Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 178 (2007) 3–13

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

An Evaluation of the Effortless Approach to Build Algorithm Animations with WinHIPE

Jaime Urquiza-Fuentes[1]

Department of Languages and Systems, Rey Juan Carlos University, Móstoles, Spain

J. Ángel Velázquez-Iturbide[2]

Department of Languages and Systems, Rey Juan Carlos University, Móstoles, Spain

Abstract

The use of algorithm visualizations in computer science education is not a new thing. Although there is a firm belief that graphical representations of algorithms are learning aids, empirical studies show that what is important is what the students do with the animations rather than what they see in them. In this paper we compare to kinds of interaction: viewing animations vs constructing animations. We have conducted a controlled experiment where a group of students (n=15) had to study an algorithm and complete a knowledge test about it and a subjective opinion questionnaire. Students were randomly divided in constructing and viewing groups. Results have been measured by means of learning outcomes, efficiency issues and student's subjective opinion. Results significantly evidence that builders obtained better results than viewers.

Keywords: Algorithm visualization, engagement levels, pedagogical evaluation.

1 Introduction

Algorithm visualizations (AV) are used in computer science education since the early eighties [1]. There are various surveys on using visualization as an aid for computer science education [3,4,6,7,8]. In spite of their educational potential, they have not been incorporated into the mainstream of computer science education. This lack of use has two main reasons: from the instructors' point of view, animations are not usually easy to use, deploy and adapt to the course [6,5]; from the students' point of view, more interaction, than just viewing animations, is needed to obtain learning improvements [3,4,7]. In this paper, we will focus on the interaction between students and animations.

[1] Email: jaime.urquiza@urjc.es

[2] Email: angel.velazquez@urjc.es

doi:10.1016/j.entcs.2007.01.038

1.1 The taxonomy of engagement levels

Naps et al. [7] defined a taxonomy of six engagement levels for the different ways of interaction between the students and the animations: no viewing, viewing, responding, changing, constructing and presenting. Quoting to Naps et al. [7]:

> *(...) the first category is "No viewing" , which indicates that no visualization technology is used at all.*
>
> *(...) "Viewing" can be considered the core form of engagement, (...) a learner can view an animation passively, but can also exercise control over the direction and pace of the animation, use different windows (each presenting a different view), or use accompanying textual or aural explanations. (...) The remaining four categories all include viewing.*
>
> *(...) "Responding". The key activity in this category is answering questions concerning the visualization presented by the system. (...) In the responding form of engagement, the learner uses the visualization as a resource for answering questions.*
>
> *(...) "Changing", entails modifying the visualization. The most common example of such modification is allowing the learner to change the input of the algorithm under study in order to explore the algorithms behavior in different cases.*
>
> *(...) "Constructing". In this form of engagement, learners construct their own visualizations of the algorithms under study. Hundhausen and Douglas [27] have identified two main ways in which learners may construct visualizations: direct generation and hand construction. (...) It is important to note that the Constructing form of engagement does not necessarily entail coding the algorithm.*
>
> *(...) "Presenting", entails presenting a visualization to an audience for feedback and discussion.*

The pedagogical effectiveness of these engagement levels in AV has been analyzed [3,4,7]. The general conclusion is: the higher the engagement of students with AV technology, the better the learning outcomes. For instance Grissom et al. [3] found learning improvements, at the understanding level of Bloom's taxonomy [2], with AV extended with stop-and-think questions, the responding engagement level. We have found in the literature neither studies about AV technologies improving learning further than the understanding level of Bloom's taxonomy, nor studies about engagement levels in AV further than responding.

1.2 Our approach

We have developed an effortless approach to build and maintain algorithm/program animations [9]. Thus we have extended the WinHIPE IDE (see Fig. 1) with visualization facilities [11] to produce web-based algorithm/program animations (see Fig. 2). These animations consist of four components: the animation itself (a sequence of visualizations), the source code, the description of the algorithm implemented, and the description of the problem solved by the algorithm.

