PROCEEDINGS

of the Union of Scientists - Ruse

Book 5

Mathematics, Informatics and Physics

Volume 9, 2012

RUSE

The Ruse Branch of the Union of Scientists in Bulgaria was founded in 1956. Its first Chairman was Prof. Stoyan Petrov. He was followed by Prof. Trifon Georgiev, Prof. Kolyo Vasilev, Prof. Georgi Popov, Prof. Mityo Kanev, Assoc. Prof. Boris Borisov, Prof. Emil Marinov, Prof. Hristo Beloev. The individual members number nearly 300 recognized scientists from Ruse, organized in 13 scientific sections. There are several collective members too – organizations and companies from Ruse, known for their success in the field of science and higher education, or their applied research activities. The activities of the Union of Scientists – Ruse are numerous: scientific, educational and other humanitarian events directly related to hot issues in the development of Ruse region, including its infrastructure, environment, history and future development; commitment to the development of the scientific organizations in Ruse, the professional development and growth of the scientists and the protection of their individual rights.

The Union of Scientists – Ruse (US – Ruse) organizes publishing of scientific and popular informative literature, and since 1998 – the "Proceedings of the Union of Scientists- Ruse".

BOOK 5

"MATHEMATICS, INFORMATICS AND PHYSICS"

VOLUME 9

CONTENTS

AN APPROACH TO FLOW CHARTS COMPARING

Galina Atanasova, Plamenka Hristova, Katalina Grigorova

Angel Kanchev Ruse University

Abstract: *The paper justifies the necessity to ensure the accurate and immediate feedback in algorithms learning. An approach is proposed based on software engineering using the control flow graph and its cyclomatic complexity metric. The representation of algorithms is given using flow charts and their internal representation by double linked lists. Two algorithmic problems and their teacher's and students' solutions are described. The corresponding control flow graphs and their cyclomatic complexity are calculated. Conclusions about the reliability and feasibility of the proposed approach are summarized.*

Keywords: *Computer Science, Algorithm, Algorithm Learning Environments, Novice Programmer, Software Engineering, Control Flow Graph, Cyclomatic Complexity*

1. INTRODUCTION

Computer technology has progressed quickly and personal computers and the Internet have become closely linked to all areas of human life. Technology has also substantially influenced education. Educational applications together with individual approaches within the didactics process give an excellent chance to increase the quality of education. With the help of a virtual learning environment teachers can make students' study more effective and time-efficient. Carefully prepared study material placed into a virtual learning environment enables to demonstrate and visualize the subject matter more clearly and comprehensibly, to develop students' logical thinking, to increase their imagination and to help them to solve various problems.

One of the most recommended Computer Science program outcomes are able to analyze a given problem, identify the computing requirements and develop an algorithm appropriate to its solution. Novice programmers face many difficulties in designing an algorithm. This leads to an overwhelming perception of incapability and uphill struggle in a large proportion of them. We found that the provision of a useful and accurate mental model would positively influence a novice's success in programming. Flowcharts are a very appropriate form visualisation for novices. They are easy to learn and provide an accurate and useful mental model of an algorithm and its components. Flowcharts aid the processes of abstracting a problem into a solution and the solution into a working program. We worked out a tool, called Flow chart Interpreter, with the basic aim to improve students' abilities in algorithm learning. This tool ensures an algorithm representation via flow chart and its step-by-step interpretation with concrete variable values.

2. TEACHING AND THE FEEDBACK

Teaching in the university has been under pressure to change in recent years. On one hand there is a financial pressure to decrease resources. On the other, there is a need to keep quality and quantity of education offered high considering the changes in technology and learning methods. One responce to these pressures has been to build, if possible, a learning environment for algorithms that is available to students virtually. It could help to distribute materials, make exercises and facilitate overall communication from course information through students feedback. It is very important to ensure that the feedback given to the learner is aligned with the overall learning outcomes of the programme, teaching session or activity in which the learner is engaged. Giving feedback can be seen as a part of experiential learning. Kolb [3] proposed that learning happens in a circular fashion, that learning is experiential (learning by doing), and that ideas are formed and modified through experiences. These ideas underpin the idea of the 'reflective

practitioner' and the shift from 'novice to expert' which occurs as a part of professional development. The feedback plays an important role in helping learners move forward in their awareness.

