Teaching a Complex Process: Insertion in Red Black Trees

C.W. Liew(✉) and F. Xhakaj

Department of Computer Science, Lafayette College, Easton, PA 18042, USA
{liewc,xhakajf}@lafayette.edu

Abstract. Red black trees (and all balanced trees) are an important concept in computer science with many applications. This paper describes a new approach using an example tracing tutor and our experince in using it to teach insertion in red black trees.

Keywords: Computer science · Data structures · Learning process

1 Introduction

Data structures are part of the programming fundamentals and core topics in a computer science curriculum [1]. An example of a data structure with complex algorithms is the red black tree and other balanced tree structures. Learning the red black tree algorithms can be quite challenging and difficult for the students for many reasons. Many approaches to teaching red black trees have been tried [3–5] but our students still have quite a bit of difficulty with the concepts. We have designed a new approach using an example tracing tutor (RedBlackTree Tutor) that allows us to break down the algorithm into small steps explicitly and capture the students decisions while trying to construct a red black tree using a top-down insertion algorithm. This paper describes the tutoring system and our experience in using it to teach insertion in red black trees.

2 Red Black Trees

A red black tree is a self balancing binary search tree data structure, that has the following properties [5]:

1. The nodes of the tree are colored either red or black.
2. The root of the tree is always black.
3. A red node cannot have any red children.
4. Every path from the root to a null link contains the same number of black nodes.

A red black tree must display all of the properties listed above. In addition, every operation performed on a red black tree such as insertion or deletion,

C. Conati et al. (Eds.): AIED 2015, LNAI 9112, pp. 698–701, 2015.
DOI: 10.1007/978-3-319-19773-9_95

should preserve these properties resulting in a changed, but still correct red black tree. The top-down insertion algorithm described in [5] starts at the root of the tree and in a single iterative pass, modifies the tree by applying one or more of the insertion rules described below and eventually adds a new item to the tree. Just as with a binary search tree, the insertion algorithm starts at the root and traverses down to the leaf by comparing the value at the current node with the value to be inserted and selects the next node accordingly (left child if larger, otherwise right child). When a leaf is reached, a new node with the new value is inserted as a child of the node. The difference between the binary search tree and red black tree algorithms is that the red black tree algorithm can apply transformations on the tree at every iteration modifying the connections of the tree but not the values.

The insertion rules are: *color flip*, *single rotation*, *double rotation*, *simple insertion*, and *color root black*. The first rule (color flip) is to minimize the transformations arising from inserting a new leaf node that is colored red. The second (single rotation) and third rules (double rotation) are used to correct any violations (two red nodes in sequence) that arise from either applying rule 1 or by inserting a new node (simple insertion). The rule *color root black* is applied at the end of every insertion operation to ensure that the root of the tree is always black. The rules together ensure that only a single traversal from the root of the tree to a leaf is required to insert a new node and still maintain the red black tree properties.

Whether a rule is applicable at the current node is determined by the color and structural relationships of other nodes in relation to the current node. For example, for a color flip (rule 1) to be applicable the current node (X) must be black and its two children (C1,C2) must be red.

Insertion in red black trees is an example of a complex problem solving process. At every step of the process (each iteration in the algorithm) there are multiple choices and each choice has pre-conditions and they are affected by the decisions made in the previous step. At each iteration, a student must (1) select the current node, (2) identify the context - parent, sibling, grandparent, children and their colors, (3) select the applicable rule, and (4) apply the selected rule. Past approaches to teaching red black trees (or other balanced trees such as the AVL tree) [3–5] and the associated algorithms have treated the decisions at each iteration as a single compound decision. In addition, they have assumed (without data) that the students had difficulty in applying the rules and have focused on the mechanics of applying the rules, specifically the single and double rotation rules.

2.1 A New Approach

In contrast to previous approaches, our approach breaks down the decision point at each iteration into the following steps:

1. *identification of the current node* and its context (color, sibling, children, parent and grandparent) when iteratively traversing the tree and applying the rules,

2. *selection of the rule* to be applied at the current node, and
3. *application of the rule* correctly.

