Olympiads in Informatics, 2018, Vol. 12, 43–52
© 2018 *IOI, Vilnius University*
DOI: 10.15388/ioi.2018.04

Algorithmic Cognition and Pencil-Paper Tasks

David GINAT
Tel-Aviv University, Science Education Department
Ramat Aviv, Tel-Aviv, Israel 69978
e-mail: ginat@post.tau.ac.il

Abstract. Pencil-paper algorithmics was displayed by several IOI studies, which examined various structural and scientific characteristics of offered tasks, including "what makes a good task". We offer an additional facet, of cognitive considerations. We underline the aspects of abstraction, heuristics, creativity, and declarative conceptions, which are relevant already with pencil-paper algoithmics. We describe the settings of our country's Stage-A, pencil-paper exam, and display cognitive considerations of the exam tasks. We illustrate these considerations with several single-input tasks, in which hidden patterns should be recognized for effective "computation by hand".

Keywords: algorithmic problem solving, cognitive aspects.

1. Introduction

Consider the following Reversing algorithmic task. Given the following sequence of twenty eight A/B characters: A B B B A B B A A B A A B A A B A B A A A B B A B B B A, what is the minimal number of `reverse-sub-sequence` operations necessary for obtaining the alternating sequence `A B A B` ... `A B`? (A `Reverse-sub-sequence` operation may be applied on any sub-sequence of consecutive characters; e.g. if applied on the first 4 characters A B B B it will yield B B B A.)

The task specification is short. It requires no special knowledge and involves a simple operation, to be repeatedly applied on a particular input. The input is not long, but still long enough so that one will have to reason in an ordered analysis, which will yield an operational computation from the starting point (the input) to the desired goal. Due to the limited size of the input, the operational computation can be performed by hand, without a computer.

More can be said. In solving the task, one should elevate the point of view and focus on particular task elements, while ignoring others. This involves *abstraction*. One should also cleverly process these elements. This involves *creative* employment of *heuristics*. And one should convince herself (even if intuitively) of optimality. This involves *declarative* conceptions. The cognitive aspects of abstraction, heuristics, creativity, and declarative conceptions are essential in algorithmic problem solving. In this paper, we relate them to the preliminary pencil-paper exam in algorithmics.

Previous IOI-related studies of pencil-paper algorithmics display and discuss tasks to be solved at home or in an actual on-site exam (e.g., Burton, 2010; Kubica and Radoszewski, 2010; van der Vegt, 2012; Radoszewski, 2014). Additional studies display a variety of related forms of non-programming activities (e.g., Dagienė, 2006; Dagienė and Futschek, 2008; Opmanis, 2009). These studies describe contest settings and offer task considerations, including those that "make good tasks". The considerations focus on task features of structural and scientific characteristics (including time frames, tools used, mathematical and algorithmic features, sub-tasks and more). In this paper, we relate to these characteristics, and add considerations of algorithmic cognition.

We introduce and discuss an approach of preliminary pencil-paper algorithmics, which comprises the initial OI activity in Israel. In the next section we motivate and describe the considerations underlying our approach. In the section that follows we illustrate the approach with different examples, additional to the Reversing task. In the last section we mention our implementation experience and enquire about the correlation and differences between mathematical competence and algorithmic competence.

2. Considerations of Pencil-Paper Algorithmics

We display below setting considerations and cognitive consideration that yield our approach of pencil-paper problem solving in algorithmics.

Setting Considerations

- ***Wide population.*** In Israel, a limited amount of students study high school computer science (CS), from 10-th grade. In the initial IOI activity we try to reach as many motivated students as possible, including young mathematics students who are not necessarily acquainted with programming.
- ***Two-fold goal.*** We aim at promoting interest in algorithmic challenges, as well as identifying competent algorithmics students. About 3000 students, nation-wide, show interest in our Stage-A.
- ***National 2-hour exam.*** We announce a national Stage-A exam in the beginning of the academic year. The exam takes place in the schools, under the supervision of teachers who print the exam just before it starts.
- ***Teacher involvement.*** We encourage teachers to take part, even if small. We need the teachers for encouraging students to prepare-for, and attend the exam.
- ***Student preparation.*** Students should have an idea of the exam format. We display in our website exams of previous years (from 2010).
- ***Four/five different exam tasks.*** We estimate 15 to 45 minutes for a task, in the 2-hour exam. The tasks are ordered according to their levels of difficulty. The first one is rather simple, so that students will feel that they managed to solve at least one task. Some questions are optimization questions, some involve a combinatorial computation, some display a two-player game and ask for the first

move, some are related to generic computational schemes, and some just require logical reasoning.