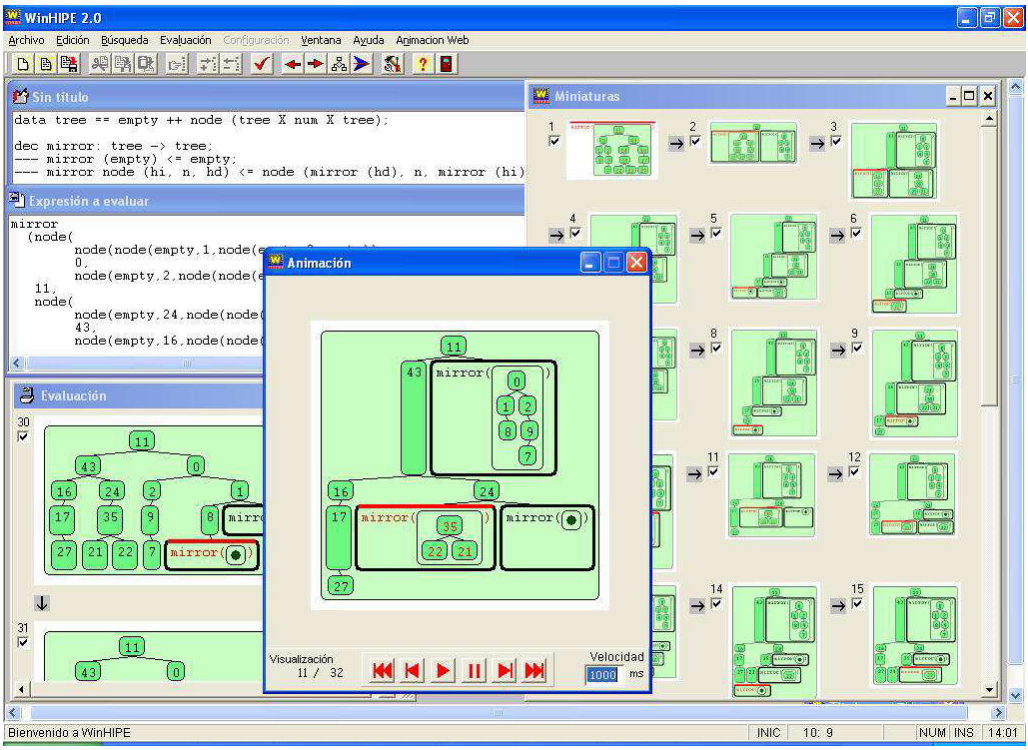


Fig. 1. A snapshot of the WinHIPE environment: with typical windows of a programming environment — source code, expression, and evaluation —, the miniatures window to build the animation, and the animation window to play it

The main aim is to minimize the work needed to produce the animation. Animations are built from a set of static visualizations representing the execution stages of a program, which are automatically generated. Apart from the typical edition-compilation-execution process, the user will have to select which visualizations will form part of the animation, and type the problem and algorithm descriptions. We have designed an information visualization technique called R-Zoom [10] that helps on the task of selecting the visualizations, and typing text is not a complex task. Thus, we have an animation generation process very similar to the edition-compilation-execution process of a program, where the additional tasks are not complex; this is why we call it an effortless approach.

To ways of algorithm construction are considered in the definition of the constructing engagement level by Naps et al. [7]: *direct generation* and *hand construction*. Our approach has some of both ways. First, visualizations of the different execution stages are automatically generated by the environment (direct generation), but finally, the student has to select which static visualizations will form part of the final algorithm animation (hand construction).

Following the framework provided by the engagement levels, we have conducted a controlled evaluation to test if our effortless approach of building AV (construction engagement level) improves learning. In our evaluation, we have compared the viewing engagement level against the constructing engagement level. Learning im-

