Technium Vol. 2, Issue 5 pp.161-171 (2020)
ISSN: 2668-778X
www.techniumscience.com

The Algorithms and Data Structures course multicomponent complexity and interdisciplinary connections

Gregory Korotenko

Dnipro University of Technology, Ukraine
korotenko.g.m@nmu.one

Leonid Korotenko

Dnipro University of Technology, Ukraine
leonid_korotenko@ukr.net

Abstract. The experience of modern education shows that the trajectory of preparing students is extremely important for their further adaptation in the structure of widespread digitalization, digital transformations and the accumulation of experience with digital platforms of business and production. At the same time, there continue to be reports of a constant (catastrophic) lack of personnel in this area. The basis of the competencies of specialists in the field of computing is the ability to solve algorithmic problems that they have not been encountered before. Therefore, this paper proposes to consider the features of teaching the course Algorithms and Data Structures and possible ways to reduce the level of complexity in its study.

Keywords. learning, algorithms, data structures, data processing, programming, mathematics, interdisciplinarity.

1. Introduction

The analysis of electronic resources and literary sources allows for the conclusion that the Algorithms and Data Structures course is one of the cornerstones for the study programs related to computing. Moreover, this can be considered not only from the point of view of entering the profession, but also from the point of view of preparation for an interview in hiring as a programmer, including at a large IT organization (Google, Microsoft, Apple, Amazon, etc.) or for a promising startup [1]. As a rule, this course is characterized by an abundance of complex data structures and various algorithms for their processing. This article describes one of the possible approaches to teaching the Algorithms and Data Structures course to students of Ukrainian universities in order to provide training for future IT professionals.

The fundamentals of computer science constitute an integral component of computing since they support all the processes of designing and constructing software products. Despite the constant increase in the number of training courses programs and their elements, algorithms are the fundamental basis for computer science and software development [2].

For example, Curriculum Guidelines for Undergraduate Degree Programs in Computer Science [2] include the Algorithms and Complexity Knowledge Area to study algorithms and data structures, which contains the following subsections: Basic Analysis, Algorithmic

Strategies, Fundamental Data Structures and Algorithms, Basic Automata, Computability and Complexity, Advanced Computational Complexity, Advanced Automata Theory and Computability, Advanced Data Structures, Algorithms, and Analysis.

The content of the Computing Essentials Knowledge Area in the Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering [3] is somewhat less congested. Among the various elements, purely algorithmic components include: Programming Fundamentals (control and data, typing, recursion), Algorithms, data structures, and complexity, Construction technologies.

The Higher Education Standards of Ukraine [4], which determine the basic competencies related to the results of undergraduate studies in the 12 Information Technologies body of knowledge provide that the last components are implemented in the course of studying disciplines (courses) being part of the educational and professional programs curricula. As a rule, the curricula for all six majors in this body of knowledge (121 Software Engineering, 122 Computer Science, 123 Computer Engineering, 124 System Analysis, 125 Cybersecurity, 126 Information Systems and Technologies), include three courses for the study of algorithms and data structures: Algorithmization and Programming, Algorithms and Data Structures and Object-oriented Programming. The first course usually involves the solution of educational as well as practical problems by means of any of the commonly used programming languages. The second course is implemented as a description of the use of one of the popular programming languages (Java, C #, Python, etc.) for the implementation of abstract data types based on a variety of built-in data structures, as well as the construction of algorithms on their basis for processing the elements, data stored in them with algorithm complexity analysis. The third course considers the concepts of object-oriented programming (encapsulation, inheritance and polymorphism) based on the creation and use of classes and objects, operations overloading, multiple inheritance of virtual functions and classes, as well as abstract classes [1].

2. Description of the method

Any course studied at the University exists in the structure of many constraints and influencing factors. On the one hand, it is a component of a certain undergraduate program and represents a block either inside or outside the body of knowledge of a particular specialty. On the other hand, this course is influenced by rapidly developing components of that body of knowledge which future specialists are trained for. And an important component of the learning process is the educational and methodological support of the course, which recommends the most effective, rational options, action patterns in relation to the type of activity studied therein. Therefore, it has been proposed to analyze the level of elements complexity by means of analytical and graphoanalytical methods.

