Educational matters
Научно-методически статии

КАК КОМПЮТЪРЪТ РЕШАВА СУДОКУ – МАТЕМАТИЧЕСКИ МОДЕЛ НА АЛГОРИТЪМА

Красимир Йорджев
Югозападен университет „Неофит Рилски“ – Благоевград

Резюме. В работата се разглеждат някои аспекти на обучението по програмиране. Набляга се на занимателния момент при подбора на подходящи примери за демонстрация на отделните езикови конструкции и структури от данни. Такъв пример е разгледаният алгоритъм за решаване на широко разпространената в последно време главоблъсканица судоку. Това е направено във връзка с понятието множество и неговото приложение в програмирането.

Keywords: education in programming; data structures; set; Sudoku matrix; Sudoku puzzle; mathematical model of an algorithm

Настоящата работа е замислена да подпомогне преподавателя по програмиране в стремежа му да даде съдържателен, интересен и забавен пример за ползата на понятието множество в програмирането. За целта обучаемите трябва добре да познават основните дефиниции от теория на множествата и да владеят основните операции с множества. Оттук следва и добре известният факт, че за да си добър програмист, е необходимо (но не и достатъчно) да си добър математик.

Класически пример за използването на множества в програмирането е станала програмната реализация на задачата за намиране на прости числа по метода, наречен „Решето на Ератостен“ (Jensen & Wirth, 1985; Nakov & Dobrikov, 2005; Stoyanova & Gochev, 1994). Тук сме длъжни да отбележим, че в (Stoyanova & Gochev, 1994) при реализацията на този алгоритъм е допусната грешка, причислявайки числото 1 към множеството на простите числа. Подобна грешка е недопустима за едно учебно помагало, тъй като би довела до объркване от страна на ползващите го.

Тук ще се спрем на занимателна и актуална задача, която се приема с голям интерес от страна на учащите – алгоритми за решаване на судоку. Това е широко разпространена в днешно време главоблъсканица, неизменно присъстваща в развлекателните страници на повечето вестни-

ци и списания и в развлекателни интернет сайтове. Судоку е игра, която развива у човек най-вече математическите способности, логическото мислене, комбинативността, въображението, умението за прогнозиране и други качества.

Ние няма да се спираме на програмната реализация на разглежданата от нас задача. Това до голяма степен зависи от езика за програмиране, избран от учебното заведение. Удобен за програмната реализация на разглеждания от нас алгоритъм е езикът „Паскал“, тъй като в него има вградени средства за работа с множества (Jensen & Wirth, 1985; Stoyanova & Gochev, 1994). Същият все още се избира от някои учебни заведения в системата на средното образование в качеството му на първи език за програмиране. Изложените идеи могат да бъдат реализирани и на произволен друг език за програмиране, например C++. Но в този случай трябва да търсим допълнителни средства за работа с множества – например реализираните в Standard Template Library (STL) асоциативни контейнери **set** и **multiset** (Azalov, 2008). Също така може да се използва и шаблонният клас **set** от системата за компютърна алгебра “Symbolic C++”, чийто програмен код е даден в подробности в (Tan, Steeb & Hardy, 2000). Разбира се, с учебна цел би могло да се създаде и собствен клас множество, като се опишат функциите, реализиращи теоретико-множествените операции (Todorova, 2011-a), (Todorova, 2011-b). Това би било прекрасно упражнение, като се има предвид и фактът, че базовото („универсално“) множество е с относително малка мощност. Например стандартната главоблъсканица судоку е с базово множество целите числа от 1 до 9, плюс празното множество.

Тук ще разгледаме само математическия модел на алгоритъма. По този начин съставянето на компютърна програма, решаваща произволна главоблъсканица судоку, се превръща в една нелоша тема за разработка на курсов проект по програмиране за ученици и студенти. Много често, както и в случая, описанието на математическия модел е достатъчно за програмната реализация на съответния алгоритъм.

Математически модел на алгоритъма

Нека n е произволно цяло положително число и нека $m = n^2$. Нека $S = (s_{ij})$, $1 \le i, j \le m = n^2$ е квадратна $m \times m$ матрица (квадратна таблица, двумерен масив), всички елементи на който са цели числа, принадлежащи на затворения интервал $[1, m]$. С помощта на $n-1$ хоризонтални и $n-1$ вертикални линии, прекарани съответно между някои от редовете и някои от стълбовете, матрицата S е разделена на n^2 непресичащи се $n \times n$ квадратни подматрици, които ще наричаме блокове. На фиг. 1 е показана матрицата S при $n = 3$.