This approach explicitly separates out the identification of the current node and selection of the applicable rule from the the application of the rule itself.

3 The RedBlackTree Tutor

We implemented our approach both in the classroom (lecture) and in laboratory exercises. This section describes how the students use the laboratory exercises. The laboratory exercises were provided through an example tracing tutor (RedBlackTree Tutor) that we developed using the Cognitive Tutor Authoring Tools (CTAT) [2].

The RedBlackTree Tutor imposes ordering restrictions on the problem solving path of the student. The first restriction requires the student to provide the correct answer for the current step before moving to the next step. The second restriction is imposed within a particular step, and requires students to answer the two questions at the top, namely identify the current node and select a rule, before going on to applying the rule. The tutor will not allow the student to work on the application of the rule before completing the identification questions correctly. The order restrictions make the students follow the granularity approach, while separating (1) steps from each other, and (2) the identification and selection parts of the problem from the application part.

We developed exercises to provide practice for the use of our approach to applying the five rules of top-down insertion, namely color flip, single rotation, double rotation, simple insertion and color the root black. The exercises varied in the context for the selection of each rule. For example, we created exercises where students could apply the color flip rule with the current node being (1) the root of the tree, (2) the left child of the root, and finally (3) the right child of the root. Similarly, we designed exercises for the other rules of top-down insertion.

4 Experiment Design

We evaluated the approach on the students in our data structures class during the fall semester of 2014. Students are introduced to red black trees during week 8, after they have covered binary trees and binary search trees.

This is the first balanced tree data structure that they will have seen. There were sixteen students in the class, mostly computer science and computer engineering majors and all the students participated in the study. We only analyzed the results for 12 of the students because 4 students had worked and practiced on additional problems on their own in the period between steps 2 and 3 of the evaluation process described below. Their results were discarded from the overall evaluation because we could not disambiguate between the effects of using the RedBlackTree Tutor and the work that they did on their own. The evaluation process followed the following steps (1) one week of lecture to cover red black

trees including the granularity approach, (2) in the following week, a pre-test of 25 mins followed by, (3) a 1 hr session with the RedBlackTree Tutor, and (4) two days later, a 25 min post-test.

The pre-test and post-test were graded based on the correctness of each part of each step. We broke down each step in the exercises into 3 parts - identification of the current node, selection of the applicable rule, application of the rule - and graded the answers accordingly. We analyzed the scores of the students in (1) identifying the current node at a particular step, (2) selecting the rule, and (3) applying the rule, between the pre-test and post-test. The pre-test data shows that the students had the most difficulty (scoring 10.5 out of 25) in identifying the current node at each iteration even after a week of lectures that included examples and group work on practice problems. They had the least difficulty (score of 16.2 out of 25) in applying the selected rules. If these data are a true indication of student work, they would explain why the past approaches that focused on rule application were unsuccessful. The scores improved from pre-test to post-test from (1) 10.5 to 18.83 for node identification, (2) 14.17 to 21.92 for rule selection, and (3) 16.2 to 20.3 for rule application. Thus the scores for node identification increased by 79.33%, rule selection increased by 54.7% and rule application improved by 25.73%. The average score in the pre-test was 40.8 (out of 75) and it improved to 61.1 in the post-test.

5 Conclusion

This paper has described our approach for teaching red-black trees and its implementation in a tutoring system. The system has been evaluated in a class of 16 students (that is the size of the class in our college) and the results indicate that the approach can help improve the students performance.

References

1. ACM/IEEE-CS Joint Task Force on Computing Curricula. ACM/IEEE Computing Curricula 2001 Final Report (2001). http://www.acm.org/sigcse/cc2001
2. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: A new paradigm for intelligent tutoring systems: Example-tracing tutors. International Journal of Artificial Intelligence in Education **19**(2), 105–154 (2009)
3. Galles, D.: Data structure visualizations. http://www.cs.usfca.edu/galles/visualization
4. Ha, S.: VisuAlgo. http://www.comp.nus.edu.sg/stevenha/visualization
5. Weiss, M.A.: Data Structures & Problem Solving Using Java, 3rd ed. Pearson Education Inc. (2011)