Olympiads in Informatics, 2012, Vol. 6, 178–191

Competitions' Tasks in Informatics for the "Pre-Master" Group of School Students

Emil KELEVEDJIEV[1], Zornitsa DZHENKOVA[2]

[1]*Institute of Mathematics and Informatics, Bulgarian Academy of Sciences*
Akad. G. Bonchev str., block 8, 1113 Sofia, Bulgaria
[2]*Ivan Vazov High School*
54 Mitko Palauzov str., 5300 Gabrovo, Bulgaria
e-mail: keleved@math.bas.bg, zornica.dzhenkova@gmail.com

Abstract. Some features of informatics tasks given at the Bulgarian National Competitions for the "pre-master" school students group (9–10th grades in 12 grade school, i.e., 16–17 year old students) are discussed. The study covers the period 2004–2011. The considered group of students is important, because among them will appear candidates for the national team in the next one or two years. While tasks for the "master" group are complicated and not easily made into a system, the tasks examined here are more suitable to be analyzed. In the paper, an attempt is made to arrange these tasks using keywords, and to introduce a coefficient for measuring relative difficulty. The work continues two previously published papers of the same authors in this journal, devoted to the tasks for the intermediate and youngest students.

Key words: tasks in competitive informatics, informatics for the school students.

1. Introduction

In recent years, the competitions in informatics have been continually expanding and involving various age groups. This process can be observed in Bulgaria, as well in many other countries in the world. An example of this development is the establishment in 2007 at Belgrad, Serbia, of a new kind regional Balkan Youth Olympiad in Informatics for students up to 15.5 years old, as well the International Autumn Tournament in Shumen, Bulgaria, starting in 2009, where the competition includes two age groups – juniors and seniors.

In Bulgaria since 2001, several age group systems have been applied to divide school students for the national informatics competitions (the Autumn, Winter, and Spring Tournaments, as well for the three rounds of the National Olympiad in Informatics). Starting in 2002, groups were introduced with letter names: A, B, C, and D, which comprised 11–12, 9–10, 7–8, and 4–6th school grades, respectively. Now, we have a slightly modified system with 5 age groups A, B, C, D, and E, that cover 11–12, 9–10, 7–8, 6, and 4–5th school grades, respectively.

2. Systematization and Classification

After accumulating an archive of enough tasks (Websites *Infoman* and *Infos*) previously given at competitions, it becomes possible to start attempting classification using keywords. Information obtained in this way may serve the needs of teaching, and also for the preparation of future competitive tasks. The produced table is presented at the end of this article. We have to mention that, with the development of informatics competitions in the world, research on competition tasks has appeared (Verhoeff, 2009). The authors of the present article continue here their own ongoing research (Kelevedjiev and Dzhenkova, 2008a, 2008b, 2009), where they covered age groups from 4 to 8th school grades (groups E, D and C).

Tasks for the age group considered here have an intermediate character in the interval between the easy for the group C and the essentially more difficult, with Olympic features tasks, for the group A. While the A group tasks often have complex nature, which leads to difficulties in their classification, the tasks of the group B in most of the cases are more easily identifiable.

The curriculum for the master group A in the Bulgarian olympiads coincides with the curriculum of IOI, while topics expected for tasks of the group B are easier. The reader may get some pictures of themes for this group in Table 1, where the curriculum for the Bulgarian National Training Camps is presented.

For our classification, as seen from the Table 2, the main subdivision is made by the 9 main characteristics presented below. An important note is that this subdivision is not a strict classification, because many of the tasks of a given type have indications of the tasks to another type – e.g., tasks of type "graphs" are often solved by a method of dynamic programming. We are reflecting this multiformity in the "addition" column of Table 2.

1) *Graphs* (20 tasks). From the widely developed algorithmic area of the theory of graphs, prevalent methods in considered tasks are for finding the shortest path, determining the connectivity, cycles, topological sorting and minimum spanning trees. A part of these tasks relate to geometrical graphs, and methods of solving are dynamic programming, divide and conquer, and etc.

2) *Dynamic programming* (20 tasks). In this section we treat tasks, which are directly related to this basic approach to the compilation of algorithms. We mention that tasks of the other sections can also often be solvable with this approach.

3) *Combinatorics* (12 tasks). Contains tasks related to non-equivalent configurations, coding, representation of numbers, etc. One of the tasks requires "only an output file".