- ***Short and simple specifications.*** Reading and understanding a task takes time. We try to minimize this time. Task specifications are very short, often with a short illustration, and with a single input on which to perform a computation.
- ***Colourful challenges.*** The tasks pose challenge. Usually, there is no story in them, but students find the tasks colourful, due to their challenge.
- ***No programming knowledge.*** One may do well without programming background. In addition, programming knowledge in an early age often involves a lot of technical details, and this may not help much here. What may sometimes help is prior experience with problem solving.
- ***Single, non-trivial input.*** The vast majority of the tasks involve a single input. The input is in a size that requires insight and competence. One may sometimes guess an answer, but this is very unlikely to succeed with a set of tasks. When insight is obtained, a pencil-paper computation may be performed in a reasonable time. If no insight is obtained, the computation may take long time, or not take place at all.
- ***Single integer output.*** The answers of the few thousand students are submitted electronically. Due to the large amount of data, the task outputs are very short – usually a single integer per task. We believe that it is sufficient for evaluation.
- ***Hints for checking the output.*** Once insight is gained, a careful computation should yield the right result. Still, in order to help avoiding erroneous calculations, we provide hints, such as “The digits unit is odd” or “The output is a multiple of 5”.
- ***Partial credit.*** Since a task answer is a single-integer, there is usually no partial credit. Yet, there are exceptions. Occasionally, upon the electronic checking we notice that a group of students obtained the same result, which is different from the correct one. We figure out the rationale for this result, and if we realize that it may have been obtained from partial recognition of patterns, we give partial credit.
- ***Passing criteria.*** We choose about 300 students, out of up to 3,000 (the number varies in different years) based on their performance with the more challenging questions. These students are invited to a more thorough Stage-B exam.

Cognitive Considerations

Although the tasks are single-input/single-output tasks, solved with pencil and paper, we regard them as reflecting cognitive competencies that are essential for algorithmic problem solving. We believe that problem solvers should dedicate non-negligible time for solving a challenge. In a 2-hour exam, with 4 tasks (and perhaps a bonus one), one may have about 30 minutes per task, on the average. This may be sufficient, for a competent student in a pencil-paper stage, for gaining insight and capitalizing on it. In this amount of time one may employ relevant cognitive faculties.

- ***Abstraction.*** Algorithmic problem solving involves abstraction (e.g., Wing, 2006; Armoni *et al.*, 2006; Ginat and Blau, 2017). Abstraction may be expressed in a variety of forms. One may notice mapping (reduction) from a given question to another; or offer an illuminating representation that considerably simplifies the viewpoint on a given task; or focus on particular elements while ignoring others. In looking at the Reversing task, one should momentarily ignore the inner characters of a reversed sub-sequence and focus only on its ends, in order to notice the asset of concurrently "breaking" AA's and BB's.

- ***Heuristics and reasoning.*** Challenging tasks are solved by employing various kinds of heuristics, such as problem decomposition, backward reasoning, generalization, and more (Polya, 1945; Schoenfeld, 1992). Rigorous reasoning and case analysis are combined with the application of heuristics. Hidden patterns are unfolded. For example, in the Reversing task, careful reasoning/analysis may decompose the input disorders into two cases – the case of AA's and BB's and the case where the ends are improper (e.g., an A in the right end). A single `reverse` operation may concurrently "break" an AA and a BB. How should a disordered end be handled? Creativity may help here.
- ***Creativity.*** Solution processes require divergent thinking (e.g., Ginat, 2008). In doing so one may need to examine several solution directions, demonstrate flexible associations, and invoke original ideas. In attempting the case of disordered ends in the Reversing task, one may combine flexibility with the heuristic of auxiliary construction, and add an auxiliary A to the right end of the sequence. This will transform the end case into an AA/BB case.
- ***Declarative perspective.*** Algorithmic problem solvers naturally turn to operational reasoning, and go for the "how" computation. Yet, an operative perspective may be insufficient when one wants to be convinced of correctness and efficiency (e.g., Ginat, 2008). For this, one needs to see the declarative meaning, even if not formally, for believing correctness. In the Reversing task, in order to be convinced of optimality, one should notice that after the auxiliary construction (of an A added to the right end), the number of AA's is exactly equal to the number of BB's, and a single `reverse` operation may reduce at most 1 of each.

In the next section we demonstrate the above elements with additional tasks of different types, which were posed in our Stage-A exams during the last five years.

3. Illustrations

In the previous section we exemplified cognitive considerations with the Reversing task. We regard the Reversing task as relatively easy. It was one of the first two questions (out of four) in the 2012/13 Stage-A exam. The patterns to recognize are not immediate, but also not very challenging. The following is an additional task of limited challenge.

