Nottingham Trent University

School of Science and Technology

Domain Specific Language For Balancing Binary Search Trees

by

Joseph Woodward

in

2018

Project report in part fulfilment

of the requirements for the degree of

Bachelor of Science with Honours

in

Software Engineering

I hereby declare that I am the sole author of this report. I authorize the Nottingham Trent University to lend this report to other institutions or individuals for the purpose of scholarly research.

I also authorize the Nottingham Trent University to reproduce this report by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

Joseph L. Woodward

Abstract

A Binary Search Tree (BST) is an effective data storage technique used within programming languages, allowing one to store, organize and retrieve data in a—potentially—performant way.

There is, however, one bottleneck that may nullify the performance of BSTs, and that is the inefficient, non-hierarchical organization of the data stored within the BST itself: if data is stored linearly, then retrieving data from the BST will be linear, providing no positive usage over many other inherently linear structures. To combat this, one may use a BST Balancing Algorithm, which is used to ensure that all data is stored in a non-linear, hierarchical structure, thus increasing the efficiency of the retrieval of said data.

This work introduces the general topic of BSTs, including how data is stored and retrieved within them, how balancing algorithms can be used to maximize the efficiency of the structure itself. The primary bulk of this work, however, will be introducing a new Domain Specific Language (DSL), a DSL that will be used to allow one to easily experiment with one’s own design and implement of balancing algorithms, providing the capability to benchmark said algorithms with performant and common algorithms used within both the industry and within academia today.

No DSL currently exists for this specific purpose and, although the DSL discussed here has not fully integrated the capability to experiment with balancing algorithms themselves, information and details regarding how this would be achieved shall be discussed.

Fully completed and presented, however, is the base language itself. This language provides all capabilities of a basic programming language, from variable declaration to functions to loops and much more. So complete, in fact, is that, aside from only currently supporting primitive types, this language provides much of the capability of commonly-used languages such as Lisp and Python.

Acknowledgements

Firstly, I would like to thank my supervisor Dr Neil Sculthorpe, who has supported me greatly with both personal and academic queries and issues. His critique and guidance of my work and interest within the domain of programming languages—academic and beyond—has vastly increased both the quality of my work and my interest in programming language theory.

A special thanks to my love, Katarzyna, who not only took on the task of completing my extra-curricular activities, allowing me to solely focus my attention on this project, but whos love and kindness infuses within me the ability to stay calm, to stay peaceful, to stay thoughtful.

To my Mother, Mandy, whose sheer determination and ability to endure has motivated me more than She will ever know. I love you, Mum.

Lastly, to the Earth and all of Her inhabitants, whose interconnectivity has and continues to allow me to be.

Table of Contents

[Introduction 1](#__RefHeading___Toc1485_386191692)

[1.11.1GENERAL PROBLEM OVERVIEW 1](#__RefHeading___Toc1923_1488950273)

[1.11.1TOPIC INTRODUCTION 2](#__RefHeading___Toc1925_1488950273)

[1.1.11.1.1What is a Binary Search Tree? 2](#__RefHeading___Toc1927_1488950273)

[1.1.21.1.2What is the difference between Balanced and Unbalanced BSTs? 3](#__RefHeading___Toc1929_1488950273)

[1.21.2PROJECT INTRODUCTION 3](#__RefHeading___Toc1931_1488950273)

[CONTEXT 6](#__RefHeading___Toc1487_386191692)

[2Introduction 6](#__RefHeading___Toc2564_1488950273)

[New Ideas 7](#__RefHeading___Toc1489_386191692)

[3Introduction 7](#__RefHeading___Toc2566_1488950273)

[IMPLEMENTATION 8](#__RefHeading___Toc1491_386191692)

[4Introduction 8](#__RefHeading___Toc2568_1488950273)

[RESULTS / DISCUSSION 10](#__RefHeading___Toc1493_386191692)

[5Introduction 10](#__RefHeading___Toc2570_1488950273)

[CONCLUSIONS / FUTURE WORK 11](#__RefHeading___Toc1495_386191692)

Insert > Reference > Index and Tables … > Table of Contents

List of FigureS

List of Tables

dwqddwqdwqdwq

CHAPTER 1

Introduction

## GENERAL PROBLEM OVERVIEW

A Data Structure is a specific way that data is stored and organized. The importance of choosing precisely what data structure to use varies greatly and is dependent on myriad variables: necessary capacity, frequency of both storing and searching, the need to be sortable, the stored data type, and many more.