S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}
S_{21}	S_{22}	S_{23}	S_{24}	S_{25}	S_{26}	S_{27}	S_{28}	S_{29}
S_{31}	S_{32}	S_{33}	S_{34}	S_{35}	S_{36}	S_{37}	S_{38}	S_{39}
S_{41}	S_{42}	S_{43}	S_{44}	S_{45}	S_{46}	S_{47}	S_{48}	S_{49}
S_{51}	S_{52}	S_{53}	S_{54}	S_{55}	S_{56}	S_{57}	S_{58}	S_{59}
S_{61}	S_{62}	S_{63}	S_{64}	S_{65}	S_{66}	S_{67}	S_{68}	S_{69}
S_{71}	S_{72}	S_{73}	S_{74}	S_{75}	S_{76}	S_{77}	S_{78}	S_{79}
S_{81}	S_{82}	S_{83}	S_{84}	S_{85}	S_{86}	S_{87}	S_{88}	S_{89}
S_{91}	S_{92}	S_{93}	S_{94}	S_{95}	S_{96}	S_{97}	S_{98}	S_{99}

Нека да означим блоковете в дефинираната по-горе матрица $S = (s_{ij})$ с A_{kl}, $1 \le k, l \le n$. Тогава по дефиниция, ако $s_{ij} \in A_{kl}$, то

$$(k-1)n < i \le kn$$

и

$$(l-1)n < j \le ln.$$

Нека s_{ij} принадлежи на блока A_{kl} и нека да са известни i и j. Тогава лесно можем да съобразим, че k и l могат да бъдат изчислени с помощта на формулите

$$k = \left[\frac{i-1}{n}\right] + 1$$

и

$$l = \left[\frac{j-1}{n}\right] + 1,$$

където с $[x]$ сме означили както обикновено функцията цяла част на реалното число x.

Ще казваме, че $S = (s_{ij}), 1 \le i, j \le m = n^2$ е *судоку-матрица*, ако във всеки ред, всеки стълб и всеки блок съществува точно по едно число от множеството $Z_m = \{1,2,\ldots,m = n^2\}$.

Широко разпространена е главоблъсканицата, наречена *судоку*. Дадена е судоку-матрица, на която са изтрити някои от елементите. Липсващите елементи на S ще отъждествяваме с 0. Задачата на главоблъсканицата се състои в това да се възстановят липсващите елементи на судоку-матрицата. Предполага се, че авторите на конкретната главоблъсканица така са подбрали липсващите елементи, че задачата да има единствено решение. Това условие ще пропуснем и няма да се съобразяваме с него. Нещо повече, препоръчваме в заданието към обучаемите да бъде поставено условието да бъде намерен и

броят на възможните решения. Ако задачата няма решение, то този брой ще бъде нула.

Най-разпространените главоблъсканици судоку са при $n=3$, т.е. при $m=9$.

Тук ще направим математически модел, описващ алгоритъм за създаване на компютърна програма, намираща всички решения (ако съществуват) на произволна главоблъсканица судоку. За целта съществено ще използваме познанията от теория на множествата.

Разглеждаме множествата R_i, C_j и B_{kl}, където $1 \le i,j \le m = n^2$, $1 \le k,l \le n$. За всяко $i = 1,2,\ldots,m$, множеството R_i се състои от всички липсващи числа в i-я ред на матрицата. Аналогично дефинираме и множествата C_j, $j = 1,2,\ldots,m$ съответно за липсващите числа в j-я стълб и B_{kl}, $k,l = 1,2,\ldots,n$ съответно за липсващите числа в блоковете A_{kl} на S. По такъв начин, ако стартираме програмата и дадем нулеви стойности на всички елементи на матрицата *S*, то с помощта на създадената програма ще получим всевъзможните $m \times m$ судоку-матрици.

Началото на алгоритъма се състои в многократното обхождане на всички елементи $s_{ij} \in S$, такива че $s_{ij} = 0$, т.е. това са елементите, чиито истински стойности трябва да открием.

Нека $s_{ij} = 0$ и нека $s_{ij} \in A_{kl}$. Полагаме

$$P = R_i \cap C_j \cap B_{kl}.$$

Тогава са възможни следните три случая:

i) $P = \phi$ (празното множество). В този случай задачата няма решение.

ii) $P = \{d\}$, $d \in Z_m = \{1,2,\ldots,m\}$, т.е. броят $|P|$ на елементите на P е равен на 1 (P е едноелементно множество). Тогава единствената възможност за s_{ij} е $s_{ij} = d$, т.е. в този случай сме открили неизвестната стойност на s_{ij}. Премахваме общия елемент d от множествата R_i, C_j и B_{kl}, след което преминаваме към следващия нулев елемент на матрицата S (ако съществува такъв).

iii) $|P| \ge 2$. Тогава нищо не можем да кажем за неизвестната стойност на s_{ij} и преминаваме към следващия нулев елемент на матрицата S.

Обхождането на нулевите елементи на матрицата S продължава, докато не настъпи едно от следните събития:

e1) за някои $i,j \in \{1,2,\ldots,m\}$ е изпълнено $s_{ij} = 0$, но $P = R_i \cap C_j \cap B_{kl} = \phi$. В този случай задачата няма решение;

e2) всички елементи на S станат различни от нула (строго положителни). Намерили сме едно решение на главоблъсканицата;

e3) обходени са всички нулеви елементи на S, но не е настъпило нито събитие e1, нито събитие e2. С други думи, при всички оставащи нулеви елементи на S винаги е в сила описаният по-горе случай iii.

В случай че настъпи едно от събитията e1 или e2, то процедурата спира своята работа и извежда получения резултат.

