Memo

Date: 02 June, 2017

To: Ms. Karrin Peterson, Professor of English at Bellevue College

From: Mark Willisford, student in the iBit Department, Information Systems, Bellevue College

Subject: Recommendation Report for Including Self-Balancing Binary Search Trees in the curriculum of Programming 260

Attached is my report for my study: “A Research Project for the inclusion of Automatically Balancing Binary Search Trees at Community and Technical Colleges.”

The tasks described in the proposal dated 22 May, 2017, have been completed: I researched the benefits of adding self-balancing binary search trees versus the effects of continuing to not include more advanced data structures in the curriculum. My research also included examining the cost in time, student stress levels, as well as a mathematical analysis of both types of trees. Finally I took a look at the jobs these students are getting and the use of these trees in the tech world.

To perform these tasks, secondary and primary research was performed. Technical publications from public and educational institutions were studied. Two interviews with Quality Assurance Specialists were performed, as well as a series of student surveys. Then, the data was collected and analyzed and a report was written.

The results of this research indicate that despite what colleges promise their students, companies want employees with a solid educational or experienced background for software development positions. Because of this, adding more complex data structures to the degree may not be as helpful as this paper would make it seem. One can easily see why companies want a stronger background for their employees. The students interviewed taking classes at the 200 level at Bellevue College struggle with the concepts of a simple, unbalanced binary tree. Without a major restructuring of our curriculums, we would be setting our students up to fail.

On the basis of these findings, I recommend that Bellevue College not take any action in regards to the addition of self-balancing binary search trees to the two-year curriculum. While students will be less prepared for the tech world, both in tools and understanding, this level of understanding does not fall within the goals of the two-year programs here at Bellevue College.

In the following sections, I provide additional details regarding the research methods, the results obtained, the conclusions I drew from those results, and my recommendation.

I appreciate the trust you have shown in allowing me to produce this recommendation report, and I look forward to working with you on any follow-up activities. If you have any questions or comment, please contact Mark Willisford at markwillisford@hotmail.com or at (425)-890-7855.

**A Research Project for the inclusion of Automatically Balancing Binary Search Trees at Community and Technical Colleges:**

**A Recommendation Report**

Prepared for: Ms. Karrin Peterson, Professor of English at Bellevue College

Prepared by: Mark Willisford, student in the iBit Department, Information Systems, Bellevue College

30 May, 2017

# Abstract

“A Research Project for the inclusion of Automatically Balancing Binary Search Trees at Community and Technical Colleges:

A Recommendation Report”

Prepared by: Mark Willisford, student in the iBit Department, Information Systems, Bellevue College

On 16May, 2017, Ms. Karrin Peterson, Professor of English at Bellevue College, requested a proposal for a research project to study the feasibility of adding self-balancing binary search trees to the curriculum of Programming 260, Advanced Topics in Object-Oriented Programming. Currently, students study a number of basic data structures including stacks, queues and linked lists before being introduced to techniques used to understand the final few data structures. Unbalanced binary search trees, hash tables and finally the heap round out the tools given to students as they enter the work force. This is worth addressing because unbalanced binary search trees are of questionable value as a programming tool and the ease with which it appears a balanced version could be taught. The author researched the benefits of adding self-balancing binary search trees, versus the effects of continuing to not including more advanced data structures in the curriculum. The research also included examining the cost in time, student stress levels, as well as a mathematical analysis of both types of trees. Finally, the author examined the jobs these students are getting, and the use of these trees in the tech world. Most results, and the conclusions are based off of comparisons drawn from data, or from primary source information. The recommendation is that Bellevue College not take any action in regards to the addition of self-balancing binary search trees to the two-year curriculum. While students will be less prepared for the tech world, both in tools and understanding, this level of understanding does not fall within the goals of the two-year programs at Bellevue College.