One such data structure used frequently is a Binary Search Tree (BST). Due to its inherent capability to be performant, BSTs are frequently used for a multitude of domains, from compression algorithms—jpeg, mp3—to syntax trees—used by compilers for expression parsing.

There is, however, one primary problem with BSTs: the performance of insertion into or extraction from them can be greatly decreased if no balancing algorithm exists. A BST balancing specifies how the structure will be balanced, to ensure that it’s data storage is non-linear, for if the structure is unbalanced, it may store data linearly—its benefits virtually entirely nullified.

Many balancing algorithms exist, such as Red Black Tree and AVL, but the ability to implement and experiment with such algorithms is limited. The limitation primarily exists because it requires one to know the intricacies of both the BST implementation within the implementation language used and the intricacies of the implementation language itself. Only with such knowledge could one even begin to implement a balancing algorithm on a BST.

Parallel to this, there are few—if any—languages or frameworks that sufficiently benchmark balancing algorithms, which raises the aforementioned issue of experimentation, and disallows one to accurately benchmark self-designed algorithms against those frequently used within the industry, such as the aforementioned AVL or Red Black.

## TOPIC INTRODUCTION

### What is a Binary Search Tree?

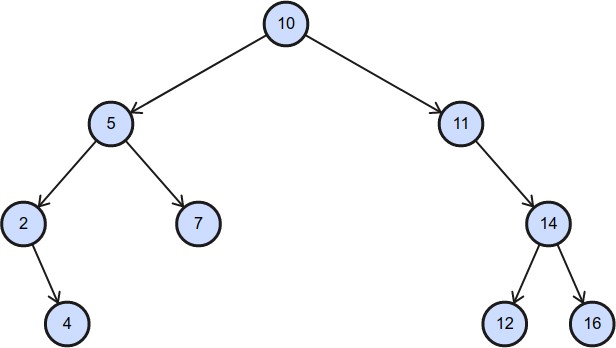
A Binary Seach Tree (BST) is method of organising data hierarchically, allowing its efficient manipulation in a reversed, tree-like structure, with the root sitting at the top of tree. BSTs are commonly used to implement other data structures, such as Hash Trees and Abstract Syntax Trees. It consists entirely of nodes, nodes that store a particularly type—either composite or primitive—and pointers to other nodes. There are essentially two types of nodes: an internal node and a leaf node. An internal node is any node which has pointers to other existing nodes

i.e. an internal node is the parent of its child nodes (the root node is the only internal node which has no parent); whereas a leaf node is any node whose pointers to its children are non- existent, and thus is not a parent.

BSTs typically provide three operations:

* **Search**: when given a value, traverses the BST until either:
  + the value is found; or
  + all nodes have been searched.
* **Insertion**: when given a value, inserts a node into the correct location.
* **Removal**: when given a particular value, searches the BST for the correct node; if found, the node is removed.

An example of a BST:



Within this example, the root node is 10, and its children 5 and 11. Node 11 has one child only, node 14, and nodes 4, 7, 12 and 16 have no children, and thus are leaves.

### What is the difference between Balanced and Unbalanced BSTs?

Given a BST with n nodes and a height h, a BST is balanced if and only if 2h-1 ≤ n ≤ 2h; ergo, the previous example is balanced. Balancing BSTs is essential for providing maximum performance, reducing the time complexity of its aforementioned operations. A balanced tree requires less time to performs its operations than an unbalanced tree—with the difference exponentially increasing with node count (an unbalanced tree takes O(log n) for its basic operations; an unbalanced tree, however, is O(n).

## PROJECT INTRODUCTION

This project aims to provide a solution to combat the difficulty of easily implementing and benchmarking BST balancing algorithms. One may ask: “Would not a simple framework for a currently existing language suffice?” To which the answer would be a resounding no. And for what reason the arrival at this answer is that, as aforementioned, this would still require one to understand the intricacies of language that the framework would be created for, which in and of itself causes an even greater problem: it restricts the experimentation with algorithms only to said persons who have knowledge of the implementation language.

Therefore, a fully operational Domain Specific Language (DSL) shall be presented here, which has all common features of basic programming language, such as loops, variable—both scoped and global—functions, conditional expressions, and more. It is hoped that the simplicity of the DSL provides the ease and capability to effectively allow one to implement balancing algorithms.

The language interpreter was originally written Haskell, due to its inherent capability to easily implementt DSLs. However, as this was a new technology learned solely for the purpose of this project—and exists as part of a programming paradigm that differs largely to what was, at that time, known—the interpreter was translated into Python, and thus Python was used to expand and finalise the DSL itself. However, the language choice for the DSL implementation matters very little for this particular project for one primary reason: the language implementation does not necessitate optimal performance.