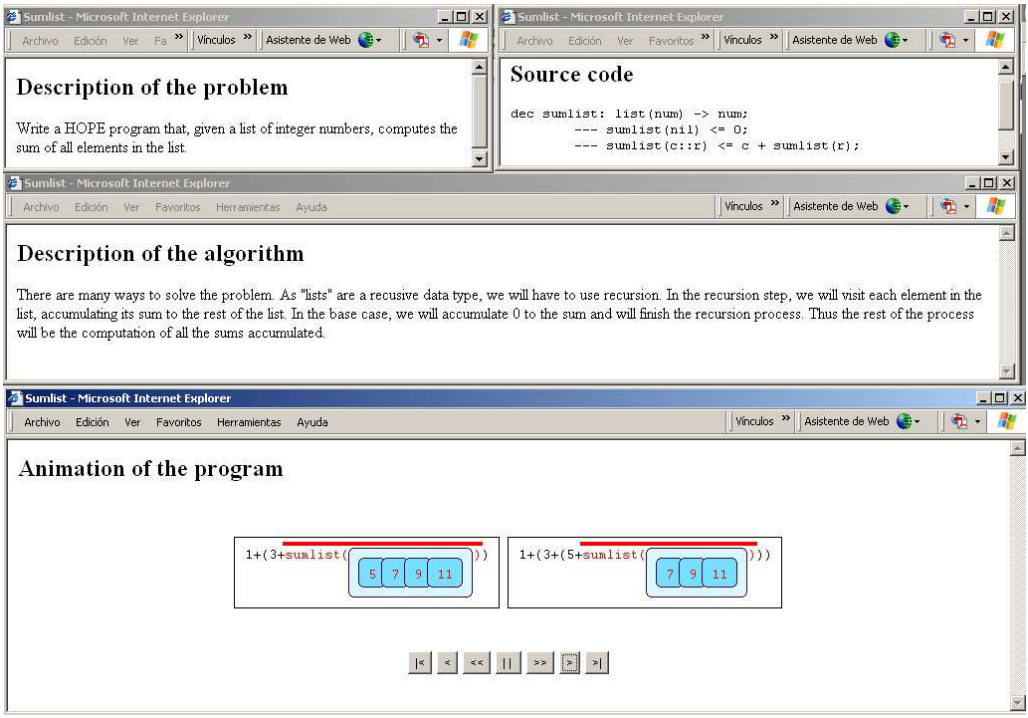


Fig. 2. An example of a web-based animation generated with WinHIPE, each part of the animation — the description of the problem, the description of the solution, the source code, and the animation itself — is shown in a separate window

provement has been measured in terms of the comprehension and application levels of Bloom's taxonomy. We have completed this evaluation with measurements of efficiency and students' satisfaction.

The rest of the paper is structured as follows. Section two describes the evaluation: participants, variables studied, method and procedure. Then results of the evaluation are shown in section three, and discussed in section four. Finally, in section five, conclusions and future work are described.

2 Description of the evaluation

This evaluation attempts to find if there is any performance difference between viewing algorithm visualizations (viewing engagement level) and constructing algorithm visualizations (constructing engagement level). Improvements will be measured in terms of Bloom's taxonomy, with tasks related to the comprehension and application levels.

The context of this evaluation is an Algorithm Design and Analysis course at the Rey Juan Carlos University, where a group of students just had to view algorithm animations, while the other group had to build them using our effortless approach. In the evaluation, the tree breadth traversal algorithm was used.

2.1 Participants

Fifteen different subjects participated in the evaluation, thirteen were male and two female. Participation in the evaluation was voluntary. All of the subjects were students from the Algorithms course.

Participants were randomly divided in two groups: the viewers group and the builders group (VG and BG respectively for the rest of the paper). Both groups were asked about their previous knowledge about the algorithm, only one student having previous knowledge. Therefore both groups belong to the same population and further results can be compared.

2.2 Variables

The independent variables of the evaluation were: pedagogical effectiveness and efficiency, and users' opinion about both approaches (viewing and building). The dependent variables were: the answers to a number of questions about the algorithm (see Appendix A for a copy of this questionnaire), the time expended in studying the algorithm, the time used to complete a knowledge test about the algorithm, and the user's subjective opinion.

The questions about the algorithm were mapped to the comprehension and application levels of Bloom's taxonomy. Learning improvements related to the comprehension level were measured with the following questions:

- What are the main ideas of this algorithm?
- Given the following tree, write the result of applying the algorithm to it.
- Given the following list, write the tree to which the algorithm was originally applied (note that the tree was balanced).
- Which is the existing relationship among the nodes of the tree, if the result is a strictly ascending ordered list?

Learning improvements related to the application level were measured with the following question: What modifications should be done on the algorithm to change the traverse direction to right-to-left?

Users' opinion was measured with a questionnaire where students were asked:

- if they thought that *building* (or *viewing*) algorithm animations had helped them in understanding the algorithm, and
- if they though that algorithm animations are easy to *build* (or *use*) with WinHIPE.

2.3 Method and procedure

The evaluation was divided into two sessions: a training session where the IDE was shown to the students, and the experimental session where knowledge about the algorithm was evaluated. Participation was ten and thirteen students respectively.

The training session was two hours long. The instructor demonstrated the tool,

he generated two web-based animations with WinHIPE as an example, and students generated two more animations. The animations used were unrelated to the algorithm that would be used in the experimental session. None of the students appeared to have problems using the tool. At the end of this session a questionnaire about the tool was completed by the students thus, we got their first impression about the tool.

