THE CONCEPT OF "ALGORITHM EFFICIENCY" IN THE HIGH SCHOOL CS CURRICULUM

Judith Gal-Ezer[1] and Ela Zur[2]

Abstract *¾ Algorithms and the efficiency of algorithms are basic concepts to every computer science (CS) curriculum. These concepts are at the core of a new CS curriculum being implemented in Israeli high schools. Initial observations of the implementation of the program and examination of students' achievements confirmed our assumption that efficiency is a difficult concept to conceive. The research questions were: To what extent do 10th and 11th grade students succeed in internalizing efficiency and to what extent are they able to analyze an algorithmic problem and design an efficient algorithm? Is there a difference in the understanding of the material between students who take mathematics on different levels? The conclusions were that most 10th grade students are unable to internalize the concept of algorithm efficiency whereas in the 11th grade, more students are able to internalize the concept. A significant difference is observed between students taking 5-point mathematics and those taking 4-point mathematics.*

Index Terms ¾ algorithms, curriculum development, pre-college programs

BACKGROUND

The Israeli education system is basically a centralized one. The Ministry of Education sets educational policy on all levels and implements it with the help of specialized committees, professional curriculum developers, and inspectors. Students go through ten years of mandatory education, usually divided into elementary school (grades 1 through 6), junior high school (grades 7 through 9), and one year of high school (grade 10). Two additional years (grades 11 and 12) are optional. Subjects are taught in "study units", each unit representing approximately 90 hours of study. Many subjects can be studied at various levels, the most common being 3- and 5-unit programs that differ significantly in the quantity and conceptual depth of the material. A 5-unit program would typically require studying the subject for 5 weekly hours throughout the three years of high school.

These three years of high school culminate in an extensive set of matriculation examinations, which are crucial for admission into Israeli universities (except for the Open University of Israel). Students take examinations in a score of required subjects and several electives including math, Hebrew language and literature, history, English, physics, chemistry, computer science (CS) and others. To gain a matriculation certificate, a student must successfully pass examinations in at least six subjects, accumulating at least 20 study units, though most students earn more. Programs of study are updated from time to time. Sometimes all-new curricula are developed, as in the case of computer science, where a new curriculum was developed and is now being implemented in Israeli high schools. The new CS high school program includes all the basic elements of traditional CS programs. One of the most important and basic concepts is the efficiency of algorithms.

Algorithms and the efficiency of algorithms appear in most of computer science literature. As a consequence, algorithms and the efficiency of algorithms are incorporated into every university computer science curriculum as well as into high school programs. An examination of university curricula, from ACM's Curriculum 68 [1], through its "offspring" Curriculum 91 [15] and CC-2001 [6], and of high school curricula [9], shows that indeed, the concepts of algorithms and algorithm efficiency are central to the study of CS.

At the core of the new CS curriculum are algorithms, algorithmic thinking and the efficiency of algorithms. The following are the major principles that guided the professional committee which designed the curriculum:

- Computer science is a full-fledged scientific subject that should be taught in high school on a par with other sciences, such as physics, chemistry and biology.
- The curriculum should focus on scientific principles and basic concepts. The central concept is the algorithmic problem and the algorithms that solve it.
- The program of study should be composed of required units and electives.
- Conceptual and experimental issues should be interwoven throughout the program in a zipper-like fashion.
- Two quite different programming paradigms should be taught that suggest alternative ways of algorithmic thinking. This emphasizes the fact that algorithmics is the central subject of study.
- New course material must be written for all parts of the program.
- Teachers certified to teach the subject should have formal CS education as well as teacher training.

