Comparison and Efficiency of Different Algorithms to Count and Determine the Composition of All Possible Riffle Shuffles Used in Magic

Pierre Schott* and Zhenghan Pei

LISITE, Institut Supérieur d'Electronique de Paris (ISEP), Paris, France
Email: pierre.schott@isep.fr (P.S.); zhenghan.pei@eleves.isep.fr (Z.P.)
*Corresponding author

Manuscript received July 26, 2023; revised August 13, 2023; accepted August 31, 2023; published February 20, 2024

***Abstract*—As part of a Technical and Scientific Project offered to L2 students of ISEP over a period of four months at the rate of one hour per week supervised, a study of the composition and the total number of all riffle shuffles used in magic was requested. The riffle shuffle consists of taking a deck of cards, cutting it into any two parts, then inserting randomly the cards of the first packet into the second. In first approximation there are 52! possible riffle shuffles. This number is an upper limit since the cards in a subdeck cannot change their relative position to their neighbours. A card A below a card B in the same subdeck will not be able to find itself above in the final shuffle. The first goal of the project is to compare one single algorithm implemented either recursively or iteratively in terms of execution time depending on the number of cards. The second goal is to compare the implementation of the same algorithm (either iterative or recursive) in several different programming languages (Python, Language C, Matlab, etc.) and to compare execution times according to the used language. Thus, a recursive algorithm was proposed to the students. All the students tried to code this algo and make it iterative … except one! He imagined another algorithm. The algorithm of the student and the teacher were compared in terms of execution time: do the student exceed the master? The aim of this article is to share my experience. I leave it to you to imagine educational sequences around this project in adequacy with your desires and the level of your students.**

***Keywords*—informatics langage, magic, riffle shuffle, Matlab, C, Python, recursive algorithm, iterative algorithm, higher education**

I. Introduction

A. Why Use a Magic Shuffle as the Project?

The teacher is fascinated by magic (or rather, conjuring) and for many years he has used this way of teaching, both in his physics classes and as higher education teacher trainer.

The aim of the magician is to hide the principles he uses (using maths, physics, psychology, sleight of hand, etc.) by disguising the trick so that the audience has no way of discovering how it is done; thus allowing the magic to remain.

The teacher can do exactly the opposite: unraveling a magic trick to highlight the principles used!

B. The Specifications: Count and Determine the Composition of All Possible Riffle Shuffles

The subject was given for the third semester students in ISEP which is a higher education school in France (named 'Grandes écoles d'ingénieurs') as a Scientific and Technical Project (STP) over a period of four months at the rate of one hour per week supervised.

Fifteen students have chosen this project and have had a 40 hour Python course before the project. A third of them have achieved success for the entire project but following the teacher's algorithm. Only one student has achieved the project by following his own algorithm!

Whatever the chosen algorithm, we need to obtain the number and the composition of all riffle shuffles between a p card deck and a q card deck (p+q=n cards).

We introduce here firstly many algorithmic ameliorations and secondly a pedagogic point of view by giving, through the article construction, a possible approach for teaching a high level informatics language.

II. Literature Review

Lesser and Glickman wrote in [1]: "The use of magic has recently been gaining attention in advancing areas of science such as cognitive neuroscience [2] (Martinez-Conde & Macknik, 2008), biology [3] and cognitive psychology [4]. Several papers and books are available on classroom uses of magic involving mathematics, especially elementary algebra [5]". Then Wissemenn and Watt added: "For hundreds of years, magic tricks have been employed within a variety of pedagogic contexts, including promoting science and mathematics, delivering educational messaging, enhancing scepticism about the paranormal, and boosting creative thinking for product design [6]".

A theoretical solution of the number of riffle shuffles is given by Aldous and Diaconis [7]. However, the composition was not given. A first mathematical and algorithmically study to determine the composition of all riffle shuffles is given in french by Lachal [8].

However, there appear to be no books and only a few isolated articles presenting the use of magic to explain concepts in probability or statistics. Not only mathematicians and scientists, but the general public as well have shown much interest in the card randomization problem, as reported in popular science periodicals and major news media [9].

Since 1955 until today, determining the number of riffle shuffles to be made so that an initially arranged ordered deck is well shuffled, is a problem that fascinates mathematicians. In 1955, the Gilbert–Shannon–Reeds model is a probability distribution on riffle shuffle permutations that has been reported to be a good match for experimentally observed outcomes of human shuffling [10] and that forms the basis for a recommendation that a deck of cards should be riffled seven times in order to thoroughly randomize it is given by Diaconis in 1992 [11].

