A learning tool for the visualization of general directed or undirected rooted trees

K. Paparrizos, N. Samaras & A. Sifaleras
Department of Applied Informatics, University of Macedonia, Greece

Abstract

This paper presents a new learning tool developed for the visualization of general directed and undirected rooted trees. This visualization tool for teaching graph and network algorithms provides an interactive view of the subject being taught to the students. The tool is designed for three different groups of users: developers, instructors and students. It could be used for the visualization of any algorithm that generates a sequence of trees. This new tool minimizes the instructor's effort and gives him the possibility of drawing trees easily with many different properties. It can be used efficiently in courses like Graph Theory or Network Programming or in an introductory course like Algorithms and Data Structures. We believe that this tool will be an effective supplement to traditional classroom education. It is a commonly accepted fact that teaching can be vastly amplified when it is done visually and interactively instead of in the traditional way. Therefore, a summary of other educational tools concerning tree drawing is presented. Benefits and drawbacks are thoroughly described in order to support the efficiency of modern learning technology.
Keywords: interactive learning environment, intelligent tutoring systems, tree visualization, Matlab.

1 Introduction

Data Structures and Algorithms is a fundamental course in Computer Science. However, many students find it difficult because it requires abstract thinking. It would be very helpful for students, if there was a visualization tool of some basic data structures such as trees and graphs. Students are able to learn interactively by receiving feedback provided either by the instructor or by a computer program. The tool would allow students to see how a tree is traversed in different


The Internet Society: Advances in Learning, Commerce and Security, K. Morgan & M. J. Spector (Editors)

traversals (pre-order, in-order, post-order, and level-order). This tool could be used as an effective supplement to the traditional classroom education and textbook for Data Structures and Algorithms courses. Usually an instructor who teaches Graph Theory topics such as trees had to draw on a whiteboard a tree and explain its pre-order using a pen. That surely is not a flexible way to make many different examples, furthermore it is very much time consuming. This paper describes how the teaching of Graph Theory topics such as trees can be improved in compare to the traditional tutoring system.

The teaching of an algorithm or a data structure can be greatly enhanced with such a visualization tool. It is a commonly accepted fact that teaching can be vastly amplified when it is done visually and interactively instead of purely theoretically, Tal and Dobkin [1] and Naps [2].

The paper is organized as follows. In section 2 we briefly describe any previous work which has been done. In section 3 we give the motivation which led us to develop this visualization tool. In section 4, we present the analysis and design of our tool and give an illustrative example. In section 5 we focus on implementation principles. Finally in section 6 we show that the implemented visualization tool can be extended in order to visualize Simplex type algorithms for Network Programming problems.

2 Previous work

There is a vast amount of research of different visualization tools conducted over the years. The development of technologies and the evolvement of the World Wide Web have influenced education. Instructional Web sites and courses on the Web have grown dramatically. Web-based courses that consist of the syllabus, assignments and lecture notes are now widely used. Two interesting Web sites are the ones developed by Duane [3] and Mukundan [4]. In particular, Mukundan's homepage has a comprehensive applet which demonstrates the three different types of binary tree traversals, i.e., pre-order traversal, in-order traversal, and post-order traversal. The user can see the three types, but not create his own general trees. Our educational function is intended to aid Computer Science students facing for the fist time Data Structures and Algorithms. Therefore, as mentioned by Foley [5], ease of use becomes our main consideration. The design of educational software needs to be learner centred rather than user centred Soloway *et al* [6].

There are many tools for graph drawing. A very good source containing a large collection of graph drawing tools is a link provided by Tamassia [7] homepage. This paper is based on ideas presented in the books by Battista *et al* [8] and Goodrich and Tamassia [9]. Both books are very good sources and contain useful information. Also, *GraphEd* is a graph editor, by Himsolt [10], which provides graph layout algorithms and an interface for application modules. It provides a wide variety of graph drawing algorithms including spring embeddings and special algorithms for drawing trees, and planar graphs. Graphlet [11], the successor of *GraphEd*, is a toolkit for graph editors and graph

The Internet Society: Advances in Learning, Commerce and Security, K. Morgan & M. J. Spector (Editors)

algorithms. Both of them have been developed at the University of Passau, Germany.

