An interactive AVL-Tree Intelligent Tutoring System.

Gautam Yadav
Computer Science & Engineering
IIIT Jabalpur
Jabalpur, India
gautamyadav@iiitdmj.ac.in

Arpit Jain
Computer Science & Engineering
IIIT Jabalpur
Jabalpur, India
arpit.jain@iiitdmj.ac.in

ABSTRACT

This paper describes our new visual understanding based intelligent-state-feedback approach, for teaching the insertion algorithm for the AVL trees. Our aim is to supplement basic class lectures with an interactive intelligent tutoring system that integrates our proposed approach for better understanding. A more extensive explanation of the developed tutor is available here in the demo video link - https://youtu.be/8YIk7LpQTKI

Introduction. It is more probable that a motivated student is a productive learner. Interesting assignments encourage learning by actively involving students in the material. Active learning for students is very essential in an introductory course on data structures where students learn the basics of programming [1]. It can be both challenging and hard to learn for any individual student. Many approaches have been tried but not with very successful implementation outcomes. One excellent suggestion was proposed in [2] to use visual programming for teaching the concept of Binary Trees. AVL (Adelson-Velskii and Landis) trees in Data Structures is one such topic where many students face difficulty in understanding the integral rotations for balancing. Past approaches for teaching AVL tree using an online tutor include resubmissions to restart the exercise with new random data [3], using animations and visualizations to show how the rules are applied [4, 5]. It has been seen that even with the aid of these approaches, students still have difficulty with creating an AVL tree from a series of input values. This is due to the fact that almost all of these approaches lack a proper feedback system that is required for essential persistent learning [6]. Studies have indicated that immediate feedback method actively engages learners in the discovery process and that this engagement promotes correction and the retention of initially inaccurate response strategies [7, 8].

Theory. An AVL tree is a self-balancing binary search tree. The heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property. In AVL tree, after performing operations like insertion and deletion we need to check the balance factor of every node in the tree. If every node satisfies the balance factor condition then we conclude the operation otherwise we must make it balanced. Whenever the tree becomes imbalanced due to any operation we use rotation operations to make the tree balanced. In AVL Tree, a new node is always inserted as a leaf node. A new element is inserted into the tree using Binary Search Tree insertion logic. After this insertion, the balance factor of each of the node is being checked. If the Balance Factor of every node is 0 or 1 or -1 then go for the next operation. otherwise, If the Balance Factor of any node is other than 0 or 1 or -1 then that tree is said to be imbalanced. In this case, a suitable Rotation is performed to make it balanced and then the next operation is being performed. There are essentially 4 kinds of rotation for AVL tree balancing: Single Left (LL), Single Right (RR), Left Right (LR) and Right Left (RL).

Implementation. This Intelligent Tutoring System was developed using Cognitive Tutor Authoring Tools (CTAT) [9]. CTAT is free for research tool developed by Carnegie Melon University for Easy

demonstration and development of Learning tools. This was developed to significantly reduce the amount of time needed to create a virtual tutor. AVL-Tree tutor, which we propose, speaking holistically, is a single page web application which consists of two basic parts. First part is a tutorial section specifying main concepts related to insertion in an AVL tree and the second part is the tutor section which students can use to solve example problems. The Tutor screen consists of a set of numbers that the student is required to fill in the empty tree. This is done by inputting the appropriate value of the nodes by filling in the textboxes inside each node. It imposes ordering constraints on problem-solving with multiple paths leading to the correct answer. When the student makes a mistake at any step, that step (node) is highlighted by red color and the tutor won't move to the next one until it's corrected. When a student requests for a hint, the first hint from the relevant link is displayed into the white hint box. A link can have multiple levels, where each successive request by the student displays the next level of hint. The last or lowest level of hint tells the student where to exactly insert the element in the tree. When all elements are inserted, the student can let the tutor know by clicking on the Done button. CTAT basically utilizes two basic components to model a tutor application – An HTML Interface and A state-behavior graph. Using CTAT we made the behavior graph to model different problems of AVL trees. This detailed graph can be used to control whether or not to show hints and other feedback to the student at any point of time or step. Basically, this graph is a set of states (Incorrect or correct) that the tutor traverses through in order to guide a student through the correct path with proper feedback at each step.

Discussions. For future work, we aim to implement our tutor in the running classroom course of data structures and collect data by taking 2 tests of the enrolled students after completing classroom lectures - one before using this tutor and another after using it for measuring the changes in student scores. The change in the scores of students would be a reflector of the effectiveness of this tutor.

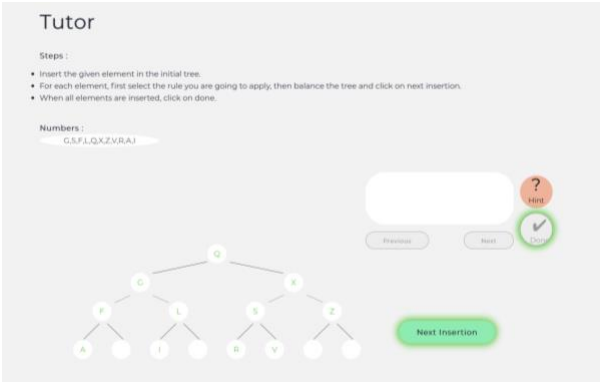

Fig. 1. Interface of AVL Tree tutor showing a sample problem

REFERENCES

[1] Lawrence, R. (2004). Teaching data structures using competitive games. IEEE Transactions on Education, 47(4), 459-466.

[2] Michail, Amir. "Teaching binary tree algorithms through visual programming." In Proceedings 1996 IEEE Symposium on Visual Languages, pp. 38-45. IEEE, 1996.

[3] Malmi, L., & Korhonen, A. (2004, August). Automatic feedback and resubmissions as learning aid. In IEEE International Conference on Advanced Learning Technologies, 2004. Proceedings. (pp. 186-190). IEEE.

[4] Segura, C., Pita, I., del Vado Vírseda, R., Saiz, A. I., & Soler, P. (2008, June). Interactive Learning of Data Structures and Algorithmic Schemes. In International Conference on Computational Science (pp. 800-809). Springer, Berlin, Heidelberg.

[5] Rößling, G., & Schneider, S. (2007). An integrated and "engaging" package for tree animations. Electronic Notes in Theoretical Computer Science, 178, 69-78.

[6] Nicol, David J., and Debra Macfarlane-Dick. "Formative assessment and self-regulated learning: A model and seven principles of good feedback practice." Studies in higher education 31, no. 2 (2006): 199-218.

[7] Epstein, M. L., Lazarus, A. D., Calvano, T. B., Matthews, K. A., Hendel, R. A., Epstein, B. B., & Brosvic, G. M. (2002). Immediate feedback assessment technique promotes learning and corrects inaccurate first responses. The Psychological Record, 52(2), 187-201. Chicago

[8] Epstein, M. L., & Brosvic, G. M. (2002). Students prefer the immediate feedback assessment technique. Psychological reports, 90(3_suppl), 1136-1138.

[9] Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. R. (2006, June). The cognitive tutor authoring tools (CTAT): preliminary evaluation of efficiency gains. In International Conference on Intelligent Tutoring Systems (pp. 61-70). Springer, Berlin, Heidelberg.