László Lovász

Nový pohled na algoritmickou matematiku

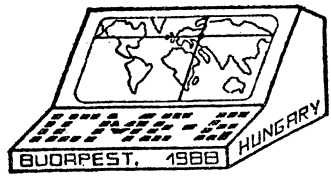
Pokroky matematiky, fyziky a astronomie, Vol. 35 (1990), No. 1, 12--22

Persistent URL: http://dml.cz/dmlcz/137994

[9] PRESTON, E.: *On Choosing a Technical Wordprocessor*, The Australian Math. Soc. Gazette *15* (1988), 31–34.
[10] PRESTON, E.: *User Support for TEX and Chiwriter*, The Australian Math. Soc. Gazette *15* (1988), 117-119.
[11] SEYBOLD, J. W.: *The Desktop-Publishing Phenomenon*, Byte (May 1987), 149.
[12] SPIVAK, M. D.: *PC TEX Manual*, Personal TEX, Inc. (1987, Third Edition).
[13] SPIVAK, M. D.: *The Joy of TEX: A Gourmet Guide to Typesetting with the AMS-TEX macro package*, Amer. Math. Soc. (1986).
[14] TATARKIEWICZ, J.: *TEXnika Ładnego Druku*, Komputer 8 (1988), 36–37.
[15] ULRYCH, O.: *AmSTEX za 59 minut*, (interní text), MÚ UK MFF UK (1988).
[16] VARIAN, H. R.: *PCTEX and MicroTEX*, BYTE (Apr. 1986), 267.
[17] VOGLER, T.: *Satzprogramm TEX*, CHIP (Nov. 1987), 260.
[18] WALLSTROM, T.: *The Equation Processor in Word 3.0*, Notices Amer.Math.Soc. *35* (1988), 263-265.
[19] WILF, H. S.: *TEX: A Non – review*, Amer. Math. Monthly *93* (1986), 309–315.
[20] PALAIS, R. S.: *Rubrika: Mathematical Text Processing*, Notices Amer. Math. Soc. *33* (1986), 3–7, 303–308, 741–751 a další.
[21] *AMS Electronic Manuscript Program; New Program Announced by the Society*, Notices Amer. Math. Soc. *33* (1986), 299–302.

Nový pohled na algoritmickou matematiku

László Lovász



Úvod

Rozvoj počítačů je asi jediný zásadní technický převrat tohoto století. Je přirozené, že se dotkl i tak úzce souvisejících odvětví vědy, jako je matematika a její výuka. Je také přirozené, že ve všech oborech, které se dostaly do styku s rozvojem počítačů, se začalo bouřlivě diskutovat. Diskutující měli nejrůznější názory: extrémní i umírněné, progresívní i konzervativní. Má algoritmická matematika větší cenu než klasická strukturně orientovaná matematika typu věta – důkaz? Nebo jenom zakrývá podstatu věcí tím, že je dělá komplikovanější, než je třeba? Vede výuka algoritmů k lepšímu pochopení

Překlad přednášky *Algorithmic Mathematics: An Old Aspect with a New Emphasis*, přednesené na plenárním zasedání ICME 6. Přeložila HELENA NEŠETŘILOVÁ.

Autor, L. LOVÁSZ (1947), je profesorem na Eötwösově univerzitě v Budapešti a na Princetonské univerzitě. Je autorem řady významných výsledků v matematice i computer science. Jeho hlavními obory jsou kombinatorika, teorie grafů a teorie složitosti.

podložené struktury nebo se toho spíše dosáhne abstraktnějším a elegantnějším výkladem? Je algoritmický způsob života nejlepší (Maurer 1985) nebo je aplikovaná matematika prostě špatná matematika (Halmos 1981)? Mají se počítače zavést do výuky matematiky na základních (středních) školách?

Nebudu se snažit odpovědět na všechny uvedené otázky. Mým cílem bude pokusit se ukázat, že algoritmická matematika (na kterou se díky počítačům soustředila pozornost, ale která existovala a měla význam i před rozvojem počítačů!) nepředstavuje antitezi ke klasické matematice typu „věta – důkaz". Naopak, algoritmická matematika obohacuje některá klasická odvětví o nový pohled, nové druhy problémů a nové způsoby, jak je řešit. Tedy: ne algoritmická matematika *nebo* strukturální matematika, ale algoritmická *a* strukturální matematika!

Vzájemné ovlivňování mezi algoritmickou a strukturální stránkou matematiky je rozmanité, zmíním se jen o dvou nejdůležitějších směrech. Návrhy i analýza algoritmů a studium algoritmické řešitelnosti na jedné straně používají stále podstatnějším způsobem nástroje klasické strukturální matematiky a na druhé straně algoritmický pohled má stále zásadnější vliv na mnoho odvětví klasické matematiky.

