przecinki, dwukropki i średniki, wszystkie te elementy należą do naszego alfabetu
- można sprawdzić czy taki zapis jest rzeczywiście zapisem grafu
- w konkretnych zastosowaniach można stosować bogatsze dane, np. długości krawędzi, czy inne etykiety

Problemy teorii grafów

- osiągalność: dany graf (skierowany) G, wierzchołki $v, w \in V$, czy istnieje ścieżka od v do w?
- silna spójność: czy istnieje ścieżka od u do v dla dowolnych wierzchołków u, v grafu
- ścieżka/cykl Hamiltona: czy istnieje ścieżka/cykl, który odwiedza wszystkie wierzchołki grafu bez powtórzeń
- kolorowanie grafu: ustalona liczba $k > 0$, pytanie, czy można wierzchołki podzielić na k kolorów tak by krawędzie zawsze miały końce o różnych kolorach
- def: problem jest rozstrzygalny jeśli istnieje maszyna Turinga taka, że dla każdej instancji problemu maszyna zatrzyma się i da prawidłową odpowiedź

Deterministyczna maszyna Turinga rozstrzygająca problem osiągalności w grafie

- maszyna ma trzy taśmy
- taśma pierwsza: zakodowany graf G oraz wierzchołki u i v
- taśma druga: wpisywane będą wierzchołki osiągalne z u, na początku wpisany jest u, będą one sukcesywnie zaznaczane (jako odwiedzone)
- maszyna powtarza cykl, w którym
  - zaznacza nowy wierzchołek w na drugiej taśmie
  - na trzecią taśmę wpisuje wierzchołki połączone z w
  - wierzchołki z trzeciej taśmy wpisuje na drugą, jeśli ich nie było
  - czyści trzecią taśmę
- kończy pracę gdy
  - v znajdzie się na trzeciej taśmie – jest ścieżka
  - lub wszystkie wierzchołki na drugiej taśmie są zaznaczone – nie ma ścieżki

Niedeterministyczna maszyna Turinga akceptująca problem osiągalności w grafie

- są tylko dwie taśmy, pierwsza z grafem i wierzchołkami u i v
- na drugiej taśmie zaczynamy od wierzchołka u i wpisujemy po kolei (dowolnego) sąsiada ostatniego wierzchołka
- jeśli wpiszemy v, to istnieje ścieżka od u do v
- jeśli istnieje w grafie ścieżka od u do v, to istnieje obliczenie akceptujące tę instancję problemu
- jeśli takiej ścieżki nie ma, to żadne obliczenie nie zaakceptuje tej instancji
- jeśli w grafie jest cykl, to istnieje obliczenie nieskończone (skoro nie dbaliśmy o zaznaczanie odwiedzonych wierzchołków)
- od maszyny niedeterministycznej nie oczekujemy własności stopu

Złożoność obliczeniowa (czasowa)

Notacja asymptotyczna

- rozważamy ciągi liczb naturalnych, równoważnie, funkcji $b, c : \mathbb{N} \to \mathbb{N}$
- $b_n = O(c_n)$ (b jest „O duże od" c) jeśli
  $\exists\, C.\ \exists\, n_0.\ \forall\, n.\ n > n_0 \Rightarrow b_n < C \cdot c_n$
  „prawie wszystkie b są mniejsze niż c, stała jest nieważna"
- $b_n = o(c_n)$ (b jest „o małe od" c) jeśli
  $\forall\, C.\ \exists\, n_0.\ \forall\, n.\ n > n_0 \Rightarrow b_n < C \cdot c_n$
  „prawie wszystkie b są dowolnie mniejsze niż c"
- $b_n = \Theta(c_n)$ (b jest „Θ od" c) jeśli
  $\exists\, C_1.\ \exists\, C_2.\ \exists\, n_0.\ \forall\, n.\ n > n_0 \Rightarrow C_1 \cdot c_n < b_n < C_2 \cdot c_n$
  „prawie wszystkie b są mniej więcej równe c"

Notacja asymptotyczna c.d.