An another study by Mann from Harvard was published in 1998 [12]. Then in 2014, Aldous has done a lecture in Berkley university introducing the Markov chain.

Colm Mulcahy, whose mentor was Gardner ([13, 14]), has worked on the relation between magic and science since 40 years and has done recently a minicourse in 2019 [15].

The most important is that Magic is one of the 20 modalities of fun identified by Lesser and Pearl [15].

[16] as having potential for motivating students in statistics courses.

III. Materials and Methods

First the goals of the teacher and the student are presented and thus the definition of the riffle shuffle.

A. The Goals

1) The initial goal of the teacher: Iterative vs recursive method

The aim of the professor is to compare the execution time of the same implemented algorithm either with an iterative method or with a recursive method. This is an ultra-classic problem. The teacher hopes that the approach with the cards would be both more fun and easier for students to find the algorithm.

2) The second goal of the teacher: Matlab vs Python

The professor has coded his algorithm with Matlab and his students learned Python programming, the secondary objective was to compare on the same computer the execution time of his algorithm encoded in Python and in Matlab, whether for the iterative or recursive method.

3) The main objective proposed by the student: Algorithms comparison

The algorithm that meets the project’s specifications is not trivial. Thus, the teacher introduced his algorithm to the students during two sessions using the cards.

A student distinguished himself by finding another algorithm! So the idea was quite naturally to compare the execution time of our algorithms with both the iterative method and the recursive method.

4) The second goal brought by the student: coding in C to manage memory and improved the efficiency of algorithms

Whether in Python or Matlab, the interpreter manages automatically the memory. If all function calls are done by values, it is very clear that the recursive method will saturate the computer’s memory faster than the iterative method. But what happen if we call these functions by address?

B. The Riffle Shuffle

Let us imagine a n card deck be cut into two parts, called subdecks. Each subdeck is riffled with the hands, and the cards of each subdeck become entangled, as presented in Fig. 1.

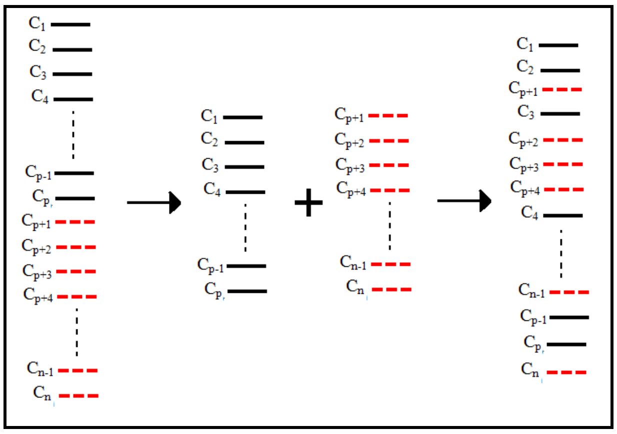


Fig. 1. Example of a cut for a n card deck and then a riffle shuffle.

1) Number of shuffle for a given cut of the deck

The deck can be cut at different places. So we end up with two subdeck consisting of p and q cards. We can determine analytically the number of possible shuffles, but not the composition of these.

2) Number of shuffle for all possible cuts of the deck

Let us think about what we have, a n card deck. To know the total number of possible shuffles (named M(n)), simply add all possible shuffles for each of the (n-1) possible cuts.

However, how can we find the composition? It is humanly ‘impossible’ if the number of cards is greater than seven because the number of different riffle shuffles is given by:

$$M\ (n)=2^n - n. \tag{1}$$

A demonstration was proposed in [17].

However, this number did not count of the "duplicates", which are, identical shuffles obtained by different cuts.

C. Computer Representation of a Deck of Cards

To represent the card deck, we allocate a value to each single card of the deck either between 1 and 52 or between 101 and 113, 201 and 213, 301 and 313 and finally 401 and 413 (the hundred value represents the card family). But there are so many others possibilities.

Whatever the choice, a bijection between each single card of the deck and a number is created.

A new bicycle^{TM} card deck is presented in Fig. 2. A computer representation of this deck is presented in Fig. 3 whose card number is between 1 and 52.

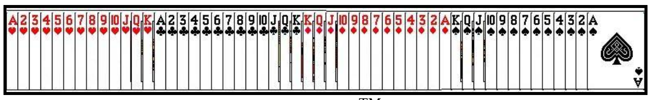


Fig. 2. A new bicycle^{TM} card deck.