3. Case study

At the first stage, the following approach has been used in order to try to assess the structure and possible theoretical filling of the field of knowledge Algorithms and Data Structures. Four books were selected from a wide range of books by well-known authors that had become textbooks [5] and which were known because they were used to develop courses of the same name taught at renowned universities: Massachusetts Institute of Technology, Princeton University, State University of New York, Cornell University and Stanford University [6, 7, 8, 9].

The first stage analysis of the structural construction of the content of these books has been conducted, on the basis of which a list of the main topics presented in them has been created

(Table 1). For each topic, the number of pages allocated for the presentation of the corresponding part of the material has been determined. When creating this table, it has been taken into account that the algorithms shall be considered as the same technological product, such as hardware, a graphical user interface, object-oriented systems or networks [6].

Table 1. Distribution of topics by page

Sl. No.	Learning topics	T.Cormen, pages	R.Sedgewick, pages	S.Skiena, pages	A.Aho, pages
1	Introduction to Algorithm Design	40	–	33	10
2	Analysis of algorithms	22	69	34	14
3	Abstract data types	–	56	0	6
4	Elementary Data Structures	21	52	48	30
5	Sorting	82	116	29	41
6	String Sorts	–	30	–	–
7	Sorting algorithms and priority queues	–	53	–	–
8	Searching	–	–	13	–
9	Searching in Hash tables	33	34	–	24
10	Searching in Symbol Tables	–	34	–	–
11	Combinatorial Search and Heuristic Methods	–	–	43	–
12	Binary Search Trees	22	27	–	33
13	Balanced Search Trees	–	34	–	29
14	Trie Search Trees	–	26	–	–
15	Substring Search	–	29	–	–
16	Regular Expressions	–	21	–	–
17	Red-Black Trees	31	–	–	–
18	Elementary Graph Algorithms	35	86	86	29
19	Minimum Spanning Trees	19	34	48	7
20	Shortest Paths	41	56	–	6
21	All-Pairs Shortest Paths	24	–	–	8
22	Maximum Flow	61	–	–	–
23	Graph Hard Problems	–	–	39	23
24	Dynamic Programming	55	–	43	11
25	Matrix Operations	30	–	41	–
26	Set and String Problems	29	–	37	–
27	Computational Geometry	34	–	58	–
28	Approximation Algorithms	37	–	40	–
29	Sets, Relations, Functions, Graphs, Trees & etc.	25	–	27	16
30	Divide-and-Conquer. Recurrences & Recursions	49	–	–	19
31	Greedy Algorithms	37	–	–	4
32	Augmenting Data Structures	18	–	–	–
33	Amortized Analysis	30	–	–	–
34	Advanced Data Structures. B-Trees	21	–	–	–
35	Advanced Data Structures. Fibonacci Heaps	26	–	–	–
36	Advanced Data Structures. van	30	–	–	–

Sl. No.	Learning topics	T.Cormen, pages	R.Sedgewick, pages	S.Skiena, pages	A.Aho, pages
	Emde Boas Trees				
37	Data Structures for Disjoint Sets	26	–	–	–
38	Linear Programming	55	–	–	–
39	Polynomials and the FFT	28	–	–	–
40	Probabilistic Analysis and Randomized Algorithms	33	–	–	–
41	Multithreaded Algorithms	41	–	–	–
42	Number-Theoretic Algorithms	59	–	–	–
43	NP-Completeness	58	–	–	–
44	Counting and Probability	34	–	–	–

A radar chart has been constructed (Fig. 1) to implement the graphical method to analyze the data obtained.