4) *Geometry* (12 tasks). Covers topics related to polygon's area, convex hull, crossing line segments, etc. Some of these tasks are solved with the approach of recursion and with elementary numerical methods.

5) *Search and sorting* (10 tasks). This group is related to a broad theme, containing binary search, exhaustive search, application of greedy approaches and some elementary numerical methods.

6) *Arithmetic* (5 tasks). These tasks require knowledge for divisibility of numbers, fractions, numerical systems, etc.

7) *Games* (4 tasks). Small group containing tasks to search for strategies in games with used the tabular methods, odd-parity methods, etc.

8) *Data Structures* (4 tasks). Here are allocated tasks, in whose solution the main approach is aimed at organizing the data, with which to improve response time for necessary operations. We see that in the given tasks are used the following data structures: Queue, Pyramid, Priority Queue and System of Disjoint Sets ("Union-Find").

Table 1

Curriculum for the Bulgarian National Training Camps, group B

	Topics	Year	Hours
1	STL containers and iterators.	1	4
2	Permutations.	1	4
3	Combinatorial configurations. Encoding and decoding.	1	4
4	Data structures for set presentations.	1	4
5	Algorithmic geometry. Intersections of points and lines.	1	4
6	Algorithmic geometry. Polygons. Convex hull.	1	4
7	Hash tables.	1	4
8	Graphs. Bi-connectivity. Strong connectivity.	1	4
9	Graphs. Euler's and Hamilton's cycles.	1	4
10	Graphs. Minimum spanning trees.	1	4
11	Graphs. Critical paths.	1	4
12	Catalan's numbers.	1	4
13	Dynamic programming tasks.	1	4
14	Games. Min-max strategies.	1	4
15	Recursion tasks. Recurrences.	1	4
16	Strings. Searching. Distances.	1	4
17	Data compression. Huffman's codes.	1	4
18	STL algorithms.	2	4
19	Combinatorial configurations on sets.	2	4
20	Gray's sequences.	2	4
21	Data structures for string algorithms.	2	6
22	Introduction to networks and maximal flows.	2	4
23	Partitions of numbers.	2	4
24	Partitions of sets.	2	4
25	Matching in graphs.	2	4
26	Algorithmic geometry. Closets and farthest points.	2	4
27	Special trees.	2	4
28	Strings. Syntax analysis.	2	4
29	Graph coloring. Planar graphs. Geometric graphs.	2	4
30	Introduction to formal grammars and automata.	2	6
31	Recursion tasks.	2	4
32	Games. Alpha-beta pruning.	2	4
33	Dynamic programming tasks.	2	6
34	System of linear equations.	2	4

Table 2

Classification of the tasks given to National Competitions in Informatics in Bulgaria (2004–2011) for the group B

	Task & competition			x	y	k	Type	Addition
1	fib	R2	2010	64	25	-0.44	aritmetic	
2	div	AT	2004	10	25	0.43	aritmetic	divisibility
3	seq	WC	2008	3	90	0.94	aritmetic	digits
4	even	ST	2009	2	94	0.96	aritmetic	number systems
5	dfrac	R2	2008	1	76	0.97	aritmetic	fractions
6	polygon	WC	2011	17	25	0.19	geometry	
7	abc	AT	2006	26	42	0.24	geometry	recursion, fractions
8	green	AT	2005	37	62	0.25	geometry	polygon area
9	rtri	AT	2006	20	75	0.58	geometry	integer points
10	lines	AT	2008	10	77	0.77	geometry	segment intersection
11	hull	WC	2004	4	95	0.92	geometry	convex hull
12	segment	R2	2006	3	91	0.94	geometry	
13	plate	WC	2008	3	93	0.94	geometry	numerical method
14	points	R2	2004	1	95	0.98	geometry	
15	lines	AT	2007	0	79	1.00	geometry	integer points
16	mov	ST	2007	0	70	1.00	geometry	segment intersection
17	walk	R2	2005	0	100	1.00	geometry	numerical method
18	ban	ST	2011	50	27	-0.30	graphs	connectivity
19	apples	WC	2010	60	34	-0.28	graphs	dynamics in graphs
20	alei	WC	2011	43	53	0.10	graphs	cycles in graphs
21	repair	ST	2010	25	45	0.29	graphs	MST
22	nails	R2	2011	26	67	0.44	graphs	connectivity
23	wght	ST	2004	24	65	0.46	graphs	
24	circle	ST	2008	24	68	0.48	graphs	shortest paths, geometry
25	recon	AT	2010	17	58	0.55	graphs	divide and conquer
26	circle	R2	2007	17	70	0.61	graphs	topological sorting, geometry
27	rings	AT	2008	10	57	0.70	graphs	shortest paths
28	folders	AT	2005	10	75	0.76	graphs	trees
29	race	ST	2010	10	75	0.76	graphs	shortest paths
30	triangles	R2	2009	11	81	0.76	graphs	topological sorting, geometry
31	trees	ST	2007	4	40	0.82	graphs	trees
32	trip	ST	2006	5	66	0.86	graphs	
33	avio	R2	2004	5	92	0.90	graphs	shortest paths
34	amb	R2	2009	2	92	0.96	graphs	shortest paths
35	edge	WC	2008	1	76	0.97	graphs	cycles in graphs
36	graths	WC	2005	1	97	0.98	graphs	shortest paths
37	obez	ST	2005	0	100	1.00	graphs	
38	shift	R2	2010	53	46	-0.07	DP	
39	salam	ST	2011	40	36	-0.05	DP	
40	calc	R2	2011	26	42	0.24	DP	
41	skok	WC	2007	34	61	0.28	DP	
42	stambs	R2	2008	33	66	0.33	DP	
43	tab	R2	2007	22	59	0.46	DP	
44	balls	ST	2006	22	69	0.52	DP	
45	brd	R2	2004	13	71	0.69	DP	
46	sum	AT	2004	8	55	0.75	DP	
47	jobs	R2	2010	10	75	0.76	DP	