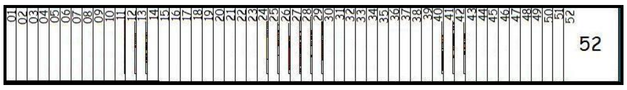


Fig. 3. A new bicycle™ card deck informatics representation with card numbered between 1 and 52.

IV. Teacher’s Algorithm

The (p+q) card deck is cut and then, one of the two subdecks is broken down into a series of q cards. The first card is shuffled into the p card deck and all the resulting decks are shuffled with the second card and so on.

A. All Possible Cuts from a n Card Deck

Let us take a N card deck in this order:

$C_1 - C_2$ - … - Ci - … - C_N. Only (N-1) ways are possible to cut this deck in two subdecks as presented in Fig. 4:

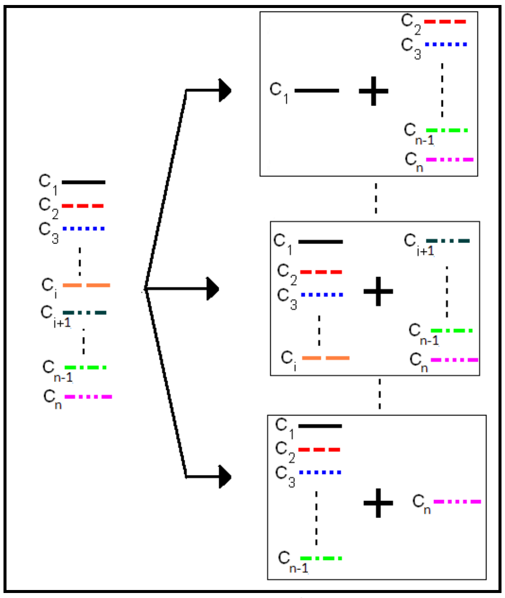


Fig. 4. All possible cuts of a n card deck.

B. Break down One of the Both Subdeck into a Series of Single Cards

Let us imagine an i < N card deck in this order : $C_1 - C_2$ - … - Ci The top card C_1 is put away and the new subdeck contains (i-1) cards in this order : C_2 - … - Ci. This operation is repeated as many times as the number of card of the "new" subdeck is equal to one, as presented in Fig. 5.

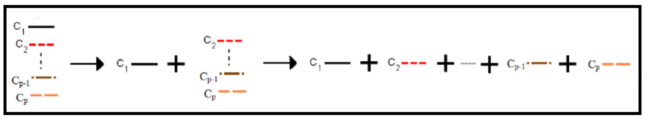


Fig. 5. Decomposition of the p card subdeck in order to shuffle with a q card subdeck.

C. Shuffle One Card into a q Card Deck

The operating mode used in order to obtain the composition of all possible riffle shuffles between one card and a q card deck is presented in Fig. 6.

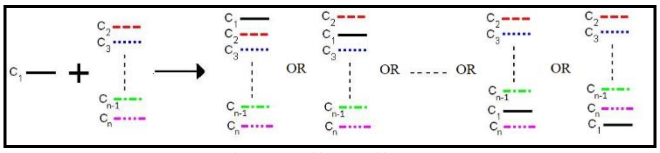


Fig. 6. Modus operandi to find all riffle shuffles between a 1 card deck and a q card deck enumeration and decomposition.

D. Shuffle One Card into a q Card Deck above the Position i

The operating mode used in order to find all of possible riffle shuffles between one card and a q card deck over the card C_i is presented in Fig. 7.

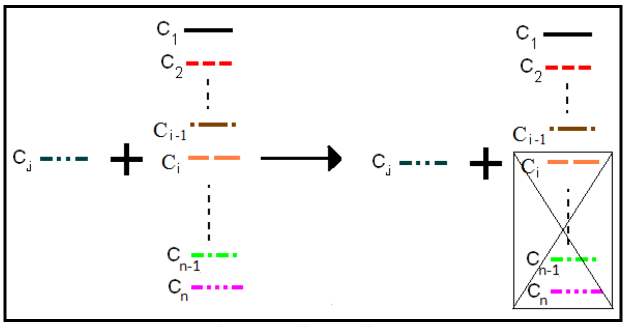


Fig. 7. Modus operandi to find all riffle shuffles between a 1 card deck and a q card deck over the card Ci (enumeration and decomposition).

E. Teacher's Algorithm

To determine the number and the composition of all possible riffle shuffles, thanks to the study with a few cards, we have to follows this strategy:

If the p card subdeck contains more than one card: the deck is decomposed in p single cards as described in Fig. 5 (the last card is C_p.). Otherwise we have got exactly one card, say C_p.

