The International Innovations Journal of Applied Science (IIJAS) Vol. 2, No.1, 15-03-2025
المجلد الثاني العدد الاول 15-3-2025 (IIJAS)مجلة ابتكارات الدولية للعلوم التطبيقية
https://doi.org/10.61856/xa3tkc02

SCIENTIFIC EVENTS GATE

The International Innovations Journal of Applied Science

Journal homepage: https://iijas.eventsgate.org/iijas

ISSN: 3009-1853 (Online)

The contribution of mathematical reasoning to the teaching-learning of recurrent algorithms and recursion

Soltani Walid[1*], Chellougui Faïza[2]
[1]ISEFC-Virtual University of Tunisia
[2] Faculty of sciences of Bizerte – University of Carthage -Tunisia

ARTICLE INFO

Article history:
Received 1 Jan 2025
Revised 30 Jan 2025,
Accepted 3 Feb. 2025,
Available online 15 Mar. 2025

Keywords:
mathematical reasoning
mathematical induction
recurrent algorithm
algorithmical approach
recursion

ABSTRACT

This article examines the teaching and learning of the concepts of recursive algorithms and recursion in the 4th year secondary school, *Computer Science* section class. The main objective is to identify the difficulties and obstacles students face when these concepts introduced in the classroom. The results of this study highlight the intrinsic complexity of these concepts, which pose a major challenge for students. Furthermore, the fundamental link between algorithm and mathematical problem-solving, although essential, is not explicitly established in the new computer science curricula. This limits students' overall understanding and ability to process these concepts, along with the diversity of formulations of recurring algorithms further complicate their learning, contributing to an often-limited understanding. The algorithm, at the intersection of mathematics and computer science, is a central point of interaction between these disciplines. Its construction and processing are based on logical and mathematical frameworks, which give rise to the notions of recursion, mathematical induction and recurrent algorithm all of which are of particular importance for both fields. However, this study highlights persistent difficulties among students, particularly in mastering recursion and recurrent algorithms. These findings underscore the importance of a thorough didactic approach to overcome these barriers and improve consistency between math and computer skills in secondary education.

1. Introduction

New educational trends have been introduced in the teaching of computer science in Tunisian secondary schools since 2021. These programs emphasize the development of learners' mathematical reasoning and problem-solving skills, with a particular focus on the area of «Computational Thinking and Programming» which stresses the importance of algorithms (the algorithmic convention, 2022). Recursion and the algorithmic aspects hold a considerable place in the teaching of computer science in Tunisia (Soltani, 2022). However, persistent difficulties have been identified among students, particularly in studying the processing of algorithms, including complex notions such as recurrent algorithms and recursion, Taught in the 4th year of secondary school in the computer science section (Polycarpou, 2006). This study aims to propose remediation methods to help students improve their mastery of these notions.

Methodologically, this work is structured into three complementary sections:

The first section is didactic and pedagogical in nature, focusing on a review of the literature

* Corresponding author.
E-mail address: wsoltani2016@gmail.com

https://doi.org/10.61856/xa3tkc02

on teaching of recursive algorithms and recursion.

The second section of a curricular nature focuses on *computer science* curriculum. It examines the proposed algorithmic approaches, as well as the connection between reasoning, formalization and logic in the treatment of mathematical induction and recursion.

The third part, devoted to an experimental study, was conducted with 64 students in the 4th year of secondary school[1], *Computer Science* section. This study evaluates their skills in recursion and algorithms applied to computer science, as well as their mastery of mathematical induction in mathematics.

This research aims to make a significant contribution to improving the teaching of recursion and recurrent algorithms, while reinforcing the interconnection between mathematical and computer science disciplines.

2. Didactic and pedagogical approach to the notions of recurrent algorithm, algorithmic and recursion

This explores the concept of recursion and recurrent algorithms which will the developed and illustrated with practical examples to demonstrate their importance and practical application.

2.1. Recursion

León and Modeste (2020) highlighted the complexity of defining recursion, a difficulty that is evident in the responses of researchers in mathematics and computer science. These authors specify that a definition is qualified as recursive when the object to be defined is used in its own definition. For example, a common definition in computer science might be: « A list is either an empty list, or a pair composed of a first element and a list». Here, the word «list» reappears in its own definition (León and Modeste, 2020).

León and Modeste also point out that there are essential conditions to be met for a recursive definition not to lead to an infinite sterile regression. In computer science, an object is called recursive if it is defined in terms of itself or refers to a reduced version of itself to accomplish a given task.

Recursion is applied in a multitude of contexts, with meanings that may vary slightly depending on the domain of application, whether it is definitions, algorithms, data types, strings, numerical suites, and more (León, Modeste and Durand-Guerrier, 2020).