The experimental session also was two hours long, and two weeks after the training session. First, we explained to the students that we were carrying out the evaluation, and that their participation would be voluntary. Next, we randomly formed the VG (n=7) and BG (n=6), and we checked that all students in the BG had attended the previous training session. Students of both groups were asked about their previous knowledge about the algorithm. Then, we gave the students all the materials they were allowed to use to study the algorithm, which was a textual description of the algorithm for both groups, and:

- a number of web-based algorithm animations to be viewed, built with WinHIPE, for the VG, and
- the source code of the algorithm, to build web-based algorithm animations, for the BG.

Students of both groups were asked to study the algorithm using the materials, until they thought that they had enough knowledge about it. Then, they completed the knowledge test and another questionnaire to collect their subjective opinion about their viewing/building learning experience.

3 Results of the evaluation

3.1 Pedagogical effectiveness

Learning effectiveness was tested by means of two levels of Bloom's taxonomy: comprehension and application. Both levels were graded in the range $[0.0, 1.0]$. See Fig. 3 for a summary of all questions.

Four questions were asked to test performance related to the *comprehension level*. The first question asked students to identify the main ideas underlying this algorithm; these ideas were: (1) operations with lists, (2) left-to-right direction in tree traversing, and (3) accumulation of recursive operations with subtrees. Students of both groups performed the same identifying ideas (1) and (2), while idea (3) was only identified by 14% of students (1/7) in the VG, but by 83% of students (5/6) in the BG ($U = 7.000, p < .05$). We did not find significant differences in performance in the second, third and fourth questions. Thus, the average grades for the understanding level were 0.88 for the BG and 0.73 for the VG, a 16% of learning improvement.

We found significant differences ($U = 7.000, p < .05$) in the answers to the question related to the *application level*. The VG obtained an average grade of 0.33, while the BG obtained an average grade of 0.77, a 60% of learning improvement.

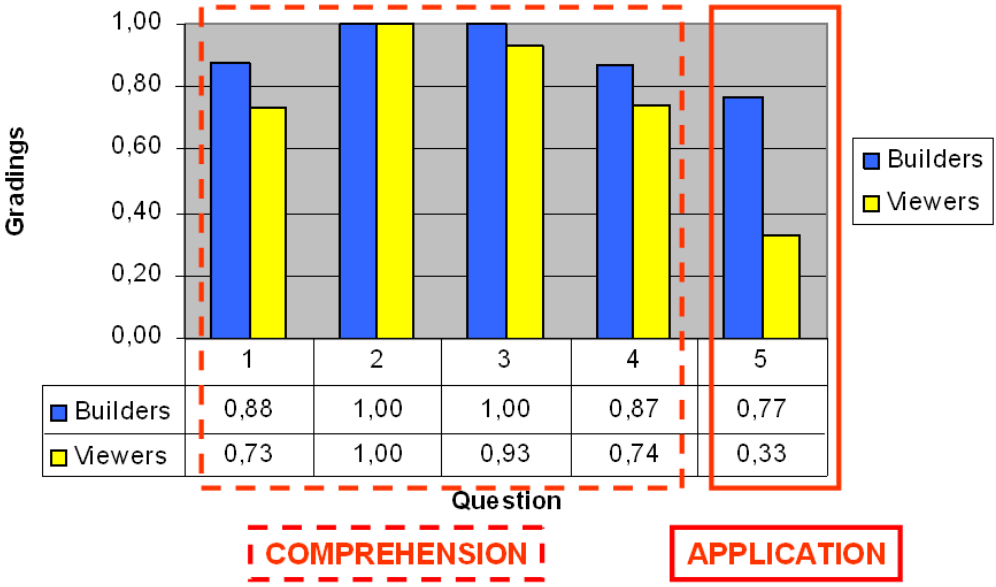


Fig. 3. Gradings to the knowledge questions. Dashed red square highlights the questions 1 to 4, that are mapped to the comprehension level of the Bloom's Taxonomy. Solid red square highlights the question 5, that is mapped to the application level of the Bloom's Taxonomy. Gradings in the comprehension level were quite similar, while those in the application level were significantly different

3.2 Time used by students

We also measured the time expended with the materials and the time used to complete the knowledge test.