Dovolte mi několik příkladů. Zřejmě první případ, kdy byl pojem „algoritmu" definován natolik přesně, aby bylo možné ptát se po „algoritmické řešitelnosti", byl pojem geometrické konstrukce pomocí pravítka a kružítka, formulovaný řeckými geometry. Na tomto případu je z matematického hlediska zajímavé to, že existují konstrukční úlohy obojího typu, řešitelné i neřešitelné.

Návrhy konstrukčních algoritmů motivují geometrii už dlouhou dobu, přispěly i k rozvoji důležitých nástrojů (které jsou užitečné také nezávisle na konstrukčních úlohách), jako je inverze nebo zlatý řez. Důkaz *neřešitelnosti* základních konstrukčních úloh (trisekce úhlu, konstrukce čtverce, jehož plocha je rovna ploše daného kruhu, konstrukce krychle s dvojnásobným objemem, konstrukce pravidelného sedmiúhelníka atd.) však tento vliv ilustruje dramatičtěji. Takové negativní výsledky byly pro řeckou matematiku, a ještě dlouho po ní, nedostupné. Dokázat nebo jenom formulovat tyto úlohy tak přesně, aby byly přístupné matematickým metodám, vyžadovalo totiž pojem reálného čísla a podstatnou část moderní algebry. Moderní algebra byla vlastně inspirována snahou dokázat takové negativní výsledky; mimoto podstata toho, že nelze zkonstruovat jisté obrazce a neřešitelnost rovnic alespoň 5. stupně, je stejná.

Jako další příklad uvažme pojem prvočísla. Staří Řekové studovali také tato čísla a dokázali některé jejich základní vlastnosti. Také teorie prvočísel, krásná, ale velice těžká, byla během vývoje moderní matematiky jedním z hlavních směrů (i inspirací pro mnoho směrů dalších). Základní algoritmická úloha se však dostala do popředí zájmu až s rozvojem počítačů nebo spíše se vznikem teorie výpočetní složitosti. Jde o úlohu *zjistit, je-li dané číslo prvočíslo a v případě, že není, najít jeho prvočíselný rozklad.* (Už matematici v 18. a 19. století, zvláště Gauss, prováděli rozsáhlé výpočty týkající se prvočísel a nalezli vtipné triky, kterými si v práci pomáhali. Své algoritmy však zjevně nepovažovali za matematické výsledky.)

Ve výpočtech mají tyto otázky velmi důležité aplikace a jednoduché elementární postupy, jak je řešit, zdaleka nejsou uspokojivé (v praxi je třeba uvažovat čísla až s několika sty ciframi). Přibližně během posledních 10 let se k vyřešení těchto otázek použí-

vají stále náročnější číselně teoretické metody, pomocí kterých se navrhují stále důmyslnější a výkonnější algoritmy.

Všimněte si, že vzájemná interakce mezi matematikou a algoritmy se v těchto dvou příkladech liší. V prvním případě máme jistý pojem „algoritmu" (konstrukční postup) a chceme dokázat, že nestačí k řešení jistých úloh. Takové problémy mohou být velice těžké a jak ukazuje i náš příklad, jejich důkaz může vyžadovat netriviální matematiku nebo dokonce vznik zcela nových oborů. Teorie výpočtů je dnes plná nevyřešených problémů podobného druhu. Je k dispozici jen velmi málo metod, pomocí kterých lze dokazovat negativní výsledky o algoritmické řešitelnosti problémů, zvláště tam, kde jsou na algoritmus kladena různá omezení, například na spotřebu výpočetního času. Informatika*) je v takových případech „externím" uživatelem matematiky: dodává těžké problémy, které je třeba modelovat a řešit podobně jako těžké problémy v mechanice nebo astronomii.

Ve druhém příkladě proniká informatika do klasické matematiky tak, že klade staré otázky z nového zorného úhlu. Často tím způsobem, že požaduje *konstrukci* tam, kde klasická teorie dodala „čistý" existenční důkaz. Jindy se požaduje *efektivní* metoda tam, kde je k dispozici „teoreticky konečná" analýza všech případů (jako v úloze testování prvočíselnosti). Některým oborům (např. teorii grafů) vtiskl tento vývoj zcela nový rámec (viz Lovász 1986). Ve své přednášce bych se chtěl soustředit na vývoj v tomto směru.

Nový pohled na některé záležitosti v klasické matematice vyvolává jistě i výzvu k matematické výchově. Zavedení počítačů do výuky (na jakékoliv úrovni) je jen velmi částečné řešení. Chtěl bych uvést několik svých poznámek o tom, jak lze nahlížet na uvedené otázky. Jsem však přesvědčen, že bude třeba ještě hodně experimentální i teoretické práce, než budou hlavní rysy vhodného řešení jasné.