Many tools have been based on *VGJ* tool, (Visualizing Graphs with Java, McCreary C.) The VGJ Graph Drawing Tool [12] is a tool for graph drawing and graph layout. Graphs can be input into *VGJ* in two ways: with a textual description (GML) or through a drawing the user creates using the graph editor. The user can then select an algorithm to layout the graph in an organized and aesthetically pleasing way. Also there is the aiSee [13] tool, created in *AbsInt, Angewandte Informatik GmbH Germany*, which reads a textual graph specification and visualizes the graph. Its design has been optimized to handle large graphs generated by applications automatically. Another graphic tool which can handle tree - tuple automata is Taja 2.0, by Lecland and Rety [14]. Furthermore, data structures libraries in Java, *JDSL,* have been developed. Information for tree structures can be found in the following papers: Tamassia *et al* [15] and Goodrich *et al* [16].

Finally, commercial graph drawing systems are available from Tom Sawyer Software [17]. There are systems and libraries for UNIX and MS-Windows, mostly written in C++.

3 Motivation

Recently many scientists and engineers are using Matlab [18] not only as a modeling and analysis tool, but also as a visualization tool. There is a function in Matlab which plots undirected trees, the *treeplot(p)* function, where p is the vector of parent pointers, (the node i is the root if and only if $p(i) = 0$). There is also the *treelayout(parent,post)* function, where parent is the vector of parent pointers and post is an optional post-order permutation on the tree nodes. Function *treelayout(parent,post)* does not plot trees, it just computes the coordinates of the nodes.

Function *treeplot(p)* does not plot the tree structures correctly. For example, if the tree defined by the vector p = [0 1 1 1 1 2 2 2] is plotted in Matlab by typing *treeplot(p)* in the command window, the tree shown in Figure 1 is drawn.

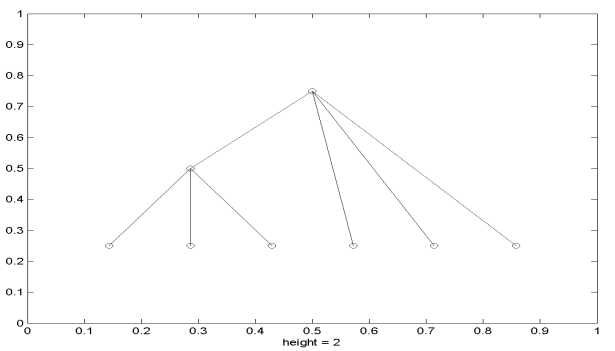

Figure 1: An incorrect tree plot.

The Internet Society: Advances in Learning, Commerce and Security, K. Morgan & M. J. Spector (Editors)

We see in Figure 1, that some leaves of the tree are not drawn at the correct level. In fact, all leaves are drawn at the bottom level. Furthermore *treeplot(p)* does not support directed trees and it does not show the label of each node.

4 Design and implementation issues

We developed a new function for the correct visualization of general directed and undirected rooted trees. Our function *Drawtree(p, pre, x)* is an educational purpose function which has the following three inputs:

a. *p* is the vector of parent pointers in which *p(r) = r* implies that *r* is the root.
b. *pre* is the vector of the pre-order traversal.
c. *x* is a vector which describes the arc's directions.

For any non root node *i, x(i)* = 1, (*x(i)* = -1) implies that arc *(p(i), i)*, *(arc (i, p(i)))* is contained in the tree. Also, it is set *x(r)* = 0 for the root node *r*. This vector is an optional choice for the user. In case it is passed as an argument to the function the tree will be directed.

For aesthetic reasons we slightly modified Matlab function *quiver*. In particular the size of the arrow head and the width of the base is 25% instead of the size 33% used in Matlab.

Currently *Drawtree(p, pre, x)* consists of 186 lines of source code. *Drawtree(p, pre, x)* exploits the power of Matlab Programming Language, see for example Marchand and Holland [19], and therefore it is not only flexible and customizable, but also easy to use. The student, who wishes to install the Drawtree.m file, can install it as he would have done with any other Matlab m - file. First he will select File → Select Path → Add with subfolders and then choose the appropriate folder which contains the above file. In addition, the student can choose some parameters with the help of certain pop up lists. He may select the color and the shape of the nodes. The pop up menus are illustrated in Figures 2 and 3.