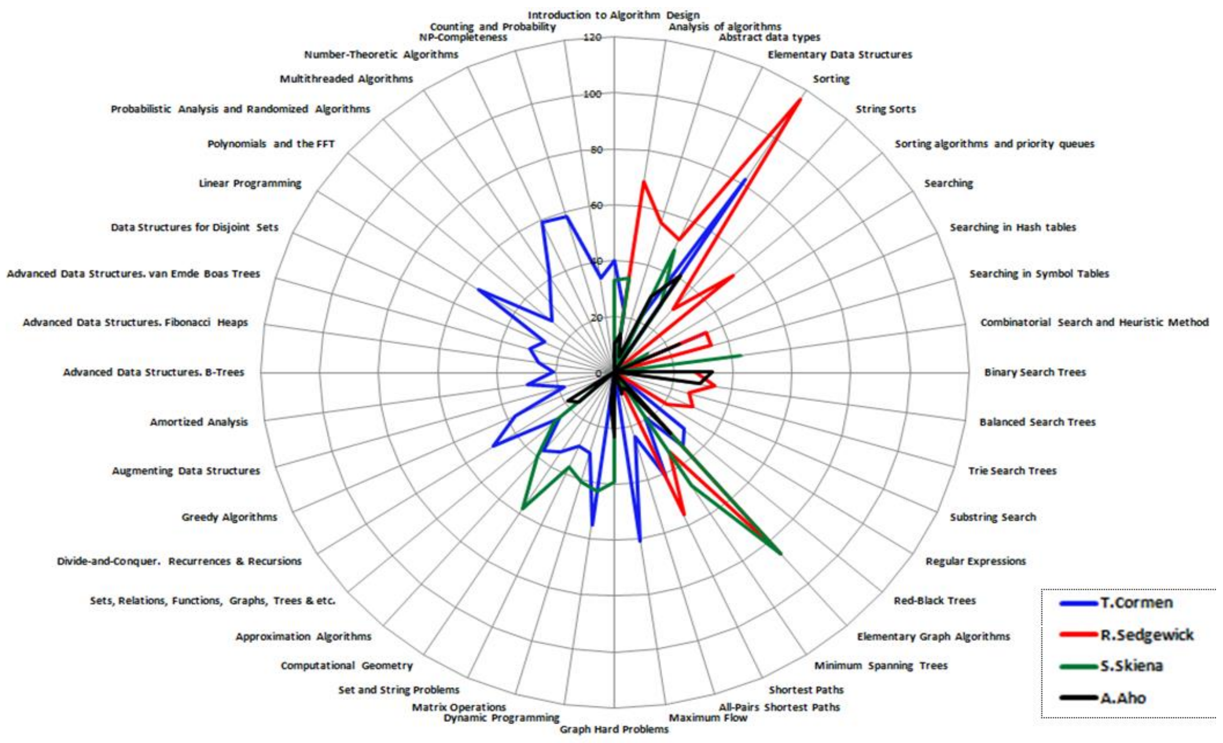


Fig. 1. The absolute distribution of the values of the number of pages of four authors' books, which relate to the topics communicated

Graphoanalysis of the absolute number of pages of the topics presented has been implemented to make more objective analysis. For this, a number of pages of each author has been taken as 100% (Fig. 2).

Based on the analysis of the results obtained, it can be concluded that the number of topics is large enough and this does not allow the authors to sufficiently cover each of them.

Moreover, an analysis of the approaches in the presentation of the material of the above-mentioned authors allowed to draw the following conclusions.

1. The books require students to know one or more high-level programming languages and, at the same time, use different tools for representing algorithms:

a) pseudo-code [6];

b) a subset of the Java programming language, several copyright libraries for input / output and for statistical calculations; Java ADT implementations shall be performed by defining an application programming interface (API), and then the Java class mechanism shall be used to develop and implement the use in a client code [7];

c) pseudocode and language C [8];

d) Pascal (programming language) [9].