This concept is particularly valued in mathematics and computer science for its ability to solve problems efficiently. However, it is crucial to include a breakpoint or termination condition in any recurrent definition or algorithm to ensure that the process does not continue indefinitely. Without this precaution, the process could enter an infinite loop, making it impossible to obtain a result (León, Modeste, and Durand-Guerrier, 2020).

2.2. The recurrent algorithm and algorithmics

In computer science, the term ‘algorithm’ can have several meanings. It is therefore essential to clarify this concept and to specify the definition that will be used in our work. Modeste (2012) defines an algorithm as: «*a problem-solving procedure, [disregarding any specific characteristics the problem may have] applying to a family of problem instances and producing, in a finite number of constructive, effective, unambiguous and organised steps, the answer to the problem for any instance in that family*.». (Modeste, 2012, p. 25).

Laval (2018) cited an example of an algorithm: the sorting algorithm. This does not solve the problem of sorting a particular dataset but aims to sort any dataset. The sorting problem applies to different instances, i.e. different datasets.

Among the various types of algorithms in computer science, we will concentrate on recurrent algorithms, which are at the heart of our didactic study. An algorithm is said to be

[1] Year 4 is the final year of secondary education, at the end of which students, aged 19, sit for the baccalaureate exam.

https://doi.org/10.61856/xa3tkc02

recurrent when it uses an iterative or recursive process to generate a result that may depend on several previous results. For example, a recurrent algorithm of order *p* is said to exist when the current result depends on *p* previous results (Léon and Modeste, 2020). On the other hand, an algorithm or program is said to be recurrent when it calls itself within its own body (León and Modeste, 2020).

A classic example of a recurrent algorithm is the calculation of the Fibonacci sequence, defined by for any natural integer $n \geq 2$,
$F_n = F_{n-1} + F_{n-2}$.

This sequence illustrates a recurrent algorithm of order 2, where each term is calculated as a function of the two preceding terms.

When talking about the concept of algorithm, it is also relevant to mention a closely related concept: algorithmics. Laval (2018) defines algorithmics as: «*The set of rules and techniques involved in defining and designing systematic processes for solving a mathematical problem, making it possible to describe precisely the steps needed to solve this problem using an algorithmic approach. Algorithms are therefore the science of algorithms. It is concerned with the art of constructing algorithms as well as characterising their validities, robustness, reusability, complexity and efficiency*. » (Laval, 2018, p. 42).

According to this definition, algorithmics consists of analysing a mathematical problem using rules and techniques and then developing an algorithm to solve it.

In conclusion, recursion and algorithmics play a central role in problem solving in computer science and mathematics, offering powerful tools for tackling complex tasks. A thorough understanding of these concepts, although demanding, is essential for developing efficient and innovative algorithms. Teaching them requires a clear and structured approach to overcome the challenges associated with their abstraction and ensure that learners have a solid grasp of them.

2.3. Recursion as a pedagogical tool

In computing, recursion, often implemented through recurrent algorithms, can also play a key role in education as a tool to confirm or disprove incomplete induction assumptions. Given the difficulty of mentally experiencing totalizing induction, which deals with infinity, recursion offers a concrete and iterative method for exploring and verifying these concepts in a finite framework.

2.4. Relevance of mathematical induction in teaching

Nowadays, mathematical induction has become essential in education because of its relevance and practical usefulness. The theory of recursion in computer science is nothing other than the applied study of this mathematical reasoning, enabling recurrent algorithms to efficiently solve complex problems.

In short, the integration of the concepts of mathematical induction and recursion, supported by concrete examples and interdisciplinary applications, enriches the teaching of these notions and facilitates student understanding, while highlighting the complementary nature of mathematics and computer science in problem solving (Soltani, 2022; Soltani and Chellougui, 2024).

The concepts of mathematical induction and recursion occupy an essential place in both mathematics and computer science, revealing a deeply intertwined and inseparable relationship (Leon and Modeste, 2020). This interconnection is not just theoretical; it could also offer promising avenues for overcoming the challenges of teaching and learning these complex notions. Indeed, several studies (Leron and Zazkis, 1986; Polycarpou, 2006) suggest that a deeper understanding of this relationship could be the key to addressing and resolving the difficulties that these concepts raise in learners, making their teaching more accessible and their application more effective.

3. Reasoning, Recurrent Algorithm and Recursion in Computer Science Curricula

In this section, the focus will be on curriculum analysis, such as the four

https://doi.org/10.61856/xa3tkc02

institutional documents[2]. Our analysis focuses mainly on the 4th year secondary class in the *computer science* section, to determine the place given to the notions of mathematical induction, recurrent algorithm and recursion and to examine how they are integrated into teaching at this level. More specifically, the institutional conditions for their introduction and implementation.