Sum of sub-sequences. In the following sequence of integers: 4 11 3 5 3 there are 4 sub-sequences of consecutive integers whose sums are multiples of 3. These are the sub-sequences: 4 11; 3 (the left 3); 4 11 3; 3 (the right 3). Notice that a single integer is regarded as a sub-sequence. So is the whole sequence. Given the following list of integers, output the total number of sub-sequences whose sums are multiples of 3?

6 1 4 124 3 6 512 3 1 33 2 2 32 100 813 4 41 1 8 213 5 7 61 8 42 1 4 2 20 8

<u>Hint</u>: the total sum is a multiple of 5.

This task was one of the first two tasks of the Stage-A exam of 2014/15. The challenge here is to devise a simple scheme that offers an ordered way of counting. Problem solvers may attempt various types of counting here. They may be based on two observations: **1.** Counting is simplified by examining remainders of 3; and **2.** An ordered counting may be carried out with a single pass over the input in which, for each integer – the number of sub-sequences that it "ends" will be added to a total sum. We may specify a corresponding declarative notion.

> *The number of sums that are multiples of 3, which an integer v ends, is equal to: the number of integers to the left of v in the input, for which the mod-3 remainder of the sum from the left-end to each of them equals the mod-3 remainder of the sum from the left-end to v.*

One does not need to specify the above notion explicitly, but should be able to see it (or an equivalent one) in order to apply an ordered operational computation with pencil and paper. The counting involves a feature of working backwards and an abstraction aspect in which the relevant sums are distinguished from all the possible sums. The hint may help avoiding calculation mistakes. A student that will not gain corresponding insight will face difficulties, and may spend a long time on the task.

* * *

The next task involves the two-player game of Chomp. This game was also posed in the Australian Informatics Competition with a 3×3 chocolate block (Burton, 2010). We displayed it in our Stage-A of 2017/18 in a different form.

Board game. Given a board of N×M squares, two players play against each other. Each player on her turn marks one of the rectangle squares. As a result, this square, and all the squares to its right and/or above are removed. The game ends when no squares remain. The player who makes the last move loses the game. If, for example, in the 3×4 board below the square F is marked, then the squares B, C, D, F, G, H will be removed. We assume that each player plays the <u>best</u> she can.

A	B	C	D
E	F	G	H
I	J	K	L

A. In the 2×10 board below, the first player will win the game if she marks in her first move one particular square. Which square should she mark?

A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T

B. Answer the same question for the upper 3×4 board.

We ask about the first move in a 2-player game in some of our Stage-A exams. Students who prepare, and look at previous exams should be ready for such questions. In part A one should examine small rectangles, such as 2×2, 2×3, and 2×4 boards, and generalize. The heuristic of generalization from simple cases will yield the following declarative, invariant pattern for winning as the first player.

> *After every move of mine, the top line will be a square shorter than the bottom line.*

One does not need to specify the invariant explicitly, but should see the pattern it involves. In part B, the board is smaller, and no generalization is needed. But thorough case analysis should be carried out. All in all, each part involves somewhat different competencies. In our experience, students felt enthusiastic about this task, though it was not trivial, and not fully solved by many.

* * *

The next task involves a combinatorial computation that may be solved in various ways. Combinatorial computations are common in algorithmic problem solving. They appear in various forms in some of our Stage-A exams. The following task was designed by our coaching team member Daniel Hadas, for Stage-A of 2017/18.

> **Watch colouring**. Given a round watch with the integers 1..12, and four different colours; what is the number of different ways to colour the 12 integers so that every two adjacent integers will be coloured with different colours, and every two opposite integers (e.g., 5 and 11) will be coloured with the same colour? Hint: the units digit of the answer is 2.

This task is harder that the previous ones. A student who studied combinatorics may have some advantage, but the solution will not be immediate. One may try diverse ways of case analysis, but should be careful not to count some colouring twice or miss a colouring. In our view, one needs to demonstrate creativity with suitable heuristics in order to simplify as much as possible the view of the task.

Since the colour of every two opposite integers in the clock is the same, we may reduce the task to the colouring of 6 integers in a circle. Counting would have been much simpler if the integers were in a line and not a circle, since a line has two explicit ends. The challenge here stems from the need to avoid colouring the two ends, 1 and 6 in the same colour. How should we do that? A creative embedment of the notion of "complement" in the heuristic of decomposition paves the way. We specify the relevant observation in an operative (rather than declarative) manner.

