Analiza dla informatyków I

DANI1 - ver. 2

Mieczysław Cichoń - WMI UAM

Liczby rzeczywiste

Komputer może posługiwać się typem *real*, ale nie są to jednak liczby rzeczywiste, a jest on

skończonym zbiorem reprezentantów przedziałów.

Jak się przekonamy na wykładzie - liczby rzeczywiste to skomplikowany obiekt...

Całą pracę za komputer musi wykonać *programista*....

Obliczenia numeryczne = obliczenia (przybliżone) na danych typu *real* .

Np. liczba rzeczywista $e \in \mathbb{R}$ będzie bezpośrednio potrzebna (np. przy zagadnieniach wizualizacji) - a wprowadzimy ją na wykładzie...

Unikamy problemów... liczby rzeczywiste

Na początek banalne zagadnienie:

$$\underbrace{\frac{1}{3} \cdot \frac{1}{3} \cdot \ldots \cdot \frac{1}{3}}_{\text{n - krotnie}} \cdot \underbrace{3 \cdot 3 \cdot \ldots \cdot 3}_{\text{n - krotnie}} = ??$$

Drugie pytanie: w tym przypadku stosujemy wzory na pierwiastki trójmianu czy wzory Viète'a ?? Co lepiej? Sprawdzić na

$$w(x) = x^2 - k \cdot x + 1$$

dla odpowiednio (do stosowanej arytmetyki zmiennopozycyjnej) dużych k...

Proszę być świadomym ograniczeń obliczeń komputerowych:

"Diagonal arguments are used to prove many fundamental results about the limitations of computation, such as the undecidability of the Halting Problem for programs and the inherent, unavoidable inefficiency (exponential time or worse) of procedures for other computational problems. So computer scientists do need to study diagonal arguments in order to understand the logical limits of computation. Ad a well-educated computer scientist will be comfortable dealing with countable sets, finite as well as infinite."

(Mathematics for Computer Science, Eric Lehman, F Thomson Leighton, Albert R Meyer)

Zero. Przybliżenia.

Pewnie większość czytelników napotkała w "C++" funkcję *epsilon()* lub FLT_EPSILON, DBL_EPSILON , LDBL_EPSILON (dla typów zmiennoprzecinkowych). Ta pierwsza wg dokumentacji to "najmniejsza liczba większa niż 1 reprezentowalna w podwójnej precyzji". Czyli przy 64-bit IEEE podwójnej precyzji, mamy 52-bitową mantysę i 11-bitową cechę:

$$1.0000000000000000000000000000000000000000000000000000 \times 2^0 = 1$$

A więc:

$$1.0000000000000000000000000000000000000000000000000001 \times 2^0 = 1 + 2^{-52}$$

Czyli

$$epsilon() = (1 + 2^-52) - 1 = 2^{-52}.$$

to takie "zero"...

Kresy i punkty skupienia zbiorów.

Kresy zbiorów: pozwala to np. na obliczanie $\sqrt{x}$ (tu: bazujące na nierówności pomiędzy średnimi, inne algorytmy później...): niech $a_0 = 1$, $b_0 = x > 0$. Definiujemy

$$a_{n+1} = \frac{2a_n \cdot b_n}{a_n + b_n},$$

$$b_{n+1} = \frac{a_n + b_n}{2}.$$

Sprawdzić, że $\sqrt{x} = \sup_{n\in\mathbb{N}} \{a_n\} = \inf_{n\in\mathbb{N}} \{b_n\}$.

Co do **punków skupienia zbiorów**, to czasami ich wykorzystanie jest nieoczywiste. Przywołam algorytm funkcji skrótu MD5. W obliczeniach potrzebne są stałe $K[i] := floor(2^{32} \cdot abs(sin(i + 1)))$. Ale właściwie dlaczego takie? Dlaczego w ogóle wiemy, że to różne stałe? Co wiemy o zbiorze $\{\sin i : i \in \mathbb{N}\}$? Jak usprawniono ten algorytm? Chcecie coś usprawnić? To trzeba zrozumieć jak dział to co jest...

Funkcje.

Fakt, że badanie funkcji jest niezbędne informatykom nie podlega chyba (??) dyskusji (a już funkcje logarytmiczna i wykładnicza przy szacowaniach błędów metod, to już absolutna podstawa). Ale twierdzenia o ich własnościach?

Funkcje 2.