The project outcome itself is not entirely dependent on the language developed; Instead, this report forms the foundation for the entire project. Sections existing within this report are:

* ***Introduction*:** including the section within which this text is written, the *Introduction* is a description of precisely what the project completion entails, including the problem domain, aims and objectives, and benefits of project outcome.
* ***Context:*** the context refers primarily to provide and explain the current State-of-the-Art work that currently exists within the field of the problem domain. This provides the elucidation of the gaps within the relevant field, and forms a foundation with which the problem solution is derived.
* ***New Ideas:*** As aforementioned, with the *Context* section detailing current State-of-the-Art—thus elucidating the gaps within the relevant field—this section extensively details the precise aspect that the outcome will focus on.
* ***Implementation or Investigation:*** This section largely discusses how the aims and objectives attempt to be achieved, methodologies used for software development, and the tools and resources used to ensure the correct, efficient implementation of the software.
* ***Results / Discussion:*** Detailed within this section is the oucome of the project itself—specifically, how the outcome relates to the proposed improvement within the field.
* ***Conclusions / Future Work:*** This section is a summary of the *Results / Discussion* section, including what has been developed relevant to the proposed aims, and what is uncompleted.

CHAPTER 2

CONTEXT

**Need to:**

* discuss current state of BST balancing algorithms: where they’re used and why they need experimentation
* is there a simply way to experiment with balancing algorithms, in code?
* Demonstrate that no DSL currently exists for balancing binary search trees.

**Also, format:**

* Author and Study
* Description of research

CHAPTER 3

New Ideas

# Introduction

As a result of your 'Limitations' chapter you should have narrowed down your area of research. This 'focussing' of attention on one aspect of the field will have been aided by reading about other peoples' work in the field. You may be proposing a development of one of their ideas or perhaps an idea that came to you that differs from anything tried before.

For a software development you might include an explicit list of the requirements, a description of investigation of requirements ( if appropriate), and a discussion of how requirements relate to Background research.

For a research-based investigation you might include the planning for the process (methodology) to be adopted, the criteria to be used for evaluation, and a discussion of reasons for this process and comparison with alternatives.

The proposed development or investigation must be realistic bearing in mind the entire project is supposed to take 400 hours of your time. Thus, evidence of project planning must be included in this chapter; estimates of work load for the various phases, setting these in context with other estimated workloads (e.g. course work and revision) and other deadlines. This should allow you to establish your project timetable (perhaps in the form of a Gantt chart) showing the interaction of these various factors and the set objectives/milestones. In your planning you should include contingency planning to allow for the unexpected disaster. Various project planning tools are covered in the course to allow you to do this.

CHAPTER 4

IMPLEMENTATION

# Introduction

Here you give details of the development or investigation of the new material proposed in 'New Ideas'. This must be done in a business-like manner. The development of any software must follow a suitable analysis and design methodology. There are CASE tools available to you for some methodologies, others will have to be a 'paper' design. An investigation must also follow a suitable methodology and use appropriate techniques and tools.

Software-based projects, requiring the production of a software solution for a set of requirements, should demonstrate that the software development has undergone appropriate analysis, design, project management, structured programming and testing. Research-based projects, requiring an investigation of a research question or client’s requirements, or being used to test a hypothesis, should demonstrate that the investigation has been properly conducted, is based on scientific principles and uses appropriate tools, techniques and standards. An investigation must produce a technical outcome from some development (software or hardware (e.g. networks, displays)) or testing (e.g. of system/network performance, system security, HCI/usability analysis). Sometimes a software prototype or a testing framework will be produced for the evaluation or testing of the research or hypothesis. Work based purely on literature review is not acceptable.

Some projects aim to provide software for general use as their final product and these must include relevant aspects of HCI (Human Computer Interaction) and address such features of usability such as 'user friendliness' and most likely employ GUI (graphical user interface) standards such as Windows.

In any case, students often ask what should go in this chapter, how to describe what they have done, what is relevant, how much of existing work to include, what to include from what they have done, etc. The simplest and surest way is to refer to your diary of the work you have done and report on it in chronological order.