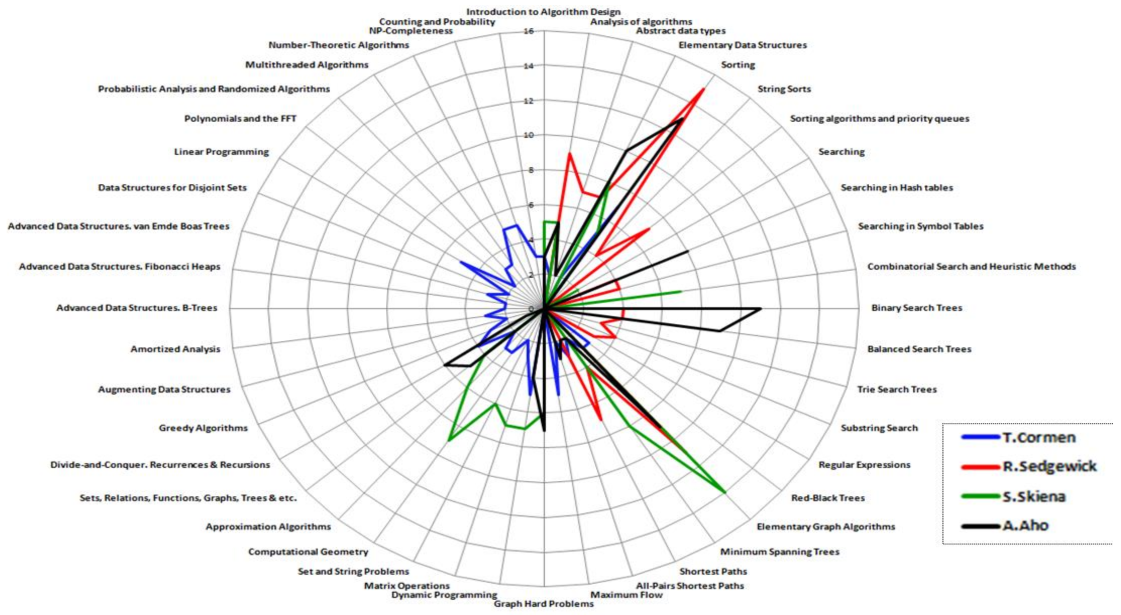


Fig. 2. Distribution of absolute values of the number of pages by topics under consideration

2. The concept of an algorithm in these books is defined as:

a) any correctly defined computational procedure, at the input of which a certain quantity or set of values shall be supplied and the result of which is an output value or set of values; the algorithm can also be considered as a tool designed to solve a properly posed computational problem [6];

b) a method to solve a problem, suitable for implementation in the form of a computer program [7];

c) a procedure that takes any of the possible input instances and converts it in accordance with the requirements specified in the condition of an objective [8].

d) a finite sequence of instructions, each of which has a clear meaning and can be performed with finite computational costs in a finite time [9].

3. The idea of data structures is given in the books as follows:

a) a method of storing and organizing data that facilitates access to these data and their modification [6];

b) algorithms are methods of organizing the data involved in computing; objects created in this way are called data structures [7];

c) *data structures* are used to represent the ADT, which are a set of variables, possibly various types of data, combined in a certain way [9].

4. Abstract data type (ADT) is also defined differently:

a) it is a data type (a set of values and a set of operations for these values), access to which is carried out only through the interface. The program that uses the ADT will be called the *client*, and the program that contains the specification of this data type will be the *implementation* [7];

b) it is a mathematical model with a set of operators defined within a framework of this model. A simple example of an ADT is a set of integers with the operators of union, intersection, and difference of sets [9].

The complexity of the newly created and accumulated sets and complexes of various data requires the development of new structures for their storage and processing. More than 15 types of data storage structures have been known at the beginning of 2019 (Table 2). The total number of actually elementary structures and types has exceeded 200 units, for each of which some finite number of processing algorithms can be applied [10-12].

In turn, the implementation of each of the well-known structures is possible in more than 50 of the most popular programming languages. This makes the task of forming structures and algorithms for their processing quite complex and non-trivial.

