Abstract

Games perform a certain number of educational purposes. In this respect, they aim to motivate students to learn more effectively and encourage them to learn from their mistakes. Subjects of Data Structure and Algorithms course offered to computer science students are very virtual and are very difficult to interpret visually. In this study, we develop a computer game for teaching of basic operations (construction, insertion, deletion) of binary search trees (BST) including rotation operation to convert unbalanced BST to AVL tree. BST operations and rotation are mostly complex concepts encountered by IT students. We test effectiveness of the proposed system by an examination. An examination related to BST operations and rotation is applied to two groups of students in Kocaeli University Computer Engineering Department: one group takes classical education and the other group learns the subject by means of computer games. The findings show that the second group learns BST operations and rotation better than the first group.

Keywords: Binary search tree, data structure, computer education with games, game-based learning, educational computer game;

1. Introduction

Data Structures and Algorithms are essential topics in computer science education. These topics are very difficult to understand and interpret virtually by students. Also, as learning data structures and algorithms is generally carried out based on coding in programming environment and they are taught
via classical education, students scarcely ever find it interesting. Games perform a certain number of educational purposes. In this respect, they aim to motivate students to learn more effectively and encourage them to learn from their mistakes. Preferring games instead of other teaching methods is an innate behaviour of human beings. A baby prefers his/her meal with a game. If the mum plays a game with the food, the baby eats. Similarly, students learn easier and learn in a short period of time.

In the literature, many researchers have been worked on teaching data structures visually and interactively. Baker and et al. [1] focused on techniques to visualize and test student written data structures. Lawrence [2] described a project for a data structures course based on the idea of competitive programming motivating student learning by allowing students to evaluate and improve their programs. Matzko and Davis [3] presented an approach to teaching data structure concepts using advanced graphics algorithms. Budd [4] purposed to teach data structures through graphics-based approach and Biernat [5] via active teaching tools. Tan and Kim Seng [6] presented a case study on using a computer game for teaching data structures on stacks and queues. Amoroso and Marfia [7] discussed to teach distributed system principles via distributed games. Karapınar et al. aimed to make pointers be tested and see some computer reactions visually in [8] and also, they aimed to make tree traversal algorithms learn easier and permanently in [9].

BST operations (construction, insertion, and deletion) and rotation operation to convert unbalanced BST to AVL tree are mostly complex concepts encountered by IT students. In this paper, a game-based approach is developed to provide a helpful method for studying basic operations which are performed on BST and AVL trees. And an application is developed in this context. By this application, students which studied theoretically could be putted into a visual practice environment that consolidates their knowledge. This application can also be used by teachers as an education material in the classroom. The operations that examined in application are adding a node, deleting a node, rotation and balancing.

The remainder of this paper is organized as follows. In Section 2, BST operations, rotation and balancing are explained. In Section 3, methodology of the study is presented. Section 5 draws a conclusion.

2. Computer Game Approach to BST and AVL Operations

A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. According to the algorithm, data need to be adjusted data structures such as arrays, stacks, queues, trees and graphs. Different kinds of data structures are suited to different kinds of applications. In tree data structures, data held in a hierarchical structure composed of nodes and branches. Parts of the data or themselves are kept at nodes and branches indicate connection relationships between other nodes. Due to the nature of tree data structure, it is suitable solution or modelling of a lot of problem. For this reason, there are many varieties of tree. BST data structures is one of the most widely used varieties of trees. BST makes data kept in memory easily searchable. The search time becomes O(logn) for an n-node BST. The most important aspect of BST is searching on it. BST has following conditions with the properties of tree data structure:

- Each node must have at most two child nodes. (Both the left and right sub-trees must also be binary search trees.)
- The left sub-tree of a node contains only nodes with keys less than the node’s key.
- The right sub-tree of a node contains only nodes with keys greater than the node’s key.
- There must be no duplicate nodes.
2.1. BST Operations

Operations on BST can be listed as following:

- Construction: building a tree.
- Destroy: deleting all items in the tree.
- Empty: checking the tree is empty or not. It returns true if the tree is empty, otherwise false.
- Add or insert: inserting a new node into the tree.
- Delete: deleting a node from a tree.
- Traversal: traversing, accessing, and processing an item in the tree.
- Search: searching an item in the tree.

