Promoting computational thinking by artistically enhanced algorithm visualization

Zoltán KÁTAI[1], Erika OSZTIÁN[2], Géza Károly VEKOV[3]

{ [1]katai_zoltan, [2]osztian, [3]vgeza }@ms.sapientia.ro
SAPIENTIA University

Abstract. In this paper we present a novel approach to promote computational thinking by introducing students with common computer algorithms (searching, sorting, backtracking) illustrated by hip-hop, folk and ballet dance choreographies. We also suggest question-sequences to familiarize them with basic computer science concepts like algorithm complexity and optimization.

Keywords: computational thinking, art-science combination, algorithms, computer science concepts

1. Introduction

Therefore, a major responsibility of educational systems in the 21st century is to prepare the generations to come for the challenges involved with the increasing computerization of our everyday lives and to meet the demands of the one of the fastest–growing job markets: computing [1, 2]. The current paper describes a method which has as its scope the initiation of all students at all levels in computational thinking through algorithmic concepts presented in artistically enhanced context.

The term computational thinking has been the subject of much debate since its introduction in 2006 [3]. In this study we will use the term in accordance with its revised definition, i.e. as the thought process involved in formulating problems so "their solutions can be represented as computational steps and algorithms" [4]. We plan to promote students' computational thinking by inviting them on an inspiring tour of the exciting world of computer algorithms.
Since computer algorithms are deeply abstract entities, teaching-learning them effectively is a challenging task. As Turing [5] stated: "One's object is then to have a clear mental picture of the state of the machine at each moment in the computation. This object can only be achieved with a struggle". In line with this statement, an often used method to illustrate how algorithms work is to represent experts' "clear mental picture" graphically. According to a recent review "while many good algorithm visualizations are available, the need for more and higher quality visualizations continues" [6,7].

Usually, the purpose of algorithm visualizations is to promote students' understanding regarding the procedural behavior of algorithms. A recent survey comparing animation with static pictures confirmed the superiority of instructional animations in particular when procedural-motor knowledge had to be transmitted [8]. Over the years several methods were used to visually illustrate computer algorithms in action: videos, computer-based animations, etc [6,7]. Beside these technology-based approaches, animations performed by teachers or students are also used to make algorithms more tangible [9,10,11]. In [12] we presented how inviting students to play recursive scenarios helps them to imagine how recursion works.

Dance is a complex human activity involving the entire person: physically, cognitively and affectively. Dance choreographies include movements and structure, and the accompanying music is also characterized by repetitive rhythmic patterns. These patterns and structures are the key elements for bridging between sciences and arts. In [13] we introduced the notion of danced algorithm. In [14] we replaced amateur dancers with professional ones. In this paper we extend our collection of algorithmic folk dance choreographies (bubble sort, insertion sort, selection sort, merge sort, quick sort, shell sort) with other two algorithm categories: searching and backtracking. We also introduce two new dance styles: hip-hop and ballet. We present a gradually increasing syllabus for introducing students in the exciting world of computer algorithms (searching, sorting, backtracking), and familiarizing them with basic computer science concepts like algorithm complexity, optimization, parallel algorithms, etc.

2. Art and science: a winning combination

According to Palmer [15] motivation is a 'necessary prerequisite and co-requisite for learning'. Research results in this field emphasize the critical importance intrinsic motivation has in promoting effective learning [16,17]. Providing novelty, incongruity and surprise are effective ways to promote intrinsic motivation since it has potential to arouse curiosity and combat apathy [18,19]. We have proposed to gain such-like impact by combining arts (dances) with science (computer algorithms), traditional/classic (folk-dance/ballet) with modern (IT).