To be continued

Continuation of Table 2

	Task & competition			x	y	k	Type	Addition
48	palind	R2	2011	9	82	0.80	DP	
49	flat	ST	2009	5	62	0.85	DP	
50	tre	ST	2007	4	61	0.88	DP	
51	knapsack	WC	2010	5	81	0.88	DP	
52	march	WC	2005	6	91	0.88	DP	
53	points	AT	2010	5	84	0.89	DP	bit masks
54	jumps	AT	2007	3	83	0.93	DP	
55	plate	R2	2005	0	71	1.00	DP	
56	ldist	WC	2006	0	96	1.00	DP	strings
57	king	WC	2009	0	97	1.00	DP	
58	handshakes	AT	2009	22	45	0.34	other	
59	average	AT	2004	20	60	0.50	other	algebraic transformations
60	fib	AT	2006	13	73	0.70	other	algebraic transformations
61	moves	ST	2004	9	86	0.81	other	integer points
62	irc	ST	2004	0	100	1.00	other	
63	inter	WC	2009	0	100	1.00	other	programming language
64	game	WC	2004	27	64	0.41	games	
65	phrope	ST	2006	5	69	0.86	games	
66	even	R2	2009	4	90	0.91	games	
67	decbin	AT	2008	2	87	0.96	games	number systems
68	sum	WC	2007	53	29	-0.29	combinatorics	
69	coins	AT	2009	45	32	-0.17	combinatorics	
70	toto	ST	2010	37	57	0.21	combinatorics	
71	code	WC	2010	23	55	0.41	combinatorics	output only
72	keokak	R2	2006	13	83	0.73	combinatorics	
73	square	ST	2005	11	79	0.76	combinatorics	
74	substr	WC	2011	10	82	0.78	combinatorics	strings
75	six	ST	2005	8	88	0.83	combinatorics	
76	squ	ST	2009	8	91	0.84	combinatorics	non-equvalent configurations
77	cart	R2	2005	5	90	0.89	combinatorics	presentations of numbers
78	signals	WC	2005	4	91	0.92	combinatorics	coding
79	perm	AT	2007	1	27	0.93	combinatorics	
80	y1984	WC	2006	31	51	0.24	DS	union-find
81	music	WC	2007	21	61	0.49	DS	priority queue
82	compATition	AT	2010	2	92	0.96	DS	pyramid
83	n23	AT	2005	0	82	1.00	DS	queue, divisibility
84	common	ST	2011	63	13	-0.66	srch. & srt.	
85	sqsum	WC	2009	37	24	-0.21	srch. & srt.	search in a table
86	table	WC	2004	30	50	0.25	srch. & srt.	exaustive search
87	dist	R2	2007	23	58	0.43	srch. & srt.	
88	riddle	AT	2009	12	65	0.69	srch. & srt.	greedy, binary search
89	points	R2	2006	12	72	0.71	srch. & srt.	
90	parATo	ST	2008	8	80	0.82	srch. & srt.	
91	triangle	ST	2008	7	73	0.83	srch. & srt.	geometry
92	plates	R2	2008	4	94	0.92	srch. & srt.	
93	Festb	WC	2006	0	100	1.00	srch. & srt.	interpolation, binary search

Legend: R2 – Round 2 of the National Olympiad (District round), AT – Autumn Tournament, WC – Winter Competitions, ST – Spring Tournament, DS – data structures, srch. – search, srt. – sorting, DP – dynamic programming, MST – minimum spanning tree.