В случай че настъпи събитие e3, то алгоритъмът трябва да продължи, използвайки други способи, например да приложим метода на „*пробите и грешките*“. В конкретния случай този метод се състои в следното.

Избираме произволно $s_{ij} \in S$, такова че $s_{ij} = 0$ и нека $k = \left[\frac{i-1}{n}\right] + 1$, $l = \left[\frac{j-1}{n}\right] + 1$. Нека $P = R_i \cap C_j \cap B_{kl} = \{d_1, d_2, \ldots, d_t\}$. Тогава за всяко $d_r \in P$, $r = 1,2,\ldots,t$ полагаме $s_{ij} = d_r$. Такова полагане ще наричаме *случайна проба*. (Добре би било в програмната реализация на алгоритъма да броим и всички случайни проби до достигане на дадено решение.) След това решаваме задачата за намиране на неизвестните елементи на судоку-матрицата с един неизвестен елемент по-малко. Тук е удобно използването на рекурсия. База на рекурсията, т.е. ще излезем от процедурата, ако настъпи събитие e1 или e2. Това неминуемо би трябвало да се получи (т.е. няма да попаднем във „вечен цикъл“), тъй като при случайните проби намаляваме броя на нулевите елементи с 1.

На базата на описания тук модел ние реализирахме компютърна програма, която намира решението на произволно судоку (Yordzhev, 2014).

Стартирахме създадената програма при нулеви стойности на всички клетки. При n = 2 получихме, че броят на всички 4 X 4 судоку-матрици е равен на 288. При това, за да се получи този резултат, компютърът е направил 568 случайни проби. При n = 3 получихме, че съществуват точно 96 670 903 752 021 072 936 960 на брой 9 X 9 судоку матрици. Този резултат е известен и напълно съвпада с резултата, получен от други учени, вероятно и с други методи. На нас не ни е известен броят на всички 16 X 16 судоку-матрици и смятаме, че това е открит за науката проблем. Но за да бъде решен този проблем, за целта е необходимо да ползваме много мощни и много бързи компютри.

REFERENCES/ЛИТЕРАТУРА

Azalov, P. (2008). *Obektno orientirano programirane – strukturi ot danni* i *STL*. Sofia: Ciela (ISBN 978-954-28-0184-9). [Азълов, П. (2008). *Обектно ориентирано програмиране – структури от данни и STL*. София: Сиела (ISBN 978-954-28-0184-9).]

Jensen, K. & Wirth, N. (1985). *Pascal User Manual and Report*. 3rd ed., Springer-Verlag, (ISBN 3-540-96048-1.)

Nakov, P. & Dobrikov, P. (2005). *Programirane=++Algiritmi*. 3-thEdition, Sofia (ISBN 954-8905-16-X).[Наков, П. & Добриков,

П. (2005). *Програмиране =++ Алгоритми*. Трето издание. София (ISBN 954-8905-16-X).]

Stoyanova, R. & Gochev,G. (1994). *Programirane na Pascal s 99 primerni programi*. Sofia: Paraflow (ISBN 954-564-012-X). [Стоянова, Р. & Гочев, Г. (1994). *Програмиране на Pascal с 99 примерни програми*. София: Paraflo (ISBN 954-564-012-X).]

Tan, K. S., Steeb, W. -H. & Hardy, Y. (2000). *Symbolic C++: An Introduction to Computer Algebra using Object-Oriented Programming*. Springer (ISBN 1-85233-260-3.)

Todorova, M. (2011-a). *Obektno orientirano programirane na bazata na ezika C ++*. Sofia: Ciela, ISBN 978-954-28-0909-8. [Тодорова, М. *Обектно ориентирано програмиране на базата на езика C++*. София: Сиела, ISBN 978-954-28-0909-8.]

Todorova, M. (2011-b). *Strukturi ot danni i programirane na ezika C ++*. Sofia: Ciela (ISBN 978-954-28-0990-6). [Тодорова, М. (2011) *Структури от данни и програмиране на езика C++*. София: Сиела (ISBN 978-954-28-0990-6).]

Yordzhev, K. (2014). *Pobitovi operatcii, grafi i kombinatorni prilozheniya*. Blagoevgrad: SWU "Neofit Rilski" (ISBN 978-954-680-961-2). [Йорджев, К. (2014). *Побитови операции, графи и комбинаторни приложения*. Благоевград: ЮЗУ „Н. Рилски" (ISBN 978-954-680-961-2).]

HOW DOES THE COMPUTER SOLVE SUDOKU – A MATHEMATICAL MODEL OF THE ALGORITHM

Abstract. Some aspects of programming education are examined in this work. The emphasis is on the entertainment value. The most appropriate examples are chosen to demonstrate the different language constructions and data structures. Such an example is the demonstrated algorithm for solving the widespread nowadays sudoku puzzle. This is made because of the connection with the term set and putting it into practice in the programming.

✉ **Assoc. Prof. Krasimir Yordzhev, DSc.**
Faculty of Natural Sciences
South-West University "Neofit Rilski"
Blagoevgrad, Bulgaria
E-mail: yordzhev@swu.bg