Table 2. Types of data storage element structures

Sl. No.	**Type of data structure**	**Examples of structures**	**Total Qty**
1	Primitive pes	Boolean, Character, Floating-point, Integer & etc.	**7**
2	Composite types	Array, Structure, Union	**3**
3	Abstract data types (ADT)	Container, List (sequence), Tuple, Multimap, Set, Multiset (bag)., Stack, Queue, Double-ended queue (deque) & etc.	**10**
4	Linear data structures	Bit array, Bit field, Bitboard, Bitmap, Circular buffer, Control table, Dope vector, Hashed array tree, Iliffe vector, Hashed array tree & etc.	**19**
5	Lists	Doubly linked list, Array list, Linked list, Self-organizing list, Skip list & etc.	**13**
6	Heaps	Heap, Binary heap, B-heap, Weak heap, Fibonacci heap, Leonardo Heap, 2-3 heap & etc.	**18**
7	Binary trees	AA tree, AVL tree, Binary search tree, Cartesian tree, Left-child right-sibling binary tree, Red–black tree Tango tree & etc.	**24**
8	B-trees	B-tree, B+ tree, B*-tree, B# tree (B sharp tree), Dancing tree & etc.	**11**
9	Trees	Radix tree, Parent pointer tree, Splay tree, Suffix tree (PAT tree), Suffix array & etc.	**15**
10	Multiway trees	Ternary tree, K-ary tree, And–or tree, (a,b)-tree, Link/cut tree, Spaghetti stack & etc.	**14**
11	Space-partitioning trees	Segment tree, Interval tree, Range tree, K-d tree, Quadtree, Relaxed k-d tree & etc.	**28**
12	Application-specific trees	Abstract syntax tree, Parse tree, Decision tree, Minimax tree, Finger tree & etc.	**10**
13	Hash-based structures	Bloom filter, Distributed hash table, Double hashing, Dynamic perfect hash table, Hash list, Hash tree & etc.	**16**
14	Graphs	Graph, Adjacency list, Adjacency matrix, Graph-structured stack, Scene graph, Multigraph, Hypergraph, Directed graph & etc.	**14**
15	Other data structures	Lightmap, Winged edge, Quad-edge & etc.	**5**
		Totally:	**207**

It is also important to note that the formation of derived structures is based on basic data types. Thus, many data processing operations fall into two groups:

a) operations with values stored in structural elements;

b) operations with the structural elements themselves (tops of trees and graphs, elements of arrays and stacks, nodes of lists and queues, etc.).

At the same time, the requirement for first-year students to know different branches of mathematics and have experience in mathematical transformations, as well as knowledge at a certain level of programming languages (PL) of a high level [6-9] seem quite problematic, since a deeper study of PL is carried out at an undergraduate level.

When using programming languages, in turn, it is necessary to take into account the following important factors that determine the thinking style of programmers, as well as many additional difficulties that arise at different stages of algorithms and data structures implementation:

a) existing programming paradigms, the number of which has already exceeded fifteen denominations [13];

b) a large number of programming languages, among which fairly new languages can be distinguished (Rust, Swift, Go etc.) [14];

c) intralingual features based on the particular paradigms to which they relate;

d) various types of operations and different levels of priority for their implementation.

A peculiarity of mastering the Algorithms and Data Structures course is the complexity of the organization types of the means for presenting various data and the possible operational and algorithmic capabilities of specific programming languages (PL) used at different stages of achieving the objectives defined.

The programming language is essentially an intermediary between specific structures for storing data elements and their scope (development of Web applications, databases, object-oriented information systems, e-commerce systems, etc.). On the other hand, in addition to the ever-expanding range of specific programming languages (JavaScript, Python, C ++, C #, Java, Ruby, Pascal, Visual Basic for Applications, Perl, etc.), there is an increasing number of:

a) constantly evolving hardware components;

b) information technology and digital platforms;

c) data structures of ever-increasing complexity;

d) constantly updated tools of Rapid Application Development – RAD.

Moreover, different languages have different number of operations (operators) for working with data (for example, multiplication, division, addition, subtraction, etc.), as well as the number of priorities for their use (Table 3).