Proste **zastosowania**:

- twierdzenie o złożeniu funkcji obliczalnych (teoria obliczalności),
- funkcje tworzące i ich własności przy badaniach rekurencji,
- interpolacja trygonometryczna (funkcje okresowe),
- funkcje skótów (haszujące),
- problemy złożoności obliczeniowej (np. funkcje logarytmiczne i wielomianowe),
- w metodach numerycznych własność Darboux przy badaniu istnienia rozwiązań równań nieliniowych (powiemy o tym przy okazji metody bisekcji),
- funkcje tworzące - dla "matematyki dyskretnej" zastosowanej w informatyce,
- grafika komuterowa, wizualizacja, analiza obrazów (a tam funkcje trygonometryczne, pochodne) itd.

Funkcja pierwiastek $\sqrt{x}$.

Metoda Newtona-Raphsona (bazująca na geometrii - jako bok pola kwadratu o boku pierwiastek z x, rozpoczynamy od prostokąta i zmniejszamy różnicę pomiędzy długościami boków korzystając ze średniej arytmetycznej) $a > 0$:

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) \ , \ x_1 = \frac{a}{2}.$$

Sprawdzić, że ten ciąg jest zbieżny do $\sqrt{a}$ - lub np. znaleźć kres dolny zbioru takich liczb...

(por. też ćwiczenia (!) z analizy)

Funkcje 2.

A jak programy obliczają wartości funkcji? Czy jest "najlepszy algorytm"? Dla zainteresowanych **przegląd** algorytmów dla funkcji $f(x) = \sqrt{x}$ można znaleźć tu:

https://www.codeproject.com/Articles/69941/Best-Square-Root-Method-Algorithm-Function-Precisi

Funkcje 3.

Z bardziej zaawansowanych zastosowań (bez metod numerycznych):

- Grafika komputerowa: interpolacja, transformaty (Fouriera w JPEG czy falkowa w formacie JPEG 2000) (i algebra liniowa),
- Optymalizacja: cały rachunek różniczkowy (i algebra liniowa),
- Robotyka (i inne modelowania fizyczne): analiza funkcji wielu zmiennych,
- Transmisja danych (np. oszczędne algorytmy przesyłu strumieniowego): rachunek różniczkowy stosowany do probabilistyki, (przesył danych - transformaty Fouriera itp.),
- Analiza algorytmów - o tym szerzej poniżej (np. asymptotyka)...
- Jako metoda komunikacji z użytkownikami oprogramowania!!
- Algorytmy kryptograficzne: istotna różnowartościowość funkcji, a także własności pewnych klasycznych funkcji (np. funkcja sinus w algorytmie MD5),
- ...

A może rekurencja? [N. Wirth]

Obliczanie $\frac{1}{x}$ dla rzeczywistych liczb x (poprzez mnożenie i dodawanie).

Dane: $a_0 = 1$, $c_0 = 1 - x$ $(x \neq 0)$. Definiujemy:

$$a_n = a_{n-1} \cdot (1 + c_{n-1}) \qquad c_n = c_{n-1} \cdot c_{n-1}$$

Wtedy (dlaczego?)

$$\lim_{n \to \infty} a_n = \frac{1}{x}$$

Ale **najpierw (!!)** wypadałoby sprawdzić, że te **ciągi** są **zbieżne** (tw.: Ciąg monotoniczny i ograniczony **jest** zbieżny.) !!

Dokładnie ta sama idea służy do obliczeń sum szeregów (a więc i **wartości funkcji** np. $\sin x$, czy e^x).
I ponownie: najpierw zapewnić zbieżność szeregu, potem liczyć.
Problem dodatkowy: oszacowanie błędu !!?? [N.Wirth str. 52-53]

Rekurencja - pytanie.

Dane są ciągi:

$$a_{n+1} = (a_n)^2 \text{ o ile } (a_n)^2 < 2 \text{ oraz } a_{n+1} = \frac{1}{2}a_n^2 \text{ o ile } (a_n)^2 \geq 2,$$

$$b_{n+1} = \frac{b_n}{2},$$

$$s_{n+1} = s_n \text{ o ile } (a_n)^2 < 2 \text{ oraz } s_{n+1} = s_n + b_{n+1} \text{ o ile } (a_n)^2 \geq 2.$$

Pytania: czy istnieje i jaka jest granica ciągu (s_n)?

Kiedy istnieje (dla jakich wartości początkowych: rozpocznij od $a_0 = x > 0$, $b_0 = 1$ i $s_0 = 0$)?

Jak zakończyć obliczenia (warunek stopu)?

Rekurencja - c.d.