Keywords: binary search trees, Prog260, preparation for the real world,

Contents

[Abstract 4](#_Toc484547427)

[Executive Summery 6](#_Toc484547428)

[Introduction 7](#_Toc484547429)

[Research Methods 9](#_Toc484547430)

[Task 1: Research and define what BSTs are 9](#_Toc484547431)

[Task 2: Define the problems with unbalanced BSTs 9](#_Toc484547432)

[Task 3: Define a self-balancing BST 10](#_Toc484547433)

[Task 4: Explain and define the current course work in Prog 260 10](#_Toc484547434)

[Task 5: Research if students in the class would understand the concept 10](#_Toc484547435)

[Task 6: Analyze the data and write a recommendation report 11](#_Toc484547436)

[Results 11](#_Toc484547437)

[Task 1: Research and define what BSTs are 11](#_Toc484547438)

[Task 2: Define the problems with unbalanced BSTs 11](#_Toc484547439)

[Task 3: Define a self-balancing BST 13](#_Toc484547440)

[Task 4: Explain and define the current course work in Prog 260 14](#_Toc484547441)

[Task 5: Research if students in the class would understand the concept 14](#_Toc484547442)

[Conclusions 15](#_Toc484547443)

[Conclusions for the addition of a self-balancing BST into the curriculum 15](#_Toc484547444)

[Conclusions against the inclusion of a self-balancing BST into the curriculum 15](#_Toc484547445)

[Recommendation 16](#_Toc484547446)

[References 17](#_Toc484547447)

# Executive Summery

On 16May, 2017, Ms. Karrin Peterson, Professor of English at Bellevue College, requested a proposal for a research project to study the feasibility of adding self-balancing binary search trees to the curriculum of Programming 260, Advanced Topics in Object-Oriented Programming.

The course is one of the primary capstone courses in the Associate in Applied Science – T Degree offered at Bellevue College. It is a popular choice of Worker Retraining students who are given a very short amount of time to reenter the world as a productive member of the work force, due, in large part, to the promise of a “good job” upon graduation.

Currently, students study a number of basic data structures including stacks, queues and linked lists, before being introduced to techniques used to understand the final few data structures. Unbalanced binary search trees, hash tables and the heap round out the tools given to students as they enter the work force.

This is worth addressing because unbalanced binary search trees are of questionable value as a programming tool, and the ease with which it appears a balanced version could be taught to some students.

The tasks described in the proposal dated 22 May, 2017, have been completed: I researched the benefits of adding self-balancing binary search trees, versus the effects of continuing to not include more advanced data structures in the curriculum. My research also included examining the cost in time, student stress levels, as well as a mathematical analysis of both types of trees. Finally I took a look at the jobs these students are getting and the use of these trees in the tech world.

Based on these findings, I recommend that Bellevue College not take any action in regards to the addition of self-balancing binary search trees to the two-year curriculum. While students will be less prepared for the tech world, both in tools and understanding, this level of understanding does not fall within the goals of the two-year programs here at Bellevue College.

# Introduction

On 16May, 2017, Ms. Karrin Peterson, Professor of English at Bellevue College, requested a proposal for a research project to study the feasibility of adding self-balancing binary search trees to the curriculum of Programming 260, Advanced Topics in Object-Oriented Programming.

The course is one of the primary capstone courses in the Associate in Applied Science – T Degree offered at Bellevue College, and a popular choice of Worker Retraining students who are given a very short amount of time to reenter the world as a productive member of the work force.

Currently, students study a number of basic data structures including stacks, queues and linked lists before being introduced to techniques used to understand the final few data structures. Unbalanced binary search trees, hash tables and finally the heap round out the tools given to students as they enter the work force.

This is worth addressing because according to the Bellevue College Institute for Business & Information Technology’s (iBit) web site, “The Information Systems degree(s) includes concentrations for students interested in software development or business analyst. The degree prepares graduates for entry-level developer/analyst positions . . . “ (Bellevue College, iBit, 2017). Unfortunately BSTs do very little to help prepare the students for any real world position.