Additionally, science-art combinations could be effective because of the parallel involvement of both side of brain. Whereas algorithmics is more closely associated with the left side of the brain, artistic expressions are more active in the right hemisphere. Research in this field revealed that efficient teaching-learning assumes a balanced involvement of both sides of the brain [20]. Gardner's [21,22] work emphasizes that a mixture of intelligences (musical intelligence, bodily-kinesthetic intelligence, logical-mathematical intelligence, etc.) characterizes all people. He argues that students need to learn in various ways and teachers should not allow them to rely only on their most comfortable intelligence. In line with these findings, since 1998 mathematicians, artists, musicians and scientists have been coming together at the annual Bridges Conferences to discuss possible art-science connections [23].
Finally, our previous experience with the folk danced sorting algorithms confirmed that the presence of arts –beyond the cognitive benefits– gives the class a touch of liveliness.

2.1. Searching algorithms visualized by hip-hop dance choreographies

Computational thinking involves pattern recognition, generalization and abstractions [24]. Since humans are efficient pattern recognizers, given a few examples we are able to establish whether a new object belongs to the same class or not (generalization, classification) [25]. Evidently, effective learning from examples assumes that these are selected carefully [26].

Since searching algorithms are probably the most directly perceived and intuitive computer algorithms by IT users, we propose this category to be analyzed firstly. Search algorithms can be classified based on their mechanism of searching. Linear search algorithms check, successively, every element of a list (often stored in a one-dimensional array) for the one that equals the value we are looking for (target key). Binary searches repeatedly target the middle element of an ordered list reducing the search space in half after each step.
We have chosen to illustrate the linear and binary searching strategies by hip-hop dance choreographies. Hip hop is considered by many a worldwide cultural communication

style including music, art and dance. Especially youth and hip hop are tied up together. The list to be searched could be represented by a hip-hop dancer-sequence, and the target key by an extra dancer (the protagonist of the performance). Since each dancer wears the corresponding value on his/her back, these are not visible in the case of list-elements, but visible on the back of the target key (see Figure 1).

In the case of linear searching the result of the comparison operation between the target key and the current element of the searched list is yes (equal) or no (not equal). These yes/no **comparisons** could be illustrated by pieces of choreographies where the two dancers are dancing identically/differently (see Figure 1). Eliminated elements (not equal) from the searching space are falling down.

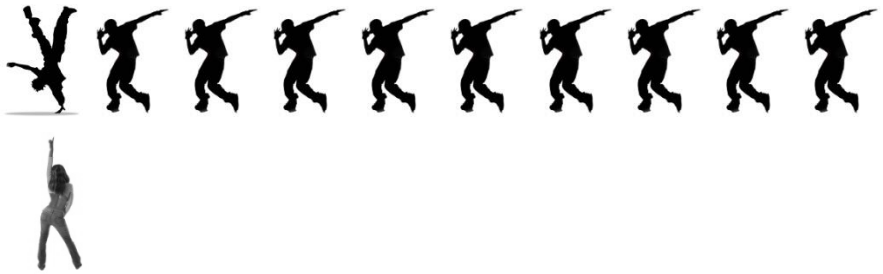

Figure 1. The target key (representing number 59) and the first element of the list to be searched are dancing a "not-equal hip-hop choreography".

Figure 2 presents the whole linear searching choreography for the following scenario: we are looking for number 59 in the list of values 19, 80, 30, 2, 89, 59, 71, 66, 7, 34.

In the case of binary searching comparison operations results in one of the following three possibilities: less/greater/equal. To illustrate these variants the corresponding dancer-pair could follow the same piece of choreography, but in a slower/faster/in-synchrony rhythm. Figure 3 shows the binary searching choreography we proposed to implement for the following scenario: we are looking for number 59 in the list of values 2, 7, 19, 30, 34, 59, 66, 71, 80, 89.

2.2 Sorting algorithms visualized by folk-dance choreographies

As a next step we suggest sorting algorithms to be studied. Sorting a list is a common operation in many fields of work and is one of the most fundamental problems in computer science. In 2012 we posted six folk dance choreographies on YouTube illustrating different sorting strategies (https://www.youtube.com/user/AlgoRythmics/videos). We selected for our current the syllabus four of them: 1) selection-sort with Gipsy folkdance, 2) bubble-sort with "Csángó folkdance", 3) insertion-sort with Romanian folkdance, 4) merge-sort with Transylvanian Saxon folkdance.

