INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA STROJNÍ

APLIKOVANÁ INFORMATIKA

6. LEKCE – EXTERNÍ TŘÍDĚNÍ

prof. Ing. Radim Farana, CSc.

Ostrava 2013

ISBN 978-80-248-3017-9

Tento studijní materiál vznikl za finanční podpory Evropského sociálního fondu (ESF) a rozpočtu České republiky v rámci řešení projektu: CZ.1.07/2.2.00/15.0463, MODERNIZACE VÝUKOVÝCH MATERIÁLŮ A DIDAKTICKÝCH METOD

OBSAH

6 EXTERNÍ ŘÍZENÍ

OBSAH KAPITOLY:

lgoritmus dvoucestného slučování (2 - way merging)

MOTIVACE:

Velké objemy dat není možno uložit do paměti, a přesto musíme být schopni je efektivně setřídit. K tomu je potřeba znát alespoň základní algoritmy externího třídění a umět je efektivně použít.

CÍL:

Pro externí algoritmy třídění umět použít algoritmus jednocestného slučování, dvoucestného slučování, přirozeného slučování a umět zhodnotit jejich efektivitu.

Metody třídění, kterými jsme se až doposud zabývali, předpokládaly, že se všechna tříděná data vejdou do operační paměti počítače. V mnoha případech však nelze tento základní předpoklad splnit. V tomto případě musíme použít metody využívající principy externího třídění. U externího třídění však vyvstává problém přístupu na paměťové médium, na němž jsou uložena data. Jde o externí paměťová média, nejčastěji pevné disky, avšak mohou to být i různá magnetopásková zařízení. V této chvíli je si nutné také uvědomit způsob přístupu k souborům na těchto médiích. Některá média umožňují jak sekvenční, tak náhodný přístup. Jiná média umožňují pouze sekvenční přístup. U algoritmů externího třídění bude používat pouze ***sekvenční přístup***, protože i u zařízení s náhodným přístupem je rychlejší. Při posuzování složitosti algoritmu externích třídících metod se budeme věnovat pouze sledování počtu přístupů k externím paměťovým médiím. Počty porovnání dvou prvků v paměti nebo jejich přesuny jsou naprosto zanedbatelné oproti počtům načtení souboru z média. Navíc i sledování I/O operací není zcela vypovídající, protože řešení je velice silně závislé na použitém hardware (buffery zařízení, propustnosti sběrnic,...) a konfiguraci operačního systému (použití vyrovnávacích pamětí apod.)