Algorithmic convention. According to the recommendations in this document, the importance given to reasoning, computational thinking and algorithms is noteworthy. These concepts are presented as essential foundations

for learning computer science. The document states: « *With the aim of developing learners' reasoning and problem-solving skills, the Computational thinking and programming area focuses on algorithms. Algorithms must be written in accordance with the conventions set out in this document.* » (Algorithmic conventions- September 2022, p.2).

The document also details the syntaxes of different algorithmic structures, thus providing precise guidelines on how the concepts should be formally represented and taught:

-*Simple elementary operations*: Reading, writing etc.
-*Simple data types*: real, integer, character etc.
-*Data structures*: table, file etc.
-*Declarations*: simple data type objects, tables, files etc.
-*Conditional control structure*: simple, complete, generalised and multiple choice.
- *Iterative control structure*: complete and stop condition.
- *Modules*: functions and procedures.
- *Arithmetic and logical operators*.
-*Predefined functions*: functions on numeric types (rounding, square root, etc.), character type functions (ord(c) and chr(d)) and functions on the character string type.
-*Predefined functions and procedures on files*.

Python implementation of algorithmic conventions. The authors of this document have translated the algorithmic solutions defined in the first document using the Python programming language. This approach aims to provide teachers and students with practical examples of the implementation of algorithmic conventions using a widely used and accessible language, thus facilitating the practical learning of algorithmic concepts.

Specific computer teaching aids. The designers of these other institutional documents have opted for an integrated approach by rehabilitating Computational Thinking and Programming as central learning areas. These documents impregnate the whole of secondary education with this associated knowledge, by proposing teaching methods and guidelines adapted to each level of study.

A strategic choice in the development of computer science curricula in Tunisian secondary schools is to promote interaction between this subject and other areas of learning. This approach promotes an interdisciplinary pedagogy: «*Establish links and threads between the different learning areas, breaking with the linear aspect of the curriculum*». (SCTA[3], 2022).

The table sets out the associated knowledge for the 4th year of secondary school in the *computer science* section, providing an overview of the skills and knowledge to be developed at each stage of learning:

Table 1: Associated computer science knowledge in secondary education, 4th year secondary school section computer sciences. SCTA, 2022, pp.10-13

Year of study	Associated knowledge

[2]There is as yet no official curriculum or standardised textbook for computer science. Only four institutional documents are currently available: the *algorithmic convention*, the *Python implementation of algorithmic conventions*, and *Specific computer teaching aids* (2022-2023) published by the Tunisian Ministry of Education.

[3] Specific computer teaching aids

https://doi.org/10.61856/xa3tkc02

4th *Computer Sciences*	-Use advanced algorithmic concepts to solve problems involving: *Data structures; Sorting methods; Recursion.* *Recurrent processing and algorithms.* *Optimization and approximation* - Use a programming environment to implement a solution

The computer science programs at high school place particular emphasis on the importance of rigorous curriculum writing and the articulation between reasoning, formalization, and logic.

The recent integration of mathematical content (such as real suites, arithmetic, optimization, approximation, etc.) into computer science curricula for the 4th grade class (*Computer Science* section) has several key objectives (SCTA): « - *Mainly deal with arithmetic calculations (PGCD, PPCM, prime numbers, etc.), optimisation problems, approximate values, etc.*
- *Demonstrate the transition from iterative to recursive formulations.*
- *Deal only with the case of simple recursion (neither crossed nor indirect) on naturally recursive problems (factorial, palindrome, PGCD, etc.).*
- *Various problems will be dealt with, focusing on mathematical induction relations of order one and higher (sequences, Pascal's triangle, golden ratio, etc.) ».* (SCTA, pp.10-11).

Recursion, mathematical induction and recurrent algorithms are thus highlighted in the associated knowledge of the 4th year secondary (section: *Sciences computer*), as key objectives of teaching.

4. Experimental Investigation

Our experiment was carried out with the 4th year secondary school class (*Computer Science* section), using a test to explore their relationship with the objects of mathematical induction in mathematics and recursion in computer science. The test, which consisted of two exercises with the two objects articulated, was offered to the students for their own performance. The aim of this experiment is to analyse the steps used by students[4] to conduct a mathematical proof requires mathematical induction and an algorithm in computer science after a mathematical analysis.

The analysis of this test will provide a lot of information on the knowledge acquired by students. On the one hand, on the understanding and mastery of mathematical induction in mathematics and on the other hand on the processing of a recurrent algorithm in computer science. This experiment raises the question: What are the skills that pupils in the fourth year of secondary school, Computer Science section, need to acquire in order to process algorithms that require purely mathematical knowledge?

