Modele Obliczeń

Wykład 1 - Wprowadzenie

Marcin Szczuka

Instytut Matematyki, Uniwersytet Warszawski

Wykład fakultatywny w semestrze zimowym 2014/2015

Outline

Informacje podstawowe

Prowadzący:

dr Marcin Szczuka - `szczuka@mimuw.edu.pl`
Konsultacje (tymczasowo?) w środy 10:15 – 12:00 pokój 1240
tel. 5544124

Zajęcia:

- Wykład - wtorki 8:30 w sali 5820
- Ćwiczenia - wtorki 10:15 w sali 5820

UWAGA: chciałbym zamienić kolejność ćwiczeń i wykładu!

Informacje dodatkowe

Strona WWW zajęć (chwilowo prawie pusta)
`www.mimuw.edu.pl/~szczuka/modobl/`

Tryb i warunki zaliczania przedmiotu

Wykład

Ocena końcowa i zaliczenie zajęć jest wynikiem egzaminu pisemnego. Na wynik ostateczny ma także wpływ ocena z ćwiczeń.

Ćwiczenia

Ocena z ćwiczeń jest ustalana na podstawie zadania (zadań) zaliczeniowego. Zadanie zaliczeniowe będzie wymagało wykorzystania w praktyce wiedzy z zajęć. Potrzebna może się okazać podstawowa sprawność programistyczna.

Outline

Plan wykładu

Część I - klasyczna teoria obliczeń

1. Wprowadzenie do teorii obliczeń.
2. Podstawowe modele - automat skończony i maszyna Turinga.
3. Model RAM, funkcje rekurencyjne i obliczalne.
4. Równoważność modeli, teza Churcha i jej konsekwencje.
5. Złożoność obliczeniowa, zadania trudne obliczeniowo.
6. Algorytmy aproksymacyjne i heurystyki.

Część II - niestandardowe modele obliczeń

1. Algorytmy randomizowane, symulowane wyżarzanie i maszyny Boltzmanna.
2. Algorytmy genetyczne i programowanie ewolucyjne.
3. Modele sieci neuronowych: wsteczna propagacja, pamięci asocjacyjne i sieci samoorganizujące.
4. Nowe idee (jeśli czas pozwoli) : komputery kwantowe i molekularne, obliczenia na DNA.

Outline

Czym są modele obliczeń?

- Matematyka obliczeniowa vs. matematyka obliczeń.
- Jak opisać proces obliczania?
- Czym jest obliczenie?
- Co to jest algorytm?
- Co można, a czego nie da się policzyć przy współczesnym stanie wiedzy?
- Jak dużo kosztuje obliczenie i czy można ten koszt ograniczyć?

Obliczenie jako funkcja

W teorii obliczeń mamy do czynienia z zadaniami skończonymi (dyskretnymi). Jeżeli zatem przyjmiemy, że wszystkie możliwe konfiguracje wejściowe i wszystkie możliwe konfiguracje wyjściowe ponumerujemy liczbami naturalnymi, to obliczenie jest funkcją:

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

Pytanie: Jak wygląda klasa funkcji naturalnych, którą jesteśmy w stanie zrealizować za pomocą współczesnych komputerów?

Outline

Reprezentacja obliczeń - języki fomalne

Najbardziej powszechny sposób reprezentowania problemów z zakresu teorii obliczeń polega na zastosowaniu opisu za pomocą języków formalnych. Zagadnienie wykonania obliczenia w tym ujęciu sprowadza się do odpowiedzi na pytanie:

Czy jesteśmy w stanie sprawdzić czy podane słowo (ciąg znaków) należy do języka?

Języki formalne bardziej formalnie

- Alfabet Σ – dowolny zbiór znaków wykorzystywanych w danym języku, np.:
  - $\mathbf{B} = \{0, 1\}$ - alfabet dla systemu dwójkowego;
  - $\mathbf{C} = \{0, 1, \ldots, 9\}$ - alfabet dla systemu dziesiętnego;
  - $\mathbf{A} = \{a, b, \ldots, z\}$ - alfabet łaciński wspólny dla wielu języków europejskich.
- Słowo nad alfabetem Σ – dowolny ciąg symboli z alfabetu Σ.
- Σ^n – zbiór słów o długości dokładnie n nad alfabetem Σ.
- $\Sigma^0 = \{\epsilon\}$. Słowo ϵ nazywamy **słowem pustym**.
- Σ^* - zbiór wszystkich skończonych słów nad alfabetem Σ

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

Pytanie: Jaką moc ma zbiór Σ^*?

- **Językiem** nad alfabetem Σ nazywamy dowolny podzbiór $\mathbf{L} \subset \Sigma^*$ – czyli zbiór słów nad alfabetem Σ.
- Język można zadać *explicite*, przez wyliczenie wszystkich należących do niego słów. W praktyce języki zadaje się przez podanie zbioru reguł dla tworzenia słów. Taki zbiór reguł nazywamy **gramatyką** (formalną).

W przypadku języków formalnych interesuje nas odpowiedź na pytanie:

Czy dane słowo $w \in \Sigma^*$ należy do języka $\mathbf{L} \subset \Sigma^*$?