Obrázek 6.1 Externí třídění

6.1 ALGORITMUS DVOUCESTNÉHO SLUČOVÁNÍ (2 - WAY MERGING)

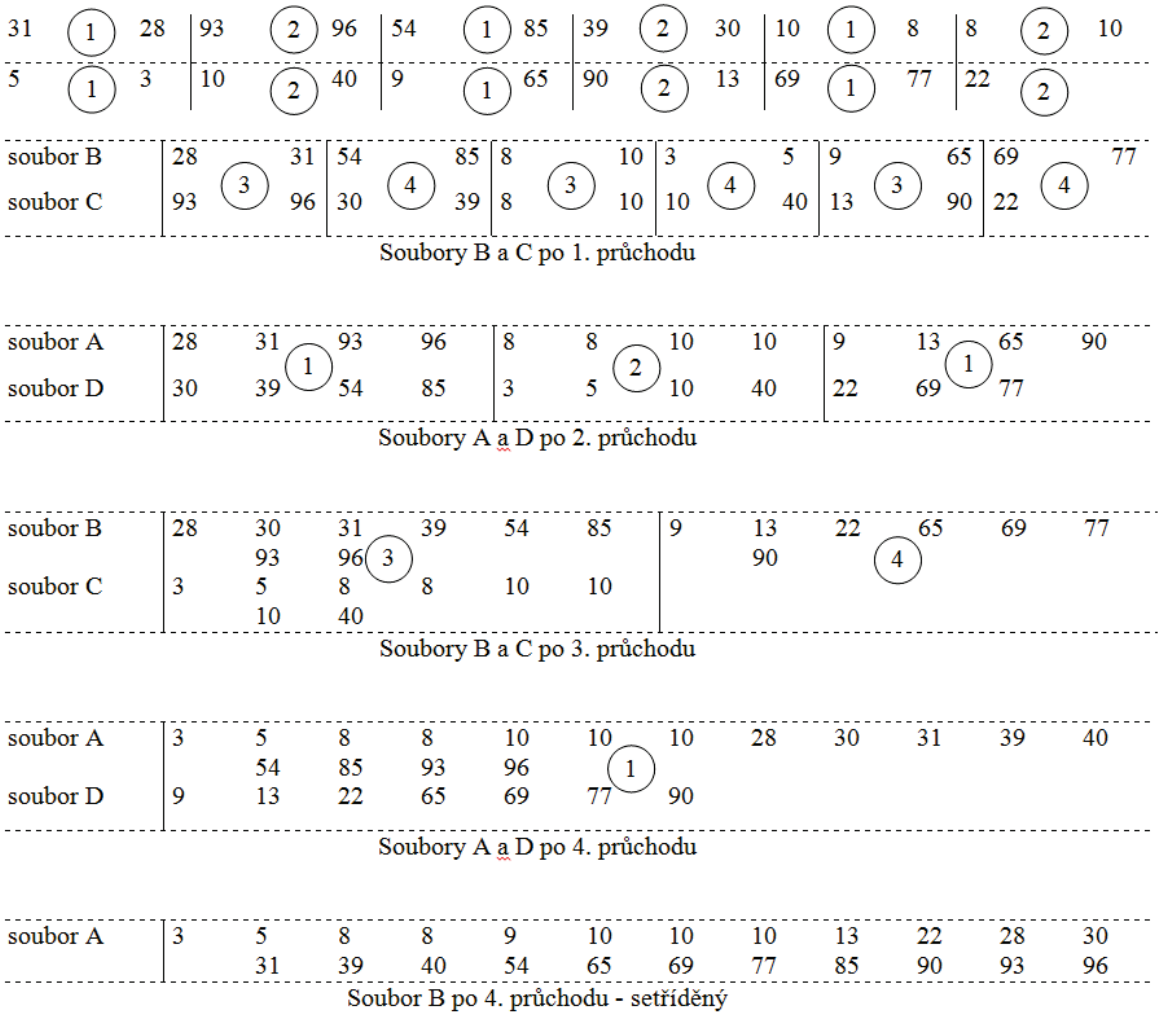
Algoritmus dvoucestného slučování vychází z metody slučování (merge sort) popsané v předchozí kapitole.

Základní myšlenka:

Každý průchod programem lze rozdělit do fáze rozdělování a do fáze slučování. Ve fázi rozdělování jde o rozdělení souboru, řekněme ***A***), do dvou dalších souborů tak, že do souboru ***B*** budou uloženy liché prvky souboru ***A*** a do souboru ***C*** sudé. Následovně je provedeno sloučení souborů ***B*** a ***C*** do souboru ***A*** tak, že jsou vždy složeny jednotlivé prvky ze souboru ***B*** a ***C*** tak, že soubor ***A*** je nyní tvořen setříděnými dvojicemi prvků. V dalším průchodu je soubor ***A*** rozdělen do souborů ***B*** a ***C*** tak, že každého jsou zapsány liché respektive sudé dvojice prvků. Následuje opět slučování, nyní však po dvojicích ze souborů ***B*** a ***C*** do souboru ***A***. Celá operace se opakuje, dokud nejsou v souboru ***A*** data setříděna. Uvedený algoritmus však není moc dobře navržen, protože fáze rozdělování je neefektivní – neprobíhá během ní třídění a znamená jedno čtení a jeden zápis celého souboru dat.