In its first two units, the program emphasizes the basic concepts and principles underlying CS [3]. These materials

[1] Judith Gal-Ezer, Open University of Israel, 16 Klausner Street, Tel Aviv, Israel 61392, galezer@openu.ac.il
[2] Ela Zur, Open University of Israel, 16 Klausner Street, Tel Aviv, Israel 61392, ela@openu.ac.il

0-7803-7444-4/02/$17.00 © 2002 IEEE **November 6 - 9, 2002, Boston, MA**
32nd ASEE/IEEE Frontiers in Education Conference

integrate both theoretical and practical aspects. In keeping with the zipper principle, the pedagogical approach taken is the following: Each topic is introduced first on a conceptual level, including manual exercising; e.g., the development of algorithms and their analysis. The topic is then recast in practical form; e.g., the implementation of algorithms using a programming language. The first unit of the program, *Fundamentals 1*, includes a chapter on efficiency. The second unit, *Fundamentals 2* and the third, *Software Design*, expand on the concept of efficiency introduced in the first unit. Each unit consists of 90 hours.

The efficiency of an algorithm is determined by the complexity measures needed to perform it, the most important of which are memory space and time. In many cases, more than one algorithmic method can solve a specific problem. By analyzing the efficiency of algorithms, different solutions to the same algorithmic problem can be compared and the most efficient algorithm identified. If a problem is not analyzed, and in the absence of planning, the algorithm could be too expensive, and thus useless. That is, it would be impossible to run the algorithm in practice within a reasonable amount of time (the myth concerning the computer's speed notwithstanding). Teaching algorithm efficiency at a relatively early stage makes it possible to expand the range of algorithmic methods, plan algorithms, evaluate alternative methods and perform analyses [5].

Spohrer and Soloway have found that experts in programming spend more time on designing the algorithm while novice programmers go directly to writing the program [11]. The difference between the two is expressed, among other aspects, in the program's efficiency. Incorporating the topic of efficiency early in the teaching process may help to narrow the gap between the two.

Following this, *Fundamentals 1* indeed introduces the notion of efficiency at a relatively early stage. However, our initial observations of the implementation of the new CS curriculum confirmed our assumption that efficiency is a difficult concept to conceive, in particular in high school. This motivated our research.

The study presented here is part of a wider study of the implementation of the new CS curriculum and also investigated high school students' misconceptions regarding the efficiency of algorithms [4]. Studies have shown that students' conceptions of scientific issues are often not in line with accepted scientific thinking; that is, they have misconceptions regarding various notions. In their study of mathematics and science education, Stavy and Tirosh [12, 13, 14] examined common thinking patterns underlying misconceptions in different content areas. We assumed that this would apply to algorithm efficiency as well. And, in fact, we found that students mistakenly relate the efficiency of an algorithm to its length; they think that if the algorithm uses fewer variables, it is more efficient; and they incorrectly assume that if two programs accomplish the same task they are equally efficient. Relating to these misconceptions is crucial since the new CS curriculum rests on interweaving conceptual and practical aspects, and students' misconceptions would comprise a serious stumbling block to understanding the concepts and therefore to their ability to write an efficient program in practice.

THE STUDY

The study was carried out during the 1998-1999 school year, among 10th and 11th graders in five high schools in which the new CS curriculum was being implemented.

The first two units of the new curriculum, *Fundamentals of Computing*, are taught in most high schools in the 10th grade (*Fundamentals 1*, chapters 1-11) and in the 11th grade (*Fundamentals 2*, chapters 1-4). In most cases, CS is mandatory in the 10th grade and an elective in the 11th grade. As a result, since there are students who choose not to continue their CS studies in the 11th grade, 10th grade classes are more heterogeneous in composition and 11th grade classes are more homogeneous.

The research questions were:

1. To what extent do 10th and 11th grade students succeed in internalizing efficiency?
2. To what extent are 10th and 11th grade students able to analyze the algorithmic problem and to design an efficient algorithm?
3. Is there a difference in students' comprehension of the material when they take mathematics at the 4-point level and when they take mathematics at the 5-point level?

The participants were:

- 174 students in the 10th grade after they had studied seven chapters of the first unit - *Fundamentals 1*, including iterative control structures, but before studying the chapter on efficiency.
- 141 of the same students at the end of the 10th grade, after completing the first unit.
- 145 students in the 11th grade, after completing *Fundamentals 1* and *2*.