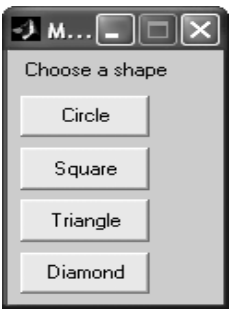

Figure 2: Selection of the node's shape.

The Internet Society: Advances in Learning, Commerce and Security, K. Morgan & M. J. Spector (Editors)

Figure 3: Selection of the node's color.

In order to use our software, the user may type the next commands in the Matlab Command Window:

p = [18 1 1 2 2 2 3 3 18 15 12 9 15 12 9 15 12 18 3];

pre = [18 9 12 14 17 11 15 16 10 13 1 2 4 5 6 3 8 7 19];

If the user wants the tree to be undirected, he may now type the command *Drawtree(p, pre)*. The resulting tree is the one illustrated in Figure 4.

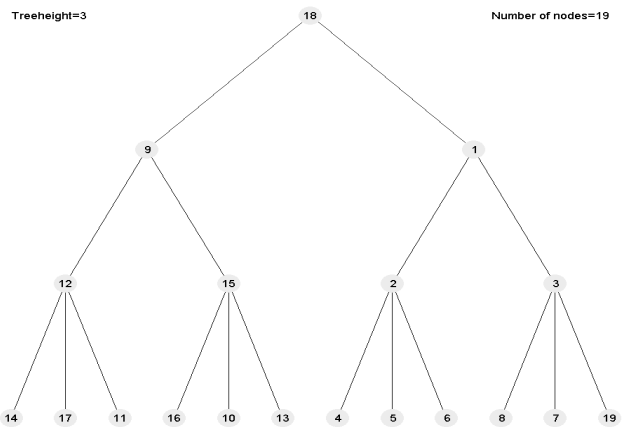

Figure 4: An undirected rooted tree plotted by function Drawtree.

The Internet Society: Advances in Learning, Commerce and Security, K. Morgan & M. J. Spector (Editors)

In case the user wants the tree to be directed, he must insert as an input in the function the following *x* vector:

$$x = [1\ 1\ -1\ -1\ -1\ 1\ 1\ 1\ 1\ -1\ -1\ 1\ 1\ 1\ -1\ 1\ 1\ 0\ -1];$$

Now, the command *Drawtree(p, pre, x)* will plot the tree in Figure 5.

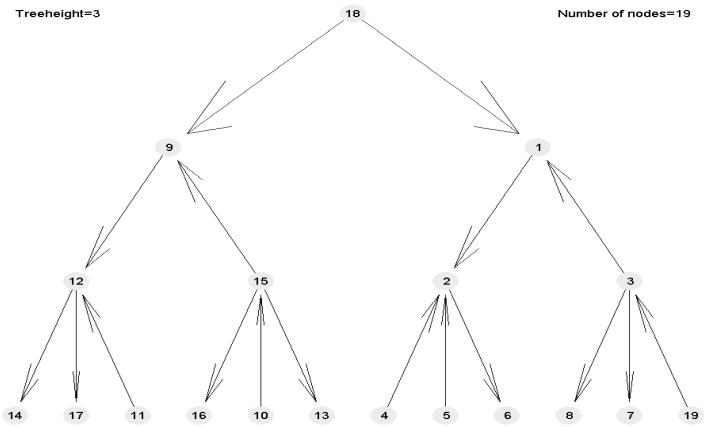

Figure 5: A directed rooted tree plot plotted by function Drawtree.

The difference between the trees shown in Figure 1 and Figures 4 and 5 is now clear. In figures 4 and 5, the trees are drawn correctly. In particular, the nodes are drawn at the right level. Moreover the plot includes the tree height and the total number of nodes.

File *Drawtree.m* contains analytical instructions, so that the user can verify the facts mentioned earlier. It is obvious that instructors can draw any kind of tree easier than they do at the traditional classroom teaching of trees. Besides, they can interactively modify the tree structure and finally the students can adjust easier to this new learning environment.

In order to achieve the results just mentioned, an algorithm drawing trees has been developed. The main part of the algorithm is devoted to the calculation of the nodes coordinates. The tree is drawn in such a way, so that its arcs do not cross with each other.

