To appear in International Conference on Education and Information Systems: Technologies and Applications (EISTA'03)
Orlando, USA, July 31 to August 2, 2003

A toolbox to teaching and learning data structures: "Live Algorithms"

Pedro Luis Kantek Garcia Navarro.
Centro Universitário Positivo, UNICENP.
Curitiba, Paraná, 80410-180, Brazil.
E-mail: kantek@pr.gov.br

ABSTRACT

This article proposes a toolbox to help teachers and students in a hard task of teaching and learning algorithms and data structures. The name of that toolbox is "Live Algorithms". The text starts with issues of abstracting manipulations. Next, Live Algorithms is explained, where I intend to show how it could help us. The toolbox is described, its components are commented and the advantages of its use are listed. Finally, we compare the results obtained in two regular classrooms: the first worked with Live Algorithms and the other, without it, in a traditional approach.

Key-words: data structures learning, algorithms learning, toolbox algorithms teaching.

1. INTRODUCTION

Many authors, as Niklaus Wirth (1986) for instance, has pointed how the duty of teaching and learning algorithms as well as programming is very hard. In fact, to develop the capacity of building software, mainly those big size and complexity is very arduous challenge to people and organizations.

In the past, some kind of approaches has been taken to make easier this learning. One, and very important, was the utilization of a new algorithmic language; nevertheless, seems like any programming language, already known. To justify the approach, authors may say "When we separate *what* and *how* will be made something, we introduce an important task in the project of someone program: the write, test and expurgation of the algorithm. Not as a piece of programming code, but as an application of one idea.

When someone combines the project algorithm task with it's implementation in a real computer using available program language, probably introduces unnecessary complexity. As an example, when someone studies the Quad Tree algorithm, all the emphasis must be placed in the structures idea, functionality, and its fundamental characteristics. The organization used in the implementation (language, kind of data, primitive or derived structures, statements, and/or available facilities of the development and environment) must come right after its understanding, projecting, testing, expurgating and approving as an algorithm. As soon as someone could be sure about its robustness and proper functionality, the work in a real computer could be initiated.

An important comment here, is that the only pre-request to begin the algorithm study, and also the program construct, is only a project, paper, pencil, and a lot of human understanding. No computer is needed

2. PREVIOUS PROPOSES

A very frequent one, in scientific literature, is when someone tries to study the actions in a production of animated sequences that go together with the algorithm. This approach has as the main advantage of materializing function. One example of this approach was the sounded and colored_30 min movie *Sorting Out Sorting* made in 1981. [Baecker, 1981]. It compares several kind of methods. Another examples are BALSA method by Mark Brown (1984) and John Stasko TANGO (1990). At Internet there are JELIOT software [Haajanenn et alli, 1997]. The start message of TANGO is emblematic: *Welcome to the algorithm theater!".*

One problem here is the passive role lived by the student. He is invited to admire, to see the algorithm spatial development. How much learning each student can acquire and how much real utility by increasing knowledge this method brings to people is an open question. See [Byrne at Alli 1996]. More recently, wider proposals that appear will ask to the student to predict the algorithm behavior.

Another important question is the low generality from the proposals of the method. The majority concentrates a unique aspect of the computer's science. So, the movie above shows and compares only few sort methods. The Hill work [2002] deals only with tree algorithms. It seems there is very difficult to be generalist in animated matter.

3. THIS PROPOSAL

The Live Algorithm begins showing to the student one precise algorithm. So, he (or she) is invited to answer some kind of questions about the algorithm showed. It can be a variable value in some algorithms' point. Or can be an algorithm full answers for some input data. What or how many statements will be performed.

The exercise doesn't ask to development of a new algorithm. This task comes later, when the outstanding algorithms knowledge stand. Wirth [1986], by example, defends this strategy.

4. LIVE ALGORITHMS OBJECTIVES

- The student is demanded to make a task which has a knowledge of functionality algorithm already presented as pre-requested
- Help the student to understand the algorithms' dynamic aspect and its structures of outstanding data;
- Have wide spectrum including the main themes studied in data structure and algorithms' project;
- Be independent of having computers in classroom;
- Have facility and agility for correcting papers; as a result, the pupil will have fast feed back;
- Make distinct postulation for each student in a way that, everyone should produce his own work, not allowing treachery.
- Allow matching duties in order to obtain best grade's average in which the theory test has less weight.
- Permit the students to make papers out of classroom as a contribution end effort to raise their grades
- Show practicing how algorithms, which seems simple can be very complex behavior.

5. LIVE ALGORITHMS DESCRIPTION

Many programming computers (presently 108) generate papers containing individual exercises. All exercises have the same aspect, because they share an expressive part of the produced code.
If the code is not shared then it has: introductory text about the studied problem; a case generator for a set of input data for this problem; a solver of the above case.

Each exercise paper has a complete example of a resolved exercise with comments and one new the student is asked to solve. To teacher, Live Algorithms produce a list with the correct answers questions. Each of generators was code in APL one of the bests programming language for this kind of application. The APL program generates a LATEX file, so, before the student's use, this file needed to be compiled. Now, the visual aspect of Live Algorithms is very nice.

The generator case is very hard to produce. How the generation of data is random this part of software has the responsibility to find an exercise that would be interesting to the student. It must be non trivial, and it must have an acceptable complexity. People (well, almost all people) must solve the exercise in 45 minutes. Particularly cases, which not deal with algorithm main idea, must be rejected.