Pure BSTs are known throughout the industry as simply a teaching tool, nothing more than a step in understanding on the way to useful data structures. (Stack overflow, 2017). Prog260 is taken during the final quarter of the degree program leaving no time to take further steps in understanding to data structures that are actually used. This leaves students believing that they have useful, marketable information and skills without preparing them for a career in software development.

I wish to thank Mrs. Peterson, Instructor, English Department at Bellevue College, for her research assistance. I also wish to thank my peers for their reviewing of the draft report.

For this project I determined to perform these tasks:

* Task 1: Research what BSTs are
* Task 2: Define the problems with unbalanced BSTs
* Task 3 Define a self-balancing BST
* Task 4: Explain and define the current course work in Prog 260
* Task 5: Research if students in the class would understand the concept.

Entering the competitive job market of programmers with only an associate’s degree sets students at a disadvantage. It is our task, as a college, to reduce that disadvantage as much as possible. One way to do this is to ensure that we equip our students with industry leading practical knowledge and experience.

# Research Methods

To perform the requested analysis, I broke the project into five tasks:

1. Research and define what BSTs are
2. Define the problems with unbalanced BSTs
3. Define a self-balancing BST
4. Explain and define the current course work in Prog260
5. Research if students in the class would understand the concept
6. Analyze the data and write a recommendation report

In the following discussion of how I performed each task, I explain the reasoning and methodology that guided my research. For tasks 1, 2, and 5, research assistance was provided by Ms. Peterson, Professor, English Department, Bellevue College, Bellevue, WA.

## Task 1: Research and define what BSTs are

I sought to define what the common definition of a BST is, in regards to Computer Science. To do this, I performed secondary research on respected technical websites. I also sought out, and requested, an interview with a few respected computer scientists in the Bellevue area.

I began with an internet search using the search phrase "Binary search tree" with no domain restrictions. From this search, I found an article hosted on the Princeton University website (Princeton University, 2017), which explains in a concise clear way, exactly what a BST is, the general operations used on a BST, and the rules of creation. A second search using the same search term led me to geeksforgeeks.org, an online community for interview questions, algorithms, and programming advice led by a team of accomplished developers.

Unfortunately, I was unable to secure an interview with any computer scientists familiar with the use of and implementation of BSTs

## Task 2: Define the problems with unbalanced BSTs

For this task, I answered the following questions:

* How do we use binary search trees in the “real world?”
* What problems do these uses lead to?

To answer these questions I performed secondary research on the internet. I interviewed both a Quality Assurance Specialist here at Bellevue College and a Quality Assurance Specialist at Microsoft as part of my primary research. Finally, I performed basic mathematical analysis of both balanced and unbalanced BSTs.

An internet search using the phrase “uses of binary search trees in the business world” led me to a discussion on Stackoverflow.com, a leading source for programming professionals. The discussion stated that unbalanced search trees are good for little other than teaching students the fundamentals of computer science (Stackoverflow.com, 2017).

Using basic calculus and Big O notation, I was able to easily determine the worst case run times for both balanced, and unbalanced, binary search trees.

## Task 3: Define a self-balancing BST

I searched the internet using the phrase “Self-balancing search tree.” This search led me to two articles which were both very helpful in explaining self-balancing BSTs.

## Task 4: Explain and define the current course work in Prog 260

To accomplish this task, I referenced the course syllabus given to students and available with an account on the Bellevue College Canvas Learning Management System.

## Task 5: Research if students in the class would understand the concept

I met with four current and previous students of Prog 260 and discussed the following topics with each.

* How they felt their previous classes had prepared them for the topics taught in Prog 260
* The pace of the class
* Their understanding of the concepts taught
* Their interest in a more in-depth coverage of those concepts
* How they felt the topics prepared them for jobs in the Information Systems field

## Task 6: Analyze the data and write a recommendation report

After analysis, I drafted the recommendation report. The report was peer reviewed by my fellow interns. I included most of my peer’s suggestions into the final version of the report.

# Results

In this section, I present the results of my research. For each of the tasks I carried out, I present the most important data I acquired.