In order for the tree to be drawn correctly, all the nodes should have appropriate coordinates. The root node is drawn, of course, at the top of the tree. All the nodes being in the same level are at the same distance, in the y axis, with the nodes of the previous and the next level. Therefore, the distance in the y axis, between all the nodes of the first and the second level is the same as for the nodes of the second and third level.

The Internet Society: Advances in Learning, Commerce and Security, K. Morgan & M. J. Spector (Editors)

As far as concerns the x axis, the computation of the nodes x axis coordinates begin from the x axis coordinates of the leaves. Specifically, the node being at the bottom level, at the left corner has been assigned with zero for the x axis coordinate. Every next node of the same level has such coordinates for the x axis, so it is at the same distance from its left or right side, node.

At this point the tree is not ready yet to be drawn. In order to avoid the edge's intersection, for all the nodes above the bottom level, their coordinates for the x axis are being edited so that the parent node is drawn in the middle of its children distance (node's movement). Hence, all the nodes at the final tree will be drawn at the correct place.

The details mentioned before, concerning the node's coordinates for the x, y axis, are shown in Figure 6. It is the same as figure 5, except that now the user can see the axis.

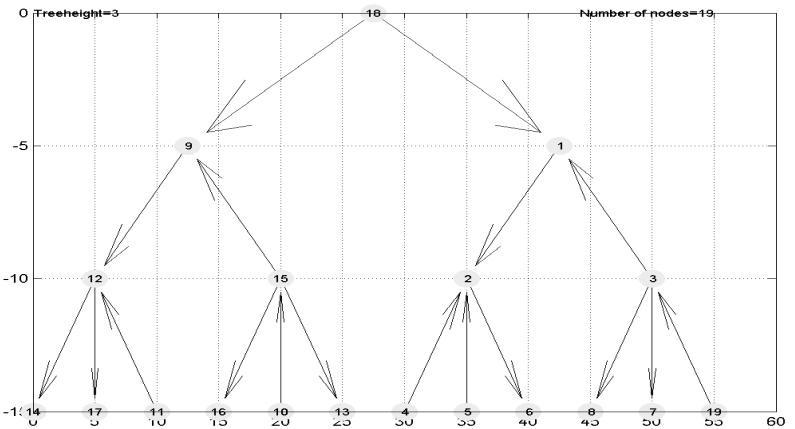

Figure 6: The user can verify the x, y coordinates.

In order to achieve this result, the pre-order traversal is used. In particular, the order of the nodes in each depth level is the same with that in the pre-order traversal. Therefore, a vector including the tree's pre-order, *pre*, is used.

The optional directions of the tree arcs are visualized using the *x* vector. The *Drawtree* function is programmed in such a way that can have two or three inputs, depending on the user's choice. Finally the user can select any time the nodes colour and shape.

5 Conclusions and future work

In this paper, we have presented a visualization tool designed to aid computer science students learn basic Data Structures. This tool not only lets students see

The Internet Society: Advances in Learning, Commerce and Security, K. Morgan & M. J. Spector (Editors)

the visualization of a general, directed or undirected, rooted tree, but also allows them to customize their own trees and make use of the pre-order traversal.

This new tool minimizes the instructor's effort and gives him the possibility of drawing trees easily with many different properties. It can be used efficiently in courses like *Graph Theory* or *Network Optimization* or in an introductory course like *Algorithms and Data Structures*. We believe this tool will be an effective supplement to traditional instruction.

Should we want to find the educational value of our tool; then certain tests must take place inside the classrooms during the lectures, for the needs of our Department's courses. More significant is to find how students make use of this specific software, rather than just explaining to them what might be the potential benefits, Hundhausen *et al* [20] and Lawrence *et al* [21]. Other researches show how to use algorithm animations in learning situations, as for example Kehoe *et al* [22]. This is the best way for measuring the efficiency of any software dealing with Algorithm Visualization, described by Price *et al* [23], as a subclass of software visualization, concerned with illustrating computer algorithms in terms of their high-level operations, usually for the purpose of enhancing computer science students' understanding of the algorithms' procedural behaviour. We shouldn't forget on the contrary, the fact that there have been conducted researches which support that pedagogical advantages can be only partially attributed to Algorithm Visualization Technology, as for example Byrne *et al* [24].

A possible future enhancement for our educational tool is to add options for post-order and in-order permutation and to visualize the Simplex Algorithm and also some exterior point Network Flow Algorithms as for example, Paparrizos [25], where we meet significant tree modifications. This tool would help the student to better follow the steps of these specific type algorithms.

