MAGAZÍN
KATEDRY INFORMATIKY

číslo 1 | duben 2014 Univerzita Palackého v Olomouci

Úvodní slovo

Milí čtenáři,

dostává se Vám do rukou první číslo Magazínu katedry informatiky. Rádi bychom jeho prostřednictvím oslovovali absolventy a studenty katedry, ale i širší veřejnost a informovali je o dění na katedře, na univerzitě i o zajímavostech ze světa informatiky. Magazín bude vycházet dvakrát do roka a o jeho redakci se budou starat Petr Krajča a Petr Osička, které bych rád představil. Petr Krajča získal po absolvování naší katedry doktorát na State University of New York a odborně se zabývá zejména návrhem programovacích jazyků a překladačů, dále pak návrhem rychlých algoritmů pro binární data. Petr Osička absolvoval magisterské i doktorské studium na naší katedře a odborně se zabývá analýzou diskrétních dat a složitostí algoritmů.

V tomto čísle nabízíme krátké ohlédnutí za vývojem katedry, počtení o jednom našem výzkumném tématu, vyprávění o průkopníku informatiky Donaldu Knuthovi a jeden zajímavý problém pro pobavení.

Přeji vám příjemné čtení.

Radim Bělohlávek

vedoucí Katedry informatiky Univerzity Palackého v Olomouci

Obsah

Ohlédnutí za katedrou

Petr Krajča

Určitě si mnozí kladete otázku, co všechno se za ta léta, co jste opustili školu, změnilo. Stalo se toho opravdu hodně, proto nám dovolte vyjmenovat alespoň ty nejdůležitější změny.

Nejviditelnější změnou je přestěhování katedry z Hejčína do nové budovy Přírodovědecké fakulty v areálu na Envelopě. Katedra se v posledních letech také znatelně rozrostla co do počtu studentů a z tohoto pohledu teď patří mezi největší katedry na Přírodovědecké fakultě. V současné době máme přes pět set studentů, z toho 114 studuje informatiku, 216 aplikovanou informatiku a 185 studentů se rozhodlo studovat aplikovanou informatiku kombinovanou formou. Vedle toho má katedra ještě devět studentů v doktorském programu.

Před dvěma lety také na katedře vznikla dvě specializovaná výzkumná centra, DAMOL (Data Analysis and Modeling Lab) a Mezinárodní centrum pro informaci a neurčitost, což je společný projekt s Katedrou optiky. Tato centra umožnila ukotvit katedru na mezinárodní scéně. V posledních letech tak katedru navštívila více než třicítka předních vědců v oboru zpracování dat a zároveň lidé z katedry získali možnost vycestovat na řadu prestižních zahraničních pracovišť. Mezi úspěchy, které si naše katedra v poslední době připsala, patří i publikace ve vědeckých časopisech. Podle naší nedávné analýzy dosáhla naše katedra v hodnocení výsledků publikovaných v předních (tzv. impaktových) časopisech, které se provádí podle vládní metodiky, počet bodů srovnatelný s počty dosahovanými na jiných vysokých školách fakultami popř. jednotkami sestávajícími z několika kateder.

Nová budova Přírodovědecké fakulty UP. Katedra informatiky sídlí v pátém patře.

Podobnostní databáze

Petr Krajča

Relační databáze se staly téměř standardem pro ukládání a zpracování dat a svou úlohu plní bezchybně na řadě míst od malých webových aplikací o pár tabulkách až po velké systémy zajišťující bezchybný chod bank s miliardami transakcí za den. Z tohoto pohledu se zdá být vše v pořádku. Přesto je tu jeden problém, který je přirozený pro drtivou většinu počítačových systémů: „berou vše doslova“.

Dejme tomu, že máte autobazar, který má v nabídce automobily z Tabulky 1. Pokud jako zákazník budete hledat auto za 200 tisíc, databáze vám odpoví, že v tabulce žádné takové auto není, respektive tradiční databázový systém není schopen poznat, že je tam auto nabízené za baťovskou cenu 199 tisíc, které by pro vás mohlo být zajímavé. Z pohledu člověka 199 a 200 tisíc není příliš velký rozdíl, z pohledu počítače je však tento rozdíl markantní. Jedním z možných řešení by bylo nechat vyhledávat automobily, které stojí méně než 200 tisíc, ale to nám bude vracet i třeba evidentně nevhodné vraky za 10 tisíc nebo na druhou stranu to bude ignorovat vozy za něco málo přes 200 tisíc, které by pro vás mohly být také přijatelné. Ideální by bylo mít možnost hledat automobily, které stojí *přibližně* 200 tisíc a nechat si je setřídit podle toho, jak moc se cena blíží naší představě. Na katedře informatiky se věnujeme tzv. podobnostním databázím, které právě toto umožňují.

