"Wybrałem studia z informatyki - po co mi matematyka i analiza matematyczna?!"

Przemyślenie (DLA) studentów ...

Studiujecie informatykę tylko dla "papierka", ale wiecie już najlepiej co będziecie robić i "wiecie", że matematyka nie będzie Wam potrzebna. Serio?? Będziecie pracowali ze 40 lat - a teraz pomyślcie jak wyglądała informatyka 40 lat temu i co robią ówcześnie kształceni informatycy! Kto się nie dostosował - ten nie pracuje! Taki kierunek wybraliście... Aby zrozumieć co i dlaczego zmienia się w informatyce oraz mieć zdolność dostosowania - trzeba zrozumieć matematyczne podstawy informatyki...

No dobrze, **nie jest** argumentem, że program studiów układają (również) informatycy, a więc widzą co robią... (choć to prawda). Wiecie swoje. Na pewno? Czy poniższe uwagi są dla wszystkich *"informatyków"*? Tak, chyba, że ktoś ma specyficznie wąską wizję pracy (obsługa jednego programu np. bazy danych czy sieci komputerowej), ale nawet wtedy wypadałoby *rozumieć* co się robi...

1. Nie jest to nasz wymysł. Ciekawe programy matematyki (i analizy matematycznej) mają uczelnie z Cambridge czy Princeton (patrz wykład...). I mówimy tu tylko o wstępnych informacjach ("Mathematics for Computer Science"), bo dalszy program (i ich, i nasz...) zawiera kolejne przedmioty poszerzające wiedzę matematyczną (to też będzie pokazane na wykładzie). Nie wspomnę o uczelniach, które tego nie uczą bo... to zbyt łatwe i studenci taki "wstęp" sami sobie opracują, a omawia się zaawansowane działy matematyki (np. Politechnika w Zurychu i teoria falek).

   Polecam też książkę z której korzystam: M. Oberguggenberger, A. Ostermann, *Analysis for Computer Scientists: Foundations, Methods, and Algorithms*, Springer Science and Business Media, 2011.

2. To mało konkretne, a kto miałby to czytać? :-) Dla matematyków - zbyt nieściśle podane, dla informatyków - niezauważalne (oczywiste?).

   A jak mam podać przykłady, skoro Czytelnik dopiero studiuje informatykę? Wybiorę więc kilka prostych (informatycy-praktycy zgodzą się lub nie - zależy co robią...). Zacznę od liczb: w szkole średniej były działania na licznach naturalnych, całkowitych czy wymiernych. A liczby rzeczywiste były ... mało objaśniane. Nie bez powodu. Na wykładzie podamy sobie aksjomatykę liczb rzeczywistych (z konsekwencjami) - czyli to, czego nie było w szkole średniej (za trudne?).

   Niestety - dla komputera to JEST za trudne! Tak naprawdę obliczenia są prowadzone na zbiorze liczb całkowitych (i to nie wszystkich :-)). Arytmetykę komputerową omówi kto inny, ale wypadałoby "pomóc" komputerowi (znając liczby rzeczywiste - ograniczać błędy, przyspieszać obliczenia itp., np. kolejność wykonywania operacji).

3. Ależ nie potrzebujemy liczb rzeczywistych (no, skoro komputer ich nie "zna", to nie potrzeba). To po co funkcje, ich własności itp.?

   Popatrzmy: biorąc ciągi rzeczywiste dodatnie a_n i b_n (np. iteracje pewnych obliczanych wielkości) mamy wybrać "lepszy" z nich i oczekujemy $a_n = O(b_n)$ (pojęcie powszechne w ocenach algorytmów). Jak to **najłatwiej** sprawdzić? Obliczyć pewne granice górne (a cóż to jest?)... Nie jest łatwiej? To np. (w uproszczeniu - przepraszam matematyków) skorzystać z funkcji f i g takich, że $f(n) = a_n$ i $g(n) = b_n$ i (o ile można) z reguły de l'Hôspitala (por. wykład z analizy...).

   Odnośnie granic ciągów: zastosowania w informatyce są oczywiste (?), definicję podamy na wykładzie, a ja proponuję wymyśleć algorytm obliczeniowy (z zadaną dokładnością ε). Pytanie: jak zakończyć obliczenia? Oby nie poprzez badanie różnicy pomiędzy dwoma kolejnymi krokami (tj. $|a_{k+1} - a_k| < \varepsilon$)! Proszę spróbować dla $a_n = \log n$, $\varepsilon = 2^{-10}$ i odpowiednio dużych k...

4. Wróćmy na chwilę do ogółu: ścisłość rozumowania (dowody) - przecież sprawdzenie poprawności każdego **algorytmu** to **dowód**, czyli należy poznać zarówno metody dowodowe (np. indukcja), jak i weryfikację założeń. Typowy (niestety) błąd w opracowaniach "dla informatyków" to brak weryfikacji, czy badany obiekt **istnieje** i **jest jednoznacznie** określony. A to z kolei wymaga często kolejnej wiedzy matematycznej.

   Prosty przykład: wiele osób utożsamia algorytm ze wzorem! A gdzie założenia gwarantujące poprawność? Gdy szukamy rozwiązań równania nieliniowego $f(x) = 0$, $x \in [a, b]$, to wzór na iteracje jest prosty: wybieramy punkty "początkowe" iteracji x_0 i x_1 (no dobrze, tu też jest reguła jak to zrobić np. "falsi") i kładziemy $x_{n+1} = x_n - \frac{f(x_i 1)(x_i 1 x_i 2)}{f(x_i 1) f(x_i 2)}$. I już? Działa?