The C_p card is shuffled in the q card subdeck, as presented in Fig. 6.

All the calculated riffle shuffles must be shuffled with the card C_{p-1}. All calculated riffle shuffles must be shuffled with the card C_{p-2} and so on, as presented in Fig. 8.

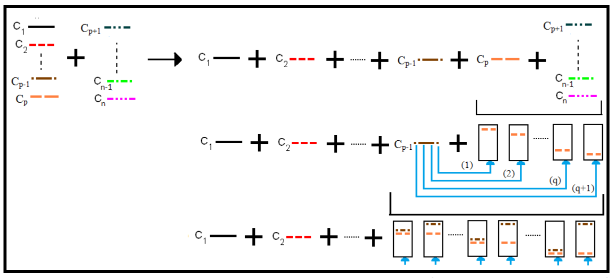


Fig. 8. Algorithm to calculate and determine all of the compositions of possible riffle shuffles between a p card subdeck and a q card subdeck.

F. Computer Implementation

An iterative and recursive implementation was carried out in Matlab by the teacher and in other languages by the student.

V. Student's Algorithm

The student's algorithm breaks down into two distinct parts. Firstly, the 'insertion' algorithm calculates almost instantaneously all the insertion vectors (which allows to know the total number of possible shuffles). Secondly, the 'combination' algorithm, thanks to the insertion vectors, allows to obtain the exact composition of each shuffle.

A. The Main Idea: The Insertion Vector or How not to Work Directly on the Card Deck

Instead of working directly on an array that represents the card deck, the student uses an insertion vector $[a_0, a_1, \dots, a_p]$ for a p card deck. The value of $(a_j)_{0 \le j \le p}$ represents the number of cards inserted. The position in the array represents the place where these a_j cards are inserted in the p card deck. Fig. 9 illustrates this process.

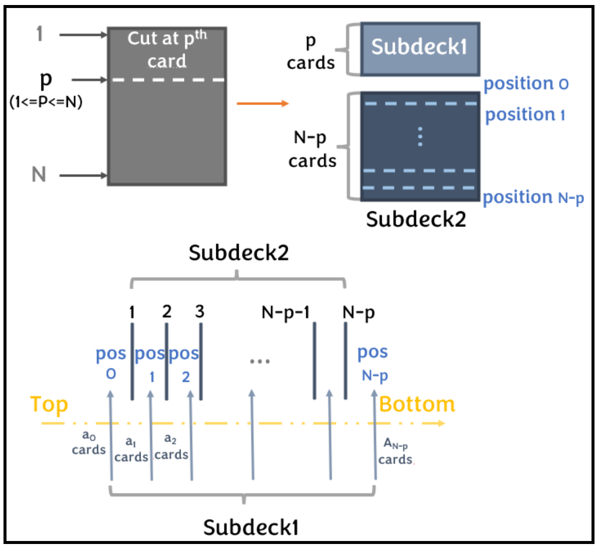


Fig. 9. Student's algorithm broken down into 2 phases: insertion and combination.

B. *Finding the Total Number of Possible Shuffles Is Like Generating All Insertion Vectors*

For each possible cut, it is enough to generate all possible insertion vectors. Each vector allows to manipulate array whose size can be up to half the size of the arrays used by the teacher's algorithm. We therefore expect this algorithm to be much faster than that the teacher ones!

C. *Determine the Composition of a Shuffle from the Insertion Vector*

Once the insertion vector has been determined, the previously established single match must be used to determine the final shuffle composition by inserting array fragments of the first packet subdeck into the second subdeck. A computer representation with list is presented in Fig. 10.

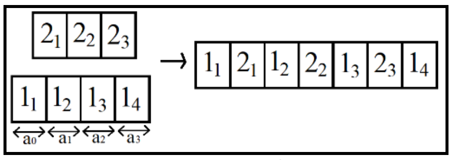


Fig. 10. Computer representation by list of the 'combination' algorithm.

D. *Determine the Composition of All Shuffles: Use both Previous Algorithms Presented*

To determine the composition of all shuffles, we must therefore use the algorithm presented in the previous section with all the insertion vectors found.

E. *Computer Implementation*

An iterative implementation was carried out in all informatics languages by the student.

VI. Results: Comparison between Different Programming Languages and Algorithm Complexity

Firstly, the figures presented in this paragraph show that, for the same algorithm, programming in C language makes it possible to obtain results faster than in Python and even faster than in Matlab. Secondly, using the t_{n+1}/t_n ratio, we can determine when the RAM is saturated and when the computer stores in ROM.

