LUCAS – An Interactive Visualization System Supporting Teaching of Algorithms and Data Structures

Erich Schikuta
Department of Knowledge and Business Engineering, University of Vienna, Austria
erich.schikuta@univie.ac.at

Sylvia Schikuta
Ferdinand Porsche Fernfachhochschule, Vienna, Austria
sylvia.schikuta@hotmail.com

Abstract: The LUCAS (Learning and Understanding of the Characteristics of Algorithms and data Structures) system is an interactive Web based program to visualize the dynamical behavior of data structures and algorithms. It is used as a supportive tool for under-graduate and graduate courses on these topics. It aims for clarity and vividness, simplicity, portability, and extendibility and supports user controlled dynamics following a so called "tape recorder approach".

Introduction

The main problem of teaching and studying algorithms and data structures is, due to the inherent complexity, to visualize how these algorithms operate. Conventionally, algorithms are described in a static way by their code representation within a (mostly) formally defined description framework (programming language or pseudo code). This makes it often hard to understand their specific flow of control.
Several lecture books try to overcome this problem by sketching the state of an algorithm by the graphical representation of the data it is operating on (e.g. (Sedgewick 1992), (Schikuta 2008)), see Figure 1 as example for the graphical representation of character data for insertion in a Trie data structure). This is done by graphically depicting specific states of the data structure during the execution of the algorithm. However this approach gives snapshots of the execution flow at specific predefined points only and does not allow students to interactively explore the behavior of the algorithm in focus.
To overcome these problems we present in this paper the LUCAS (Learning and Understanding of the Characteristics of Algorithms and data Structures) System (Babic & Deischler 2008), which was developed as a portable and extensible Web-based interactive system providing for simplicity, homogeneity, and user controlled dynamics for the presentation and animation of data structures and algorithms. LUCAS builds up on and extends the original VADer system, which was developed several years ago as master thesis (Maderbacher 1999).

Figure 1. Graphical character data set "Lara" for Trie insertion[1]

The paper is organized as follows. In the next section a state of the art of similar systems is given followed by the presentation of the LUCAS system and its design principles. Then we illustrate the implemented data structures and algorithms and describe the technical implementation. The paper closes with the practical

[1] The "Lara" data set input was provided by Jürgen Mangler.

experiences of the use of the system in an undergraduate course and the future plans and development of LUCAS.

State of the Art

For the visualization of algorithms and data structures a large number of respective system were developed. A taxonomy of such systems is presented by Price et al. (Price & et al. 1993), but basically all these systems can be mapped to two groups taking their focus on teaching algorithms and data structures into account (Baker & et al. 1999). On the one hand are visualization systems, which provide for elaborate visualizations of the dynamic behavior of algorithms and data structures in a more or less automatic way, but do not support interaction of the users. Original members of this group are systems like TANGO (Stasko 1990) and BALSA (Brown 1988), or systems like JAWAA (Pierson & Rodger 1998) and GAIGS (Naps & Bressler 1998).
On the other hand are these systems, which allow for interactive control by the user. We believe that these systems, which follow the ideas of Incense (Meyers 1983) qualify definitely better for the teaching purposes, because the user has the possibility to create and explore specific situations of the data structures in focus at her will according to her learning process. Members of this group are Amethyst (Price & et al. 1993) and Field (Reiss 1997) and also the systems developed by Sangwan et al. (Sangwan & Korsh & LaFolette 1998). However, these systems follow a “debugging” approach by displaying the physical organization of data in memory and no conceptual view of the behavior of the algorithms and data structures. This situation motivated us to design a system which provides for interactive user control on a conceptual level of description and led to the development of LUCAS.

The LUCAS System

The aim of the LUCAS project (LUCAS 2008) was on the one hand to deliver a tool to support courses on algorithm and data structures in the curriculum of computer science studies, and on the other hand to allow students the self study of these topics. However, it is advantageous for users to have some basic knowledge about the supported algorithms and data structures, but the goal of LUCAS is to make it easy to explore them and make it even easier to understand them.
At the moment the LUCAS system supports a useful number of algorithms and data structures, which covers the contents of a basic course on data structures. The program visualizes the data structures Binary tree, AVL-tree, B+ tree, and Trie. For each of this data structures LUCAS supports methods for creation, insertion, search and deletion of data. Further LUCAS supports three groups of sort algorithms, simple, advanced, and linear sort algorithms. The implemented simple sort algorithms are Selection sort and Bubble sort, and the advanced are Mergesort, Quicksort and Heapsort. As examples for sorting in linear time Bucket sort, Radixsort and Counting Sort are visualized. Additional modules for hashing and graph algorithms are just under development and will be integrated into the system soon.