2.3 Backtracking algorithm visualized by ballet choreography

Sudoku is one of the world's most popular brain games. What kind of computer algorithms is mostly related to this number game? Backtracking! Backtracking is a programming strategy for finding all solutions to a given computational problem that incrementally builds solutions, and immediately abandons all those partial solutions that evidently cannot be completed to a valid final solution. The classic backtracking example is the so-called "eight queens puzzle", that asks

for all valid arrangements of eight chess queens on a chessboard (no queen attacks any other). We plan to illustrate the recursive version of the four-queen variant of this classic backtracking algorithm by classic ballet choreography. Figure 5 presents the order and the rows where the 15 ballerinas (corresponding to the 15 calls of the backtracking recursive function) are going to dance. The following pieces of choreographies are attached to each ballerina:

- The queen comes to life at cell 0 of her row (new recursive call)
- The queen dies at cell 5 of her row (the current recursive call ends)
- The queen goes into "hibernation mode" (the current call is suspended; a new one begins at the next row)
- The queen comes back from the "hibernation mode" (the suspended call from the previous row continues)
- The current queen moves to the next cell of her row
- The current queen successively considers those queens that hibernates at previous rows
- These ones temporally wake up for a "pair of mutually attacking" / "pair of mutually non-attacking" dance

Additionally, four-queen "victory dances" illustrate the two valid solutions (see Figure 6). Figure 7 shows our backtracking ballet choreography draft .