With respect to the time expended by students using the materials to study the algorithm, students of the BG expended an average time of 49 minutes ($M = 49.0, SD = 4.97$), while students expended an average time of 18 minutes($M = 18.0, SD = 5.41$). Difference of this time between groups was significant, $t(11) = 10.670, p < .05$. But no differences were found in the time used to complete the knowledge test between the BG ($M = 19.6, SD = 5.12$) and VG ($M = 15.4, SD = 5.79$), $t(11) = 1.384, p > .05$.

3.3 Students' subjective opinion

The students' first impression (after the training session) about WinHIPE and the building process was very good. None had previously used WinHIPE. All of them ($n = 10$) thought that the web-based animations were easy to build, and that building animations would help them in understanding the algorithms.

This opinion was maintained by students after the experimental session. Answers to the questionnaire about users' satisfaction showed that students in the BG agreed with both ideas: building algorithm animations helped them understanding the algorithm, and web-based animations were easy to build with WinHIPE. All of the students in the VG agreed with: web-based animations helped them in understanding the algorithm, and web-based animations were useful and easy to use. We also asked these students about what approach would be more helpful in

learning algorithm concepts: viewing or building. 71% (5/7) thought that both approaches were equally helpful: two of them just said this, but three also said that both approaches should be used together.

4 Discussion

Two aspects may be at the origin of the learning outcomes detected: the engagement level and the time used to study. We have not data to differentiate the importance of each aspect, but a natural explanation comes to us: our building approach has engaged students more than the viewing approach, therefore builders dedicated more time to study the algorithm, which probably, has been a key factor of their learning outcomes.

But the difference between constructing and viewing task should not be ignored. Constructing an animation entails the students deciding if a snapshot is relevant enough to be in the animation, this requires a deeper knowledge of the algorithm. Viewing an animation just ask the students to control the pace and direction of the animations, leaving them the decision of studying in depth what has happened between two snapshots. Obviously, the construction task takes more time than the viewing task, but the deeper is the knowledge about the algorithm, the better are the learning outcomes.

We think that the engagement level has, both directly and indirectly, affected the learning outcomes (see Fig. 4). It has affected in a direct way because the constructing task requires a deeper knowledge of the algorithm. And it has affected in an indirect way because the constructing task requires more time than the viewing task, so students are kept working with the algorithm more time, facilitating them to obtain a deeper understanding of the algorithm.

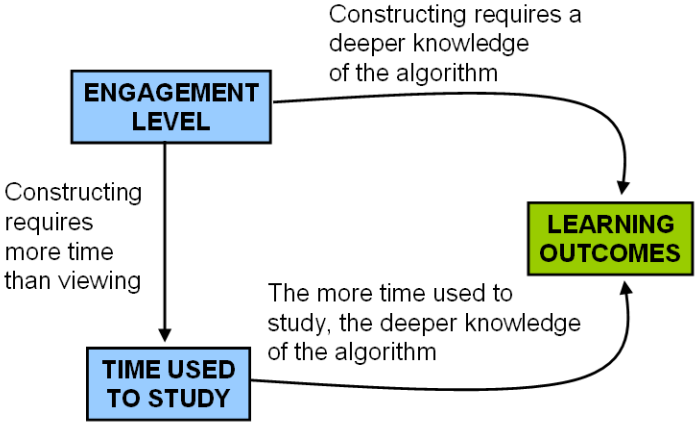


Fig. 4. How the engagement level and the time used to study the algorithm have influenced the learning outcomes

5 Conclusions and future work

We have developed an effortless approach to build and maintain algorithm animations, and we have made a short term evaluation of it. This evaluation compares the learning improvements achieved with two engagement levels: viewing (viewers) and constructing (builders), where our approach is used. Learning improvements were measured in terms of two levels of Bloom's taxonomy: understanding and application.

Results show that, at the understanding level, builders obtained slightly better results than viewers, 16% of improvement; at the application level, builders improved learning the 60% more than viewers. Builders expended much more time than viewers, but they did not complain about it: they thought it was necessary and fruitful to use this amount of time.

According to the students' opinion, builders believed that building algorithm animations had helped them in understanding the concepts of the algorithm, and thought that animations were easy to build. Most of the viewers thought that viewing and building were equally helpful, but half of them thought that both engagement levels should be used together.

We realize that the generalization of these results is limited because of: the low number of students (15), the short period of time evaluated (two sessions of two hours), and the topic used (just the tree breadth traversal algorithm). But we think that this is a promising result because: we have empirical evidence of learning improvements at the understanding and application level of Bloom's taxonomy, using our effortless approach together with the constructing engagement level; and students felt comfortable with this approach, perceiving it as effective and helpful.