Algoritmus lze proto zefektivnit tím, že použijeme ještě jeden pomocný soubor – ***D***. To může být, jak uvidíme, výhodnější z hlediska počtu přístupů k souboru, avšak někdy nemožné z hlediska kapacity dostupných paměťových médií. Podívejme se však, v čem bude modifikace spočívat. Na počátku rozdělíme soubor ***A*** do souboru ***B*** a ***C*** podle známého klíče. Nyní však budeme provádět slučování tak, že každou sloučenou dvojici střídavě ukládáme do souborů ***A*** a ***D***. Tím jsme eliminovali fázi rozdělování, která měla následovat a může proto ihned spustit fázi slučování, kdy jednotlivé dvojice ze souborů ***A*** a ***D*** slučujeme a ukládáme střídavě do souborů ***B*** a ***C***. Zamyslíme-li se nad uvedeným algoritmem, zjistíme, že úpravou jsme zkrátili potřebný čas k setřídění souboru na polovinu.

Příklad: počáteční soubor je dán následující tabulkou, ve které je přímo označeno, jak budou jednotlivé dvojice sloučených prvků umísťovány do souborů ***B*** a ***C***:



Obrázek 6.2 Příklad dvoucestného slučování

Analýza algoritmu:

Pro jednoduchost předpokládejme, že máme množinu o $n = 2^k$ prvků, kde k je přirozené číslo. V prvním průchodu budeme slučovat 2^k jednoprvkových úseků, ve druhém 2^{k-1} atd. V posledním k-tém průchodu pak sloučíme dva úseky o 2^{k-1} prvcích. Celkově tedy musíme provést $k = \log_2 n$ průchodů. Pro libovolné n pak platí:

$$C_p = \lceil \log_2 n \rceil \qquad (6.1)$$

Poznámka:

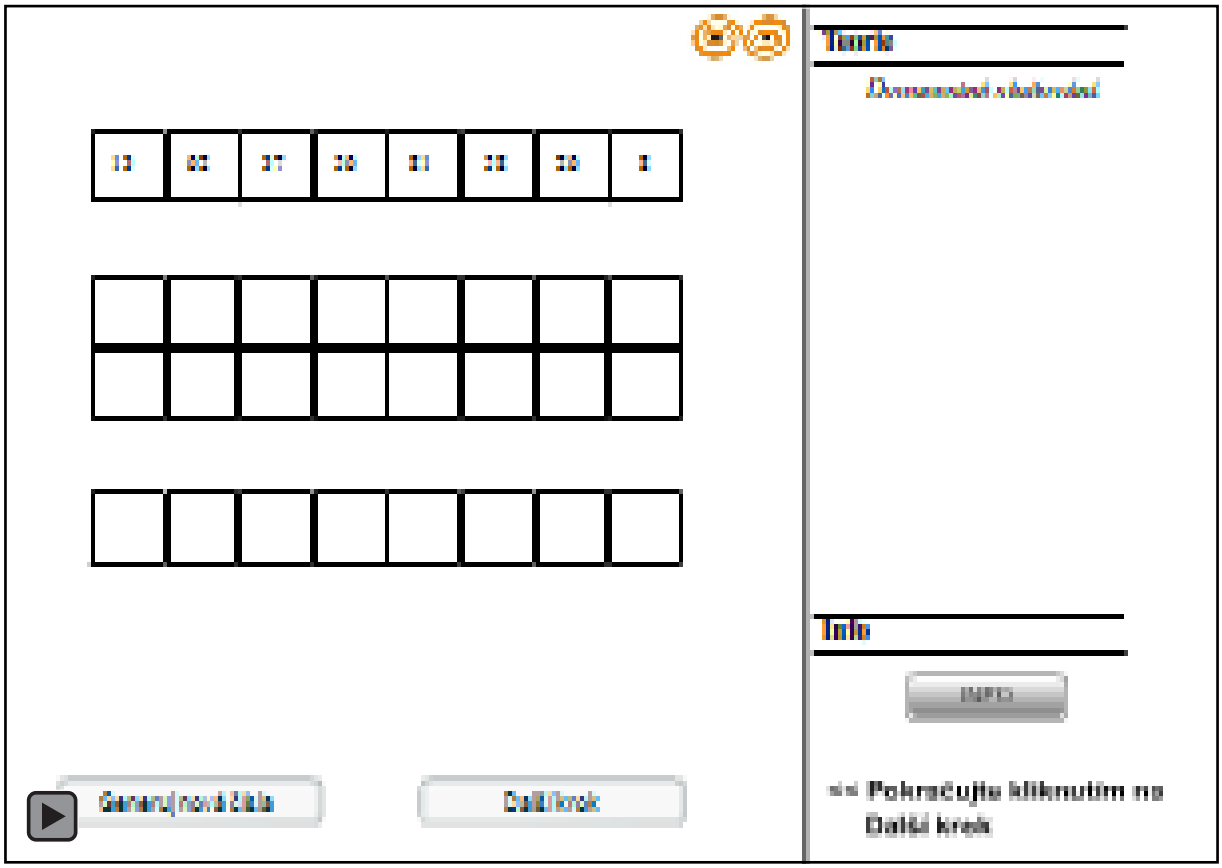
$\lceil \rceil$ - stropní funkce

protože, je-li ***n*** v intervalu $(2^{k-1}, 2^k)$, pak bude třeba vždy ***k*** průchodů. Počet průchodů se tedy vždy zaokrouhlí směrem nahoru. U první verze algoritmu se využívají dva pomocné soubory a každý průchod byl složen z fáze rozdělování a fáze slučování. Fáze rozdělování znamená načtení a uložení celého souboru a fáze slučování taktéž. Tedy při každém průchodu dojde ke dvěma načtením a dvěma uložením celého souboru dat. Pak tedy:

$$C_{ps} = 4n(\lceil \log_2 n \rceil) \tag{6.2}$$

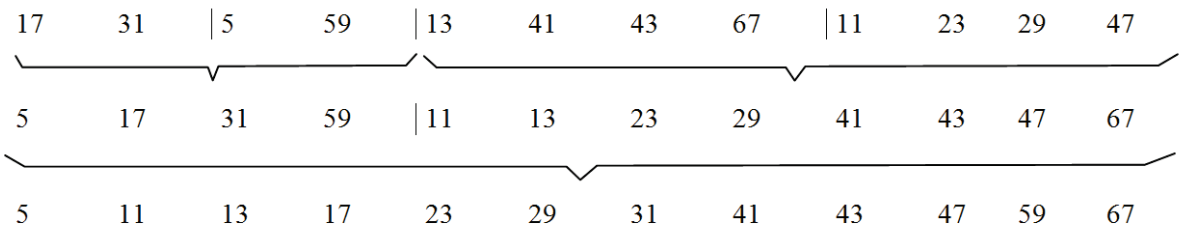
udává celkový počet přístupů do souboru. Přístupem se rozumí načtení nebo zápis jednoho prvku. U druhé verze algoritmu byla eliminována fáze rozdělování a tak počet přístupů do souborů bude poloviční.

Dalšího zrychlení algoritmu lze dosáhnout, pokud budeme slučovat ***n***-tice nekonstantní délky. To proto, že v každém souboru dat se mohou vyskytovat úseky (shluky) dat, které jsou již podle daného klíče setříděny. Pak je určitě zbytečné tyto setříděné shluky rozdělovat dříve uvedeným algoritmem, aby následovně došlo k jejich opětovnému sloučení. Algoritmus, který umožňuje pružně reagovat na uvedenou situaci je označován jako metoda ***Přirozeného slučování***.



Příklad:

původní soubor (setříděné úseky jsou odděleny svislou čarou)