A. *Computing Time for the Teacher's Algorithm Written in C, Python and Matlab*

Fig. 11 clearly shows that the same iterative teacher's algorithm written in C is much more efficient than the one written in Python and Matlab. The memory management is automatic in Python and Matlab; it is not the case in C. Thus the good efficiency of the C-programs can be explained by optimized C memory management.

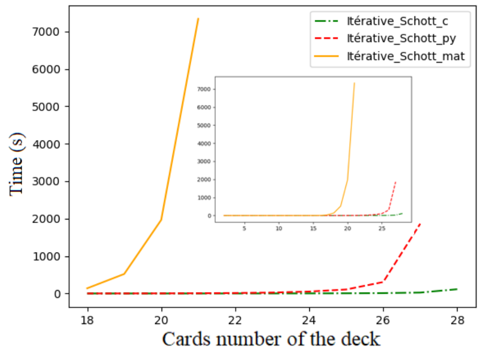


Fig. 11. Computing time vs card number in the deck for the iterative teacher's algorithm written in C, Python and Matlab.

Fig. 12 represents the logarithm of the computing time as a function of the number of cards contained in the deck. It shows that the property previously seen is still valid for the recursive teacher's algorithm.

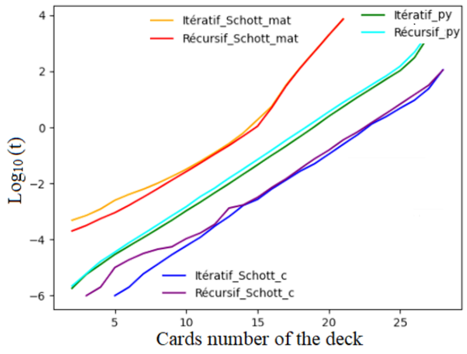


Fig. 12. Logarithm of computing time vs card number in the deck for the iterative and recursive teacher's algorithm written in C, Python and Matlab.

B. *Computing Time for the Student's Algorithm Written in C, Python and Matlab*

Fig. 13 and Fig. 14 clearly show that the same student's algorithm written in C is much more efficient than the one written in Python and Matlab.

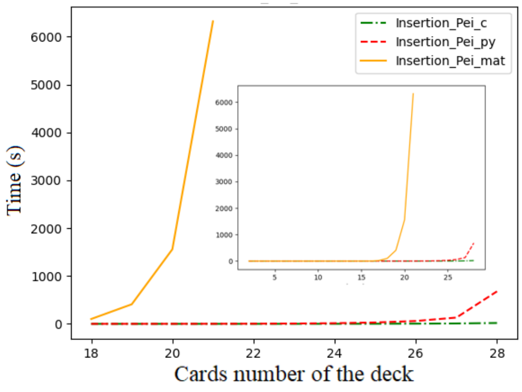


Fig. 13. Computing time vs card number in the deck for the insertion student's algorithm written in C, Python and Matlab.

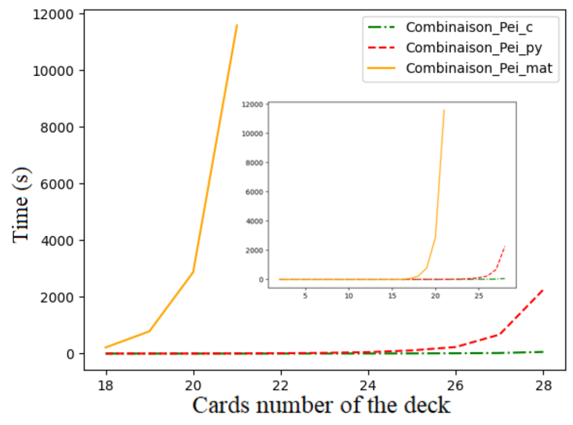


Fig. 14. Computing time vs card number in the deck for the combination student's algorithm written in C, Python and Matlab.

C. Plot $t_{(n+1)} = f(t_n)$ Allows to Determine Roughly the Algorithm Complexity and Automatic Memory Management

The curves in the Fig. 12 are almost straight lines (or are very close to the linear regression line). Thus, we can conclude that Log(t) ∝ n.

The curve in Fig. 15 represents the computing time vs iterative professor's algorithm written in Matlab. It rises steeply and then descends steeply.

This means that t_{15}=3 t_{14}, t_{16} = 6 t_{15} and t_{17} = 3 t_{16}. So there has been a significant change that can be attributed to ROM storage because RAM should (or seems to) be saturated.