Naše řešení vychází z toho, že místo klasické dvouhodnotové logiky využíváme fuzzy logiku. Zjednodušeně řečeno, zatímco klasická Booleova logika nám umožňuje říct pouze, zda řádek v tabulce odpovídá, nebo neodpovídá našemu dotazu, fuzzy logika nám umožňuje říct, že řádek odpovídá našemu dotazu pouze částečně, případně určit míru, jak moc naše kritéria splňuje.

Tabulka 1: Nabídka fiktivního autobazaru

	model	cena	karoserie	rok
1,00	BMW X5	245.000	SUV	2004
1,00	Ford Fiesta	230.000	Kombi	2011
1,00	Ford Focus	199.000	Hatchback	2011
1,00	Honda Accord	212.000	Kombi	2010
1,00	Hyundai i30	234.000	Hatchback	2010
1,00	Fiat Uno	10.000	Hatchback	1993

Jak to tedy funguje

Náš databázový model pracuje s určitou škálou pravdivostních hodnot, mohou to být například reálná čísla z intervalu od 0 do 1. Kdy 0 udává nepravdu (např. řádek v tabulce neodpovídá dotazu) a 1 udává pravdu (např. řádek v tabulce plně odpovídá dotazu). Cokoliv uvnitř tohoto intervalu pak představuje částečnou pravdu, např. pokud řádek v tabulce odpovídá dotazu pouze částečně. Pro tyto pravdivostní hodnoty máme celou řadu logických spojek, jednou z nejdůležitějších je $\otimes$, představující konjunkci. Tato spojka může být zavedena různými způsoby, např. jako násobení $a \otimes b = a \cdot b$ nebo jako minimum $a \otimes b = \min(a, b)$. Volba konkrétní spojky je plně v režii uživatele databáze a ten tak může ovlivňovat výsledky dotazů.

Dále pro každý datový typ (tzv. doménu) zavádíme nový operátor $\approx$, který slouží k vyjádření toho, jak jsou si dvě hodnoty podobné. Tento operátor musí splňovat dvě přirozené podmínky:

(1) $(u \approx u) = 1$, tedy každá hodnota je podobná sama sobě,

(2) $(u \approx v) = (v \approx u)$, tedy nezáleží v jakém pořadí jsou hodnoty porovnávány.

Například pro podobnost cen automobilů můžeme mít operátor $\approx_{cena}$ vyjádřený pomocí rovnice:

$$u \approx_{cena} v = \max\left(1 - \frac{|u - v|}{50000}, 0\right)$$

Pokud bychom si vzali ceny 200.000 a 199.000, zmíněné v úvodu článku, dostáváme hodnotu 0,98, to nám říká, že hodnoty si sice nejsou rovny, ale jsou si velmi podobné. Případně, pro menší data, jako je v našem příkladě typ karoserie, můžeme podobnost hodnot vyjádřit tabulkou:

$\approx_{karoserie}$	Hatchb.	Kombi	SUV
Hatchb.	1	.5	
Kombi	.5	1	.7
SUV		.7	1

Posledním dílem pomyslné skládačky jsou tzv. tabulky s ranky. Každý řádek této tabulky má kromě sloupců s daty, ještě tzv. *rank*, který udává stupeň do jaké míry daný řádek tabulky odpovídá dotazu. Tabulky s ranky se používají jak pro reprezentaci výsledků dotazů, tak pro uložená data. V takovém případě má řádek většinou rank roven 1. Tabulky 1 a 2 jsou příkladem tabulek s ranky, rank je v tomto případě zobrazen jako první sloupec.

Zpracování dotazu

Dejme tomu, že chceme z Tabulky 1 vybrat automobily stojící přibližně 200 tisíc. V takovém případě pro každý řádek v tabulce určíme hodnotu $(cena \approx_{cena} 220000)$ a s pomocí operátoru $\otimes$ a ranku daného řádku získáme nový rank, který nám

Tabulka 2: Nabídka automobilů, které stojí *přibližně 200 tisíc*

	model	cena	karoserie	rok
0,98	Ford Fiesta	199.000	Kombi	2011
0,76	Hyundai i30	212.000	Hatchback	2010
0,40	Ford Focus	230.000	Hatchback	2011
0,32	Honda Accord	234.000	Kombi	2010
0,10	BMW X5	245.000	SUV	2004

Tabulka 3: Nabídka automobilů, které stojí *přibližně 200 tisíc* a jsou *něco jako hatchback*

	model	cena	karoserie	rok
0,76	Hyundai i30	212.000	Hatchback	2010
0,49	Ford Fiesta	199.000	Kombi	2011
0,40	Ford Focus	230.000	Hatchback	2011
0,16	Honda Accord	234.000	Kombi	2010

udává, jak moc daný řádek odpovídá našemu dotazu. Výsledek dotazu můžete vidět v Tabulce 2.