3. TWO FLOW CHARTS COMPARISON

We consider the problem to ensure the feedback in a learning environment for Algorithms, etc. to develop Flowchart Interpreter's function. We need to work out two flow charts comparison to be able to verify the student's decision to that of the teacher. For a given algorithmic task, there is more than one correct solution. This is the reason it cannot be used for the comparison of images. We looked for a solution in the theory of software engineering and in particular the possibilities of using a control flow graph and its McCabe's [4] cyclomatic complexity. In the literature review, we have found that control flow graphs are widely used in the study of software and software testing [2, 5, 6, 7]. It is not reported for concrete research for flow graph application in algorithm analysis and comparison.

Cyclomatic complexity measures the amount of decision logic in a single software module. It is used for two related purposes in the structured testing methodology. Firstly, it gives the number of recommended tests for software. Secondly, it is used during all phases of the software lifecycle, beginning with design, to keep software reliable, testable, and manageable. Cyclomatic complexity is based entirely on the structure of software's control flow graph. Control flow graphs describe the logic structure of software modules. A module corresponds to a single function or a subroutine in typical languages. It has a single entry and an exit point, and is able to be used as a design component via a call/return mechanism. Each flow graph consists of nodes and edges. The nodes represent computational statements or expressions and the edges represent the transfer of control between nodes.

Cyclomatic complexity is defined for each module to be $e - n + 2$, where e and n are the number of edges and nodes in the control flow graph, respectively. Cyclomatic complexity is also known as $v(G)$, where v refers to the cyclomatic number in graph theory and G indicates that the complexity is a function of the graph. The word "cyclomatic" comes from the number of fundamental (or basic) cycles in connected, undirected graphs [1]. More importantly, it also gives the number of independent paths through strongly connected directed graphs. A strongly connected graph is one in which each node can be reached from any other node by following directed edges in the graph. The cyclomatic number in graph theory is defined as $e - n + 1$. Program's control flow graphs are not strongly connected, but they become strongly connected when a "virtual edge" is added connecting the exit node to the entry node. The cyclomatic complexity definition for program's control flow graphs is derived from the cyclomatic number formula by simply adding one to represent the contribution of the virtual edge. This definition makes the cyclomatic complexity equal the number of independent paths through the standard control flow graph model, and avoids explicit mention of the virtual edge.

Our suggestion for two block schemes comparison is based on flow graph's properties. We can make control flow graph for each block scheme and compute its cyclomatic complexity. We can store the correct (teacher's) $v(G)$-T for each block scheme. When a student's solution proceeds we can build the corresponding control flow graph and to compute its cyclomatic complexity $v(G)$-S. If $v(G)$-S equals to $v(G)$-T we can state that the student's solution is correct.

4. INTERNAL REPRESENTATION OF A BLOCK SCHEME AND CYCLOMATIC COMPLEXITY CALCULATION

We store each block scheme as a double linked list. Its elements have an information part and two pointer fields respectively the previous and next list's element (Fig. 1). The information fields have the following meaning:

- ***ID*** – sequent item number;
- ***Item*** – string for kind of the block;
- ***X1, Y1, X2, Y2*** – bottom left and top right coordinates of the item's surrounding rectangle;
- virtual ***Draw*** (Canvas : TCanvas; HP, VP : Integer) – the item's drawing method;
- ***PrevID*** – ID of the previous item in the block scheme;
- ***Next1ID*** - ID of the following item in the block scheme;
- ***Next2ID*** - ID of the second following item in the block scheme, used only for conditions;
- ***Text*** – string field for the item's text;
- ***Bool*** – boolean field for visit item.