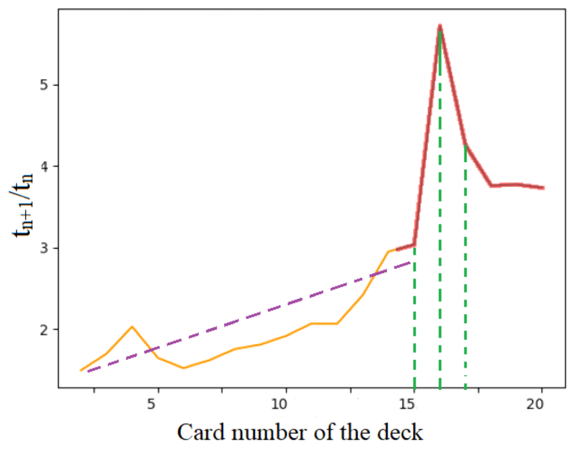


Fig. 15. The computing time vs iterative professor's algorithm written in Matlab.

VII. Comparisons between Iterative and Recursive Programming

The figures also show that there is almost no difference between an iterative implementation and a recursive implementation, which seems to be non-intuitive result - until a certain value n_0 then a divergence between the curves appears . Then we show our memory management in C.

A. Comparison in Python

Fig. 16 shows the computing time vs iterative and recursive professor's algorithm written in Python. We determine the value n_0, value from which the iterative method is more efficient than the recursive method.

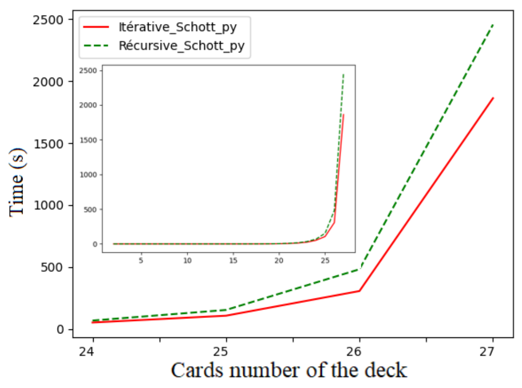


Fig. 16. The computing time vs iterative and recursive professor's algorithm written in Python.

B. Memory Management Is the Key of Algorithm Implementation Efficiency

The Fig. 12 shows that the Matlab implementation is less efficiency than the Python and C implementation. Indeed, when a function call is made, a copy of all the data passed in the function, is done. In the recursive algorithm, many calls are done and the RAM memory will be saturated "faster" than in the iterative algorithm.

Fig. 17 shows the memory management for a recursive module with data passed by values, with several copies of the almost same data.

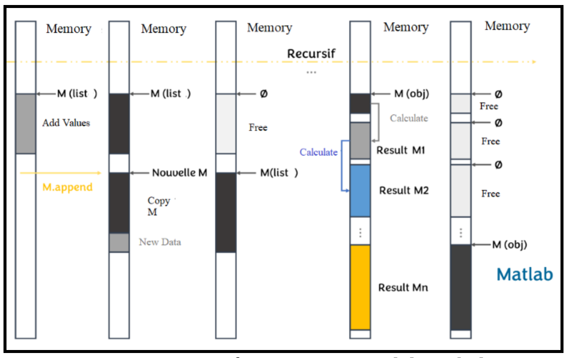


Fig. 17. Memory management for a recursive module with data passed by values.

Even if the python arrays are dynamic arrays, the 'automatic' memory management is less efficient than the student's management in C!

Fig. 18 shows the memory management for a recursive module with data passed by address, with only one copy of the same data.

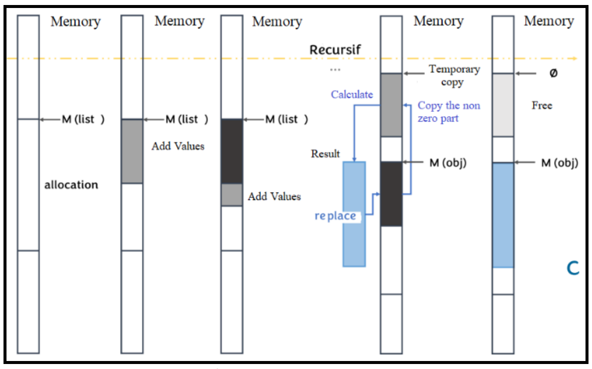


Fig. 18. Memory management for a recursive module with data passed by address.