Obrázek 6.3 Metoda přirozeného slučování

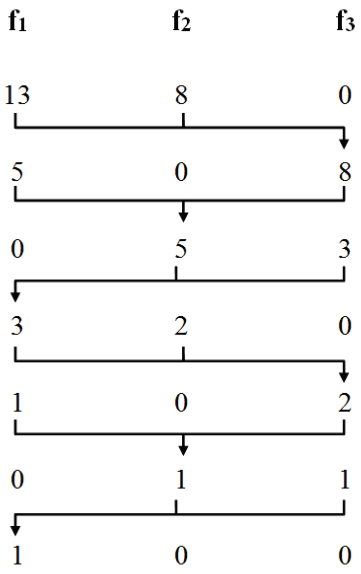
Při provádění rozdělování ***n***-tic (nestejné délky), je důležité, aby obě dílčí posloupnosti měly stejný počet ***n***-tic. Stejná délka je druhořadá. Při slučování jednotlivých ***n***-tic – ***běhů*** – může dojít k jistým komplikacím. Ty vyplývají především z faktu, že počet běhů může být v jednotlivých souborech rozdílný, i když v rozdělovací fázi zapisujeme střídavě do obou souborů. Počty běhů by se tedy mohly teoreticky lišit pouze o hodnotu ***1*** tak, jak tomu bylo v původní verzi algoritmu. V souborech však může dojít k automatickému sloučení dvou po sobě jdoucích běhů. K tomu dojde, pokud poslední prvek ***i-1*** běhu je menší než první prvek běhu ***i***-tého.

Dalším zrychlením může být použití více souborů (pásek), což vede na metodu ***Polyfázového slučování***. Princip spočívá ve slučování z ***n-1*** souborů do ***n***-tého souboru. Jakmile je některý ze vstupních souborů prázdný, přepíná se výstup do tohoto souboru a původní výstupní soubor se stává vstupním.

Uvedený algoritmus je výhodný zejména při použití více souborů (pásek). Důležitá je u tohoto algoritmu především počáteční distribuce *n*-tic. Pro tři soubory počty *n*-tic tvoří ***Fibonacciho posloupnost*** (1. řádu)

0 1 1 2 3 5 8 13 21 34 55

kdy každé číslo je součtem dvou předcházejících.



Obrázek 6.4 Polyfázové třídění se třemi soubory

Pro více souborů platí, že počáteční počet *n*-tic se má rovnat součtu (*n* – 1), (*n* – 2), ..., 1 po sobě jdoucích čísel Fibonacciho čísel (*n* – 2) řádu. Obecně je Fibonacciho posloupnost čísel definována pro řád ***p***:

$$f_{i+1}^{(p)} = f_i^{(p)} + f_{i-1}^{(p)} + \ldots + f_{i-p}^{(p)} \qquad \text{pro i} \geq \text{p}$$
$$\left.\begin{aligned} f_p^{(p)} &= 1 \\ f_i^{p} &= 0 \end{aligned}\right\} \quad \text{pro } 0 \leq \text{i} < \text{p} \tag{6.3}$$

POUŽITÁ LITERATURA

[1] Arlow, J. & Neustadt, I. *UML a unifikovaný proces vývoje aplikací*. 1. Vyd. Brno, Computer Press, 2003, 388 s. ISBN 80-7226-947-X.

[2] Barton, D. P. & Pears, A. N. Application of Evolutionary Computation. In *Proceedings of First International Conference on Genetic Algorithms "Mendel '95"*. Red. Ošměra, P. Brno, VUT 1995, s. 15 - 21.

[3] Bayer, R. & McCreight, E. M. Organization and Maintenance of Large Ordered Indices. *Acta Informatica*, 1, 1972.

[4] Březina, T. *Informatika pro strojní inženýry I*. 1. vyd. Praha ČVUT 1991, 187 s.

[5] Brodský, J. & Skočovský, L. *Operační systém UNIX a jazyk C*. 1. vyd. Praha, SNTL 1989, 368 s.

[6] Cockburn, A. *Use Case – Jak efektivně modelovat aplikace*. 1. vyd. Brno, CP Books a.s., 2005, 262 s. ISBN 80-251-0721-3.

[7] Častová, N. & Šarmanová, J. *Počítače a algoritmizace*. 3. vyd. Ostrava, skriptum VŠB 1983, 190 s.

[8] Donghui Zhang. *B Trees*. Northeastern University, 22 pp. Dostupný z webu: http://zgking.com:8080/home/donghui/publications/books/dshandbook_BTree.pdf

[9] Drózd, J. & Kryl, R. *Začínáme s programováním*. Praha, Grada 1992, 312 s.