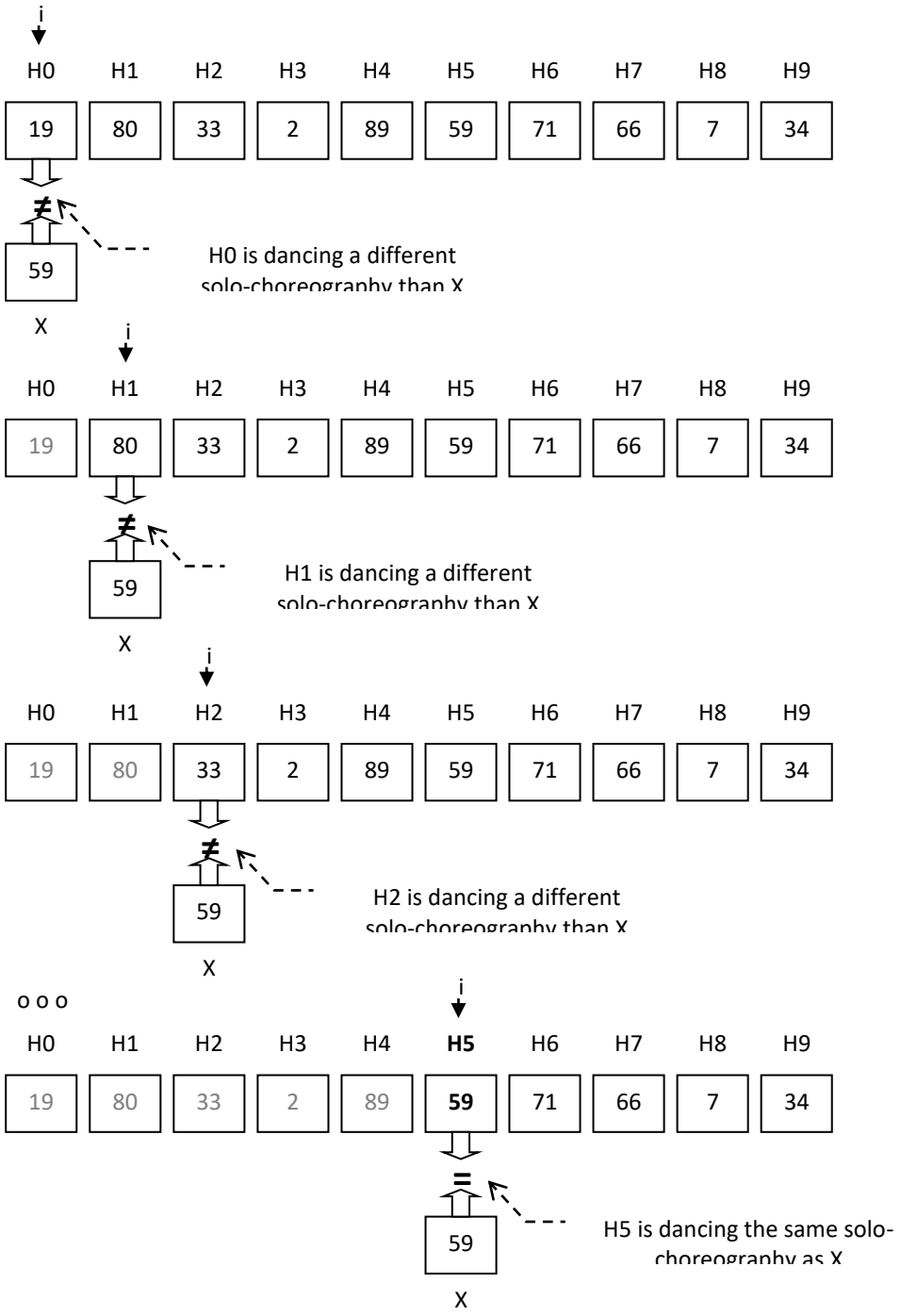

Figure 2. Linear searching choreography: we are looking for the value of X in the number sequence stored in array H[0..9].