> *Rather than adding-up all the legal colouring cases, count all the ways to colour 1 to 6 in a line, and then* ***remove*** *the ways in which the colours of 1 and 6 are the same; the number of removed ways is exactly the number of all the circular colourings of 1 to 5, as we may* ***view*** *1 and 6 as one unit, with one colour.*

The notion of "complement" appears here with the recognition that the number of legal circular colourings of 1..5 **complements** the number of legal circular colourings of 1..6 **to** the number of line colourings of 1..6. Thus, a "line case" may be viewed as being composed, number-wise, of two different "circular cases".

At this point, one should realize that in order to answer the case of "circular 1 to 5", one needs to know the solution of "circular 1 to 4"; which requires the solution of "circular 1 to 3"; and so on. This observation involves the heuristic of backward reasoning. The pencil-paper calculation should be built bottom-up, in reverse to the backward reasoning analysis.

All in all, this colourful challenge requires both of the heuristics of decomposition and backward reasoning, and an elegant creative invocation of the notion of "complement".

* * *

Our next example is different from the previous ones. It asks a question about the execution of a given algorithmic process. The question is about a particular state that will be reached after a given amount of time. Algorithmic problem solvers need not only design an algorithmic solution but also comprehend a given one. The question is an extension of an old challenge (about ants), which we assumed to be unfamiliar to the students. Since the question is somewhat different from others in our Stage-A exam, we posed it as a bonus, fifth question (in the exam of 2015/16).

Balls on a track. Eleven identical balls are spread on a 100 cm track. The two ends of the track – location 0 and location 100 are blocked with barriers. Each ball is put initially in a location indicated (in cm) by an integer below. The balls start moving at time 0, each in the direction indicated (near its location) below.

|| ← 6   14 →   ←24   ←38   44 →   ←50   ←54   64 →   ←74   82→   88→ ||

Each ball moves in a constant velocity of 1 cm per second. When a ball collides with a barrier in one of the ends, or when it collides with another ball, it switches its moving direction (and continues to move in the same velocity). For example, the balls that are initially in locations 44 and 50 will collide 3 seconds from the start, and will switch their moving directions. What will be the location of the ball that is initially fourth from the left (in location 38) after 5 minutes from the beginning?

The solution of this task requires abstraction. An examination of the movements of explicit balls yields a cumbersome, probably impossible pencil-paper computation. One may do much better with an alternative perspective, in which some details are disregarded.

Since the balls are identical, and their velocities are the same, one may regard each ball as anonymous and *overlook* the ball collisions. This viewpoint enables a view of a collision between two balls as an event that does not have any impact on the movements of the two identical balls in their original directions. This abstract point of view considerably simplifies the view of the task.

In addition, one may further extend this train of thought to disregard collisions with the right and the left ends. One may *pretend* that there are no left-end and right-end, and extend the track with a sequence of its copies in each direction. This allows one to view each ball as moving steadily in its original direction for 300 seconds. Once one obtains the final location of each ball in the sequence of copies, one may return to the original task, and transform this location into a concrete location in the given track. Since the balls remain in their original order, the answer would be the location that will be fourth from the left.

All in all, they key feature of the solution is abstraction, in which one does not view the task in its original arrangement, but rather as one with an arrangement yielded from an "as if" perspective (Ginat, 2010), of "anonymous" balls that collide neither with each other nor with the barriers.

3. Discussion

The examples presented in this paper display tasks with diverse characteristics that may be relevant for a preliminary stage of the OI activity. Elements that were embedded in the examples include: algorithmic design with a repeatedly used operator, summation schemes, game instances, invariance, the notion of complement, case analysis, logical reasoning, algorithmic tracing, and more. Such elements are apparent in algorithmic problem solving.

Yet, the features that we tried to underline are related to essential cognitive aspects involved in algorithmic problem solving; in particular those of abstraction, heuristics employment, creativity, and declarative conceptions. The relevance of these aspects was shown here with pencil and paper tasks.

We exemplified diverse appearances of these aspects with several illustrations. Abstraction was exemplified in the first and second tasks (Reversing and Sum of subsequences) with focusing on particular elements and ignoring others. It also appeared in the fifth task (Balls on a track) with the notion of "as if", which involved a change of perspectives. The employment of heuristics appeared in all the tasks. Particular heuristics that were relevant included problem decomposition, auxiliary construction, generalization from simple cases, and backward reasoning. Creativity appeared in two of the tasks – creative auxiliary construction in the Reversing task, and creative decomposition, using the notion of "complement" in the fourth, Watch colouring task. Declarative conceptions were relevant in all the tasks. It was particularly apparent in observing optimality in the Reversing task and in recognizing an invariant pattern in the third, Board game task.