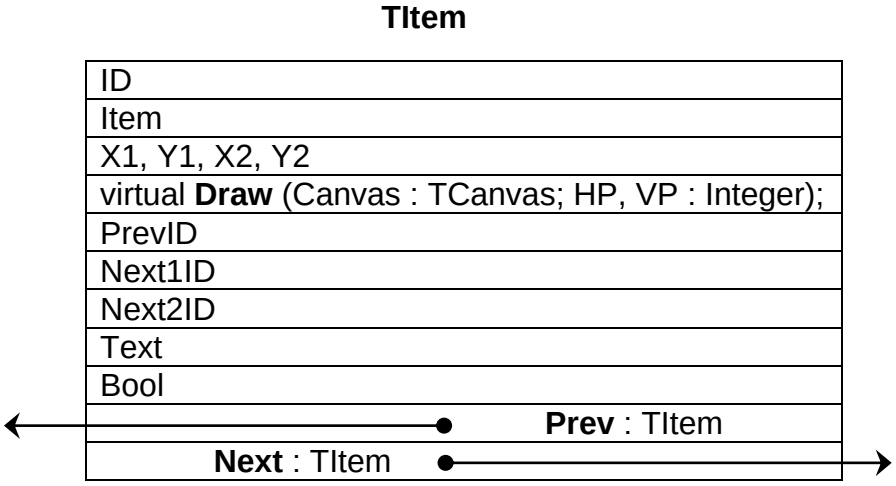

Fig. 1. An element of double linked list

To calculate the cyclomatic complexity of the relevant control flow graph we need the number of block items and the number of connections between them. Each block item corresponds to a control flow graph's node. We can search the list, check the 'Item' field content and count the number of nodes ***n***. In a similar way by counting the number of elements of the content 'Arrow' we get the ***e*** value. When we already have the ***n*** and ***e*** it is sufficient to substitute their values in the formula and obtain ***v(G)***. Thereby it is possible to calculate the cyclomatic complexity of the relevant control flow graph for each block scheme. Significant advantage of this approach is that we work with linear data structure, i.e. list and avoid the complexity of graph's presentation.

5. ALGORITHM COMPARING PRESENTED VIA FLOWCHARTS AND ITS CYCLOMATIC COMPLEXITY. UNIVERSALITY AND APPLICABILITY OF THE PROPOSED APPROACH.

Many novice students approach to seeking an algorithm decision by brute force due to lack of sufficient knowledge and experience. In most cases, they construct correct algorithm, but different from those the teacher. The reason for this is that often problems allow more than one correct solution. Therefore, to assess the reliability and correctness of the proposed approach, we compare different solutions to a couple of problems. In the learning process, actors are a teacher and learners. We have a correct teacher's algorithm for every given task. Consequently, we can build control flow graph and calculate its

cyclomatic complexity *v(G)-T*. After a student's solution received, the flow graph is building and it is calculating its cyclomatic complexity *v(G)-S*. Let consider two problems to study the proposed approach.

Problem 1. Determine the smallest integer k, for which the condition $a^k \geq b$, where (b>1 and a>1) is fulfilled.