Table 3. Programming language operations and their priority groups

The name of programming language	**The number of operators / operations**	**The number of priority levels of operations**
Turbo Pascal 7.0	20	4
Visual Basic 6.0	23	4
Visual Basic for Applications	23	4
VBScript	23	9
Python	35	23
C	43	15
Ruby	43	23

Java	44	9
Java Script	48	14
C#	51	14
C++	52	18
Perl	64	17

It is also necessary to take into account that a variety of program units representing the components of data structures. Blocks in programming [15] use different syntax in different programming languages:

a) in Pascal a function or procedure using operator brackets begin ... end;
b) in C a block limited by braces;
c) in Python a block using spaces as delimiters, etc.

4. Conclusions

During the academic year 2019-20 the authors have developed a training Algorithms and Data Structures course for the 1st year students in accordance with the curricula of educational and professional programs of several specialties (121 Software Engineering, 122 Computer Science, 123 Computer Engineering, 124 System Analysis, 125 Cybersecurity and 126 Information Systems and Technologies) in 12 Information Technology body of knowledge based on the following ideas.

C ++ has been chosen as the main programming language. The main property for this choice is the presence of a procedural paradigm, which is not possessed, for example, by Java and C#. This ensures a comfortable entry of students into the specialty, starting with the supporting Algorithmization and Programming course. The second important feature is the presence of a large number of C-like languages [16].

Accordingly, as of June 2020, TIOBE [17], PYPL [18] and RedMonk [19] services reported the popularity of programming languages as follows (Table 4), where C-like languages are highlighted in gray.

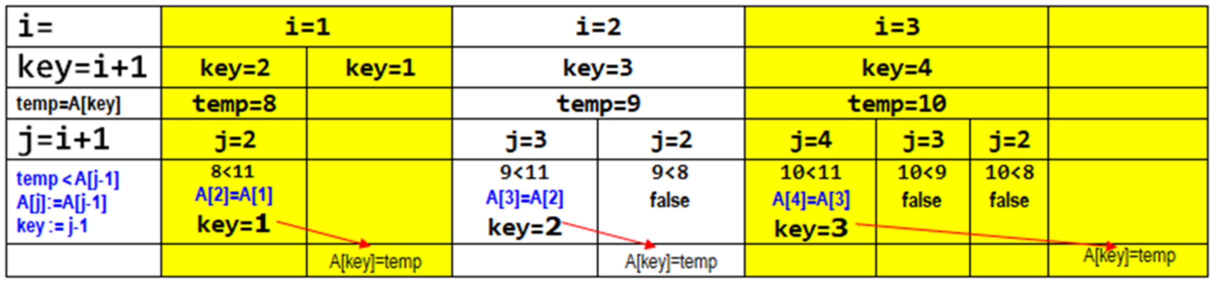
Table 4. Programming language ratings according to TIOBE, PYPL and RedMonk

Place	TIOBE June 2020	PYPL June 2020	RedMonk January 2020
1	C	Python	JavaScript
2	Java	Java	Python
3	Python	Javascript	Java
4	C++	C#	PHP
5	C#	PHP	C#
6	Visual Basic	C/C++	C++
7	Javascript	R	Ruby
8	PHP	Objective-C	CSS
9	R	Swift	TypeScript
10	SQL	TypeScript	C
11	Swift	Matlab	Swift
12	Go	Kotlin	Objective-C
13	Ruby	Go	Scala
14	Assembly language	VBA	R
15	MATLAB	Ruby	Go
16	Perl	Scala	Shell
17	PL/SQL	Visual Basic	PowerShell
18	Scratch	Rust	Perl
19	Classic Visual Basic	Dart	Kotlin
20	Rust	Perl	Haskell

Based on the provisions of the Böhm–Jacopini theorem (structured program theorem) [20], the authors have focused on the basic structures of the C++ language: SEQUENCE, SELECTION and ITERATION. At the same time, it has been taken into account that SEQUENCE construct in C++ language does not belong to the group of control structures, which include the if, if / else, switch, for, while, do / while statements. Therefore, commenting on the functional purpose of these data structures has been highlighted in separate subsections. The proposed approach has been applied with the intention that the didactic "highlight" for the student shall be the fact that he/she should think about its purpose when commenting on the introduced structure.