Our experience with the presented tasks (posed in different years) show that the first two were solved by about a third of the students, the third – by about a fifth of the students, the fourth – by less than a tenth of the students and the fifth – by an even smaller number of students. We noticed that many students spent a long time on the simpler tasks, and often obtained only partial insight. They were then left with little time for the harder tasks.

Nevertheless, students were enthusiastic about the tasks. Many spent extra time after the exam to solve the tasks that they did not manage to solve during the exam. Some of the teachers were enthusiastic as well, although several of them felt uncomfortable, since they could not solve the tasks themselves. Interestingly, some of the students invited to the next stage did not attend it, and said that they attended the Stage-A exam "just for the challenge" of coping with colourful tasks.

Each year, after checking the answers we choose about one tenth of the students (250–300 students) for the next stage, according to the tasks they solved. About 15% of those invited to the next stage are girls. We pay particular attention to students that do really well in the Stage-A exam. Quite a few remain at the top in the next stages.

Some of the top students are also very competent in our country's IMO activity. We wonder about the correlation between success in the IOI and success in the IMO. Obviously, there is a correlation, but there are also differences.

Mathematical thinking and algorithmic thinking involve the recognition of hidden patterns, which is strongly tied to declarative conceptions and the notion of "knowing that" (Ryle, 1949). Algorithmic thinking also involves a strong facet of "knowing how". The observations sought by algorithmic problem solvers should yield suitable operational computations. Competent IOI problem solvers effectively combine the "that" and the "how", and usually turn to the "how" only after seeing the "that". For competent IMO problem solvers seeing the "that" may often be sufficient.

At the preliminary level of pencil-paper algorithmics we may not expect involved operational computations. However, it is still relevant to aim for an operational computation that will be carried out only after hidden patters are recognized. The cognitive aspects underlined here illuminated a facet of recognized "that" that paved the way to an operational "how". Suitable awareness of these aspects may assist task designers in their designed and posed tasks, including pencil-paper tasks.

Acknowledgement

We thank Hanit Galili and Nir Lavee, from the Israel IOI team, for statistical information about students' success in our Stage-A tasks.

References

Armoni, M., Gal-Ezer, J., Hazzan, O. (2006). Reductive thinking in computer science. *Computer Science Education*, 16(4), 281–301.

Burton, B. (2010). Encouraging algorithmic thinking without computer. *Olympiads in Informatics*, 4, 3–14.

Dagienė, V. (2006). Information technology contests – introduction to computer science in an attractive way. *Informatics in Education*, 5(1), 37–46.

Dagienė, V., Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In: Mittermeir, R.T. and Syslo, M.M. (Eds.) *Informatics Education – Supporting Educational Thinking, Lecture Notes in Computer Science*, 5090. Springer, 19–30.

Ginat, D. (2008). Learning from wrong and creative algorithm design. *Proc of the 39th ACM Computer Science Education Symposium – SIGCSE*. ACM Press, 26–30.

Ginat, D. (2010). The baffling CS notions of "as-if" and "don't care". *Proc of the 41st ACM Computer Science Education Symposium – SIGCSE*. ACM Press, 385–389.

Ginat, D., Blau, Y. (2017). Multiple levels of abstraction in algorithmic problem solving. *Proc of the 48th ACM Computer Science Education Symposium – SIGCSE*. ACM Press, 237–242.

Kubica, M., Radoszewski, J. (2010). Algorithms without programming. *Olympiads in Informatics*, 4, 52–66.

Opmanis, M. (2009). Math contests: solutions without solving. *Olympiads in Informatics*, 9, 147–161.

Polya, G. (1954). *How to Solve it*. Princeton University Press.

Radoszewski, J. (2014). More algorithms without programming. *Olympiads in Informatics*, 8, 157–168.

Ryle, G. (1949). *The Concept of Mind*. The University of Chicago Press.

Schoenfeld, A.H. (1992). Learning to think mathematically: problem solving, metacognition, and sense making in mathematics. In: Grouws D.A. (Ed.), *Handbook of Research on Mathematics Teaching and Learning.* 334–370.

van der Vegt, W. (2012). Theoretical tasks on algorithms; two small examples. *Olympiads in Informatics*, 6, 212–217.

Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35.

D. Ginat – heads the Israel IOI project since 1997. He is the head of the Computer Science Group in the Science Education Department at Tel-Aviv University. His PhD is in the Computer Science domains of distributed algorithms and amortized analysis. His current research is in Computer Science and Mathematics Education, with particular focus on various aspects of problem solving.