Research Instruments

The research instruments were three versions of a questionnaire which dealt with various aspects of the concept of efficiency, such as the relation between the speed of computers and efficiency, the relation between actual measured time and efficiency, the major aspect of a program which effects efficiency; a comparison of two programs designed to accomplish the same task which differ in length, number of variables, or are different because a statement was removed from the loop. The questions were multiple-choice and each left room to explain the choice of response. In addition, students were assigned basic algorithmic problems and asked to write computer programs to solve them.

In the 10th grade, the multiple-choice questionnaire was designed to determine students' intuitive conception of

algorithmic efficiency before studying chapter 8 in *Fundamentals 1* - the chapter on efficiency (pre), and the extent to which they internalized the concept after studying it (post). This was compared with 11th graders' perception of algorithm efficiency after they had completed both units of the materials.

Item analysis was used to enable us to differentiate among students' responses. In addition, we used McNemar's test to examine the significance of changes between the first administration of the questionnaire and the second for 10th grade students. This test is particularly applicable to pre/post designs [7, 8].

Results

The first research question dealt with the extent to which 10th and 11th graders are able to internalize the concept of program efficiency. To answer this question, we compared the results of multiple-choice questions from the pre- and post-questionnaires in the 10th grade and the questionnaires administered in the 11th grade. Figure 1 shows a sample of the results graphically.

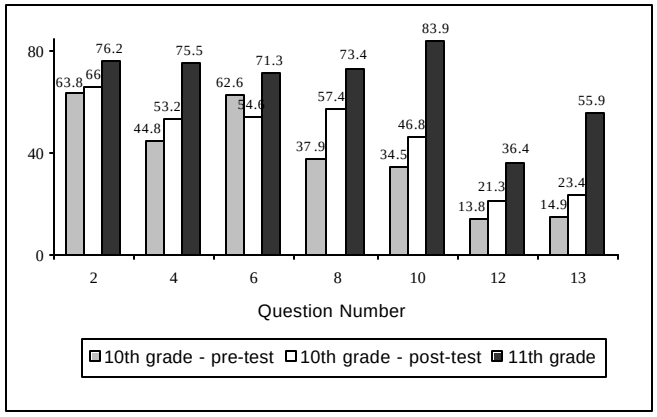

FIGURE 1
SAMPLE OF CORRECT RESPONSES ON MULTIPLE-CHOICE QUESTIONS IN THE 10TH GRADE (PRE AND POST) AND THE 11TH GRADE.

All of the questions discriminated well. An examination of Figure 1 indicates that for most of the questions, there were only small differences between the pre and post administrations of the questionnaires in the 10th grade, and larger differences between the 10th and the 11th grade.

Table I shows the extent to which the 10th graders learned about efficiency between the first administration of the questionnaire and the second. We labeled a correct answer as + and an incorrect answer as –. For instance, + / + means a correct answer in the pre test and a correct answer in the post test. We are especially interested in the column showing how many students answered a question incorrectly before studying the chapter on efficiency, and then answered the same question correctly after studying the material (column – / + in bold).

TABLE I
DISTRIBUTION OF 10TH GRADE STUDENTS' RESPONSES ON PRE AND POST QUESTIONNAIRES (IN PERCENTAGES)

Number of question and issue	+ / +	+ / –	– / –	– / +	McNemar test (df=1)
Q1: relation between speed and efficiency	36.07	17.21	19.67	**27.05**	2.667
Q2: relation between actual measured time and efficiency					3.457
Q3: relation between speed and efficiency					1.000
Q4: major aspect of program that effects efficiency					4.412
Q5: relation between length of input and efficiency					10.526 *
Q6: relation between length of input and efficiency					0.220
Q7: major aspect of program that effects efficiency					17.163
Q8: two programs of same length; statement in loop in prog. 2, not in loop in prog. 1					8.696 *
Q9: program 1 has more variables than program 2					5.828 *
Q10: program 1 shorter than program 2					5.765

* $p<0.05$

From table I we see that the percentage of students who actually "learned" (i.e., answered incorrectly before learning the material and correctly after studying efficiency) was relatively small and averaged around 20% for most of the questions. In addition, only for three of the questions was this finding significant ($p<0.05$) according to McNemar's test.