1. Nový pohled na některé staré výsledky

Aproximace iracionálních čísel pomocí čísel racionálních má dlouhou historii. Předpokládejme, že máme reálné číslo α, které chceme aproximovat racionálním číslem s malým jmenovatelem. První, triviální způsob je zaokrouhlit: vezmeme libovolné celé kladné číslo q a součin $q\alpha$ zaokrouhlíme na nejbližší celé číslo p. Potom p/q je přijatelnou racionální aproximací α, přesněji

$$\left|\alpha - \frac{p}{q}\right| \leqq \frac{1}{2q}.$$

Lze tento výsledek zlepšit v tom smyslu, že 2 nahradíme nějakým celým kladným číslem Q? Přesněji řečeno, můžeme pro dané reálné číslo α a celé kladné číslo Q nalézt racionální číslo p/q takové, že $|\alpha - p/q| \leqq 1/Qq$? Je zřejmé, že v tomto případě už nelze q zvolit libovolně; jaké však musí být?

Dirichletova věta tvrdí, že *pro každé dané reálné číslo α a celé kladné číslo Q lze*

*) V anglickém originálu „computer science" (pozn. překl.).

najít celá čísla p a q taková, že $0 < q \leqq Q$ *a* $|\alpha - p/q| \leqq 1/Qq$. Jinými slovy, že $|q\alpha - p| \leqq 1/Q$.

Toto fundamentální tvrzení má dva základní důkazy, které bych zde chtěl rozebrat, abych ilustroval rozdíl mezi algoritmickým a nealgoritmickým přistupem.

První důkaz: Uvažme $Q + 1$ čísel

$$0\alpha - [0\alpha], 1\alpha - [1\alpha], \ldots, Q\alpha - [Q\alpha].$$

Všechna tato čísla leží v intervalu $[0, 1)$, takže rozdíl mezi nějakými dvěma z nich, řekněme $k\alpha - [k\alpha]$ a $l\alpha - [l\alpha]$, $(0 \leqq k < l \leqq Q)$, je menší než $1/Q$. Nechť $q = l - k$ a $p = [l\alpha] - [k\alpha]$. Pak $q \leqq Q$ a

$$|q\alpha - p| = |(l - k)\alpha - ([l\alpha] - [k\alpha])| = |(k\alpha - [k\alpha]) - (l\alpha - [l\alpha])| < 1/Q.$$

Racionální číslo p/q tedy dokazuje Dirichletovu větu.

Druhý důkaz (nástin): Je dobře známo, že každé reálné číslo α lze vyjádřit ve tvaru *řetězového zlomku*

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ldots}}}.$$

Tento rozvoj může být konečný (je-li α racionální) nebo nekonečný (je-li α iracionální). Jako příklad uveďme rozvoj

$$\frac{11}{8} = 1 + \frac{3}{8} = 1 + \frac{1}{2 + \frac{2}{3}} = 1 + \cfrac{1}{2 + \cfrac{1}{1 + \frac{1}{2}}}$$

a

$$\sqrt{2} = 1 + (\sqrt{2} - 1) = 1 + \frac{1}{\sqrt{2} + 1} = 1 + \frac{1}{2 + (\sqrt{2} - 1)} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \ldots}}}.$$

Ukončíme-li rozvoj do tvaru řetězového zlomku v k-tém kroku, dostaneme racionální číslo

$$\frac{p_k}{q_k} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots + \cfrac{1}{a_k}}}.$$

Toto racionální číslo se nazývá *k-té přiblížení* α a lze o něm dokázat řadu základních tvrzení. Zde budeme potřebovat to, že p_k/q_k je velmi dobrou aproximací α,

$$\left|\alpha - \frac{p_k}{q_k}\right| \leqq \frac{1}{q_k Q} ,$$

tzn. že racionální číslo p_k/q_k vyhovuje podmínkám Dirichletovy věty.

Který z obou důkazů je „lepší"? První důkaz je nepochybně mnohem jednodušší; je nejenom kratší, ale nevyužívá ani žádná další tvrzení, zatímco druhý důkaz se opírá o tvrzení z teorie řetězových zlomků.

Ale dejme tomu, že chceme zmíněné racionální číslo také *najít*. Než začneme tuto otázku rozebírat, musíme si ujasnit, jak je zadáno α. Teď chceme dostat algoritmus, bude proto výhodnější předpokládat, že α je zadáno v nějakém konečném explicitním tvaru. Předpokládejme, že α je racionální, řekněme $\alpha = a/b$. Musíme přirozeně předpokládat, že $Q < b$, protože jinak by byla aproximace triviální.