## Task 1: Research and define what BSTs are

A binary search tree (BST) is defined in Computer Science as a node-based data structure with the following properties:

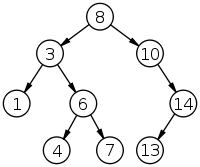
* Each node may have up to two ‘child’ nodes
* The left subtree of a node contains only nodes with keys less than the node’s key
* The right subtree of a node contains only nodes with keys more than the node’s key
* Each subtree must also be a binary search tree
* There can be no duplicate nodes

This structure allows a search function to quickly find any given node by comparing the key of the desired node to the key of the node at the top of the tree, called the root node. If the desired node has a larger key, continue looking in the right subtree. If the desired node has a smaller key, continue looking down the left subtree. Insertion of a new node follows a similar pattern to locate the appropriate location for the new node.

## Task 2: Define the problems with unbalanced BSTs

BSTs were designed to decrease the time required to sort and retrieve data. Unlike a simple list, sorting the data is done when the data is added using a simple comparison method called IComparable. The insert function compares the key of the new data to the key of the root, then continues down the tree until an open slot is found.

For example, consider the following diagram.

In figure 1, we can see that if we wanted to insert a 12, we would start by comparing it to the root node, the 8. Following the rules discussed above, we next compare it to the child node on the right, the 10. We continue traversing the tree until we compare our node to the 13. 12 is smaller than 13 so we need to insert our new node as the left child node of the 13.

This example shows that the insert function requires four comparison operations and the insert operation. Thus we can say that the running time of algorithms on binary search trees is directly based on the height of the tree. The height of a tree is, in turn, based directly on the order in which the data is entered.

Figure : A basic binary tree

Consider a simple tree of seven nodes { “A” “C” “E” “H” “R” “S” “X” }. If we enter the “A” first, all following nodes will be greater than our root, thus extending the right subtree and leaving the left subtree empty. If we instead enter the “H” first, the left and right subtrees will both contain three nodes. In a worst case scenario, our little tree is stored in a seven layer tree, taking seven operations to find the “X”. In a best case scenario, the tree needs three layers, taking only three operations to find the same node, equating to a significant percentage of time saved.

BSTs are used in many different types of applications, including search applications where data is changing often, and nearly all high-bandwidth routers for storing the routing tables that keep the internet connected. The larger these trees get, the larger the difference between best and worst case run times.

Big O notation is used in Computer Science to measure the performance of an algorithm. It assumes a worst case scenario to give an accurate description of how much time is required to perform an operation. For example, O(N) describes an algorithm whose runtime will grow in direct proportion to the size of the dataset used.

An unbalanced BST is an O(N) algorithm. If there are seven nodes in the dataset, we must allow time for seven operations. If there are seven million nodes in the dataset, we must allow time for seven *million* operations.