Analysis of Algorithms and Writing Efficient Programs

The second research question dealt with the extent to which 10th and 11th grade students were able to analyze an algorithmic problem, and to design an efficient program. To answer this question, students in the 10th and 11th grades were given algorithmic problems on different levels of complexity. Tenth graders were given the problems twice: before and after learning about efficiency.

The following task was presented to 10th grade students: Write an **efficient** program whose input is a positive integer larger than 1 and whose output are the divisors of the input. The divisors of an integer are all the positive numbers into which the integer can be divided without a remainder. The students who answered the

question correctly solved the problem in one of the following ways:

(1) Execute the loop to the integer:

```
read (num);
for i: = 1 to num do
if (num mod i) = 0 then
    write (i);
```

(2) Execute the loop to half of the integer. Identical to (1) except for:

```
for i: = 1 to (num div 2) do
```

(3) Execute the loop to the square root of the integer. Identical to (1) except for:

```
for i: = 1 to (trunc(sqrt(num))) do
```

Table II shows how 10th graders solved the problem before (pre) and after (post) studying efficiency. From the table we see that a large percentage of 10th grade students were unable to solve the problem. About 30% of the students wrote a correct program (pre and post). The results for the pre-test show that only slightly over 1% of the students wrote an efficient program, and after studying the topic (post), more than 6% wrote an efficient program. Thus, of the students who were able to write a correct program, only about 25% seem to have internalized the notion of efficiency, similar to the findings for the multiple-choice questionnaire shown in table I.

TABLE II
PROGRAMS WRITTEN BY 10TH GRADERS BEFORE AND AFTER LEARNING ABOUT EFFICIENCY (IN PERCENTAGES)

Solution	Pre-test N = 174	Post-test N = 141
No answer	37.93	41.13
Incorrect program	32.75	28.36
Execute the loop to num (while or for)	28.16	24.11
Execute the loop to **(num div 2) ***	**1.14**	**4.25**
Execute the loop to **(trunc(sqrt(num))) ***	**0**	**2.12**

* Efficient program

In the 11th grade, students were presented with the following problem: Write an **efficient** program whose input is a positive integer and whose output is a statement that the integer is a prime number or is not a prime number. The 11th grade students who answered the question correctly solved the problem in one of the following ways:

(1) Execute the loop to the integer:

```
read (num);
flag: = true;
for i: = 2 to num-1 do
if (num mod i) = 0 then
    flag: = false;
if flag then
    write ('the number', num, 'is prime')
else
    write ('the number', num, 'is not prime');
```

(2) Execute the loop to half the integer. Identical to (1) except for:

```
for i: = 2 to (num div 2) do
```

(3) Execute the loop to the square root of the integer. Identical to (1) except for:

```
for i: = 2 to (trunc(sqrt(num))) do
```

(4) Execute the loop up to half the integer, stopping if the integer is divisible:

```
read (num);
flag: = true;
i: = 2;
while (i <= n div 2) and flag do
begin
    if (num mod i) = 0 then
        flag: = false
    else
        i: = i+1;
end;
if flag then
    write ('the number', num, 'is prime')
else
    write ('the number', num, 'is not prime');
```

(5) Execute the loop to the square root of the integer, stopping if the integer is divisible. Identical to solution (4), except for:

```
while (i <= (trunc(sqrt(num)))) and flag do
```

Solutions 4 and 5 are the most efficient, but even solutions 2 and 3, which only shorten the loop, provide evidence that the student performed an initial level of analysis of the problem and planned the algorithm, which indicates some degree of internalization of the concept of efficiency. Table III shows the results for this question.