Design Principles

To succeed in the development of such a supportive educational tool we defined at the beginning the following design principles, which were followed and adhered during the development of LUCAS.
For the presentation it was made sure to choose a clear graphical layout of the data structures and/or the control flow of the algorithms. Further it is possible for the students to choose the inputs to the algorithms without any restriction. This makes the program even more useful for test preparations by allowing solving and understanding old exams.

Simplicity and Homogeneity

The user interface was designed to allow the usage of the system without studying manuals to give also casual users the possibility to make use of the system. The different forms follow a similar structure and layout of the control elements (buttons, text fields, etc.). Further conventions for labeling, coloring and formatting of textual and graphical elements were defined.

Portability

A major design criterion was portability to support all available hardware and software platforms accessible at the university campus. Thus made necessary a Web based approach and led to the usage of Java as development environment. The Web is obviously also the ideal tool for the distribution of the system. Therefore we provide a stand-alone program and an applet which can execute in any modern Java-enabled Internet browser.

Extendibility

The possibility of adding new data structures and algorithms is inevitable for the design of such systems. This allows on the one hand the adaptation of the system to new courses and on the other hand the usage of the system as a framework for practical student's work.
Extending LUCAS is a simple task. New modules are developed and integrated easily by following a guided tour consisting of few steps. This makes LUCAS extremely viable for the use in practical courses where the presentation system in focus serves also as the development system for new modules.
Technically implements the LUCAS system a new modularized approach. The basic system provides only the means for the interface (presentation and user control). The specific data structure and algorithm modules are separated and realized by classes with a standardized call interface. This allows an arbitrarily configured LUCAS for personal use, where the specific modules can be provided via the Web, added locally or accessed remotely via the inherent Java functionality. This approach allows arbitrary users to add their own modules and therefore makes a continuous and easy extension of LUCAS possible.

Tape recorder approach

A specific feature of the program is the so called "Tape recorder approach". Hereby the user can arbitrarily go forth and back in the execution of specific data structure methods or algorithm steps. The user is able to introspect all specific situations of a data structure or of an algorithm by going back in the trace of the executed steps. Therefore, it is possible, that the user can make new decisions at any point of the execution flow allowing the user to answer "what would have been" questions.
Compared to the conventional static book approach, that the program stores all proceeded steps during the execution of the specific data structure methods (e.g. insertion, deletion, search, etc.) or algorithms that the student can easily go back and forth one or more steps (similar to a tape recorder) to analyze what happened. Thus all operations can be re-done again and the central points of interest of the algorithm behavior (e.g. split of root node, swap of pivot element, etc.) can be identified easily.

Interface

The LUCAS interface (see Figure 2) is held simply on purpose. Basically the user is confronted with only one window, which contains all necessary control and presentation elements.
The data structure or algorithm in focus is chosen by clicking the algorithm folder opening the respective form specifically designed according to the data structure's or algorithm's characteristics. All supported methods of a data structure, respective the specific algorithms, are realized by method buttons to activate them. Aiming for the presented design principles the user interface follows, as we call it, a tape recorder approach. This allows the user to go forward and backward step-by-step during the method invocation. The graphic windows depicts the respective state of the data structure or the advance of the algorithm according to the execution flow. It shapes dynamically to the size of the data structure by inserting scroll bars, if the available window is to small. The employment of a homogenous color scheme tells the user what has happened to a specific data structure element in the algorithmic last step (going forth or back).
The creation of the graphical representation of the data structures or algorithm elements is done automatically by the program. It is always guaranteed that a clearly arranged and user-understandable layout is chosen.

Implemented Data Structures and Algorithms

Data Structures

The supported data structures in LUCAS are based on the standard definitions found in the literature (Sedgewick 1992). However, a help system provides specific information on the different data structures and gives an algorithmic description in a Java-based pseudo code.
At the moment are four tree structures implemented in LUCAS.