4.1. A priori analysis

4.1.1. Presentation of the test

The answers are collected in the form of written data, in the blank spaces provided. The choice of this test is linked to our problem, which is to test whether students are capable of processing a recurrent algorithm after a mathematical analysis.

The test[5] consists of two exercises, most of which are in accordance with computer programs. The test consists of two exercises, most of which conform to the computer science syllabus. They are fairly close to the applications proposed in the teachers' courses presented in part C of the thesis. We have considered certain particularities for these exercises which can be summarised as follows: clarity of vocabulary in the statements and simplicity of calculations in the answers.

[4] The chosen population was made up of three classes in the fourth year of secondary school, in the computer science section, with a total of 64 pupils. The experiment was carried out in three Tunisian schools (Secondary schools: Cité Ennasr, Cité Ibn Khaldoun, Imam Moslem) belonging to two Regional Departments of Education (Ariana and Tunis 2).

[5] Appendix 1

https://doi.org/10.61856/xa3tkc02

We noted that the recommendations of computer science teaching aids operate a shift between solving mathematical problems by processing algorithms in a general and implicitly computerized way.

Briant (2013) has distinguished the double transposition of solving a mathematical problem with a view to programming it. Using the two exercises proposed in this test, we want to identify their reactions to the first transposition based on their productions.

4.1.2. Suggested answers for Exercise 1

$U_0 = 1$, for any natural number n, $U_{n+1}= 2U_n +1$

Part1 :

1/ $U_1 = 2U_0 +1 = 2\times1+1 = 3$
$U_2 = 2U_1 +1 = 2\times3+1 = 7$
$U_3 = 2\ U_2 +1 = 2\times7+1 = 15$

2/ Consider the property P(n): «Where n is a natural number, $U_n = 2^{n+1} -1$».

Initialisation: for n =0, we have $2^{0+1}-1=1= U_0$, then P (0) is true.

Heredity: Let $n \geq 0$, assume that P(n) is true and show that (P(n) implies P(n+1)) is true.

Let $n\geq0$, We have: $U_{n+1}= 2\ U_n +1$ and as $U_n = 2^{n+1}-1$ then: $U_n= 2.\ (2^{n+1}-1) +1= 2^{n+2} -1$

So, for any $n\geq 0$; (P(n) implies P(n+1)) is true.

Conclusion: According to the mathematical induction principle, we can affirm that for any natural integer n 0, P(n) is true, that is to say for all, $n\geq 0$, $U_n= 2^{n+1}-1$.

Part2:

There are two ways of answering this question, in line with the syllabus for the fourth year of secondary school, computer science section: an iterative solution or a recursive solution.

The sequence (U_n) is recursive of order 1:

Algorithm1: *Iterative solution*

```
Procedure Term-N (n: integer)
Beginning
  | U←1
   Write (U)
   For i from 1 to (n-1) do
   U←2*U + 1
   Write (U)
  | End for
End
```

Algorithme2: *Recursive solution*

```
Function Term (n: integer): integer
Beginning
   If n = 0 then
     | return 1
   If no return (2*Term (n-1) +1)
End
```

4.1.3. Suggested answers for Exercise 2

Part1:

$F_2 = F_1 + F_0 = 2+1 = 3$
$F_3 = F_2 + F_1 = 2+3 = 5$
$F_4 = F_3 + F_2 = 5+3 = 8$

Part2:

There are two solutions to this question in accordance with the syllabus for the fourth year of secondary school, computer science section: an iterative solution and a recursive solution.

Algorithm1: *Iterative solution*

```
Fibo function (n: integer): integer
Beginning
| F0 = 1
F1 = 2
For i from 2 to (n-1) do
| F← F0 + F1
F0← F1
F1←F
End for
Return F
End
```

Algorithme2: *Recursive solution*

```
Fibo function (n: integer): integer
Beginning
If n = 0 then
| return 1
If no if n = 1 then
| return 2
If no
| return Fibo (n-1) + Fibo (n-2)
End if
End
```

4.1.4. A priori analysis of expected answers

It is expected that many students will be able to answer the first question in Part 1 correctly in both exercises. However, it is also expected that the majority of them will have difficulty in using mathematical induction. This type of reasoning, although introduced in secondary education as a proof algorithm, is often perceived by students as a complex procedure (Soltani, 2019, 2023). They may struggle to grasp its conceptual foundations, especially as this method is mainly approached in a formal and algorithmic way.