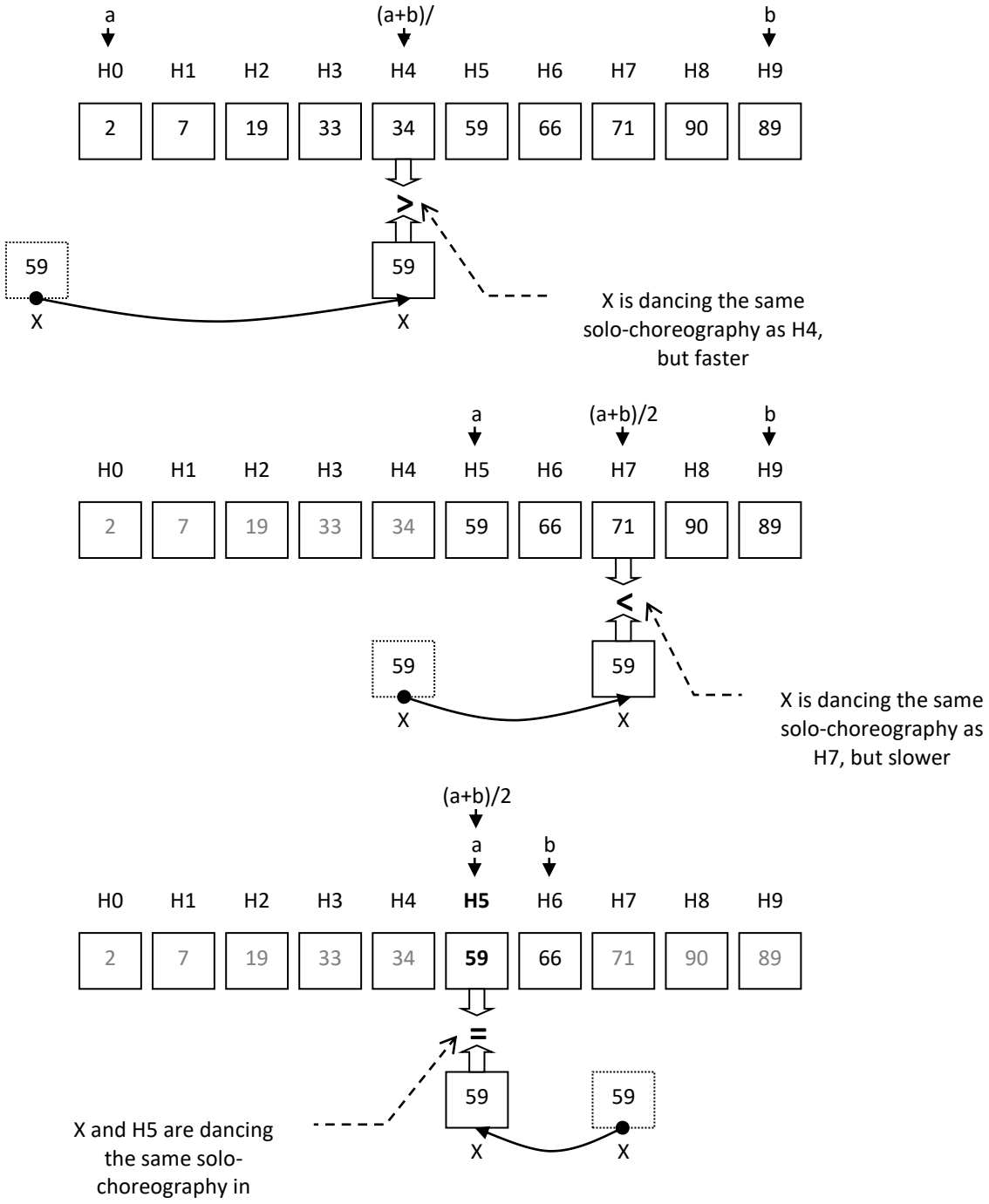

Figure 3. Binary searching choreography: we are looking for the value of X in the ordered number

sequence stored in array H[0..9].

Figure 4.a. Selection sort with Gipsy folkdance

Figure 4.b. Bubble sort with “Csángó” folkdance

Figure 4.c. Insertion sort with Romanian folkdance.

Figure 4.d. Merge sort with Transylvanian Saxon folkdance.

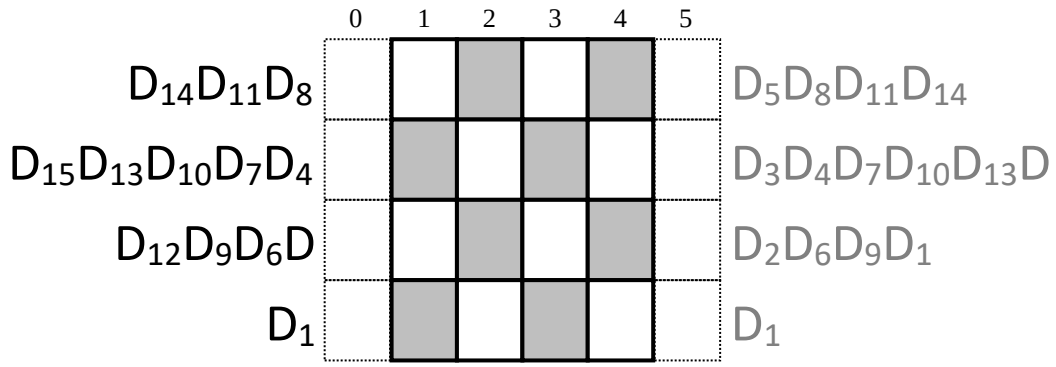

Figure 5. The order and the rows where the 15 ballerinas (representing the queens) are going to dance (traversing the chessboard from left to right).

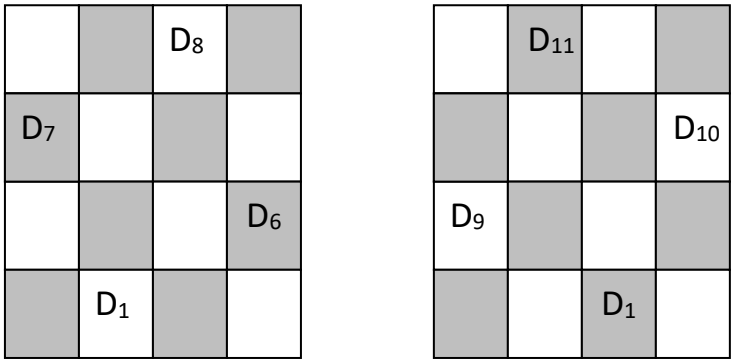

Figure 6. The two valid solutions.