Since the teaching experience has shown low informativeness of the pseudocode, the emphasis has been made on programs with detailed comments. The development of students' approaches and skills in designing software units based on *reading, studying and understanding* other person's code as the most important component of team software development has been put at the forefront. For this purpose, each section of the laboratory workshop has contained programs with detailed comments on each line [21].

The commented programs implementing specific algorithms have been added with their table-graphical representation. In particular, each of the sorting methods studied in the methodological materials has been presented in two forms: program and table-graphic (Fig. 4).



i=	i=1		i=2		i=3			
key=i+1	key=2	key=1	key=3		key=4			
temp=A[key]	temp=8		temp=9		temp=10			
j=i+1	j=2		j=3	j=2	j=4	j=3	j=2	
temp < A[j-1] A[j]:=A[j-1] key := j-1	8<11 A[2]=A[1] key=1		9<11 A[3]=A[2] key=2	9<8 false	10<11 A[4]=A[3] key=3	10<9 false	10<8 false	
		A[key]=temp		A[key]=temp				A[key]=temp

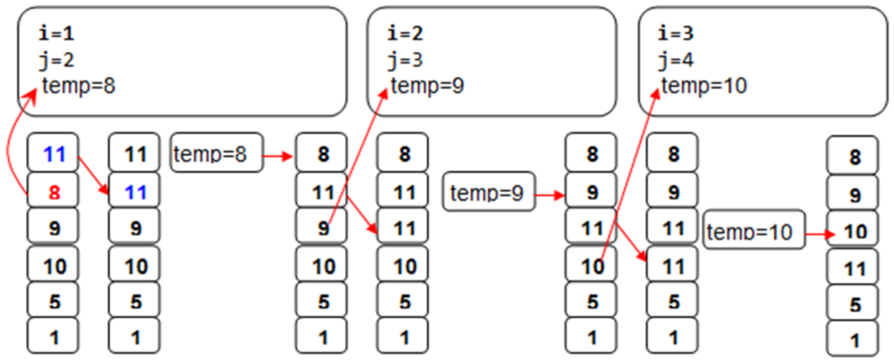


Fig. 4. One example of a tabular graphical representation of the beginning of an insertion sorting

The structure and format of comments in the computer programs offered for learning by students have been based on the well-known standards: "Coding standards on 64-bit platforms IBM Knowledge Center" (IBM) and "Google C ++ Style Guide" (Google) [22, 23]. Moreover, the authors have used the so-called CamelCase to describe variables. The main provisions of the Google standard have been taken as a basis. Unfortunately, it has been necessary to exclude some points related to the use of OOP since the course was presented for first-year students.

Based on the experience of teaching this course and taking into account the interdisciplinarity of its components [24], the Algorithmization and Programming, Algorithms and Data Structures and Object-oriented Programming courses has been proposed to be considered as a continuous chain. The Algorithmization and Programming course requires to immediately focus on the students' understanding of operations with embedded data of various types: integer, real, logical, symbolic, pointers, etc. Then it is possible to proceed with the students to study arrays of embedded data, including dynamic ones. Various sorting methods can be considered at this stage. When considering functions, recursion shall be considered. And when considering structures – stacks and lists shall be considered.

Then in the Algorithms and Data Structures course the attention can be drawn to the following issues: Sorting, String Sorts, Sorting algorithms and priority queues, Searching, Searching in Hash tables, Searching in Symbol Tables, Binary Search Trees, Balanced Search Trees, Elementary Graph Algorithms, Minimum Spanning Trees, Shortest Paths etc.

And the implementation of basic structures according to [1] and practical examples of their application shall be considered in the Object-Oriented Programming course.

It is also necessary to provide support for the above-mentioned courses in such field of knowledge as Mathematical Analysis.

In addition, one of the possible solutions in the educational space of Universities may be the formation of modern algorithmic competencies on the basis of the Special Structures and Data Processing Algorithms with OOP Elements sample course.