Nemusíme se pouze omezovat na jedno kritérium, na výsledek předchozího dotazu (Tabulku 2) můžeme aplikovat další podmínku, třeba, že máme zájem o *hatchback*. V takovém případě opět projdeme celou tabulku a pro každý řádek určíme hodnotu $(karoserie \approx_{karoserie}$ *Hatchback*$)$, výsledek takového dotazu můžete vidět v Tabulce 3. Všimněte si, jak jsou uspořádány výsledky. Hyundai i30 je sice dražší, ale je to Hatchback, Ford Fiesta je sice „kombík“, ale stojí 199 tisíc, takže by o něj zakazník mohl mít zájem. Povšimněte si také, že z výsledků vypadlo SUV, protože nevyhovuje ani částečne jednomu ze zvolených kritérií a má tedy rank 0.

Databázový model, se kterým pracujeme, má pevné teoretické základy a je zobecněným Coddova databázového modelu, který je základem relačních databází, a stejně jako v těchto databázích máme k dispozici celou řadu operací, jako jsou například projekce, průnik, sjednocení, dělení nebo spojení. To umožňuje formulovat další a složitější typy dotazů. V případě našeho ukázkového autobazaru můžeme mít například jednu tabulku s nabízenými automobily, jednu se zákazníky a jejich požadavky a spojením těchto tabulek získáme, jakému zakazníkovi bude vyhovovat který automobil.

Podobnostní databázi, jak jsme ji zde nastínili, je možné prakticky implementovat jako rozšíření běžné databáze, dá se totiž vystačit s běžnými uloženými procedurami. Avšak není to úplně ideální řešení, protože taková databáze není schopna provést řadu optimalizací, zejména efektivně zpracovávat dotazy typu: „vrať deset nejlepších výsledků“, což je docela častá operace, která umožňuje odfiltrovat nezajímavé výsledky. Z tohoto důvodu jsme začali vyvíjet úplně nový databázový systém RESIQL, na kterém jsou zkoušeny nové algoritmy pro zpracování dotazů, nový dotazovací jazyk pro pohodlnou práci s podobnostmi a řada dalších vlastností podobnostních databází. Máme v plánu, že časem bude tento systém k dispozici i pro použití mimo univerzitní půdu.

Podobnostní databáze na katedře informatiky

- Přístup popsaný v článku u nás na katedře vznikl. Vilém Vychodil získal na výzkum v oblasti podobnostních databázi již dva granty Grantové Agentury ČR.
- Příležitost zapojit se do práce na podobnostních databazích získali formou závěrečných prací i studenti doktorského, magisterského i bakalářského studia.

Donald Knuth

Petr Osička

Donald Knuth je jedinečnou osobností spojující řemeslnou znalost programování a precizní přístup k analýze algoritmů. Patří mezi nejvýznamnější informatiky současnosti. Je znám především jako autor mnohasvazkového The Art of Computer programming *(TAOCP), knihy, kterou sám považuje za své životní dílo. Všechny jeho práce jsou charateristické svou precizností, obrovským citem pro detail a velmi často také autorovým jemným humorem. Mimo TAOCP se Donald Knuth během své kariéry věnoval mnoha oblastem informatiky. Významně přispěl k počítačové typografii, je autorem typografického systému TeX, v současnosti standardu v akademickém nakladatelském průmyslu. Aktivně se zajímá také o programování, pravidelně píše několik menších programů týdně. Je zastáncem takzvaného literárního programování. Donald Knuth je také autorem mnoha původních vědeckých prací, známý je například jím navržený algoritmus pro rychlé vyhledávání řetezců.*