In this study, the operations that examined for BST are adding a node, deleting a node. Inserting a new node into a BST need to follow the left or right branch to down the tree until null sub-tree is found. There are four possible cases when we delete a node: (i) the node (leave node) to be deleted has no children. (ii) The node to be deleted has only a right sub-tree. (iii) The node to be deleted has only a left sub-tree. (iv) The node to be deleted has two sub-trees.

2.2. AVL Tree and Rotation Operation

AVL tree (height-balanced BST) is a search tree in which the heights of the sub-trees differ by no more than one. Let \( H_L \) represent the heights of left sub-trees and \( H_R \) represent the heights of right sub-trees. Absolute value of AVL sub-trees height must equal or less than 1 \(| H_L - H_R | \leq 1\). Rotation is a transformation process to convert unbalanced binary search tree to AVL tree. Rotation is a transformation process to convert unbalanced binary search tree to AVL tree. Unbalanced BST falls into one of four cases:

- Left of left (LL): A sub-tree of a tree that is left high has also become left high.
- Right of right (RR): A sub-tree of a tree that is right high has also become right high.
- Right of left (RL): A sub-tree of a tree that is left high has become right high.
- Left of right (LR): A sub-tree of a tree that is right high has become left high.

2.3. Game Information

2.3.1. Development platform

Developed game is a web application running on the client side. The game can be operated as desktop application by copying to any computer or can be accessed via the internet by installing to a server. We have prepared our game with HTML5 and oCanvas that is open source JavaScript library [11]. The library enables working with objects instead of working with pixels. Figure 1 shows interface of our game.
2.3.2. How to play

When the user opens the game, the main menu showing BST basic operations and rotation operation is displayed. The user chooses one of the nine operation sections and starts playing. When any section is selected, one of the trees of the selected section is displayed. We consider status of AVL balancing (LL-RR case) as an example (see Figure 2). The operation/operations expected to perform by the user is shown at the bottom of screen. Here, the operations are “Add the given node 71 and balance the tree”. The node is to be added or deleted is placed on operation explanation. User throws the nodes to basket in deleting operation. The user can work freely on the tree. The tree can be taken shape in desired form by dragging or dropping nodes. Spots appearing on the screen help the user for placing nodes. When the user finishes the operation, he/she pushes “OK” button to check if the process is correct or not. Also, the user can view the correct result of the operation by clicking “Solution” button. If the user wants to initiate the game at anytime, the user can click “Reset” button. If the user wants to play for different tree of same operation, the user can click “New” button.
3. Methodology

In this section, we present the methodology of our study for learning BST basic operations and rotation operation.

3.1. Sample Space

We test effectiveness of the proposed system by an examination. An examination related to BST operations and rotation is applied to two groups of students in Kocaeli University Computer Engineering Department: one group takes classical education and the other group learns the subject by means of computer prepared game.

3.2. Procedure

BST and AVL education is given to two groups over a period of two weeks. In the classical education, just verbal and written content is served to the students. After a brief explanation about BST operations and rotation operation, prepared game is given to second group to play and learn the topic. We test effectiveness of the proposed system by an examination. An examination related to
BST operations and rotation. We compare the learning performances of them. Some of exam questions are the following:

- Considering the below BST, add or delete node ‘X’.
- Perform LL rotation for the given node and balance tree.

4. Results and Future Works

At the end of examination results, the findings show that the second group learns BST operations and rotation better than the first group. Average grade is about 60 for first group; average grade is about 87 for second group. So, learning BST operations and rotation operation with game-based leaning is very helpful for students and timesaving method for both students and teachers. In the future, we will visualize other data structures such as arrays, linked lists, stacks, queues, heaps, and sets.

References