It is likely that these difficulties are accentuated by a lack of adequate training in mathematical situations which require both the formulation of conjectures and the rigorous justification of answers. It is anticipated that students who have received more in-depth training on the articulation of inductive reasoning and algorithms will be better equipped to answer these questions. However, for the majority, the challenge will be to

https://doi.org/10.61856/xa3tkc02

reconcile the theoretical aspects with the practical requirements of writing algorithms, and to overcome the abstraction that characterises mathematical induction.

For the second part of each exercise, it is expected that some students will encounter obstacles in solving an algorithm. The writing of an algorithm follows specific conventions, such as those defined in the teaching aids (2022-2023).

4.2. A posteriori analysis

It should be noted that the number of students is large (64) and sufficient to achieve our objective, as the students taking part in this test belong to different schools, and these institutions have a very significant success rate in the national baccalaureate examination.

In our analysis, we proceeded to read the responses collected, which we grouped into types of response. Responses that complied with the institutional contract were classified as 'admissible'. We classified the responses that dealt with recurring algorithms into three categories: 'incomplete', 'valid' and 'invalid'. The validity of a treated algorithm implies compliance with all conventions. Among those that have processed incomplete recursive algorithms, we find those that do not respect one of the conventions or that present syntactic errors. We choose to analyse the results of the first part (Part 1) of both exercises, which is purely mathematical, and then the second part (Part 2) of both exercises, which concerns the recurrent algorithm in computer science. Finally, we will establish a correlation between the two parts of each exercise.

4.2.1 Results and interpretation

The results of reading the responses of the students in the total sample for the first mathematical part of the two test exercises are summarised in the statistical table below.

Table 2: Total responses from students in the total sample for the first part of both test exercises

Answers			No answer	Valid answer	Invalid answer
Exercise1	Part 1	Question1	00	63	01
		Question2	08	12	44
Exercise2	Part 1		17	40	07
Percentages			27%	60%	27%

The first observation that immediately emerges from this table is the above-average rate of mathematically valid knowledge (60% of students have answers). The students in this sample were also able to calculate the first terms of a recurrent sequence (63/64 valid answers) and the first terms of a Fibonacci sequence (40/64 valid answers). However, there were some gaps in the students' answers to the second question in the first part of the first exercise (only 12/64 of the answers were valid). The answers to this question showed that the pupils had not mastered mathematical induction (Soltani and Chellougui, 2023). Thus (27%) of the pupils did not give any answers, a percentage that is an indicator of mathematical weakness among some pupils.

The results of reading the responses from this sample to the second part of both test exercises are summarized in the following statistical table:

Table 3: All the responses of students from the total sample to the second part of both test exercises.

Answers		No answer	Incomplete algorithm	Valid algorithm	Invalid algorithm
Exercise1	Part 2	07	02	31	24
Exercise2	Part 2	13	22	20	09
Percentages		16%	19%	40%	26%

https://doi.org/10.61856/xa3tkc02

The first observation to be made from this table is that the rate of computer literacy is close to average (40% of the students in this sample processed the valid algorithms in the test). The majority of students processed the first algorithm for calculating the terms of the recursive sequence with an iterative solution, and the second algorithm for calculating the terms of the Fibonacci sequence with a recursive solution. As a result, 19% of the students dealt with incomplete algorithms. The majority of the algorithms in the second part of the second exercise did not respect the generalised conditional control structure because some students forgot the initial natural numbers of the Fibonacci sequence 1 and 2 and gave other numbers in their place. These results show that a significant number of students have memorised the treatment of algorithms in the form of ready-made models, which may hinder their creativity and thinking in this area.

The results of the analysis of the responses of students in this sample to the correlation between Parts 1 and 2 of the two exercises are summarized in the following statistical table:

Table 4: Total responses of students in the total sample: Correlation between the two parts in both exercises.

Answers to both parts of each exercise (part1, part2).	Valid / Incomplete	Valid / Valid	Valid / Invalid	Invalid / Valid	Others
Exercise1	00	16	02	09	37
Exercise2	14	15	04	04	27
Percentages	11%	24%	5%	10%	50%

The first thing to note is the low rate of valid answers in the test (24% of students gave valid solutions for each of the two exercises). Furthermore, (11%) of the students' answers showed that the first part was valid and the second part incomplete. This shows that students' mathematical knowledge has an impact on their handling of recurrent algorithms in computer science. What interests us is the lack of mastery of mathematical induction among the majority of students in three secondary schools. This has an impact on the validity of the first exercise. Thus, a good number of students do not respect the generalized conditional control structure in the treatment of the algorithms of the second part of the second exercise by the fact that some of them have forgotten the initial natural integers of the Fibonacci sequence 1 and 2 and gave other numbers in their places. These results show that a significant number of students have memorized the processing of algorithms in the form of ready-made models.

In this regard, it will illustrate below two examples of student productions, designated by student A and student B.