   No - nie zawsze (przykłady na wykładzie lub ćwiczeniach). A założenia **gwarantujące** zbieżność metody? Funkcja musi mieć pochodne rzędu I i II ciągłe (czyli $f \in C^{(2)}(a, b)$) i mające stały znak w (a, b), no i ma mieć *jedno* rozwiązanie w tym przedziale. Uaaa... to jednak pojęcia z analizy matematycznej... (a inne algorytmy korzystają nawet z teorii interpolacji).

5. Inna sprawa: czy wiesz skąd wynikają zmiany formatu zapisu plików (przynajmniej większość :-))? Przykład dla grafiki. Jeśli "nie potrzeba" matematyki, funkcji i analizy matematycznej, to pozostaniemy przy grafice bitmapowej: punkt po punkcie podajmy jego atrybuty (np. położenie, składowe barwy czy jasności). Ale jeśli to dla kogoś niewygodne, to pozostaje grafika rastrowa (lub nawet wektorowa choćby PDF, CDR czy SVG) np. JPEG. I nie wchodząc w szczegóły: potrzebna jest (dyskretna) transformata Fouriera. A jej wprowadzenie "matematyczne" to ciągi i szeregi funkcyjne, szeregi Fouriera i ciągła transformata Fouriera... Za skomplikowane? To dlaczego **i jak** powstał kolejny format JPEG 2000 (podpowiem: dyskretna transformata falkowa, bardzo "porządna" matematyka!)?

   Jeśli chcemy coś jeszcze usprawnić, to najpierw wypada zrozumieć jak działa aktualne rozwiązanie, wdrożyć nowe **matematyczne** idee i sprawdzić (pod kątem zastosowań w informatyce(!)). Może warto rozważyć naukę matematyki?

6. Klasyczną procedurą w informatyce jest stosowanie ciągów zdefiniowanych rekurencyjnie. Co prawda, to będzie omawiane poza podstawową analizą matematyczną (Analiza I), ale bez jej podstaw nie da się tego nauczyć, Do konstrukcji funkcji tworzących (podstawowa metoda dla rekurencji) potrzeba będzie zrozumienie m.in. pojęć: funkcja i jej własności, szereg potęgowy i jego promień zbieżności (do tego sporo kombinatoryki i algebry). W uzupełnieniu: skoro komputer prowadzi obliczenia na liczbach całkowitych, to proszę przemyśleć jak korzystamy przy jego pomocy z funkcji (np. w arkuszu kalkulacyjnym)? Jak oblicza wartości takich funkcji jak $f(x) = \sin x$, bo przecież od czasu do czasu z tego korzystamy, nieprawdaż?

7. Jeśli gdzieś napotkacie hasło "ten wielomian (lub inna funkcja) przybliża daną...", to podstawowe pytanie brzmi: co to znaczy "przybliża"? Musimy sprecyzować co jest "odległość pomiędzy funkcjami" (metryka), no i podać jak to można obliczyć (oszacować) oraz *dlaczego* jest więcej niż jedna taka metoda ("np. "zbieżność średniokwadratowa").

8. Metody numeryczne, to z punktu widzenia informatyki "matematyka stosowana" czy "metody matematyczne informatyki". Wszystkie metody całkowania numerycznego czy numerycznego rozwiązywania równań różniczkowych bazują na definicjach czy własnościach opartych na analizie matematycznej. Podobnie interpolacja czy aproksymacja numeryczna. Nie będę wchodził w szczegóły - pozostawię to na osobny przedmiot "Metody numeryczne" (proszę zabrać na niego notatki z analizy!).

9. Kolejny przykład: często słyszycie "błąd metody numerycznej", "ta metoda jest lepsza (dokładniejsza), (szybciej zbieżna)". Co się za tym kryje? Ogólnie: bardzo często będzie potrzebne rozwinięcie funkcji zgodnie ze wzorem Taylora (analiza!) i oszacowanie reszty (niekiedy w postaci całkowej...). Choć (niestety) studenci kojarzą takie zagadnienie jako "był taki wzór...".

10. Coś spoza wykładu z "Analizy I" (ale na jej podbudowie): analiza harmoniczna, szeregi i transformaty Fouriera itp. mają szerokie zastosowania w informatyce (np. algorytm Cooleya i Tuckeya FFT z zastosowaniami czy algorytm uczenia Kushilevitza i Mansoura - ponownie o szczegóły proszę pytać na właściwych przedmiotach informatycznych). Nie mówiąc już o teorii sygnałów... Por. też Daniel Stefanković, *F*ourier transforms in computer science, University of Chicago, Department of Computer Science, TR-2002-03.

Czy można podać więcej przykładów? A lepsze? TAK, ale jeśli kogoś to co napisano powyżej nie przekona (choć trochę), to szkoda czasu - sam się musi przekonać (po szkodzie...). Mam nadzieję, że krytycy dopiszą swoje przykłady - zawsze jest to punkt wyjścia do dyskusji. Co roku będę dodawał coś na wykładach - dla *zainteresowanych*...

©Mieczysław Cichoń
Wydział Matematyki i Informatyki UAM Poznań
Poznań 2018