9) *Other* (6 tasks). There are tasks of various types, and it is difficult to classify them to any of the above types. To note, for some tasks it was necessary to apply secondary school algebraic technique and for some other tasks it was necessary to apply modeling.

Using tasks' names in the Table 2, the reader can find in Websites *Infoman* and *Infos* the full descriptions of the tasks, which are representatives of the above described types (with problem statements, translated also in English, test examples, authors' solutions and, in most cases, detailed analysis).

3. Examples of Tasks

As examples, we present abridged formulations of typical tasks, which represent the above groups. Task's numbers are taken from Table 2. The coefficient k is explained in the next section; a values of about 0.5 mean that the task is "typical".

3.1. *Graphs*

Example 1. Task 24, $k = 0.48$ (This task includes also geometric graphs, shortest path). In the plane are given n circles: $C_1, C_2, \ldots, C_n$. Consider the undirected graph with vertices at given circles and there exits an edge between two circles, when both circles have exactly two common points. Write a program that computes the number of edges in the shortest path from C_1 to C_n. The first line of standard input contains $n(2 \leqslant n \leqslant 1000)$; each of the next n lines contains 3 integers $x,\ y,\ r$, giving the coordinates of the center and radius of a circle (x, y are in the range $-10000, \ldots, 10000$, and $0 < r < 10{,}000$).

Example 2. Task 25, $k = 0.55$ (This task includes also "Divide and conquer"). The company of Peter has designed a new space station, composed of N modules, labeled from 1 to N. Some pairs of different modules are linked by corridors, such that it is possible to go from each module to each other by a unique path of corridors. The length of each corridor is a positive integer. There is no more then one corridor linking two modules. The chiefs of Peter would like to keep in secret the project. That is why Peter encoded the topology of the station giving, for each two modules, the distance between them (i.e., the sum of the lengths of the corridors on the unique path between the modules). Write a program to decrypt the coding of Peter and to reconstruct the topology of the station. The first line of the standard input contains the number N of the modules ($3 \leqslant N \leqslant 1024$). Then $N - 1$ lines follow. On the first of these lines, the distances from module 1 to modules 2, 3,. . . , N are given. On the second line are given the distances from module 2 to modules 3, 4,. . . , N, and so on. The last line contains the distance from module $N - 1$ to module N. All distances are positive integers not grater than 1024. The program has to print N lines. The first line has to contain the list of the neighbors of module 1, i.e., the modules that are linked with corridors to it. The list has to start with the number L of neighbors, followed by their labels, sorted in increasing order. All numbers has to be separated by single spaces. On the second row of the output, formatted in the same way,

the list of the neighbors of module 2 has to be printed, and so on. The output has to finish with the list of the neighbors of module N.

3.2. *Dynamic Programming*

Example 1. Task 43, $k = 0.46$. Given is a table with M rows and N columns ($0 < M < 1000, 0 < N < 1000$). Each cell contains an integer in a range from -100 till 100. We consider all sub-tables with m rows and n columns that contain successive rows and columns ($0 < m < M,\ 0 < n < N$). Write a program that inputs M, N, m, and n, the integers in the given table, row by row, and outputs the maximum sum of integers, which may be exist in some of the considered sub-tables.

Example 2. Task 44, $k = 0.52$. There are given n balls, arranged in a row along a straight line in a hall. Opposite, there are placed m empty baskets in a row parallel to the row of balls. Each ball and each basket is colored with one of the following colors: r – red, b – blue, g – green, w – white, p – pink, y – yellow. We play the game by taking successive balls in the order which they are arranged from left to right. We can return back a ball taken to the same place in its row, but we cannot get a ball that we have already put back, and we cannot go back in the row of the balls. A ball, that we do not put back, we put in the basket of the same color. This basket should be located just right of the last nonempty basket at the current moment (if we are at the beginning of the game, we may put the taken ball in any basket of the same color). Each basket can take at most one ball. The aim of the game is to score the maximum number of balls. Write a program that inputs two strings, containing some of the letters $r,\ b,\ g,\ w,\ p,\ y$. Each of the strings is no longer than 999 characters. The first string specifies the position of the balls and the second – the position of the baskets. The program has to output the maximum number of balls that can be placed in baskets according to the described rules.