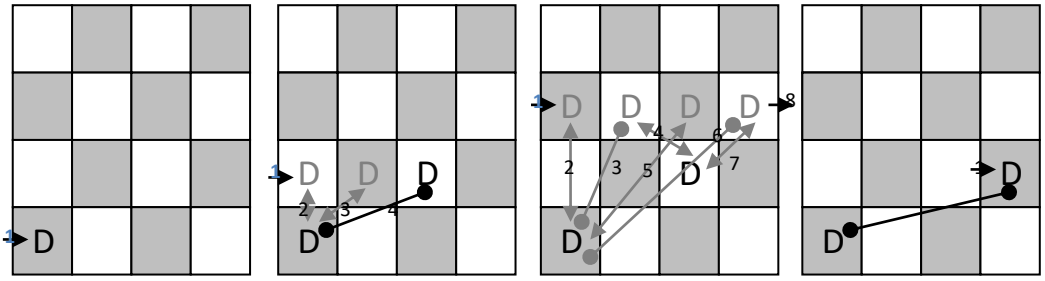

Figure 7. Backtracking ballet choreography (first four scenes). The numbers represent consecutive piece of choreographies for the current ballerina: double arrow with stealth heads – mutually attacking dance, double arrow with circle heads – mutually non-attacking dance.

3. Familiarizing students with basic computer science concepts

Based on the above presented dance choreographies we propose the following questions to be analyzed for introducing students with some basic computer science concepts.

Algorithm complexity concepts

Searching algorithms

- How many comparison operations does the linear search algorithm imply for the sample presented in figure 2?
- How many comparison operations does the binary search algorithm imply for the sample presented in figure 3?
- What is the "best case" with respect to the linear search algorithm?
- What is the "worst case" with respect to the linear search algorithm?
- What is the "best case" with respect to the binary search algorithm?
- What is the "worst case" with respect to the binary search algorithm?
- How many comparison operations does the linear search algorithm imply (for a list with n elements) in the "best case"?
- How many comparison operations does the linear search algorithm imply (for a list with n elements) in the "worst case"?
- The (worst case) time complexity of the linear search algorithm is $O(n)$. Why?
- How many comparison operations does the binary search algorithm imply (for a list with n elements) in the "best case"?
- How many comparison operations does the binary search algorithm imply (for a list with n elements) in the "worst case"?
- The (worst case) time complexity of the binary search algorithm is $O(\log n)$. Why?

Sorting algorithms

- How many comparison and swapping operations does the selection sort algorithm imply for the sample presented in the video from figure 4.a?
- How many comparison and swapping operations does the bubble sort algorithm imply for the sample presented in the video from figure 4.b?
- How many comparison and swapping operations does the insertion sort algorithm imply for the sample presented in the video from figure 4.c?
- How many comparison operations does the merge sort algorithm imply for the sample presented in the video from figure 4.d?
- What is the "best case" with respect to a sorting algorithm?
- What is the "worst case" with respect to a sorting algorithm?
- How many comparison and swapping operations does the selection sort algorithm imply (for a list with n elements) in the "best case"?
- How many comparison and swapping operations does the selection sort algorithm imply (for a list with n elements) in the "worst case"?
- The (best case) time complexity of the selection sort algorithm is $O(n^2)$. Why?
- The (worst case) time complexity of the selection sort algorithm is $O(n^2)$. Why?
- How many comparison and swapping operations does the bubble sort algorithm imply (for a list with n elements) in the "best case"?