Jaký algoritmus se dá odvodit z prvního důkazu? První důkaz říká, že je třeba utvořit všechna čísla $k\alpha - [k\alpha]$, nalézt dvě z nich, která jsou blízko sebe, ... Ale když už budeme chtít všechna taková čísla skutečně vytvořit, dá nám stejnou práci zjistit, které z nich je menší než $1/Q$. *První důkaz tedy neposkytne žádný netriviální algoritmus, pomocí kterého by bylo možné najít racionální číslo, jehož existenci zaručuje*; v tomto smyslu je to „čistý existenční důkaz".

(Lze namítnout, že strukturální vhled, který získáme z prvního důkazu, lze rozvinout dále tak, abychom získali efektivní algoritmus. To však vyžaduje hlubší pochopení a další práci.)

Je druhý důkaz z tohoto hlediska lepší? Kolik práce dá výpočet rozvoje do tvaru řetězového zlomku? Ukážeme, že získat tento rozvoj je nejenom jednoduché, ale také to, že na jeho použití v aproximační úloze není nic nepřirozeného.

Předpokládejme, že chceme najít dobrou racionální aproximaci nějakého čísla α. V prvém přiblížení bychom mohli použít racionální číslo $a_0/1$, kde $a_0 = [\alpha]$. Abychom získali lepší aproximaci, musíme aproximovat chybu $x_1 = \alpha - a_0$. Nahraďme tuto chybu převrácenou hodnotou a tu aproximujme celou částí $a_1 = [1/x_1]$. Vzhledem k tomu, že x_1 je číslo menší než 1, je tato myšlenka velmi přirozená. Druhá aproximace je zatížena chybou $x_2 = (1/x_1) - a_1$. Vezměme opět převrácenou hodnotu, atd. Celá kladná čísla $a_0, a_1, \ldots$, která tímto způsobem dostaneme, jsou právě koeficienty rozvoje α do tvaru řetězového zlomku.

Je-li $\alpha = a/b$, dostaneme a_0 tak, že a dělíme b se zbytkem, pak a_0 je podíl a je-li r zbytek, chyba $x_1 = r/b$. $1/x_1$ je tedy racionální číslo b/r, pro které zopakujeme celý postup prováděný v prvním kroku pro a/b. Je jasné, že tento postup má konečný počet kroků a rozvoj α do tvaru řetězového zlomku je tedy možný. Zbytek důkazu dává jednoduchý návod, jak najít racionální číslo, které je hledanou aproximací.

(Možná si někteří moji čtenáři v tomto místě všimli, že pro to, abychom získali rozvoj racionálního čísla do tvaru řetězového zlomku, provádíme zcela stejné aritmetické operace jako v eukleidovském algoritmu, který se používá k výpočtu největšího společného dělitele a a b.)

Druhý důkaz je tedy pouze analýza velmi přirozeného iterativního algoritmu k hledání lepších a lepších aproximací nějakého čísla.

Je však tento algoritmus lepší než triviální algoritmus, odvozený z prvního důkazu?

Odpověď zní: je mnohem lepší. Počet kroků v rozvoji a/b do tvaru řetězového zlomku (nebo ekvivalentně: počet kroků eukleidovského algoritmu při výpočtu největšího společného dělitele (a, b)) je úměrný počtu cifer a a b; počet kroků prvního, triviálního algoritmu je úměrný Q, které může být i tak velké jako b. Jsou-li a, b a Q stociferná čísla, potřebuje první algoritmus 10^{100} kroků, zatímco druhý algoritmus méně než 500.

Běžný způsob, jak měřit rychlost algoritmu, je srovnávat počet bitových operací s počtem bitů potřebných k zápisu vstupu. k-bitové celé číslo (číslo, které má v binárním zápisu k bitů) zvětšuje tedy délku vstupu o k. V úloze diofantické aproximace je délka vstupu rovna počtu bitů a, b a Q, což je v podstatě $\log_2 a + \log_2 b + \log_2 Q$. V tomto výpočtovém modelu ovlivňuje délka čísel také čas potřebný k provedení jedné aritmetické operace. Například násobení dvou k-bitových celých čísel metodou, která se vyučuje na školách, vyžaduje zhruba k^2 bitových operací, taková operace tedy prodlužuje dobu výpočtu o k^2.

Mezi algoritmy, pro které čas potřebný ke zpracování vstupních dat roste s délkou vstupu polynomiálně nebo rychleji, je podstatný rozdíl. Polynomiální algoritmy bývají po matematické stránce zajímavé a obvykle — i když ne vždy — také prakticky použitelné. Metoda „hrubé síly", která spočívá v uvážení všech možností, vede často k exponenciálně mnoha případům, které je třeba probrat (všechny podmnožiny dané množiny, atd.), a tedy také k exponenciálně rostoucím časovým nárokům. Algoritmus založený na prvním důkazu pracuje v exponenciálním čase, algoritmus založený na druhém důkazu je polynomiálně časově omezen.