3.3. *Combinatorics*

Example 1. Task 71, $k = 0.41$ (Output only). Consider all 5-letter words with letters 0, 1 or 2: $00000, 00001, 00002, \ldots, 22222$. The distance between two words is the number of corresponding elements which are different. For example, the distance between 01201 and 11011 is 3. Select as many words, so that the distance between any two of them is greater than or equal to 3. Save results to file.

Example 2. Task 72, $k = 0.73$. An artificial language has 4 sounds: A, E, O, and K. Each word may contain only those sounds and is correct, if the following four requirements hold: 1. A word has at least one vowel sound; 2. One after another cannot be two identical sounds; 3. A word has no more than two consecutive vowels; 4. The letter A cannot be followed by a vowel. For example, strings A, OA, EO, OKO, KOAK, AKEO and KAKA are correct words, but strings K, OKKA, KEOA, KOOK and KAO are not. A word is called "reversible" if it is correct and also correct when it read from right to left, also. For example, the words A, EO, OKO, AKEO, KAKA are reversible, and the words KOAK, OA and are not reversible. Write a program that inputs an integer n ($2 \leqslant n \leqslant 20$) and outputs the number of words with at most n letters that are not reversible.

3.4. *Geometry*

Example 1. Task 9, $k = 0.58$. A rectangle with side lengths m and n (m and n are integers, $0 < m < 30,\ 0 < n < 30$) is divided into mn squares, each with side 1. Each point that is a vertex of at least one of these squares, we will call a node. Write a program that inputs m and n, and outputs the number of rectangular triangles, whose vertices are nodes.

Example 2. Task 10, $k = 0.77$. In the plane are given N different line segments. Write program that computes which is the maximum number of the given line segment, that can be crossed by a vertical or horizontal line. The program inputs N, then N lines, each containing 4 integers with coordinates of segment's endpoints ($0 < N < 300000$, coordinates are in a range: $-10^8, \ldots, 10^8$).

3.5. *Search and sorting*

Task 87, $k = 0.43$. Given are positive integers M and N ($1 < M < 3000,\ 1 < N < 3000$) and we consider all points with integer coordinates (x, y) in the plane, for which $1 \leqslant x \leqslant M$ and $1 \leqslant y \leqslant N$. We construct all segments that have endpoints at two different considered points. Given is a decimal number $d,\ 0 < d < 10000$), and with at most 4 digits after the decimal point. Write a program that finds out a segment among the constructed ones which length is the closest to the value of d. A line of the standard input contains values of M, N and d. The program should output the matching value as a number with fixed decimal point with exactly 4 digits in its fractional part, obtained with rounding by cutting.

3.6. *Arithmetic*

Task 2, $k = 0.43$. Write a program, which inputs two integers N and P, $0 < N < 1000000001,\ 0 < P < 50001$, and outputs the largest integer M, such that $N!$ is divisible by P^M.

3.7. *Games*

Task 64, $k = 0.41$. Candies are divided into two piles, containing respectively A and B pieces. Two persons play a game with alternating moves, where each player on his turn must choose one of the piles and divide it into two parts. The other pile is given away and the game continued with both new piles. Of course, new piles may not have equal number of candies, but each must contain at least one candy. The game ends when the two new piles each contain one candy, i.e., the players cannot make more moves. The winner is the one, who made the last move. Write a program that inputs the number of candies A and B before the next move of a player $1 < A < 3001,\ 1 < B < 3001$). The program must determine whether the player who is to move loses the game if the opponent plays optimally (i.e., the best possible for himself to win) or the player can win no matter how

the other player plays. If the player, who is to move, loses the game, the program should output number 0, but if the same player can win, the program should output values C and D, which is both new piles after his move. Under the rules, each of C and D is greater than or equal to 1 and $C + D$ is equal to one of A, or B. Because the player may have several different possible moves, the program has to output those for which $C + D$ is the least possible and then to choose a move for which C is the least.