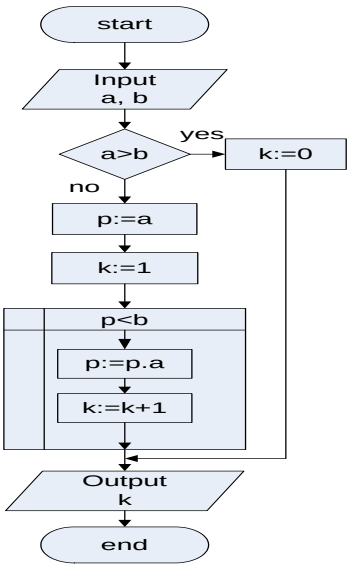

Fig. 2. Problem 1 teacher's solution

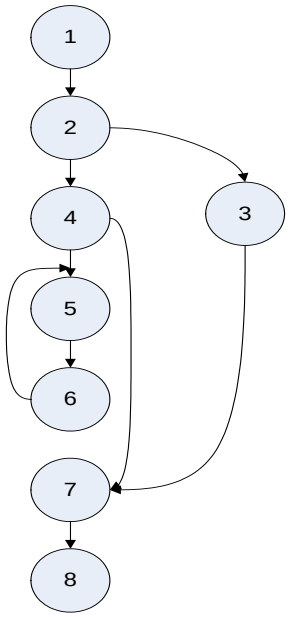

Fig. 3. Teacher's solution flow graph

On the Fig. 2 it is shown the teacher's correct algorithm for the given problem 1. Its corresponding flow graph is on Fig. 3. We count ***n***=8, ***e***=9 and calculate cyclomatic complexity ***v(G)***=E-N+2=9-8+2=3. Let consider two students X and Y solutions.

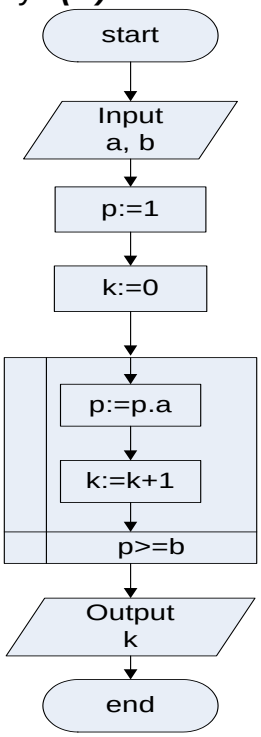

Fig. 4. Problem 1 student X solution

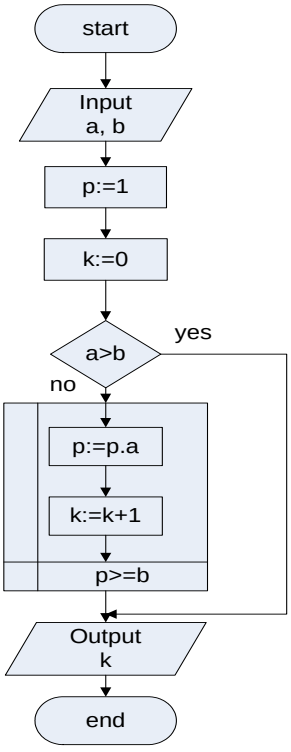

Fig. 5. Problem 1 student Y solution

The student X constructed the algorithm presented on the Fig. 4. The corresponding flow graph is on Fig. 6. In this case we have ***n***=6, ***e***=6 and ***v(G)***=E-N+2=6-6+2=2. We can see that the cyclomatic complexity for this solution is unequal to those of the teacher's one. So we can conclude that the student X solution is incorrect. The student Y solution is shown on Fig. 5 and the relevant control flow graph on Fig. 7. For this solution we have ***n***=7, ***e***=8 and ***v(G)***=E-N+2=8-7+2=3. In this case the obtained value ***v(G)*** matches that of the teacher.