Lze tedy říci:

První důkaz je existenční důkaz. Je elegantní a krátký, ale nedává algoritmus pro nalezení aproximace. „Dotáhnout" ho do efektivního algoritmu předpokládá nové nápady a další práci.

Druhý důkaz není nic jiného než analýza elegantního, přirozeného a efektivního algoritmu pro konstrukci dobré aproximace. Zjistit kvalitu aproximace však předpokládá další práci.

První důkaz lze snadno zobecnit pro případ simultánní aproximace několika čísel racionálními čísly se společným jmenovatelem. Pro tento případ lze několika způsoby zobecnit také algoritmický důkaz (řetězové zlomky). Žádné z těchto zobecnění však není tak elegantní a nedává tak dobrou aproximaci jako zobecnění prvního důkazu. Pro úlohu simultánní diofantické aproximace nebyl dosud nalezen algoritmus, který by v polynomiálně omezeném čase dokázal *najít* racionální čísla, která představují hledané aproximace a jejichž *existence* je zaručena poměrně jednoduchým zobecněním prvního důkazu.

2. Nahlédnutí do teorie složitosti

Většina z nás už se setkala s problémem formulovaným nějak takto: „Charakterizujte množiny (čísla, ...), které mají tu vlastnost, že ***" a také se studentem, který provokativně odpověděl: „Množina má vlastnost ***, právě když má vlastnost ***". Je taková

odpověď v pořádku? Co se stane, když bude student zastírat triviální tvrzení tím, že trochu přeformuluje vlastnost ***? Kdy začíná být jeho odpověď netriviální, a tedy přijatelná? Nebo jsou takové úvahy prostě nesmyslné?

Jeden z velkých úspěchů teorie výpočetních procesů*) spočívá v tom, že *je schopna* (alespoň pro velkou třídu struktur a vlastností) *matematicky přesně definovat, které charakterizace jsou „dobré" a které jsou víceméně jenom přeformulováním původní otázky.* V této přednášce není přirozeně dost prostoru na vybudování této teorie. Chtěl bych se však pokusit ilustrovat uvedenou myšlenku na jednom důležitém příkladu.

Uvažme soustavu nerovností

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\leqq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &\leqq b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\leqq b_m . \end{aligned} \tag{1}$$

Protože nás zajímají algoritmické aspekty, budeme předpokládat, že nerovnosti mají racionální koeficienty

Kdy má (1) řešení? Zkoumejme následující dvě odpovědi:

Věta A. *Soustava nerovností* (1) *má řešení, právě když soustava*

$$\begin{aligned} a_{11}(u_1 - v_1) + a_{12}(u_2 - v_2) + \ldots + a_{1n}(u_n - v_n) &\leqq b_1 \\ a_{21}(u_1 - v_1) + a_{22}(u_2 - v_2) + \ldots + a_{2n}(u_n - v_n) &\leqq b_2 \\ &\vdots \\ a_{m1}(u_1 - v_1) + a_{m2}(u_2 - v_2) + \ldots + a_{mn}(u_n - v_n) &\leqq b_m \end{aligned} \tag{2}$$

má nezáporné řešení.

Věta B. *Soustava nerovností* (1) *má řešení, právě když soustava*

$$\begin{aligned} a_{11}y_1 + a_{21}y_2 + \ldots + a_{m1}y_m &= 0 \\ a_{12}y_1 + a_{22}y_2 + \ldots + a_{m2}y_m &= 0 \\ &\vdots \\ a_{1n}y_1 + a_{2n}y_2 + \ldots + a_{mn}y_m &= 0 \\ b_1y_1 + b_2y_2 + \ldots + b_my_m &= -1 \end{aligned} \tag{3}$$

nemá nezáporné řešení.

(Všimněte si, že na řešení (3) se lze dívat jako na lineární kombinaci nerovností v (1) s nezápornými koeficienty (y_i), která dává triviálně neřešitelnou nerovnost $0x_1 + \ldots + 0x_n \leqq -1$.)

Obě věty vyjadřují nutnou a postačující podmínku pro řešitelnost (1). Věta A je však v podstatě jednoduchý trik, jak ukázat, že řešitelnost lze snadno převést na nezápornou řešitelnost, zatímco věta B představuje důležitý výsledek (nazývaný Farkasovo lemma). V čem je rozdíl?

Představte si, že jako příklad soustavy řešitelných lineárních nerovností chci v této přednášce použít konkrétní případ (1). Jak vás mohu přesvědčit, že soustava je skutečně řešitelná? Jednoduše, prostě vám řešení ukážu. Dále si představte, že chci použít jiný

*) V originálu „theory of computing" (pozn. překl.).

konkrétní případ (1), abych ilustroval neřešitelnou soustavu. Jak vás mohu přesvědčit o tom, že je skutečně neřešitelná ? Tím, že vyzkouším všechny možné hodnoty pro všechny proměnné ? Tady žádný jednoduchý způsob neexistuje.