4.2.2. Production by student A[6]

Comments:

Exercise 1:

Proof of the first part: from the evidence given in the first question, this student has correctly calculated the first terms of the following order1 (recurring sequence) presented. He was therefore able to reason by mathematical induction in the second question: Indeed, the steps in his reasoning were demonstrated. The initialization was badly written, the heredity was justified and the conclusion identified. Moreover, the writing in this respect contains linguistic errors (on the one hand, the property P(n) to be demonstrated is not identified and the initialisation stage contains semantic errors (the written sentence makes no sense). On the other hand, the heredity step contains syntactic errors that manifest themselves in the absence of the universal quantifier and the invisibility of the implication of heredity). (Soltani and Chellougui, 2023).

[6] Appendix 2

https://doi.org/10.61856/xa3tkc02

This suggests that the student is having difficulty in writing relevant evidence requiring mathematical induction. The answers are valid because the school accepts this type of writing for mathematical induction. For the second part algorithm: the student has given an iterative solution to process an algorithm that calculates the terms of the recurring sequence (Un) of order1. Its writing respects the general form of an algorithm, the syntaxes of algorithmic structures, the declaration of procedure and the iterative control structures. This algorithm is valid.

Exercise 2:

In the proof of the first part, the student has explicitly calculated the first terms of the Fibonacci sequence. This proof is valid. For the algorithm in Part 2, the student has proposed a recursive solution for dealing with an algorithm. His writing respects the general form of an algorithm and the declaration of the Fibo function is correct. However, the generalised control structure contains errors on lines 4 and 6. The initial natural numbers 1 and 2 in the sequence are replaced by 0 and 1. This algorithm is considered incomplete.

4.2.3. Production by student B[7]

Comments:

Exercise 1:

In the proof of the first part, the student has correctly calculated the first terms of the recursive sequence of order 1 presented. However, in the second question he is unable to reason by mathematical induction. The steps in his reasoning have not been demonstrated. We deduce from this that this student has difficulty with mathematical induction. We consider the first answer to be valid and the second invalid. For the algorithm in the second part: the student gave an iterative solution for an algorithm that calculates the terms of a sequence (Un). It respects the general form of an algorithm, the syntax of the algorithmic structures, the declaration of the procedure and the iterative control structures. This algorithm is valid.

Exercise 2:

In the proof in the first part, the student has calculated the first terms of the Fibonacci sequence. This proof is valid. For the algorithm in the second part, the student proposed a recursive solution for processing an algorithm. His writing respects the general form of an algorithm and the declaration of the Fibo function is correct. However, the generalised control structure contains errors on lines 3 and 4. The initial natural numbers 1 and 2 in the sequence are forgotten, so this algorithm is considered incomplete.

5. Discussion

In the new computer science programmes, algorithms occupy a central place in mathematical activity. Indeed, the main objective of this discipline is to deal with algorithms that enable mathematical problems to be solved. However, the fundamental link between algorithms and problem solving is never clearly explained, and it seems that the teaching of algorithms is mainly focused on learning rigour. The development of algorithms in computing requires the application of rules of mathematical logic, and the new syllabuses emphasise the link between reasoning, formalisation and logic. Many mathematical concepts are used, such as mathematical induction and recursion.

The results of the analysis of the productions of the students in the 4th year (Computer sciences section), highlights persistent difficulties in developing and applying mathematical induction. These difficulties encountered by students are sometimes related to the lack of logical knowledge (Chellougui, 2009; Soltani and Chellougui, 2023). As regards the handling of algorithms in the test, the results were acceptable overall. However, a significant number of students are experiencing difficulties with recurrent algorithms related to recursion. These difficulties are manifested by a tendency to memorize and reproduce models of ready-made algorithms, thus treating these algorithms

[7] Appendix 3

in a mechanical rather than comprehensive way. This situation seems to be the consequence of a lack of effective didactic transposition of mathematical analysis into the computerized algorithm, aggravated by an institutional didactic vacuum (lack of guidelines in official curricula and school textbooks).

These results showed that the mathematical and computer science knowledge content available to students should not be a prerequisite for solving mathematical problems using the algorithmic aspect of computing.

6. Conclusions

In conclusion, the knowledge taught in the Tunisian secondary school computer science curriculum emphasises an algorithmic and programming approach, which is in itself a type of logical reasoning. However, since the official texts do not provide a clear picture of what is expected in algorithmics, its place between mathematics and computer science remains unclear. In addition, the analysis of student output highlights the complexity of teaching/learning recurrent algorithms and recursion in computer science. It is necessary to pay close attention to the notions of language, logic and mathematical reasoning when studying the educational reform in computer science at secondary level.