VIII. Comparison between the Teacher's Algorithm and the Student's Algorithm

To compare the speed of the two algorithms, we must compare the results obtained from the same data. Thus, we compare the execution time of the professor's algorithm with the student's algorithms, which are more efficient.

A. Teacher's Algorithm vs 'Insertion' Algorithm of the Student

Fig. 19 confirms what we thought, that the algorithm named 'Insertion' is much faster than the professor's algorithm. The additional information is the minimum number of cards where there is a large discrepancy between the both algorithms: 27 cards.

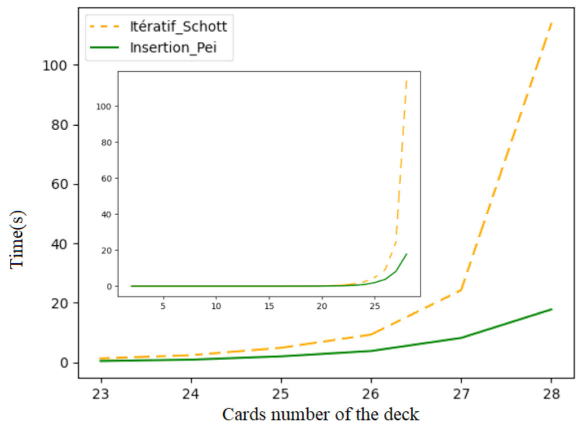


Fig. 19. Comparison between the 'insertion' algorithm of the student and the iterative algorithm of the teacher.

B. Teacher's Algorithm vs 'Combination' Algorithm of the Student

Fig. 20 clearly shows that the student's algorithm is more efficient than the professor's algorithm. By comparing figures 19 and 20 we can deduce that the overall time saving is achieved mainly thanks to the insertion vector!

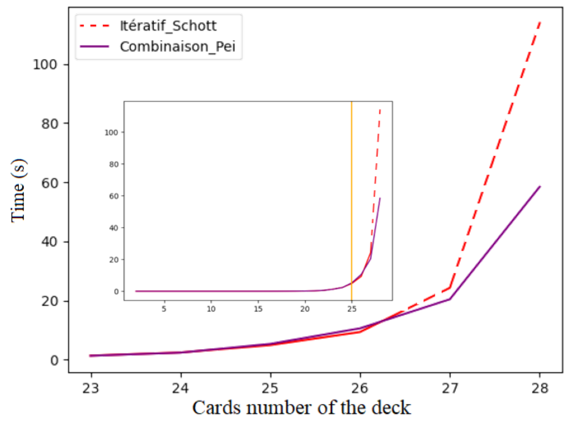


Fig. 20. Comparison between the 'combination' algorithm of the student and the iterative algorithm of the teacher.

IX. Synthesis, Conclusion, Going Further and Prospects

Before asking the question 'is it opportune to develop this type of teaching?', let us make a synthesis of students' work and then a conclusion in relation to the teacher's expectations. After this, we can answer the aforementioned question by suggesting ways to go further.

A. Synthesis and Conclusion

The aim of the paper was not only to show the result of a high education project, but to also to give ideas for a course or using project pedagogy.

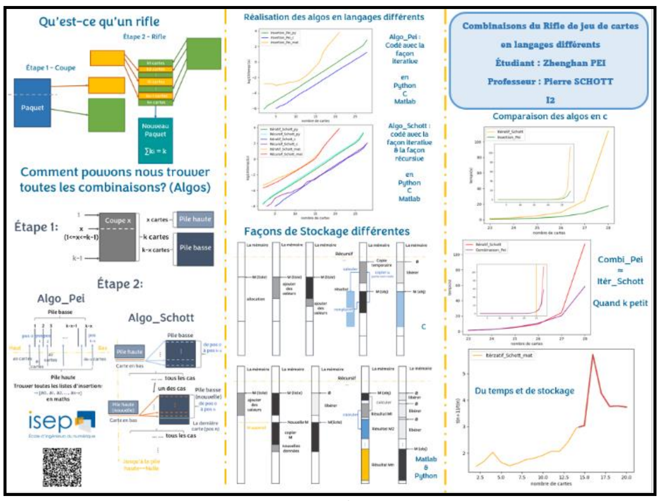
Fifteen students have chosen this project and have had a 40 hour Python course before the project.

A third of them has achieved success for the entire project but following the teacher's algorithm. The remaining two-thirds except one have achieved:

- the program to find all possible cuts,
- the program to shuffle one card in a q card deck,
- the iterative program to find all possible riffle shuffle.