Pomohou mi nutné a postačující podmínky formulované ve větách A a B ? Použiji-li větu A, musím ukázat, že (2) *nemá* nezáporné řešení, to však není o nic jednodušší než původní úloha. Použiji-li však větu B, stačí dokázat, že (3) *má* nezáporné řešení a to mohu udělat tak, že vám jedno řešení ukážu. Podmínka věty B má tedy úplně jinou logickou strukturu než původní vlastnost a než podmínka uvedená ve větě A.

Je tedy patrné, že velká část teorie grafů, optimalizace, teorie čísel, atd. má podobnou strukturu. Důležité vlastnosti grafů, čísel, ... jsou takové, že má-li graf, číslo, ... tuto vlastnost, existuje jednoduchý způsob, jak to dokázat (např. složená čísla, čtyřbarevné grafy, ...). Podstata je v tom, že tyto vlastnosti jsou definovány *existencí* určitého objektu (číslo je složené, jestliže má vlastního dělitele, graf je čtyřbarevný, jestliže má dobré obarvení čtyřmi barvami, ...). Budu-li vás chtít přesvědčit, že číslo

$$617532176011725742711064069114397791170668109866281$$

je složené, stačí, když vám ukážu dělitele

$$7858321551080267055879091\,.$$

(Musíte si samozřejmě ještě ověřit, že je to skutečně dělitel, ale k tomu, jak víme, stačí polynomiálně omezený čas.)

Takové vlastnosti se nazývají NP-*vlastnosti*; byly pojmenovány podle technické definice, ve které se hovoří o Nedeterministických Turingových strojích s Polynomiálně omezeným časem; přesnou definici tu není třeba uvádět. Negace NP-vlastnosti je někdy (ale ne vždy!) zase NP-vlastnost. Věty, které mají tvar takových ekvivalencí, často patří k nejdůležitějším výsledkům v příslušném oboru (jako například výše uvedené Farkasovo lemma). Někdy se o nich mluví jako o *dobrých charakterizacích.*

Tato klasifikace vlastností úzce souvisí také s algoritmy. Většina vlastností, pro které bychom chtěli najít algoritmus s polynomiálně omezeným časem, abychom je mohli rozhodnout, patří přirozeným způsobem mezi NP-vlastnosti. Je-li vlastnost rozhodnutelná polynomiálním algoritmem, pak zároveň tato vlastnost i její negace jsou NP-vlastnosti, tzn. že mají dobrou charakterizaci. Opačné tvrzení platit nemusí: není známo, je-li každá dobře charakterizovaná vlastnost rozhodnutelná v polynomiálně omezeném čase (pravděpodobně ne, ale jak už jsem uvedl, v této oblasti nejsou k dispozici prostředky, jak takové negativní výsledky dokázat). Nalézt dobrou charakterizaci však obvykle znamená důležitý krok ke konstrukci polynomiálního algoritmu. Tuto situaci dobře ilustruje příklad Farkasova lemmatu: polynomiální algoritmus k rozhodování, má-li (1) řešení, byl nalezen téměř sto let po důkazu lemmatu (Čhačijan 1978, známá elipsoidová metoda).

V tomto článku není samozřejmě možné zabývat se teorií algoritmů podrobněji. Stručně jsem se zmínil jenom o tom, co jsem potřeboval, abych zdůvodnil svůj názor na to, jak bude asi vypadat nový rámec mnoha matematických disciplín. (Pro úvod do teorie algoritmů viz např. Sedgewick 1983.)

3. Co z toho vyplývá pro výuku matematiky?

Ať už z toho vyplývá cokoliv, mělo by se k tomu přistupovat velmi uvážlivě a umírněně. Myslím si, že výuka matematiky by se měla (alespoň do jisté míry) orientovat tím směrem, kterým se ubírá matematický výzkum. Zvláště v těch (vzácných) případech, kdy se od základů mění celý rámec oboru. Algoritmická matematika k nim patří. Rozsah uplatnění algoritmického pohledu v klasické matematice však zatím vůbec není jasný a podstatně se liší případ od případu (a také od jednoho přednášejícího k druhému). Například teorie grafů a optimalizace byly z hlediska výpočetní složitosti celé důkladně přepracovány; teorie čísel a části algebry se právě z tohoto hlediska studují, mnoho základních otázek je však zatím bez odpovědi; v analýze a diferenciálních rovnicích může, ale nemusí, mít tento přístup úspěch, a zdá se, že teorie množin nemá s algoritmy mnoho společného.