Donald Knuth se narodil 10. ledna 1938 v Milwaukee ve Spojených státech amerických. Jeho otec byl učitelem na střední škole a majitelem malé tiskárny. Během střední školy se Donald Knuth zajímal o fyziku, matematiku a hudbu. Původně chtěl na vysoké škole studovat hudbu, nakonec se ale rozhodl pro fyziku, na jejíž studium získal stipendium. Přihlásil se na Case Institute of Technology v Clevelandu. S počítačem se poprvé setkal během léta po prvním ročníku. Při brigádě ve statistické laboratoři měl přístup k počítači IBM 650. Ten ho zaujal natolik, že se na něm rychle naučil programovat, a psaním programů pro něj strávil většinu zbytku studia. Ve druhém ročníku přešel z fyziky na matematiku. Traduje se, že k tomuto rozhodnutí ho inspirovala skutečnost, že při čekání na autobusové zastávce velmi snadno vyřešil matematický problém tak obtížný, že se za jeho vyřešení udělovala bez dalších otázek jednička z matematického kurzu. Během zbytku studia Donald Knuth prokázal tak velké matematické nadání, že mu po dokončení bakalářského vzdělání univerzita rovnou udělila také magisterský titul.

Umění programování

Kniha The Art of Computer Programming byla původně plánována jako pojednání o překladačích. Donald Knuth si totiž během doktorského studia matematiky na California Institute of Technology přivydělával psaním překladačů pro různé počítače. Zajímavostí je, že si přitom vydělal více než jeho profesoři na univerzitě. V roce 1962, během druhého roku postgraduálního studia, ho o napsání knihy o překladačích požádalo nakladatelství Addison-Wesley.

Donald Knuth začal na knize pracovat v roce 1963, až poté, co ukončil doktorské studium. Při psaní musel přečíst mnoho technických článků a byl znepokojen jejich kvalitou a především svou neorganizovaností. Rozhodl se, že je potřeba knihy, která by přehledně prezentovala všechny dosavadní důležité poznatky v informatice. Když v roce 1965 dokončil první draft dvanácti kapitol, obsahovala kniha už 3000 ručně psaných stran. Knuth nejdříve píše rukou a teprv poté vkládá obsah do počítače, používá editor Emacs.

Nakladatel se rozhodl knihu rozdělit do 7 svazků po dvou kapitolách. První svazek pojednávající o základních pojmech a o datových strukturách vyšel v roce 1968. Další dva svazky, obsahující kapitoly o náhodných číslech, aritmetice, vyhledávání a třídění vyšly o 5 let později. První část čtvrtého svazku, pojednávající o kombinatorických algoritmech – Knuth jej totiž pro obrovské množství materiálu musel rozdělit na dvě poloviny – vyšla v celku až za 30 let. V současnosti Donald Knuth pracuje na druhé polovině čtvrtého svazku. Nechal se slyšet, že by považoval za úspěch, kdyby se mu povedl dokončit pátý svazek.

Knihy měly na informatiku obrovský vliv. Donald Knuth přistupoval k algoritmům a jejich analýze s matematickou precizností, jaká nebyla v tehdejší době obvyklá. Vlastnosti algoritmů, například jejich efektivita důležitá nejenom pro porovnávání algoritmů a korektnost, nebyly již od té doby zkoumány pouze experimentálně, ale často i pomocí matematických technik, které Knuth shromáždil a prezentoval ve své knize. Donalda Knutha tak můžeme bez přehánění označit za otce analýzy algoritmů.

Typografie a TEX

Při přípravě druhého vydání prvních dvou svazků The Art of Computer Programming přešlo nakladatelství Addison-Wesley z technologie mechanického sázení knih na technologii digitální. Knihy vysázené tímto novým systémem ani zdaleka nesplňovaly Knuthovy estetické požadavky. Knuth hnán svým perfekcionismem se s nekvalitní sazbou nesmířil a v roce 1977 začal vyvíjet nový počítačový typografický systém. S pomocí svých studentů navrhl a implementoval TEX. Tento se skládal z tří hlavních částí. Samotného sázecího jádra, jehož odpovědností bylo vypočítat optimální rozložení obsahu v dokumentu, systému pro návrh fontů METAFONT a rodiny fontů Computer Modern vytvořené právě METAFONTem. Knuth původně plánoval, že systém napro-

gramuje za 1 rok. Ale podobně jako v případě TAOCP mu jeho tvorba zabrala o mnoho víc času, celých 10 let.

Díky svým kvalitám i tomu, že Knuth celý balík uvolnil pro veřejnost, časem vznikla kolem TEXu široká komunita, která vytvořila mnoho rozšíření. Jedním z nich je LATEX, dnešní de facto standard pro akademické publikace. V současnosti pro TEX a LATEX existuje obrovské množství balíčků určených pro sázení velkého množství různých druhů dokumentů.