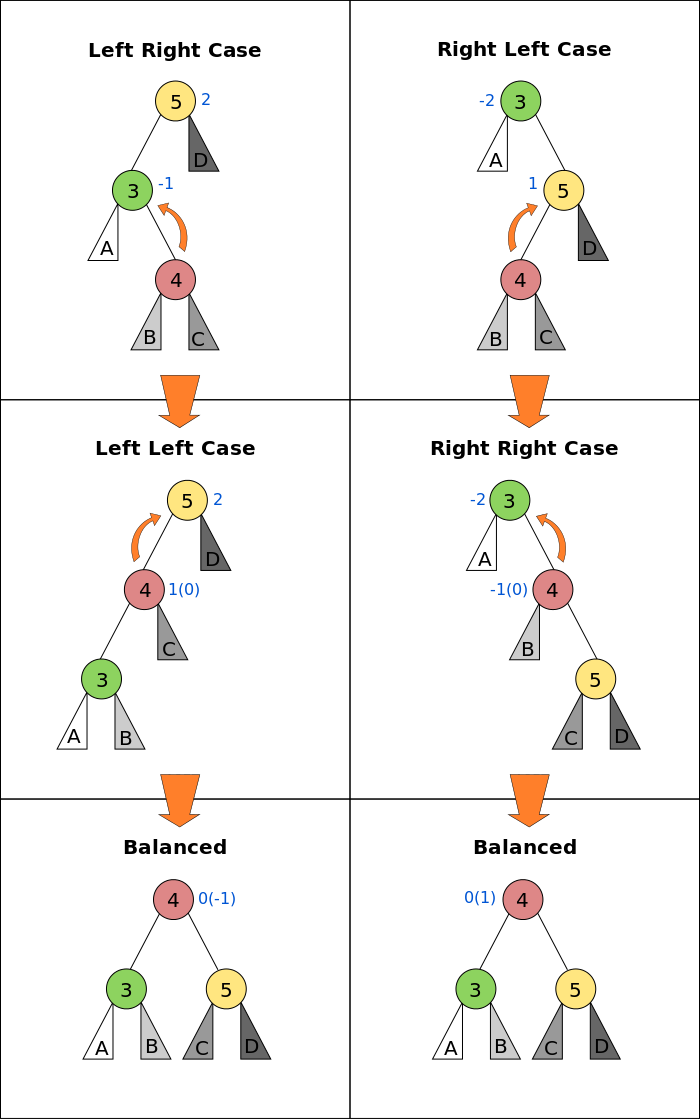
Even with an operation time of one millisecond, this search would take almost two hours.

A fully balanced BST is an O(log N) algorithm. In Computer science, we assume log2 is being used. Using basic calculus, we find that at any given height, there can be at most 2^h nodes where h equals height. Thus, given n nodes, the height of a tree is found by n = 2^( h + 1 ) -1. Simplifying the equation and solving for h with n equal to 7,000,000 gives us log2( 7000000 + 1 ) - 1 = 22.

After balancing the tree, our search takes 22 milliseconds.

Unbalanced binary search trees are not used in the tech industry, unless it is known that the dataset will be extremely small, in which case a simple list or array is generally used instead.

## Task 3: Define a self-balancing BST

A self-balancing BST is any node-based binary tree that automatically reorganizes its nodes to keep the minimum layers possible. These small transformations on the trees, known as tree rotations, do increase the time required for insertions and deletions, however, in the long run, they save time by greatly decreasing runtime of later operations.

There are several different types of self-balancing BSTs in use today. An AVL tree, for example, tracks a balance factor for each node. A balance factor is defined as the height of the left subtree minus the height of the right subtree. If at any point, that balance factor reaches either two, or negative two, a rebalancing function is called.

There are four types of rebalancing functions. Figure 2 illustrates the four possible situations that can arise. The simplest form is the Left-Left case. It is easily seen, that when “A” or “B” are added to the tree, an imbalance occurs in node 5. The left subtree containing “4,” “3,” “A,” “B,” and “C” has three layers, while the right subtree only contains one, “D.” Node 4 is then raised and node 5 is lowered and the tree becomes balanced. Note how the right child of Node 4 becomes the left child of node 5. This happens because “C” is larger than the data in 4 and less than the data in 5.

Figure : courtesy of Hackerrank.com

The Right-Right case is simply a mirror of the Left-Left. As you can see, each of these four functions is simply a slight reordering of the nodes within our tree.

## Task 4: Explain and define the current course work in Prog 260

The Programming 260: Advanced Topics in Object-Oriented Programming course syllabus states that it follows a logical chain of topics beginning with simple stacks and queues. These are very basic data structures where data is stored in a list. Data is removed for use at the top of the list and entered at the top and bottom respectively. The next data structure students learn is linked lists. These are special lists where each node holds the name of the next node in the list. Linked lists are very useful structures when the data being held isn’t necessarily all of the same types and doesn't need to be searched quickly.

The next two topics covered are Recursion and Big O notation. Recursion is simply functions that call themselves and we have already discussed Big O notation.

Once students have an understanding of these topics, unbalanced BSTs are introduced. Two weeks are spent on BSTs before moving on to HashTables, The Heap, and finally, sorting methods.