Zkušenost s „Novou matematikou", úpravou množinově teoretických základů matematiky pro nižší stupeň školní výuky, nás varuje, že příliš násilné změny mohou vést ke špatným koncům, a to i v takovém případě, že je nový systém dobře zavedený ve výzkumu i ve vysokoškolské výuce. Dovolte mi proto jenom naznačit některé myšlenky o výuce algoritmů, i když — zdůrazňuji — před jakýmkoliv pokusem o jejich zavedení v širokém měřítku je nutné je důkladně prodiskutovat a vyzkoušet.

V zásadě by bylo možné vyučovat některé algoritmy a jejich analýzu přibližně ve stejné době, kdy se ve výuce poprvé objeví věty a jejich důkazy, tedy asi ve 14 letech. Jsou samozřejmě jisté algoritmy (pro násobení, dělení, atd.), které se v učebních osnovách objevují velice brzy. To jsou však spíše návody než algoritmy, jejich správnost se přirozeně nedokazuje, ani se nezkoumá, jak jsou efektivní. Žáci se musí naučit (a procvičit si), jak tyto jednoduché algoritmy provádět. Je to stejné jako vyučovat věty (axiómy) bez důkazu nebo vyučovat empirické vědecké poznatky bez experimentů: je to sice nutné, ale nevede to k opravdu hlubokému pochopení.

Podle mého názoru začíná výuka „algoritmiky" tam, kde se studenti učí algoritmy navrhovat a ne je provádět. (Tato myšlenka je rozvedena např. v Maurer 1984.) Například eukleidovský algoritmus je takový, že ho mohou „objevit" sami studenti. Tak jako dnes existují rozsáhlé sbírky úloh a cvičení na algebraické identity, geometrické konstrukce a elementární geometrické důkazy, vznikne časem sbírka „úloh na návrhy algoritmů". Kromě konkrétních algoritmů by se studenti měli seznámit také se základními pojmy teorie algoritmů: vstup — výstup, správnost a důkaz správnosti, analýza časové a paměťové náročnosti, dobrá charakterizace, která je důsledkem algoritmu, algoritmus založený na dobré charakterizaci, atd.

Některé typy úloh na návrhy algoritmů, které by mohly přicházet v úvahu už na úrovni střední školy: enumerační problémy, pro které neexistuje předpis v kompaktním tvaru, jednoduché optimalizační problémy v teorii grafů (např. maximální nezávislé množiny stromů, nejkratší cesty, vytvoření seznamu všech klik, kružnic, atd.), třídění a vyhledávání, jednoduché (i když neefektivní) metody k ověřování prvočíselnosti, rozkladu na prvočísla a mnoho jiných úloh z teorie čísel, Gaussova eliminační metoda a další úlohy z lineární algebry, konvexní obal a další jednoduché geometrické konstrukce v rovině.

Přechod k algoritmičtěji orientovanému výkladu na vysoké škole by pak měl být

snazší a rychlejší. Už v současné době se na řadě vysokých škol vyučují některé předměty (např. teorie grafů) v podstatě algoritmicky. Součástí většiny kursů z teorie grafů jsou už standardně algoritmy pro hledání minimální kostry grafu, určení maximálního toku v síti nebo maximálního párování. Vzhledem k tomu, že teorie výpočetní složitosti sjednocuje pohled na mnoho základních úloh v teorii grafů, je to zcela přirozené. V ostatních oborech je v současné době situace trochu jiná, i když některé úlohy, jako je ověřování prvočíselnosti nebo kryptografické protokoly, představují hezké aplikace pro velkou část klasické teorie čísel.

4. Počítače a algoritmy

Chtěl bych se nyní zamyslit nad použitím počítačů při výuce uvedených témat. Je přitom třeba rozlišovat mezi algoritmem a jeho implementací ve formě počítačového programu. Samotný algoritmus je matematický objekt; program závisí na počítači a/nebo na programovacím jazyce. Jistě, je třeba, aby studenti poznali cestu od algoritmu k programu, který lze spustit na počítači, ale není nutné, aby všechny algoritmy, o kterých se učí nebo které navrhnou, byly implementovány. Tato situace je (zase) obdobná jako v případě geometrických konstrukcí pomocí pravítka a kružítka: student musí nějaké konstrukce skutečně narýsovat, ale u dalších může stačit, když uvede matematické řešení (protože smyslem není naučit ho rýsovat, ale dát mu možnost aplikovat různé geometrické pojmy a výsledky).

Tady bych chtěl varovat před nedostatky algoritmického jazyka. Neexistuje (dokonce ani v odborné literatuře) jednotný způsob zápisu algoritmů. Pokud je mi známo, jsou v tomto smyslu středoškolské učebnice informatiky ještě méně jednotné a často dokonce výstřednější než odborná literatura. Praxe se pohybuje od zcela neformálního popisu až k programům v konkrétním programovacím jazyce. Obě řešení lze podpořit důvody, které mluví v jejich prospěch, já osobně se však přikláním k neformálnímu popisu, protože si myslím, že podrobnosti vlastní implementace často zakryjí matematickou podstatu věci. Chtěl bych to ilustrovat dvěma příklady.