Literární programování

Knuth implementoval první verzi TEXu v jazyce SAIL (Stanford Artificial Intelligence Language). Poté se jej rozhold přepsat do Pascalu, aby tak byl lépe přenositelný. Pro tvorbu dokumentace používal systém, který si sám napsal. Komentáře psal jako zdrojové soubory zpracovávané samotným TEXem. Postupem času se tento systém vyvinul ve WEB. Ve WEBu píše programátor zdrojový kód jako kombinaci programovacího jazyka a dokumentu pro TEX tak, aby byl kód dobře čitelný pro ostatní programátory. Jednotlivé části kódu lze rozčlenit a uspořádat způsobem usnadňujícím pochopení kódu lidskému čtenáři, nikoliv tak, jak to vyžaduje programovací jazyk. Současně je ale možné jednotlivé části provázat tak, aby bylo možné kód zpracovat a přeložit překladačem daného programovacího jazyka. Sám Knuth vydal zdrojové kódy TeXu přepsané v systému WEB jako knihu. Původně byl WEB vyvinut jenom pro Pascal, dnes už existují i verze pro jiné jazyky, například systém CWEB pro jazyk C.

Citáty

O používání emailu

I have been a happy man ever since January 1, 1990, when I no longer had an email address. I'd used email since about 1975, and it seems to me that 15 years of email is plenty for one lifetime.

O programování

Premature optimization is the root of all evil (or at least most of it) in programming.

My main conclusion after spending ten years of my life working on the TEX project is that software is hard. It's harder than anything else I've ever had to do.

O rovnováze

If you find that you're spending almost all your time on theory, start turning some attention to practical things; it will improve your theories. If you find that you're spending almost all your time on practice, start turning some attention to theoretical things; it will improve your practice.

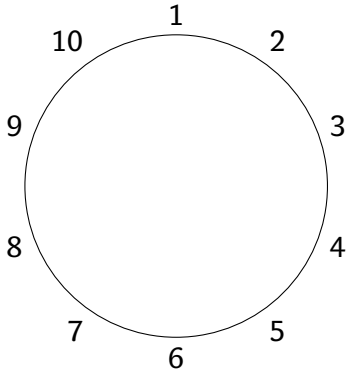
Josefův problém

Petr Osička

Dnešní hádanka je převzata z knihy D. E. Knutha Concrete mathematics. Je inspirován příhodou Josefa, starověkého židovského historika. Traduje se, že během židovsko-římských válek bylo 41 židovských rebelů skrývajících se v jeskyni obklíčeno římskými vojsky. Mezi rebely byl i Josef. Rebelové se rozhodli, že upřednostní sebevraždu před zajetím. Rozhodli se, že se postaví do kruhu a budou zabíjet každého třetího živého, dokud to bude možné. Josef byl jeden ze dvou přeživších, dokázal si spočítat, kam se má postavit.

Pro jednoduchost předpokládejme, že začneme s n lidmi očíslovanými 1, 2, …, n a budeme eliminovat každého druhého přeživšího. *Úkolem je zjistit, který člověk přežije*.

Pokud například vezmeme n = 10, pak situace vypadá následovně.



Protože začínáme od čísla 1, v prvním kole eliminujeme postupně 2, 4, 6, 8, 10. V dalším kole 3, 7 a nakonec 1, 9. Přežije číslo 5.

Nápověda

- Pro n = 2k zůstanou po prvním kole pouze lichá čísla. Tato čísla lze jednoduchým trikem převést opět na 1,2, …, k, tedy instanci Josefova problému pro k osob.
- Pro n liché zůstanou po prvním kole opět lichá čísla, ale tentokrát (protože máme lichý počet osob), smažeme nakonec osobu 1. Zůstane nám tedy 3, 5, …. Opět jednoduchým trikem lze tento problém převést na Josefův problém s n/2-1 osobami
- Pomocí předchozích dvou kroků lze problém vyjádřit jako rekurenci a tu posléze vyřešit. Užitečné je zamyslet se nad tím, jak situace dopadne, je-li n mocninou dvou.

Řešení naleznete na webových stránkách magazínu na adrese http://www.inf.upol.cz/magazin.

Redakce:
Petr Krajča, Petr Osička
Katedra informatiky PřF UP
17. listopadu 12
771 46 Olomouc

web: www.inf.upol.cz/magazin
email: magazin@inf.upol.cz
telefon: 585634701
GPS: 49.591870, 17.262336