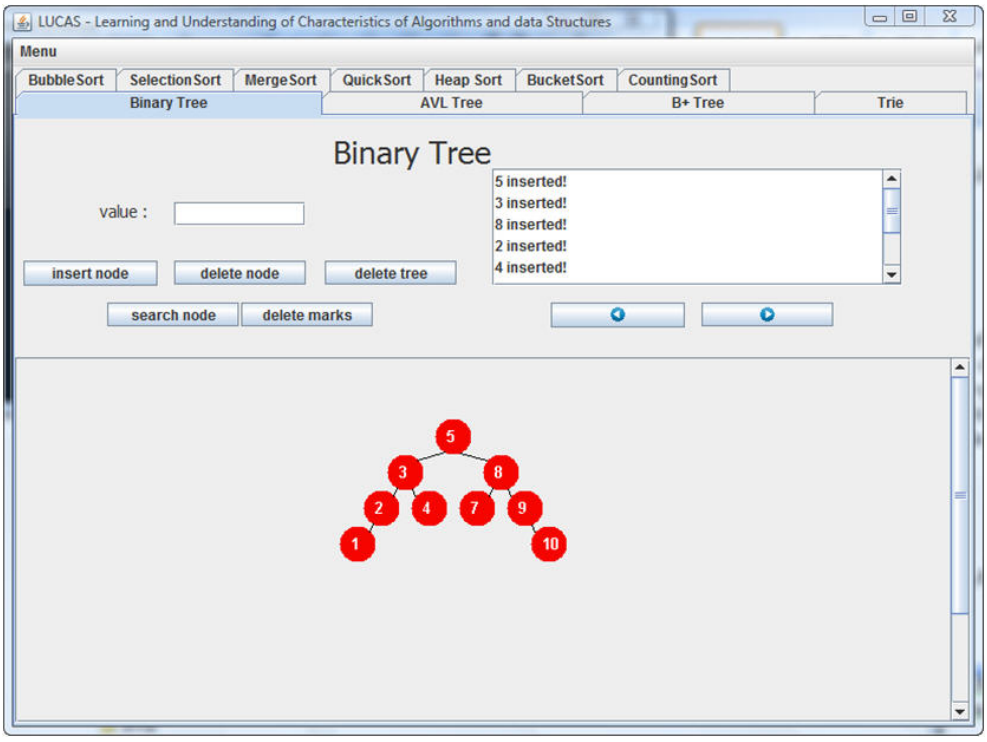

Figure 2. LUCAS Interface

The binary search tree (see Figure 2) is an ordered binary tree with the property that the value stored in a node is greater than all elements stored in the nodes of the left sub-tree and less than all elements stored in the nodes of the right sub-tree. However, doubled elements are not allowed, but the user is informed, if he tries to insert one. The color scheme informs the user, which node has been visited or not, found, moved, inserted or deleted.
Methods are creation, insertion, search and deletion.
The AVL (Adelson-Velski, Landau) tree is a balanced binary search tree where smart rotational algorithms keep the balance property under inserts and delete operations.
The B+-tree is a balanced, key-interval-based, multi-way tree derived from the well-known B-tree. We chose the B+-tree, because it is the generally used index structure in database systems. The order of the B+-tree, which is defined at the creation of the tree, specifies how many elements can be stored in each node. Methods are creation, insertion, search and deletion.
The Trie is a prefix tree to store character strings. The end of a string is denoted by the “Do not enter” sign.
Methods are creation, insertion of a string, search and word-list.

Algorithms

Besides the above listed data structures LUCAS presents also some selected sorting algorithms. Basically representatives of simple ($O(n^2)$ time), advanced ($O(n \log n)$), and linear ($O(n)$) sorting algorithms are provided until now. Two simple sorting algorithms (order $O(n^2)$), Selectionsort and Bubblesort, two advanced sorting algorithms (order $O(n \log n)$), Quicksort, Mergesort and Heapsort, and three linear sorting algorithms, Bucketsort, Radixsort, and Countingsort are implemented.
At the beginning the values (number and starting sequence) to sort are defined by the user and are depicted by different sized bars (see Figure 3). Then it can be specified, if the algorithm executes step-wise or at once, similar to an animated film. All important steps, as comparisons and swaps, are depicted.