Algoritmus může například obsahovat krok „Vyberte libovolný prvek množiny S". V implementaci je však nutné specifikovat, který prvek se vybírá. Tento krok se tedy musí změnit na něco jako „Vyberte první prvek množiny S". Může však existovat jiný algoritmus, pro který je podstatné, že vybíráme první prvek. Převedeme-li oba algoritmy do programů, zakryjeme tím tento důležitý rozdíl. (Může se také stát, že je výhodné vybrat *poslední* prvek S. Při neformálním popisu necháváme možnost volby otevřenou, převedeme-li však algoritmus do programu, tuto možnost ztratíme.)

Než uvedu druhý příklad, připomenu, že Fibonacciova čísla jsou definována rekurentním vztahem

$$F_{k+1} = F_k + F_{k-1}$$

(a $F_0 = 0$, $F_1 = 1$). Tento rekurentní vztah vyjadřuje jednoduchý algoritmus, jak počítat Fibonacciova čísla. Přepíšeme-li ho do programu, bude vypadat asi takto:

je-li F aktuální Fibonacciovo číslo a G číslo předchozí, pak (s použitím pomocné proměnné H)

$$H := F, \quad F := G + F, \quad G := H.$$

Je tedy patrné, že program obsahuje řadu podrobností, které k metodě výpočtu Fibonacciových čísel *matematicky* nepatří: program ukládá F_{k+1} do stejného registru, kde bylo předtím uloženo F_k, aby to však mohl udělat, musí uchovat F_k, protože bude jeho hodnotu v příštím kroku potřebovat, ale k tomu je třeba použít „pomocnou proměnnou", atd.*)

Abychom však ukázali také druhou stranu mince: hlavní problém neformálního popisu algoritmů je v tom, že je obtížné definovat „časovou náročnost" nebo „počet kroků" (už jsme na to narazili výše). To vše bohužel závisí na podrobnostech implementace (a to až pod úroveň programovacího jazyka; závisí to na použité reprezentaci dat a na datové struktuře). Někdy to lze vyřešit tak, že přesně uvedeme, které kroky se započítávají (např. porovnání v třídicím algoritmu nebo aritmetické operace v algebraickém algoritmu). To je však „podvod" v tom smyslu, že nezapočítáváme čas potřebný k manipulaci s daty a ten může být tak dlouhý nebo dokonce delší než čas potřebný k provedení „matematicky podstatných" kroků. Měl bych ještě dodat, že polynomialita algorimu je „robustní", tzn., že nezávisí na implementaci (přestože různé implementace mohou být časově omezeny různými polynomy).

Cesta od matematické představy algoritmu k počítačovému programu je dlouhá. Začíná pečlivým návrhem algoritmu a vede přes analýzu časové a paměťové spotřeby a jejich zlepšování, výběr datových struktur (který může být po matematické stránce velmi náročný) až k programování. Pro vysokoškolské studenty je velmi poučné projít celou cestu. Ale i ve středoškolské výuce by se mělo rozlišovat alespoň mezi matematickou stránkou algoritmu a jeho implementací.

Před pedagogy, kteří vyučují matematiku na vysokých a středních školách, stojí důležitý úkol: vypracovat v blízké budoucnosti jednotný a adekvátní způsob popisu a analýzy algoritmů. Způsob, který ukáže matematické pozadí návrhu algoritmu, který umožní jeho analýzu a který bude stručný a elegantní. Velmi by to pomohlo překonat pohrdání, které v současné době cítí k algoritmům obě strany, tj. učitelé i studenti.

Literatura

P. R. Halmos (1981): *Applied mathematics is bad mathematics.* V *Mathematics Tomorrow* (ed. L. A. Steen), Springer, 9—20.

L. Lovász (1986): *An Algorithmic Theory of Numbers, Graphs, and Convexity.* CBMS NSF Reg. Conf. Series in Appl. Math. *50*; SIAM.

S. Maurer (1984): *Two meanings of algorithmic mathematics.* Mathematics Teacher, 430—435.

S. Maurer (1985): *The algorithmic way of life is best; reflexions by R. G. Douglas, B. Korte, P. Hilton, P. Renz, C. Smorynski, J. M. Hammersley and P. R. Halmos.* College Math. Journal *16*, 2—18.

R. Sedgewick (1983): *Algorithms.* Addison—Wesley.

*) Děkuji Jacku Edmondsovi za oba příklady i komentář k nim a lituji, že jsem je podrobněji nerozebral.