Zadania do przemyślenia (etapy rozumowania, różnice pomiędzy przykładami):

$$a_0 = \sqrt{3}\ ,\ a_{n+1} = \sqrt{3 + a_n}.$$

$$b_0 = \sqrt{3}\ ,\ b_{n+1} = (3 + b_n)^2,$$

$$f_0 = 1\ ,\ f_1 = 1\ ,\ f_{n+2} = f_{n+1} + f_n.$$

(liczby Fibonacci'ego)

$$y_0 = c = const.\ ,\ y_{n+1} = \frac{1}{n+1} - 5 \cdot y_n$$

(ostatnia relacja to obliczanie rekurencyjne pewnych całek, tylko y_0 należy obliczyć wstępnie).

Bisekcja.

To prosta (czyżby?) metoda znajdowania przybliżonego rozwiązania równania.

Weźmy równanie (dla ułatwienia jest to wielomian o współczynnikach całkowitych):

$$W(x) = x^5 - 8x^4 + x + 11.$$

Na początek jest łatwo: wiemy (dzięki matematyce!), że mamy 1, 3 lub 5 rozwiązań rzeczywistych (można wykonać wykres, o ile potrafimy...).

Obliczamy kilka wartości w różnych punktach (zastanowić się jak je wybierać). Np. $W(0) = 11 > 0$, $W(2) = -83$, $W(-2) = -151$, $W(10) = ... > 0$. Zlokalizowaliśmy co najmniej 3 rozwiązania (a ile ich jest?).

Bisekcja - c.d.

Teraz stosujemy znany algorytm bisekcji, **ALE ...**

1. Ile jest rozwiązań?
2. Czy na pewno w przedziałach $[-2, 0]$, $[0, 2]$ i $[2, 10]$ mamy rozwiązanie i jest ono **jedyne**?
3. Czy (i jak?) można oszacować błąd przybliżenia?
4. O ile przyjmiemy zakładaną dokładność przez ε, to co się stanie z algorytmem, gdy $|x_1 - x_2| < \varepsilon$ (x_1, x_2 - rozwiązania)?

Dobre? No to

$$\sin \frac{2\pi}{x} = 0 \qquad x \in \left[\frac{1}{10000000}, 1\right]$$

i powodzenia... Problemem jest też powolna zbieżność metody...

Pomoże matematyka (i analiza matematyczna)...

Błędy...

Proces aproksymacji to ważny punkt analizy matematycznej. Obliczanie przez komputer wyrażenia z zadaną dokładnością nie jest banalne gdy obliczamy wartości rzeczywiste x. Przecież już w punkcie wyjścia mamy wartość przybliżoną (np. przekątna kwadratu o boku 1 ...). Przykłady ograniczania błędów (f - "trudna", g - "łatwa" obliczeniowo):

$$f(x) = x \cdot \sin x \quad \text{oraz} \quad g(x) = x^2$$

mają "bliskie" wartości dla x w otoczeniu zera

$$f(x) = \frac{x^2 + 1}{x} \quad \text{oraz} \quad g(x) = x$$

mają "bliskie" wartości dla "dostatecznie" dużych x.

A jak rola analizy matematycznej? Koniec z cudzysłowami, skorzystamy z **granic i asymptot** (symbole o "małe" i O "duże").

Lubimy wielomiany...

Czy to był przypadek, że do przybliżeń użyliśmy wielomianów? Oczywiście nie...

Mają one oczywistą cechę ułatwiającą ich stosowanie w **informatyce**: można je utożasamić z ciągami ich współczynników, czyli $f(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ wystarczy zapisać jako $(a_n, a_{n-1}, ..., a_1, a_0)$ i operacje obliczanie pochodnej czy całki są dokładne, np. $f'(x) \sim (0, n \cdot a_n, (n-1) \cdot a_{n-2}, ..., a_1)$. Jeśli więc funkcję można przybliżyć wielomianem (potrzebne metryki = odległość pomiędzy funkcjami!), to informatycy chętnie to stosują (np. interpolacja wielomianowa, szeregi potęgowe). No i oczywiście aproksymacja wielomianowa, gdzie wspomniane wyżej cechy są szczególnie ważne...

Nie zapominajmy o funkcjach tworzących (wielomianowych) i ich roli w informatyce...

Funkcje asymptotycznie niewiększe.

Prawie przy każdej okazji przedstawiania algorytmu podany będzie np. rząd jego złożoności obliczeniowej: "O" duże = symbol Landau'a, np. sortowanie o złożoności $O(n \log n)$, co posłuży do oceny i porównywania algorytmów. Alternatywą są nierówności pomiędzy ciągami dowodzone poprzez indukcję matematyczną...