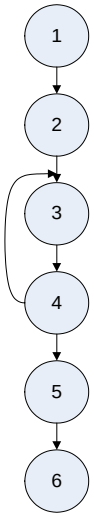

Fig. 6. Problem 1 student X flow graph

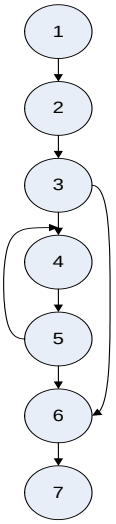

Fig. 7. Problem 1 student Y flow graph

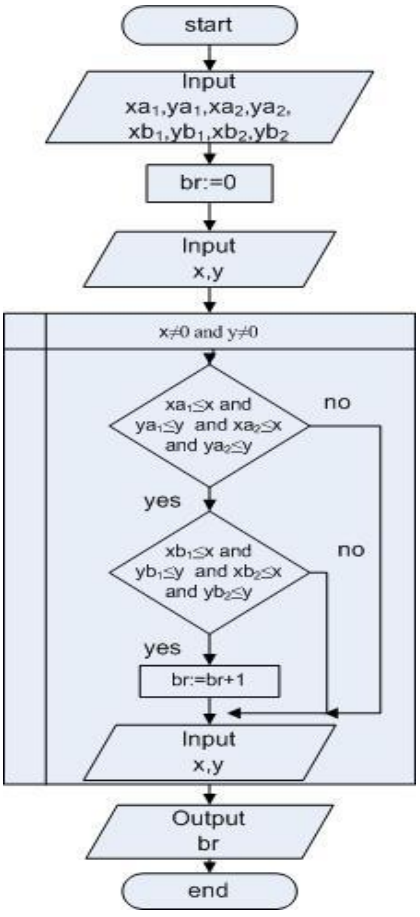

Fig. 8. Problem 2 teacher's solution

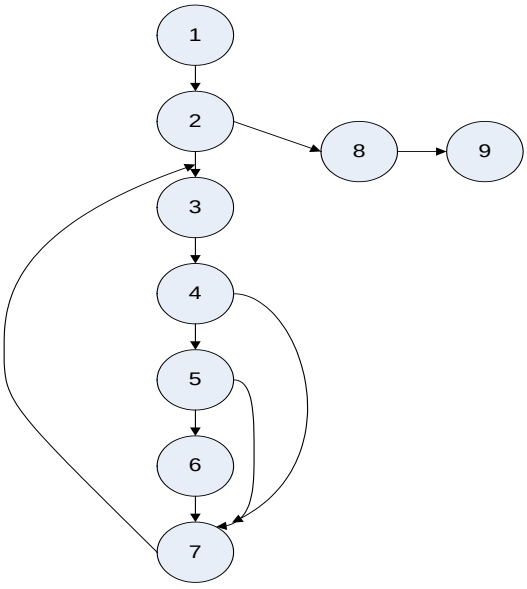

Fig. 9. Problem 2 teacher's flow graph

We can see that the values of both the teacher's and the student X flow graphs cyclomatic complexity are equal regardless the used algorithm instructions. This confirms that for all flow graphs corresponding to correct solutions of the given problem the cyclomatic complexity is constant.

For more reliance let discuss one more problem.

Problem 2: Let we have two rectangles with sides parallel to the axes. The rectangles are defined with the coordinates of the lower left and upper right corner. Construct an algorithm that read the data on two rectangles, then – a sequence of coordinates of points in the plane until the entering of the point with coordinates (0, 0) and displays the number of points, which belong to the both rectangles.

For the given problem 2 figures 8 and 9 present the teacher's solution and corresponding control graph. For this solution we count ***n***=9, ***e***=11 and calculate ***v(G)***=E-N+2=11-9+2=4. We consider two students designed X and Y solutions. They are presented respectively on figures 10 and 11. The corresponding to the student X solution flow graph is shown on fig. 12. For this student we have values ***n***=8, ***e***=10 and calculate cyclomatic complexity ***v(G)***=E-N+2=10-8+2=4. We can see that student's X ***v(G)*** equals to the value ***v(G)*** obtained for the teacher's decision. Let consider the student Y solution. We build the corresponding control flow graph as is shown on fig. 13. In this case we have ***n***=8, ***e***=9 and ***v(G)***=E-N+2=9-8+2=3. The obtained value for the student Y is different from that of the teacher and we can state that this solution is incorrect. Problem 2 we use for confirming the reliability of the proposed approach for two algorithm comparison.