[10] Drozdová, V. & Záda, V. *Umělá inteligence a expertní systémy*. 1. vyd. Liberec, skriptum VŠST 1991, 212 s.

[11] Farana, R. *Zaokrouhlovací chyby a my*. Bajt 1994, č. 9, s 243 – 244.

[12] Flaming, B. *Practical data structures in C++*. New York, USA, Wiley, 1993.

[13] Hodinár, K. *Štandardné aplikačné programy osobných počítačov*. 1. vyd. Bratislava, Alfa 1989, 272 s.

[14] Holeček, J. & Kuba, M. *Počítače z hlediska uživatele*. Praha, SPN 1988, 240 s.

[15] Honzík, J. M., Hruška, T. & Máčel, M. *Vybrané kapitoly z programovacích technik*. 3. vyd. Brno, skriptum VUT 1991, 218 s.

[16] Hudec, B. *Programovací techniky*. Praha, ČVUT 1990.

[17] Jackson, M. A. *Principles of Program Design*. New York (USA), Academic Press 1975.

[18] Jandoš, J. *Programování v jazyku GW BASIC*. Praha, NOTO - Kancelářské stroje 1988, 164 s.

[19] Kačmář, D. *Programování v jazyce C++. Objektová a neobjektová rozšíření jazyka*. Ostrava, ES VŠB-TU 1995, 92 s.

[20] Kačmář, D. & Farana, R. *Vybrané algoritmy zpracování informací*. 1. vyd. Ostrava: VŠB-TU Ostrava, 1996. 136 s. ISBN 80-7078-398-2.

[21] Kaluža, J., Kalužová, L., Maňasová, Š. *Základy informatiky v ekonomice*. 1. vyd. Ostrava, skriptum VŠB 1992, 193 s.

[22] Kanisová, H. & Müller, M. *UML srozumitelně*. 1. vyd. Brno, Computer Press, 2004. 158 s. ISBN 80-251-0231-9.

[23] Kapoun, K. & Šmajstrla, V. *Základní fyzikální problémy - programy v jazyce BASIC a FORTRAN*. 1. vyd. Ostrava, skriptum VŠB 1987, 312 s.

[24] Kelemen, J. aj. *Základy umelej inteligencie*. 1. vyd. Bratislava, Alfa 1992, 400 s.

[25] Knuth, D. E. *The Art of Computer Programming*. Volumes 1-4A, 3rd ed. Reading, Massachusetts, Addison-Wesley, 2011, 3168 pp. ISBN 0-321-75104-3.

[26] Kopeček, I. & Kučera, J. *Programátorské poklesky*. Praha, Mladá fronta 1989, s. 150-155.

[27] Krček, B. & Kreml, P. *Praktická cvičení z programování. FORTRAN*. 1. vyd. Ostrava, skripta VŠB 1986, 199 s.

[28] Kučera, L. *Kombinatorické algoritmy*. 2. vyd. Praha, SNTL 1989, 288 s.

[29] Kukal, J. *Myšlením k algoritmům*. 1. vyd. Praha, Grada 1992, 136 s.

[30] Marko, Š. - Štěpánek, M. *Operačné systémy mikropočítačov SMEP*. 2. vyd. Bratislava/Praha, Alfa/SNTL 1988, 264 s.

[31] Medek, V. & Zámožík, J. *Osobný počítač a geometria*. 1. vyd. Bratislava, Alfa 1991, 256 s.

[32] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. 1. vyd. Heidelberg, Springer-Verlag Berlin 1992, 250 s.

[33] *Microsoft Developer Network*. Development Library January 1995. Microsoft Corporation 1995, CD - ROM.

[34] Molnár, Ľ. & Návrat, P. *Programovanie v jazyku LISP*. 1. vyd. Bratislava, Alfa 1988, 264 s.

[35] Molnár, Ľ. *Programovanie v jazyku Pascal*. Bratislava/Praha, Alfa/SNTL 1987, 160 s.