As future work, we plan to conduct a long term evaluation of this approach with functional program animations; also, effortlessness have to be evaluated from the instructor's point of view [5] so, we will evaluate the usability of the building/management process of these web-based algorithm/program animations and collections of them.

6 Acknowledgments

This work is supported by the research project TIN2004-07568 of the Spanish Ministry of Education and Science. Also authors want to thank: the reviewers and attendees of the Fourth Program Visualization Workshop for their comments and discussions, Carlos Lázaro-Carrascosa for his valuable help during the experimental session, and the students of the Algorithm Design and Analysis course of the Rey Juan Carlos University for participating in the evaluation.

References

[1] Baecker, R. and B. Price, *The early history of software visualization*, in: J. Stasko, J. Domingue, M. Brown and B. Price, editors, *Software Visualization*, MIT Press, Cambridge, Massachusetts, USA, 1998 pp. 29–34.

[2] Bloom, B., E. Furst, W. Hill and D. Krathwohl, "Taxonomy of Educational Objetives: Handbook I, The Cognitive Domain," Addison-Wesley, 1959.

[3] Grissom, S., M. McNally and T. Naps, *Algorithm visualization in CS education: comparing levels of student engagement*, in: *Proceedings of the 2003 ACM Symposium on Software Visualization* (2003), pp. 87–94.

[4] Hundhausen, C., S. Douglas and J. Stasko, *A meta-study of algorithm visualization effectiveness*, Journal of Visual Languages and Computing **13** (2002), pp. 259–290.

[5] Ihantola, P., V. Karavirta, A. Korhonen and J. Nikander, *Taxonomy of effortless creation of algorithm visualizations*, in: *ICER '05: Proceedings of the 2005 International Workshop on Computing Education Research* (2005), pp. 123–133.

[6] Naps, T., S. Cooper, B. Koldehofe, C. Leska, G. Rößling, W. Dann, A. Korhonen, L. Malmi, J. Rantakokko, R. Ross, J. Anderson, R. Fleischer, M. Kuittinen and M. McNally, *ITiCSE 2003 working group reports: Evaluating the educational impact of visualization*, ACM SIGCSE Bulletin **35** (2003), pp. 124–136.

[7] Naps, T., G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger and J. Velázquez-Iturbide, *ITiCSE 2002 working group report: Exploring the role of visualization and engagement in computer science education*, ACM SIGCSE Bulletin **35** (2003), pp. 131–152.

[8] Stasko, J. and A. Lawrence, *Empirically assessing algorithm animations as learning aids*, in: J. Stasko, J. Domingue, M. Brown and B. Price, editors, *Software Visualization*, MIT Press, Cambridge, Massachusetts, USA, 1998 pp. 419–438.

[9] Urquiza-Fuentes, J. and J. Velázquez-Iturbide, *Effortless construction and management of program animations on the web*, in: R. W. Lau, Q. Li, R. Cheung and W. Liu, editors, *Advances in Web-Based Learning - ICWL 2005, 4th International Conference*, LNCS **3583** (2005), pp. 163–173.

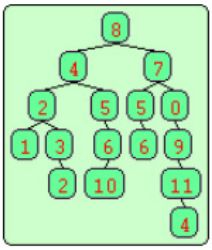
[10] Urquiza-Fuentes, J. and J. Velázquez-Iturbide, *R-zoom: A visualization technique for algorithm animation construction*, in: *Proceedings of the IADIS International Conference Applied Computing 2005* (2005), pp. 145–152.

[11] Velázquez-Iturbide, J., C. Pareja-Flores and J. Urquiza-Fuentes, *An approach to effortless construction of program animations*, Computers & Education **to appear**.

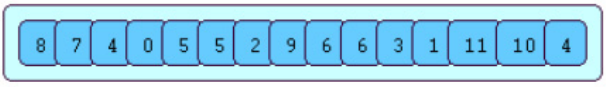
A Knowledge test

1. What are the main ideas of this algortihm?

2.Given this tree, write the result of applying the algorithm to it.



3.Given this list, draw the tree to which the algorithm was originally applied (note that this tree was balanced):



4. Which is the existing relationship among the nodes of the tree, if the result is a strictly ascending ordered list?

5.What modifications should be done on the algorithm to change the traverse direction to right-to-left?