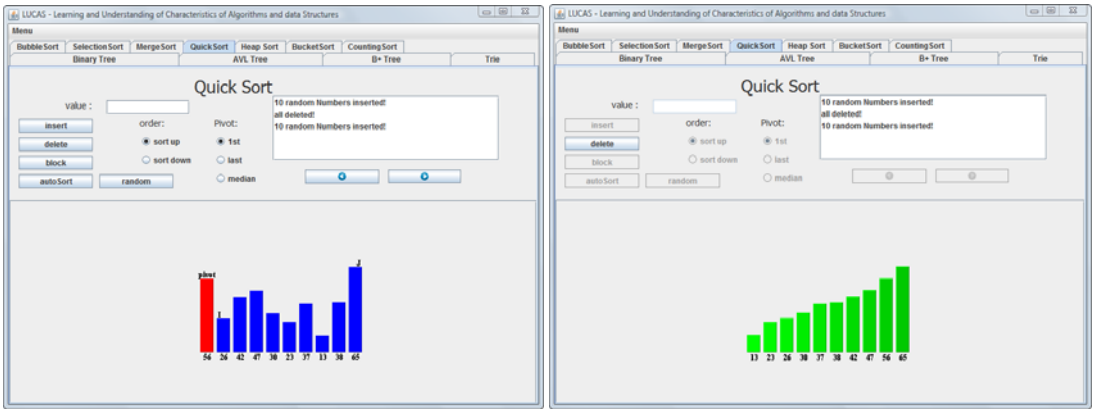

Figure 3. Sorting

Implementation

The program is written in Java using the Java Beans classes purely. These classes provide all needed methods to realize the visualizing. This framework gives unlimited portability and makes it possible to extend the contents of lectures easily. Thus we had to use standardized libraries for the development and made the decision for the Java Swing library. Consequently we have a unique looking system on Windows and Unix environments now.
The system can be executed as an applet or a stand-alone Java program. The applet mode allows executing the program from the Web within a Java enabled browser. This version supports also the easy access and the comfortable distribution over the Web. However, it is also possible to run the program in a stand-alone mode.

Experiences

LUCAS proved very well in practice and gained amazing acceptance by the students. The program is used as a supporting tool for an undergraduate course on algorithms and data structures in the Bachelor Curriculum of Compute Science at the University of Vienna. However, the experiences showed that it gave its best results as a supporting tool besides practical exercises and an accompanying lecturing course. The development of the first version of LUCAS (which was called VADer then) was motivated by the need for a tool for the students for the preparation of exams. The students could get the answers to exam questions by this system. E.g. typical exam questions as “Depict graphically the structure of a binary tree after inserting the numbers of your student id.” could be solved by the students themselves without the need for a tutor. So LUCAS allows them to enter the examination questions directly into the system, presenting the correct result and even more the way how to produce it. By the feature of the “Tape-recorder-approach” the students have also the possibility to discuss some solutions and to compare algorithmic behavior for similar inputs.
LUCAS is also used and very well received in algorithm and data structures courses at the Ferdinand Porsche Fernfachhochschule, an applied University specializing on distant learning offering all courses via e-Learning.
The system proved very well and the course supported by LUCAS was even awarded by the price of “Innovative Lecture” of the University of Vienna.

Future Plans

In the near future we plan to extend LUCAS with several new algorithms and data structures to cover the contents of graduate and post-graduate courses. Further we intend to use the system also as the basis for practical exercises. From the educational point of view the choice of Java as implementation language proves advantageous, due to its clear concepts and general spreading (Java is one of the entry languages in the computer science curriculum).
LUCAS is a living system. That means it is continuously improved and extended by the community, which consists mainly from students in the undergraduate and graduate courses on algorithms and data structures. The code and basic version of LUCAS (with the described basic modules) is available for everyone via its Web site, which is http://www.pri.univie.ac.at/workgroups/lucas.

References

Sedgewick, R. (1992). Algorithms in C++, Addison-Wesley

Schikuta, E. (2008). Materials to the Bachelor Course “Algorithms and Data Structures (Algorithmen und Datenstrukturen)” held at the University of Vienna and the “Ferdinand Porsche FFH” University of Applied Studies

Babic, N., & Deischler, G. (2008). Bachelor thesis, LUCAS, University of Vienna

Maderbacher, H. (1999). Visualisierung von Algorithmen und Datenstrukturen, Master thesis, University of Vienna

Price, B. A., & et al. (1993). A principled taxonomy of software visualization. J. Visual Languages and Computing, 3(3):211-264

Baker, R.S., & et al. (1999). Testers and J. Visualizers for Teaching Data Structures, ACM SIGCSE Bulletin, vol. 31, Issue 1, March 1999.

Stasko, J. T. (1990). TANGO: a framework and system for algorithm animation. IEEE Computer, 23(9):27-39

Brown, M. H. (1988). Algorithm Animation. MIT Press

Pierson, W. C., & Rodger, S. H. (1998). Web-based animation of data structures using JAWAA. Proc. SIGCSE

Naps, T. L., & Bressler, E. (1998). A multi-windowed environment for simultaneous visualization of related algorithms on the WorldWideWeb. Proc. SIGCSE

Meyers, B. (1983). A system for displaying data structures. Computer Graphics, 17(3)

Reiss, S. (1997). Visualization for Software Engineering Programming Environments. In J. Stasko, J. Domingue, M. H. Brown, and B. Price, editors, Software Visualization: Programming as a Multimedia Experience, chapter 18, pages 259-276. MIT Press

Sangwan, R., & Korsh, J., & LaFolette, P. (1998). A system for program visualization in the classroom. Proc. SIGCSE, 1998.

LUCAS (2008). LUCAS project web site, http://www.pri.univie.ac.at/workgroup/lucas

Acknowledgements

I want to thank all students who contributed in classes to the development of the LUCAS system. On this occasion my special thanks go to Nebojsa Babic and Gernot Deischler, who implemented the first version of LUCAS. Further I also want to thank all students who contributed to the extension of LUCAS by providing additional modules as part of their class work.