TABLE III
PROGRAMS WRITTEN BY 11TH GRADE STUDENTS (N=143, IN PERCENTAGES)

Solution	%
No answer	19.6
Incorrect program	22.4
Execute the loop to num (while or for)	15.4
Execute the loop **to (n div 2)***	**7.0**
Execute the loop **to (trunc(sqrt(num)))***	**8.4**
Execute a while loop with two conditions: **to (n div 2) + Boolean variable (not prime)****	**13.2**
Execute a while loop with two conditions: **to (trunc(sqrt(num))) + Boolean variable (not prime)****	**14.0**

* shortened loop
** most efficient program

Table III shows that in the 11th grade, a large percentage of students (42%) are able to write an efficient program for a given algorithmic problem. From tables II and III, it is evident that the percentage of students who were able to solve the problem correctly, perform a mathematical analysis and write an efficient program for a given algorithmic problem was significantly higher in the 11th grade than it was in the 10th grade.

Comparison Between Students Studying 4- and 5-point Mathematics in Understanding Computer Fundamentals

From our experience, we assumed intuitively that high school students with well-based mathematics knowledge; in other words, those who were taking a 5-unit program in mathematics (commonly referred to as 5-point math) together with their CS studies, would more successfully absorb new concepts in CS. Students choose their mathematics level (3-, 4- or 5-points) at the end of the 10th grade, and it is mainly 4- and 5- point math students who continue their CS studies in the 11th grade. We decided to investigate whether there was a difference in the knowledge of CS fundamentals between 4 and 5-point students of mathematics in the 11th grade. To do this, the students were given a score based on their responses to the multiple-choice questions, their explanations for these responses, and the computer programs they wrote as solutions to the problems they were given. Table IV compares the average scores for students taking 4- and 5-point mathematics using a t-test.

TABLE IV
COMPARISON OF GRADES BY LEVEL OF MATHEMATICS

	4-point Math	5-point Math	t	p value
N	48	87		
Mean	51	58	-3.0601	0.0027
SD	12.89	13.11		

Table IV shows a wide distribution of grades for both groups, however the comparison indicates that the difference between the two groups was significant ($p = 0.0027$). We can therefore conclude that there is a correlation between the level of mathematics studied and success in learning fundamentals of computer science.

CONCLUSIONS AND RECOMMENDATIONS

Having examined the implementation of the CS curriculum in the schools, we can conclude that in heterogeneous 10th grade classes, most students are unable to internalize the concept of algorithm efficiency (see table I). The percentage of students who internalized the concept in theory, as tested in multiple choice questionnaires, was relatively small; for most questions, about 20%. Moreover, these students' ability to perform a mathematical analysis of an algorithmic problem in a way that would lead them to write an efficient solution was practically non-existent. Only 6% were able to do this after studying the chapter on efficiency (see table II). We can therefore cautiously conclude that most 10th grade students are unable to internalize the concept of algorithm efficiency even in theory, and their intuitive perception of the concept is retained despite the learning process.

This may be related to the misconceptions that students hold regarding the concept of algorithm efficiency. One possible explanation of the source of the misconception 'the more variables, the more execution time', is that students had incorrect conceptions regarding computers and how they operate. This may be because the students did not learn about computer organization before studying *Fundamentals 1* and *2*. We therefore suggest developing a short unit on computer organization to be taught before the two *Fundamentals* units, in the hope that this will contribute to lessening students' misconceptions regarding the notion of efficiency.

In the 11th grade, where only students who opted for CS continue their studies, the situation is somewhat better. A larger percentage of students was able to internalize the concept of algorithm efficiency (see figure 1). When asked to write an efficient program to solve an algorithmic problem, close to 42% were able to do so (see table III).

Since we still believe that it is important to incorporate efficiency into the early stages of every CS curriculum, we recommend introducing the concept of algorithm efficiency in the 10th grade, but to actually teach it only in the 11th grade. This means a radical change in the study materials, which will also include the integration of examples that emphasize the process of developing algorithms stage by stage until arriving at an efficient algorithm, rather than providing only the efficient solution. There should also be many more examples of algorithms that solve the same problem which differ only in their efficiency.

We have seen that when solving complex algorithmic problems (which generally demand mathematical analysis), students tend to write inefficient programs because they have not analyzed the problem sufficiently to produce an efficient program. Teachers should therefore invest more effort in teaching mathematical analysis before allowing students to attempt to write a program.