Funkcja asymptotycznie niewiększa od funkcji $g(n)$ to taka funkcja $f : \mathbb{N} \to \mathbb{R}$, dla której istnieją $c > 0$ i $n_0 \in \mathbb{N}$, że $|f(n)| \leq c \cdot |g(n)|$ dla (prawie) wszystkich $n \geq n_0$.

Podstawowym zastosowaniem notacji asymptotycznej **w informatyce** jest szacowanie długości działania programów, w szczególności procedur rekurencyjnych, ktrych złożoność łatwo opisać równaniem rekurencyjnym.

Patrz też notacje: "duże Theta" $\Theta(n)$ i "duże Omega $\Omega(n)$ (ich warunki wystarczające w języku granic ciągów)...

Czasowa złożoność obliczeniowa.

Oznacza to, że $|f(n)| \leq c \cdot |g(n)|$ zachodzi dla (prawie wszystkich) liczb naturalnych n, czyli po prostu (warunek wystarczający)

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

lub nawet niekiedy warunek stosowany ogólniej:

$$\limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty.$$

Zbiór funkcji asymptotycznie nie większych niż $g(n)$ oznaczamy przez $O(g(n))$. Przykładowe zastosowanie w informatyce: **twierdzenie o rekursji uniwersalnej** (szacowanie długości działania programu wraz ze wzrostem ilości danych - oczywiście asymptotyczne oszacowanie).

http://th-www.if.uj.edu.pl/~erichter/dydaktyka/Dydaktyka2013/TPI-2013/TPI-wyklad-3-2013-newTempl.pdf

Funkcje asymptotycznie podobne (równe).

Jeżeli $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1 < \infty$, to funkcje są asymptotycznie równe (czyli $f(n) \sim g(n)$). Studenci matematyki uczą się np. wzoru Stirliga

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

(o liczbie e powiemy oczywiście na wykładzie...), a w informatyce (kryptografia) np. przybliżenie na ilość liczb pierwszych nie większych niż n

$$\pi(n) \sim \frac{n}{ln\ n}.$$

Możliwe zastosowanie wzoru Stirlinga dla **informatyków**: np. oszacowanie liczby cyfr rozwinięcia dziesiętnego liczby 999!.

Funkcje asymptotycznie mniejsze.

Kolejny szczególnie ciekawy przypadek:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$$

czyli symbol "o" małe..., czyli istnieje n_0, takie, że dla dowolnego $c > 0$ nierówność $|f(n)| < c \cdot |g(n)|$ zachodzi dla wszystkich liczb naturalnych $n \geq n_0$.

W **informatyce** - np. przydatne w badaniach złożoności obliczeniowej, jak w twierdzeniach o hierarchii czasowej i pamięciowej czy w szacowaniu reszty we wzorze Taylora = błędu lub w badaniach złożoności czasowej algorytmów (np. istotny wynik: dla każdego k mamy $\log_2 n = o(n^k)$ - algorytm przeszukiwania połówkowego), patrz też - później - tw. Stolza i reguła de l'Hôspitala....

Poza tym dzięki twierdzeniu: jeżeli $f(n) = o(g(n))$, to $f(n) = O(g(n))$, pojęcie będzie przydatne bezpośrednio.

Pochodne.

Najprostsze przykłady zastosowań:

- metoda stycznych,
- szacowanie błędów wzorów interpolacyjnych (np. Lagrange'a),
- AI i automatyka : modelowanie dynamiki bardziej złożonych układów,
- przy korzystaniu z funkcji tworzących,
- grafika komputerowa i wizualizacja (w tym metody numeryczne),
- .....

Całki 1.

Klasyczny przykład zastosowania całeg w **grafice komputerowej** to równanie renderowania Kajiya (co gorsza - potrzebne na ogół metody całkowania numerycznego :-)).
Inny przypadek: w **teorii kolejkowania** - np. równanie całkowe Pollaczka.

Absolutna "klasyka" - szacowanie sum (częste w obliczeniach) poprzez całki:

$$\int_{m-1}^{n} f(x)\,dx \leq \sum_{k=m}^{n} f(k) \leq \int_{m}^{n+1} f(x)\,dx.$$

A teraz zadanie: jakie są **założenia**, aby powyższy wzór był prawdziwy? Odpowiedzi szukaj na wykładach z analizy...

Na pocieszenie...

Zawsze da się znaleźć jakąś pracę dla “informatyka” (”praktyka”) nie znającego matematyki!