B. Share the Project Results with a Poster and Video

The ISEP communication service asked the student to create a poster (presented in Fig. 21) to present his project during an ISEP open day. This rewards this student's investment and increased motivation.



Qu'est-ce qu'un rifle
Étape 1 - Coupe
Étape 2 - Rifle
Paquet
Nouveau Paquet
Comment pouvons nous trouver toutes les combinaisons? (Algos)
Étape 1:
Étape 2:
Algo_Pei
Algo_Schott
Réalisation des algos en langages différents
Façons de Stockage différentes
Combinaisons du Rifle de jeu de cartes en langages différents
Étudiant : Zhenghan PEI
Professeur : Pierre SCHOTT
I2
Comparaison des algos en c
Combi_Pei vs Itér_Schott
Quand k petit
Du temps et de stockage
isep

Fig. 21. Student's poster for an ISEP open day.

C. Going Further and Prospects

There are many magical shuffles (Faro, Monge, …) which are not in fact aleatory shuffle in the common sense but a simple reorganization of the cards, whose positions are calculated from a deterministic function. An integer number of shuffles allows to find the initial state of the card deck. So we can use this property to introduce students to encryption theory [18].

We can even imagine teaching the major phases of communication between two points using digital electronics where encryption would be done by Faro shuffles

The most important thing for us remains to use the passion for the magic as a teaching vector which also weaves a human bond between the professor and the student.

Conflict of Interest

We declare the submitted work was carried out without a conflict of interest.

Author Contributions

P. Schott conducted the research, created and programmed the teacher's algorithm and wrote the paper. Z. Pei created and programmed the student's algorithm, compared the algorithms. All authors had approved the final version.

References

[1] L. Lesser and M. Glickman, "Using magic in the teaching of probability and statistics," *Model Assisted Statistics and Applications*, December 2009.

[2] S. Martinez-Conde and S. L. Macknik, "Magic and the brain," *Scientific American*, vol. 299, no. 6, pp. 72–79, 2008.

[3] G. Kuhn and K. Land, "There's more to magic than meets the eye," *Current Biology*, vol. 16, pp. 950–951, 2006.

[4] G. Kuhn, A. Amlani, and R. Rensik, "Towards a science of magic," *Trends in Cognitive Science*, vol. 12, no. 9, pp. 349–354, 2008.

[5] R. Edwards, "Algebra magic tricks: Algecadabra! (Vol. 1)", *Critical Pacific Grove*, CA: Thinking Press and Software, 1992

[6] R. Wisemen and C. Watt, "Conjuring cognition: a review of educational magic-based interventions," *PERJ*, March 2020.

[7] D. Aldous and P. Diaconis, "Shuffling cards and stopping times," *Amer. Math' Monthly*, vol.93, no. 5, pp. 333–348, 1986.

[8] A. Lachal and P. Schott, "Cartomagie: principes de Gilbreath (I)," *Quadrature*, no. 85, pp. 24–35, 2012.

[9] J. C. McGinty, "The trick behind properly shuffling cards: Casual players don't typically randomize the deck, but a perfect "riffle" also doesn't work," *Wall Street Journal*, 11 May 2018.

[10] E. Gilbert, "Theory of shuffling," *Technical Memorandum*, Bell Labs, 1955.

[11] Bayer, Dave, and Diaconis, "Trailing the dovetail shuffle to its lair," *Annals of Applied Probability*, vol. 2, no. 2, pp. 294–313, 1992.

[12] B. Mann, "How many times should you shuffle a deck of cards?" *Mathematics*, 1998.

[13] M. Gardner, "Mathematics, magic and mystery," *Dover*, 1958

[14] M. Gardner, "Martin Gardner's mathematical games: The entire collection of his scientific American columns," Mathematical Association of America, 2005.

[15] C. Mulcahy, "Mathematical card magic," *MAA Math Fest*, Cincinnati, OH, August 2019.

[16] L. Lesser and D. Pearl, "Functional fun in statistics teaching: Resources, research, and recommendations," *Journal of statistics education*, vol. 16, no. 3, pp. 1–11, December 2008.

[17] P. Schott, "How to introduce the basis of algorithmics? Thanks to the enumeration and composition of all riffle shuffles from a N card deck used in MathMagic," *Creative Education*, vol. 3, no. 4, pp. 540–556, August 2012.

[18] P. Schott, "Initiation of cryptology thanks to a Scilab™ project using perfect magic shuffles," *Japan Journal of Research*, vol. 4, no. 5, pp. 1–7, September 2023.