W rzeczywistości znacznie bardziej interesuje nas odpowiedź na równoważne, ale inaczej postawione pytanie:

Jak skonstruować funkcję $f_{\mathbf{L}} : \Sigma^* \to \{0, 1\}$ (algorytm obliczający tę funkcję), która dla zadanego słowa $w \in \Sigma^*$ zwraca 1 gdy słowo w należy do języka $\mathbf{L} \subset \Sigma^*$ i 0 w przeciwnym przypadku?

Zadanie rozpoznawania języka

Problem decyzyjny: czy język jest rozpoznawalny?

Dane: Język $\mathbf{L} \subset \Sigma^*$
Szukane: Algorytm obliczający funkcję $f_{\mathbf{L}} : \Sigma^* \to \{0, 1\}$ taką, że

$$f_{\mathbf{L}}(w) = 1 \Leftrightarrow w \in \mathbf{L}.$$

Trudność problemu rozpoznawania

Nie każdy język jest równie łatwy (trudny) do rozpoznania.
Na przykład dla dowolnej liczby $l \in \mathbb{N}$ zapisanej w systemie dwójkowym (lub dziesiętnym) sprawdź czy:

- l jest parzysta;
- l jest podzielna przez 13;
- l jest pierwsza.

Outline

Obliczalność i złożoność obliczeniowa

Obliczalność

Istnieją klasy zadań (języków, funkcji), o których wiadomo, że nie dadzą się w ogólności policzyć dla każdej konfiguracji danych wejściowych. Takie problemy (języki, funkcje) nazywamy **nieobliczalnymi**.

Złożoność obliczeniowa

Możemy charakteryzować różne rodzaje obliczeń (algorytmów, funkcji, języków) poprzez szacowanie kosztu czasowego (liczby elementarnych kroków w algorytmie) i pamięciowego (rozmiaru struktur danych, które musimy przechowywać) w stosunku do rozmiaru danych wejściowych. Charakterystyki złożoności przeważnie biorą pod uwagę rodzaj zależności (np. wielomianowy) między rozmiarem danych wejściowych, a rozmiarem (kosztem) obliczenia.

Pytanie: Jakie klasy funkcji (zadań, języków) można obliczyć korzystając z istniejących środków technicznych?

- Klasa funkcji praktycznie obliczalnych jest zawarta w klasie **funkcji rekurencyjnych**.
- Istnieją funkcje rekurencyjne, kórych obliczanie w praktyce przekracza nasze możliwości (funkcja Ackermanna, funkcja Sudana). W tym sensie są praktycznie "nieobliczalne".
- Istnieją zadania, które zachowują się jeszcze gorzej. Nieobliczalne algorytmicznie funkcje są zwykle związane z problemami **nierozstrzygalnymi**.

Złożoność obliczeniowa

Interesuje nas przede wszystkim złożoność czasowa, czyli liczba elementarnych kroków potrzebnych do wykonania obliczenia w zależności od rozmiaru danych wejściowych.

Możemy wyróżnić kilka podstawowych typów zadań:

- Bardzo proste i proste – złożoność stała, logarytmiczna, liniowa, ...
- Względnie proste – złożoność wielomianowa.
- Trudne, ale nie beznadziejne – złożoność co najwyżej wykładnicza.
- Beznadziejne – złożoność co najmniej wykładnicza.

W przypadkach trudnych (ale nie beznadziejnych) możemy się posiłkować algorytmami aproksymacyjnymi, heurystykami i niestandardowymi modelami obliczeń.

Działania w sytuacjach trudnych

Zadania, w których koszt czasowy jest co najwyżej wykładniczy będą nas szczególnie interesować. Będziemy badać własności tej klasy problemów.

Czasem będziemy skłonni pójść na ustępstwa, pozwalając algorytmowi zwracać nieoptymalne (ale dopuszczalne) rozwiązania, w zamian za znaczące ograniczenie kosztu.

Będziemy się także zajmować różnorodnymi nieklasycznymi modelami obliczeń, które pozwalają otrzymywać pożądane efekty przy wykorzystaniu np. inspiracji płynącej z nauk przyrodniczych (genetyka, neurofizjologia). Te modele jednak, są znacznie mniej regularne i trudniejsze do kompletnego opisania.

Outline

Literatura

- T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein
  Wprowadzenie do algorytmów (nowe wydanie)
  WNT, Warszawa, 2004.
- J.E. Hopcroft, J.D. Ullman
  Wprowadzenie do teorii automatów, języków i obliczeń.
  Wydawnictwo Naukowe PWN, Warszawa, 2003.
- C.H. Papadimitriou:
  Złożoność obliczeniowa.
  WNT, Warszawa, 2002.
- Z. Michalewicz:
  Algorytmy genetyczne + struktury danych = programy ewolucyjne.
  Wydanie 3, WNT, Warszawa, 2003.
- D.E. Goldberg:
  Algorytmy genetyczne i ich zastosowania.
  WNT, Warszawa, 2003.

Literatura c.d.

S. Osowski
Sieci neuronowe w ujęciu algorytmicznym.
WNT, Warszawa, 1996.

J. Hertz, A. Krogh, R.G. Palmer
Wstęp do teorii obliczeń neuronowych.
WNT, Warszawa, 1993.

Ponadto bardzo wiele innych książek, artykułów i notatek do wykładów dostępnych w bibliotece i (przede wszystkim) w Internecie. Należy tylko brać poprawkę na różnorodność metod opisu używanych w zagadnieniach algorytmiki i złożoności obliczeniowej.