One important point to which we have not related is differences among teachers. We found that there were significant differences in the performance of students of different teachers. There is an acute shortage of good teachers, among other reasons, because of the high salaries offered in industry [10, 2]. This phenomenon needs to be examined carefully in an effort to determine (cautiously) what can be done.

We assumed that students with a strong background in mathematics would more successfully internalize CS concepts than those with a weaker background in mathematics. Our findings indicate that this is, indeed, the case. Thus, since the concept of algorithm efficiency seems to be so strongly related to mathematical ability, we recommend teaching the concept only in the 11th grade

when classes are more homogeneous and composed of 4- and 5-point students of mathematics. In addition, as this study has shown, there is a significant difference between 4- and 5-point students of mathematics in terms of their success in learning the fundamentals of CS. This should be taken into consideration when advising students who are planning to take CS in the 11th grade.

Let us sum up our recommendations: We suggest to develop a short unit on computer organization to be taught before the two *Fundamentals* units; moreover we recommend a radical change in the study materials so as to include the integration of examples that emphasize the process of developing algorithms stage-by-stage until arriving at an efficient algorithm, rather than providing only the efficient solution. In addition, there should be many more examples of algorithms that solve the same problem which differ only in their efficiency. To prepare students more effectively, teachers should invest more effort in teaching mathematical analysis before allowing them to attempt to write a program.

Finally, in this study we focused on the perception of the notion of efficiency of algorithms. We strongly recommend to continue to investigate how high school students perceive and internalize other issues in computer science as well as other aspects of the implementation of this relatively new program.

ACKNOWLEDGMENT

We would like to thank Yael Alberton of the Evaluation and Training department of the Open University of Israel for her contribution in performing the statistical analysis, and Gila Haimovic for joining us in fruitful discussions and help in editing this paper.

REFERENCES

[1] ACM Curriculum Committee on Computer Science, "Curriculum 68: Recommendations for academic programs in computer science", *Communications of the ACM*, 11, 3, 1968, 151-197.

[2] Gal-Ezer, J., "Computer science teachers' certification program", *Computers and Education*, 25, 3, 1995, 163-168.

[3] Gal-Ezer, J., Beeri, C., Harel, D., & Yehudai, A., "A high school program in computer science", *Computer*, 28, 10, 1995, 73-80.

[4] Gal-Ezer, J., & Zur, E., "The Efficiency of Algorithms – Misconceptions", submitted.

[5] Ginat, D., "Efficiency of algorithms for programming beginners", *Proc of the 27th ACM Computer Science Education Symposium*, ACM Press, 1996, 256-260.

[6] IEEE Computer Society/ACM Task Force, "Year 2001 model curricula for computing (CC-2001)", http://www.computer.org/education/cc2001/report/, 2001.

[7] Linn, R. L., *Educational measurement*, 3rd ed., New York: Macmillan,1989.

[8] McNemar, Q., *Psychological statistics*, 2nd ed., New York: Wiley, 1955.

[9] Merritt, S. M. et al., "ACM model high school computer science curriculum", *Communications of the ACM*, 36, 5, 1993, 87-90.

[10] Poirot, J., et al., "Proposed curriculum for programs leading to teacher certification in computer science", *CACM*, 28, 3, 1985, 275-279.

[11] Spohrer, J. C., & Soloway, E., "Novice mistakes: Are the folk wisdoms correct?", *Communications of the ACM*, 29, 7, 1986, 624-632.

[12] Stavy, R., & Tirosh, D., "Intuitive rules in science and mathematics: The case of 'more of A - more of B'", *International Journal of Science Education*, 18, 6, 1996, 653-667.

[13] Stavy, R., & Tirosh, D., "Intuitive rules in science and mathematics: The case of 'everything can be divided by two'", *International Journal of Science Education*, 18, 6, 1996, 669-683.

[14] Stavy, R., & Tirosh, D., *How students (mis-)understand science and mathematics: Intuitive rules*, New York: Teachers College Press, 2000.

[15] Tucker, A. et al., "Computing Curricula 1991, A summary of the ACM/IEEE-CS joint curriculum task force report", *Communications of the ACM*, 34, 6, 1991, 69-84