- jeśli b jest „o małe od" c, to jest również „O duże"
- z kolei c na pewno nie jest „O duże od" b
- „Θ od" oznacza „O duże" w obie strony
- jeśli istnieje granica ilorazu $\frac{b_n}{c_n}$ skończona, niezerowa, to b jest Θ od c, ale bez tej granicy też może być
- relacja „O duże" jest zwrotna i przechodnia, ale brak słabej antysymetrii, relacja „Theta" jest relacją równoważności
- relacja „o małe" jest porządkiem (przeciwzwrotna i przechodnia)

- $\log n$ jest „o małe od" n, n^k jest „o małe od" n^ℓ, $n < \ell$
- n^ℓ jest „o małe od" 2^n dla dowolnej potęgi ℓ
- $100 \cdot n$ jest „o małe od" $n \cdot \log_2 n$, ale dla $n < 2^{100}$ $100 \cdot n > n \cdot \log_2 n$

Złożoność czasowa

- niech M będzie deterministyczną maszynę Turinga,
- zakładamy, że zatrzymuje się dla każdej konfiguracji początkowej
  - tj. mówimy o rozstrzygalności i języku rekurencyjnym
- def: maszyna działa w czasie $f(n)$, $f : \mathbb{N} \to \mathbb{N}$, jeśli dla dowolnego słowa w długości n, zaczynając działanie w konfiguracji początkowej, kończy po najwyżej $f(n)$ krokach
- język L należy do klasy $TIME(f(n))$ jeśli jest akceptowany przez maszynę Turinga działającą w czasie $f(n)$
- dodajemy literkę N do oznaczenia klasy, jeśli maszyna akceptująca jest niedeterministyczna
  - w tym kontekście nie żądamy rozstrzygnięcia tzn. gwarancji stopu maszyny Turinga

Ważne klasy złożoności czasowej

Czas działania	deterministyczna	niedeterministyczna
logarytm $O(\log(n))$	L	NL
wielomian $O(p(n))$	P	NP
wykładniczy $2^{p(n)}$	EXPTIME	NEXPTIME

- dla każdej maszyny niedeterministycznej da się zbudować maszynę deterministyczną akceptującą ten sam język kosztem złożoności
- problem: czy każdą maszynę niedeterministyczną działającą w czasie wielomianowym da się zamienić na deterministyczną działającą również w czasie wielomianowym?
- czyli: P=NP?
- główny problem teoretycznej informatyki
- w zasadzie nie wierzymy się da

Przykłady złożoności czasowej

- czy liczba n jest pierwsza: dzielimy przez kolejne liczby $\leq \sqrt{n}$, złożoność dzielenia $O(n \cdot \sqrt{n})$ razy liczba prób $O(n^2)$, zapis liczby n zajmuje $\log_2 n$ bitów, czyli złożoność wykładnicza względem długości wejścia
  - istnieją algorytmy, które z dużą pewnością testują pierwszość i mają złożoność wielomianową (np. test Rabina-Millera, $O(n^3)$)
  - istnieją też algorytmy o złożoności wielomianowej (2002, $O(n^{12})$)
- czy słowo jest akceptowane przez gramatykę: algorytm CYK, złożoność $O(n^3)$, gdzie n liczba produkcji gramatyki
- problem osiągalności w grafie: algorytm przeszukiwania wszerz czy wgłąb tylko raz przegląda każdą krawędź – złożoność linowa względem długości wejścia

Problem SAT

- wejście: formuła 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*
- SAT jest *NP*
- czy istnieje maszyna deterministyczna sprawdzająca spełnialność w czasie wielomianowym?
- nie wiadomo (pewnie nie, skoro do dziś nikt nie znalazł rozwiązania)

Problemy *NP*-zupełne

- def.: język L redukuje się wielomianowo do języka K, $L \leq_p K$, jeśli istnieje funkcja f obliczalna przez deterministyczną maszynę Turinga w czasie wielomianowym taka, że $w \in L \Leftrightarrow f(w) \in K$
- tw.: jeśli $L \leq_p K$ 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 $L \leq_p K$ 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*-zupełny

- tw.: dowolny problem z klasy NP można zredukować wielomianowo do problemu SAT (bez dowodu)
- wniosek: problem SAT jest NP-zupełny

Inne problemy *NP*-zupełne

- problem 3SAT (każda klauzula ma najwyżej 3 literały)
- problem kliki w grafie: dla danego grafu G i liczby $k > 0$ rozstrzygnąć, czy istnieje klika (podgraf pełny grafu) zawierający k wierzchołków
- problem pokrycia wierzchołkowego: czy można wybrać k wierzchołków tak, by każda krawędź zawierała co najmniej jeden z nich
- k-kolorowalność grafu
- istnienie ścieżki Hamiltona
- problem komiwojażera: czy istnieje cykl Hamiltona o ograniczonym koszcie