The complete requirements analysis, problem analysis & design of software must be done rigorously and included in full in an appendix. Avoid cross-referencing it too often, thus causing the reader to keep flicking pages back and forth, rather reproduce sections that you wish to draw the reader's attention to. That is, highlight the parts that you found particularly difficult to implement and feel rather proud of having solved. Do not include lengthy descriptions of standard techniques or methodologies, simply state that 'such-and-such was designed using such-and-such technique (give a reference, not just 'SSADM' but 'SSADM [James 1996]' where the reference is a standard text on the technique!)' and highlight where you found shortcomings in the technique that didn't quite cope with your particular problem. Highlight exceptions to the standard.

CHAPTER 5

RESULTS / DISCUSSION

# Introduction

The technique developed as your project is supposed to show improvement on techniques previously available. Therefore it may be necessary to spend time investigating whether this is true. Perhaps you need to set up some sort of quantitative test and do a little statistical analysis to confirm the improvement. Perhaps a group of your friends could test out the user interface and provide comment on its suitability for the task. Try to estimate the limitations of your work and if it does not cover certain aspects that a user might expect then say so and make sure the system will reject input it is not expected to cope with.

CHAPTER 6

CONCLUSIONS / FUTURE WORK

Introduction

Whatever it was that your results showed should be summarised here. Hopefully the conclusion will be that your proposals proved to be brilliant and now the results bear this out. On the other hand your proposals may, in the light of the results obtained, prove to be less successful than you had hoped. In this case the conclusions should state why.

In either case there should be some reference to future work, either to forward and expand on the successful outcome or to test ways of overcoming the shortfall in your ideas that didn't work out quite as expected but there should be something that shows you can see further implications of what you have achieved.

This chapter should also include a discussion of the four PSEL issues (Professional, Social, Ethical and Legal) and the way in which you project has/will/could impact on each.

ReferenceS

Vogt, C. 1999. Creating Long Documents using Microsoft Word. Published on the Web at the Nottingham Trent University.

**Note:** References are a list that includes the essential bibliographical details for each item to which you have referred in the body of your paper. It should ONLY include items to which you have made direct reference. A direct reference is where you have quoted/reproduced text or diagrams from another author or mentioned/referred to the work of another author in your report. That is quoted directly what they have said about something or mentioned their views or conclusions in your report. For details of citation and references see the information in the Project Guide.

A Bibliography is a list of published materials that you have read or consulted for general information in the preparation of your work, concerning the subject of your Project, but have not made any direct reference to in your report i.e. 'background reading'.

You should always provide a Reference List. **A Bibliography is optional but when provided it should include all items in your Reference List as well as any additional items consulted in preparation of your work.**

Bibliography

Vogt, C. 1999. Creating Long Documents using Microsoft Word. Published on the Web at the Nottingham Trent University.

Coote, H., Dobbs, B. & Jones, C. (1996). Defining databases. Wiley: Melbourne.

Applications and Science in Soft Computing, Lotfi, Ahmad; Garibaldi, Jonathon M. (Eds.) 2004, X, 346 p. Springer, ISBN: 3-540-40856-8

**Note:** References are a list that includes the essential bibliographical details for each item to which you have referred in the body of your paper. It should ONLY include items to which you have made direct reference. A direct reference is where you have quoted/reproduced text or diagrams from another author or mentioned/referred to the work of another author in your report. That is quoted directly what they have said about something or mentioned their views or conclusions in your report. For details of citation and references see the information in the Project Guide.

A Bibliography is a list of published materials that you have read or consulted for general information in the preparation of your work, concerning the subject of your Project, but have not made any direct reference to in your report i.e. 'background reading'.

You should always provide a Reference List. **A Bibliography is optional but when provided it should include all items in your Reference List as well as any additional items consulted in preparation of your work.**

Appendix A

The content of these will differ with the different types of project. Any design and analysis charts/diagrams will be included here in full. In projects where software has been developed there will be an appendix for this. Our departmental requirement is that a CD, DVD or USB memory stick of all source code is submitted to your project supervisor. The appendix contained in the report will refer to this CD, DVD, or USB memory stick, provide a directory style listing of the files submitted and instructions for rebuilding and running the software. This might be source code of programs written in high level languages (C, C++, etc) together with any pertinent files ('make' files, non-standard libraries, etc). Alternatively, or in addition, you can place some or all of the source code in the appendix. In any case the source code needed to reconstruct any software you have developed must be submitted in its entirety in the CD, DVD, or USB memory stick. (Any code that has been used from a third party should reference the original developer).

Hardware designs will require schematics/circuit diagrams, PCB layouts, simulation tests and pin outs.

Most projects will require some form of user documentation to explain how to use the software/hardware produced. A researcher following up the work may wish to utilise the work of the original author and an appendix laying out the format of input files and how to interpret the output is required.