# Assessing Student Answers to Balanced Tree Problems

Chun W. Liew(✉), Huy Nguyen, and Darren J. Norton

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

**Keywords:** Computer science · Data Structures · Learning process

## 1 Introduction

Problems in the domain of balanced binary tree operations usually involve the students constructing a sequence of transformations to insert or delete a value. An Intelligent Tutoring System (ITS) in this area must be able to perform automated assessment of student performance even if there can be multiple correct solution sequences and the input is graphical in nature. Previous works involve either generating all possible solutions and finding the closest match with the student's answer [1] or restricting the student's inputs to one predefined solution [3]. This paper describes a more flexible approach that uses domain knowledge along with a very small restriction on the input method to determine (1) accurately the correctness of the answer and (2) the location and type of the first error in the answer.

## 2 Red-Black Trees

A red-black tree is a self balancing binary search tree that has the following properties [4]:

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.

The top-down algorithm to insert or delete a value from a red-black tree starts at the root and, at every iteration, moves down to the next node, which is a child of the current node. At each node, it applies one or more transformation rules so that when the actual insertion (or deletion) is performed no subsequent actions are needed to maintain the tree's properties. Other types of balanced trees also use a similar approach. In our work we used red-black tree as an exemplar to evaluate our ideas and implementations, but they should be applicable to balanced trees in general.


E. André et al. (Eds.): AIED 2017, LNAI 10331, pp. 532–535, 2017.
DOI: 10.1007/978-3-319-61425-0_57

## 3 Our Approach

An essential constraint in the assessment environment is that the system does not provide any hints about the correctness of the answer or any intermediate states. We developed a web interface that displays a "blank" binary tree of 31 nodes, i.e., the interface looks like a sheet of paper with an outline of a binary tree that is 5 levels deep. The student submits an answer tree by entering the value and color of every non-empty node.

In grading the answer, we take an approach similar to [2], which involves developing a solution module that for any problem will generate a solution in canonical form using only primitive operators. The system also has a list of transformations that can potentially belong to a different solution; these are used to modify the canonical version into any other equivalent form, eliminating the need to generate all possible solutions and instead relying on heuristics to find the closest match to the student's answer. The grading algorithm for both insertion and deletion operations is as follows (Fig. 1).

1. retrieve the problem (the sequence of numbers to be inserted)
2. generate a correct solution which consists of the transformations (and resultant trees) for each number to be inserted
3. set the current subproblem to the first subproblem (insertion of the first number)
4. retrieve the generated solution to the current subproblem
5. retrieve the corresponding answer submitted by the student
6. compare the solution and answer trees for each subproblem. If the last tree in each sequence matches, the subproblem has been solved correctly by the student. Otherwise, the student is assigned a partial score based on what has been correctly solved to this point (including the subproblems prior to the current one)

**Fig. 1.** Grading algorithm for insertion

Once the assessment module has detected that the final state in the student's answer is different and therefore incorrect, it goes on to determine (1) where the first error occurred and (2) the type of error made. Figure 2 describes this procedure. Currently the algorithm can detect insertion errors in *color flip, single rotate, double rotate* and *insert*. It can also detect deletion errors in *drop and rotate, single rotate, double rotate, recolor root* and *delete*.

## 4 Evaluation

The system was evaluated on 30 students in a Data Structures class in Fall 2016, with 120 answers recorded. The evaluation had the following steps:

1. Week 1 - lectures on the material (2.5 h)
2. Week 2, day 1 - pre-test (0.5 h)
3. Week 2, day 1 - use of tutoring system (1 h)
4. Week 2, day 2 - post-test (0.5 h)

1. retrieve the current subproblem (insertion or deletion of a value)
2. generate the sequence of solution trees for the current subproblem
3. retrieve the sequence of trees submitted by the student
4. set the current tree to the first tree in each sequence (generated and submitted)
5. compare the current tree from the student with the trees in the generated sequence
6. if there is a match, set the current trees to the next tree after the trees match and repeat with step 5 until all the trees in the student's sequence have been compared.
7. otherwise, an incorrect transformation was applied. Terminate the algorithm and return the type of transformation that was attempted.

**Fig. 2.** Finding The Error In The Tree: starting with the first tree in the student's answer, compare with the first tree in the generated solution. Repeat until there is a tree in the student's answer that does not map to the generated trees even with transformations. Use heuristics to try and categorize the error.

The pre and post tests were identical and contained 4 questions, each composed of the insertion (deletion) of 4 to 6 numbers. The students used a web browser to take the pre and post test using the assessment interface described in the previous section.

Tables 1 and 2 show a breakdown of the errors made for insertion and deletion problems. Our algorithm could effectively identify the first error 78% of the time in insertion problems and 63% in deletion problems. The portion of unrecognized errors were due to either the students combining multiple steps or performing completely incorrect transformations.

The system performs comparably to a human grader and can effectively recognize most single errors. Currently it is unable to detect the combination of more than one errors or assign partial credit if one step is incorrect but subsequent steps are correct based on the resulting tree. Furthermore, we encountered difficulty in determining whether the errors were in identifying the current node or the applicable transformation. These are important features to be added in the next iteration.

**Table 1.** Distribution of errors made in insertion - pre and post test

| | Color flip | Single rotate | Double rotate | Insert | Unrecognized |
|---|---|---|---|---|---|
| Pre test | 54.9% | 3.7% | 8.5% | 10.9% | 22.0% |
| Post test | 29.4% | 8.6% | 27.6% | 12.0% | 22.4% |

**Table 2.** Distribution of errors made in deletion - pre and post test

| | Drop rotate | Single rotate | Double rotate | Recolor root | Delete | Unrecognized |
|---|---|---|---|---|---|---|
| Pre test | 30.7% | 14.5% | 6.5% | 1.6% | 9.7% | 37% |
| Post test | 45.7% | 8.7% | 4.3% | 0% | 2.2% | 39.1% |

## 5 Conclusion

This paper has described our approach to automated assessment of graphically inputted answers to red black tree insertion and deletion problems. The approach has been successful in distinguishing between correct and incorrect answers and also in generally identifying the location and type of error. There are still additional features that we intend to implement that would help us to develop a more effective tutoring system that is customized to the individual student.

## References

1. Liew, C., Shapiro, J.A., Smith, D.: Determining the dimensions of variables in physics algebraic equations. Int. J. Artif. Intell. Tools **14**(1&2), 25–42 (2005)
2. Liew, C., Shapiro, J.A., Smith, D., McCall, R.J.: Understanding student answers in introductory physics courses. In: Luckin, R., Koedinger, K.R., Greer, J. (eds.) Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work, July 2007
3. Liew, C.W., Xhakaj, F.: Teaching a complex process: insertion in red black trees. In: Conati, C., Heffernan, N., Mitrovic, A., Verdejo, M.F. (eds.) AIED 2015. LNCS, vol. 9112, pp. 698–701. Springer, Cham (2015). doi:10.1007/978-3-319-19773-9_95
4. Weiss, M.A.: Data Structures & Problem Solving Using Java, 3rd edn. Pearson Education Inc., Boston (2011)