References

[1] Tal A. & Dobkin D. Visualization of Geometric Algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 1(2), pp. 194-204, 1995.

[2] Naps T. L., Incorporating Algorithm Visualization into Educational Theory: A Challenge for the Future, *The European Online Magazine for the IT Professional*, 2(2), 2001.

[3] Duane J. J., www.seas.gwu.edu/~idsv/idsv.html

[4] Mukundan R., www.cosc.canterbury.ac.nz/people/mukundan/dsal/BTree.html

[5] Foley B. J., Designing Visualization Tools for Learning. *Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems,* Los Angeles, pp. 309–310, 1998.

[6] Soloway E., Guzdial M. & Hay K. E., Learner - centered design: the challenge for HCI in the 21st century. *Interactions*, 1(2), pp. 36-48, 1994.

[7] Tamassia R., rw4.cs.uni-sb.de/users/sander/html/gstools.html

The Internet Society: Advances in Learning, Commerce and Security, K. Morgan & M. J. Spector (Editors)

[8] Battista G., Eades P., Tamassia R. & Tollis I. G., *Graph Drawing: Algorithms for the Visualization of Graphs,* Prentice Hall: Upper Saddle River, N.J., 1999.
[9] Goodrich M. T. & Tamassia, R., *Data Structures and Algorithms in JAVA,* Wiley, 1998.
[10] Himsolt M., GraphEd: A graphical platform for the implementation of graph algorithms. *Proc. of the Graph Drawing, DIMACS International Workshop,* pp. 182-193, 1994.
[11] Graphlet, www.infosun.fmi.uni-passau.de/Graphlet
[12] VGJ, www.eng.auburn.edu/department/cse/research/graph_drawing/vgj.html
[13] aiSee, www.absint.com/aisee
[14] Lecland B. & Rety P., Taja 2.0, 2001, www.univ-orleans.fr/SCIENCES/LIFO/Members/lecland/taja.php
[15] Tamassia R., Goodrich M. T., Vismara L., Handy M., Shubina G., Cohen R., Hudson B., Baker R. S., Gelfand N. & Barandes U., JDSL: The data structures library in java, *Dr. Dobb's Journal*, 26(4), pp. 21–31, 2001.
[16] Goodrich M. T., Handy M., Hudson B. & Tamassia R., Accessing the Internal Organization of Data Structures in the JDSL library, *Lecture Notes in Computer Science*, 1619, pp. 124-139, Springer-Verlag Heidelberg: Baltimore, 1999, selected papers from the *International Workshop Algorithm Engineering and Experimentation ALENEX '99*, 1999.
[17] Tom Sawyer Software, www.tomsawyer.com
[18] The MathWorks, Inc., MATLAB: The Language of Technical Computing, Massachussetts, © 1994-2003.
[19] Marchand P. & Holland O., T., *Graphics and GUIs with MATLAB*, 3rd. Edition, CHAPMAN AND HALL/ CRC, 2002.
[20] Hundhausen, D. C., Douglas, A. S. & Stasko, T. J., A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages and Computing*, 13(3), pp. 259-290, 2002.
[21] Lawrence A. W., Badre A. N. & Stasko J. T., Empirically Evaluating the Use of Animations to Teach Algorithms. *Proc. of the IEEE Symposium on Visual Languages*, St. Louis, 1994.
[22] Kehoe C., Stasko J. T. & Taylor A., Rethinking the evaluation of algorithm animations as learning aids: an observational study. *International Journal of Human-Computer Studies*, 54, pp. 265-284, 2001.
[23] Price B. A., Baecker R. M. & Small I. S., A principled taxonomy of software visualization. *Journal of Visual Languages and Computing*, 4, pp. 211–266, 1993.
[24] Byrne M. D., Catrambone R. & Stasko J. T., Evaluating animations as student's aids in learning computer algorithms. *Computers and Education*, 33, pp. 253–278, 1999.
[25] Paparrizos K., An infeasible (exterior point) simplex algorithm for assignment problems. *Mathematical Programming*, 51, pp 45-54, 1991.

The Internet Society: Advances in Learning, Commerce and Security, K. Morgan & M. J. Spector (Editors)