## Task 5: Research if students in the class would understand the concept

According to the students that I spoke with, they universally felt prepared for the curriculum in Prog 260 and found that it spend ample time discussing each topic until Recursion was introduced. Of the four students, two felt rushed to learn recursion, and three stated they didn’t feel as if they understood BSTs enough to talk about them. The final student was able to talk about BSTs, but did not present a strong understanding of the concepts and methods used to build and operate one. None of the four were interested in delving deeper into data structure concepts unless a specific job required them to do so.

These same four students did believe that they were prepared to use data structures in their future jobs and felt as if their understanding was appropriate for the job market. Only one of the four admitted that it was difficult to “know what he didn’t know.”

Interestingly, all four students indicated that there was no homework assigned regarding BSTs. The only work that required knowledge of these data structures was a project at the end of the quarter.

# Conclusions

## Conclusions for the addition of a self-balancing BST into the curriculum

Self-balancing binary search trees are far more efficient in both time and operations performed than a regular BST. As a result, an unbalanced BST is rarely used in the business world. In fact, they are simply used as an interview question to gauge the understanding of data structures of the potential employee.

Our goal as a college should be to prepare the students in the best way possible for the challenges and tasks of working in the tech world. This includes equipping these students with the tools they will need to do the jobs that the world requires.

## Conclusions against the inclusion of a self-balancing BST into the curriculum

While outside the scope of this document, it is an unfortunate fact that the majority of students with a two-year degree in Information Systems will not, in fact, get jobs as developers. Despite what colleges promise their students, companies want employees with a solid educational or experienced background for these positions. Because of this, adding more complex data structures to the degree may not be as helpful as this paper would make it seem.

One can easily see why companies want a stronger background for their employees. The students I spoke with, taking classes at the 200 level at Bellevue College, all struggle with the concepts of a simple, unbalanced binary tree. Without a major restructuring of our curriculums, we would be setting our students up to fail.

# Recommendation

It is with surprise, that I recommend that Bellevue College not take any action in regards to the addition of self-balancing binary search trees to the two-year curriculum. While students will be less prepared for the tech world, both in tools and understanding, this level of understanding does not fall within the goals of the two-year programs here at Bellevue College.

# References

**Binary Search Tree | Set 1 (Search and Insertion). (2017, May 27). Retrieved May 29, 2017, from** [**http://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/**](http://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/)

**Basic implementation. (n.d.). Retrieved May 28, 2017, from** [**http://algs4.cs.princeton.edu/32bst/**](http://algs4.cs.princeton.edu/32bst/)

**Stoimen's web log. (2012, July 3). Retrieved May 28, 2017, from** [**http://www.stoimen.com/blog/2012/07/03/computer-algorithms-balancing-a-binary-search-tree/**](http://www.stoimen.com/blog/2012/07/03/computer-algorithms-balancing-a-binary-search-tree/)

**Solve Self Balancing Tree. (n.d.). Retrieved May 28, 2017, from** [**https://www.hackerrank.com/challenges/self-balancing-tree**](https://www.hackerrank.com/challenges/self-balancing-tree)

**What are the applications of binary trees? (n.d.). Retrieved May 28, 2017, from** [**https://stackoverflow.com/questions/2130416/what-are-the-applications-of-binary-trees**](https://stackoverflow.com/questions/2130416/what-are-the-applications-of-binary-trees)

**Bellevue College. iBit (2017). *PROG260: Adv. topics in object-oriented programming [Course syllabus]*. Retrieved May 21, 2017, from** [**https://bc.instructure.com/courses/1423675/files/folder/01-Reference?preview=79572661**](https://bc.instructure.com/courses/1423675/files/folder/01-Reference?preview=79572661)

**Information Systems. (n.d.). Retrieved May 21, 2017, from** [**http://www.bellevuecollege.edu/ibit/programs/information-systems/**](http://www.bellevuecollege.edu/ibit/programs/information-systems/)