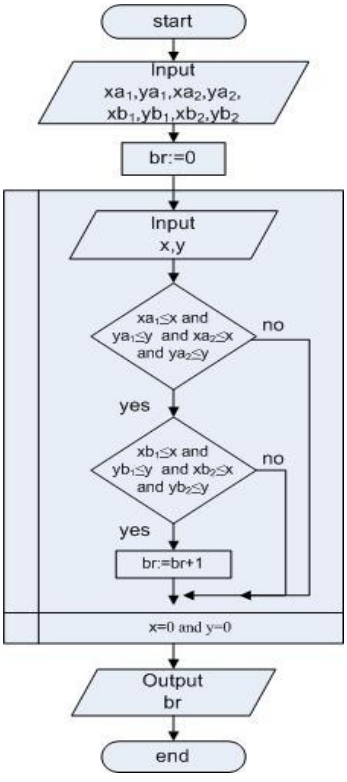

Fig. 10. Problem 2 student X solution

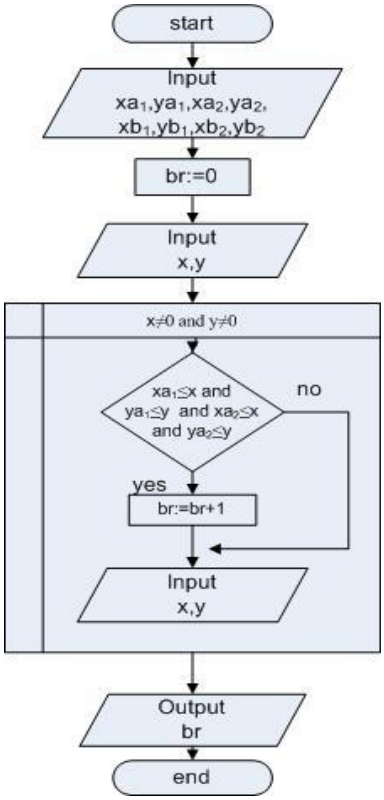

Fig. 11. Problem 2 student Y solution

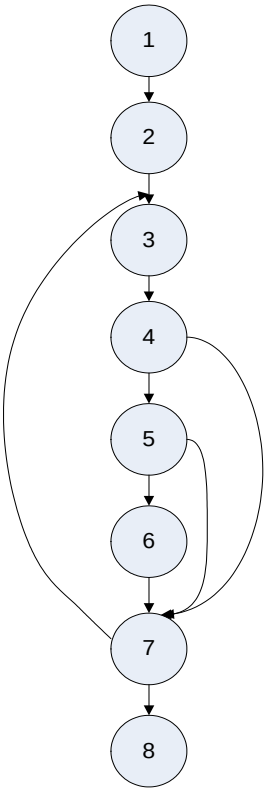

Fig. 12. Pr. 2 student X solution flow graph

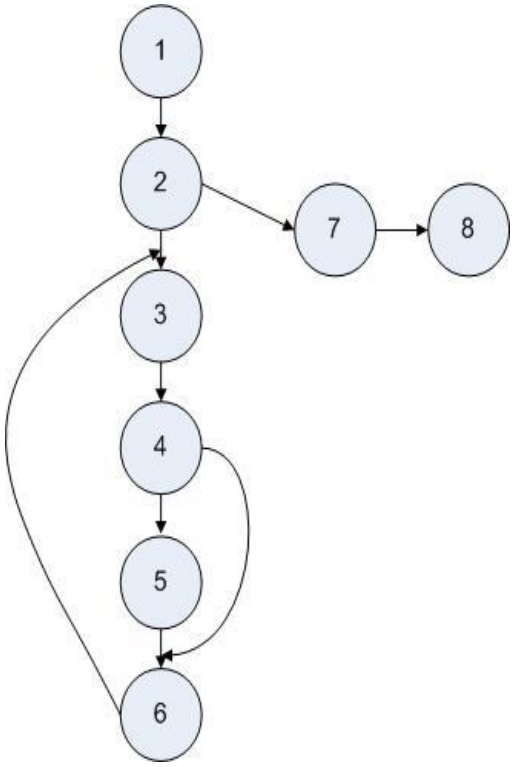

Fig. 13. Pr. 2 student Y solution flow graph

CONCLUSIONS AND FUTURE WORK

The accurate and immediate feedback is very important for the learning process. We need to ensure such feedback in algorithms. We searched a decision in the software engineering methods and proposed an approach using the control flow graph and its cyclomatic complexity metric. We use representation of algorithms using flow charts and their internal representation by double linked lists. This internal representation allows us to extract the data necessary to calculate the cyclomatic complexity of the control flow graph without being necessary a further internal representation of the graph. Therefore we believe that the proposed approach is easy applicable. By dint of two problems, we examine the applicability and reliability of such a way to algorithm comparing. At this moment, we can state that we can use the cyclomatic complexity for ensuring the feedback in an environment for learning algorithms.

The proposed approach gives results whether a given algorithm is correct. We have an intention to upgrade the proposed approach so that we can receive more information about the error. This could bring us more pedagogical benefits.

REFERENCES

[1] Berge, C., Graphs and Hypergraphs, North-Holland, 1973.
[2] Jalote, P., An Integrated Approach to Software Engineering, Springer, New York, NY, 2005.
[3] Kolb, David A., Experiential Learning: Experience as the Source of Learning and Development. Prentice-Hall, Inc., Englewood Cliffs, N.J, 1984.
[4] McCabe, T., A Complexity Measure, IEEE Transactions on Software Engineering, December 1976.