References

[1] G.L. McDowell: Cracking the Coding Interview. 6th Edition. 189 Programming Questions and Solutions. CareerCup, LLC, Palo Alto, CA. 2016.

[2] Computer Science Curricula 2013 (CS 2013); Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, Association for Computing Machinery (ACM), IEEE Computer Society, available at https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf (accessed 18 June 2020).

[3] Software Engineering 2014 (SE 2014); Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, Association for Computing Machinery (ACM), IEEE Computer Society, available at https://www.acm.org/binaries/content/assets/education/se2014.pdf (accessed 18 June 2020).

[4] Approved standards of higher education. MES of Ukraine, available at https://mon.gov.ua/ua/osvita/visha-osvita/naukovo-metodichna-rada-ministerstva-osviti-i-nauki-ukrayini/zatverdzheni-standarti-vishoyi-osviti (accessed 18 June 2020).

[5] 10 Data Structure & Algorithms Books Every Programmer Should Read, available at https://hackernoon.com/10-data-structure-algorithms-books-every-programmer-should-read-d50487313127 (accessed 18 June 2020).

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein: Introduction to Algorithms, Third Edition. The MIT Press. Massachusets Institute of Technology, 2009.

[7] R. Sedgewick, K. Wayne: Algorithms. Fourth Edition. Addison-Wesley, 2011.

[8] S.S. Skiena: The Algorithm Design Manual, Second Edition. Springer-Verlag London

Limited, 2008.
[9] A.V. Aho, J.E. Hopcroft, J.D. Ullman: Data Structures and Algorithms. Addison-Wesley, 1985.
[10] List_of_data_structures, available at https://en.wikipedia.org/wiki/List_of_data_structures (accessed 18 June 2020).
[11] Fundamental Data Structures, available at http://www.sncwgs.ac.in/wp-content/uploads/2015/11/Fundamental-Data-Structures.pdf (accessed 18 June 2020).
[12] Advanced Data Structure and Algorithms. EXCEL BOOKS PRIVATE LIMITED, Phagwara Punjab, India, 2011.
[13] Programming paradigm, available at https://en.wikipedia.org/wiki/Programming_paradigm (accessed 18 June 2020).
[14] Programming languages: Rust enters top 20 popularity rankings for the first time, available at https://www.zdnet.com/article/programming-languages-rust-enters-top-20-popularity-rankings-for-the-first-time/ (accessed 18 June 2020).
[15] Block (programming), available at https://ru.qwe.wiki/wiki/Block_(programming) (accessed 18 June 2020).
[16] List of C-family programming languages, available at https://en.wikipedia.org/wiki/List_of_C-family_programming_languages (accessed 18 June 2020).
[17] Programming Languages: TIOBE Index June 2020, available at https://www.geeks3d.com/forums/index.php/topic,6508.0.html (accessed 18 June 2020).
[18] PYPL PopularitY of Programming Language Index. Worldwide, Jun 2020 compared to a year ago, available at http://pypl.github.io/PYPL.html (accessed 18 June 2020).
[19] The RedMonk Programming Language Rankings: January 2020, available at https://redmonk.com/sogrady/2020/02/28/language-rankings-1-20/ (accessed 18 June 2020).
[20] E. Yourdon: Program Structure and Design. Prentice-Hall, 1975.
[21] R.L. Glass: Facts and Fallacies of Software Engineering. Addison-Wesley Professional, 2002.
[22] Google C++ Style Guide, available at https://google.github.io/styleguide/cppguide.html (accessed 18 June 2020).
[23] The Stanford University C++ Style Guide, available at https://hownot2code.com/2017/01/18/the-stanford-university-c-style-guide/ (accessed 18 June 2020).
[24] Moti Nissani. Fruits, Salads, and Smoothies: A Working Definition of Interdisciplinarity. The Journal of Educational Thought (JET) / Revue de la Pensée Éducative. – Vol. 29, No. 2 (August, 1995), pp. 121-128. available at https://www.jstor.org/stable/23767672 (accessed 18 June 2020).