3.8. *Data Structures.*

Task 81, $k = 0.49$. (Priority queue) Each participant in a casting has an initial rating, which is expressed as a decimal fraction from 0 to 100, with an accuracy of 4 digits after the decimal point. The jury has to evaluate all participants for N days, $1 < N < 200$. The organizers made the casting, so that participants arrived in town only the night before the working day of the jury (including the night before the first day) and accommodated in a hotel. Every day, the jury chose to examine a number of participants, which was accommodated in the hotel, and which had the highest rating, but if there was such with equal rating, they are chosen according to the earlier registration at the hotel. Once a participant is examined, he immediately has to leave the hotel. The hotel has 10000 beds and never overflows during the time of the casting. There was left unexamined participants in the hotel after the Nth day, although new candidates were not accommodated at the night before the $(N + 1)$st day. Write a program that finds the name of the first unexamined participant, i.e., one that should be examined first in the $(N + 1)$st day, if the jury would have decided to work one day longer. The program inputs the number of days N, followed by N lines, each with a number M of participants arrived at the night before that day, $0 \leqslant M < 500$, followed by M pairs consisting of the name of a participant and his rating. The last line contains the number of participants K that the jury actually can examine for a day, $0 \leqslant K < 300$. Each name contains no more than 10 letters.

4. Assessment of the Relative Difficulty of a Task

Let us denote by x a percentage of the contestants received a score over 60 points (of 100 maximum possible) for a given task (i.e., solved the task "successfully") and by y a percentage of the contestants received a score less than 30 points (i.e., solved the task "failed"). We define a coefficient for **relative difficulty**: $k = (y - x)/(x + y)$.

This relative value varies between 1 and -1, and takes the its maximum when $x = 0$ and its minimum – when $y = 0$. In the first case the value is $k = 1$ and there are no competitors, which have solved the task "successfully" (all have worse results), and in the second case, when $k = -1$, no competitor has not solved the task "failed" (all have better results). So, when the value of k is close to 1, we can consider that the task is relatively difficult, but when $k < 0$, the task is easy.

From the presented data and graphics we may conclude, that at the national competitions in informatics very rarely are there tasks with a negative coefficient k. This is a natural expectation – at these competitions there are not given easy tasks. The defined

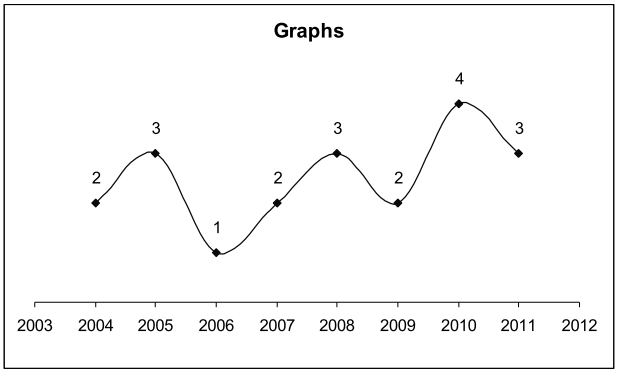

Fig. 1. Number of tasks of type "Graphs" in years.

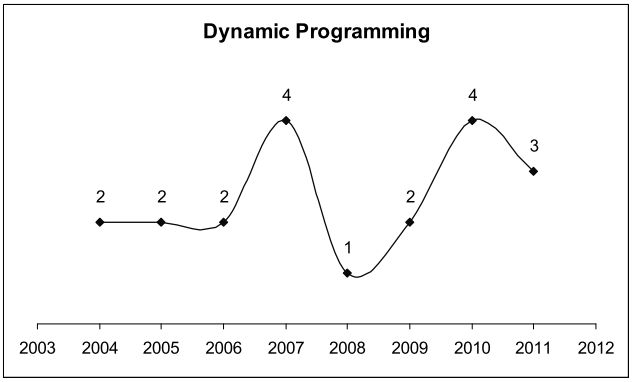

Fig. 2. Number of tasks of type "Dynamic Programming" in years.

coefficient has no direct connection with the absolute difficulty of the particular task, nor it is a measure of the feasibility of the competitors, but rather it shows to what extent the authors of the tasks are selected them in accordance with the age of the competitors.

We note that at the Bulgarian national competitions there is no rule that a tasks' set should contain a task that is easy (supposed to be solved by most participants) and a task that is intentionally chosen to be a hard.