References

Briant, N. (2013). *Étude didactique de la reprise de l'algèbre par l'introduction de l'algorithmique au niveau de la classe de seconde du lycée français.* Thèse de doctorat, université Montpellier 2, Montpellier, France.

Chellougui, F. (2009). L'utilisation des quantificateurs universel et existentiel en première année universitaire entre l'explicite et l'implicite. *Recherches en didactique des mathématiques (RDM)*, Vol.29, n°2, pp.123-154. La Pensée Sauvage Editions.

Laval, D. (2018). *L'algorithmique au lycée entre développement de savoirs spécifiques et usage dans différents domaines mathématiques*. Education. Université Sorbonne Paris. Français. https://tel.archives-ouvertes.fr/tel-01943971.

Leon, N., Modeste, S., & Durand-Guerrier. V (2020). Récurrence et récursivité: analyses de preuves de chercheurs dans une perspective didactique à l'interface mathématiques-informatique. INDRUM2020, Université de Carthage, Université de Montpellier, Sep 2020, Cyberspace (virtually from Bizerte), Tunisie. hal-03113854

Leron, U., & Zazkis, R. (1986). Computational recursion and mathematical induction. For the learning of Mathematics, 6(2), 25-28.

Modeste, S. (2012). *Enseigner l'algorithme pour quoi? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ?* Thèse de doctorat, université Joseph Fourier, Grenoble, France.

Soltani, W. (2022). Le raisonnement inductif dans l'enseignement secondaire tunisien : Interaction entre les mathématiques et informatique, In C. A. Adihou & F. Chellougui (Eds), *Actes du3ème colloque de L'Association de Didacticiens des Mathématiques Africains (ADiMA3),* pp. 260-270, Hammamet 15-20 août 2022. https://adima3.sciencesconf.org/resource/page/id/27.

Soltani, W. (2023). Une approche didactique sur le raisonnement par récurrence en classe de 3ème année section Mathématiques. In Achour, S., Ben Nejma, S.,Dhieb, M., Ghedamsi, I., Khalloufi, F., & Kouki, R. (Eds.). *Actes du 13ème Colloque de Didactique des Mathématiques (ATDM 2023),* pp. 43-52. Editions ATDM. ISBN 978-9938-78-716-0.

Soltani, W., & Chellougui, F. (2023). Analyse des erreurs de nature langagière chez les élèves en arithmétique. *Mediterranean Journal of education*, 2023, 3(2), p269-278, ISSN: 2732-6489.

Soltani, W., & Chellougui, F. (2024). Inductive reasoning: Problems, methods of justification and interaction between mathematics and computer science. The International Innovations Journal of Applied Science (IIJAS) Vol. 1, N0.2. https://doi.org/10.61856/095nzv52

Secondary programme

- *Specific computer teaching aids* (2022), Computer sciences section. Republic of Tunisia Ministry of Education.
- *Algorithmic conventions* (2022), Ministry of Education General Directorate for Curricula and Continuing Education.
-*Implementing algorithmic conventions in Python* (2022), Ministry of Education General Directorate for Curricula and Continuing Education.

The International Innovations Journal of Applied Science (IIJAS) Vol. 2, No.1, 15-03-2025
المجلد الثاني العدد الاول 15-3-2025 (IIJAS)مجلة ابتكارات الدولية للعلوم التطبيقية
https://doi.org/10.61856/xa3tkc02

Appendix 1: Test for 4th year secondary students, Computer Science section (*English translation*)

4th Computer sciences section *November 2022*

Exercises on recurrent algorithms

We offer you two independent exercises on recurrent algorithms.
Please write your answers in the sections reserved for answers.

Exercise 1

Consider the sequence of numbers $(u_n)_n$ defined by:

$u_0 = 1$ and for any natural number n, $u_{n+1} = 2\,u_n + 1$.

Part1

1/ Calculate the following terms:

u_1 =……

u_2 =……

u_3 =……

2/ Show by recurrence (Mathematical induction) that: for any natural number n, $u_n = 2^{n+1} - 1$.

Part2

Suggest an algorithm for the **TERM-N** procedure which displays the first **n** terms of the sequence $(u_n)_n$.

Exercise 2

Consider the Fibonacci sequence $(F_n)_n$ defined by:

$F_0 = 1$, $F_1 = 2$ and for naturel number n≥ 2, $F_n = F_{n-1} + F_{n-2}$.

Part1

Calculate the following terms:

F_2 =……

F_3 =……

F_4 =……

Part2

Suggest an algorithm for the recursive **Fibo** function that calculates the nth of the Fibonacci sequence $(F_n)_n$.