[5] Tan, L., TheWorst Case Execution Time Tool Challenge 2006: The External Test, Technical report, 2006, http://www.absint.com/ait/wcet tool challenge 2006 final

report.pdf.
[6] Sommerville, I., Software Engineering, 7th Edn., Pearson Education Limited, Boston, MA, 2004.
[7] Zhu, H., Hall, P. and May, J., Software unit test coverage and adequacy, ACM Computing Surveys 29(4): 366–427, 1997.

CONTACT ADDRESSES

Pr. Assist. Galina Atanasova
Department of Informatics and Information Technologies, FNSE
Angel Kanchev University of Ruse
7017 Ruse, Studentska Str. 8, Bulgaria
Phone: (++35982) 888 326
E-mail: gea@ami.uni-ruse.bg

Assoc. Prof. Plamenka Todorova Hristova, PhD
Department of Informatics and Information Technologies, FNSE
Angel Kanchev University of Ruse
7017 Ruse, Studentska Str. 8, Bulgaria
Phone: (++35982) 888 326
E-mail: ptx@ami.uni-ruse.bg

Assoc. Prof. Katalina Grigorova, PhD
Department of Informatics and Information Technologies, FNSE
Angel Kanchev University of Ruse
7017 Ruse, Studentska Str. 8, Bulgaria
Phone: (++35982) 888 464
E-mail: katya@ami.uni-ruse.bg

ЕДИН МЕТОД ЗА СРАВНЯВАНЕ НА БЛОК СХЕМИ

Галина Атанасова, Пламенка Христова, Каталина Григорова

Русенски университет “Ангел Кънчев”

Резюме: *Статията обосновава необходимостта от гарантиране на точна и непосредствена обратна връзка при обучението по алгоритми. Предлага се подход, основан върху основите на софтуерното инженерство, използващ графа на протичане и оценката на неговата цикломатична сложност. За представянето на алгоритмите се използват блок-схеми и тяхното вътрешно представяне посредством двойно свързани списъци. Описани са алгоритмите на две задачи, разработени от обучаващ и двама обучаеми. Представени са съответните им графи на протичане и е изчислена цикломатичната им сложност. Направени са изводи относно надеждността и приложимостта на предложения подход.*

Ключови думи: *Компютърни науки, Алгоритми, Среди за обучение по алгоритми, Начинаещи програмисти, Софтуерни технологии, Граф на протичане, Цикломатична сложност*

ISSN 1314-3077