[36] Molnár, Z. *Moderní metody řízení informačních systémů*. Praha, Grada 1992, s. 211 - 221.

[37] Moos, P. *Informační technologie*, 1. vyd. Praha, ČVUT 1993, 200 s.

[38] Nešvera, Š., Richta, K. & Zemánek, P. *Úvod do operačního systému UNIX*. 1. vyd. Praha, ČVUT 1991, 185 s.

[39] Olehla, J. & Olehla, M. aj. *BASIC u mikropočítačů*. 1. vyd. Praha, NADAS 1988, 386 s.

[40] Ošmera, P. Použití genetických algoritmů v neuronových modelech. In *Sborník konference "EPVE 93"*. Brno VUT 1993, s. 88 - 95.

[41] Paleta, P. *Co programátory ve škole neučí aneb Softwarové inženýrství v reálné praxi*. 1. vyd. Brno, Computer Press, 2003, 337 s. ISBN 80-251-0073-1.

[42] Petroš, L. *Turbo Pascal 5.5* – Uživatelská příručka. 1. vydání. Zlín, MTZ 1990, s. 20 – 21.

[43] Plávka, J. *Algoritmy a zložitosť*. Košice, TU Košice, 1998. ISBN 80-7166-026-4.

[44] Podlubný, I. *Počítať na počítači nie je jednoduché*. PC World, 1994, č. 2, s. 112 – 115.

[45] Rawlins, G. J. E. *Compared to what – an introduction to the analysis of algorithms*. Computer Science Press, New York, 1992.

[46] Reverchon, A. & Ducamp, M. *Mathematical Software Tools in C++*. West Sussex (England) John Wiley & Sons Ltd. 1993, 507 s.

[47] Rychlík, J. *Programovací techniky*. České Budějovice, KOPP 1992, 188 s.

[48] Sedgewick, R. *Algorithms*. 1st ed. Addison-Wesley. ISBN 0-201-06672-6.

[49] Sirotová, V. *Programovacie jazyky*. 1. vyd. Bratislava, skriptum SVTŠ 1985, 138 s.

[50] Soukup, B. SGP verze 2.30. *Referenční a uživatelská příručka systému*. Uherské hradiště, SGP Systems 1991, s. 25-55.

[51] Synovcová, M. *Martina si hraje s počítačem*. 1. vyd. Praha, Albatros 1989, 144 s.

[52] Šarmanová, J. *Teorie zpracování dat*. Ostrava, FEI VŠB-TU Ostrava, 2003, 160 s.

[53] Šmiřák, R. *Unified Modeling Language*. Softwarové noviny, 2004, č. 12, s. 76 – 77.

[54] Tichý, V. *Algoritmy I.* Praha, FIS VŠE v Praze, 2006, 190 s. ISBN 80-245-1113-4.

[55] Vejmola, S. *Hry s počítačem*. 1. vyd. Praha, SPN 1988, 256 s.

[56] Virius, M. *Základy algoritmizace*. Praha, ČVUT 2008, 265 s. ISBN 978-80-01-04003-4.

[57] Víteček, A. aj. *Využití osobních počítačů ve výuce*. 1. vyd. Ostrava, ČSVTS FS VŠB Ostrava 1986, 202 s.

[58] Vítečková, M., Smutný, L., Farana, R. & Němec, M. *Příkazy jazyka BASIC*. Ostrava, katedra ASŘ VŠB Ostrava 1989, 44 s.

[59] Vlček, J. *Inženýrská informatika*. 1. vyd. Praha, ČVUT 1994, 281 s.

[60] Wirth, N. *Algoritmy a štruktúry udajov*. 2. vydání. Bratislava, Alfa 1989, s. 19 – 89.

[61] *Základy algoritmizace a programové vybavení*. 2. vyd. Praha, Tesla Eltos 1986, 168 s.

[62] Zelinka, I., Oplatková, Z., Šeda, M., Ošmera, P. & Včelař, F. *Evoluční výpočetní techniky. Principy a aplikace*. 1. vyd. Praha, BEN – Technická literatura, 2009. ISBN 978-80-7300-218-3.