5. Conclusion

The main topics in competition tasks for 9 and 10th school grades includes: Theory of Graphs and Dynamic Programming. To a lesser extent, there are tasks related to Combinatorics, Arithmetic, Search and Sorting, and Geometry. There is a trend to decrease in the number of tasks with geometrical nature, which is explained by the rare presence of these tasks in the international competitions.

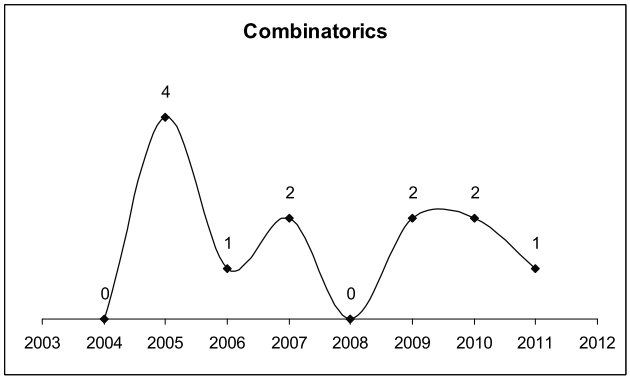

Fig. 3. Number of tasks of type ”Combinatorics” in years.

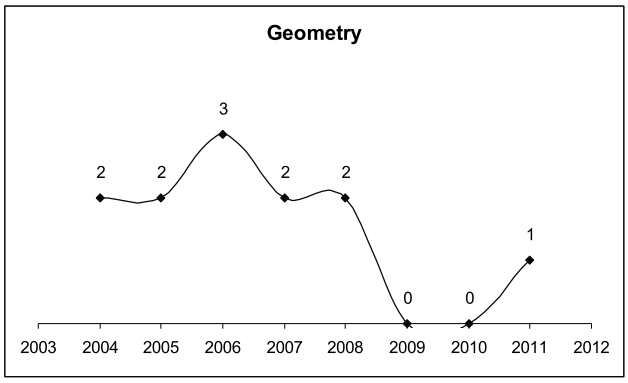

Fig. 4. Number of tasks of type ”Geometry” in years.

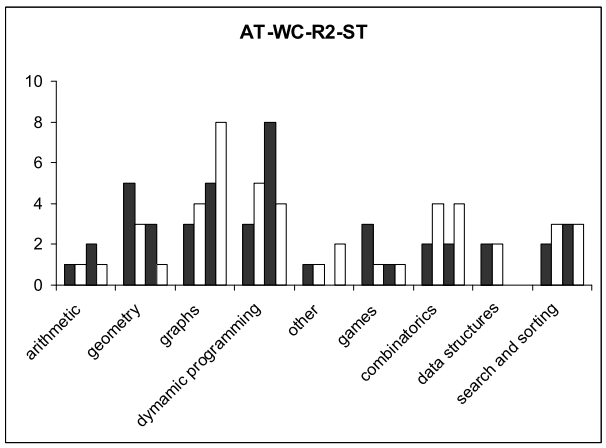

Fig. 5. Number of tasks according to their type, allocated in competitions (2004–2011).

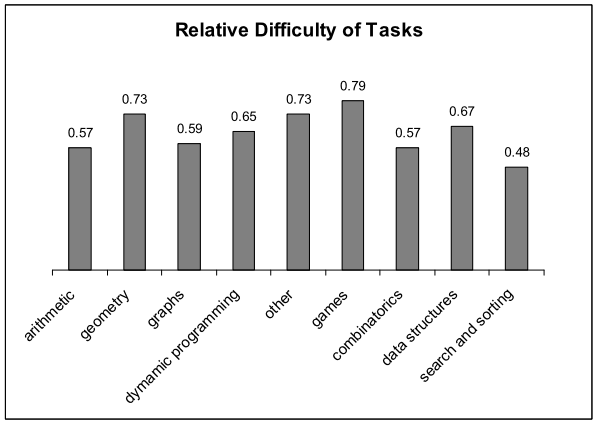

Fig. 6. Relative difficulty of the tasks, according to their type (2004–2011).

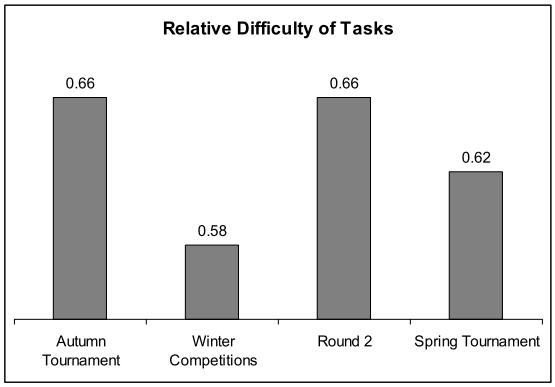

Fig. 7. Relative difficulty of the tasks, allocated in competitions.