Appendix 2: *Proof of test produced by student A*

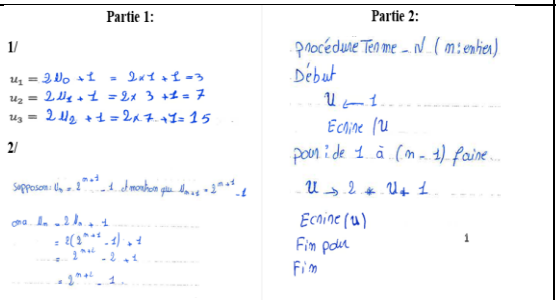
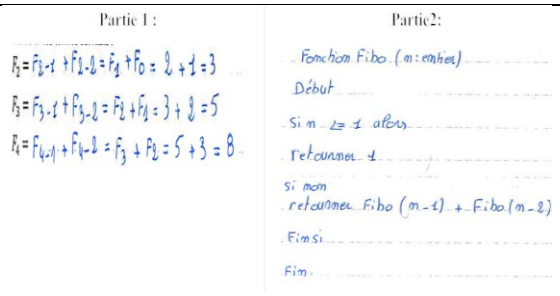
<table>
<tr><td colspan="2">Exercise1:</td><td>Exercise2:</td></tr>
<tr><td colspan="2">Partie 1:
1/
$u_1 = 2u_0 + 1 = 2 \times 1 + 1 = 3$
$u_2 = 2u_1 + 1 = 2 \times 3 + 1 = 7$
$u_3 = 2u_2 + 1 = 2 \times 7 + 1 = 15$
2/
$\forall n \in \mathbb{N}$ on a $U_n = 2^{n+1} - 1$ vraie
supposons que $U_n = 2^{n+1} - 1$
Demontrons alors que $U_{n+1} = 2^{(n+1)+1} - 1$
on a : $U_{n+1} = 2U_n + 1$
$= 2(2^{n+1} - 1) + 1$
$= 2^{n+2} - 2 + 1 = U_{n+1}$
conclusion $\forall n \in \mathbb{N}$ on a $U_n = 2^{n+1} - 1$

Partie 2:
Procédure terme_N (n : entier)
Début
u← 1
Ecrire (u)
Pour i de 1 à (n-1) faire
u← 2*u+1
écrire(u)
Fin Pour
Fin</td><td>Partie 1:
$F_2 = F_{2-1} + F_{2-2} = F_1 + F_0 = 3$
$F_3 = F_{3-1} + F_{3-2} = F_2 + F_1 = 5$
$F_4 = F_{4-1} + F_{4-2} = F_3 + F_2 = 8$

Partie 2:
Fonction Fibo (n : entier)
Début
Si n <= 1 alors
retourner 1
Si non
retourner Fibo (n-1) + Fib(n-2)
Fin si
Fin</td></tr>
<tr><td colspan="2">English translation</td><td>English translation</td></tr>
<tr><td>Part1:
2/ ∀n∈IN we have $u_0 = 2^{0+1}-1 = 1$ true
suppose that $u_n = 2^{n+1}-1$
Let us then show that:
$u_{n+1} = 2^{(n+1)+1}-1$
We have: $u_{n+1} = 2\,u_n +1 = 2\,(2^{n+1}-1)$
$=2^{n+2}-1 = u_{n+1}$
Conclusion ∀n∈IN we have $u_n = 2^{n+1}-1$</td><td>Part2:
Procedure term-N (n: integer)
Beginning
u←1
write (u)
for i of a (n-1) do
u→2*u+1
write(u)
End for
End</td><td>Part2:
Function Fibo (n: integer)
Beginning
If n <=1then
return 1
If no
return Fibo (n-1) +Fib(n-2)
End if
End</td></tr>
</table>

https://doi.org/10.61856/xa3tkc02

Appendix 3: *Proof of test produced by student B*

<table>
<tr><td colspan="2">Exercise1:</td><td>Exercise2:</td></tr>
<tr><td colspan="2"> 
</td><td> 
</td></tr>
<tr><td colspan="2">English translation</td><td>English translation</td></tr>
<tr><td>Part1:
2/ Suppose $u_n = 2^{n+1}-1$ and show that $u_{n+1}= 2^{n+2}-1$
We have $u_n = 2\ u_n +1$
$= 2\ (2^{n+1}-1)$
$= 2^{n+2}-2+1$
$= 2^{n+2}-1$</td><td>Part2:
procedure term-N (n: integer)
Beginning
u←1
write (u
for i of a (n-1) do
u→2*u+1
write(u)
End for
End</td><td>Part2:
Function fibo (n: integer)
Beginning
If n <=1then
return 1
If no
return Fibo (n-1) +Fib(n-2)
End if
End</td></tr>
</table>