- How many comparison and swapping operations does the bubble sort algorithm imply (for a list with n elements) in the "worst case"?
- The (best case) time complexity of the bubble sort algorithm is $O(n)$. Why?
- The (worst case) time complexity of the bubble sort algorithm is $O(n^2)$. Why?
- How many comparison and swapping operations does the insertion sort algorithm imply (for a list with n elements) in the "best case"?
- How many comparison and swapping operations does the insertion sort algorithm imply (for a list with n elements) in the "worst case"?
- The (best case) time complexity of the insertion sort algorithm is $O(n)$. Why?
- The (worst case) time complexity of the insertion sort algorithm is $O(n^2)$. Why?
- How many comparison operations does the merge sort algorithm imply (for a list with n elements) in the "best case"?
- How many comparison operations does the merge sort algorithm imply (for a list with n elements) in the "worst case"?
- In what sense is the merge sort algorithm a divide and conquer strategy?
- The (best case) time complexity of the merge sort algorithm is $O(n \log n)$. Why?
- The (worst case) time complexity of the merge sort algorithm is $O(n \log n)$. Why?
- What is a limitation (regarding space complexity) of the merge sort algorithm?

Backtracking algorithms

- How many possibilities are to arrange 4 pawns on a 4x4 chessboard if the only restriction is: no pawns in the same row? How many pawn-ballerinas are needed to a four-pawn backtracking choreography?
- How many possibilities are to arrange 4 rooks on a 4x4 chessboard in such a way that no rook-pairs attacking each other (no rook-pairs in the same row or column)? How many rook-ballerinas are needed to a four-rook backtracking choreography?
- How many possibilities are to arrange 4 queens on a 4x4 chessboard in such a way that no queen-pairs attacking each other (no queen-pairs in the same row, column or diagonal)? How many queen-ballerinas are needed to a four-queen backtracking choreography?
- How many possibilities are to arrange n pawns on an nxn chessboard if the only restriction is: no pawns in the same row?
- How many possibilities are to arrange n rooks on an nxn chessboard in such a way that no rook-pairs attacking each other?
- How many possibilities are to arrange n queens on an nxn chessboard in such a way that no queen-pairs attacking each other?
- Usually, backtracking algorithms have exponential time complexity. Why?

Algorithm optimization

- What is the maximum number of elements that can *certainly* be eliminated from the searching space with one comparison operation in the case of the linear search algorithm?
- What is the maximum number of elements that can *certainly* be eliminated from the searching space with one comparison operation in the case of the binary search algorithm?
- Why is binary searching superior to linear searching? What is a limitation of the binary search algorithm?

- After how many searching operations is the "merge-sort + binary-search" recipe superior to linear searching?
- How many elements reach, *certainly*, their final positions (in the ordered list) after each traverse the bubble sort algorithm performs?
- Why did two dancers turn back after the first traversed in the bubble sort implementation presented in the video from figure 4.b?
- Why did only one dancer turn back after the second traversed in the bubble sort implementation presented in the video from figure 4.b?
- In what way could the insertion operation be optimized in the insertion sort implementation presented in the video from figure 4.c?
- What parallelization possibility is suggested by the merge sort choreography presented in the video from figure 4.d?
- Although implementing the n-pawns backtracking algorithm is easier than implementing the n-queens one, why is "the valid n-queens arrangements are such valid n-pawns arrangements where no pawns are in the same column or diagonal" approach inefficient?

4. Conclusions

In this paper we have presented a novel approach to promote computational thinking by introducing students with common computer algorithms illustrated by hip-hop, folk and ballet dance choreographies. We have also suggested question-sequences to familiarize them with basic computer science concepts like algorithm complexity and optimization.

5. References

[1] S. Grover and R. Pea, "Computational thinking in K–12. A review of the state of the field", Educational Researcher, vol. 42, no. 1, pp. 38–43, 2013.

[2] "US Bureau of Labor Statistics." Retrieved from http://www.bls.gov/ooh/.

[3] J. Wing, "Computational thinking," Communications of the ACM, vol. 49, no. 3, pp. 33–35,2006.

[4] A. V. Aho, "Computation and computational thinking," The Computer Journal, vol. 55, no. 7,pp. 832–835, 2012.

[5] A. M. Turing. Computing machinery and intelligence. Mind, 59(236):433–460, 1950.