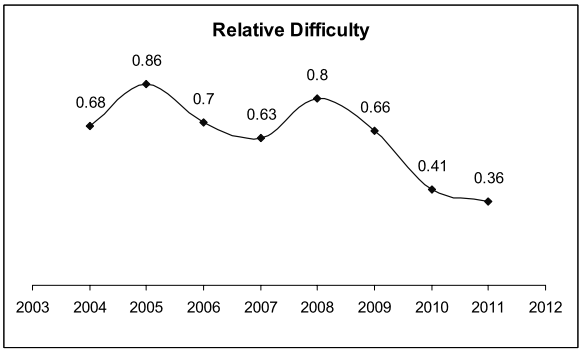

Fig. 8. Relative difficulty of all the tasks in years.

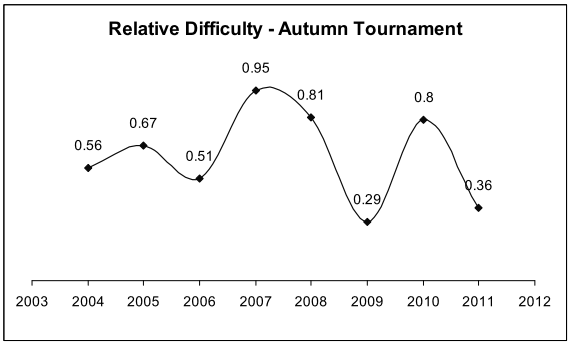

Fig. 9. Relative difficulty of the tasks, given at the Autumn Tournament in years.

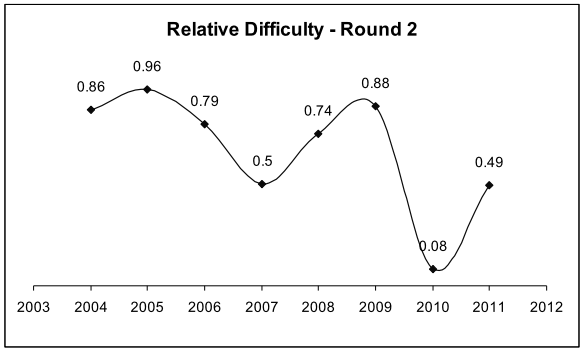

Fig. 10. Relative difficulty of the tasks, given at the Round 2 of the National Olympiad in years.

References

Kelevedjiev, E., Dzhenkova, Z. (2008a). Tasks and training the youngest beginners for informatics competitions. *Olympiads in Informatics*, 2, 75–89.

Kelevedjiev, E., Dzhenkova, Z. (2008b). Competition's tasks in informatics for the 4–7th school grades. *Mathematics, Informatics and Education in Mathematics and Informatics*, 37, 367–378 (in Bulgarian).

Kelevedjiev, E., Dzhenkova, Z. (2009). Tasks and training the intermediate age students for informatics competitions. *Olympiads in Informatics*, 3, 26–37.

Manev, K., Kelevedjiev, E., Kapralov, S. (2007). Programming contests for school students in Bulgaria. *Olympiads in Informatics*, 1, 112–123.

Verhoeff, T. (2009). 20 Years of IOI competition tasks. *Olympiads in Informatics*, 3, 149–166.

Website Infoman. `http://infoman.musala.com` (retrieved on 30 March 2012).

Website Infos. `http://www.math.bas.bg/infos` (retrieved on 30 March 2012).

E. Kelevedjiev is a research fellow in the Institute of Mathematics and Informatics at the Bulgarian Academy of Sciences. His field of interests includes algorithms in computer science, operation research, digitization techniques, etc. He was a member and now is the chairman of the Bulgarian National Committee for Olympiads in Informatics since 1993; leader of the Bulgarian teams for many IOI's and BOI's.

Z. Dzhenkova is a teacher in the Ivan Vazov High School in Gabrovo, Bulgaria. She is coauthor of a manual for beginner's training in competitions and Olympiads in Informatics. Her field of scientific interests includes Education in Informatics and Information Technology. She is a member of the Bulgarian National Committee for Olympiads in Informatics.