How is produced an exercise by computer program if each student has a different case about the same algorithm? So, the effort demanded is the same. One student can't copy the neighbor's paper, because of the differences.

6. EXAMPLES

Here, it is a title of Live Algorithms.

Turing Machine: the exercise shows theory, and a small TM. There is also, an input stream of bits. The student is invited to say in what state the TM was conducted.

Many have algorithms examples (sorts, searches, arithmetic manipulation, nested ifs and else's, mix of whiles and repeats, etc). In this exercises the student is invited following the flow and predict what are the results found when the algorithms end.

One LA has the Clavius and Lilus algorithm to find the day of Eastern in all the years past 1587. The student must have found the Eastern in a random year offered by the program.

The recursive approach has around 10 different exercises: The chess horse path across n x n board, the anagrams producer, and the quad tree implementation. In all cases, the student must answer what is the final computation result.

The fundamental structures (stacks, queues, lists, linked lists, skip lists and similar) have 14 exercises. One important question to student is finding errors in linked lists dumped.

Application of linked lists is also presented. An MS-DOS alocator (FAT processor) and a UNIX alocator ask to the student to construct the control tables in both environments.

The tree chapter is the wider. There are theory exercises, with inclusions, deletions, and replacing in binary trees, B-trees, binary search trees. There is also, practical approach: Huffman trees, dBase trees (ndx files), Sybase trees.

Graph theory is also presented. Minimal path, Dijkstra, Kruskal, Warshall, Ford-Fulkerson, Edmonds-Karp Algorithms, topological sort, etc.

In sort methods, there are 5 papers of exercises. Beginning with the simply ones (bubble, insertion, selection, shake, etc) and beyond: Shell, quick, heap etc. There are also nets of comparison.

In artificial intelligence world there are some algorithms: A*, Minimax, Genetic algorithms, neural nets, expert systems. By the way, these examples are almost trivial and the objective is demonstrating the main functionality of each one.

Petri Nets is here. 3 exercises were produced. An ATM machine, semi-adder and a randomic petri net. Theory needed is described in the exercise, to help the student to answer the question.

Images BMP and GIF are also studied. The internal characteristics are showed. All kind of bit mapped images, the LZW algorithm, the convolution filters, the histogram equalization algorithms are here.

Cryptography is other main chapter: The DES and the RSA approach are exposed and also, the ENIGMA machine. Steganography exercise, signature in files, and an invitation to break one RSA communication are examples of the existence exercises.

7. COMPARISON BETWEEN CLASSIC AND LIVE ALGORITHMS APPROACHES

Before the existence of this toolbox, any matter of algorithms or data structures was presented through the sequence tasks:
a) The topic presentation, its importance in the computer science and informatics profession.
b) Main algorithms presentation, working with the students in algorithms flows.
c) Aspects of generality, complexity, advantages and disadvantages of each one
d) Practical cases presentation, where these algorithms are used
e) Question the students to produce a new exercise about these matters.

Comments about:
1. The a,b,c and d tasks don't demand the student an active work. So, the feedback that teacher receives is small and without meaning.
2. When the "e" task begins, the majority of the students copy each other.
3. Serious difficulty: or the students quit during the scholar time, or the teacher needs to open the criteria necessary to graduate the student.

When the Live Algorithms is used, the task list above can be rewrite as
a) Topic presentation and comments about its importance in computer science
b) Main algorithms presentation
c) Practical classes using Live Algorithms: Each student gets an individual challenge. Some can do it in fewer minutes. Other needs one or two working days. It doesn't matter.

Comments about:
1. The students play an active role in learning with LA. The majority said the learning is facilitated. More people can produce some kind of programming code when compared with other classes that use traditional approach
2. Use LA implements an important rule in modern pedagogy: Give to each student the time he needs to solve any problem. In a student set, "*treat each on as an individual and not as a group*".

8. SUMMARIZING:

In the 10 last years, I used LA in data structure and Advanced Topics classes. The students are improver and happy with this approach. Average grades are better today as an opposite on 12 years back. The computer power is been used to enrich matters and topics in classroom.
This is more special and noble function for the computer, than just use it as a tool in presentations using power point.

9. REFERENCES

[1] Baecker R. "Sorting out Sorting", sonorous and colored 30 min movie, Dynamic Graphics Project, University of Toronto, 1981.

[2] Brown M. e Sedgewick R. "A system for algorithm Animation", Proceedings of ACM SIGGRAPH 1984, Minneapolis, July 1984, pp 177-186.

[3] Byrne M, Catrambone R. e STASKO J. "Do algorithms animations aid learning?" Technical Report GIT-GVU-96-18, GVU Center, Georgia Institute of Technology, Atlanta, august 1996.

[4] Haajanen, M. Pesonius, E. Sutinen, J. Tarhio, T. Teräsvirta, P. Vanninen: Animation of user algorithms on the Web. In: Proc. VL '97, IEEE Symposium on Visual Languages, IEEE 1997, 360-367.
http://www.cs.helsinki.fi/research/aaps/Jeliot/

[5] Hill T. "Assessing the instructional value of students predictions in tree animations", SIGCSE Doctoral Consortium Application, University of Mississippi, 2002.

[6] Stasko J. "TANGO: A framework and system for Algorithm Animation", Computer, vol 23, no.9, September 1990, pp-27-39.

[7] Wirth, Niklaus. Algorithms and Data Structures. Prentice
Hall. 1986.

.