[6] C. Shaffer, M. Cooper, and S. Edwards. Algorithm visualization: a report on the state of the field. In SIGCSE '07 Conference Proceedings, pages 150–154. ACM Press, New York, 2007.

[7] C. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, S. H. Edwards. Algorithm visualization: the state of the field. ACM Transactions on Computing Education, 10(3):1–22, August 2010.

[8] T. N. Hoffler and D. Leutner. Instructional animation versus static pictures: a meta-analysis. Learning and Instruction, 17(6):722–738, December 2007.

[9] Bell, T., Witten, I. H., & Fellows, M. (2005). Computer science unplugged: An enrichment and extension programme for primary-aged children. Canterbury, New Zealand: Computer Science Unplugged. Retrieved from http://csunplugged.com

[10] Futschek, G. (2007). Logo-like learning of basic concepts of algorithms – having fun with algorithms. In I. Kalas (Ed.), Proceedings of the 11th European Logo Conference (p. 51). Bratislava: Comenius University. Retrieved from http://publik.tuwien.ac.at/files/pub-inf_4696.pdf

[11] Futschek, G., & Moschitz, J. (2010) Developing algorithmic thinking by inventing and playing algorithms. Proceedings of Constructionist Approaches to Creative Learning Thinking and Education, pp. 1–10. Retrieved from http://publik.tuwien.ac.at/files/PubDat_187461.pdf

[12] Katai, Z. (2011). Multi-sensory method for teaching-learning recursion. Computer Applications in Engineering Education, 19(2), 234–243.

[13] Katai, Z., & Toth, L. (2010). Technologically and artistically enhanced multi-sensory computer-programming education. Teaching and Teacher Education, 26, 244–251.

[14] Katai, Z. (2014). Intercultural computer science education. In: Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education. ACM New York, NY, USA, 183–188.

[15] Palmer, D. (2005). A motivational view of constructivist informed teaching. International Journal of Science Education, 27(15), 1853–1881.

[16] Fair, E. M., & Silvestri, L. (1992). Effects of rewards, competition, and outcome on intrinsic motivation. Journal of Instructional Psychology, 19, 3–8.

[17] Martens, R. L., Gulikersw, J., & Bastiaensw, T. (2004). The impact of intrinsic motivation on e-learning in authentic computer tasks. Journal of Computer Assisted learning, 20, 368–376.

[18] Berlyne, D. E. (1960). Conflict, arousal, and curiosity. New York: McGraw Hill Book Company Inc.

[19] Keller, J. M. (1983). Motivational design of instruction. In C. M. Reigeluth (Ed.), Instructional-design theories and models (pp. 383–434). Hillsdale, NJ: Erlbaum.

[20] Eisenhower SCIMAST. (1997). How can research on the brain inform education. Classroom Compass, 3(2). 1–2, 10. Retrieved from. http://www.sedl.org/pubs/classroom-compass/cc_v3n2.pdf.

[21] Gardner, H. (1993). Frames of mind (The tenth anniversary ed.). New York: Basic Books.

[22] Gardner, H. (2000). Intelligence reframed. Multiple intelligences for the 21st century. New York: Basic Books.

[23] Bridges Organization. (2004). Bridges: Mathematical connections in art, music, and science. Retrieved from The Bridges Organization: Arts and Mathematics Web Site: http://www.bridgesmathart.org/past-conferences/2004-2.

[24] Wu,M. L., & Richards, K. (2011). Facilitating computational thinking through game design. In M. Chang,W.Y. Hwang, M. P. Chen, & W. Müller (Eds.), Proceedings of Edutainment Technologies. Educational games and virtual reality/augmented reality applications (pp. 220–227). Heidelberg: Springer Berlin.

[25] Jain, A., Duin, R., & Mao, J. (2000). Statistical pattern recognition: Areview. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22, 4–37.

[26] Jain, A. K., & Duin, R. P. W. (2004). Pattern recognition. In R. L. Gregory (Ed.), The Oxford companion to the mind (2nd ed., pp. 698–703). Oxford, UK: Oxford University Press.